## INITIATION AU TRAITEMENT D'IMAGES Contours, Couleurs, Mouvements

### Franck Luthon

Franck. Luthon @univ-pau. fr

Laboratoire d'Informatique de l'Université de Pau et des Pays de l'Adour EA3000 Dpt G.I.M., IUT 2 Allée du Parc de Montaury, 64600 Anglet, France

 $http://www.iutbayonne.univ-pau.fr/~luthon/img0.pdf^1$ 

21ème édition

année 2024-2025

1. Ce document peut être reproduit sous réserve d'en citer la source.

# Table des matières

1	Tra	Fraitement d'image							
	1.1	Introd	uction	9					
	1.2	Rappe	ls de théorie de l'information	10					
		1.2.1	Entropie d'une source discrète	10					
		1.2.2	Redondance d'une source d'information	11					
		1.2.3	Codage entropique	11					
	1.3	Histog	ramme	12					
	1.4	Seuilla	ge	14					
	1.5	Contraste							
		1.5.1	Contraste de Michelson	14					
		1.5.2	Contraste de Weber	14					
		1.5.3	Contraste de Gordon	14					
		1.5.4	Contraste de Beghdadi	14					
		1.5.5	Contraste de Peli	14					
		1.5.6	Contraste de Köhler	15					
	1.6	Filtrag	e linéaire	15					
		1.6.1	Convolution spatiale	15					
		1.6.2	Transformée de Fourier discrète	15					
		1.6.3	Transformée en Z	18					
		1.6.4	Typologie des filtres	18					
		1.6.5	Fonction de transfert	20					
	1.7	Détect	ion de contours	20					
		1.7.1	Gradient et laplacien	20					
		1.7.2	Détecteur de Roberts	21					
		1.7.3	Détecteur de Prewitt	22					
		1.7.4	Détecteur de Sobel	22					
		1.7.5	Détecteur de Canny-Deriche	22					
		1.7.6	Détecteur de Shen-Castan	22					
		1.7.7	Algorithme de Rosenfeld & Kak	22					
	1.8	Morph	ologie mathématique	25					
		1.8.1	Introduction	25					
		1.8.2	Opérations binaires (images N&B)	25					
		1.8.3	Propriétés	26					
		1.8.4	Autres opérations	27					
		1.8.5	Opérations sur images en NdG	28					
	1.9	Segme	ntation couleur	29					
		1.9.1	Vision des couleurs	29					
		1.9.2	Trichromie	30					
		1.9.3	Segmentation	32					

<b>2</b>	Trai	itement de séquences d'images 37	7
_	2.1	Détection de mouvement	7
		2 1 1 Introduction 33	7
		2.1.2 Principe	7
	$\mathcal{D}\mathcal{D}$	Régularisation statistique markovienne	<u>ה</u>
	2.2	2.2.1 Fonctions d'énergie	, ג
		2.2.1 Follemention des paramètres	, I
		2.2.2 Estimation des parametres	L 1
		2.2.5 Algorithmes de l'elaxation	ן ר
	<u></u>	2.2.4 Synoptique a un algorithme de detection de mouvement	2 2
	2.3	Wultiresolution spatio-temporene	2 2
	2.4	Volsinage 3D spatio-temporel	5
	2.5	Mises en œuvre materielles	)
		2.5.1 Adequation algorithme-architecture	)
		2.5.2 Solutions envisageables	5
		2.5.3 Circuit VLSI analogique	3
	2.6	Exemples d'applications	3
		$2.6.1  \text{Télésurveillance}  \dots  \dots  \dots  \dots  \dots  \dots  \dots  \dots  \dots  $	3
		2.6.2 Analyse du mouvement des lèvres d'un locuteur	3
		2.6.3 Remarque conclusive	)
	2.7	Estimation de mouvement	L
		2.7.1 Méthodes différentielles	L
		2.7.2 Méthodes fréquentielles	3
		2.7.3 Mise en correspondance de blocs	1
		2.7.4 Modèles paramétriques de mouvement	1
	2.8	Compensation de mouvement	5
		2.8.1 Introduction	5
		2.8.2 Estimation de mouvement pel-récursive	3
		2.8.3 Principe du codage vidéo	7
	29	Méthodes hybrides de compression vidéo	2
	2.0	291 Transformée en ondelettes	2
		2.9.2 Ondelettes et compensation de mouvement	) )
		2.9.2 Cohérence du mouvement entre sous bandes?	י ג
		2.9.5 Conference du mouvement entre sous-bandes :	י ו
		2.9.4 Autres approches	2
3	Nor	rmes et standards du multimédia	3
Č	3.1	Introduction 65	, X
	3.2	Manipulation d'objets multimédia	, Z
	0.2	3.2.1 Saisie et numérisation	, z
		3.2.2 Codage at compression	5
		3.2.2 Obtage et compression	, ;
		2.2.4 Transmission	, :
		$\begin{array}{c} 3.2.4  \text{ITARSHISSION} \\ 2.2.5  \text{Destitution} \\ \end{array}$	) 2
		$3.2.5  \text{Restitution}  \dots  \dots  \dots  \dots  \dots  \dots  \dots  \dots  \dots  $	) 2
	<u></u>	3.2.6 Representation $3D$	)
	3.3		( 
		3.3.1 Format BMP $\dots$	(
		3.3.2 GIF et PNG	(
		$3.3.3  \text{TIFF} \qquad \dots \qquad $	(
		3.3.4 Formats issus de la norme JPEG	(
		3.3.5 FlashPix	3
		3.3.6 Formats liés à la photo numérique	3
	3.4	Codage des applications multimédia 68	3
		3.4.1 Catégories d'applications	3
		3.4.2 Normes de codage des applications	)
		3.4.3 HTML	)

	3.5	3.4.4       XML       XML       XML         Compression d'image       X       X       X         3.5.1       JPEG : emploi de la transformée DCT       X       X	70 70 70
		3.5.2 JPEG2000 : emploi de la transformée en ondelettes	71
		3.5.3 Emploi des fractales	72
	3.6	Compression de séquences vidéo	72
		3.6.1 Principe du codage hybride	72
		3.6.2 Normes MPEG-4 et H.264	74
		3.6.3 MPEG-5 et H.265	74
4	Exe	rcices 7	<b>′</b> 5
	4.1	Débit d'information	75
	4.2	Modification d'histogramme	75
	4.3	Principe de l'égalisation d'histogramme	75
	4.4	Egalisation et étalement	76
	4.5	Détection de contours	77
	4.6	Calcul de laplacien	77
	4.7	Formules de dérivation numérique	78
	4.8	Filtrage linéaire	78
	4.9	Effet de moiré par repliement de spectre	78
	4.10	Transformée de Fourier	78
	4.11	Morphologie mathématique	79
	4.12	Poursuite de contour binaire « -2+4 »	79
	4.13	Ouverture et fermeture	79
	4.14	Lissage morphologique	30
	4.15	Squelettisation morphologique	30
	4 16	Zone aveugle	32
	4 17	TV couleur Secam	32
	4 18	ACP couleur	23
	<i>1</i> .10	Détection de mouvement	23
	4.15	Etiquetare statistique contextuel du menurement	22
	4.20	Equation de contrainte du mouvement	25
	4.21	Equation de contrainte du mouvement	25
	4.22	Algorithme de Herre le Colombia	50 57
	4.23		50
	4.24	Estimation par mise en correspondance de bloc	55
	4.25	Estimation d'un modele de mouvement	50
	4.26	Estimateur robuste	30
	4.27	Filtre de Canny-Deriche	57
	4.28	Transformée couleur logarithmique	37
	4.29	Filtrage linéaire	37
	4.30	Transformée couleur non-linéaire	38
	4.31	Compensation de mouvement	38
	4.32	Teinte du visage	39
	4.33	Spectres d'images	39
	4.34	Application industrielle	<b>)</b> 1
	4.35	Résolution et contraste	<del>)</del> 1
	4.36	Filtrage médian	<del>)</del> 2
	4.37	Filtrage linéaire	<del>)</del> 3
	4.38	Codage de contour	<del>)</del> 3
	4.39	Filtrage linéaire	<del>)</del> 3
	4.40	Remplissage morphologique	<b>)</b> 4
	4.41	Analyse de mouvement	95
	4.42	Filtre médian	<u>)</u> 5
	4.43	Codage de contour	<u>)</u> 6
		J	

	4.44	Filtrage linéaire : filtre de Sobel
	4.45	Gradient morphologique
	4.46	Codage entropique de Huffman
	4.47	Effet 2D d'un filtre RIF 1D
	4.48	Filtrage non-linéaire
	4.49	Etiquetage en composantes connexes
		4.49.1 Travail demandé
		4.49.2 Algorithme
	4.50	Programmation OpenCV
	4 51	Détection de contour
	4 52	Morphologie mathématique
	4.52	Codage entropique
	4.55	Opérations morphologiques 10
	4.04	Membalacia mathématicus
	4.55	
	4.50	
	4.57	Estimation de mouvement
	4.58	Résolution d'une caméra linéaire 10
	4.59	Aberration chromatique d'un capteur
	4.60	Seuillage d'image
	4.61	Codage de chaîne
	4.62	Binarisation par seuillage entropique
	4.63	Spectre de Fourier
	4.64	API JMF-RTP 11
	4.65	Phénomène d'aliasing
	4.66	QCM sommatif
	4.67	QCM fiche de lecture (1 ou plusieurs bonnes réponses)
	1.01	
	4.68	Annexe numérique
_	4.68	Annexe numérique
5	4.68 Trav	Annexe numérique
5	4.68 Trav 5.1	Annexe numérique   11     vaux Pratiques   11     Filtres numériques RII   11
5	4.68 Trav 5.1	Annexe numérique       1         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11
5	4.68 Trav 5.1	Annexe numérique       1         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11
5	4.68 Trav 5.1	Annexe numérique       1         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12
5	4.68 Trav 5.1	Annexe numérique       1         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12
5	4.68 Trav 5.1	Annexe numérique       11         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12         Estimateur de mouvement       12
5	4.68 <b>Trav</b> 5.1	Annexe numérique       1         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.2.1       Algorithme de Horn et Schunck       12
5	4.68 <b>Trav</b> 5.1	Annexe numérique       1         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.2.1       Algorithme de Horn et Schunck       12         5.2.2       Implantation en C de l'estimateur de vitesse       12
5	<ul> <li>4.68</li> <li>Trav</li> <li>5.1</li> <li>5.2</li> <li>5.3</li> </ul>	Annexe numérique       11         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.2.1       Algorithme de Horn et Schunck       12         5.2.2       Implantation en C de l'estimateur de vitesse       12         Correction d'histogramme       12
5	4.68 <b>Trav</b> 5.1 5.2 5.3 5.4	Annexe numérique       1         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.2.1       Algorithme de Horn et Schunck       12         5.2.2       Implantation en C de l'estimateur de vitesse       12         Transformation de Fourier rapide       12
5	4.68 <b>Trav</b> 5.1 5.2 5.3 5.4 5.5	Annexe numérique       1         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12         Estimateur de mouvement       12         5.2.1       Algorithme de Horn et Schunck       12         5.2.2       Implantation en C de l'estimateur de vitesse       12         Correction d'histogramme       12         Transformation de Fourier rapide       12         Filtre de Shen-Castan : implantation Java       12
5	<ul> <li>4.68</li> <li>Trav</li> <li>5.1</li> <li>5.2</li> <li>5.3</li> <li>5.4</li> <li>5.5</li> </ul>	Annexe numérique11vaux Pratiques11Filtres numériques RII115.1.1Filtre de lissage : opérateur de flou115.1.2Implantation en C du flou réglable115.1.3Filtre de dérivation : détecteur de contours125.1.4Implantation en C du détecteur de contour réglable125.2.1Algorithme de Horn et Schunck125.2.2Implantation en C de l'estimateur de vitesse12Correction d'histogramme12Transformation de Fourier rapide12Filtre de Shen-Castan : implantation Java125.1Présentation125.1Présentation12
5	<ul> <li>4.68</li> <li>Trav</li> <li>5.1</li> <li>5.2</li> <li>5.3</li> <li>5.4</li> <li>5.5</li> </ul>	Annexe numérique       11         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.2.1       Algorithme de Horn et Schunck       12         5.2.2       Implantation en C de l'estimateur de vitesse       12         Correction d'histogramme       12         Transformation de Fourier rapide       12         Filtre de Shen-Castan : implantation Java       12         5.5.1       Présentation       12         5.5.2       Travail demandé       12
5	4.68 <b>Trav</b> 5.1 5.2 5.3 5.4 5.5	Annexe numérique       11         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.2.1       Algorithme de Horn et Schunck       12         5.2.2       Implantation en C de l'estimateur de vitesse       12         Correction d'histogramme       12         Transformation de Fourier rapide       12         Filtre de Shen-Castan : implantation Java       12         5.5.2       Travail demandé       12         5.5.3       Programme exemple       12
5	4.68 <b>Trav</b> 5.1 5.2 5.3 5.4 5.5	Annexe numérique       11         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.2.1       Algorithme de Horn et Schunck       12         5.2.2       Implantation en C de l'estimateur de vitesse       12         Correction d'histogramme       12         Transformation de Fourier rapide       12         Filtre de Shen-Castan : implantation Java       12         5.5.1       Présentation       12         5.5.2       Travail demandé       12         5.5.3       Programme exemple       12         0penCV : implantation Java dans Eclipse       12
5	<ul> <li>4.68</li> <li>Trav</li> <li>5.1</li> <li>5.2</li> <li>5.3</li> <li>5.4</li> <li>5.5</li> <li>5.6</li> </ul>	Annexe numérique       11         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.2.1       Algorithme de Horn et Schunck       12         5.2.2       Implantation en C de l'estimateur de vitesse       12         Correction d'histogramme       12         Transformation de Fourier rapide       12         5.1.1       Présentation       12         5.2.2       Travail demandé       12         5.3       Programme exemple       12         5.5.3       Programme exemple       12         5.5.1       Présentation       12         5.5.3       Programme exemple       12         5.5.4       Présentation       12         5.5.3       Programme exemple       12         5.5.4       Présentation       12         5.5.4       Présentation       12         5.5.4       Présentation       12
5	<ul> <li>4.68</li> <li>Trav</li> <li>5.1</li> <li>5.2</li> <li>5.3</li> <li>5.4</li> <li>5.5</li> <li>5.6</li> </ul>	Annexe numérique       11         vaux Pratiques       11         Filtres numériques RII       11         5.1.1 Filtre de lissage : opérateur de flou       11         5.1.2 Implantation en C du flou réglable       11         5.1.3 Filtre de dérivation : détecteur de contours       12         5.1.4 Implantation en C du détecteur de contour réglable       12         5.1.4 Implantation en C du détecteur de contour réglable       12         5.2.1 Algorithme de Horn et Schunck       12         5.2.2 Implantation en C de l'estimateur de vitesse       12         5.2.2 Implantation en C de l'estimateur de vitesse       12         5.2.1 Algorithme de Horn et Schunck       12         5.2.2 Implantation en C de l'estimateur de vitesse       12         Correction d'histogramme       12         Transformation de Fourier rapide       12         5.5.1 Présentation       12         5.5.2 Travail demandé       12         5.5.3 Programme exemple       12         0penCV : implantation Java dans Eclipse       12         5.6.1 Présentation       12         5.6.2 Installation et test de JavaCV       12
5	<ul> <li>4.68</li> <li>Trav</li> <li>5.1</li> <li>5.2</li> <li>5.3</li> <li>5.4</li> <li>5.5</li> <li>5.6</li> </ul>	Annexe numérique       11         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.2.1       Algorithme de Horn et Schunck       12         5.2.2       Implantation en C de l'estimateur de vitesse       12         Correction d'histogramme       12         Transformation de Fourier rapide       12         5.5.1       Présentation       12         5.5.2       Travail demandé       12         5.5.3       Programme exemple       12         5.5.4       Invaliation Java dans Eclipse       12         5.6.1       Présentation       12         5.6.2       Installation et test de JavaCV       12         5.6.3       Programmation d'un traitement d'image au choix       12
5	<ul> <li>4.68</li> <li>Trav</li> <li>5.1</li> <li>5.2</li> <li>5.3</li> <li>5.4</li> <li>5.5</li> <li>5.6</li> </ul>	Annexe numérique       11         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.2.1       Algorithme de Horn et Schunck       12         5.2.2       Implantation en C de l'estimateur de vitesse       12         5.2.2       Implantation en C de l'estimateur de vitesse       12         Correction d'histogramme       12         Transformation de Fourier rapide       12         5.5.1       Présentation       12         5.5.2       Travail demandé       12         5.5.3       Programme exemple       12         5.5.4       Présentation       12         5.6.1       Présentation       12         5.6.2       Installation et test de JavaCV       12         5.6.3       Programmation d'un traitement d'image au choix       12         5.6.4       Listing du programme exemple
5	<ul> <li>4.68</li> <li>Trav</li> <li>5.1</li> <li>5.2</li> <li>5.3</li> <li>5.4</li> <li>5.5</li> <li>5.6</li> </ul>	Annexe numérique       11         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.2.1       Algorithme de Horn et Schunck       12         5.2.2       Implantation en C de l'estimateur de vitesse       12         Correction d'histogramme       12         Transformation de Fourier rapide       12         Filtre de Shen-Castan : implantation Java       12         5.5.1       Présentation       12         5.5.2       Travail demandé       12         5.5.3       Programme exemple       12         2.5.4       Installation Java dans Eclipse       12         2.6.1       Présentation       12         2.6.2       Installation et test de JavaCV       12         5.6.3       Programme exemple       12 </td
5	<ul> <li>4.68</li> <li>Trav</li> <li>5.1</li> <li>5.2</li> <li>5.3</li> <li>5.4</li> <li>5.5</li> <li>5.6</li> </ul>	Annexe numérique       11         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.1.4       Implantation en C du détecteur de contour réglable       12         5.2.1       Algorithme de Horn et Schunck       12         5.2.2       Implantation en C de l'estimateur de vitesse       12         5.2.1       Algorithme de Horn et Schunck       12         5.2.2       Implantation en C de l'estimateur de vitesse       12         Correction d'histogramme       12         Transformation de Fourier rapide       12         Filtre de Shen-Castan : implantation Java       12         5.5.1       Présentation       12         5.5.3       Programme exemple       12         5.6.1       Présentation       12         5.6.2       Installation Java dans Eclipse       12         5.6.3       Programmation d'un traitement d'image au choix       12         5.6.4       Listing du pr
5	<ul> <li>4.68</li> <li>Trav</li> <li>5.1</li> <li>5.2</li> <li>5.3</li> <li>5.4</li> <li>5.5</li> <li>5.6</li> <li>5.7</li> </ul>	Annexe numérique       11         vaux Pratiques       11         Filtres numériques RII       11         5.1.1 Filtre de lissage : opérateur de flou       11         5.1.2 Implantation en C du flou réglable       11         5.1.3 Filtre de dérivation : détecteur de contours       12         5.1.4 Implantation en C du détecteur de contour réglable       12         5.1.4 Implantation en C du détecteur de contour réglable       12         5.2.1 Algorithme de Horn et Schunck       12         5.2.2 Implantation en C de l'estimateur de vitesse       12         Correction d'histogramme       12         Transformation de Fourier rapide       12         5.5.1 Présentation       12         5.5.2 Travail demandé       12         5.5.3 Programme exemple       12         5.6.1 Présentation       12         5.6.2 Installation et test de JavaCV       12         5.6.3 Programmation d'un traitement d'image au choix       12         5.6.4 Listing du programme exemple       12         5.6.5 Structure d'image       12         Détecteur de mouvement       12         Détecteur de mouvement       13
5	4.68 <b>Trav</b> 5.1 5.2 5.3 5.4 5.5 5.6 5.6	Annexe numérique       11         vaux Pratiques       11         Filtres numériques RII       11         5.1.1 Filtre de lissage : opérateur de flou       11         5.1.2 Implantation en C du flou réglable       11         5.1.3 Filtre de dérivation : détecteur de contours       12         5.1.4 Implantation en C du détecteur de contour réglable       12         5.1.4 Implantation en C du détecteur de contour réglable       12         5.2.1 Algorithme de Horn et Schunck       12         5.2.2 Implantation en C de l'estimateur de vitesse       12         Correction d'histogramme       12         Transformation de Fourier rapide       12         Transformation de Fourier rapide       12         5.5.1 Présentation       12         5.5.2 Travail demandé       12         5.5.3 Programme exemple       12         5.6.1 Présentation       12         5.6.2 Installation et test de JavaCV       12         5.6.3 Programmation d'un traitement d'image au choix       12         5.6.4 Listing du programme exemple       12         5.6.5 Structure d'image       12         Détecteur de mouvement       13         5.7.1 Présentation       13
5	4.68 <b>Trav</b> 5.1 5.2 5.3 5.4 5.5 5.6 5.7	Annexe numérique       11         vaux Pratiques       11         Filtres numériques RII       11         5.1.1 Filtre de lissage : opérateur de flou       11         5.1.2 Implantation en C du flou réglable       11         5.1.3 Filtre de dérivation : détecteur de contours       12         5.1.4 Implantation en C du détecteur de contour réglable       12         5.1.4 Implantation en C du détecteur de contour réglable       12         5.2.1 Algorithme de Horn et Schunck       12         5.2.2 Implantation en C de l'estimateur de vitesse       12         Correction d'histogramme       12         Transformation de Fourier rapide       12         Filtre de Shen-Castan : implantation Java       12         5.5.1 Présentation       12         5.5.2 Travail demandé       12         5.5.3 Programme exemple       12         5.6.1 Présentation       12         5.6.2 Installation et test de JavaCV       12         5.6.3 Programmation d'un traitement d'image au choix       12         5.6.4 Listing du programme exemple       12         5.6.5 Structure d'image       12         Détecteur de mouvement       13         5.7.1 Présentation       13         5.7.2 Programmation en Java       13
5	4.68 <b>Trav</b> 5.1 5.2 5.3 5.4 5.5 5.6 5.7	Annexe numérique       11         vaux Pratiques       11         Filtres numériques RII       11         5.1.1       Filtre de lissage : opérateur de flou       11         5.1.2       Implantation en C du flou réglable       11         5.1.3       Filtre de dérivation : détecteur de contours       12         5.1.4       Implantation en C du détecteur de contour réglable       12         Estimateur de mouvement       12         5.2.1       Algorithme de Horn et Schunck       12         5.2.2       Implantation en C de l'estimateur de vitesse       12         Correction d'histogramme       12         Transformation de Fourier rapide       12         Filtre de Shen-Castan : implantation Java       12         5.5.1       Présentation       12         5.5.2       Travail demandé       12         5.5.3       Programme exemple       12         5.6.1       Présentation       12         5.6.2       Installation et test de JavaCV       12         5.6.3       Programmation d'un traitement d'image au choix       12         5.6.4       Listing du programme exemple       12         5.6.5       Structure d'image       12         5.6.6

	5.8.1	Objectifs	132			
	5.8.2	Cahier des charges	132			
	5.8.3	Consignes de travail	133			
	5.8.4	Liste de traitements proposée	133			
	5.8.5	Algorithme de rehaussement de détails	133			
5.9	Initiati	on au traitement d'image avec Matlab	134			
	5.9.1	Introduction	134			
	5.9.2	Programmes démonstratifs	134			
	5.9.3	Lecture-écriture, affichage, manipulation d'image	134			
	5.9.4	Modification d'histogramme et seuillage	135			
	5.9.5	Filtrage linéaire : contours, gradient, laplacien, lissage	135			
	5.9.6	Transformée de Fourier et fonction de transfert	135			
5.10	Traiter	nent d'image statique avec Matlab	135			
	5.10.1	Morphologie mathématique	135			
	5.10.2	Détection de teinte chair	135			
5.11	Simula	tion des exercices avec Matlab	136			
	5.11.1	Filtrage linéaire	136			
	5.11.2	Visualisation scientifique : tracé de courbes	136			
Références bibliographiques						

## Chapitre 1

## Traitement d'image

## Préambule

Ce premier chapitre traite de l'image numérique individuelle : on y aborde les notions essentielles de quantité d'information, de contraste, de spectre, de contour, de forme et de couleur ; et l'on présente les outils mathématiques associés : codage statistique, transformations intégrales, filtrages, morphologie mathématique, segmentation et classification. On en profite pour exposer aussi les principales propriétés du système visuel humain reposant sur la trichromie.

Le second chapitre portera sur le traitement de séquences d'images numériques (vidéos) introduisant la notion de mouvement. Enfin le troisième chapitre présente les formats et les standards de codage des images et vidéos, développés notamment pour la compression.

## 1.1 Introduction

La vision est la **perception** du monde extérieur [19, 56]. L'image est la **représentation** bidimensionnelle d'une scène 3D (à trois dimensions). C'est l'information issue d'un capteur de vision (caméra, œil). Le traitement d'image [81, 105, 119] comporte donc différentes tâches : capture, analyse et traitement (bas niveau), interprétation et décision (haut niveau). On distingue différents traitements :

- amélioration, restauration et correction d'image : augmentation du contraste, correction des distorsions optiques, filtrage du bruit,
- analyse [39, 25] : détection et localisation d'objets, segmentation, reconnaissance de formes, estimation et mesures,
- codage pour la compression, l'encryptage, l'archivage et la transmission numérique [66].

Le traitement d'image couvre plusieurs domaines [11, 20, 4]:

- l'électronique (capteur, optique, acquisition),
- le traitement de signal donc la mathématique (détection, segmentation, estimation, décision)
   [15],
- l'informatique [13, 14, 27] et l'intelligence artificielle [43] (reconnaissance, interprétation, guidage robotique).

On ne s'intéresse ici qu'aux **images numériques**, c'est-à-dire échantillonnées et quantifiées. Une image numérique est définie par un tableau de pixels de taille  $L \times C$  (typ. 256 × 256). Chaque pixel, portant l'information d'intensité lumineuse perçue en ce point, est codé sur n bits. Typ. n = 8 pour une image à 256 niveaux de gris (NdG) ou n = 24 pour une image couleur RVB, avec 8 bits par composante couleur.

On définit dans l'image un repère Oxy et une notion de distance et de voisinage (Fig. 1.1).



FIGURE 1.1 – a) Repère relatif à l'image; b) 8-voisinage d'un pixel.

#### 1.2 Rappels de théorie de l'information

#### 1.2.1 Entropie d'une source discrète

L'entropie H (exprimée en bits) d'une source discrète d'observations à valeurs dans  $\{0...M - 1\}$  (typ. M = 256) est définie par [120] :

$$H_{bit} = -\sum_{i=0}^{M-1} p_i \log_2 p_i \tag{1.1}$$

où  $p_i$  représente la probabilité pour que l'observation en le site  $s = (x, y) \in S$  vaille :  $o_s = i$  (S étant le support des observations, en l'occurrence ici une image de taille  $L \times C$ ).

Rappelons que le choix de la base du logarithme est arbitraire et correspond au choix de l'unité de mesure (le bit si l'on choisit le logarithme binaire, le nat si l'on prend le logarithme népérien); et l'on a :

$$H_{nat} = H_{bit} \cdot \log_e 2 \approx 0.7 H_{bit} \tag{1.2}$$

En effet, le passage d'une base a à une base b requiert simplement une multiplication par la constante  $K = \log_b a$ .

Comme l'information élémentaire apportée par un événement de probabilité  $p_i$  vaut par définition <sup>1</sup> :  $I_i = \log \frac{1}{p_i}$ , on voit que l'entropie est une mesure de l'**information moyenne** débitée par une source :  $H = \sum_i p_i I_i$ .

Shannon a démontré que l'entropie (exprimée en nats) d'une source gaussienne centrée d'écart-type  $\sigma$  vaut  $H_{nat} = \log(\sigma \cdot \sqrt{2\pi e})$  et qu'à  $\sigma$  fixé, la source donnant l'entropie maximale est la distribution gaussienne.

Pour une source que lconque d'entropie H donnée, Shannon définit la notion de « puissance entropique »  ${\cal P}_e$  :

$$P_e = \frac{1}{2\pi e} \exp(2H) \tag{1.3}$$

C'est la puissance de la distribution gaussienne équivalente à la distribution d'entropie H. Pour un bruit gaussien centré d'écart-type  $\sigma$ , on retrouve bien sûr :  $P_e = \sigma^2$ . Comme le bruit blanc gaussien possède l'entropie maximale à puissance fixée, la puissance entropique d'un bruit quelconque est toujours inférieure ou égale à sa puissance effective. Toujours selon Shannon, un bruit blanc gaussien a la propriété d'absorber tout autre signal qui lui est ajouté. La puissance entropique résultante est à peu près égale à la somme de la puissance du bruit blanc et de la puissance du signal (supposé centré), à condition que la puissance du signal soit faible, « dans un certain sens », comparée au bruit [120].

<sup>1.</sup> Que l'information varie à l'inverse de la probabilité relève du sens commun : un message très probable, tel que « Demain, le soleil se lèvera », apporte très peu d'information. Au contraire, un message très peu probable, comme « Le soleil ne se lèvera pas demain » apporte beaucoup d'information (s'il est vrai!). C'est la notion même des « nouvelles », au sens journalistique (cf. le *scoop* du journaliste).

En supposant le bruit additif gaussien et majoritaire sur le support, on peut donc estimer un écart-type entropique  $\sigma_e$  équivalent en calculant l'entropie H des observations [97] :

$$\sigma_e = \sqrt{P_e} = \frac{\exp(H)}{\sqrt{2\pi e}} = \frac{2^{H_{bit}}}{\sqrt{2\pi e}}$$
(1.4)

En prenant une mesure de seuil  $\theta$  estimée par :

$$\theta = 4\sigma_e \approx 2^{H_{bit}} \tag{1.5}$$

on obtient une technique automatique de seuillage adaptatif, applicable entre autres à la détection du mouvement dans les séquences d'images [96, 98] **§4.62**.

#### 1.2.2 Redondance d'une source d'information

L'entropie est une notion fondamentale car elle renseigne sur la redondance spatiale présente dans une image et permet de mettre en œuvre les outils de codage définis par Shannon pour réaliser une compression sans perte de l'information contenue dans l'image (économie de débit pour la transmission : codage de Huffman, de Shannon-Fano). La redondance d'une source X est définie par :

$$R = 1 - \frac{H(X)}{H_{max}} \tag{1.6}$$

Pour une source discrète à M événements, l'entropie maximale est atteinte quand les événements sont équiprobables et elle vaut :

$$H_{max} = \log_2 M. \tag{1.7}$$

#### 1.2.3 Codage entropique

Les méthodes de codage entropique visent à coder les messages les plus probables avec les mots-codes les plus courts : on obtient un code à longueur variable, en anglais VLC. Elles permettent d'assurer la propriété du préfixe (obtention d'un code « irréductible »), et de faire en sorte que  $p(0) \approx p(1)$ , pour réduire la redondance et atteindre un taux de compression optimal sans perte.

#### Méthode de Shannon-Fano

La procédure est la suivante **§4.53** :

- 1. Classer dans un tableau les messages (ou événements) par ordre de probabilité non croissante.
- 2. Découper l'ensemble X des messages en 2 sous-ensembles  $X_1$  et  $X_2$  ayant des probabilités voisines (aussi proches que possible de l'équiprobabilité).
- 3. Attribuer à l'un des sous-ensembles le préfixe 0 et à l'autre 1 (comme première lettre du motcode).
- 4. Réitérer la séquence d'opérations 2 & 3 sur  $X_1$  puis sur  $X_2$  séparément, etc... jusqu'à avoir isolé chaque message individuellement.
- 5. Le code obtenu se lit directement dans le tableau, simplement de gauche à droite.

#### Méthode de Huffman

La procédure est la suivante **§4.46** :

- 1. On classe les événements dans l'ordre des probabilités décroissantes (idem Shannon-Fano).
- 2. A chaque étape, on attribue aux 2 derniers messages  $m_{M-1}$  et  $m_M$  (donc les 2 moins probables) un symbole 0 et un symbole 1 respectivement.
- 3. On construit à partir d'eux un super-message (texte  $m_{M-1}m_M$ ) de probabilité somme  $p_{M-1}+p_M$  que l'on reclasse dans un nouveau tableau en l'intercalant pour respecter toujours l'ordre de probabilités décroissantes.
- 4. On continue jusqu'à ce qu'il ne reste que 2 messages isolés.
- 5. On revient en arrière dans le tableau pour le codage (lecture du code de droite à gauche).

## 1.3 Histogramme

L'histogramme est la courbe n(i) où n est le nombre de pixels d'intensité i. Il peut comporter plusieurs pics ou modes (Fig. 1.2), ou bien être uniforme.



FIGURE 1.2 – Exemple d'histogramme comportant deux modes, avec un seuil  $\theta$  positionné en un minimum local.

Si l'on note  $N = L \times C$  le nombre total de pixels de l'image, et M le nombre de niveaux de gris possibles  $(i = 0 \cdots M - 1)$ , la probabilité d'un niveau de gris i vaut  $p_i = \frac{n_i}{N}$ , et la fonction de densité de probabilité cumulée (CPDF) s'exprime :

$$F(i) = \sum_{k=0}^{i} p_k \quad \text{avec } F(0) = p_0 \quad \text{et } F(M-1) = 1.$$
(1.8)

Les deux principales corrections d'histogramme sont l'égalisation et l'étalement.

L'égalisation consiste à redistribuer les niveaux de gris en remplaçant le niveau i par le niveau  $l_i$  tel que §4.3 :

$$l_i = (M-1)F(i) = \frac{M-1}{L \times C} \sum_{k=0}^{i} n_k$$
(1.9)

La Fig. 1.3 illustre l'effet typique d'une égalisation d'histogramme, réalisée grâce au logiciel libre éditeur d'images GIMP (GNU Image Manipulation Programme) sur une photo prise en sous-exposition (personnages dans l'ombre).



FIGURE 1.3 – Image originale Vernet, et image égalisée.

L'étalement consiste à utiliser toute la dynamique des NdG **§4.4** :

$$l_i = (M - 1) \frac{i - i_{min}}{i_{max} - i_{min}}$$
(1.10)

où  $i_{min}$  et  $i_{max}$  sont les plus petit et plus grand NdG présents dans l'image. L'intérêt de ces deux techniques est d'être complètement automatiques, donc facilement programmables.

Dans le cas général, une correction d'histogramme consiste à appliquer une transformation non linéaire T sur les niveaux de gris **§4.2** :

$$l_i = T(i) \tag{1.11}$$

T doit être une fonction monotone croissante et bornée dans l'intervalle [0, M-1] si l'on veut conserver la gradation des nuances du sombre au clair. Cela permet amélioration ou trucage, en appliquant un facteur d'amplification non linéaire sur l'échelle de gris (Fig. 1.4).



FIGURE 1.4 – Exemple de correction d'histogramme par rehaussement : a) du sombre ; b) du clair ; c) central.

La Fig. 1.5 présente en haut à gauche une image de visage avec l'histogramme correspondant, et illustre les variations réalisables par modification de cet histogramme avec diverses transformations T linéaires par morceaux.



FIGURE 1.5 – Image de visage, et variations obtenues par transformations de son histogramme.

#### Seuillage 1.4

Le seuillage (sur l'histogramme des NdG, sur leurs dérivées, sur l'entropie, sur les transformées ...) est une opération très fréquente (pour ne pas dire systématique) en traitement d'image (nombreuses applications en détection de contours, détection de mouvement ...) Il permet de passer d'une image en NdG à une image N&B (étiquetage binaire e(s)) :

$$\forall s = (x, y) \in S, \quad e(s) = \begin{cases} "1" & \text{si} & I(s) > \theta \\ "0" & \text{sinon} \end{cases}$$
(1.12)

La difficulté est évidemment le choix du seuil adéquat  $\theta$ . Sur un histogramme, on positionne souvent le seuil en un minimum local entre deux maximums (Fig. 1.2). Il existe de nombreuses méthodes et techniques adaptatives, semi-automatiques, ... de sélection du seuil [111, 97]. L'intérêt est de pouvoir ensuite appliquer des outils spécifiques aux images binaires (morphologie mathématique binaire, suivi de contours binaires) **§4.60** 

#### 1.5Contraste

Il existe différentes définitions du contraste dans une image, mais c'est toujours de la forme  $\Delta L/L$ , pour exprimer un rapport relatif contextuel (écart local) de luminance. On note ici L la luminance d'un pixel (*i.e.* son NdG), et  $L_{moy}$ ,  $L_{min}$ ,  $L_{Max}$  les valeurs moyenne, minimale et maximale de la luminance sur une zone donnée §4.35

#### Contraste de Michelson 1.5.1

- c'est à l'origine une mesure de visibilité de franges d'interférence,
- Michelson considère le cas d'une luminance L sinusoïdale,

- $\begin{array}{l} -- \text{ l'estimateur de contraste vaut }: C_M = \frac{L_{Max} L_{min}}{L_{Max} + L_{min}} \\ -- \text{ il est utilisé avec succès en psychophysique,} \\ -- \text{ il existe des variantes }: C_M = \frac{L_{Max} L}{L_{Max} + L} \text{ et } C_M = \frac{L L_{min}}{L + L_{min}} \end{array}$

#### 1.5.2Contraste de Weber

- Weber considère le cas d'un fond uniforme,
- l'estimateur est :  $C_W = \frac{L L_{moy}}{L_{moy}}$  il est utilisé en psychophysique (CIE, météorologie).
- N.B. : La courbe expérimentale de seuil de visibilité de Weber est largement utilisée.

#### 1.5.3Contraste de Gordon

- Gordon [62] considère le cas d'un stimulus complexe (image naturelle de mammographie pour diagnostic de cancer du sein),
- c'est un estimateur local basé sur le calcul du NdG moyen de deux régions :  $C_G = \frac{L_{moy_1} L_{moy_2}}{L_{moy_1} + L_{moy_2}}$
- c'est un estimateur signé.

#### 1.5.4 Contraste de Beghdadi

— c'est une mesure entre les NdG moyens des contours estimés sur 2 régions [10] — l'estimateur s'exprime par :  $C_B = \frac{C_{moy_1} - C_{moy_2}}{C_{moy_1} + C_{moy_2}}$ 

#### 1.5.5Contraste de Peli

- c'est un estimateur local à bande limitée (gamme de fréquences spatiales  $\omega_x, \omega_y$ ) [113]. Il est basé sur les propriétés du HVS (Human Visual System)
- un inconvénient est qu'il a un coût de calcul élevé.

#### 1.5.6 Contraste de Köhler

On note f(x) la fonction de luminance en la position x.

- soit un seuil s tel que : min  $(f(x), f(x_1)) \le s \le \max(f(x), f(x_1))$
- c'est un estimateur ponctuel du contraste [73] :

$$C_{xx_1}(s) = \min(|s - f(x)|, |s - f(x_1)|)$$
  
ou  $C_{xx_1}(s) = \max(|s - f(x)|, |s - f(x_1)|)$   
variantes possibles :  
$$C_{xx_1}(s) = \min\left(\frac{|s - f(x)|}{s + f(x)}, \frac{|s - f(x_1)|}{s + f(x_1)}\right)$$
  
ou  
$$C_{xx_1}(s) = \min\left(\frac{|s - f(x)|}{\max(s, f(x))}, \frac{|s - f(x_1)|}{\max(s, f(x_1))}\right).$$

### 1.6 Filtrage linéaire

Le filtrage linéaire est une opération classique pour le prétraitement des images avec deux applications principales : la réduction de bruit (filtrage passe-bas) et le rehaussement pour la détection de contours (filtrage passe-bande ou passe-haut). Les outils mathématiques utilisés sont la convolution, la transformée de Fourier et la transformée en Z.

#### 1.6.1 Convolution spatiale

Dans le cas 1D continu, l'opération (commutative) de convolution s'exprime :

$$f(x) * h(x) = \int_{-\infty}^{+\infty} f(\chi)h(x-\chi)d\chi$$
(1.13)

Le filtrage 2D peut être réalisé directement dans le domaine spatial en calculant le produit de convolution de l'image I(x, y) par la réponse impulsionnelle h(x, y) du filtre. La convolution discrète est commutative et s'exprime par J(x, y) = I(x, y) \* h(x, y):

$$I(x,y) * h(x,y) = \sum_{i} \sum_{j} I(i,j)h(x-i,y-j) = \sum_{i} \sum_{j} I(x-i,y-j)h(i,j)$$
(1.14)

Si h(i, j) est à support borné symétrique, on la représente sous forme d'un tableau de coefficients  $H_{m \times n}$ appelé masque de convolution que l'on applique sur la fenêtre image correspondante pour chaque pixel **§4.39**. Par exemple, pour m = 3 et n = 5, on aura :

$$H_{3\times5} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} \\ h_{31} & h_{32} & h_{33} & h_{34} & h_{35} \end{bmatrix}$$
(1.15)

La convolution de deux masques 1D donne un masque 2D (propriété de séparabilité) :

$$\begin{bmatrix} a & b & c \end{bmatrix} * \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} a\alpha & b\alpha & c\alpha \\ a\beta & b\beta & c\beta \\ a\gamma & b\gamma & c\gamma \end{bmatrix}$$
(1.16)

#### 1.6.2 Transformée de Fourier discrète

Cet outil permet de passer du domaine spatial au domaine fréquentiel. Le filtrage fréquentiel s'obtient en multipliant la TF (transformée de Fourier) de l'image par la fonction de transfert du filtre (gabarit). L'inconvénient principal est le coût de calcul élevé, bien qu'il existe un algorithme rapide appelé FFT (*Fast Fourier Transform*). On réserve donc le filtrage fréquentiel aux cas où la précision est le facteur important. La TFD 2D d'une image s'exprime par :

$$F(u,v) = \frac{1}{\sqrt{LC}} \sum_{x=0}^{C-1} \sum_{y=0}^{L-1} I(x,y) e^{-i2\pi(\frac{u}{C} + \frac{v}{L})}$$
(1.17)

Dans l'Eq. (1.17),  $u \in v$  sont les fréquences spatiales horizontale et verticale.

On définit de même la transformée de Fourier inverse, qui permet de remonter à l'image originale I(x, y) connaissant sa transformée F(u, v):

$$I(x,y) = \frac{1}{\sqrt{LC}} \sum_{u=0}^{C-1} \sum_{v=0}^{L-1} F(u,v) e^{+i2\pi(\frac{u}{C} + \frac{v}{L})}$$
(1.18)

On appelle spectre d'amplitude (resp. de phase) le module |F(u, v)| (resp. l'argument  $\Phi(u, v)$ ) de la transformée de Fourier, qui est une grandeur complexe. Si la dynamique du spectre est importante, on travaille sur le module en dB :  $20 \log_{10} |F(u, v)|$ . Les valeurs de module étant des réels, on peut se ramener au cas d'une source discrète d'observations en quantifiant les modules pour obtenir un ensemble de M valeurs discrètes (e.g., quantum de 0,5 dB pour M = 256).

Une interprétation physique intéressante de la TF repose sur l'examen du lieu de phase nulle. Dans l'espace normalisé  $\left(X = \frac{x}{C}, Y = \frac{y}{L}\right)$ , ce lieu correspond à :  $u.X + v.Y = k \Leftrightarrow Y = -\frac{u}{v}X + \frac{k}{v}$ . C'est une série de droites parallèles distantes de  $d = \frac{1}{\sqrt{u^2 + v^2}}$  et de direction perpendiculaire à la droite de pente tan  $\alpha = \frac{v}{u}$ . Cette relation montre que les hautes fréquences (fortes valeurs de u et v) correspondent à des lignes de phase nulle rapprochées dans le plan image (faible valeur de d).

En terme d'image, un contour se traduit dans le plan fréquentiel par une zone énergétique (forte valeur du module de la TF) située à distance D de l'origine. Plus le contour est net, plus D est grande (présence d'énergie en hautes fréquences). Plus le contour est flou, plus D est petite (présence d'énergie en basses fréquences). Donc les hautes fréquences traduisent des contours nets, et les basses fréquences traduisent des contours flous ou des régions uniformes. L'orientation des fréquences spatiales renseigne sur l'orientation des contours dans l'image. L'utilisation de la TF permet donc de pratiquer des filtrages fréquentiels sur l'image  $[\S4.10]$   $[\S4.33]$   $[\S4.63]$ .

La Fig. 1.6 montre l'allure typique du spectre d'une image réelle.



FIGURE 1.6 – a) Image réelle Couloir; (b) Spectre correspondant.

Notons que l'on obtient pour cette image une entropie H=6,5 bits < 8 bits, ce qui est cohérent pour des observations codées sur 8 bits.

#### Repliement de spectre

On sait que l'échantillonnage se traduit par une périodisation fréquentielle. Pour éviter le phénomène de repliement de spectre (*aliasing* en anglais), le théorème de Shannon stipule que l'on doit respecter la condition suivante entre fréquence d'échantillonnage  $F_e$  et fréquence maximale  $\nu_{max}$  du signal (Fig. 1.7) :



FIGURE 1.7 – a) Respect du théorème de Shannon : aucune distorsion dans la bande ] –  $F_e/2$ ;  $F_e/2$ [; b) Repliement de spectre : distorsions dans la bande ] –  $F_e/2$ ;  $F_e/2$ [. N.B. Notations de la figure :  $F_m = \nu_{max}$  dénote ici la fréquence maximale du signal analogique.  $X_a$  est le spectre du signal analogique et  $X_e$  le spectre du signal échantillonné à la fréquence d'echantillonnage  $F_e$ .

Sinon, on perd de l'information en hautes fréquences, et l'on voit apparaître à la place de fausses basses fréquences  $^2$ .

Un filtrage passe-bas permet de respecter cette contrainte : ce filtrage est souvent inhérent au système de capture pour des images réelles. En revanche, on devra se méfier particulièrement de ce problème quand on génère des images synthétiques pour tester des algorithmes de traitement §4.9§4.65.

<sup>2.</sup> En vidéo, un cas typique de repliement de spectre est celui des scènes de films avec des voitures roulant très vite, où les roues paraissent pourtant tourner lentement en sens inverse.

#### Passage de la TF à la TFD

Partant de la TF continue 1D (où la fréquence est notée  $\nu$ ) :

TF: 
$$F(\nu) = \int_{-\infty}^{+\infty} f(x)e^{-i2\pi\nu x}dx$$
 (1.20)

$$\mathrm{TF}^{-1}: \quad f(x) = \int_{-\infty}^{+\infty} F(\nu) e^{i2\pi\nu x} d\nu \qquad (1.21)$$

la discrétisation temporelle (cas des signaux fonction du temps) ou spatiale (cas des images fonction de l'espace) donne (échantillonnage à la période  $T_e$ ) :

$$x = kT_e = \frac{k}{F_e}.$$
(1.22)

Sans perte de généralité, on pose souvent  $T_e = 1$  ou bien on utilise la notation de **fréquence** réduite  $u = \frac{\nu}{F_e}$ , ce qui implique  $u_{max} \leq \frac{1}{2}$  car d'après Shannon :  $F_e \geq 2\nu_{max}$ .

On obtient ainsi la TFDT calculée sur une durée d'observation évidemment limitée :  $T = NT_e$ .

TFDT : 
$$F(u) = \sum_{k=0}^{N-1} f(k)e^{-i2\pi uk}$$
 (1.23)

Si l'on y ajoute l'échantillonnage fréquentiel avec un pas  $\Delta_e = \frac{F_e}{N}$  où N est le nombre d'échantillons, alors  $\nu = n\Delta_e \Leftrightarrow u = \frac{n}{N}$  et l'on obtient la TFD :

TFD: 
$$F(n) = \sum_{k} f(k) e^{-i2\pi \frac{nk}{N}}$$
 (1.24)

TFD<sup>-1</sup>: 
$$f(k) = \sum_{n} F(n)e^{i2\pi n\Delta_e k} = \sum_{n} F(n)e^{i2\pi \frac{nk}{N}}$$
 (1.25)

Si dans la dernière équation on remplace k par x et  $\Delta_e$  par  $f_0$  où  $f_0$  est le fondamental du signal (ou bien si l'on considère une périodisation arbitraire sur la durée d'observation T), on retrouve le lien avec le DSF classique :

$$f(x) = \sum_{n} c_n \exp\left(i\frac{2\pi nx}{T}\right) \quad \text{avec } c_n = F(n) = \frac{1}{T} \int_{[T]} f(x) \exp\left(-i\frac{2\pi nx}{T}\right) dx \tag{1.26}$$

#### 1.6.3 Transformée en Z

La transformée en Z est l'outil classique pour l'étude des signaux échantillonnés. Pour un signal 1D f(x), elle est définie par :

$$F(z) = \sum_{k=-\infty}^{+\infty} f(k) z^{-k}$$
 (1.27)

z est un nombre complexe lié à la variable symbolique p de Laplace par l'équation :

$$z = \exp\left(pT_e\right) = \exp(j2\pi\nu T_e) = \exp(j2\pi u) \tag{1.28}$$

où  $T_e$  est la période d'échantillonnage.  $z^{-1}$  est donc l'opérateur retard pur.

#### 1.6.4 Typologie des filtres

Un filtre peut avoir de nombreuses propriétés [60, 115, 33, 40] : invariance, causalité, anti-causalité, support borné, symétrie, séparabilité, RIF ou RII. Un point particulier à résoudre concerne les effets de bord et la représentation numérique du résultat de filtrage (problème de visualisation de valeurs négatives) [§4.50].

La séparabilité est un critère important pour réduire le temps de calcul.

#### Réponse impulsionnelle finie

Les filtres RIF (réponse impulsionnelle finie) sont les plus courants et les masques de taille  $3 \times 3$ , opérant sur les 8 plus proches voisins d'un pixel, sont très utilisés pour tous les traitements élémentaires sur les images :

$$- \text{ moyenneur} : H = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$
séparable :  $\frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} * \frac{1}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$ 
$$- \text{ binomial gaussien} : H = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$
séparable sous la forme : 
$$H = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$
$$- \text{ laplacien } \underbrace{\$4.6} \underbrace{\$4.29} : H = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$
$$- \text{ Sobel horizontal } : H = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$
séparable :  $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$ 
$$- \text{ gradient oblique } : H = \begin{bmatrix} -1 & 0 & 1 \\ -1 & -1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$
$$- \text{ rehausseur de contraste } : H = \begin{bmatrix} 1 & -3 & 1 \\ -3 & 9 & -3 \\ 1 & -3 & 1 \end{bmatrix}$$
$$\text{ qui est séparable } : \begin{bmatrix} -1 & 3 & -1 \end{bmatrix} * \begin{bmatrix} -1 \\ 3 \\ -1 \end{bmatrix}$$

La mise en cascade de filtres (associativité de la convolution) permet de générer des filtres de grande taille à moindre coût de calculs :

 $H_{3\times 3} * H_{3\times 3} = H_{5\times 5}$  par exemple :

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$
(1.29)

#### Réponse impulsionnelle infinie

Les filtres RII (réponse impulsionnelle infinie) sont caractérisés par leur fonction de transfert en Z ou leur équation aux différences :

$$H(z) = \frac{S(z)}{E(z)} = \frac{b_0 + b_1 z^{-1} + \dots + b_m z^{-m}}{1 + a_1 z^{-1} + \dots + a_n z^{-n}}$$
(1.30)

$$s_k = b_0 e_k + b_1 e_{k-1} + \dots + b_m e_{k-m} - a_1 s_{k-1} - \dots - a_n s_{k-n}$$

$$(1.31)$$

où  $s_k$  et  $e_k$  sont les échantillons resp. de sortie et d'entrée courants. Si les coefficients  $a_i$  sont tous nuls, on retrouve un filtre RIF. La stabilité du filtre est liée aux pôles de la fonction de transfert. Un exemple typique de filtre RII est le filtre de Canny-Deriche.

#### 1.6.5 Fonction de transfert

Par définition, la fonction de transfert est la TF de la réponse impulsionnelle. Dans le cas d'un filtre 1D, la TF de h(x) vaut **§4.8 §4.47** :

$$H(u) = \sum_{k} h(k)e^{-i2\pi u.k}$$
(1.32)

où h(k) sont les coefficients de la réponse impulsionnelle. Pour le filtre moyenneur symétrique à 3 coefficients  $H = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ , on obtient :

$$H(u) = \frac{1}{3}(e^{-i2\pi u} + 1 + e^{+i2\pi u}) = \frac{1}{3}(1 + 2\cos 2\pi u)$$
(1.33)

On peut aussi utiliser la TZ. Considérons par exemple le filtre binomial centré :  $H = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$ . La convolution du signal f(x) par le filtre h(x) donne : g(x) = f(x) \* h(x) = f(x-1) + 2f(x) + f(x+1). Soit, par transformée en  $Z : G(z) = (z^{-1} + 2 + z)F(z)$ . D'où la fonction de transfert en Z :

$$H(z) = \frac{G(z)}{F(z)} = z^{-1} + 2 + z \tag{1.34}$$

Pour une implantation matérielle de ce filtre, on décale la réponse impulsionnelle pour la rendre causale en multipliant la fonction de transfert par l'opérateur retard  $z^{-1}$ . D'où :

$$H^{-}(z) = z^{-2} + 2z^{-1} + 1 = (1 + z^{-1})^{2}.$$
(1.35)

On en déduit ainsi deux schémas fonctionnels possibles pour réaliser ce filtre numérique (Fig. 1.8).



FIGURE 1.8 – Schémas fonctionnels du filtre binomial 1D : a) direct b) version pipeline.

Ceci est le domaine de la synthèse des filtres numériques en traitement d'images [109, 103, 67, 117].

### 1.7 Détection de contours

#### 1.7.1 Gradient et laplacien

On distingue deux méthodes de détection de points-contours [125] (Fig. 1.9) :

— la recherche des extremums de la dérivée première \$4.7 \$4.56: on calcule alors le vecteur gradient défini par \$4.5 \$4.51

$$G(x,y) = \nabla I(x,y) = \begin{bmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{bmatrix}$$
(1.36)

— la recherche des passages par zéro de la dérivée seconde : on calcule alors le laplacien défini par

$$L(x,y) = \nabla^2 I(x,y) = \frac{\partial^2 I(x,y)}{\partial x^2} + \frac{\partial^2 I(x,y)}{\partial y^2}$$
(1.37)

Marr propose dans son schéma de principe brut du processus de vision (*raw primal sketch* ou esquisse primitive fondamentale) d'utiliser un filtre passe-bande de type laplacien d'une gaussienne  $\nabla^2 G$ , que l'on peut correctement approximer en pratique avec un filtre DOG différence de deux gaussiennes [104].

La Fig. 1.10 illustre la détection de contour par seuillage du module du gradient.



FIGURE 1.9 – a) Profil de contour 1D idéalisé; b) Lissage; c) Dérivée première; d) Dérivée seconde.



FIGURE 1.10 – Détection de contours, de haut en bas et de gauche à droite : a) image initiale; b) dérivée horizontale; c) dérivée verticale; d) module du gradient; e) histogramme du module avec seuil positionné; f) contours obtenus.

#### 1.7.2 Détecteur de Roberts

Il est basé sur deux masques qui calculent le gradient spatial dans deux directions à  $45^{\circ}$  et  $135^{\circ}$  :

$$H_{45} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad \qquad H_{135} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

En convoluant l'image avec ces deux masques, on obtient deux images filtrées  $I_1$  et  $I_2$  que l'on combine pour calculer le module et la direction du gradient :

$$M(x,y) = \sqrt{I_1(x,y)^2 + I_2(x,y)^2}$$
(1.38)

$$\Phi(x,y) = \arctan \frac{I_2(x,y)}{I_1(x,y)} + \frac{\pi}{4}$$
(1.39)

On applique ensuite une technique de seuillage du module pour ne garder que les points contours. Se pose ensuite le problème de chaînage et d'approximation polygonale pour exhiber des contours fermés. Ce détecteur est sensible au bruit haute fréquence, que l'on peut réduire par un filtrage passe-bas initial.

#### 1.7.3 Détecteur de Prewitt

Il consiste en deux masques élémentaires, qui calculent les dérivées horizontale et verticale moyennées sur un voisinage  $3 \times 3$ :

Prewitt horizontal : 
$$H_0 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$
  
et Prewitt vertical :  $H_{90} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ 

La procédure est ensuite la même pour extraire les contours, mis à part la suppression du terme additif  $\frac{\pi}{4}$  dans l'Eq. (1.39) **§4.37**.

#### 1.7.4 Détecteur de Sobel

Il utilise également deux masques, l'un horizontal  $H_0$ , l'autre vertical  $H_{90}$ :

Sobel vertical :  $H_{90} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$  séparable en lissage et dérivation :  $\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ N.B. : Incluant un pré-filtrage passe-bas, les détecteurs de Prewitt et Sobel sont moins sensibles au

N.B. : Incluant un pré-filtrage passe-bas, les détecteurs de Préwitt et Sobel sont moins sensibles au bruit que celui de Roberts §4.44.

#### 1.7.5 Détecteur de Canny-Deriche

Il fait partie des extracteurs par optimisation de critères (sensibilité, localisation et détection optimales étant donné un modèle de contour) [28, 42]. Sa réponse impulsionnelle 1D s'exprime par :

$$h(x) = cxe^{-\alpha|x|} \tag{1.40}$$

 $\alpha$  est un coefficient de lissage et c un coefficient de normalisation déterminé en maximisant l'amplitude de la réponse à un échelon unité :  $c = -\frac{(1-e^{-\alpha})^2}{e^{-\alpha}}$ .

Il se décompose en deux parties, dont on peut calculer les fonctions de transfert en Z:

— réponse causale :  $h^+(x) = cxe^{-\alpha x}$  pour  $x \ge 0$ , d'où

$$H^{+}(z) = c \sum_{k=0}^{+\infty} k z^{-k} e^{-\alpha k} = c \frac{e^{-\alpha} z^{-1}}{(1 - e^{-\alpha} z^{-1})^2}$$

— réponse anti-causale :  $h^-(x) = cxe^{\alpha x}$  pour  $x \le 0$ , d'où

$$H^{-}(z) = c \sum_{k=0}^{+\infty} -kz^{k}e^{-\alpha k} = c \frac{-e^{-\alpha}z}{(1-e^{-\alpha}z)^{2}}$$

On en déduit aisément les équations de récurrence du filtre §4.27.

#### 1.7.6 Détecteur de Shen-Castan

Le détecteur de Shen et Castan est une version de filtre RII plus simple que le filtre de Canny-Deriche : un unique paramètre permet de régler la force du filtre (Eq. (5.1), p. 119), le second paramètre étant le choix du seuil pour ne conserver que les points ayant les plus fortes valeurs de dérivées [121]. Il a été utilisé avec succès, par exemple en contrôle qualité automatique pour mesurer la mouillabilité de polymères traités par plasma [92, 76].

#### 1.7.7 Algorithme de Rosenfeld & Kak

Contrairement aux filtres précédents qui opèrent sur des images à NdG, il s'agit ici d'un extracteur de contours opérant sur des images binaires (N&B).

#### Définitions

On appelle *direction courante* du contour de l'objet, en un pixel appartenant au contour, le vecteur de translation qui joint ce pixel au pixel du contour le précédant dans le sens trigonométrique.

Partant d'un pixel donné dans l'image, les 8 premiers voisins sont définis par leur direction par rapport au pixel de départ et par les coordonnées des vecteurs de translation permettant de passer d'un pixel au suivant connaissant la direction courante (code directionnel de Freeman, Fig .1.11) **§4.38 §4.61**.



FIGURE 1.11 – Poursuite de contours dans une image binaire.

#### Principe de poursuite de contour

Duda et Hart suggèrent un algorithme très rapide pour extraire les frontières d'un objet [44]. L'inconvénient de leur algorithme provient du fait que le suiveur de contour ne prend en compte que 4 directions (directions 0, 2, 4, 6, du code de Freeman). Autrement dit, il explore 4 voisins du pixel de départ. Ainsi les pixels reliés à l'objet selon les directions diagonales sont ignorés. On lui préfère donc l'algorithme de Rosenfeld et Kak, qui utilise une exploration à 8 voisins dans le sens trigonométrique [118] §4.12 §4.43.

Le principe est le suivant : supposons que l'on connaisse un point de contour et la direction courante du contour en ce point. La recherche du point suivant consiste en un balayage des 8 voisins du point courant. Le balayage est effectué de l'extérieur vers l'intérieur de l'objet. Le premier voisin qui appartient à l'objet devient le nouveau point courant du contour et la direction courante est celle qui joint les deux points contours. En examinant tous les cas de figure possibles, on constate qu'il suffit d'explorer un secteur limité à *direction courante* -2 jusqu'à *direction courante* +4.

Pour accélérer l'exécution du programme de suivi de contour, les directions à tester et les composantes de vecteurs de translation (qui permettent de passer d'un pixel au suivant, connaissant la direction courante) sont regroupées dans des tableaux (Tab. 1.1).

Un exemple d'application pour l'extraction des lèvres d'un visage est montré Fig. 1.12.

Direction courante	0	1	2	3	4	5	6	7
$v_x$	1	1	0	-1	-1	-1	0	1
$v_y$	0	-1	-1	-1	0	1	1	1
	6	7	0	1	2	3	4	5
Directions	7	0	1	2	3	4	5	6
à explorer	0	1	2	3	4	5	6	7
	1	2	3	4	5	6	7	0
	2	3	4	5	6	7	0	1
	3	4	5	6	7	0	1	2
	4	5	6	7	0	1	2	3

TABLE 1.1 – Coordonnées  $(v_x, v_y)$  des vecteurs de translation, et directions à explorer en fonction de la direction courante



FIGURE 1.12 – Détection de contours de lèvres par l'algorithme de Rosenfeld et Kak avec deux points initiaux. En haut : masques binaires de lèvres ; en bas : contours extraits.

#### Choix du point initial

La détermination du point de départ pour la poursuite des contours est simple quand on n'a qu'un seul objet binaire connexe sans trou. Mais un masque de lèvres est constitué pratiquement de deux zones : les lèvres et l'intérieur de la bouche. Par conséquent, deux points initiaux, au moins, sont nécessaires : l'un pour détecter le contour externe, et l'autre pour détecter le contour de l'intérieur de la bouche (contour interne des lèvres).

On effectue la détection de contour externe en partant du point initial  $(x_{moy}, y_1)$ . Ce point est le point de contour externe de la lèvre supérieure situé sur l'axe vertical médian des lèvres. Partant de  $y_{min}$  dans le sens croissant des y, ce point est rapidement localisé : il s'agit du premier point appartenant au masque des lèvres sur l'axe de celles-ci (Fig. 1.13). La direction initiale  $D_0$  de contour en ce point de départ est fixée à 6 (sens de recherche du point initial) pour assurer le démarrage de la poursuite dans le sens trigonométrique.

On cherche de même un point initial du contour interne de la lèvre supérieure situé sur l'axe vertical médian. L'algorithme d'initialisation recherche, en partant de  $y_1$  dans le sens croissant des y (vers l'intérieur de la bouche), le point limite du masque comme point initial. On obtient alors le point  $(x_{moy}, y_2)$  de la Fig. 1.13. Dans ce cas, la direction initiale est fixée à 2.

N.B. On pourrait aussi bien utiliser le couple de points initiaux  $y_3$  et  $y_4$ .



FIGURE 1.13 – Les 4 points initiaux candidats.

#### 1.8 Morphologie mathématique

#### 1.8.1 Introduction

La morphologique mathématique concerne des transformations géométriques non linéaires (filtrage non linéaire). Elle est basée sur la théorie des ensembles. On applique à l'image un élément structurant E. On peut faire une analogie avec le filtrage linéaire : E correspondrait au masque de coefficients d'un filtre, et l'opération morphologique remplacerait une convolution. Les opérations de base sont l'érosion, la dilatation, l'ouverture et la fermeture. On distingue la morphologie binaire, s'appliquant aux images en N&B, et la morphologie pour images à NdG.

Le principe général consiste à comparer les objets d'une image avec un objet de référence de forme et de taille données qu'on appelle élément structurant. En morphologie binaire, l'élément structurant joue en quelque sorte le rôle d'une gomme ou d'un pinceau plus ou moins épais choisi pour modifier un dessin.

Ce type de filtrage est recommandé dans le cas d'un bruit important (bords irréguliers, trous à l'intérieur des objets), car il permet de boucher les trous, effacer les taches et adoucir les angles. L'élément structurant E translaté en chaque site s de l'image est noté  $E_s$ . La Fig. 1.14 montre des éléments structurants classiques.



FIGURE 1.14 – Eléments structurants : a) croix 4-connexe ; b) carré 8-connexe ; c) ensemble de 4 traits pivotés (×=indifférent).

Tout l'enjeu de la morphologie mathématique repose évidemment sur le bon choix de l'élément structurant, en fonction du problème à résoudre [81] §4.34.

#### 1.8.2 Opérations binaires (images N&B)

L'érosion  $\ominus$  s'obtient en considérant l'« intersection » de l'objet X avec l'élément structurant E (il faut une inclusion complète de l'élément structurant dans l'objet) **§4.11 §4.54** :

$$X \ominus E = X \cap E = \{s | E_s \subseteq X\} \tag{1.41}$$

La dilatation  $\oplus$  s'obtient en considérant l'« union » de l'objet avec l'élément structurant (il faut une intersection non vide de l'élément structurant avec l'objet) :

$$X \oplus E = X \cup E = \{s | E_s \cap X \neq \emptyset\}$$

$$(1.42)$$

L'ouverture  $\circ$  est la succession d'une érosion puis dilatation (*roll inside*<sup>3</sup>) :

$$X \circ E = (X \ominus E) \oplus E \tag{1.43}$$

La fermeture • est la succession d'une dilatation puis érosion (*roll outside*<sup>4</sup>), cf. illustration Fig. 1.15  $[\S4.13]$ :

$$X \bullet E = (X \oplus E) \ominus E \tag{1.44}$$



FIGURE 1.15 – a) objet; b) élément structurant; c) ouverture; d) fermeture.

#### 1.8.3 Propriétés

La dilatation est commutative et associative.

L'érosion et la dilatation sont invariantes en translation :

$$X_s \oplus E = (X \oplus E)_s \tag{1.45}$$

$$X_s \ominus E = (X \ominus E)_s \tag{1.46}$$

Croissance :

$$X \subseteq Y \Rightarrow \begin{cases} X \oplus E \subseteq Y \oplus E \\ X \oplus E \subseteq Y \oplus E \\ X \bullet E \subset Y \bullet E \\ X \circ E \subset Y \circ E \end{cases}$$
(1.47)

<sup>3.</sup> Cet terme traduit simplement l'opération d'ouverture : prenons comme élément structurant une balle de tennis, et comme objet une boîte de chaussures. On place la balle dans la boîte et l'on secoue assez longtemps pour que la balle roule partout à l'intérieur de la boîte. Alors, l'ouvert sera l'ensemble des points de l'espace intérieur à la boîte qui auront pu être atteints par la balle ; on comprend que seuls les points dans les angles de la boîte ne sont pas touchés : l'ouverture arrondit donc les angles comme le montre la Fig. 1.15c.

<sup>4.</sup> L'opération de fermeture s'illustre de manière analogue à l'ouverture, mais en faisant appel à la notion d'ensemble complémentaire (c'est-à-dire l'extérieur de l'objet) : on laisse la balle rouler à l'extérieur ; tous les points de l'espace qu'elle n'a pas touchés appartiennent au fermé, comme le montre la Fig. 1.15d.

Distributivité :

$$(X \cup Y) \oplus E = (X \oplus E) \cup (Y \oplus E)$$

$$X \oplus (D \cup E) = (X \oplus D) \cup (X \oplus E)$$

$$(1.48)$$

$$(1.49)$$

$$(X \cap Y) \ominus E = (X \ominus E) \cap (Y \ominus E)$$

$$(1.51)$$

 $It \acute{e} rativit\acute{e}:$ 

$$(X \oplus D) \oplus E = X \oplus (D \oplus E) \tag{1.52}$$

$$(X \ominus D) \ominus E = X \ominus (D \oplus E) \tag{1.53}$$

Complémentarité (dualité entre fond et forme) : le complémentaire de l'ensemble X étant défini par :  $X^c = \{s | s \notin X\}$ , on a :

$$(X \ominus E)^c = X^c \oplus E \tag{1.54}$$

$$(X \bullet E)^c = X^c \circ E \tag{1.55}$$

$$(X \circ E)^c = X^c \bullet E \tag{1.56}$$

Extensivité et anti-extensivité :

$$X \circ E \subset X \subset X \bullet E \tag{1.57}$$

Idempotence :

$$(X \bullet E) \bullet E = X \bullet E \tag{1.58}$$

$$(X \circ E) \circ E = X \circ E \tag{1.59}$$

#### 1.8.4 Autres opérations

L'opération tout-ou-rien (*Hit or Miss*)  $\oslash$  s'obtient à l'aide d'un élément structurant plus général  $E = (E_1, E_2)$  composé d'une forme  $E_1$  et d'un fond  $E_2$ .

$$X \oslash E = (X \ominus E_1) \cap (X^c \ominus E_2) = (X \ominus E_1) - (X \oplus E_2)$$
(1.60)

Cet ensemble représente tous les points où, simultanément,  $E_1 \ll \text{colle} \gg \text{dans } X$ , et  $E_2 \ll \text{colle} \gg \text{dans } X^c$ .

L'extraction de frontières  $\beta(X)$  s'obtient par différence entre X et son érodée par l'élément structurant carré de la Fig. 1.14b.

$$\beta(X) = X - (X \ominus E) \tag{1.61}$$

Le remplissage de région s'obtient à partir de dilatations itérées **§4.52 §4.40** :

$$X_k = (X_{k-1} \oplus E) \cap X^c \qquad k = 1, 2, 3, \dots$$
(1.62)

où X est la frontière initiale et  $X_0 = \{p\}$  un point initial intérieur à la frontière. L'union de X et des  $X_k$  représente le résultat du remplissage. L'élément structurant habituellement utilisé est la croix (Fig. 1.14a).

L'extraction de composantes connexes s'obtient par **§4.55** :

$$X_k = (X_{k-1} \oplus E) \cap X \qquad k = 1, 2, 3, \dots$$
(1.63)

où X est l'objet initial et  $X_0 = \{p\}$  un point de connexion initial connu.

La coque convexe C d'un objet (*convex hull*, Fig. 1.16) s'obtient par itérations de tout-ou-rien, à l'aide de quatre éléments structurants pivotés  $E_i$  avec i = 1, 2, 3, 4 (Fig. 1.14 c) et jusqu'à convergence des itérations k ( $k = K_{conv}$ ) :

$$X_{k}^{i} = (X_{k-1}^{i} \otimes E_{i}) \cup X \text{ pour } k = 1, 2, 3, \dots \text{ et où } X_{0}^{i} = X$$
(1.64)

$$C = \bigcup_{i=1}^{i} X^{i}_{K_{conv}} \tag{1.65}$$



FIGURE 1.16 – Coque convexe.

L'amin<br/>cissement  $\otimes$  enlève des points à la frontière de<br/> X :

$$X \otimes E = X - (X \otimes E) = X \cap (X \otimes E)^c$$
(1.66)

L'épaississement  $\odot$  ajoute à X des points de la frontière de  $X^c$  :

$$X \odot E = X \cup (X \oslash E) \tag{1.67}$$

La squelettisation s'obtient par amincissements répétés **§4.15** :

$$S = X \otimes \{E\} = \dots((((X \otimes E_1) \otimes E_2) \otimes E_3) \otimes E_4) \otimes E_1) \otimes E_2)\dots$$
(1.68)

Il existe des opérations plus complexes comme la taille (pruning) qui utilise des érosions successives avec des éléments structurants de plus en plus fins (tamis) permettant de classifier les régions.

#### 1.8.5 Opérations sur images en NdG

La morphologie mathématique s'applique aussi aux images en NdG, en remplaçant les notions d'intersection et d'union valables uniquement pour des images binaires (comportant un fond et une forme), par les notions de minimum et de maximum d'intensité dans un voisinage (valeurs sup et inf).

L'érosion  $\ominus$  donne la plus petite valeur prise par I à l'intérieur de l'élément structurant  $E_s$  centré sur le pixel s (Fig. 1.17) :

$$\forall s, (I \ominus E)(s) = \inf\{I(p); p \in E_s\}$$
(1.69)



FIGURE 1.17 – Erosion et dilatation en NdG.

La dilatation  $\oplus$  donne la plus grande valeur prise par I sur le support structurant  $E_s$ :

$$\forall s, (I \oplus E)(s) = \sup\{I(p); p \in E_s\}$$
(1.70)

L'ouverture  $\circ$  et la fermeture  $\bullet$  sont définies comme en morphologie binaire (Eq. 1.43 et 1.44).

Intuitivement, l'ouverture consiste à suivre le profil par le dessous en montant l'élément structurant le plus haut possible; cela revient à écrêter les pics sans toucher aux vallées. De même, la fermeture consiste à suivre le profil par le dessus en descendant E le plus bas possible; cela revient à combler les vallées sans toucher aux pics.

L'opération chapeau haut de forme (top hat)  $H = I - I \circ E$  permet d'exhiber les pics de I. De même, on peut extraire les creux en soustrayant  $I - I \bullet E$ .

Le lissage morphologique s'obtient par une ouverture suivie d'une fermeture §4.14 :

$$L(I) = (I \circ E) \bullet E. \tag{1.71}$$

Le gradient morphologique correspond à la soustraction entre dilatée et érodée **§4.45** :

$$G(I) = (I \oplus E) - (I \ominus E). \tag{1.72}$$

Enfin, le filtre médian appartient aussi à la famille des filtres non-linéaires de la morphologie mathématique : au lieu du minimum ou du maximum dans un voisinage, on sélectionne la valeur médiane. Il est très utilisé comme préfiltrage du bruit impulsionnel dans les images §4.36 §4.42. On peut également concevoir des filtres non-linéaires *ad hoc* dédiés à des applications spécifiques, par exemple pour la détection de formes horizontales allongées telles les lèvres §4.48.

#### **1.9** Segmentation couleur

#### 1.9.1 Vision des couleurs

L'intérêt d'utiliser la couleur en traitement d'image repose sur plusieurs constats. D'abord, la couleur est un descripteur puissant car l'œil humain peut discerner des milliers de couleurs, alors qu'il ne peut distinguer que quelques dizaines de niveaux de gris. Par ailleurs, la teinte est une grandeur peu sensible aux variations d'éclairage (ombres), contrairement à la luminance.

Les principaux constituants de l'œil sont présentés Fig 1.18 :

- la cornée : protection, filtre,
- l'iris : diaphragme (variation d'un facteur 10 en surface),
- le cristallin : indice optique variable, focus (déformable),
- la rétine : couche de capteurs photosensibles (120 millions de bâtonnets et 6 millions de cônes). Les cônes sont sensibles aux trois couleurs RVB (vision chromatique) et les bâtonnets à l'intensité lumineuse I (vision achromatique). La fovéa est le centre de la rétine. La zone aveugle correspond au rattachement du nerf optique §4.16.
- le nerf optique : transport de l'information (100000 neurones).



FIGURE 1.18 – a) Coupe horizontale de l'œil; b) Répartition des photorécepteurs sur la rétine.

La vision humaine n'a pas une sensibilité égale dans tout le spectre visible : l'énergie lumineuse perçue par l'œil est fonction de la longueur d'onde comprise entre 400 et 700 nm (Fig. 1.19).



FIGURE 1.19 – Courbe de sensibilité de l'œil en fonction de la longueur d'onde.

#### 1.9.2 Trichromie

Alors que les images achromatiques (en NdG) sont caractérisées par une seule valeur (intensité lumineuse ou **luminance** I), les images couleurs sont caractérisées par trois composantes [130].

On peut utiliser les trois couleurs primaires (RVB) : rouge, vert et bleu, correspondant à trois longueurs d'onde du spectre visible (Fig. 1.19) :

 $\lambda_R = 610 \text{ nm}, \lambda_V = 535 \text{ nm}, \lambda_B = 470 \text{ nm}.$ 

Les autres couleurs visibles s'obtiennent par combinaison des couleurs primaires (synthèse additive, Fig. 1.20a).



FIGURE 1.20 – Synthèse des couleurs : a) additive; b) soustractive.

Le modèle RVB est typiquement utilisé pour la réalisation des moniteurs et caméras couleurs **§4.59**. On représente l'espace RVB par un cube (Fig. 1.21a).

Les couleurs secondaires cyan, magenta et jaune ou *yellow* (CMY) s'obtiennent par additivité des couleurs primaires :

$$C = V + B \tag{1.73}$$

$$M = R + B \tag{1.74}$$

$$Y = R + V \tag{1.75}$$

Le modèle CMY est typiquement utilisé pour les imprimantes couleurs (synthèse soustractive, Fig. 1.20b).

Un troisième modèle nommé  $YC_rC_b$  est utilisé comme standard en TV (télévision) couleur, assurant la compatibilité avec la TV N&B **§4.17**. Y représente la luminance.  $C_r, C_b$  sont deux composantes codant la chrominance (par différences de couleurs). Leur définition (et leur appellation) varie selon



FIGURE 1.21 – a) Cube couleur RVB; b) Pyramide couleur TLS.

le format : YUV pour les formats européens (standards PAL et SECAM), YIQ pour le format nordaméricain (standard NTSC).

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.3 & 0.59 & 0.11 \\ -1.33 & 1.12 & 0.21 \\ -0.45 & -0.88 & 1.33 \end{bmatrix} \cdot \begin{bmatrix} R \\ V \\ B \end{bmatrix}$$
(1.76)
$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.3 & 0.59 & 0.11 \\ 0.6 & -0.28 & -0.32 \\ 0.21 & -0.52 & 0.31 \end{bmatrix} \cdot \begin{bmatrix} R \\ V \\ B \end{bmatrix}$$
(1.77)

D'un point de vue **perception**, les caractéristiques pertinentes pour distinguer les couleurs sont la teinte, la luminance et la saturation.

On définit ainsi le modèle TLS (appelé HSI en anglais pour hue, saturation, intensity) :

$$T = \begin{cases} \theta = \arccos \frac{1}{2} \frac{(R-V) + (R-B)}{\sqrt{(R-V)^2 + (R-B)(R-V)}} & \text{si } B < V \\ 2\pi - \theta & \text{si } B > V \end{cases}$$

$$L = \frac{R+V+B}{3}$$

$$S = 1 - \frac{\min(R,V,B)}{L}$$

$$(1.78)$$

A la place de la saturation, on peut utiliser la pureté P = S.L, et adopter un calcul de teinte basé sur l'arc-tangente (espace TLP) :

$$T = \frac{\pi}{2} - \arctan\left(\frac{2R - V - B}{\sqrt{3}(V - B)}\right) + k$$

$$L = \frac{R + V + B}{3}$$

$$P = \frac{R + V + B}{3} - \min(R, V, B)$$
(1.79)

où : k = 0 si V > B et  $k = \pi$  sinon.

Il existe donc de nombreux espaces pour modéliser l'information de couleur. Un bon espace correspond à un choix de trois composantes bien décorrélées : c'est tout l'enjeu des différentes transformées

couleurs proposées. Un autre problème essentiel est la sensibilité au bruit du calcul de la teinte, reposant sur un calcul de rapport de différences.

Pour palier ce problème, certaines approches s'inspirent de constatations concernant le système visuel humain :

- le premier traitement réalisé par l'œil est une compression logarithmique de l'information qu'il perçoit,
- les capteurs majoritaires au centre de la fovéa sont les cônes sensibles au R et au V,
- la courbe de sensibilité de l'œil est maximale pour le vert.

Ces remarques conduisent à proposer des espaces couleurs qui offrent des alternatives intéressantes pour faire de la segmentation couleur robuste, que ce soit pour des applications en robotique, en visiophonie ou en visioconférence.

Le système d'équations 1.80 (où M = 256) donne l'expression de la transformée logarithmique LUX [80, 79, 90].

$$L = (R+1)^{0.3} (G+1)^{0.6} (B+1)^{0.1} - 1$$

$$U = \begin{cases} \frac{M}{2} \left(\frac{R+1}{L+1}\right) & \text{si } R < L \\ M - \frac{M}{2} \left(\frac{L+1}{R+1}\right) & \text{sinon} \end{cases}$$

$$X = \begin{cases} \frac{M}{2} \left(\frac{B+1}{L+1}\right) & \text{si } B < L \\ M - \frac{M}{2} \left(\frac{L+1}{B+1}\right) & \text{sinon} \end{cases}$$
(1.80)

L'effet principal de cette transformée logarithmique est d'amplifier le contraste tout en s'affranchissant le plus possible de l'éclairage. La Fig. 1.22 montre que l'espace LUX augmente le contraste des composantes chromatiques tout en gardant la cohérence des teintes (rouge, bleu) §4.28 §4.30.



FIGURE 1.22 – Transformées chromatiques sur une image de visage Marc. En haut : composantes Y,  $C_r$  et  $C_b$ ; En bas : composantes L, U et X.

#### 1.9.3 Segmentation

Segmenter une image S, c'est réaliser une partition mathématique en plusieurs régions  $R_i$ , chacune étant homogène au sens d'un certain critère (par exemple la couleur, la texture, le mouvement etc.), et telles que les deux propriétés suivantes soient vérifiées :

$$R_i \cap R_j = \emptyset \quad \forall i, \forall j \tag{1.81}$$

$$\bigcup_{i} R_i = S \tag{1.82}$$

Une méthode très classique de segmentation de la couleur dans les images est l'algorithme de classification des K-moyennes (K-means [101]), qui est disponible dans la bibliothèque Matlab. Il suppose de connaître *a priori* le nombre de classes de couleur à partitionner. Un exemple illustrant l'application de cette méthode est le suivi d'alevins d'anguilles en bassin marqués avec un élastomère couleur [47].

On présente ci-après quelques approches alternatives envisageables.

#### Projection sur un axe

Pour réduire la quantité d'information apportée par la chrominance, on peut ne s'intéresser qu'à la projection sur un seul des axes. Un axe privilégié de projection est l'axe de la luminance car :

— c'est la projection intrinsèque faite par une caméra vidéo N&B,

— elle correspond à la composante L de l'espace TLS,

— elle se calcule facilement en faisant la moyenne des quantités RVB.

Pour une application spécifique (suivi de teinte de visage par exemple), toute autre projection est possible (par exemple sur l'axe rouge).

#### Analyse en composantes principales

Une image couleur peut être considérée comme un ensemble d'échantillons statistiques  $I(i,j) = \begin{bmatrix} R(i,j) \\ V(i,j) \\ B(i,j) \end{bmatrix}$  caractérisé par ses moments d'ordre 1 (vecteur moyenne  $M = \begin{bmatrix} M_R \\ M_V \\ M_B \end{bmatrix}$ ) et d'ordre 2

 $\begin{bmatrix} B(i,j) \end{bmatrix}$ (matrice de covariance symétrique  $C = \begin{bmatrix} C_{RR} & C_{RV} & C_{RB} \\ C_{RV} & C_{VV} & C_{VB} \\ C_{RB} & C_{VB} & C_{BB} \end{bmatrix}$ ).

Dans le cas discret, ces grandeurs s'estiment par le calcul de l'espérance mathématique E:

$$M_X = E[X] = \frac{1}{MN} \sum_{i=0}^{M} \sum_{j=0}^{N} X(i,j)$$
(1.83)

$$C_{XY} = E[(X - M_X)(Y - M_Y)] = \frac{1}{MN} \sum_{i=0}^{M} \sum_{j=0}^{N} (X(i,j) - M_X)(Y(i,j) - M_Y)$$
(1.84)

L'ACP permet de définir une projection plus intelligente. Elle consiste à diagonaliser la matrice de covariance (changement de base) et à ne conserver que la composante (vecteur propre principal) correspondant au coefficient maximal (valeur propre maximale), c'est-à-dire celle apportant le plus d'information. Elle est basée sur le calcul des valeurs propres et vecteurs propres de C §4.18.

#### Segmentation par seuillage de teinte

Un histogramme de couleur fournit une densité de probabilité permettant par seuillage ou filtrage, de trouver les pixels d'une région spécifique, par exemple la peau humaine.

Pour détecter les visages, la teinte à dominante rouge est un discriminateur satisfaisant, quelle que soit la couleur effective de la peau. En effet, la différence de couleur de peau entre individus provient avant tout d'une différence de saturation (c'est-à-dire de quantité de blanc mélangé à la couleur présente dans le visage) et non pas d'une différence de teinte.

L'espace logarithmique LUX peut être radicalement simplifié si l'application ne concerne que l'analyse de visage  $\boxed{\$4.32}$ :

$$I = \frac{R+V+B}{3} \tag{1.85}$$

$$H = \begin{cases} 256 \cdot \frac{V}{R} & \text{si} \quad V < R\\ 255 & \text{sinon} \end{cases}$$
(1.86)

On utilise uniquement les plans couleur R et V pour deux raisons principales : tout d'abord le visage est principalement rouge et contient peu de bleu; par ailleurs, le vert est souvent utilisé en

traitement vidéo comme une bonne approximation de la luminance. Cette expression de H comme un simple rapport de vert sur rouge (Eq. (1.86)) a déjà, sous d'autres formes, fait ses preuves en robotique et en vision par ordinateur.

Le visage correspond alors à une teinte moyenne plutôt proche de 128 par cette transformée logarithmique. L'équation 1.89 donne la justification de cette affirmation grossière :

$$I = \frac{R+V+B}{3} \text{ avec } B \sim 0 \text{ et } I \sim V$$
(1.87)

d'où 
$$R \sim 2I \sim 2V$$
 (1.88)

soit 
$$H = 256 \frac{V}{R} \sim 256 \frac{V}{2V} \sim 128$$
 (1.89)

La Fig. 1.23 montre la distribution typique de la teinte sur une image de visage centrée sur les lèvres. On constate ici que le visage et les lèvres occupent la gamme de teinte [125; 175]. Pour éliminer le mode dû au fond (situé dans la gamme [200; 255]), prépondérant dans des conditions de prise de vue télévisée, il suffit de filtrer la distribution de teinte par un filtre centré sur 128 de pente suffisamment douce pour laisser une relative souplesse à l'estimation (filtre gaussien de centre 128, et d'écart type 128).



FIGURE 1.23 – En haut : image de visage (couleur), plan teinte logarithmique correspondant. En bas : distribution de teinte, distribution filtrée.

Pour extraire le mode des lèvres, qui se trouve la plupart du temps confondu avec le mode de teinte de la peau du visage, on procède de façon hiérarchique par itération du filtrage : on applique là aussi un filtre gaussien de centre 128 mais d'écart type plus réduit, typ.  $\sigma = 64$ .

#### Classification crédibiliste de la teinte

La théorie de l'évidence, appelée aussi théorie des fonctions de croyances, est une alternative à la théorie probabiliste bayésienne classique. Elle permet de traiter des situations ambiguës, dans le cas de données incomplètes ou bruitées, ce qui est souvent la situation rencontrée dans le traitement d'images réelles [55].

On ne la détaillera pas ici, mais on peut consulter les références suivantes qui illustrent son application pour la détection et le suivi de visage basé sur l'information de teinte chair, et où l'on trouvera une bibliographie sur le sujet [49, 50, 51, 52, 53, 54, 86]. Un audioslide est également visible en ligne sur YouTube [94].

#### Exemple applicatif en supervision du littoral

La supervision du littoral est un exemple parmi d'autres qui illustre l'intérêt d'une segmentation basée sur l'information de couleur. On peut utiliser un seuillage de la teinte bleue U pour segmenter l'océan (Fig. 1.24) pour une étude de l'évolution du trait de côte.



FIGURE 1.24 – Suivi du trait de côte : a) plage de l'embouchure de l'Adour; b) contours; c) teinte bleue U, représentée en NdG; d) seuillage de teinte.
# Chapitre 2

# Traitement de séquences d'images

# Préambule

Ce deuxième chapitre concerne le cas des séquences vidéo comportant une succession d'images numériques. La dimension temporelle ainsi introduite mène aux notions supplémentaires de mouvement et de vitesse; on présente alors les outils de détection, de régularisation statistique, de multirésolution, d'estimation et de prédiction, pour finir par le principe du codage vidéo. Ce chapitre offre l'occasion d'exposer les possibilités d'implantations matérielles, point crucial car la grande quantité de données des vidéos nécessite des solutions adaptées pour pouvoir traiter en temps-réel (c'est-à-dire à la cadence vidéo).

# 2.1 Détection de mouvement

### 2.1.1 Introduction

L'analyse du mouvement dans les séquences d'images est un domaine de recherche actif [85], en raison de son importance dans de nombreuses applications : télésurveillance [129], compression pour les télécommunications ou l'archivage, diagnostic médical, météorologie, contrôle non destructif, robotique mobile dont le guidage de drones [72, 71, 69, 70], multimédia avec gestion de qualité de service [21, 75] etc.

On distingue habituellement quatre phases en analyse de mouvement : la **détection** des zones mobiles, l'**estimation** des vecteurs-vitesses (en chaque pixel ou pour chaque objet), la **segmentation** (en zones cohérentes au sens du mouvement) et l'**interprétation** de haut niveau (reconnaissance de formes faisant appel à l'intelligence artificielle). Ces quatre étapes ne sont en aucun cas indépendantes ni forcément séquentielles, mais au contraire fortement interdépendantes (notamment en ce qui concerne les deux phases estimation-segmentation, posant le fameux problème de la poule et de l'œuf). Nous nous intéressons ici à la phase de détection du mouvement des objets mobiles dans le cas d'une caméra fixe par rapport à la scène observée.

### 2.1.2 Principe

La détection de mouvement consiste à attribuer à chaque pixel, ou site s = (x, y, t) des images d'une séquence, un attribut qu'on appelle **étiquette**  $e_s$ , indiquant si le pixel appartient à un objet mobile ou au fond fixe de la scène observée :

$$e_s = e(x, y, t) = \begin{cases} a = "1" & \text{si le pixel appartient à une zone mobile,} \\ b = "0" & \text{si le pixel appartient au fond fixe.} \end{cases}$$
(2.1)

Etiqueter chaque pixel s de l'image à l'instant t permet ainsi d'obtenir une carte binaire des changements temporels (Fig. 2.1).

En faisant l'hypothèse d'une caméra fixe et d'un éclairement quasi constant de la scène observée, on peut extraire en chaque pixel une information de bas niveau, qu'on appelle **observation**  $o_s$ , portant



FIGURE 2.1 – En haut : 6 images d'une séquence Rue ; En bas : cartes binaires obtenues par seuillage entropique (pixels mobiles en blanc, seuil  $\theta = 4\sigma_e$ ).

sur la variation temporelle de l'intensité lumineuse I du pixel. On utilise comme observation la valeur absolue de la différence temporelle d'intensité lumineuse entre deux instants :

$$o_s = o(x, y, t) = |I_t(s) - I_{t-1}(s)| = |I(x, y, t) - I(x, y, t-1)|$$
(2.2)

Dans le cas idéal, cette variation temporelle entre deux instants d'acquisition t-1 et t est nulle pour un pixel du fond fixe, non nulle s'il y a eu mouvement d'un objet d'intensité non parfaitement uniforme. Cependant, cette observation de bas niveau fournit une information qui est bruitée et peu robuste (bruit d'acquisition de la caméra, quantification, etc.). Elle ne donne qu'une information partielle sur les objets en mouvement et nécessite un seuillage adéquat. Se pose alors le problème du choix automatique de la valeur du seuil  $\theta$  [1, 122, 65].

Un seuillage simple ne permet pas de bien extraire les objets mobiles. En effet, on distingue typiquement 4 zones dans le champ d'observations (Fig. 2.2) :

- 1. zone de fond fixe,
- 2. zone d'écho (zone de fond découverte par l'objet à l'instant t),
- 3. zone de recouvrement (zone de fond recouverte par l'objet à l'instant t).
- 4. zone de glissement de l'objet sur lui-même,



FIGURE 2.2 – Carte binaire des changements temporels entre les instants t - 1 et t. Cas d'un rectangle mobile et d'un camembert fixe.

Il faut d'une part éliminer l'écho, d'autre part reconstruire l'intérieur des objets mobiles (zone de glissement). Différentes techniques, très simples dans leur principe, mais relativement peu robustes, permettent d'obtenir les zones mobiles :

- la technique de simple différence par rapport à une image de référence censée ne contenir que le fond fixe de la scène : le problème est bien sûr l'obtention de cette image de référence, qu'il faut éventuellement mettre à jour régulièrement en fonction de l'évolution de la scène §4.41,
- la technique du ET logique entre cartes binaires de changements temporels, qui ne donne de bons résultats que si le mouvement est assez rapide pour qu'il n'y ait pas de chevauchement entre deux positions successives de l'objet, ce qui est assez restrictif §4.19.

C'est pourquoi on fait appel à une approche statistique probabiliste pour régulariser le problème qui est initialement mal posé (sous-déterminé). Le principe consiste à introduire une modélisation *a priori* du champ des étiquettes, à l'aide de la théorie des champs aléatoires de Markov, ou en anglais MRF [34]. On contraint alors la solution vers une configuration la plus probable du champ des étiquettes étant donné les observations et le modèle *a priori* [22]. On modélise classiquement le lien statistique entre observation et étiquette en introduisant une fonction  $\Psi$  d'attache aux données :  $o(s) = \Psi(e_s) + g_s$ , où  $g_s$  représente un bruit gaussien centré de variance  $\sigma^2$ . La fonction déterministe  $\Psi$  correspond à l'information pertinente et peut être définie de plusieurs façons [22, 91].

# 2.2 Régularisation statistique markovienne

Cette section présente le concept des champs aléatoires de Markov, pour une application particulière, à savoir la détection de mouvement dans les séquences d'images. Mais cet outil est de portée très générale, et peut s'appliquer à bien d'autres domaines.

### 2.2.1 Fonctions d'énergie

Adoptons les notations suivantes :

- S l'image à l'instant courant t,
- s un pixel (ou plutôt un site) quelconque de S,
- $\eta_s$  un voisinage spatio-temporel de s, par exemple celui défini par la Fig. 2.3,
- r n'importe quel voisin de s (spatial ou temporel),
- C l'ensemble des cliques binaires c = (s, r) constituant les voisinages  $\eta_s$ ,
- $O = \{O_s, s \in S\}$  le champ aléatoire des observations,
- $E = \{E_s, s \in S\}$  le champ aléatoire des étiquettes,
- $o = \{o_s, s \in S\}$  une réalisation particulière du champ d'observations O à l'instant t,
- $e = \{e_s, s \in S\}$  une réalisation particulière du champ d'étiquettes E à l'instant t,
- R l'ensemble des configurations possibles e du champ aléatoire E.

A chaque fois que ce sera nécessaire, on ajoutera un indice temporel pour préciser l'instant considéré : t, t-1,etc.

N.B. : Une clique binaire est une paire de sites mutuellement voisins.

Les interactions spatiales et temporelles entre étiquettes sont modélisées par un MRF, qui constitue le modèle *a priori* du champ des étiquettes.

La propriété essentielle d'un MRF relativement à un voisinage est le caractère local des interactions : la probabilité Pr d'avoir en un pixel s une étiquette  $e_s$  ne dépend que des étiquettes de ses voisins, et non pas de toute l'image.

1. 
$$\forall e \in R, Pr[E = e] > 0$$
  
2.  $Pr[E_s = e_s / E_r = e_r, r \neq s, r \in S] = Pr[E_s = e_s / E_r = e_r, r \in \eta_s]$ 

Cette propriété est cruciale car elle implique des calculs purement locaux et fortement parallélisables, d'où les possibilités de mise en œuvre en temps réel vidéo [32], [46].

Un MRF étant équivalent à une distribution de Gibbs, on peut exprimer la probabilité *a priori* du champ d'étiquettes à l'aide d'une fonction d'énergie :

$$Pr[E = e] = \frac{1}{Z} \exp(-U_m(e))$$
 (2.3)

où Z est une constante de normalisation (appelée fonction de partition) et  $U_m$  une fonction d'énergie associée au modèle *a priori*. Le champ d'étiquettes le plus probable relativement aux observations est



FIGURE 2.3 – Voisinage spatio-temporel et cliques binaires associées.

obtenu par le critère du Maximum A Posteriori (MAP). A l'aide du théorème de Bayes et de l'équation (2.3), on montre que ce critère équivaut à la minimisation d'une fonction d'énergie totale U [58] :

$$\max_{e} \Pr[E = e/O = o] \iff \min_{e} U \quad \text{où} \quad U = U_m(e) + U_a(o, e)$$
(2.4)

 $U_m(e)$  est le terme d'énergie du modèle *a priori* (en anglais *prior*), qui assure la régularisation de la solution. Il s'exprime comme une somme de potentiels énergétiques élémentaires sur les cliques :

$$U_m(e) = \sum_{c \in C} V_c(e_s, e_r)$$
(2.5)

Ces potentiels élémentaires sont définis en fonction du problème à traiter. On peut par exemple prendre des potentiels à niveaux du type :

$$V_c(e_s, e_r) = \begin{cases} -\beta & \text{si } e_s = e_r \\ +\beta & \text{si } e_s \neq e_r \end{cases}$$
(2.6)

ou des potentiels quadratiques du type :

$$V_c(e_s, e_r) = \beta (e_s - e_r)^2$$
(2.7)

où  $\beta > 0$  prend une des trois valeurs  $\beta_s$ ,  $\beta_p$  ou  $\beta_f$  selon la clique considérée (spatiale, passée ou future). Ces potentiels énergétiques favorisent un étiquetage homogène, puisque des étiquettes semblables sur des pixels voisins induisent une moindre contribution énergétique, donc une configuration plus favorable. En pratique, on prend  $\beta_f > \beta_p$ , pour bien éliminer les zones d'écho du mouvement.

 $U_a(o, e)$  est l'énergie d'adéquation, qui assure une bonne attache aux données (en anglais *data link*). Elle traduit le lien entre observations et étiquettes et s'exprime à l'aide d'une fonction  $\Psi$  censée modéliser les observations :

$$U_a(o,e) = \frac{1}{2\sigma^2} \sum_{s \in S} [o_s - \Psi(e_s)]^2$$
(2.8)

où  $\sigma^2$  est la variance des observations et  $\Psi$  est une fonction d'attache aux observations définie en fonction du problème à traiter. On peut par exemple prendre une fonction simple du type :

$$\Psi(e_s) = \begin{cases} 0 & \text{si } e_s = b \text{ (background)} \\ \alpha > 0 & \text{sinon} \end{cases}$$
(2.9)

où  $\alpha$  correspond alors à la valeur moyenne des observations non nulles **§4.20** 

### 2.2.2 Estimation des paramètres

Tous les paramètres qui interviennent dans la modélisation ( $\alpha$ ,  $\beta$ ,  $\sigma$ ) peuvent soit être déterminés de façon empirique après des tests sur des séquences typiques correspondant à l'application envisagée, soit être estimés automatiquement grâce à des algorithmes du type EM ou SEM (*Stochastic Expectation Maximisation*). On peut à ce sujet consulter par exemple les travaux de Pieczynski [12] [114].

### 2.2.3 Algorithmes de relaxation

Pour calculer la configuration d'étiquettes donnant l'énergie minimale, différents algorithmes, dits de relaxation, peuvent être utilisés.

- Les algorithmes de relaxation stochastiques, du type recuit simulé, consistent à explorer l'espace des configurations possibles avec une loi de descente en température adéquate (*i.e.* suffisamment lente) qui assure en principe de converger vers le minimum global de la fonction d'énergie, mais au prix d'un coût de calcul prohibitif.
- C'est pourquoi on leur préfère souvent les algorithmes de relaxation déterministes, du type ICM [16], beaucoup moins lourds en calculs, mais qui ne garantissent pas de converger vers le minimum global d'énergie. Ils risquent de rester piégés dans un minimum local, car ils n'autorisent aucune remontée en température pour sortir d'un minimum local. C'est pourquoi ce type d'algorithme requiert une bonne configuration initiale du champ d'étiquettes (Fig. 2.4).



FIGURE 2.4 – Influence de l'initialisation sur le résultat de l'ICM : évolution de la fonction d'énergie U en fonction de la configuration du champ d'étiquettes  $\mathbf{x} = e \in R$ .

Ces algorithmes déterministes sont récursifs et itératifs : on visite chaque pixel de l'image et on calcule, pour chaque étiquette qu'on peut lui attribuer, la contribution énergétique sur son voisinage. On retient l'étiquette qui donne l'énergie minimale, puis on passe au pixel suivant et ainsi de suite sur toute l'image. On réitère la visite des sites de l'image jusqu'à la convergence de la fonction d'énergie totale U vers une valeur qui n'évolue plus, ou très peu, au cours des itérations. Différentes politiques de visite de sites sont envisageables :

- visite séquentielle classique (*line scanning*) (ligne par ligne de gauche à droite, et de haut en bas image par image),
- visite aléatoire,
- visite chaînée (ligne par ligne alternativement de gauche à droite puis de droite à gauche, et image par image alternativement de haut en bas puis de bas en haut) pour bénéficier au mieux des derniers voisinages mis à jour.

Différentes politiques de mise à jour des sites sont aussi envisageables, le choix dépendant notamment du type de matériel utilisé pour la mise en œuvre :

- pixel-récursif : on met à jour chaque pixel au fur et à mesure de la visite,
- ligne-récursif : on attend d'avoir visité tous les pixels d'une ligne avant de mettre à jour leurs étiquettes,
- image-récursif : on attend d'avoir visité toute une image avant de faire une mise à jour simultanée de toutes les étiquettes de l'image.

De même, différents critères d'arrêt des itérations peuvent être utilisés :

- nombre de pixels instables inférieur à un seuil prédéterminé,
- variation relative d'énergie totale inférieure à un seuil prédéterminé,
- nombre maximum fixe d'itérations (en pratique un faible nombre d'itérations, typiquement 5 à 10 itérations par image, est suffisant). Ce critère est évidemment le moins coûteux pour une mise en œuvre matérielle.

### 2.2.4 Synoptique d'un algorithme de détection de mouvement

Le synoptique d'un algorithme typique est donné Fig. 2.5.



FIGURE 2.5 – Synoptique de l'algorithme de détection de mouvement.

Sur ce schéma-bloc, la notation  $\hat{E}$  représente un champ d'étiquettes initial estimé grossièrement par simple binarisation du champ correspondant d'observations. Pour réaliser la binarisation, on peut soit faire un simple seuillage, soit utiliser des techniques plus robustes de maximum de vraisemblance, avec modèle de luminance (constant, linéaire ou quadratique) sur un voisinage des pixels [65]. Etant donné le voisinage spatio-temporel utilisé pour détecter le mouvement, voisinage qui inclut un voisin futur et un voisin passé pour chaque pixel (Fig. 2.3), le résultat de traitement est obtenu avec un retard d'une image par rapport à l'acquisition, puisqu'il faut disposer de l'image à l'instant t + 1 pour estimer le champ d'étiquettes à l'instant t.

# 2.3 Multirésolution spatio-temporelle

Pour améliorer les performances de l'algorithme dans le cas du mouvement d'objets très lents (mouvement sub-pixel), ou de gros objets peu texturés, on peut envisager d'utiliser une technique du type multirésolution. On présente ici quelques résultats obtenus avec une technique de multirésolution spatio-temporelle qui consiste à filtrer passe-bas et sous-échantillonner la séquence d'images à la fois en espace et en temps selon le schéma de la Fig. 2.6, où est aussi représenté le noyau du filtre 3D appliqué à la séquence.

Un exemple de pyramide spatio-temporelle est donné Fig. 2.7. Le nombre de niveaux m dans la pyramide est fonction notamment de l'amplitude des mouvements présents dans la scène.

Le filtrage spatial permet de renforcer les observations dans le cas de gros objets uniformes (Fig. 2.8).

Quant au filtrage temporel, il permet d'intégrer l'information de mouvement de plusieurs images successives, ce qui est indispensable pour détecter des objets au mouvement très lent (Fig. 2.9).



FIGURE 2.6 – Principe de construction d'une pyramide et noyau du filtre spatio-temporel 3D.



FIGURE 2.7 – Pyramide spatio-temporelle à trois niveaux sur la séquence du présentateur Trevor (de haut en bas, m = 0, 1, 2).

# 2.4 Voisinage 3D spatio-temporel

On peut envisager une structure de voisinage 3D comme présentée Fig. 2.10, plus complexe qu'à la Fig. 2.3. [30].

Si l'on note  $\delta_x, \delta_y, \delta_t$  les coordonnées, dans l'espace 3D (x, y, t), du vecteur représentatif de chaque clique centré sur le pixel courant s, on définit :

- 4 cliques spatiales horizontales et verticales ( $\delta_x$  ou  $\delta_y = \pm 1$ ,  $\delta_t = 0$ );
- 4 cliques spatiales diagonales ( $\delta_x$  et  $\delta_y = \pm 1$ ,  $\delta_t = 0$ );
- 2 cliques temporelles ( $\delta_x$  et  $\delta_y = 0$ ,  $\delta_t = \pm 1$ );
- 8 cliques spatio-temporelles horizontales et verticales ( $\delta_x$  ou  $\delta_y = \pm 1$ ,  $\delta_t = \pm 1$ );
- 8 cliques spatio-temporelles diagonales ( $\delta_x$  et  $\delta_y = \pm 1$ ,  $\delta_t = \pm 1$ ).

Il faut alors modéliser les interactions spatio-temporelles sur toutes les cliques de ce voisinage 3D par des potentiels  $\beta(s, r)$  qui sont fonction du type de clique c = (s, r) considérée :

$$\beta(s,r) = \frac{1}{d^2(s,r) \left[\frac{\delta_x(s,r)^2}{\beta_s} + \frac{\delta_y(s,r)^2}{\beta_s} + \frac{\delta_t(s,r)^2}{\beta_t}\right]}$$
(2.10)

où  $d(s,r) = \sqrt{\delta_x^2 + \delta_y^2 + \delta_t^2}$  est la distance euclidienne entre le pixel courant s et le voisin considéré r.



FIGURE 2.8 – Détection de mouvement multirésolution : de haut en bas : 1) séquence Trevor ; 2) masques mobiles monorésolution ; 3) masques mobiles multirésolution (trois niveaux d'analyse m = 0; 1; 2).



FIGURE 2.9 – Détection de mouvement sous-pixel : de haut en bas : 1) séquence synthétique avec 3 objets mobiles dont l'un a un mouvement sub-pixel ; 2) masques mobiles monorésolution ; 3) masques mobiles multirésolution (deux niveaux d'analyse m = 0; 1).

Cette formule donne :

—  $\beta(s,r) = \beta_s$  pour les cliques spatiales horizontales ou verticales (pour lesquelles d(s,r) = 1);



FIGURE 2.10 – Voisinage spatio-temporel 3D, et cliques binaires associées (pour des raisons de clarté, toutes les cliques possibles ne sont pas représentées).

- $\begin{array}{l} \beta(s,r) = \frac{\beta_s}{4} \text{ pour les cliques spatiales diagonales } (d(s,r) = \sqrt{2}); \\ \beta(s,r) = \beta_t \text{ pour les cliques temporelles } (d(s,r) = 1); \\ \beta(s,r) = \frac{\beta_s\beta_t}{2(\beta_s+\beta_t)} \text{ pour les cliques spatio-temporelles horizontales ou verticales } (d(s,r) = \sqrt{2}); \\ \beta(s,r) = \frac{\beta_s\beta_t}{3(\beta_s+2\beta_t)} \text{ pour les cliques spatio-temporelles diagonales (pour lesquelles } d(s,r) = \sqrt{3}). \\ \end{array}$ Notons que cette définition de  $\beta(s, r)$  ne fait intervenir que deux paramètres  $\beta_s$  et  $\beta_t$ .

Cette variante de l'algorithme permet d'améliorer les performances dans le cas de séquences bruitées, comme illustré sur la Fig. 2.11.



FIGURE 2.11 – Comparaison des deux algorithmes : de haut en bas : 1) séquence synthétique bruitée ; 2) initialisation binaire; 3) masques mobiles avec l'algorithme spatial; 4) masques mobiles avec l'algorithme spatio-temporel 3D.

#### 2.5Mises en œuvre matérielles

#### 2.5.1Adéquation algorithme-architecture

Un algorithme de détection de mouvement se décompose typiquement en deux étapes principales (Fig. 2.5):

— un prétraitement qui calcule les observations et les champs initiaux d'étiquettes,

— un traitement proprement dit qui calcule le minimum d'énergie sur les voisinages spatio-temporels. Le prétraitement, qui consiste simplement en des calculs de différences d'images, de valeurs absolues, et des binarisations par seuillage ne pose pas de problème particulier d'implantation temps réel. En revanche, la minimisation d'énergie est un processus itératif gourmand en calculs et en mémoire, et doit donc faire l'objet d'une mise en œuvre sur une architecture adaptée. Une évaluation grossière montre en effet que la relaxation markovienne compte pour 90% de la charge totale de calcul. On a alors affaire à un problème d'adéquation algorithme-architecture.

#### 2.5.2Solutions envisageables

Plusieurs mises en œuvre sont envisageables [31] :

une implantation purement logicielle en langage de haut niveau (typ. C/C++) : nécessité d'optimisation des pointeurs et des types (integer vs. float...) pour minimiser les transferts et les calculs, utilisation de LUT etc.

L'utilisation de la bibliothèque OpenCV est alors très pratique.

une première solution matérielle, qui prend essentiellement en compte le caractère parallèle des calculs identiques effectués en chaque pixel, consiste à implanter l'algorithme sur une machine parallèle SIMD ou MIMD. L'inconvénient principal de ce type de mise en œuvre est l'encombrement, le coût et les transferts de données.

Une alternative pratique est la programmation sur le coprocesseur graphique de l'ordinateur (GPU qui a une structure hautement parallèle) pour décharger l'unité centrale CPU des calculs lourds.

une seconde solution, qui prend essentiellement en compte le caractère local des calculs sur un petit voisinage spatio-temporel, consiste à s'inspirer du fonctionnement de l'œil et des réseaux de neurones pour implanter l'algorithme sur un circuit résistif analogique en technologie VLSI. Il faut alors développer une analogie électrique du modèle markovien [93]. L'inconvénient est le coût de développement. L'avantage est la taille réduite du circuit. Cette solution est détaillée au paragraphe 2.5.3.

Ce type d'approche débouche sur le concept prometteur de rétine bio-inspirée implantée sur silicium (vision neuromorphique), dont les applications potentielles sont nombreuses, telle la conduite autonome de véhicule [35].

— une troisième solution alternative [45, 29] consiste à implanter l'algorithme sur une carte au format PC à base de DSP et de circuits logiques programmables FPGA (Fig. 2.12). Les avantages sont le coût et l'encombrement réduits. L'inconvénient est le manque de flexibilité.

Une alternative pratique et économique à ce type de solution est l'usage de Raspberry Pi ou d'Arduino.



FIGURE 2.12 – Photo d'une carte d'acquisition et traitement vidéo.

### 2.5.3 Circuit VLSI analogique

Un processus de minimisation d'énergie peut être efficacement implanté matériellement sur un réseau résistif analogique qu'on laisse relaxer jusqu'à son état d'équilibre (dissipation minimale d'énergie). Mais une telle approche nécessite d'adapter le modèle initial pour trouver une bonne analogie électrique avec le comportement d'un réseau résistif.

Le modèle défini dans [99] respecte cette contrainte. Les pixels sont matérialisés par les nœuds interconnectés d'un réseau électrique résistif. On remplace le champ d'étiquettes binaires (a, b) par un champ d'étiquettes continues variant de « 0 » (étiquette b) à « 1 » (étiquette a). Ces étiquettes continues sont alors représentées par des potentiels électriques variant de la tension de masse à la tension d'alimentation du circuit analogique. En utilisant un voisinage d'ordre 1 (4 voisins spatiaux et 2 voisins temporels), des potentiels énergétiques quadratiques (cf. Eq.(2.7)), et une fonction  $\Psi$ judicieuse [99], on montre qu'on aboutit à une expression de l'énergie sous la forme :

$$U = \sum_{i,j} \beta_s (e_{ij} - e_{i+1,j})^2 + \beta_s (e_{ij} - e_{i-1,j})^2 + \beta_s (e_{ij} - e_{i,j+1})^2 + \beta_s (e_{ij} - e_{i,j-1})^2 + \beta_p (e_{ij} - p_{ij})^2 + \beta_f (e_{ij} - f_{ij})^2 + K [o_{ij} - \Psi(e_{ij}, p_{ij})]^2$$
(2.11)

où  $p_{ij}$  et  $f_{ij}$  sont les potentiels électriques des voisins passé et futur,  $e_{ij}$  celui du pixel (i, j) et K une constante.

Le minimum d'énergie correspond à l'annulation des dérivées partielles par rapport à tous les potentiels électriques  $e_{ij}$  en chaque nœud (i, j) du réseau. On obtient alors une équation dont l'analogie électrique est immédiate :

$$\forall ij \quad C\frac{\partial e_{ij}}{\partial t} = \beta_s \nabla^2 e_{ij} + \beta_{pk}(e_{ij} - p_{ij}) + \beta_f(e_{ij} - f_{ij}) + K\alpha(p_{ij} - e_0)o_{ij} \tag{2.12}$$

où  $\nabla^2 e_{ij} = 4e_{ij} - e_{i+1,j} - e_{i-1,j} - e_{i,j+1} - e_{i,j-1}$  est une approximation discrète du laplacien, et où  $\beta_{pk}$  et  $K\alpha$  sont des constantes. La cellule élémentaire associée à chaque site est montrée sur la Fig. 2.13a.



FIGURE 2.13 – a) Cellule analogique élémentaire; b) Architecture du réseau VLSI analogique.

L'idée principale du fonctionnement de la cellule électrique consiste à injecter, ou pomper, du courant en chaque nœud du réseau (grâce à un générateur de courant commandé), pour faire monter, ou baisser, le potentiel électrique du nœud, selon que l'amplitude de l'observation correspondante est forte (pixel mobile) ou faible (pixel fixe).

On peut intégrer sur chaque cellule élémentaire du réseau un photorécepteur, ce qui fait du circuit électrique une caméra « intelligente ». L'architecture du réseau analogique résultant est analogue à celle d'une caméra CCD (Fig. 2.13b). Un exemple typique de résultat est présenté en Fig. 2.14.



FIGURE 2.14 – Simulation électrique. De haut en bas : 1) séquence naturelle avec un vélo et un piéton mobiles ; 2) masques binaires détectés ; 3) potentiels électriques du circuit.

L'avantage d'une telle mise en œuvre matérielle est l'encombrement réduit du système et sa vitesse de traitement (relaxation du réseau en quelques 10 ns). Les inconvénients sont le coût élevé d'un prototype et les problèmes technologiques d'intégration VLSI à résoudre (taille de la cellule, consommation, interconnexions, etc.)

# 2.6 Exemples d'applications

### 2.6.1 Télésurveillance

Une application typique de détection de mouvement est le contrôle du trafic routier [88] ou la télésurveillance à l'entrée d'un site grâce à une caméra « intelligente ». La Fig. 2.15 présente un exemple de résultat de détection automatique du mouvement d'un piéton sur un trottoir, obtenu grâce à la carte à DSP décrite ci-dessus. Notons que l'algorithme détecte également l'ombre du piéton sur le capot de la voiture en stationnement (il ne s'agit pas d'une fausse détection).



FIGURE 2.15 – Séquence Pieton acquise à la cadence de 25 images/s : le mouvement entre deux images est lent ; la convergence du champ d'étiquettes de mouvement est atteinte après 4 itérations seulement.

### 2.6.2 Analyse du mouvement des lèvres d'un locuteur

Il est bien connu que l'homme s'aide d'informations visuelles pour améliorer sa reconnaissance de la parole, notamment dans un environnement bruité. En complément du signal auditif, la vision du mouvement des lèvres du locuteur est une donnée précieuse, qui peut être exploitée pour réaliser un système automatique de reconnaissance de la parole, ou de synthèse de visages parlants [7, 8, 6]. Dans ce but, il faut développer un algorithme de détection automatique du mouvement des lèvres d'un locuteur [95].

Dans le cadre de la compression audio-visuelle en visiophonie, on peut équiper le locuteur d'un casque léger qui comporte, en plus du microphone, une micro-caméra solidaire du crâne et dirigée vers la zone des lèvres. Ainsi, on reste bien dans le cas d'une caméra fixe par rapport au locuteur, et l'on peut donc appliquer les principes de détection de mouvement exposés précédemment, en adaptant les observations et le modèle à l'application envisagée.

Les grandes lignes du traitement sont les suivantes : on acquiert une séquence couleur RVB (rougevert-bleu) du mouvement de la bouche, la région d'intérêt allant de la base du nez au menton. On transforme l'espace RVB en l'espace TLP (teinte, luminance ou intensité, pureté), cf. Eq. (1.79), qui s'avère mieux adapté au problème. En effet :

- la teinte rouge, qui est prédominante sur les lèvres, est une observation spatiale pertinente,
- la pureté permet de s'affranchir des zones d'ombre (car elle est voisine de zéro dans les zones d'ombre),

— enfin, les variations temporelles de luminance donnent l'information de mouvement. On utilise deux types d'information de bas niveau : — une information temporelle sur la différence d'images (frame difference) :

$$fd(s) = I_t(s) - I_{t-1}(s)$$
(2.13)

— une information spatiale sur la teinte rouge (hue):

$$h(s) = \left[256 - \left(\frac{H(s) - H_m}{\sigma}\right)^2\right] \times 1_{P(s) > \delta} \times 1_{|H(s) - H_m| \le 16\sigma}$$
(2.14)

 $H_m$  représente la valeur moyenne de la teinte des lèvres (déterminée au préalable sur la première image),  $\sigma$  est l'écart-type de la teinte des lèvres (valeur empirique, typ.  $4 \leq \sigma \leq 9$ ) et  $\delta$  est un seuil appliqué sur la pureté (typ.  $50 \leq \delta \leq 100$ ). La notation  $1_{condition}$  dénote une fonction binaire qui vaut 1 si la condition est vraie, 0 sinon.

En pratique, l'algorithme utilise trois observations pour chaque site :  $h_{t-1}(s)$ ,  $h_t(s)$  et fd(s). Les Fig. 2.16 et 2.17 illustrent ces champs d'observation sur une séquence typique.



FIGURE 2.16 – De haut en bas : séquence des luminances ; séquence des observations spatiales  $h_t$  à 5 instants t (régions à dominante rouge en blanc,  $\delta = 0$ ).



FIGURE 2.17 – De haut en bas : même séquence de luminances ; séquence des observations temporelles fd (valeurs positives en blanc, négatives en noir, nulles en gris).

La combinaison de ces trois observations permet de définir un jeu de douze étiquettes (Table 2.1). L'étiquette  $c_1$  par exemple correspond à la présence de la teinte rouge des lèvres en t, à l'absence de teinte rouge en t - 1, avec une observation de mouvement détecté positive. On définit alors un modèle énergétique du problème à traiter [95], et on minimise une fonction d'énergie totale faite de cinq termes : trois termes d'attache aux trois observations, un terme d'interaction spatiale et un terme d'interaction temporelle :

$$U = \sum_{s \in S} \left[ \lambda [U_{h_t}(s) + U_{h_{t-1}}(s)] + U_{fd}(s) + U_{sp}(s) + U_{tp}(s) \right]$$
(2.15)

Dans l'Eq. (2.15),  $\lambda$  est un coefficient de pondération sur les énergies d'attache aux observations spatiales. La minimisation de cette fonction d'énergie permet finalement d'extraire de la région d'intérêt les lèvres du locuteur, comme illustré sur la Fig. 2.18.

observations			détection i	nitiale		label			
$h_t(s)$	$h_{t-1}(s)$	fd(s)	teinte	mvt	$r_t(s)$	$r_{t-1}(s)$	m(s)	e(s)	
		$ .  < \theta$		Ø			0	$a_0$	
	$<\gamma$	$> \theta$	Ø	+		0	1	$a_1$	
$<\gamma$		< - heta		—	0		2	$a_2$	
		$ .  < \theta$		Ø			0	$b_0$	
	$>\gamma$	$> \theta$	t-1	+		1	1	$b_1$	
		$< -\theta$		_			2	$b_2$	
		$ .  < \theta$		Ø			0	$c_0$	
	$<\gamma$	$> \theta$	t	+		0	1	$c_1$	
$>\gamma$		$< -\theta$		_	1		2	$c_2$	
		$ .  < \theta$		Ø	]		0	$d_0$	
	$>\gamma$	$> \theta$	t & t - 1	+	]	1	1	$d_1$	
		$< -\theta$		_	]		2	$d_2$	

TABLE 2.1 – Observations, codage bas niveau et les 12 étiquettes (ou labels) initiales correspondantes (typ.  $\theta = 10, \gamma = 100$ ).



FIGURE 2.18 – Champs après relaxation : a) champs M de mouvement ; b) champs  $R_t$  de teinte rouge représentée en noir, pour un seuil  $\delta = 100$ ; c) champs  $E_t$  des 12 étiquettes représentées en niveaux de gris.

### Amélioration de la robustesse

- Pour s'affranchir de la contrainte forte du casque, on peut mettre en œuvre une boucle de rétroaction (inspirée de l'automatique) entre le modèle 3D et les contours actifs [89].
- Pour améliorer la classification de teinte, on peut travailler dans l'espace couleur LUX [79].

### 2.6.3 Remarque conclusive

On a passé en revue un certain nombre de techniques applicables en détection de mouvement 2D dans les séquences d'images.

Mais les outils de traitement présentés ici sont de portée très générale et applicables dans tout problème mal posé portant sur des signaux à deux ou trois dimensions, quelle que soit la nature de ces dimensions, et où il est nécessaire de régulariser la solution en introduisant des contraintes ou connaissances *a priori*.

# 2.7 Estimation de mouvement

L'estimation de mouvement consiste à calculer le déplacement de chaque pixel de l'image ou des régions en mouvement. On obtient alors un champ dense ou épars de vecteurs-vitesse, qu'on appelle flux optique. On distingue trois classes de méthodes :

- 1. les méthodes différentielles correspondent à une approche physique (optique);
- 2. les méthodes fréquentielles sont inspirées de la biologie;
- 3. les méthodes de mise en correspondance sont une approche purement informatique du problème (exploitée notamment dans MPEG);
- 4. les méthodes paramétriques correspondent à une modélisation mathématique du mouvement.

Une bonne revue de ces méthodes est présentée par Barron et al. [5]. Notons que des approches moins classiques permettent également d'estimer le mouvement à partir de la couleur [59].

### 2.7.1 Méthodes différentielles

On fait les hypothèses suivantes :

- déplacements d petits par rapport à la taille des objets en mouvement (typ. vitesse de l'ordre de 1 pixel/trame),
- invariance de l'intensité lumineuse I des objets au cours du temps.

La deuxième hypothèse s'exprime classiquement avec la DFD calculée en chaque pixel p:

$$DFD(p,d) = I(x + dx, y + dy, t + dt) - I(x, y, t) = 0$$
(2.16)

En pratique, on cherche à minimiser la DFD. Pour cela, on considère le développement en série de Taylor de la DFD au premier ordre, qui aboutit à l'équation de contrainte du mouvement (ECM) exprimant la relation entre les 3 composantes du gradient spatio-temporel de l'intensité lumineuse et les 2 composantes du vecteur vitesse  $[\S4.21]$ :

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$
(2.17)

Comme on a une seule équation pour deux inconnues  $u = \frac{dx}{dt}$  et  $v = \frac{dy}{dt}$ , le problème est sous-déterminé. Ceci constitue le fameux problème d'ouverture (*aperture problem* en anglais) : si l'on observe la scène à travers une toute petite ouverture<sup>1</sup>, on ne perçoit que la composante orthogonale au contour (Fig. 2.19).



FIGURE 2.19 – Le problème d'ouverture (cas d'un barreau mobile en diagonale).

Il faut donc introduire une contrainte supplémentaire, dite contrainte de lissage, pour lever l'indétermination. On ajoute donc une hypothèse d'homogénéité du champ de vitesses dans un voisinage (petit domaine D).

<sup>1.</sup> Il n'est effectivement pas correct de regarder à travers le trou d'une serrure !

Horn et Schunck [64] adoptent un terme de lissage portant sur la somme des carrés des modules des gradients des composantes de vitesse, conduisant à la minimisation de la formule :

$$\iint_{D} \left(\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t}\right)^{2} + \alpha^{2} \left[ \left(\frac{\partial u}{\partial x}\right)^{2} + \left(\frac{\partial u}{\partial y}\right)^{2} + \left(\frac{\partial v}{\partial x}\right)^{2} + \left(\frac{\partial v}{\partial y}\right)^{2} \right] dx dy$$
(2.18)

où  $\alpha$  est un coefficient de pondération du terme de lissage.

Un algorithme itératif permet de minimiser cette intégrale §4.57 :

$$u^{k+1} = \overline{u}^{k} - \frac{I_{x}[I_{x}\overline{u}^{k} + I_{y}\overline{v}^{k} + I_{t}]}{\alpha^{2} + I_{x}^{2} + I_{y}^{2}}$$
$$v^{k+1} = \overline{v}^{k} - \frac{I_{y}[I_{x}\overline{u}^{k} + I_{y}\overline{v}^{k} + I_{t}]}{\alpha^{2} + I_{x}^{2} + I_{y}^{2}}$$
(2.19)

où k représente l'indice d'itération (conditions initiales  $u^0$  et  $v^0$  nulles),  $I_x, I_y, I_t$  étant les composantes du gradient spatio-temporel selon x, y et t, et  $\overline{u}, \overline{v}$  les valeurs moyennes des 2 composantes de la vitesse, calculées sur un voisinage  $3 \times 3$  pondéré (Fig. 2.20b).

Nagel propose une contrainte de lissage orientée qui tient compte de la direction du gradient, pour éviter de lisser sur les frontières des objets en mouvement [108]. Son calcul repose sur l'usage des dérivées secondes. Le terme de lissage de Nagel (Eq.(44b) dans [107]) est plus compliqué que celui de Horn & Schunck (terme entre crochets dans l'Eq. (2.18)). Il introduit une constante supplémentaire  $\gamma$ :

$$\frac{\alpha^2}{I_x^2 + I_y^2 + 2\gamma} \left[ (u_x I_y - u_y I_x)^2 + (v_x I_y - v_y I_x)^2 + \gamma (u_x^2 + u_y^2 + v_x^2 + v_y^2) \right]$$
(2.20)

où  $\gamma$  est un paramètre de conditionnement, empêchant l'annulation du dénominateur, et pondérant les dérivées partielles des composantes du vecteur vitesse, notées  $u_x, u_y, v_x, v_y$ . Un algorithme itératif permet là aussi d'obtenir la solution correspondant au minimum de la formule.

Les méthodes différentielles reposant sur des calculs de dérivées partielles, la qualité de l'estimation est fortement dépendante des formules choisies pour ces calculs. Comme un calcul de dérivée est sensible au bruit, un pré-filtrage passe-bas spatio-temporel est souvent nécessaire. Horn et Schunck utilisent un masque de calcul de dérivée simple [1 - 1], avec un moyennage dans le cube de 8 pixels (x, x + 1, y, y + 1, t, t + 1) défini par la Fig. 2.20a [§4.23].



FIGURE 2.20 – a) Voisinage cubique; b) Masque de calcul des vitesses moyennes.

Barron et al. proposent à la place un masque  $\frac{1}{12}$  [-1 8 0 -8 1], avec un préfiltrage spatio-temporel gaussien.

D'autres auteurs ont proposé des variantes de calcul du flux optique par méthode différentielle avec l'ECM, notamment Lucas et Kanade [83], dont l'algorithme est disponible dans la bibliothèque OpenCV. Ils supposent la vitesse des pixels constante sur une petite fenêtre, et appliquent une technique des moindres carrés pour l'estimation. Un exemple d'application au suivi de la nage de poissons en bassin d'études est présenté dans [48].

### 2.7.2 Méthodes fréquentielles

#### Méthode basée sur l'énergie

Considérons le cas d'un mouvement mono-dimensionnel. La fréquence (ou pulsation) spatiale  $\omega_x$ d'une sinusoïde en mouvement s'exprime en cycles/pixel alors que le fréquence temporelle  $\omega_t$  s'exprime en cycles/unité de temps, c'est-à-dire en cycles/trame, puisque l'unité de temps en vidéo est l'intervalle temporel séparant deux trames consécutives de la séquence. La vélocité ou vitesse, qui est la distance sur le temps, s'exprime en pixels/trame et vaut donc :  $u = \frac{\omega_t}{\omega_x}$  d'où l'on a la relation :  $\omega_t = u\omega_x$ .

De façon analogue, une texture 2D translatant dans une image occupe un plan dans le domaine des fréquences spatio-temporelles :

$$u\omega_x + v\omega_y = \omega_t \tag{2.21}$$

C'est l'équation de contrainte du mouvement dans le domaine fréquentiel §4.22. En estimant l'orientation de ce plan dans le volume fréquentiel 3D, on peut en déduire les composantes u, v de la vitesse. La technique consiste donc à paver le domaine fréquentiel avec une batterie de filtres passe-bande (Fig. 2.21). Là où il y a de l'énergie, la réponse du filtre correspondant est forte, ce qui permet de localiser le plan et d'estimer son orientation, d'où les vitesses.



FIGURE 2.21 – Plan fréquentiel dans le volume 3D pavé de filtres.

Heeger [63] propose d'utiliser des filtres de Gabor 3D :

$$g(x,y,t) = \frac{1}{\sqrt{2\pi^{3/2}\sigma_x\sigma_y\sigma_t}} \exp\left[-\left(\frac{x^2}{2\sigma_x^2} + \frac{y^2}{2\sigma_y^2} + \frac{t^2}{2\sigma_t^2}\right)\right] \times \sin\left(2\pi\omega_{x_0}x + 2\pi\omega_{y_0}y + 2\pi\omega_{t_0}t\right) \quad (2.22)$$

où  $(\omega_{x_0}, \omega_{y_0}, \omega_{t_0})$  est la fréquence centrale du filtre et  $(\sigma_x, \sigma_y, \sigma_t)$  l'extension de la fenêtre gaussienne spatio-temporelle.

L'inconvénient de cette technique est son coût de calculs, lié au nombre de filtres nécessaires (12) et à la taille des masques de convolution (taille 23 pour les filtres spatiaux, et taille 7 en temporel). Mais on peut utiliser la séparabilité des filtres 3D pour réduire le coût des convolutions.

### Méthode basée sur la phase

La règle de translation de la TF s'exprime par :

$$TF[f(x)] = F(\nu) \Rightarrow TF[f(x - x_0)] = F(\nu) \exp(-i2\pi\nu x_0)$$
 (2.23)

Cette règle peut être exploitée pour estimer un mouvement de translation 1D dans une image à partir de l'étude de la phase. Hélas, dans le cas 2D, la mesure du déphasage donne une information sur la somme du déplacement en x et en y. Pour désimbriquer l'information sur la translation horizontale et verticale, on peut utiliser une approche basée sur les nombres hypercomplexes (espace des quaternions).

### 2.7.3 Mise en correspondance de blocs

La mise en correspondance de régions consiste à rechercher, dans deux images successives, la meilleure correspondance en maximisant une mesure de similarité comme l'inter-corrélation, ou en minimisant une mesure de distance comme la somme des différences au carré entre deux blocs, en anglais SSD :

$$SSD = \sum_{j=-n}^{n} \sum_{i=-n}^{n} W(i,j) \times [I_1(x+i,y+j) - I_2(x+i+dx,y+j+dy)]^2$$
(2.24)

où W est une fonction de pondération, d = (dx, dy) est le déplacement estimé, et  $I_1$ ,  $I_2$  sont deux images à deux instants successifs (Fig. 2.22).



FIGURE 2.22 – Mise en correspondance d'un bloc  $3 \times 3$ .

A la place de la SSD, on peut utiliser la somme des différences en valeurs absolues, SAD, critère moins coûteux en calculs §4.24.

On associe souvent ce type de technique avec une approche pyramidale *coarse-to-fine*, car une recherche exhaustive à pleine résolution est très coûteuse. Anandan [2] utilise par exemple une pyramide laplacienne avec 2 ou 3 niveaux, ce qui permet le calcul de grands déplacements en commençant par une mise en correspondance au niveau le plus grossier de la pyramide. Cela permet aussi de rehausser les structures importantes (contours) dans l'image.

La mise en correspondance de blocs est très utilisée pour le codage vidéo (compression MPEG). L'artefact classique est celui des effets de blocs perceptibles à la restitution. Ceci est dû aux hypothèses limitatives de cette approche : le découpage en blocs et leur taille (typ.  $8 \times 8$  ou  $16 \times 16$ ) est arbitraire, les mouvements complexes (autres qu'une simple translation) et les grands déplacements sont mal pris en compte. C'est pourquoi des variantes (approches hiérarchiques ou par transformées spatiales) sont proposées dans la littérature [82].

### 2.7.4 Modèles paramétriques de mouvement

Au lieu d'une contrainte de lissage, on peut utiliser un modèle mathématique explicite de mouvement. On base alors l'estimation de mouvement sur une modélisation polynomiale du déplacement. Les paramètres du modèle sont déterminés par une technique de type moindres carrés ou moindres carrés robustes. Il existe une hiérarchie de modèles de complexité croissante §4.25:

$$- \text{ modèle translationnel}: \begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} a_1 \\ a_4 \end{bmatrix}$$
$$- \text{ translation + rotation}: \begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} a_1 + a_3y \\ a_4 - a_3x \end{bmatrix}$$
$$- \text{ translation + rotation + divergence}: \begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} a_1 + a_2x + a_3y \\ a_4 - a_3x + a_2y \end{bmatrix}$$
$$- \text{ modèle affine complet}: \begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} a_1 + a_2x + a_3y \\ a_4 + a_5x + a_6y \end{bmatrix}$$
$$- \text{ modèle quadratique}: \begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} a_1 + a_2x + a_3y + a_7x^2 + a_8xy \\ a_4 + a_5x + a_6y + a_7xy + a_8y^2 \end{bmatrix}$$
$$- \text{ modèle panoramique}: \begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} a_1 + a_1x^2 + a_4xy \\ a_4 + a_1xy + a_4y^2 \end{bmatrix}$$

Le modèle affine est un bon compromis entre représentativité et complexité. On recherche le vecteur des paramètres  $\Theta = (a_1, a_2, a_3, a_4, a_5, a_6)$  qui minimise, sur un domaine D, un critère défini par une fonction de coût :

$$\Theta = \arg\min_{\theta} \sum_{D} \rho(r) \tag{2.25}$$

où r est le résidu pour chaque pixel du domaine (typ. r = DFD(p, d)), et où  $\rho$  est un estimateur qui peut être quadratique ( $\rho(r) = r^2$ ), auquel cas on réalise une estimation classique aux moindres carrés, ou un estimateur dit robuste (du type M-estimateur) §4.26. En effet, une bonne fonction  $\rho$  doit être définie positive, mais ne pas croître indéfiniment lorsque  $r \to \infty$ , afin de limiter l'influence des points aberrants. Un exemple classique est l'estimateur biweight de Tukey (Fig. 2.23) défini par :

$$\rho(r) = \begin{cases} \frac{r^6}{6} - \frac{C^2 r^4}{2} + \frac{C^4 r^2}{2} \text{ si} : |r| < C \\ \frac{C^6}{6} \text{ sinon} \end{cases}$$
(2.26)



FIGURE 2.23 – Estimateur robuste de Tukey pour C = 2.

# 2.8 Compensation de mouvement

### 2.8.1 Introduction

Les méthodes de compression de séquences d'images pour la transmission à bas débit (ou pour le stockage vidéo peu encombrant) reposent sur une prédiction par compensation de mouvement. Nous

décrivons ici l'approche classique d'estimation de mouvement proposée par Netravali [110] et améliorée par Walker-Rao [131] et Biemond [17].

L'algorithme d'estimation, basé sur une méthode différentielle, est pixel-récursif et itératif. L'hypothèse de travail étant l'invariance de la luminance, le champ de déplacement est estimé en minimisant la DFD en chaque pixel ou sur un voisinage causal constitué de N pixels incluant le pixel courant, selon une technique du type descente de gradient ou estimateur de Wiener. Ceci donne une image prédite à partir de l'image passée et du champ de déplacement estimé **§4.31**.

### 2.8.2 Estimation de mouvement pel-récursive

Les techniques pel-récursives (*i.e.* pixel-récursives) d'estimation de mouvement développées pour le codage ont pour objectif la reconstruction de l'image. En ce sens, elles ne visent pas forcément la meilleure estimation du déplacement réel des objets dans la scène filmée, mais la minimisation d'une erreur de prédiction d'une image à l'autre. La méthode introduite par Netravali *et al.* [110], pour le codage de séquences télévisuelles par compensation de mouvement, consiste à estimer en chaque point p de l'image le déplacement  $\hat{d}$  qui minimise le critère de prédiction donné par l'équation (2.27) où la DFD est la différence d'image déplacée définie par l'équation (2.28), similaire à l'Eq. (2.16) de la p.51 :

$$\hat{d} = \arg\min_{d} \{ DFD^2(p, d) \}$$
(2.27)

$$DFD(p,d) = I(p,t) - I(p-d,t-1)$$
(2.28)

La solution proposée par Netravali utilise la technique numérique et itérative de descente de gradient :

$$\hat{d}^i = \hat{d}^{i-1} - T_c \tag{2.29}$$

où 
$$T_c = \varepsilon.DFD(p, \hat{d}^{i-1}) \cdot \nabla I(p - \hat{d}^{i-1}, t-1)$$
 (2.30)

 $T_c$  est le terme de correction appliqué pour calculer  $\hat{d}^i$ , déplacement estimé à la i-ème itération,  $\varepsilon$  est un gain constant positif réglant la vitesse de convergence et  $\nabla I(p - \hat{d}^{i-1}, t - 1)$  est le gradient spatial.

Pour améliorer les performances de l'estimateur, Walker *et al.* proposent d'utiliser un gain adaptatif [131] :

$$\varepsilon = \frac{1}{2.|\nabla I(p - \hat{d}^{i-1}, t - 1)|^2}$$
(2.31)

A chaque itération, plusieurs tests sont effectués :

- nullité du gradient, auquel cas on ne fait pas d'itération,
- limitation du terme de correction  $T_c$  (bornes supérieure et inférieure),
- arrêt par compensation (seuil minimum atteint sur la DFD),
- arrêt par dépassement du nombre maximal d'itérations.

Pour améliorer la convergence de l'algorithme, on peut choisir pour valeur initiale  $d^0$  du déplacement en le pixel courant p, la meilleure valeur parmi celles estimées précédemment dans un voisinage causal du pixel, c'est-à-dire celle qui minimise la DFD en ce pixel p. Un voisinage couramment utilisé est celui de la Fig. 2.24(b).

 $\underline{\text{Légende}:} \quad \bullet : \text{pixel courant } p \quad \circ : \text{pixels voisins}
 \underbrace{\begin{array}{c} \circ & \circ & \circ & \circ \\ \circ & \bullet & \circ & \circ & \circ \\ \circ & \bullet & \circ & \circ & \circ \\ (a) & (b) & (c) \end{array}$ 

FIGURE 2.24 – Différents voisinages spatiaux causaux.

Les pixels dans l'image sont parcourus de gauche à droite et de haut en bas (balayage de télévision). Pour reconstruire l'image à l'instant t, on ajoute à chaque pixel de l'image à l'instant t - 1 le vecteur déplacement estimé qui lui correspond. Le critère classiquement utilisé pour évaluer la qualité de la reconstruction est le rapport signal à bruit crête ( $Peak \ SNR$ ) défini par :

$$PSNR_{dB} = 10\log_{10}\left(\frac{(I_{max} - I_{min})^2}{EQM}\right)$$
(2.32)

avec 
$$EQM = \frac{1}{LC} \sum_{p} \left( I(p,t) - \hat{I}(p,t) \right)^2$$
(2.33)

où  $I_{max} - I_{min} = 255$  pour des images codées sur 8 bits et où  $\hat{I}(p,t)$  est l'image prédite par compensation, L et C étant le nombre de lignes et de colonnes de l'image, et EQM signifiant erreur quadratique moyenne.

Pour des séquences bruitées, l'algorithme de Walker-Rao est peu robuste. Une estimation du déplacement tenant compte d'un voisinage du pixel courant améliore la robustesse. C'est la méthode proposée par Biemond *et al.* [17]. Pour N voisins, l'équation (2.28) développée en série de Taylor conduit à un système matriciel :

$$\begin{bmatrix} DFD(p_1, d^{i-1}) \\ \vdots \\ DFD(p_N, d^{i-1}) \end{bmatrix} = -\begin{bmatrix} g_x^1 & g_y^1 \\ \vdots & \vdots \\ g_x^N & g_y^N \end{bmatrix} . [d - d^{i-1}] + \begin{bmatrix} \nu(p_1, d^{i-1}) \\ \vdots \\ \nu(p_N, d^{i-1}) \end{bmatrix}$$
(2.34)

où  $g_x^j$  et  $g_y^j$  sont les composantes du gradient spatial  $\nabla I(p_j - \hat{d}^{i-1}, t-1)$  en le pixel  $p_j - d^{i-1}$  à l'instant t-1, et où  $\nu(p_j, d^{i-1})$  est l'erreur de développement.

Avec les notations de [17], l'Eq. (2.34) peut s'écrire :  $z = G.u + \nu$ . On cherche alors une estimation linéaire de u à partir de z, ce qui revient à trouver la matrice pseudo-inverse L telle que :  $\hat{u} = L.z$ , avec l'espérance mathématique  $E \{ ||u - \hat{u}||^2 \}$  minimum. La solution, obtenue par estimateur de Wiener, est établie par Biemond, moyennant quelques hypothèses simplificatrices :

$$d^{i} = d^{i-1} - \begin{bmatrix} \sum_{j=1}^{N} g_{x}^{2}(j) + \mu & \sum_{j=1}^{N} g_{x}(j) \cdot g_{y}(j) \\ \sum_{j=1}^{N} g_{x}(j) \cdot g_{y}(j) & \sum_{j=1}^{N} g_{y}^{2}(j) + \mu \end{bmatrix}^{-1} \cdot \begin{bmatrix} \sum_{j=1}^{N} g_{x}(j) \cdot DFD(p_{j}, d^{i-1}) \\ \sum_{j=1}^{N} g_{y}(j) \cdot DFD(p_{j}, d^{i-1}) \end{bmatrix}$$
(2.35)

où  $\mu = \sigma_{\nu}^2 / \sigma_u^2$  est le rapport des variances de u et  $\nu$ . Un  $\mu$  faible a un effet de lissage du champ de vecteurs estimé et assure la stabilité numérique (dénominateur non nul).

Plusieurs choix sont possibles pour le voisinage à N pixels, à condition qu'il reste causal (Fig. 2.24). Choisir N = 1 ramène à une formule proche de celle de Walker-Rao. L'algorithme de Biemond converge plus vite que celui de Walker-Rao. Il donne de meilleurs résultats lorsque l'hypothèse de mouvement uniforme est vérifiée pour tous les points du voisinage (typiquement à l'intérieur d'un objet mobile), mais devient moins efficace dans les zones de discontinuité de mouvement (frontière d'un objet mobile) où l'hypothèse de mouvement uniforme dans le voisinage n'est plus respectée. Pour résoudre ce handicap, Baaziz [3] propose une version combinée des deux algorithmes : l'idée consiste à faire itérer l'algorithme de Walker-Rao dans les cas où l'algorithme de Biemond n'a pas convergé. On bénéficie ainsi des avantages respectifs des deux algorithmes : robustesse au bruit pour l'algorithme de Biemond, calcul ponctuel adapté aux frontières pour l'algorithme de Walker. Cette démarche revient en fait à introduire un voisinage adaptatif constitué de N voisins pour l'algorithme de Biemond, ou d'un seul voisin pour l'algorithme de Walker-Rao. Mais quel que soit l'algorithme choisi, seuls de petits déplacements peuvent être estimés. Dans le cas de déplacements importants, il est nécessaire d'envisager une technique multirésolution avec une stratégie de type *coarse to fine*. La transformée en ondelettes est un des choix possibles. Elle est abordée dans le chapitre 2.9.1.

### 2.8.3 Principe du codage vidéo

A partir de l'estimation de mouvement et de la compensation de mouvement présentées précédemment, on peut concevoir une méthode de codage vidéo exploitant l'information de mouvement pour prédire les images successives d'une séquence. Le gain en compression résulte de l'idée simple qu'il suffit de ne transmettre que l'information sur ce qui a bougé dans la scène, plutôt que de systématiquement tout transmettre à chaque image (dont les pixels du fond fixe qui n'ont pourtant pas changé d'une image à la suivante).

La Fig. 2.25 présente le schéma de principe d'un Codec vidéo basé sur l'estimation de mouvement. On y revient au chapitre 3.6 qui aborde le standard MPEG.



FIGURE 2.25 – Schéma de principe d'un codec vidéo avec estimation de mouvement ( $\varepsilon$  = erreur de prédiction; v = vecteurs de mouvement); Q = quantification;  $z^{-1}$  = opérateur retard.

# 2.9 Méthodes hybrides de compression vidéo

Les méthodes hybrides de compression de séquences d'images, qui reposent conjointement sur une prédiction par compensation de mouvement et sur l'utilisation d'une transformée (par exemple une transformée de Fourier DCT, ou une transformée en ondelettes DWT des images), présentent un grand intérêt pour la transmission à bas débit.

Pour estimer des déplacements importants et/ou accélérer la convergence de l'algorithme, il est intéressant d'intégrer cette estimation de mouvement dans une approche multirésolution : à un niveau de résolution spatiale grossière, on estime les déplacements importants, puis on propage l'information vers des niveaux de résolution plus fine.

### 2.9.1 Transformée en ondelettes

La transformée en ondelettes appliquée aux images présente plusieurs avantages : analyse multirésolution, bonne localisation spatio-fréquentielle, non redondance entre les sous-bandes [102]. Elle offre une structure hiérarchique utile pour l'interprétation de l'image. Dans le cas de l'estimation de mouvement, la stratégie *coarse to fine* est intéressante pour estimer correctement les déplacements importants. Si l'on considère une suite dyadique de résolutions, alors l'amplitude du déplacement sera divisée par 2 à chaque niveau. Or l'estimation de petits déplacements est plus aisée. De plus, on peut utiliser l'estimation à un niveau de résolution grossier comme une bonne initialisation du niveau de résolution plus fin, ce qui permet d'accélérer la convergence des algorithmes.

Notons m l'indice correspondant au niveau de résolution; m = 0 correspond à l'image initiale pleine résolution, m croissant quand la résolution décroît. L'analyse par transformée en ondelettes permet d'approximer un signal f de  $L^2(\mathbb{R})$ , à différents niveaux de résolution, par sa tendance (ou approximation) au niveau m que l'on notera  $A_m \bullet f$ , et d'extraire l'information de différence contenue entre 2 niveaux (les fluctuations). Notons  $V_m$  le sous-espace vectoriel contenant l'ensemble de toutes les approximations possibles de fonctions au niveau m, et  $W_m$  son complémentaire orthogonal dans l'espace englobant  $V_{m-1}$ . On a donc :  $V_{m-1} = V_m \oplus W_m$ , où  $\oplus$  dénote la somme orthogonale.

 $A_m$  est l'opérateur de projection orthogonale sur  $V_m$ . Il a été montré qu'il existe une unique fonction  $\phi(x)$ , appelée **fonction d'échelle**, telle que les fonctions  $\phi_{m,n} = \sqrt{2^{-m}}\phi(2^{-m}x - n)_{n\in\mathbb{Z}}$  forment une base orthonormée de  $V_m$ . L'analyse consiste à décomposer le signal sur une base orthonormée d'ondelettes  $\psi_{m,n}$  de  $L^2(\mathbb{R})$ , obtenues par translation et dilatation d'une unique fonction  $\psi(x)$  appellée **ondelette mère** :

$$\psi_{m,n} = \sqrt{2^{-m}} \psi (2^{-m} x - n)_{n \in \mathbb{Z}}$$
(2.36)

On peut alors exprimer la transformation en ondelettes comme une projection du signal f sur ces différents sous-espaces :

$$A_{m-1} \bullet f = A_m \bullet f + \sum_{n \in \mathbb{Z}} \langle f, \psi_{m,n} \rangle \psi_{m,n}$$
(2.37)

Cette décomposition peut s'obtenir simplement par des convolutions du signal avec des filtres miroirs en quadrature QMF, comme l'a montré Daubechies [41]. A chaque niveau, on filtre le signal par des filtres passe-haut et passe-bas et on le sous-échantillonne. Tendance et fluctuations au niveau m ont donc un support deux fois moindre qu'au niveau m - 1, ce qui permet de ne pas augmenter le volume des données. Pour les images, on peut utiliser les filtres de Daubechies à 4 coefficients :

$$c_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}; \quad c_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}; \quad c_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}; \quad c_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$$
(2.38)

La réponse impulsionnelle du filtre passe-bas B donnant la tendance est composée des coefficients  $c_0$ ,  $c_1$ ,  $c_2$ ,  $c_3$ , tandis que celle du filtre passe-haut H donnant la fluctuation est composée des coefficients  $c_3$ ,  $-c_2$ ,  $c_1$ ,  $-c_0$ , puisqu'on a la relation :  $H_n = (-1)^n B_{-n+1}$  [41]. Les ondelettes de Daubechies ont l'avantage d'avoir un support compact, ce qui évite des calculs trop lourds.

Une image étant représentée par une matrice à 2 dimensions, la décomposition en tendance et fluctuations se fait selon 2 axes. On obtient 4 matrices, chacune de taille 4 fois plus petite que l'image initiale, que l'on dénote BB, BH, HB et HH, H correspondant au filtrage passe-haut et B au filtrage passe-bas appliqués de façon séparable sur les lignes et les colonnes. La première lettre correspond au filtrage en x (selon les colonnes), et la seconde au filtrage en y (selon les lignes).

### 2.9.2 Ondelettes et compensation de mouvement

Une mise en œuvre de l'estimation de mouvement couplée aux ondelettes est proposée par Baaziz dans [3]. Elle utilise les ondelettes dyadiques de Daubechies à 4 coefficients [41], en s'arrêtant au premier niveau de décomposition (4 sous-bandes).

La Fig. 2.26 schématise la décomposition d'une image en 4 sous-bandes, et montre aussi les pixels intervenant dans le voisinage multirésolution utilisé pour appliquer l'algorithme d'estimation de mouvement de Biemond.

BB	HB
○ ●	0
RH	НН
DII	1111
0	0

FIGURE 2.26 – Les 4 sous-bandes d'une image transformée en ondelettes, avec le voisinage multirésolution (pixel courant en noir et ses 4 voisins en blanc).

La sous-bande BB étant celle qui contient le plus d'information (entropie la plus élevée), il est proposé d'appliquer l'algorithme itératif d'estimation uniquement dans cette bande, mais en utilisant l'information de voisinage des autres bandes. Le champ de déplacement obtenu sert à compenser chacune des bandes de fréquence, c'est-à-dire que les coefficients des 4 sous-bandes de l'instant t - 1sont tous translatés par le vecteur déplacement qui leur correspond. A partir de ces 4 sous-bandes compensées, on effectue une transformation en ondelettes inverse pour obtenir l'estimée de l'image à l'instant t.

Cette méthode repose donc sur l'hypothèse de cohérence du mouvement dans toutes les sous-bandes.

### 2.9.3 Cohérence du mouvement entre sous-bandes ?

La spécificité de la décomposition en ondelettes doit être prise en compte pour une utilisation correcte dans un schéma de compensation de mouvement. L'hypothèse de cohérence du mouvement dans toutes les sous-bandes est à considérer de près. Les deux simulations ci-après montrent que l'on ne peut pas compenser toutes les sous-bandes de façon simple avec le même champ de déplacement.

En effet, pour une pyramide d'ondelettes discrètes dyadiques, une translation de l'objet dans l'image initiale (niveau de résolution m = 0) ne correspondra à une translation des coefficients de fluctuation dans la sous-bande de niveau de résolution m que si le déplacement dans l'image initiale est un multiple de  $2^m$ . Dans le cas contraire, il n'y a pas identité entre coefficients passés translatés et coefficients présents.

La figure 2.27 illustre la décomposition en ondelettes sur une séquence d'images synthétiques comportant un carré noir se déplaçant horizontalement vers la droite de 1 pixel/image.



FIGURE 2.27 – (haut) Transformée en ondelettes en 4 sous-bandes; (bas) Séquence synthétique originale.

On constate évidemment que les coefficients d'ondelettes des fluctuations HF sont non nuls uniquement au voisinage des contours de l'objet (le fond gris dans les sous-bandes HB, BH et HH correspond à des coefficients nuls, le noir à des coefficients négatifs et le blanc à des coefficients positifs).

Mais il est surtout intéressant de noter l'évolution du signe des coefficients sur cet exemple simple : alors que les coefficients des contours horizontaux gardent le même signe d'une image à la suivante (contours toujours noirs ou toujours blancs), les coefficients des contours verticaux, qui sont perpendiculaires au sens du mouvement, alternent de signe à chaque image : noir, blanc, noir, blanc ...

Ce phénomène invalide l'hypothèse de cohérence du mouvement dans les sous-bandes.

Voyons dans quel cas une translation de  $\Delta x$  dans le signal original discret correspond à une translation de  $\Delta n \in Z$  dans les coefficients, c'est-à-dire pour quelles valeurs de  $\Delta x$  l'égalité (2.39) est vérifiée :

$$\langle f(x + \Delta x), \psi_{m,n}(x) \rangle = \langle f(x), \psi_{m,n+\Delta n}(x) \rangle$$
(2.39)

On a : 
$$\int_{-\infty}^{+\infty} f(x + \Delta x)\psi_{m,n}(x)dx = \int_{-\infty}^{+\infty} f(x + \Delta x)2^{-\frac{m}{2}}\psi(2^{-m}x - n)dx$$

Avec le changement de variable  $y = x + \Delta x$ , on obtient :

$$\int_{-\infty}^{+\infty} f(y) 2^{-\frac{m}{2}} \psi(2^{-m}(y-\Delta x)-n) dy = \int_{-\infty}^{+\infty} f(y) 2^{-\frac{m}{2}} \psi\left(2^{-m}y-(2^{-m}\Delta x+n)\right) dy$$

Donc la relation (2.39) n'est vérifiée que si le déplacement de l'objet est un multiple de  $2^m$ .

En effet, il faut avoir :  $2^{-m}\Delta x = \Delta n \in Z \Leftrightarrow \Delta x = 2^{m}\Delta n$ . Sinon, un déplacement dans le signal ne se traduit pas par un simple déplacement dans les coefficients d'ondelettes. Ce résultat correspond en fait à la non-invariance par translation que Mallat avait déjà signalée dans [102]. L'hypothèse de cohérence du mouvement dans toutes les sous-bandes n'est donc pas valable dans le cas général. Le problème résulte de la discrétisation des ondelettes ( $\Delta n \in Z$ ).

D'autre part, l'algorithme de Biemond, employé avec le voisinage multirésolution de la Fig. 2.26, nécessite le calcul des gradients spatiaux dans les 4 sous-bandes. Or la différence importante de dynamique des coefficients d'ondelettes dans les différentes sous-bandes (tendance et fluctuations correspondant respectivement aux composantes BF et HF de l'image) peut poser problème pour ce calcul de gradient. De plus, la nature des coefficients est différente : alors que les coefficients d'ondelettes sont des réels signés, ceux de la tendance sont du même type que l'image originale, c'est-à-dire des entiers non-signés. Une « normalisation » est donc nécessaire si l'on veut mêler dans un même calcul des « voisins » multirésolution si différents. Le voisinage multirésolution de la Fig. 2.26 pose donc un problème d'inhomogénéité lié à la nature différente des voisins pris en compte : intensités lumineuses ou coefficients d'ondelettes HF ou BF.

La Fig. 2.28 illustre ces remarques sur un cas d'école en 1D : on a simulé un créneau lissé (par 2 convolutions avec un filtre binôminal) et on a calculé tendance et fluctuation pour différentes positions du créneau (le déplacement étant un multiple de l'unité).



FIGURE 2.28 – Simulation du mouvement d'un créneau dans un signal 1D, présentant de haut en bas : le signal, la tendance, les fluctuations.

On constate la différence de dynamique entre la tendance et les fluctuations, ainsi qu'un phénomène « stroboscopique » typique de l'influence de l'échantillonnage sur les fluctuations : on ne retrouve le motif de fluctuation identique au cas du signal non décalé de départ que pour un décalage multiple de 2, alors que pour un décalage impair (1 ou 3), le motif est notablement différent, ce qui est évidemment gênant pour un calcul ultérieur de gradient.

On conclut que l'hypothèse de cohérence du mouvement entre les 4 sous-bandes suppose le respect de conditions restrictives à la fois sur la nature des discontinuités spatiales dans l'image (amplitude des contrastes, absence de fronts raides pour respecter le théorème de Shannon), mais aussi sur l'amplitude du mouvement des objets dans l'image, conditions rarement maîtrisables en pratique.

### 2.9.4 Autres approches

La spécificité des coefficients d'ondelettes peut être exploitée de façon intéressante de différentes manières. Notamment, on peut utiliser l'information de direction propre à chaque sous-bande (contours horizontaux, verticaux, diagonaux) pour obtenir des caractéristiques fines sur le mouvement (amplitude et sens), en considérant l'évolution des fluctuations au cours du temps. La Fig. 2.29 illustre le principe pour extraire uniquement les contours en mouvement de l'ensemble des contours (que l'on peut obtenir par simple binarisation des coefficients d'ondelettes, ou par détection de passages par zéro). D'autre part, on sait qu'il est facile d'estimer un déplacement perpendiculaire à un contour (cf. Fig. 2.19 p.51). On peut donc estimer, dans chaque sous-bande, uniquement le déplacement dans la direction perpendiculaire aux contours mobiles, puis fusionner toutes ces informations, précises mais parcellaires, pour reconstituer le mouvement de l'objet global.



FIGURE 2.29 – Principe d'extraction des contours mobiles (cas d'un carré mobile et d'un triangle fixe).

Une autre possibilité consiste à utiliser une décomposition en ondelettes sur le signal à 3 dimensions que constitue la séquence d'images, ce qui revient en fait à adopter une méthode proche de celle proposée par Heeger [63]. Mais ceci est coûteux en calculs.

Signalons pour finir que la transformée en ondelettes est utilisée en pratique dans le standard d'images fixes JPEG2000, permettant d'atteindre des taux de compression spatiale plus élevés que ceux obtenus avec le standard JPEG qui est basé sur la transformée DCT (cf. section 3.5.2 au chapitre suivant).

# Chapitre 3

# Normes et standards du multimédia

# Préambule

Ce chapitre présente les standards de fichiers et normes de codage des images et des vidéos, développés notamment pour atteindre des taux de compression élevés. C'est l'occasion de décrire les formats très répandus JPEG et MPEG.

# 3.1 Introduction

Le terme multimédia naît dans les années 50 pour désigner un système de présentation avec un projecteur de diapositives couplé à un magnétophone déclenchant l'avance des diapos. Etymologiquement, la communication multimédia signifie « plusieurs moyens » pour faire passer un message [78]. Elle résulte de la synthèse entre trois métiers : l'informatique (micro-ordinateur, bases de données, internet), les télécommunications (téléphonie, réseaux haut débit) et l'audiovisuel.

Les objets manipulés sont l'image, la vidéo, le son, la parole, les données, le texte, le graphique, le modèle 3D. Dans le futur, d'autres objets, tactiles, olfactifs, voire gustatifs, entreront en jeu. Les applications multimédia sont les programmes qui manipulent ces contenus : transmission TV numérique, site web, jeu vidéo, visioconférence, visiophone, serveur vidéo interactif, banque d'images, borne interactive, catalogue vidéo, identification d'individu, encyclopédie numérique, commerce électronique.

La numérisation est la clé technologique de la communication multimédia. La synchronisation entre médias est cruciale, car un son décalé de quelques dixièmes de seconde par rapport à la vidéo est intolérable. Enfin, la **standardisation** est indispensable pour l'interopérabilité, la pérennité et la rentabilité des applications de communication multimédia : "*Standard is volume*", d'où l'importance des normes et formats [36]. La standardisation influence le marché (*e.g.* Windows qui n'offrait pas de produit capable de télécharger son système d'exploitation a été concurrencé par Java qui permet cette fonctionnalité). Le processus de normalisation implique quatre acteurs : les industriels (télécoms), les développeurs (informatique), les créateurs de contenu (audiovisuel), et les consommateurs (usage). Une norme met 5 à 10 ans à s'imposer.

# 3.2 Manipulation d'objets multimédia

On distingue différentes manipulations, qu'on passe en revue ci-après :

- les manipulations en entrée (création) concernent : la saisie, la numérisation, le codage, la compression, la protection des contenus,
- les manipulations intermédiaires concernent : le stockage, l'archivage, la transmission, la distribution,
- les manipulations en sortie (exploitation) sont : la sélection, la décompression, le décodage, la restitution analogique, la reproduction.

### 3.2.1 Saisie et numérisation

La saisie consiste en :

- la capture d'une image naturelle (appareil photo, caméra, scanner),
- ou la création d'image de synthèse (norme VRML, cf. section 3.2.6).

Physiologiquement (cf. chap. 1.9), une image couleur est décrite par trois grandeurs : la teinte (information de couleur reconnue et mémorisée par le cerveau), la luminosité (ou intensité lumineuse) allant du plus sombre (noir) au plus clair (blanc), et la saturation (ou pureté) indiquant le pourcentage de blanc mélangé à la couleur. L'œil peut discerner environ 360 000 nuances (120 couleurs pures  $\times$  30 niveaux de saturation  $\times$  100 intensités). D'autre part, le pouvoir séparateur de l'œil vaut 0.3 milliradian : il peut distinguer un motif de 3 mm à une distance de 10 m.

La **numérisation** consiste à transformer l'information visuelle analogique en une image numérique, c'est-à-dire une grille de pixels, par échantillonnage et quantification (discrétisation du signal). Le pixel (de l'anglais *picture element*) est l'élément d'image discret.

La quantification (caractérisée par le nombre de bits par pixel bpp) est conditionnée par la capacité de l'œil à discerner une nuance de gris (typ. 1/100: une nuance parmi une centaine). Donc 8 bits suffisent *a priori* pour coder la luminance. Un pixel couleur peut alors être codé sur 24 bits. Avec moins de 7 bits (128 niveaux) par composante couleur, on voit apparaître le phénomène des bandes de Mach sur un dégradé continu. C'est le cas notamment pour un codage couleur sur 16 bits : 5 bits (32 niveaux) par couleur, plus un bit de masquage pour l'incrustation (*overlay*).

L'échantillonnage spatial est caractérisé par la résolution r **§4.58** :

- la résolution d'une image s'exprime en ppm (points par millimètre) ou en dpi (*dot per inch* ou point par pouce; 1 pouce = 25 mm),
- dans le cas d'une image imprimée, on a typ. r = 300 à 600 dpi (imprimante laser),
- dans le cas d'une pellicule photo ou diapositive, on a typ. r = 100 ppm,
- la définition est le produit de la résolution par la dimension physique de l'image :  $d = r \times dim$ .

Différents formats caractérisent différents types d'images associés à une définition spécifique (Tab. 3.1) [78].

<b>T</b>	Dimension	Namelana	T_:11_
Type	Dimension	Nombre	Tame
		de points	du fichier
QCIF	$176 \times 144$	25000	50ko à 16 bpp ( $32000$ couleurs)
CIF	$352 \times 288$	100000	200ko à 16 bpp ( $32000$ couleurs)
TV CCIR	$720 \times 576$	400 000	1.2 Mo a 24 bpp
			(16  millions de couleurs)
TVHD	$1440\times1152$	1600000	4.8Mo à 24 bpp
			(16  millions de couleurs)
TVHD	$1920\times1152$	2211840	6.6Mo à 24 bpp
16/9			(16  millions de couleurs)
VGA	$640 \times 480$	300 000	300ko à 8 bpp (256 couleurs)
SVGA	$800 \times 600$	480000	480ko à 8 bpp
			1Mo à 16 bpp
			2  Mo à $32  bpp$
XGA	$1024 \times 768$	768ko	1.5 Mo a 16 bpp
	$1280\times1024$	1.3 Mo	4Mo à 24 bpp
A4 N&B	$21\times29.7\mathrm{cm}$	8640000	8.6Mo à 8 bpp (256 NdG)
diapo	$2000 \times 3000$	6 000 000	18Mo à 24 bpp
$24 \times 36$			(16  millions de couleur)
photo	$4320\times3240$	14000000	4.2Mo à 24bpp
JPEG			(avec compression $\approx 10$ )

IADLE 9.1 DEMINUM ET TAME des TYDES d'IMAges.	TABLE 3.1	<ul> <li>Définition</li> </ul>	et taille o	des types d'images.
---	-----------	--------------------------------	-------------	---------------------

- Le format CIF (standard prévu pour la visiophonie) correspond au quart de l'image de TV classique. La qualité CIF équivant à celle d'un magnétoscope VHS (Video Home System).
- Le format CCIR (Comité Consultatif International des Radiocommunications) est le seul format normalisé de présentation d'image.

Les formats informatiques ont certaines spécificités :

- format non-entrelacé (contrairement à la TV avec trames paires et impaires),
- fréquence de balayage f > 60 Hz pour éliminer le papillotement (flicker),
- le pitch est la taille du point élémentaire de l'écran (typ. 0.3 mm),
- il faut faire attention aux formats trop grands (typ. 21 pouces) car ils peuvent engendrer une fatigue visuelle.

### 3.2.2 Codage et compression

Le codage des images suit le proverbe « La nuit, tous les chats sont gris ». En effet, le contenu informatif principal d'une image est la luminance, *i.e.* le niveau de gris. La couleur est secondaire, et quand il fait trop sombre, l'œil ne distingue plus les couleurs. Ainsi, le standard TV basé sur le triplet YCrCb, où la luminance vaut Y = 0.3R + 0.59G + 0.11B (cf. Eq. (1.76), p. 31), se traduit par un codage numérique dit « 4 : 2 : 2» : 4 valeurs de Y pour seulement 2 valeurs de Cr et 2 valeurs de Cb. C'est-à-dire que parmi 4 pixels voisins, tous portent l'information de luminance, mais seulement 2 portent l'information de chrominance rouge, tandis que les 2 autres portent l'information de chrominance plus de données de luminance que de chrominance, sans nuire à la perception visuelle.

La **compression** répond au dicton « An image is worth a thousand words » : il y a effectivement un rapport de taille de fichier d'environ 1/1000 entre une page d'écriture et une image de mêmes dimensions! Plus techniquement, une minute de vidéo de définition  $640 \times 480$  en 16 millions de couleurs représente 1.39 Go. Sa transmission en temps-réel nécessite donc un débit de 185 Mbit/s. D'où le besoin évident de compression. On en distingue deux types principaux :

- la compression sans perte (*lossless*) tel GIF : facteur de compression assez faible 1 : 2 (C = 2). Elle est utile pour le médical, ou pour la reconnaissance des formes.
- la compression avec perte (*lossy*) tel JPEG : meilleur facteur de compression (C = 10 sans altération visuelle). Elle utilise trois types de procédés : DCT, ondelettes ou fractales. On y revient au chap. 3.5.

### 3.2.3 Protection, identification du contenu et stockage

Les techniques d'aquamarquage protégeant les images s'appliquent :

- soit sur l'amplitude de certains points-image,
- soit au niveau du spectre fréquentiel,
- soit au niveau du code compressé.

L'opération de contrôle des aquamarques s'appelle le monitoring.

Pour l'identification des objets numériques, 64 bits suffisent.

Pour le stockage, on retiendra surtout l'intérêt des formats multi-échelles.

### 3.2.4 Transmission

Tout protocole de transmission repose sur le modèle standard OSI en 7 couches.

La Fig. 3.1 donne, pour un facteur de compression C = 20, les temps de transmission d'image pour différents débits et formats (typ. on a un transfert à 1 s/img pour une image CIF transmise avec un débit de 128kbits/s) [78]. Ceci est évidemment à mettre en regard du besoin : par exemple pour une borne interactive, le temps d'attente maximal tolérable pour un usager est de l'ordre de la seconde. D'où le besoin des protocoles réseaux à haut débit **§4.1**].

La téléphonie mobile a connu une évolution rapide résumée ci-dessous :

- GSM (Global System for Mobile communications) : adoptée en Europe à partir de 1992, cette norme dite de deuxième génération (2G) a remplacé l'analogique. Les premiers réseaux de téléphonie mobile fonctionnent sous ce système, très lent pour la transmission de données, à l'exception des SMS (mini-messages écrits).
- WAP (Wireless Application Protocole) : ce protocole permet de naviguer sur Internet à partir du réseau GSM. Introduit en 1999, le WAP, trop lent, a été un échec commercial.



FIGURE 3.1 – Transmission d'images fixes compressées.

- i-mode : ce standard permet de naviguer sur Internet. Conçu par l'opérateur japonais NTT DoCoMo, il est développé sous licence en Europe par le français Bouygues Télécom, l'espagnol Telefonica Moviles et le néerlandais KPN.
- GPRS (General Packet Radio Service) : lancée en 2002, cette technologie dite de « deuxième génération et demie » (2,5G) améliore la transmission sur le réseau GSM grâce à une commutation de données par paquets, optimisant ses capacités. Elle permet une transmission de données multimédia 5 à 6 fois plus rapide que sur le réseau GSM classique.
- UMTS (Universal Mobile Telecommunications System) : norme dite de troisième génération (3G), elle est mise en place très progressivement en Europe. En offrant un débit beaucoup plus important que le GPRS (5 à 8 fois plus rapide), l'UMTS permet la transmission de données plus riches, notamment la vidéo.
- 4G et 5G : normes de quatrième et cinquième générations : à chaque génération, on gagne à peu près un ordre de grandeur en débit ( $\approx \times 10$ ).

### 3.2.5 Restitution

Elle repose sur les techniques de synthèse couleur (cf. Fig. 1.20, p. 30) :

- la synthèse additive : pour les écrans cathodiques. La distance optimale d'observation est fonction du *pitch* et de l'œil.
- la synthèse soustractive : pour les imprimantes.
- le calibrage des couleurs : il est basé sur une table de référence Q60A.

### 3.2.6 Représentation 3D

Elle repose sur la norme VRML (*Virtual Reality Modeling Language*). Cette norme, née en 1994, a pour origine le langage Open Inventor de Silicon Graphics. Ses objectifs sont la description des mondes 3D et la prise en compte des hyperliens. L'évolution de VRML vers Internet a abouti à Web3D.

Le principe de cette norme est le suivant : un objet est codé par des sommets (vertex) V(x, y, z) et des faces triangulaires  $F(V_1, V_2, V_3)$ . Le maillage repose sur une information géométrique (position) et sur une information topologique (connectivité). On appelle valence d'un sommet le nombre de ses voisins. Le coût de codage naïf (3F, 6V) vaut (avec un format int32) : 192 bit/sommet. Un codage adapté donne :  $6V \log_2(V) = 100$  bit/sommet. La borne inférieure (démontrée par Tutte en 1964) est : 3.24 bit/sommet. On peut donc compresser la représentation 3D.

# 3.3 Formats de fichiers d'images fixes

Un fichier image est constitué de [106, 26]:

- un en-tête comportant : le mode d'emploi du contenu, un magic number de 2 octets (SOI pour JPEG : Start Of Image), les dimensions de l'image (typ. 512 × 512 en robotique) etc.
- des données (brutes ou compressées) : codées en binaire ou hexadécimal;
- des métadonnées (sous forme d'étiquettes ou tags) : identifiant de transport, de trame intra, sous-titres, protection (copyright), colorimétrie, réglage de caméra, scénario, synchronisation, time-code, positionnement GPS, prise de vue, panoramique, mot-clé pour catalogage et moteur de recherche, etc.

Pour les images fixes, il existe de nombreux formats propriétaires. Les formats les moins courants ne sont pas présentés ici (TGA, PaintBrush PCX, Sun Raster RAS, Macintosh PICT). Les formats utilisant une table de couleurs (palette) ne présentent plus guère d'intérêt (car on dispose de cartes vidéo sans limite de mémoire à 32 bits/pixel). Les formats les plus courants sont donnés ci-dessous.

### 3.3.1 Format BMP

C'est le format bitmap de Windows. Le début du fichier comporte un en-tête de taille fixe (54 octets) qui donne les informations sur l'image (dont certaines sont inutilisées). La Fig. 3.2 est un exemple de fichier BMP, où l'on peut identifier les valeurs en hexadécimal listées ci-dessous.

- magic number : BM (2 mots), 42 4D
- taille de fichier (4 mots), 4A E7 04 00
- zone réservée pour extension  $(2 \times 2 \text{ octets})$ , 00 00 00 00
- offset de début (4 octets) : 36H (soit 54), 36 00 00 00
- à l'adresse 14: taille de l'en-tête (1 mot): 28H (soit 40 octets), 28 00
- largeur de l'image (4 octets) : ici 01A4H (soit 420), 00 00 A4 01
- hauteur de l'image (4 octets) : ici 00FFH (soit 255), 00 00 FF 00
- nombre de plans (4 octets) : ici 1, 00 00 01 00
- nombre de bits par pixel (1 octet suffit) : 18H (soit 24), 18 00
- type de compression (4 octets nuls) : RLC, 00 00 00 00
- taille des données image (4 octets), 14 E7 04 00
- résolutions horizontale et verticale (en pixels/mètre) :  $2 \times 4$  octets,
- -à l'adresse 46 : nombre de couleurs utilisées (4 octets), 00 00 00 00
- couleurs importantes (4 octets), 00 00 00.

Après l'en-tête, commence la séquence des données brutes de l'image, en l'occurrence les valeurs RVB des pixels : ici C5 A4 A8 C4 A5 A5 etc.

### 3.3.2 GIF et PNG

Ces deux formats ont des caractéristiques communes :

- format adapté aux (petites) images de synthèse (bouton, logo),
- restitution progressive,
- compression sans perte (Lempel-Ziv Welch),
- format GIF créé par Compuserve,
- le format PNG est une norme ISO : c'est l'équivalent de GIF, mais sans brevet.

### 3.3.3 TIFF

C'est un format de fichier natif (pour la création d'images avec le logiciel Adobe Photoshop). Il est précurseur des formats avec métadonnées : il a une structure de balises ou *tags*, d'où son nom (*Tagged Image File Format*).

# 3.3.4 Formats issus de la norme JPEG

 JFIF : c'est un format à la norme JPEG extrêmement réducteur. Il utilise un codage des couleurs selon le CCIR 4 : 2 : 2.

🐝 markov -	Microsoft Visua	d C++ - [G:\\cami	on420_255.bmp]	
Eile <u>E</u> dit	<u>V</u> iew <u>I</u> nsert <u>P</u> r	roject <u>B</u> uild <u>T</u> ools <u>V</u>	<u>√</u> indow <u>H</u> elp	_ 뭔 ×
1	1 🗗 🐰 🖬	<u>₿</u>   <u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u></u>	📴 ጆ 😤 🎭 MIL_ID	
		(All class members)	No members - Create C/C	++ Member Function
		(		
X 000000 000010 000020 000040 000050 000060 000060 000080 000080 000080 000080 000080 000080 000080 000060 000060 000060 000100 000100 000120 000180 000180 000180 000180 000180 000180 000180 000180 000180 000180 000180 000180 000180	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	28 00 ENJ6(. ▲ 00 00
Ready				Off 000022 Len 000000 OVR READ //

FIGURE 3.2 – Exemple de fichier BMP.

- **SPIFF** (Still Picture Interchange File Format) est le seul format du domaine public (norme JPEG, partie 3). C'est un format ISO pour garantir l'interopérabilité.
- **JTIP** (JPEG Tiled Image Pyramid) est le premier format pyramidal de JPEG. Il répond au besoin de gérer les grandes images : représentation pyramidale et découpage en tuiles. La taille des tuiles est variable (par défaut  $720 \times 576$ ).

# 3.3.5 FlashPix

C'est un format propriétaire multirésolution (pyramide et tuiles). La taille des tuiles est fixe :  $64 \times 64$ .

### 3.3.6 Formats liés à la photo numérique

- EXIF : EXchangeable Image file format for Digital Still Camera. Il est dû à un consortium japonais de fabricants d'appareils photo numériques. Le fichier contient deux images : la vignette et l'image complète.
- **CIFF** : format JFIF simple + informations relatives à l'appareil photo.

# 3.4 Codage des applications multimédia

### 3.4.1 Catégories d'applications

Les paramètres régissant la complexité d'une application sont :

- La configuration monoposte ou multiposte (selon le nombre d'usagers).
- Le temps réel ou le temps différé. Au sens informatique, c'est la capacité à réagir instantanément à un événement. Internet, qui est régi par le protocole simple IP (*Internet Protocol*), avec sa

couche de contrôle TCP (*Transmission Control Protocol*), pose le problème de synchronisme dans la gestion du temps. Or le retard maximum acceptable pour une conversation est de 300 ms. Il y a donc un besoin de resynchronisation des paquets. Les solutions consistent en l'usage (i) d'un protocole supplémentaire RTP et RTCP (*Real Time Control Protocol*) qui gère la priorité des paquets, et (ii) des applets Java.

— L'utilisation locale (off-line) ou à distance (en ligne). L'usage de CD-Rom (600 Mo), de DVD (4 à 17 Go) ou de disques blue-ray (50 à 100 Go) sont des exemples d'application locale, qui résolvent les problèmes de stockage (car 1 heure de vidéo MPEG1 = 675 Mo), de portabilité (galette de 12 cm de diamètre) et de piratage (découpage du monde en zones). L'usage d'Internet, des réseaux Intranet d'entreprises, ou des terminaux d'accès à Internet pour la TV et le téléphone (boîtiers d'interface avec modem appelés set-top box) sont des exemples d'application distante, dont l'ancêtre est le Minitel (datant de 1980). Pour ces accès, on dispose d'interfaces de plus en plus rapides : interface RNIS à 64 ou 128 kbit/s; interface xDSL à 1 Mbit/s etc.

# 3.4.2 Normes de codage des applications

Les objectifs du codage des applications sont la pérennisation de la communication, l'interopérabilité, l'indépendance vis-à-vis des plates-formes, et la portabilité [84]. On distingue deux types de normes :

- les normes de codage (du contenu, de la structure) : un document électronique comporte un contenu, une structure logique, des éléments de présentation. La structure logique du document est décrite à l'aide d'un langage de balises simple et portable, défini par une syntaxe de balisage, *e.g.* HTML ou XML.
- les normes d'échange (du contenu, du contenant) : gestion de l'interactivité. Les normes MPEG4 et MPEG7 rapprochent les notions de document et d'application (en incluant dans les documents MPEG4 des éléments d'interactivité).

### Java

La programmation en Java est largement utilisée pour les applications multimédia distribuées (JavaScript, applet). Par exemple, les moteurs MHEG-5 (*Multimedia & Hypermedia Expert Group*) pour la télévision interactive utilisent une machine virtuelle Java. De même, MPEG-J est l'intégration de Java à MPEG-4, pour décrire des scènes interactives et gérer des contrôleurs de média. Une applet (*application object*) est un élément de marquage qui assure le lien entre une page HTML et du code Java. Le langage Java ressemble au langage C++. Ses avantages sont la portabilité, la sécurité, la gestion automatique de mémoire, le modèle de pointeur sûr. Son inconvénient principal par rapport au langage C/C++ est la lenteur. L'exécution du code source repose sur le concept de machine virtuelle JVM.

## MPEG

MPEG est un format informatique non entrelacé. Un flux MPEG contient trois types d'images : I (intra), P (prédictif) et B (bidirectionnel). Une image I est compressée spatialement au standard JPEG. Une image P est compressée par prédiction et compensation de mouvement. Une image B utilise aussi une estimation de mouvement, mais avec une prédiction dans les deux sens temporels (avant et arrière).

- MPEG2 permet un débit de 20Mbit/s. Il offre plusieurs profils : le plus courant MP@ML (Main Profil at Main Level) correspond à un format 720 × 576 à 25 img/s.
- MPEG4 est un standard orienté objet, c'est-à-dire qu'il code le contenu, après une reconnaissance des objets qui composent l'image. Les objectifs sont : le haut et bas débit, l'interactivité, les compressions fortes (5 à 64 kbit/s en téléphonie mobile, 2 Mbit/s pour les films). Les fichiers vidéo MPEG4 sont nommés avec l'extension « .MP4 ».
- MPEG7 code la description du contenu. C'est une norme de description standardisée des données permettant une interprétation informatique (grâce aux métadonnées). Elle est utile pour les moteurs de recherche et l'indexation, la reconnaissance des images par leur contenu. Les outils de description du contenu visuel sont groupés en diverses catégories : couleur, texture, forme,

contour, mouvement. Les outils de description audio sont de 3 types : effets sonores, instruments, reconnaissance vocale.

 MPEG4 et 7 sont complémentaires (cf. MPEG-21). Ils ne définissent rien au niveau de la segmentation d'image.

### H261-H263

Ce sont des formats conçus à l'origine pour la visiophonie. Ils ont servi de fondement au standard vidéo MPEG.

### MP3

MP3 est un format de compression audio avec perte. MP3 correspond à la couche 3 de la norme audio de MPEG2. Il permet un retard minimal entre codage et décodage de 59 ms. Il a été très utilisé pour le stockage (intérêt pour l'autoradio : on peut stocker 10 heures de musique sur un CD). Pour un son de qualité CD (44 kHz, 16 bits) : on a un facteur de compression C = 10, et un débit D = 64kbit/s.

## 3.4.3 HTML

- HyperText Markup Language.
- Langage de balisage très simple et portable.
- La DTD (Définition de Type de Document) fournit la syntaxe des balises.
- document HTML comporte 3 parties : ligne de la version, entête déclarative, corps.
   Exemple :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
<HEAD>
<TITLE> Titre du document html </TITLE>
</HEAD>
<BODY>
<P> Ceci est le corps du document
</BODY>
</HTML>
```

### 3.4.4 XML

Extensible Markup Language : c'est un langage de balise de document avec information structurée, qui a succédé à HTML. Il est moins limité que HTML, mais moins lourd à implanter que le SGML (Standard Generalized Markup Language).

### WAP et WML

Le WAP (Wireless Application Protocol) est destiné aux applications mobiles sans fil. Il est basé sur XML et IP.

# 3.5 Compression d'image

### 3.5.1 JPEG : emploi de la transformée DCT

La norme de compression JPEG comporte 6 étapes [68] :

1. Découpage de l'image en blocs  $8 \times 8$ .

2. Transformation amplitude-fréquence : on centre les échantillons (en soustrayant la valeur moyenne) et on applique la DCT (dont le nombre d'opérations est proportionnel au carré du nombre de points).

$$S_{uv} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} I_{xy} \cos \frac{(2x+1)\pi u}{16} \cos \frac{(2y+1)\pi v}{16}$$
(3.1)

où  $C_u = 1$  (resp.  $C_v$ ) sauf pour u = 0 (resp. v), auquel cas  $C_u = 1/\sqrt{2}$  (resp.  $C_v$ ).  $I_{xy}$  représente la valeur (centrée) du pixel de coordonnées (x, y).

- 3. Filtrage et quantification : on emploie une matrice de quantification qui résulte d'essais psychovisuels (Fig. 3.3a). L'œil est en effet un filtre passe-bas, donc les valeurs  $S_{uv}$  correspondant aux hautes fréquences peuvent être atténuées (on les divise par le coefficient respectif de cette matrice).
- 4. Réorganisation en zigzag (selon l'ordre numéroté de la Fig. 3.3b) pour passer d'un bloc 2D à un tableau 1D contenant 64 coefficients.
- 5. Codage par plages RLE (*Run Length Encoding*) : au lieu de coder chacun des coefficients rangé dans le tableau 1D issu du zigzag, on ne code que les coefficients non nuls, et on leur associe le décompte de zéros (nombre appelé *run*) qui les séparent du coefficient non nul précédent. L'intérêt est qu'en HF, on a beaucoup de valeurs nulles, donc cette méthode informatique est efficace comme première étape de compression sans perte.
- Codage entropique (de Huffman) : les valeurs les plus fréquentes sont codées avec les mots-codes les plus courts (cf. section 1.2.3).
   §4.53

16	11	10	16	24	40	51	61	1	2	6	7	15	16	28	29
12	12	14	19	26	58	60	55	3	5	8	14	17	27	30	43
14	13	16	24	40	57	69	56	4	9	13	18	26	31	42	44
14	17	22	29	51	87	80	62	10	12	19	25	32	41	45	54
18	22	37	56	68	109	103	77	11	20	24	33	40	46	53	55
24	35	55	64	81	104	113	92	21	23	34	39	47	52	56	61
49	64	78	87	103	121	120	101	22	35	38	48	51	57	60	62
72	92	95	98	112	100	103	99	36	37	49	50	58	59	63	64
a)									b)						

FIGURE 3.3 – a) Matrice de quantification psychovisuelle; b) Ordre du zigzag.

La décompression correspond évidemment aux opérations inverses.

La performance atteinte pour une image couleur 24 bits/pixel (24 bpp) est :

- une bonne qualité à 0.75 bpp (facteur de compression C = 30),
- visuellement identique à l'original à 2.25 bpp (compression C = 10).

Les limites de la norme JPEG sont les suivantes :

- la compression avec perte est limitée en pratique à 0.25 bpp (en deçà, l'image devient trop dégradée visuellement),
- pas de compression sans perte (sauf avec la version *lossless* JPEG-LS),
- elle convient surtout pour des images naturelles (donc problème pour des documents composés avec du texte),
- le décodage progressif est limité,
- elle est peu robuste aux erreurs.

### 3.5.2 JPEG2000 : emploi de la transformée en ondelettes

La compression repose sur les principes suivants : multi-échelle, filtres localisés spatialement, bandes fréquentielles orientées spatialement [124, 116]. Elle se décompose en 6 étapes (Fig. 3.4) : une transformée couleur multi-composantes (MCT) suivie da la transformée en ondelettes discrètes (DWT), puis la quantification, le codage entropique, et enfin l'allocation de débit et le codage des paquets qui forment le *bitstream* en sortie. La complexité est de o(n) opérations ; et la performance atteinte vaut : 0.6 bpp.



FIGURE 3.4 – Schéma-bloc de JPEG2000.

Les avantages de JPEG2000 par rapport à JPEG sont les suivants :

- de nouvelles fonctionnalités : accès aléatoire, flexibilité, choix de l'espace couleur, région d'intérêt ROI [87, 127, 128],
- le robustesse aux erreurs : le taux d'erreur *bit error rate* vaut BER= $10^{-5}$  à  $10^{-4}$  (voire au pire  $10^{-3}$  pour une liaison radio, sans fil),
- un gain de rapport signal sur bruit SNR : +3 a + 6 dB (cf. Fig. 3.5),
- des applications variées : domaine spatial, domaine médical.

La norme comporte 11 parties :

**Part1** : format de base JP2 (décembre 2000),

- **Part2** : extension pour applications spécifiques (e.g. hyperspectral) : format JPX (équivalent à du PDF normé international),
- Part3 : Motion JPEG2000 : MJP,

Part4 : conformance,

Part5 : logiciel de référence,

Part6 : images composites (multi-couches) : JPM,

Part7 : partie vide (prévue pour les caméras numériques),

Part8 : sécurité (authentification, droits, accès) : JPSEC,

**Part9** : interactivité et protocole : JPIP,

Part10 : volumétrique : JP3D,

Part11 : sans fil (téléphone, vidéoprojecteur) : JPWL.

Pour plus d'information, on peut consulter les sites web :

```
http://www.jpeg.org (information générale)
http://linuxj2k.org/ (version système linux)
http://www.ece.uvic.ca/~mdadams/jasper/ (implantation en C)
```

### 3.5.3 Emploi des fractales

Les principes de la méthode reposent sur la géométrie fractale (autosimilarité), le théorème du point fixe (attracteur), l'itération d'une transformée affine (IFS).

# 3.6 Compression de séquences vidéo

### 3.6.1 Principe du codage hybride

Le schéma de principe d'un codage hybride (estimation de mouvement et transformée fréquentielle) est donné sur la Fig. 3.6 [37].

— Le **buffer de trames** stocke une ou plusieurs images de référence.


Compression C = 90: a) JPEG; b) JPEG2000 (YUV; DWT9/7)

FIGURE 3.5 – JPEG2000 vs. JPEG : comparaison de qualité à compression identique (soit un gain  $\times 2$  en compression à qualité égale).



FIGURE 3.6 – Schéma-bloc d'un codeur hybride.

- L'estimation de mouvement utilise la mise en correspondance de blocs entre image courante et images de référence pour calculer les déplacements (vecteurs de mouvement). C'est le module le plus important et le plus coûteux.
- La **compensation de mouvement** peut alors reconstruire l'image courante à partir des vecteurs de mouvement et de la référence (image prédite dans le domaine temporel : inter-prédiction).
- L'intra-prédiction réalise une autre forme de prédiction reposant uniquement sur l'image courante (compression spatiale de type JPEG).
- La **différence** entre l'image courante et l'image prédite (inter ou intra) constitue l'erreur résiduelle à transmettre.
- La transformée fréquentielle et la quantification réduisent la redondance spatiale : en transformant les données spatiales en données fréquentielles, la représentation devient plus compacte (plus facile à compresser).
- La quantification des coefficients fréquentiels repose sur le HVS (système visuel humain, cf. propriété de masquage de l'information haute fréquence non visible). Les coefficients quantifiés ont une meilleure distribution statistique en vue de la compression.

- Le **codage entropique** supprime la redondance statistique et encode les coefficients transformés quantifiés et les vecteurs de mouvement sous forme d'un flux de bits (*bitstream*) en sortie.
- Par ailleurs, les coefficients sont aussi utilisés après inversions pour reconstruire les trames stockées dans le *buffer*.
- Parfois, un filtre anti-blocs est appliqué avant le buffer de trames, pour réduire les effets de blocs et améliorer la qualité subjective. C'est un filtre passe-bas (cf. visiophone H.261) qui coûte 33% des calculs du décodeur H.264.

#### 3.6.2 Normes MPEG-4 et H.264

Parmi les algorithmes de compression vidéo, les standards MPEG-4 (standard ISO 1999) et H.264 (standard ITU 2003) sont les plus performants, mais complexes et très coûteux en calculs, d'où le besoin d'architecture matérielle et de circuits intégrés SoC pour encoder de la vidéo  $720 \times 480$  à 30 Hz [37].

Pour les systèmes de communication multimédia temps-réel, plusieurs technologies de pointe sont requises :

- réseau large bande (pour augmenter la bande passante),
- compression (pour réduire le *bitrate*, la vidéo étant la plus gourmande),
- technologie VLSI pour fabriquer des codecs hardware (la capacité des puces double tous les 18 mois suivant la loi de Moore).

MPEG-4 a trois objectifs : interactivité (requiert le codage basé objet), compression, accès universel (*scalability*, robustesse aux erreurs). Plusieurs profils et niveaux existent, dont le profil simple (SP) et le profil simple avancé (ASP).

H.264, aussi appelé Joint Video Team (JVT) ou MPEG-4 Advanced Video Coding (AVC), est destiné à la diffusion, à la haute définition de DVD, au streaming, au stockage et à la surveillance vidéo. Il a démarré en 1999 (projet long terme H26L) et a été finalisé en 2003 sous le nom MPEG-4 Part 10 AVC. Il offre un gain de 60 % par rapport à MPEG-2 et un gain de 40% par rapport à MPEG4-ASP. H.264 a quatre profils : basique (pour téléphonie et mobiles), étendu (pour internet streaming), principal (pour video broadcasting, jeux) et haut (pour TVHD).

Ces nouveaux standards ont les caractéristiques suivantes :

- compensation de mouvement : précision au quart de pixel,
- code à longueur variable adaptatif au contexte (CA-VLC) : choix de la LUT en fonction du voisinage du bloc,
- taille de blocs variable (41 SAD à calculer par bloc candidat),
- taille de la fenêtre de recherche : [-64, +63] en horizontal et [-32, 31] en vertical pour la première image de référence, [-32, 32] et [-16, 15] pour les autres.
- images de référence multiples,
- intra-prédiction : 4 modes possibles pour chaque bloc  $16 \times 16$ , 9 modes pour chaque bloc  $4 \times 4$ . Elle peut s'implanter sur une architecture à bus de données reconfigurable.

Notons qu'entre MPEG-4 standardisé en 1999, et H.264 standardisé en 2003, la complexité de calcul (donc le besoin en puissance de calcul) a été multipliée par plus de 10 (croissance plus rapide que la loi de Moore, qui ne donne qu'un facteur  $\times 6.36$  sur 4 ans!). Ceci justifie le besoin d'accélérations matérielles.

Le profilage d'instructions à l'encodeur indique les modules critiques à optimiser (implantation matérielle parallèle, jeu d'instructions spéciales MMX) : l'estimation de mouvement représente 95% du coût total (la recherche exhaustive jouant le rôle majeur). Pour MPEG-4, un pipeline à 2 étages sur les macro-blocs suffit. Pour H.264, un pipeline à 4 étages est nécessaire. Un codeur H.264 sur une puce en technologie microélectronique  $0.18\mu m$ , qui mesure  $8 \times 4 mm^2$ , comporte 1000K portes logiques, 35 ko de mémoire et fonctionne à 81 MHz, permet l'encodage en temps-réel (N.B. : une cadence de 30 images/s en format CIF avec le profil de base requiert 300 GIPS).

#### 3.6.3 MPEG-5 et H.265

Le standard MPEG évolue en permanence : les normes récentes s'appellent MPEG-5 EVC (pour Essential Video Coding) et HEVC/H.265 (pour High-Efficiency Video Coding). [38]

# Chapitre 4

# Exercices

## Préambule

Ce chapitre propose une soixantaine d'exercices pour s'initier aux notions décrites dans les chapitres précédents. Certains exercices sont à dessein partiellement redondants pour mieux s'entraîner. Une solution succincte de chacun des exercices est fournie dans le livre [100].

## 4.1 Débit d'information

Calculer le débit d'information (en bits/seconde) apporté par un émetteur TV noir et blanc dans les deux cas suivants :

- 1. Contraste poussé à fond : un pixel est soit noir, soit blanc.
- 2. Contraste équilibré : 256 niveaux de gris dans l'image.
- 3. Quelle conséquence peut-on en déduire en terme de besoin?
- 4. Etude de cas d'une transmission sans fil sur réseau GSM (9600 bauds). Extrapoler avec l'évolution technologique : TV couleur, réseau GPRS, UMTS, compression JPEG, MPEG etc.
- 5. Evaluer entropie et redondance de la source TV N&B dans les 2 cas étudiés.

N. B. : Standard en Europe : 25 img/seconde; 520 000 pixels/img.

## 4.2 Modification d'histogramme

Sur la Fig. 4.1 sont représentées 8 images dont l'originale en haut à gauche [40]. A droite de chaque image est représenté l'histogramme original.

Déterminer l'allure de la correction opérée (par exemple par des LUTs) pour obtenir les différentes images à partir de l'image originale.

A titre d'exemple, la correction neutre (transformation identité) est représentée en superposition sur le premier histogramme.

## 4.3 Principe de l'égalisation d'histogramme

Le principe de l'égalisation d'histogramme dans le cas continu repose sur les équations suivantes :

$$s = T(r) = \int_0^r p_r(u) du \tag{4.1}$$

$$p_s(s) = p_r(r)\frac{dr}{ds} \tag{4.2}$$

où  $r \in [0; 1]$  représente le niveau de gris de l'image en entrée,  $s \in [0; 1]$  le niveau après égalisation, et  $p_r(r)$  et  $p_s(s)$  les densités de probabilité correspondantes.



FIGURE 4.1 – Images et histogramme.

On considère un histogramme continu défini par :

$$p_r(r) = -2r + 2 \text{ pour } 0 \le r \le 1$$

$$p_r(r) = 0 \text{ ailleurs}$$

$$(4.3)$$

$$(4.4)$$

- 1. Représenter l'histogramme initial.
- 2. Calculer s = T(r) et représenter l'allure de la courbe correspondante.
- 3. Exprimer r et  $\frac{dr}{ds}$  en fonction de s et en déduire l'expression de  $p_s(s)$ .
- 4. Tracer la courbe  $p_s(s)$  en fonction de s et conclure sur l'effet d'égalisation d'histogramme.

# 4.4 Egalisation et étalement

Soit une imagette de taille  $10 \times 10$  à 8 niveaux de gris dont les probabilités sont données dans le Tab. 4.1 [40].

- 1. Réaliser l'égalisation de l'histogramme : on complétera le tableau avec les valeurs de la fonction de répartition F(i) et des niveaux de sortie j.
- 2. Réaliser l'étalement de l'histogramme : compléter les valeurs T(i).
- 3. Comparer à l'égalisation.

niveau $\boldsymbol{i}$	probabilité $p_i(\%)$	F(i)	niveau $j$	étalement $T(i)$
0	10			
1	20			
2	30			
3	10			
4	20			
5	10			
6	0			
7	0			

TABLE 4.1 – Histogramme

# 4.5 Détection de contours

Les opérateurs  $M_0 = \begin{bmatrix} -1 & 1 \end{bmatrix}$  et  $M_{90} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$  sont les approximations discrètes de la dérivation horizontale et verticale, permettant le calcul des composantes du gradient utile pour la détection de contours (Eq. (1.36), section 1.7).

On considère l'imagette I de taille  $8 \times 8$  de la Fig. 4.2a, définie par les valeurs d'intensité I(i, j) en chaque pixel de coordonnées (i, j).

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	4	4	4	1	1	1
1	1	4	4	4	1	1	1
1	1	4	4	4	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
			5	ı)			

0	0	0	0	0	0	0	0
0	0	10	0	0	0	0	0
0	0	10	10	0	0	0	0
0	0	10	10	10	0	0	0
0	0	10	10	10	10	0	0
0	0	10	10	10	10	10	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
				b)			

FIGURE 4.2 – Deux imagettes : a) un carré ; b) un triangle.

- 1. Calculer les images filtrées  $I_0$  et  $I_{90}$  par les deux masques  $M_0$  et  $M_{90}$ . Pour gérer les effets de bord, on dupliquera à l'extérieur de l'image les valeurs des pixels périphériques.
- 2. Calculer l'amplitude du gradient A(i, j).
- 3. Calculer la direction du gradient  $\Phi(i, j)$ . Pour ce calcul, on utilisera la fonction arctangente arctan  $\frac{y}{x}$  qui donne l'angle entre  $-\pi$  et  $+\pi$ .
- 4. Calculer l'image binaire E(i, j) correspondant aux extremums du gradient.
- 5. Faire une liste chaînée des points de contour (extremums).
- 6. Coder le contour avec l'algorithme de Rosenfeld & Kak en partant du point correspondant à  $x_{min}, y_{min}$ .
- 7. En déduire la compression obtenue par ce codage.

# 4.6 Calcul de laplacien

- 1. Calculer le laplacien de l'image de la Fig. 4.2b.
- 2. Comment en déduire les contours?

## 4.7 Formules de dérivation numérique

A partir de la définition de la dérivée d'une fonction dans le cas continu, démontrer les formules de dérivation numérique : (i) dérivée première, (ii) dérivée seconde et (iii) laplacien 2D.

#### 4.8 Filtrage linéaire

On veut comparer deux filtres mono-dimensionnels classiquement utilisés en traitement d'image : le filtre moyenneur centré de masque  $H = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$  et le filtre binomial centré de masque  $G = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$ .

- 1. Rappeler l'expression de la fonction de transfert H(u), où u représente la fréquence spatiale normalisée ( $0 \le u \le 0.5$ ).
- 2. Calculer la fonction de transfert G(u) du filtre binomial.
- 3. Tracer sur le graphe de la Fig. 4.3 les courbes de module des deux fonctions de transfert.



FIGURE 4.3 – Courbes de module de spectre.

- 4. De quel type de filtre s'agit-il? Commenter les courbes et comparer la qualité de ces deux filtres.
- 5. On considère maintenant le filtre dérivateur :  $D = \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ . Etudier sa fonction de transfert D(u). De quel type de filtre s'agit-il ?

## 4.9 Effet de moiré par repliement de spectre

On considère une image  $I_0$  formée de traits horizontaux de période 4/300 pouce [40]. L'orientation de  $I_0$  est donnée par un angle  $\theta = 10^{\circ}$  (Fig. 4.4).

Cette image est échantillonnée par  $I_e$ , identique à  $I_0$ , mais avec  $\theta = 0$  (c'est-à-dire à l'horizontal). L'origine du repère Oxy de l'image est pris dans le coin inférieur en bas à gauche.

- 1. Représenter l'allure du spectre de  $I_0$  (en ne tenant compte que des deux premières composantes fréquentielles).
- 2. Représenter le spectre de  $I_s$ , échantillonnée de  $I_0$  par  $I_e$ .
- 3. En déduire les paramètres du moiré c'est-à-dire : la fréquence spatiale apparente  $f_m$ , et l'angle apparent  $\alpha$ .
- 4. Vérifier ces résultats par mesure directe sur l'image de la Fig. 4.4.

N.B. : 1 pouce = 25.4 mm.

#### 4.10 Transformée de Fourier

- 1. Associer le bon spectre à chacune des images de la Fig. 4.5 [40].
- 2. Démontrer que le lieu de phase nulle de la TF 2D correspond, dans le plan image normalisé, à une série de droites parallèles, distantes entre elles de :  $d = \frac{1}{\sqrt{u^2 + v^2}}$  et de direction perpendiculaire à la droite de pente : tan  $\theta = \frac{v}{u}$ .



FIGURE 4.4 – Image originale rectangulaire  $I_0$  penchée de  $\theta = 10^\circ$ , superposée avec la grille d'échantillonnage  $I_e$  placée à l'horizontal.

- 3. Calculer la transformée de Fourier d'une porte centrée  $A.\Pi_T(x)$  de largeur T et d'amplitude A. Commenter l'importance de ce résultat.
- 4. Etablir le lien entre TFD et DSF dans le cas d'un signal 1D.

# 4.11 Morphologie mathématique

On considère [40] l'image binaire de la Fig. 4.6. Représenter les images filtrées correspondant à :

- une dilatation 4-connexe,
- une dilatation 8-connexe,
- une érosion 4-connexe,
- une érosion 8-connexe.

# 4.12 Poursuite de contour binaire « -2+4 »

- 1. Coder selon le code de Freeman le contour de l'objet binaire de la Fig. 4.6, en utilisant l'algorithme de Rosenfeld & Kak, et en partant du point initial de coordonnées (4,2) (origine en haut à gauche).
- 2. Calculer le taux de compression obtenu grâce à ce codage.

# 4.13 Ouverture et fermeture

Soit la forme binaire X et l'élément structurant circulaire E présentés Fig. 4.7 [60]. Le diamètre de E est légèrement supérieur à la plus petite des longueurs des côtés de l'objet. Représenter les résultats de : a) l'érosion, b) l'ouverture, c) la dilatation, d) la fermeture.



FIGURE 4.5 – A chacun sa transformée.

FIGURE 4.6 – Image binaire.

# 4.14 Lissage morphologique

Le lissage morphologique s'obtient par une ouverture suivie d'une fermeture [60]. Soit la forme bruitée X et l'élément structurant E de la Fig. 4.8.

Représenter le résultat du lissage et les résultats intermédiaires (érosion, puis dilatation, puis dilatation, puis érosion).

# 4.15 Squelettisation morphologique

On veut amincir l'objet binaire A de la Fig. 4.9 pour en extraire son squelette [60].



FIGURE 4.7 – Forme X de type enclume, et élement structurant rond E.



FIGURE 4.8 – Forme X d'une image binaire bruitée et élement structurant E.

Pour cela, on utilise l'ensemble d'éléments structurants E:  $E = \{E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8\},$ qui est constitué de versions pivotées à 45° du premier élément  $E_1$ . Et l'on réitère jusqu'à convergence (i.e., k = K) la séquence d'amincissements suivante :

$$A \otimes \{E\} = (((\cdots (((A \otimes E1) \otimes E2) \otimes E3) \cdots) \otimes E7) \otimes E8)$$
  

$$A_0 = A$$
  

$$A_k = A_{k-1} \otimes \{E\} \quad k = 1, 2, \cdots K$$

$$(4.5)$$

Représenter le résultat de cette opération, en détaillant les résultats intermédiaires de la procédure,



FIGURE 4.9 – Eléments structurants et objet A à traiter.

sur les imagettes jointes en annexe.

On rappelle que l'amin<br/>cissement  $\otimes$  d'une forme A par un élément structurant<br/> B est défini par :  $A \otimes B = A - (A \otimes B)$  où<br/>  $\oslash$  représente l'opération Tout-ou-Rien.

Rappel sur le tout-ou-rien  $\oslash$  : soit  $B = (B_1, B_2)$  l'élément structurant constitué d'une forme  $B_1$  et d'un fond  $B_2$ ,  $A \oslash B = (A \ominus B_1) \cap (A^C \ominus B_2)$ , c'est-à-dire que le tout-ou-rien donne l'ensemble des pixels où, simultanément,  $B_1$  colle dans A et  $B_2$  colle dans le complémentaire de A.

#### 4.16 Zone aveugle

La zone aveugle de l'œil correspond au rattachement du nerf optique sur la rétine. Le test suivant permet de s'en convaincre (Fig. 4.10).

Masquer un œil ; avec l'autre, fixer la lettre *ad hoc* (D si c'est l'œil droit qui est ouvert). L'autre lettre disparaît pour une distance feuille-œil de l'ordre de 25 cm, correspondant à un angle d'environ  $15 - 20^{\circ}$  de l'axe optique.



FIGURE 4.10 – Test de la zone aveugle.

## 4.17 TV couleur Secam

Le standard français SECAM (Séquentiel Couleur A Mémoire) utilise les signaux  $YC_rC_b$  définis par :

$$Y = 0.3R + 0.59V + 0.11B$$
  

$$C_r = -1.9(R - Y)$$
  

$$C_b = +1.5(B - Y)$$
(4.6)

Pour assurer la compatibilité avec la TV N&B, on a par ailleurs :

$$R = V = B$$
 pour toute la gamme des gris, (4.7)

$$R = V = B = 1 \qquad \text{pour le blanc (100\%)}, \tag{4.8}$$

$$R = V = B = 0 \qquad \text{pour le noir (0\%)}. \tag{4.9}$$

Un générateur de mire normalisée délivre un signal constitué de barres verticales : blanc, jaune, turquoise (cyan), vert, mauve (magenta), rouge, bleu, noir. Le blanc est saturé à 100%, les autres couleurs à 75%. Ce type de mire est notamment utilisé pour le réglage des téléviseurs.

- 1. Remplir un tableau donnant les composantes RVB de chaque barre de la mire.
- 2. Calculer les amplitudes des luminances de chaque couleur délivrée par le générateur de mire.
- 3. Sachant que la luminance est représentée par une tension (en volts), et que l'écran est balayé ligne par ligne, dessiner l'allure temporelle du signal vidéo qui commande la cathode du tube-image.
- 4. Donner les formules de matriçage permettant de reconstituer les 3 composantes RVB à partir des signaux transmis  $YC_rC_b$ .

### 4.18 ACP couleur

On considère [40] une image couleur RVB caractérisée par son vecteur moyenne M et sa matrice de covariance C :

$$M = \begin{bmatrix} 69\\74\\76 \end{bmatrix}$$
(4.10)

$$C = \begin{bmatrix} 613 & 170 & 387 \\ 170 & 477 & 458 \\ 387 & 458 & 796 \end{bmatrix}$$
(4.11)

Appliquer l'ACP (cf. section 1.9.3).

### 4.19 Détection de mouvement

On considère trois images successives d'une séquence comportant un objet rectangulaire mobile (Fig. 4.11).

- 1. Dessiner les cartes des changements temporels et le résultat du ET logique.
- 2. Que se passe-t-il en cas de recouvrement partiel?

#### 4.20 Etiquetage statistique contextuel du mouvement

On considère un pixel courant s et son voisinage spatio-temporel formé des 8 plus proches voisins spatiaux et de 2 voisins temporels (passé et futur). Pour détecter le mouvement, on dispose des observations o(t) et on suppose que les voisins sont déjà étiquetés comme indiqué sur la Fig. 4.12b.

L'étiquetage est binaire (fixe ou mobile).

Le critère de décision repose sur le calcul de l'énergie minimale.

- 1. Calculer les deux termes d'énergie (attache aux données et modèle a priori) pour  $e_s = 0$ .
- 2. Idem pour  $e_s = 1$ .
- 3. En déduire la meilleure étiquette à attribuer au pixel s.
- 4. Si l'on avait considéré uniquement la valeur de l'observation ponctuelle, et opéré un seuillage binaire avec un seuil  $\theta = 10$ , quelle étiquette aurait-on obtenue?
- 5. Conclure sur l'effet de la régularisation statistique par MRF.



FIGURE 4.11 – Localisation par opération logique.



FIGURE 4.12 – a) observations; b) étiquettes.

Rappels sur les énergies de la modélisation markovienne :

$$U = u_m + \lambda u_a \tag{4.12}$$

où :

$$u_m = \sum_{c \in C} V_c(e_s, e_r) \tag{4.13}$$

$$u_a = \frac{1}{2\sigma^2} [o_s - \Psi(e_s)]^2 \tag{4.14}$$

avec :

$$V_c(e_s, e_r) = \begin{cases} -\beta & \text{si } e_s = e_r \\ +\beta & \text{si } e_s \neq e_r \end{cases}$$
(4.15)

où  $\beta > 0$  prend une des trois valeurs  $\beta_s, \beta_p$  ou  $\beta_f$  selon la clique c = (s, r) considérée (spatiale, passée ou future).

$$\Psi(e_s) = \begin{cases} 0 & \text{si } e_s = 0\\ \alpha > 0 & \text{sinon.} \end{cases}$$
(4.16)

Les paramètres du modèle sont :

 $\beta_s = 20, \ \beta_p = 10, \ \beta_f = 30, \ \lambda = 6.$ 

 $\alpha=30$  est la valeur moyenne des observations non nulles.

 $\sigma^2 = 300$  est la variance des observations dans le voisinage spatial.

#### 4.21 Equation de contrainte du mouvement

L'équation de contrainte du mouvement ECM relie les dérivées spatiales et temporelles de l'image  $I_x, I_y, I_t$  aux composantes (u, v) du vecteur vitesse :

$$I_x u + I_y v + I_t = 0 (4.17)$$

Traduire cette équation en adoptant la notation vectorielle :

$$\vec{G} = \vec{\nabla}I = \begin{bmatrix} I_x \\ I_y \end{bmatrix}$$
(4.18)

$$\vec{v} = \begin{bmatrix} u \\ v \end{bmatrix} \tag{4.19}$$

En déduire l'interprétation vectorielle de cette équation (produit scalaire).

#### 4.22 Equation fréquentielle du mouvement

Démontrer que l'équation ECM fréquentielle :

$$u\omega_x + v\omega_y = -\omega_t \tag{4.20}$$

s'obtient par simple transformée de Fourier de l'ECM différentielle :

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$
(4.21)

Rappel de la propriété de dérivation de la TF : si  $TF[f(x)] = F(\omega)$  alors  $TF[df/dx] = i\omega F(\omega)$ .

## 4.23 Algorithme de Horn & Schunck

Pour l'implantation de l'algorithme itératif de Horn et Schunck, donner l'expression (voir Fig. 2.20, p. 52) :

— des dérivées spatiales et temporelles  $I_x, I_y, I_t$ ,

— des valeurs moyennes des composantes de vitesse  $\overline{u}, \overline{v}$ .

Quelles limitations peut-on en déduire concernant l'estimation des déplacements de zones mobiles (vitesse maximale, occultation)?

#### 4.24 Estimation par mise en correspondance de bloc

On considère les deux imagettes successives d'une séquence bruitée à NdG représentées sur la Fig. 4.13. Les pixels non représentés valent tous zéro.

On veut estimer le mouvement du bloc central  $3 \times 3$  (cf. zone mobile non nulle sur fond nul à l'instant t - 1) en utilisant comme critère d'erreur la SAD (somme des différences absolues) et en faisant une recherche exhaustive à l'intérieur d'une fenêtre de recherche centrée de taille  $5 \times 5$ .

1. Combien y a-t-il de vecteurs de déplacement possibles ?

- 2. Calculer, pour chaque vecteur de déplacement possible, la valeur correspondante de la SAD.
- 3. En déduire le vecteur de déplacement estimé par la méthode.
- 4. Indiquer les limites, avantages et inconvénients de cette méthode.



FIGURE 4.13 – Deux images successives d'une séquence bruitée.

# 4.25 Estimation d'un modèle de mouvement

On dispose des deux imagettes successives de la Fig. 4.14. On suppose qu'elles correspondent à une séquence sans bruit.



FIGURE 4.14 – Deux images successives d'une séquence non bruitée.

On veut estimer le vecteur déplacement en adoptant un modèle de mouvement à trois paramètres  $(a, b, c) : \begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} a + cy \\ b - cx \end{bmatrix}$ 

- 1. Quel est le nombre minimal de pixels à prendre en compte pour pouvoir estimer les paramètres ?
- 2. Calculer les paramètres de mouvement par estimation directe. On prendra l'origine de l'image au centre de l'objet (pixel de coordonnées (4,4) présenté en gras).
- 3. Qualifier le type de mouvement subi par l'objet mobile entre les deux instants.

# 4.26 Estimateur robuste

L'estimateur robuste de Geman et MacLure est défini par :

$$\rho(r) = \frac{r^2}{\sigma + r^2} \tag{4.22}$$

où  $\sigma$  est un coefficient de robustesse. On prendra ici :  $\sigma=1.$ 

Tracer la courbe  $\rho(r)$  et la comparer à l'estimateur quadratique. Expliquer l'intérêt de ce type d'estimateur.

## 4.27 Filtre de Canny-Deriche

La réponse impulsionnelle du filtre de Canny-Deriche monodimensionnel se décompose en une partie causale et une partie anti-causale :

$$h(x) = h^{+}(x) + h^{-}(x)$$
(4.23)

$$h^+(x) = cxe^{-\alpha x} \text{ pour } x \ge 0 \tag{4.24}$$

$$h^{-}(x) = cxe^{\alpha x} \text{ pour } x \le 0 \tag{4.25}$$

où  $\alpha$  est un coefficient de lissage et c un coefficient de normalisation.

On montre alors que la fonction de transfert en Z s'exprime par :

$$H(z) = H^{+}(z) + H^{-}(z)$$
(4.26)

$$H^{+}(z) = c \frac{e^{-\alpha} z^{-1}}{(1 - e^{-\alpha} z^{-1})^2}$$
(4.27)

$$H^{-}(z) = c \frac{-e^{-\alpha}z}{(1-e^{-\alpha}z)^2}$$
(4.28)

- 1. Démontrer l'expression de la fonction de transfert, en exprimant la TZ de la réponse impulsionnelle comme une série géométrique.
- 2. Dessiner l'allure de h(x) pour  $\alpha = 0.5$  et c = 1.
- 3. A partir des fonctions de transfert, retrouver les équations aux différences qui servent à l'implantation numérique du filtre.
- 4. De quel type de filtre s'agit-il? A quoi sert-il en traitement d'image?

## 4.28 Transformée couleur logarithmique

La transformée logarithmique fait passer des couleurs primaires (R, V, B) aux composantes logarithmiques  $(L_u, U_r, X_b)$ .

Ces composantes sont définies par les équations suivantes (où M = 256) :

$$L_y = (R+1)^{0.3} (V+1)^{0.6} (B+1)^{0.1} - 1$$
(4.29)

$$U_r = \frac{M}{2} \left( 1 + \frac{R+1}{L_y+1} - \frac{L_y+1}{R+1} \right)$$
(4.30)

$$X_b = \frac{M}{2} \left( 1 + \frac{B+1}{L_y+1} - \frac{L_y+1}{B+1} \right)$$
(4.31)

Cette transformation présente un intérêt en segmentation d'image, mais aussi en compression. Il faut alors disposer de la transformée inverse pour décompresser l'image.

- 1. Préciser la dynamique des composantes (R, V, B) pour une image en vraies couleurs (codée sur 24 bits).
- 2. Donner les équations de la transformée inverse permettant de retrouver (R, V, B) à partir de  $(L_y, U_r, X_b)$ .
- 3. Vérifier sur un triplet particulier (R, V, B) la validité des équations inverses.

\_

4. Donner au moins un argument en faveur de cette transformée, en rapport avec la perception des couleurs par le système visuel humain.

#### 4.29 Filtrage linéaire

Soit le filtre symétrique défini par le masque H, qui se décompose de la manière suivante :

\_

$$H = \begin{bmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{bmatrix} + 2 \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$
(4.32)

\_

1. Considérant que le filtre  $L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$  est séparable sous forme  $\begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix} + \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$ , montrer que sa fonction de transfert 2D vaut :

$$L(u,v) = 4 - 2\cos 2\pi u - 2\cos 2\pi v \tag{4.33}$$

où (u, v) sont les fréquences spatiales (resp. horizontale et verticale) réduites ( $\in [0; 1/2[)$ ).

- 2. Représentation graphique : Tracer le module |L(u, 0)| correspondant à la coupe en v = 0.
- 3. Calculer la fonction de transfert 2D H(u, v).
- 4. Interprétation : De quel type de filtre s'agit-il ? A quoi sert-il ?

Rappel : la forme générique d'une fonction de transfert 2D est :

$$H(u,v) = \sum_{k} \sum_{l} h(k,l) e^{-i2\pi(u.k+v.l)}$$
(4.34)

#### 4.30 Transformée couleur non-linéaire

On considère une transformée qui fait passer des couleurs primaires RVB (en anglais RGB pour red, green, blue) aux composantes dites logarithmiques  $(L_y, L_r, L_b)$  définies par les équations ci-dessous (où  $A = M^{0.3}$  et  $D = M^{0.1}$  avec M = 256):

$$L_y = (R+1)^{0.3} (V+1)^{0.6} (B+1)^{0.1} - 1$$
(4.35)

$$L_r = A(R+1)^{0.7}(V+1)^{-0.6}(B+1)^{-0.1} - 1$$
(4.36)

$$L_b = D(R+1)^{-0.3}(V+1)^{-0.6}(B+1)^{0.9} - 1$$
(4.37)

Si l'on veut utiliser cette transformée pour la compression d'images (et pas seulement pour faire de la segmentation couleur), il faut disposer de la transformée inverse pour décompresser l'image [126].

- 1. Calculer la dynamique (*i.e.* valeurs minimum et maximum) des composantes  $(L_y, L_r, L_b)$  pour une image RVB en vraies couleurs (codée sur 24 bits). Quel est le rôle des constantes A et D?
- 2. Montrer que la transformée inverse permettant de retrouver (R, V, B) à partir de  $(L_y, L_r, L_b)$  est donnée par les équations :

$$R+1 = (L_y+1)^{a_{11}} \left(\frac{L_r+1}{A}\right)^{a_{12}}$$
(4.38)

$$V+1 = (L_y+1)^{a_{21}} \left(\frac{L_r+1}{A}\right)^{a_{22}} \left(\frac{L_b+1}{D}\right)^{a_{23}}$$
(4.39)

$$B+1 = (L_y+1)^{a_{31}} \left(\frac{L_b+1}{D}\right)^{a_{33}}$$
(4.40)

où l'on déterminera la valeur des coefficients  $a_{ij}$ .

- 3. Pour le triplet particulier RVB = (200, 100, 50), calculer les valeurs transformées correspondantes :  $(L_y, L_r, L_b)$ . Puis vérifier la validité des équations inverses trouvées ci-dessus.
- 4. Donner au moins un argument en faveur de cette transformée, en rapport avec la perception des couleurs par le système visuel humain.
- 5. Donner au moins un inconvénient de cette transformation non linéaire, par rapport aux transformées linéaires usuelles du type YUV.

#### 4.31 Compensation de mouvement

Soit les deux portions d'images successives d'une séquence bruitée, représentées sur la Fig. 4.15.

1. Calculer les gradients spatio-temporels **moyennés**  $\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}, \frac{\partial I}{\partial t}\right)$  en les 4 pixels marqués en gras dans l'image à l'instant t: pour ce faire, procéder par application du masque  $\begin{bmatrix} 1 & -1 \end{bmatrix}$  et moyennage des 4 dérivées monodirectionnelles calculées dans un cube  $2 \times 2 \times 2$ , (comme pour l'algorithme de Horn & Schunck, cf. Fig. 2.20a, p. 52).



FIGURE 4.15 – Deux imagettes successives (pixels nuls non représentés par défaut).

- 2. Appliquer l'équation de contrainte du mouvement ECM (hypothèse de la DFD nulle) à ces 4 pixels. En déduire la vitesse estimée (composantes  $u = \frac{dx}{dt}$  et  $v = \frac{dy}{dt}$ ).
- 3. Réaliser la compensation de mouvement de la zone grisée entre les instants t et t + 1.
- 4. Calculer l'imagette d'erreur à transmettre en t + 1.
- 5. Conclure sur l'intérêt de cette technique dans un codec vidéo.

## 4.32 Teinte du visage

Pour le suivi automatique de locuteur, on utilise souvent l'information de teinte afin de segmenter l'image en 2 régions : le visage et le fond de la scène.

On acquiert alors une image couleur (*i.e.* 3 plans R, V, B chacun codé sur 8 bits). La luminance et la teinte peuvent être définies par :

$$L = \frac{R+V+B}{3} \tag{4.41}$$

$$T = \begin{cases} E \left[ 256\frac{V}{R} \right] & \text{si } R > V \\ 255 & \text{si } R \le V \end{cases}$$

$$(4.42)$$

où E[x] représente la partie entière de x.

- 1. Quelle est la dynamique de la teinte?
- 2. Sachant qu'un visage contient très peu de bleu  $(B \approx 0)$ , et que le plan vert est assez proche du plan luminance  $(V \approx L)$ , donner la valeur approchée de la teinte moyenne  $T_m$  caractérisant le visage.
- 3. Proposer alors une méthode simple de seuillage pour extraire de l'image la région du visage.
- 4. En se basant sur la courbe de visibilité de l'œil (Fig. 1.19 p. 30), donner un argument en faveur de l'approximation :  $V \approx L$ .

## 4.33 Spectres d'images

La Fig. 4.16 présente 8 images (notées de A à H) et 8 spectres [40].

- 1. Associer le bon spectre à chacune des images de la Fig. 4.16 en remplissant le tableau des spectres avec les bonnes lettres.
- 2. Justifier les associations.



FIGURE 4.16 - 8 images, 8 spectres.

## 4.34 Application industrielle

Proposer une technique pour restaurer un schéma électrique propre des pistes d'un circuit imprimé à partir d'une photocopie dégradée (Fig. 4.17) [57].



FIGURE 4.17 – Photocopie de schéma électrique.

#### 4.35 Résolution et contraste

La résolution R (exprimée en pixels) d'un capteur d'images et le contraste C dans une image sont définis par les équations :

$$R = 2 \times \frac{Fov}{SF} \text{ (unité : le pixel)}$$

$$(4.43)$$

$$C = \frac{I_B - I_D}{I_B + I_D} \tag{4.44}$$

Fov est le champ de vision (Field of view), SF est la taille de la plus petite structure des objets à observer (Smallest Feature), et  $I_B$  et  $I_D$  sont les intensités maximale (Brightest) et minimale (Darkest) dans l'image.

On considère les imagettes en NdG codées sur 8 bits et de taille  $5 \times 5$  représentées sur la Fig. 4.18.

	122	122	120	125	135	1	123	180	193	221	247	
	122	122	110	132	132		186	183	149	173	185	
(a)	130	120	120	125	132		255	208	89	143	136	(b)
	132	130	120	125	130		250	210	11	78	102	
	132	130	120	122	130		246	205	15	101	66	

FIGURE 4.18 – a) Imagette 1; b) Imagette 2.

- 1. Calculer pour chaque imagette le contraste correspondant.
- 2. Quelles techniques d'histogrammes peut-on utiliser pour améliorer le contraste d'une image?
- 3. Choisir une technique et la mettre en œuvre sur l'image la moins contrastée des deux. NB : Pour les calculs, arrondir les résultats à l'entier le plus proche.

- 4. Recalculer la valeur du contraste pour l'image ainsi améliorée. Comparer à l'image originale.
- 5. Conclure sur l'utilité du paramètre C. Quel est son domaine de variation?
- 6. Si l'on veut acquérir et traiter des images de codes-barres (étiquettes sur les produits du commerce) de 3,5 cm de long et dont les barres verticales sont d'épaisseur minimale 0,75 mm, quelle résolution minimale doit-on respecter pour le choix du capteur ?
- 7. Si l'on veut inspecter des pommes de largeur maximale 8 cm et de hauteur maximale 10 cm, ceci avec une précision de 1 mm, quelles sont les résolutions minimales en lignes et colonnes à respecter pour le choix du capteur?
- 8. Si l'on dispose d'un capteur de dimension  $128 \times 128$ , quelles précisions (*i.e.* valeurs de SF en mm) obtient-on dans les deux applications ci-dessus 6 et 7?

## 4.36 Filtrage médian

On donne l'imagette  $I_0$  de taille  $10 \times 10$  de la Fig. 4.19a dont les niveaux de gris vont de 0 à 15, et l'on considère le filtre médian 4-connexe (*i.e.*, en forme de croix centrée sur le pixel courant Fig. 4.19b)<sup>1</sup>.

0										
	15				15					
					14					
	13	15								
			13	10	11	12				
			12	15	10	11				
			11	10	11	15			b)	
	14		12	10	10	11				
								<b>a</b> )		

FIGURE 4.19 – a) Imagette  $I_0$ ; b) Croix 4-connexe.

NB : tous les pixels dessinés en blanc sont d'intensité égale à zéro.

Rappel : le filtrage médian consiste à trier par valeurs croissantes les intensités des pixels du voisinage considéré (ici la croix), à les ranger dans un tableau 1D (ici de taille  $1 \times 5$ ) et à retenir la valeur médiane (c'est-à-dire celle qui est classée au milieu du tableau 1D).

1. Filtrer l'image par ce filtre médian. On appeller<br/>a ${\cal I}_M$ le résultat.

NB : pour gérer les effets de bord, on supposera que les pixels manquants sont d'intensité nulle.

- 2. Que deviennent les points isolés et les petits groupes de 2 pixels?
- 3. Que devient l'objet carré de plus grande taille?
- 4. Commenter les effets et l'intérêt de ce type de filtre, en extrapolant au cas d'une image bruitée de grande taille contenant divers objets.
- 5. A quelle famille de filtres appartient ce filtre?

<sup>1.</sup> Les 3 exercices des sections 4.36, 4.37 et 4.38 forment un tout qui peut constituer les 3 questions d'un même sujet global.

## 4.37 Filtrage linéaire

On récupère l'image  $I_M$  résultant du filtrage médian précédent. On lui applique un filtre linéaire de masque  $H = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$  horizontalement.

- 1. De quel type de filtre s'agit-il? Quel est son rôle?
- 2. Dessiner le résultat du filtrage de  $I_M$  par H, qu'on appellera  $I_H$ .
- 3. On applique le même filtre sur l'image  $I_M$ , mais dans la direction verticale : c'est-à-dire avec le  $\begin{bmatrix} -1 \end{bmatrix}$

masque  $V = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ . Donner le résultat du filtrage de  $I_M$  par V, qu'on appellera  $I_V$ .

4. On calcule maintenant l'image  $I_G = \sqrt{I_H^2 + I_V^2}$ . Dessiner le résultat.

NB : on arrondira les racines carrées à l'entier le plus proche (cf. Tab. 4.2 p. 118).

- 5. On va appliquer un seuil  $\theta$  (bien choisi) sur  $I_G$ . On obtient ainsi une image binaire qu'on notera  $I_B$ . Dessiner le résultat du seuillage après avoir justifié le choix d'une bonne valeur du seuil.
- 6. Commenter l'intérêt de ce traitement, en extrapolant au cas d'une image bruitée de grande taille contenant des objets également de grande taille et d'intensité assez uniforme.
- 7. Sans faire de calculs, comparer à ce qu'on aurait obtenu si l'on avait directement appliqué les filtres H et V sur l'image initiale  $I_0$ .

## 4.38 Codage de contour

On considère désormais l'image binaire  $I_B$  résultant du seuillage précédent.

- 1. Indiquer le nombre  $N_1$  de bits informatiques *a priori* nécessaires à son stockage dans l'ordinateur.
- 2. On veut appliquer à la forme binaire obtenue l'algorithme de codage directionnel de Freeman, en partant du pixel le plus en haut à gauche appartenant à la forme détectée. Préciser les coordonnées de ce pixel initial, l'origine de l'image étant prise en haut à gauche. Combien de bits sont nécessaires pour coder l'information sur la position de ce pixel dans l'image?
- 3. Donner ensuite le code complet obtenu (point initial + codage des directions successives).
- 4. Calculer le nombre  $N_2$  de bits nécessaires à la transmission de cette information codée.
- 5. Comparer  $N_2$  à  $N_1$  et conclure sur l'intérêt de ce codage.
- 6. Calculer l'entropie H(B) (en bits/pixel) de la source d'information que constitue l'image  $I_B$ , ainsi que sa redondance définie par :  $R = 1 \frac{H(B)}{H_{max}}$ .
- 7. Si l'on mettait en œuvre un codage entropique de la source du type Shannon-Fano ou Huffman, combien de bits d'information pourraient suffire à la transmission de  $I_B$ ?
- 8. On note  $N_3$  ce nombre de bits. Comparer  $N_3$  à  $N_2$ . Quel est le codage le plus intéressant de cette image  $I_B$ : codage directionnel ou codage entropique?

## 4.39 Filtrage linéaire

On considère les contours verticaux en échelon et en rampe de la Fig. 4.20a où : b > a et  $c = \frac{a+b}{2}$ . On notera h = b - a.

- 1. Soit le masque de filtrage  $M_1 = \begin{bmatrix} -1 & 1 \end{bmatrix}$  (coefficient du pixel courant à droite). Dessiner le résultat du filtrage d'une ligne pour les deux types de contours (échelon et rampe).
- 2. Même question avec le masque de filtrage  $M_2 = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$  (coefficient du pixel courant au centre).
- 3. Comparer ces deux filtres (avantages et inconvénients) sur les deux types de contours : localisation, sensibilité au bruit ...

9	9	9	9	9	h	h	h	h	h									
a	a	a	a	a	D	D	D	D	D	0							100	
a	a	a	a	a	b	b	b	b	b	0							100	
a	a	a	a	a	b	b	b	b	b	0	0			0				
$\mathbf{a}$	а	а	а	а	b	b	b	b	b	 - V								
a	a	a	a	a	b	b	$\mathbf{b}$	b	b									
co	ntc	our	ve	ert	ica	l e	n e	éch	elon							100	100	
								1			100	100						
$\mathbf{a}$	a	a	a	С	b	b	b	b	b		100	100						
a	а	а	а	С	b	b	b	b	b		100	100						
a	а	а	а	С	b	b	b	b	b						10		0	
$\mathbf{a}$	a	а	а	С	b	b	b	b	b	00			100					
a	а	а	а	С	b	b	b	b	b	90			100					
co	ntc	our	ve	$\operatorname{ert}$	ica	l e	nı	an	npe									
									-									

FIGURE 4.20 – a) A gauche : contours verticaux; b) A droite : imagette.

On considère l'imagette de la Fig. 4.20b, où tous les pixels du fond (en blanc) valent 50. On la filtre avec le masque de filtrage  $M = \frac{1}{8} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$  et on ne retient que les pixels dont le résultat R du filtrage vérifie :  $|R| > \theta$ , où  $\theta = 45$  est un seuil *ad hoc* fixé *a priori*.

1. Représenter le résultat du filtrage (ne pas traiter les bords de l'image). N.B. : arrondir les calculs à la première décimale.

2. Représenter le résultat du seuillage : dessiner en noir les pixels retenus par la procédure.

3. Quel est le rôle de ce filtre (effet sur les zones uniformes, les groupes de pixels, les pixels isolés)?

## 4.40 Remplissage morphologique

On considère la forme binaire A et l'élément structurant E de la Fig. 4.21 [60]. Les pixels du fond blanc valent 0 et ceux de la forme grise A valent 1.



FIGURE 4.21 – a) Forme binaire (en gris) correspondant par exemple à la version scannée d'un « a » manuscrit; b) Elément structurant (croix centrée).

Soit un point initial intérieur à la forme (point p sur la figure). On lui attribue la valeur 1 et on applique la procédure suivante de dilatations itérées :

$$X_k = (X_{k-1} \oplus E) \cap A^c \qquad k = 1, 2, 3, \dots$$
(4.45)

où  $A^c$  représente le complémentaire de A et  $X_0 = \{p\}$ .

- 1. Dessiner d'abord le complémentaire de A.
- 2. Représenter les résultats  $X_k$  à chaque itération jusqu'à convergence.
- 3. Quel est le nombre N d'itérations jusqu'à la convergence (c'est-à-dire quand  $X_{k+1} = X_k$ ,  $\forall k \ge N$ )?
- 4. Le résultat est l'ensemble constitué de l'union de  $X_N$  et de A. Quel est le rôle de ce traitement?

#### 4.41 Analyse de mouvement

Soit une séquence d'images en niveaux de gris I(t) et une image de référence  $I_{ref}$ . On suppose que la séquence comporte un objet mobile (carré de taille  $5 \times 5$ ) d'intensité lumineuse plus forte que le fond, et se déplaçant dans la direction sud-est à la vitesse  $v = (v_x, v_y) = (2 \text{ pix/img}, 1 \text{ pix/img})$ . L'image de référence est la première image de la séquence correspondant à l'instant initial  $t_0 = 0$  (on suppose qu'elle contient tout l'objet mobile, cf. Fig 4.22). L'origine spatiale de l'image est en haut à gauche.



FIGURE 4.22 – Image de référence  $I_{ref}$  (à  $t_0 = 0$ ).

L'image des différences positives accumulées (PADI en anglais [60]) est définie à l'instant t en chaque site-pixel s = (x, y) de la façon suivante :

$$PADI(s,t) = PADI(s,t-1) + d(s,t)$$

$$(4.46)$$

où 
$$d(s,t) = \begin{cases} +1 & \text{si } I_{ref}(s) - I(s,t) > \theta > 0\\ 0 & \text{sinon.} \end{cases}$$

$$(4.47)$$

En chaque pixel, un compteur est donc incrémenté à chaque fois qu'une **différence positive** est détectée entre la référence et le pixel courant ( $\theta$  symbolisant la valeur de seuil utilisée). Au départ  $(t_0 = 0)$ , on a :  $\forall s$ ,  $PADI(s, t_0) = 0$ 

- 1. Représenter les 5 imagettes PADI correspondant aux 5 instants successifs t = 1 à t = 5.
- 2. Commenter ce qui se passe à partir de t = 3.
- 3. Expliquer l'intérêt de ce traitement en analyse du mouvement (détection d'objet mobile et estimation de vitesse).

#### 4.42 Filtre médian

On donne ci-dessous l'algorithme dit du filtrage médian, qui lui-même repose sur un algorithme de tri (classement des intensités des pixels dans un voisinage donné).

```
Balayage vidéo de l'image (x,y) (sauf les bords)
k=0 (indice du tableau monodimensionnel intermédiaire tab)
```

```
Balayage vidéo du voisinage 3x3:
Coordonnées relatives (i,j) où i,j varient dans {-1;0;1}
- calcul des coordonnées réelles du pixel voisin (x+i,y+j)
- tab[k]=I(x+i,y+j) /*entrée des valeurs dans un tableau*/
- k++
Classement des valeurs du tableau tab (algo du tri à bulles)
Med(x,y)=tab[5] /*valeur médiane du tableau*/
```

On rappelle le principe du tri à bulles à l'aide de l'algorithme ci-après :

```
Balayer le tableau,
en considérant l'élément courant et l'élément suivant
Si l'élément courant est supérieur à l'élément suivant
-transposer l'élément courant et l'élément suivant
Réitérer le balayage
s'il y a une transposition au cours du balayage précédent.
```

On considère d'autre part l'imagette en NdG de la Fig. 4.23 comportant quelques objets de forme différente (rectangulaire, linéaire, ponctuelle).

0	0						
0	100		10	10	10	10	
		0	10	0	10	10	
		0	10	10	10	10	
	50	0					
	20	20	20	20	20		

FIGURE 4.23 – Imagette en NdG (les pixels non renseignés valent zéro par défaut).

- 1. Préciser la taille du tableau 1D intermédiaire qui servira au tri.
- 2. Dérouler l'algorithme du filtre médian sur l'imagette fournie et dessiner l'imagette résultat du filtrage.
- 3. Interpréter le rôle du filtre en détection (action sur les trous, les points isolés, les traits horizontaux, les angles etc.)
- 4. Que donnerait ce filtrage sur un trait vertical?
- 5. Quelle est la taille des trous et des taches éliminables dans une image bruitée ? De quoi est-elle fonction ?
- 6. A quelle famille appartient ce type de filtre?

## 4.43 Codage de contour

Soit l'imagette bruitée de taille  $10 \times 10$  de la Fig. 4.24 comportant 8 niveaux de gris (NdG) du noir « = 0 » au blanc « = 7 » <sup>2</sup>.

1. Combien de bits informatiques sont nécessaires pour coder un pixel? Quelle est la taille mémoire nécessaire pour stocker l'image sur disque?

<sup>2.</sup> Les 3 exercices des sections 4.43, 4.44 et 4.45 forment un tout qui peut constituer les 3 questions d'un même sujet global.

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

FIGURE 4.24 – Imagette bruitée d'une forme triangulaire.

2. Calculer les probabilités  $p_i$  de chaque niveau de gris (en laissant les  $p_i$  sous forme fractionnaire). En déduire l'entropie H de la source d'information correspondant à cette image.

N.B. : en l'absence de calculatrice, on se référera au Tab. 4.3 page 118 pour les valeurs des logarithmes binaires, ainsi qu'aux propriétés usuelles du logarithme :  $\log(x/y) = \log(x) - \log(y)$ ;  $\log(x^a) = a \log(x)$  etc.

- 3. Quelle serait l'entropie maximale  $H_{max}$  de cette source? En déduire la redondance R.
- 4. Tracer l'histogramme de l'image. Proposer une valeur de seuil  $\theta$  pour binariser cette image en 2 niveaux : noir et blanc.
- 5. Quelle sera alors la taille sur disque nécessaire pour stocker l'image binaire?
- 6. Appliquer l'algorithme de Rosenfeld et Kak pour suivre le contour binaire obtenu (en partant du pixel de l'objet le plus en haut et à gauche).
- 7. Calculer le nombre total de bits nécessaires au codage du contour, et donc la taille mémoire sur disque pour stocker l'image binaire.
- 8. En comparant à l'image initiale, en déduire le taux de compression obtenu grâce à ce codage.

#### 4.44 Filtrage linéaire : filtre de Sobel

On filtre l'imagette présentée Fig. 4.24 avec le filtre de masque :

$$M = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

- 1. Exprimer ce filtre sous forme séparable en 2 filtres 1D selon les lignes et colonnes.
- 2. Calculer et tracer les fonctions de transfert des 2 filtres 1D qui le constituent (en module uniquement, et en fonction de la fréquence réduite u).
- 3. Commenter le type et le rôle de ces 2 filtres. Que l est l'intérêt de leur association dans le filtre  $M\,?$
- 4. Calculer l'imagette filtrée par le filtre de masque M.

NB : Ne pas traiter les rangées de pixels sur les bords de l'image (mettre simplement à zéro).

- 5. Proposer une valeur de seuil pour binariser le résultat. Que détecte-t-on?
- 6. Obtient-on le contour horizontal? Sinon, proposer une solution pour détecter aussi ce contour.

## 4.45 Gradient morphologique

On veut comparer le filtre linéaire de la section 4.44 au filtre non-linéaire correspondant au gradient morphologique avec l'élément structurant fait d'un segment horizontal centré de largeur 3 pixels.

- 1. Rappeler l'expression du gradient morphologique G pour images en NdG.
- 2. Calculer la dilatée, l'érodée et enfin le résultat du gradient morphologique de l'imagette Fig. 4.24.
- 3. Proposer une valeur de seuil pour binariser le résultat.
- 4. Comparer au résultat du filtre linéaire et commenter.

## 4.46 Codage entropique de Huffman

Soit l'image de taille  $9 \times 11$  de la Fig. 4.25 codée sur 4 niveaux de gris seulement (blanc, gris clair, gris foncé, noir). NB : Elle contient 99 pixels, mais pour simplifier les calculs, on pourra arrondir à 100 le nombre de pixels.



FIGURE 4.25 – Image affichant 4 caractères ASCII.

- 1. A priori, que vaut l'entropie maximale  $H_{max}$  de cette source d'information (en bits/pixel)?
- 2. Calculer l'entropie  $H_0$  et la redondance  $R_0$  dans l'image initiale (avant codage entropique).
- 3. Réaliser le codage entropique (de Huffman ou Shannon-Fano) de cette image. Pour cela, construire un tableau contenant les 4 messages  $M_i$  (*i.e.* 4 niveaux de gris), leurs probabilités  $P_i$  (approximées par leur fréquence d'apparition dans l'image), les mots-codes  $C_i$  et les longueurs de mots correspondantes  $L_i$ .
- 4. Calculer la longueur moyenne  $L_{moy}$  d'un mot-code. En déduire le taux de compression  $\tau$  obtenu par ce codage entropique à longueur variable.
- 5. Calculer la probabilité des symboles « 0 » et « 1 » en sortie du codeur.
- 6. En déduire l'entropie  $H_c$  et la redondance  $R_c$  après codage. Comparer aux valeurs avant codage et commenter le résultat obtenu.
- 7. Décoder le message véhiculé par cette image source d'information visuelle qui affiche quatre caractères parmi des chiffres et des lettres. Calculer la quantité d'information moyenne (entropie) de ce message textuel, en supposant équiprobables tous les caractères (les 26 lettres de l'alphabet et les 10 chiffres). Commentaire? (comparer avec les premières questions sur  $H_{max}$  et  $H_0$ ).
- N.B.: Le tableau des logarithmes binaires (Tab. 4.3 page 118) permet de travailler sans calculatrice.

## 4.47 Effet 2D d'un filtre RIF 1D

Soit le filtre RIF centré de masque  $H = \frac{1}{5}\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$ .

1. Représenter le résultat du filtrage de l'image de la Fig. 4.26 par ce filtre.

0												0
		0	0	0	0	0	0	0	0	0		
		0	100	100	100	100	100	100	100	0		
		0	100	100	100	100	100	100	100	0		
		0	100	100	100	100	100	100	100	0		
		0	100	100	100	100	100	100	100	0		
		0	0	0	0	0	0	0	0	0		
0												0

FIGURE 4.26 – Imagette d'un rectangle (tous les pixels non spécifiés valent zéro).

- 2. Calculer la fonction de transfert H(u) du filtre, où  $u = \nu/Fe$  est la fréquence réduite telle que  $0 \le u \le 0.5$  si l'on ne considère que les fréquences positives qui respectent le théorème de Shannon.
- 3. Rappeler à cette occasion ce que stipule ce théorème et quels phénomènes apparaissent si on ne le respecte pas.
- 4. Représenter graphiquement l'allure de la courbe de module |H(u)| en fonction de u.
- 5. De quel type de filtre s'agit-il : passe-bas, passe-haut, passe-bande, coupe-bande, réjecteur, passe-tout...? Commenter ses qualités et défauts.
- 6. Quel est le nom de l'effet principal produit par ce filtre : flou isotrope, détecteur de contour, rehaussement de contraste, lissage de bruit, bougé horizontal, bougé vertical...?
- 7. Que donnerait le filtre sur l'image de la Fig. 4.35 présentée à la p. 107? Dessiner le résultat.

#### 4.48 Filtrage non-linéaire

On considère l'imagette binaire de la Fig. 4.27 obtenue par un algorithme de seuillage sur la teinte d'un visage, exhibant la bouche, les oreilles, les narines et les yeux.

N.B. : L'origine de l'image est située en haut à gauche.

On lui applique le filtre non-linéaire suivant [9] :

$$M(i) = \begin{cases} 1 + \sum_{j \in V(i)} a(j)M(j) & \text{si } U(i) \in \text{ spins } \\ 0 & \text{sinon} \end{cases}$$
(4.48)

où *i* est l'indice du pixel courant, U(i) sa valeur de NdG (blanc=« fond », gris=« forme »), où M est l'image résultat (initialisée à 0) et où les a(j), avec  $j \in V(i)$ , sont les 4 coefficients rangés dans la matrice  $A = \begin{bmatrix} 1 & 1 & 1 \\ 5 & i \end{bmatrix}$  pondérant les 4 voisins j du pixel courant i restreints au voisinage causal V(i) comme indiqué dans la matrice (*i.e.* les 4 plus proches voisins précédant le pixel courant).

- 1. Calculer l'imagette filtrée en remplissant la Fig. 4.27 avec les valeurs calculées.
- 2. Indiquer les coordonnées (ligne, colonne) du point  $P_{max}$  portant la valeur maximum après filtrage.
- 3. Appliquer l'algorithme de suivi de contour de formes binaires de Rosenfeld et Kak en prenant comme point initial le point  $P_{max}$  trouvé ci-dessus et comme direction initiale l'Est  $(d_0 = 0)$ . Donner le code obtenu.

FIGURE 4.27 – Image binaire : fond en blanc, formes en gris.

- 4. Quel contour détecte-t-on ainsi dans l'imagette ? En déduire le rôle de ce filtre.
- 5. Calculer le taux de compression obtenu grâce au codage par l'algorithme de R&K, ceci uniquement sur la demi-image (de taille 8 × 16) contenant la forme détectée.
- 6. Mesurer la compacité C de chacune des formes présentes dans l'image sachant que :  $C = 4\pi \frac{S}{P^2}$ où S est la surface occupée par un objet (exprimée en nombre de pixels) et P est le périmètre de la forme (également exprimé en nombre de pixels).
- 7. La compacité est une mesure indiquant si une forme est compacte ou non. La compacité maximale théorique est obtenue pour le disque (maximum de surface entourée par un périmètre minimum).

Comparer la compacité de la forme détectée par l'algorithme de R&K avec celle d'un œil.

#### 4.49 Etiquetage en composantes connexes

On se propose d'étudier un algorithme [39] qui, à partir d'une image binaire I, génère une image d'étiquettes E. L'image binaire I de départ peut, par exemple, résulter d'une détection de mouvement ou de teinte.

Les pixels p de l'image I valent donc :  $I(p) = \langle 0 \rangle$  pour le FOND noir, ou bien  $I(p) = \langle 1 \rangle$  pour les OBJETS en blanc. A l'issue du traitement par l'algorithme, chaque pixel p se verra affecter une étiquette E(p).

#### 4.49.1 Travail demandé

- 1. Dérouler l'algorithme sur l'imagette binaire de la Fig. 4.28a et, **simultanément**, remplir **au fur et à mesure la table** d'équivalence des étiquettes (Fig.4.29) : chaque colonne de cette table correspond aux mises à jour successives des équivalences d'une étiquette. La première ligne correspond à l'état initial de la table.
- 2. Représenter sur une grille vierge telle celle de la Fig.4.28b, l'image des étiquettes résultant du **premier balayage**.
- 3. Quel est le **nombre maximal** d'étiquettes générées par l'algorithme?
- 4. Représenter, sur une autre grille, l'image finale des étiquettes après le second balayage.
- 5. Conclure sur la **fonction** réalisée par cet algorithme, et son intérêt pratique en traitement d'image.
- 6. Si l'on suppose que chaque étiquette utile est associée à une couleur distincte (par exemple rouge, vert, bleu, jaune, turquoise, mauve, blanc, noir, marron, gris, orange, etc.), illustrer (en coloriant) ce que donnera l'algorithme sur les **trois images** de la Fig. 4.30.

N.B. Les grilles vierges supplémentaires de la Fig.4.28 pourront servir de brouillon lors du déroulement de l'algorithme.

#### 4.49.2 Algorithme

L'algorithme, décrit ci-après, procède en 2 balayages séquentiels (ligne par ligne de haut en bas et de gauche à droite) et gère simultanément une table T d'équivalence entre étiquettes, table qui évolue au cours du balayage.

On considère pour chaque pixel p uniquement ses 2 voisins causaux v : c'est-à-dire le voisin nord et le voisin ouest (v = n ou v = o).

#### $\underline{Initialisation}$ :

- Les **bords extérieurs** de l'image (*i.e.* les voisins v manquants) sont supposés appartenir au FOND (I(v) = 0).
- L'image E des étiquettes est initialisée à la valeur « 0 » qui représente par convention le FOND : pour tout p, E(p) = 0.
- k = 1 : **numéro initial** de nouvelle étiquette.
- La table d'équivalence entre étiquettes est initialisée à l'identité : pour tout k,  $\mathbf{T}[\mathbf{k}] = \mathbf{k}$

#### 1<sup>er</sup> balayage (sur l'image) :

Récurrence sur les pixels de l'objet uniquement :

```
POUR tout pixel p appartenant à un objet (I(p)=1) FAIRE:
 SI ses 2 voisins v appartiennent au FOND (I(v)=0) ALORS:
  E(p) = k (nouvelle étiquette)
  k = k+1
 SINON
  SI ses 2 voisins ont 1 étiquette identique: pour tout v, E(v)=e
  (c'est-à-dire: E(n) = E(o) = e)
                                     ALORS:
  E(p) = e
  SINON
   e = minimum \{ T[E(n)], T[E(o)] \}
   (où E(n) et E(o) sont les étiquettes des 2 voisins)
   SI E(o) = 0
                ALORS:
                          e = T[E(n)]
   SI E(n) = 0
                 ALORS:
                          e = T[E(o)]
   (gestion de cas particulier où un voisin est à 0)
  E(p) = e
  POUR chaque voisin non nul (nord et ouest)
```

1	1	1	0	1	1	0	1					
1	1	0	1	1	0	1	1					
1	0	1	1	0	1	1	1					
1	1	1	0	1	1	1	1					
0	0	0	1	1	1	1	1		 			
1	1	1	1	1	1	1	1					
1	1	1	1	1	1	1	1					
1	1	1	1	1	1	1	1					
								]				

FIGURE 4.28 – Imagette binaire à traiter et grille vierge.

```
d'étiquette b telle que T[b] != e, FAIRE:
TANT QUE T[b] != e FAIRE:
tmp = T[b]
T[b] = e (mise à jour de la table d'équivalence)
```







FIGURE 4.30 – Capture d'images réelles.

b = tmp FIN TANT QUE.

**N.B.** : A l'issue de ce 1<sup>er</sup> balayage, chaque pixel est affecté d'une étiquette **provisoire**.

```
Actualisation de la table d'équivalence (à l'issue du 1<sup>er</sup> balayage) :
```

```
POUR k = 1 jusqu'à Nb maximum d'étiquettes générées par le
premier balayage FAIRE:
  m=k
TANT QUE T[m] != m FAIRE:
  m = T[m]
```

```
FIN TANT QUE.
T[k] = m
```

#### 2<sup>e</sup> balayage (sur l'image des étiquettes) :

On affecte à chaque pixel p, d'étiquette provisoire E(p) = k, une étiquette **définitive** correspondant à celle lue dans la table actualisée :

```
POUR tout pixel p d'étiquette E(p) = k FAIRE:
E(p) = T[k]
```

## 4.50 Programmation OpenCV

Soit le code C ci-dessous utilisant la structure image d'OpenCV.

```
void traitement(IplImage* img) {
  register int l,c,i;
  int haut=img->height; //nb de lignes dans l'image.
  int larg=img->width; //nb de pixels par ligne.
  int plan=img->nChannels; //nb de canaux couleur par pixel.
  int step=img->widthStep; //nb d'octets constituant une ligne.
  double pix;
  unsigned char *pin=(unsigned char *)(img->imageData);
    //pointeur sur la 1ère ligne
  for(l=1;l<haut;l++){</pre>
     for(c=1;c<larg;c++){</pre>
        for(i=0;i<plan;i++){</pre>
           pix=pin[(l-1)*step+c*plan+i]+pin[l*step+(c-1)*plan+i]
                 -4*pin[l*step+c*plan+i]+pin[l*step+(c+1)*plan+i]
                 +pin[(l+1)*step+c*plan+i];
           pin[l*step+c*plan+i]=(unsigned char)pix;
        }
     }
  }
}
```

- 1. Quel traitement d'image réalise cette fonction ?
- 2. A quoi sert-il?
- 3. Citer les 2 ou 3 problèmes les plus classiques que l'on risque de rencontrer quand on programme des algorithmes de traitement d'image.
- 4. Y a-t-il dans le code source fourni des erreurs ou problèmes?
- 5. Si oui, expliquer lesquels et les corriger.

## 4.51 Détection de contour

On considère [40] l'imagette de la Fig. 4.31 et le filtre de masque  $\begin{bmatrix} -1 & 1 \end{bmatrix}$ , approximation discrète de la dérivée première. (N.B. : le coefficient de droite est affecté au pixel courant).

- 1. Calculer les deux composantes du gradient (Gx, Gy), son module |G| et sa phase  $\Phi$  pour chaque pixel de l'imagette  $(-\pi \leq \Phi \leq \pi)$ .
- 2. Choisir un seuil adéquat  $\theta$  à appliquer sur le module du gradient pour obtenir les contours.

0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	2	3	2	0	0
0	0	2	3	4	3	0	0
0	0	5	4	4	3	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

FIGURE  $4.31 - \text{Imagette } 8 \times 8 \text{ en NdG}$ .

- 3. Représenter le résultat du seuillage du module du gradient  $|G| \ge \theta$  et commenter le résultat.
- 4. Alternative : refaire le même exercice, mais avec le filtre centré de masque  $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ .
- 5. Tester et commenter l'influence du seuil dans les 2 cas suivants :  $\theta = 3$  et  $\theta = 4$ .
- 6. Commenter l'information portée par la phase du gradient pour les pixels détectés comme contours.

## 4.52 Morphologie mathématique

Soit l'objet binaire X et les éléments structurants E et F de la Fig. 4.32 [60].



FIGURE 4.32 – Imagette binaire et éléments structurants.

- 1. Donner le résultat Y de l'opération  $\beta(X) = X (X \ominus E)$ , où  $\ominus$  représente l'érosion.
- 2. De quel opérateur s'agit-il?
- 3. Appliquer au résultat Y obtenu l'opération itérée avec l'élément structurant F:

$$Y_k = (Y_{k-1} \oplus F) \cap Y^c \qquad k = 1, 2, 3, ..., K$$
(4.49)

où  $\oplus$  représente la dilatation,  $Y^c$  le complémentaire, et en partant d'un ensemble initial  $Y_0 = \{p\}$  où p est un point quelconque intérieur à Y (par exemple celui proposé sur la Fig. 4.32). 4. Combien d'itérations K faut-il pour converger?

5. Que représente l'ensemble  $Y_K \cup Y$ ? Que lest le rôle de ce deuxième opérateur?

## 4.53 Codage entropique

Soit une image numérique à 6 niveaux de gris. On considère cette image comme une source de messages  $X = \{m_1 \cdots m_6\}$ , dont les probabilités notées  $P = \{p_1 \cdots p_6\}$  valent respectivement :

$$p_1 = \frac{3}{8} = 0.375$$
  $p_2 = \frac{1}{6} \approx 0.167$   $p_3 = p_4 = p_5 = \frac{1}{8} = 0.125$   $p_6 = \frac{1}{12} \approx 0.083$ 

- 1. A priori, combien faut-il de bits informatiques pour coder chaque message émis par cette source d'information (*i.e.*, chaque niveau de gris de l'image)?
- 2. Réaliser le codage de Huffman (ou de Shannon-Fano) de cette source.
- 3. Calculer la longueur moyenne d'un mot-code.
- 4. Evaluer le taux de compression ainsi obtenu.
- 5. S'agit-il d'une compression avec perte ou sans perte?
- 6. Quelle est la valeur de  $H_{max}$ , entropie maximale de la source X?
- 7. Calculer l'entropie et la redondance de X avant codage.
- 8. Recalculer ces deux grandeurs après le codage de Huffman. Conclure sur l'intérêt du codage.

**Indication** : pour ce faire, on pourra considérer la source codée comme une nouvelle source émettant 2 messages (les deux symboles  $\ll 0 \gg$  et  $\ll 1 \gg$ ) dont on calculera au préalable les probabilités respectives.

NB: Pour travailler sans calculatrice, se reporter au tableau des log binaires : Tab. 4.3, page 118.

## 4.54 Opérations morphologiques

Soit l'imagette binaire de la Fig. 4.33a (objet gris sur fond blanc) et quatre éléments structurants centrés  $E_1, E_2, E_3, E_4$  (site gris = objet; site blanc = indifférent).



FIGURE 4.33 – a) Forme binaire (en gris); b) Quatre éléments structurants.

- 1. Représenter les 4 résultats de dilatation par chacun des 4 éléments structurants.
- 2. Représenter les 4 résultats d'érosion par chacun des 4 éléments structurants.

## 4.55 Morphologie mathématique

On considère l'imagette binaire et l'élément structurant E de la Fig. 4.34. L'image contient deux objets (A et B) et quelques points isolés (C). On suppose que le point initial p de l'objet A est connu.

On applique alors la formule itérative suivante [60]:

$$X_k = (X_{k-1} \oplus E) \cap A$$
  $k = 1, 2, 3...$  (4.50)

où  $X_0 = \{p\}$  et  $\oplus$  symbolise la dilatation par l'élément structurant E. L'algorithme a convergé quand  $X_k = X_{k-1}$ .

1. Représenter le résultat de la première itération.



FIGURE 4.34 - a) Forme binaire (en gris sur fond blanc); b) Elément structurant E (carré centré).

- 2. Représenter le résultat de la deuxième itération.
- 3. Représenter le résultat final après convergence.
- 4. Quelle est le nombre d'itérations nécessaires pour converger (valeur de k à la convergence)?
- 5. Quelle est la nature et l'intérêt de l'opération ainsi réalisée?

## 4.56 Filtrage linéaire

1. Sachant qu'une dérivée première discrète 1D s'approxime par une simple différence entre pixels adjacents, donner le masque M (de taille  $2 \times 2$ ) du filtre approximant la dérivée seconde croisée de l'intensité lumineuse I:

$$\frac{\partial^2 I}{\partial x \partial y} = \frac{\partial}{\partial x} \left( \frac{\partial I}{\partial y} \right) \tag{4.51}$$

On précisera la position conventionnelle choisie pour le pixel courant (par exemple : en bas à droite dans le masque).

- 2. Appliquer ce filtre aux deux imagettes a) et b) de taille  $8 \times 8$  de la Fig. 4.2 fournie en p. 77.
- 3. Commenter l'action du filtre. Que permet-il de détecter?
- 4. Dans le cas d'une image de taille standard CIF (exemple de la Fig. 4.35), que donnerait ce filtre appliqué à un objet de forme circulaire, à une ligne horizontale, à une ligne verticale, à une ligne diagonale, etc.? Illustrer la réponse en dessinant le résultat pour l'image de la Fig. 4.35.
- 5. Comparer aux détecteurs classiques (gradient et laplacien) et conclure sur l'intérêt éventuel de ce filtre (avantages, inconvénients).



FIGURE 4.35 – Image CIF comportant diverses formes géométriques.

## 4.57 Estimation de mouvement

L'estimateur de mouvement de Horn & Schunck a fourni le flux optique de la Fig. 4.36. Il correspond au déplacement d'un carré sombre d'intensité uniforme sur un fond clair également uniforme.



FIGURE 4.36 – Carré en mouvement et flux optique estimé (sur 2 images successives).

- 1. Présenter un organigramme de l'algorithme de Horn & Schunck.
- 2. Rappeler en quoi consiste le problème d'ouverture.
- 3. Interpréter l'orientation des vecteurs vitesses obtenus par l'algorithme (sur les côtés et dans les angles du carré).
- 4. En déduire la direction du déplacement global de l'objet.
- 5. Les vecteurs obtenus sur les côtés horizontaux et verticaux du carré ayant un module égal à 1, en déduire la vitesse estimée par l'algorithme.

## 4.58 Résolution d'une caméra linéaire

Pour inspecter un matériau plan en défilement continu [67] (production de tissu, papier, tôles etc.), on utilise une caméra monochrome linéaire, c'est-à-dire constituée d'une seule rangée de photosites ou barrette de pixels (Fig. 4.37a). Le nombre de pixels de la barrette définit la résolution transversale de la caméra. Une grande résolution permet des contrôles dimensionnels précis. Pour déterminer la résolution transversale R, on divise la dimension du champ de visée Fov (Field of view) par la moitié de la taille x du plus petit détail à détecter (car pour une bonne perception visuelle, il faut que le plus petit détail soit vu par au moins deux pixels voisins).

La résolution longitudinale est fonction de la vitesse de déplacement relative entre l'objet observé et la caméra, et de la fréquence d'acquisition des lignes successives. Pour déterminer la fréquence d'acquisition  $F_a$ , on divise la vitesse relative V du déplacement de l'objet par rapport à la caméra par la moitié de la dimension longitudinale y du plus petit détail à détecter (car il est conseillé que le plus petit détail soit au moins vu sur 2 lignes successives acquises par la caméra linéaire). Evidemment, plus cette fréquence est élevée, plus le temps d'intégration est réduit et donc plus l'éclairage de la scène doit être intense.

Enfin, si l'objet est suffisamment éloigné de la caméra (d'une distance D), la connaissance de la largeur du champ de visée et de la taille l du capteur permet de déterminer la distance focale f de l'objectif, appelée optique de la caméra (Fig. 4.37b) :

$$R = \frac{Fov}{x/2} \qquad ; \qquad F_a = \frac{V}{y/2} \qquad ; \qquad f = \frac{l.D}{Fov} \qquad (4.52)$$


FIGURE 4.37 – a) Inspection d'un matériau plan en défilement; b) Géométrie optique.

**Application :** On désire détecter des points de corrosion de taille supérieure ou égale à  $1 \times 1$  mm sur une tôle d'un mètre de large défilant sous une caméra linéaire de contrôle à la vitesse de 10 mètres par minute. La distance entre tôle et caméra vaut D = 1,50 m et la longueur de la barrette de pixels est l = 10 mm.

Déterminer les caractéristiques de la caméra à utiliser :

- 1. Résolution transversale minimale (en pixels).
- 2. Fréquence d'acquisition minimale (en Hz).
- 3. Focale de l'objectif adaptée aux conditions de prise de vue (en mm).
- 4. Taille d'un photosite élémentaire du capteur.

## 4.59 Aberration chromatique d'un capteur

Dans une caméra mono-CCD [67], les couleurs sont recueillies par la technique du filtrage en mosaïque : des filtres colorés sont placés sur la surface sensible du capteur de telle sorte que chaque pixel ne perçoit que l'une des trois composantes primaires Rouge, Vert ou Bleu (Fig. 4.38). Il est donc impossible d'obtenir simultanément les valeurs des 3 composantes colorimétriques en chaque pixel : les réponses de 9 pixels voisins sont nécessaires pour connaître la couleur d'un point de l'image, ce qui entraîne une perte de résolution et des aberrations chromatiques lors de transitions brusques.

Voisinage 3x3															
											$\setminus$				
							_				1		a		
	R	V	В	R	V	В	R	V	B	R	V	В	R	V	
	R	V	В	R	V	В	R	V	В	R	V	В	R	V	
	R	V	В	R	V	В	R	V	В	R	V	В	R	V	
	R	V	В	R	V	В	R	V					.1	V	
			NOIR							BLANC					

FIGURE 4.38 – Structure d'un capteur mono-CDD.

On considère la zone d'image avec la transition verticale Noir/Blanc de la Fig. 4.38. Sachant que la couleur de chaque pixel est évaluée par analyse de son voisinage  $3 \times 3$ , déterminer les couleurs perçues au niveau de la transition :

- 1. Sur la colonne à gauche de la transition Noir/Blanc.
- 2. Sur la colonne à droite de la transition Noir/Blanc.

## 4.60 Seuillage d'image

On considère l'image de micrographie de cellules de la Fig. 4.39(a) [23]. Son histogramme est donné en b), avec deux seuils positionnés à 210 et 235.

- 1. Les images c) et d) sont des versions binarisées obtenues par seuillage de l'histogramme. Indiquer quel seuil donne quelle image.
- 2. Si l'on doit choisir un seuil pour un rendu optimal des cellules, où positionner ce seuil ? Justifier.
- 3. Si l'on veut ne récupérer que la cellule noire, quelle valeur de seuil choisir?



FIGURE 4.39 – Micrographie de cellules : a) image originale; b) histogramme et position de deux seuils; c) et d) résultats de binarisation.

## 4.61 Codage de chaîne

Le codage de chaîne est une représentation efficace d'images binaires contenant des contours.

1. En utilisant l'algorithme de Rosenfeld et Kak avec le code de Freeman (à 8 directions), coder le contour présent dans l'imagette de taille  $11 \times 11$  représentée sur la Fig. 4.40.

- 2. Combien de bits sont nécessaires à ce codage? Justifier le résultat.
- 3. Combien de bits seraient nécessaires si l'on voulait transmettre toutes les coordonnées de tous les points-contours?
- 4. En déduire le taux de compression obtenu grâce au codage chaîné.
- 5. Dans le cas d'une image binaire de dimension  $512 \times 512$ , calculer le gain en taille mémoire qu'on obtient théoriquement grâce au codage chaîné des contours, par rapport à la représentation basique qui consisterait à stocker les coordonnées de chaque point de contour.



FIGURE 4.40 – Contour avec point initial de coordonnées  $(m_0, n_0)$ .

## 4.62 Binarisation par seuillage entropique

Pour binariser les observations issues d'une source discrète (typiquement une image) entachée d'un bruit blanc additif majoritaire sur le support, on peut utiliser la technique du seuillage entropique [97] qui consiste à calculer l'entropie H et à choisir un seuil  $\theta$  défini par :

$$H = -\sum_{i=1}^{M} p_i \log_2 p_i$$
 (4.53)

$$\theta = 2^H \tag{4.54}$$

où :

- -M est le nombre de valeurs distinctes prises par les observations,
- $p_i$  est la probabilité pour qu'une observation  $o_s$  prenne la valeur i en n'importe quel site-pixel s.
- enfin  $\log_2(x)$  dénote le logarithme binaire.

On rappelle que :  $\log_2(x) = \frac{\ln x}{\ln 2}$  et que  $\log_2(2^n) = n$ .

On considère l'imagette de la Fig. 4.41, comportant M = 5 NdG distincts. Elle pourrait représenter par exemple une différence temporelle d'images en valeur absolue, comme en détection de mouvement, ou un module de gradient spatial, comme en détection de contours.

- 1. Donner les probabilités des observations et dessiner l'histogramme.
- 2. Calculer l'entropie H.
- 3. Question facultative :Comparer à l'entropie maximale  $H_{max}$  qu'on peut obtenir avec 5 NdG. En déduire la redondance R.
- 4. Calculer le seuil entropique  $\theta$  correspondant à H.
- 5. Appliquer le seuillage à l'imagette (en conservant les pixels tels que  $o_s > \theta$ ). Commenter le résultat de la binarisation et son intérêt en détection (de mouvement ou de contour).

0	1	0	1	1	0	0	0
0	0	3	2	2	1	0	0
0	1	4	4	4	1	0	0
0	0	4	3	4	1	0	0
0	1	4	4	4	1	0	0
0	1	3	2	3	0	0	0
0	0	1	1	2	1	0	0
0	1	1	0	1	0	0	0

FIGURE 4.41 – Imagette d'observations bruitée.

## 4.63 Spectre de Fourier

Le spectre d'une image est le module au carré de la transformée de Fourier de l'image. Il se représente avec l'origine des fréquences spatiales au centre (qui correspond donc à la composante continue ou valeur moyenne de l'image), les fréquences  $\nu_x$  (ou pulsations  $\omega_x$ ) étant en abscisses, et les fréquences  $\nu_y$  (ou pulsations  $\omega_y$ ) en ordonnées. Plus on s'éloigne du centre, plus on va vers les hautes fréquences. Six spectres sont représentés sur la Fig.4.43.

1. Associer le bon spectre à chacune des images de la Fig. 4.42.

2. Justifier les associations.



FIGURE 4.42 - 6 images.

## 4.64 API JMF-RTP

Etablir un dessin synoptique de la chaîne vidéo complète (acquisition, traitement, transmission, réception, restitution) implantée en TP à l'aide de l'API JMF du langage Java.

Pour chaque opération, indiquer les objets à utiliser ainsi qu'une brève description de leurs fonctionnalités respectives.

Les liens entre les objets devront être précisés afin d'avoir l'enchaînement complet permettant de mettre en œuvre une telle chaîne.

Toute autre information est bienvenue (protocoles, etc.).



FIGURE 4.43 – 6 spectres d'images.

## 4.65 Phénomène d'aliasing

L'image analogique texturée de la Fig. 4.44a est décrite par la fonction continue de l'espace 2D :

$$s(x,y) = 1 + \cos\left[2\pi(u_0x + v_0y)\right] \tag{4.55}$$

avec  $u_0 = 0.2$  et  $v_0 = 0.8$ . Elle est échantillonnée avec un pas d'échantillonnage unitaire en x et en  $y \iff Fe_x = Fe_y = Fe = 1$ ), puis filtrée par un filtre passe-bas idéal de fréquence de coupure  $F_c = \frac{Fe}{2} = \frac{1}{2}$  (cf. résultat Fig. 4.44b pour une taille 50 × 50).

- 1. Quelle est l'orientation des structures visibles dans l'image initiale sur la Fig. 4.44a? Préciser la valeur de la pente observée.
- 2. Etudier, puis dessiner sommairement, les périodicités en x et y de cette texture en calculant les périodes horizontale et verticale (en pixels/cycle). Pour cela, considérer les deux cas de figure typiques :  $x = x_0 = \text{constante d'une part, et } y = y_0 = \text{constante d'autre part.}$
- 3. Dessiner la représentation fréquentielle (spectre 2D) de l'image initiale dans le plan des fréquences réduites  $(u, v) \in [-1; +1] \times [-1; +1]$
- 4. Dessiner ensuite le spectre périodisé de l'image échantillonnée correspondante.
- 5. Déterminer le couple de fréquences de l'image échantillonnée et filtrée passe-bas de la Fig. 4.44b.
- 6. Quelle est l'orientation des structures visibles dans l'image Fig. 4.44b? Préciser la valeur de la pente.
- 7. Comparer qualitativement les textures apparentes des images a) et b).
- 8. Comparer par ailleurs l'image BF analogique de la Fig. 4.44c avec son échantillonnée présentée en Fig. 4.44d (obtenue dans les mêmes conditions que b)). Commentaire de ce cas BF (image contenant les basses fréquences  $u_0 = v_0 = 0.2$ )?
- 9. Enoncer le théorème qui explique les phénomènes observés dans les 2 cas.

## 4.66 QCM sommatif

Dans ce questionnaire à choix multiples (QCM), il peut y avoir une ou plusieurs bonnes réponses par question.



FIGURE 4.44 – Illustration du sous-échantillonnage : a) Image HF analogique ; b) Image HF échantillonnée ; c) Image BF analogique ; d) Image BF échantillonnée.

1.	. Soit une image à 4 niveaux de gris $X = \{I_1; I_2; I_3; I_4\}$ de probabilités respectives	$p_1 = \frac{1}{2} ; p_2 =$
	$\frac{1}{4}$ ; $p_3 = \frac{1}{8}$ ; $p_4 = \frac{1}{8}$ . Son entropie $H(X)$ exprimee en bits/NdG vaut :	
	$\Box -1.75 \qquad \Box 1.75 \qquad \Box 2 = H_{max} \qquad \Box 4$	
2.	2. La cadence vidéo en Europe vaut :	
	$\Box 25 \text{ img/s}$ $\Box 30 \text{ img/s}$ $\Box 33 \text{ img/s}$ $\Box 40 \text{ img/s}$	
3.	B. L'idempotence est vérifiée par les fonctions :	
	$\Box$ érosion $\Box$ dilatation $\Box$ ouverture $\Box$ fermeture	
4.	. Dans une image, le gradient local est un :	
	$\Box$ vecteur parallèle au contour $\Box$ vecteur orthogonal au contour	$\exists$ scalaire entier
	positif $\Box$ scalaire réel	
5.	b. Le code de Freeman est un :	
	$\Box$ algorithme de compression d'image $\Box$ codage de directions	$\Box$ format vidéo
	$\Box$ droit à l'image	
6.	5. La fonction de transfert $H(u)$ du filtre de masque spatial $M = \begin{bmatrix} -1 & 3 & 1 \end{bmatrix}$ est :	
	$\Box \text{ une grandeur complexe} \qquad \Box \frac{1}{3}(1+2\cos 2\pi u) \qquad \Box \frac{1}{2}(1+\cos 2\pi u)$	$\Box \exp\left(-\alpha u\right)$
7.	7. Si l'on transforme une image en son négatif, cela change son :	
	$\Box$ entropie $\Box$ contraste $\Box$ histogramme $\Box$ spectre	
	$\begin{bmatrix} 1 & -3 & 1 \end{bmatrix}$	
8.	B. Le masque $H = \begin{bmatrix} -3 & 9 & -3 \end{bmatrix}$ correspond à un filtre :	
	$\begin{bmatrix} 1 & -3 & 1 \end{bmatrix}$	
	🗌 dérivateur 👘 intégrateur 👘 nasse-has 👘 nasse-haut	

9.	Dans cette liste, trouver l'intrus : □ lissage morphologique □ chapeau haut-de-forme □ Sobel □ tout ou rien
10.	L'estimation de mouvement est un problème : □ bien posé □ sous-déterminé □ sur-déterminé □ d'ouverture
11.	Pour détecter les contours, on peut utiliser un filtre : $\Box$ laplacien $\Box$ binomial-gaussien $\Box$ gradient $\Box$ moyenneur
12.	Le filtre 1D de réponse impulsionnelle $h(k) = a^{ k }$ avec $0 < a < 1$ est un filtre : $\Box$ R.I.F. $\Box$ R.I.I. $\Box$ non-linéaire $\Box$ instable
13.	Lorsqu'on utilise la DCT pour la compression d'image au format JPEG, on traite l'image selon une approche :
14	$\Box$ statistique $\Box$ spatiale $\Box$ frequentielle $\Box$ ensembliste
14.	Pour ameliorer le contraste d'une image, ou peut utiliser :         □ égalisation d'histogramme       □ lissage morphologique       □ étalement d'histogramme         togramme       □ filtrage passe-bas       □ étalement d'histogramme
15.	Le filtre médian est un filtre :
	$\hfill\square$ linéaire $\hfill\square$ utile pour rehausser le contraste dans une image $\hfill\square$ non-linéaire $\hfill\square$ utile pour éliminer un bruit impulsionnel
16.	Le laplacien calculé en un pixel est une grandeur :
	$\Box$ scalaire $\Box$ plus sensible au bruit que le gradient $\Box$ vectorielle $\Box$ moins sensible au bruit que le gradient
17.	Le codage de Shannon-Fano (ou de Huffman) :
	$\Box$ est utilisé pour la compression JPEG
	$\Box$ est une application du theoreme de Shannon (a savoir $F_e > 2\nu_{max}$ )
	$\Box$ le respecte pas la propriete dite « du prenxe » $\Box$ est un codage entropique de l'information
	$\Box$ est un codage de type VLC (variable length coding)
	$\Box$ consiste à coder par des messages courts les événements les plus probables
18.	En Europe, le standard d'acquisition vidéo correspond à une image toutes les : $\Box$ 25 ms $\Box$ 30 ms $\Box$ 33 ms $\Box$ 40 ms
19.	Laquelle de ces approches ignore complètement le caractère spatial d'une image?
	$\hfill approche par masque de filtrage linéaire \hfill approche statistique \hfiltrage fréquentiel \hfill morphologie mathématique$
20.	La morphologie mathématique est un outil du traitement d'image qui ne permet pas de :
	$\Box$ numériser et capturer une image $\Box$ localiser un objet $\Box$ faire l'ana- lyse géométrique d'un objet $\Box$ piloter un robot $\Box$ repérer c'est-à-dire détecter un
	objet

## 4.67 QCM fiche de lecture (1 ou plusieurs bonnes réponses)

- 1. Dans le secteur de la production industrielle [77], les 2 principaux clients du traitement d'image sont l'industrie :
  - (a) métallurgique
  - (b) électrique et des semi-conducteurs
  - (c) automobile et équipementiers
  - (d) pharmaceutique, cosmétique et médicale
  - (e) agroalimentaire
- 2. La croissance du chiffre d'affaires de l'industrie européenne du traitement d'image est due à :
  - (a) la demande en capteurs de vision et caméras

- (b) la demande en systèmes spécifiques dédiés
- (c) la demande en produits standards bon marché et faciles à utiliser
- (d) la demande de l'Amérique
- (e) la demande de la Chine
- 3. Pour la reconnaissance d'objets par vision industrielle, le facteur le plus limitant est actuellement :
  - (a) le temps de traitement en secondes
  - (b) la sensibilité du capteur
  - (c) la flexibilité du système
  - (d) la précision de la mesure
  - (e) la qualité de l'IHM
- 4. Le progrès technologique le plus important pour un opérateur utilisant le traitement d'image est :
  - (a) l'intégration et la compacité du système matériel, p.ex. sans PC
  - (b) le pilotage intuitif de l'IHM et la convivialité du logiciel
  - (c) la technologie multi-core augmentant la capacité de calculs
  - (d) les caméras et les capteurs performants
  - (e) les techniques de communication avec l'outil de production
- 5. Pour la pérennité d'un sous-traitant, l'application de traitement d'image déterminante est :
  - (a) la mesure sans contact
  - (b) le contrôle de la qualité pour l'assurance qualité
  - (c) le stockage automatique de pièces
  - (d) la saisie automatique de pièces
  - (e) la surveillance, par génération automatique de signaux d'arrêt
- 6. Pour le contrôle qualité, le traitement d'image est intéressant car il permet :
  - (a) le zéro défaut
  - (b) le contrôle à 100%
  - (c) un mode opératoire sans contact
  - (d) un contrôle très rapide
  - (e) une solution très flexible
- 7. La tendance technologique la plus forte pour les caméras est :
  - (a) la grande capacité mémoire
  - (b) les caméras rapides
  - (c) l'autonomie (fonctionnement sans PC externe)
  - (d) l'infrarouge
  - (e) la haute résolution
- 8. La caméra la plus rapide a une cadence [112] (exprimée en fps frame per second) de :
  - (a) 2500 images/s
  - (b) 500 fps
  - (c) 30000 fps
  - (d) 16 fps
  - (e) 120 fps
- 9. La plus forte résolution des caméras industrielles est :
  - (a) 1,3 MPixels
  - (b) 10 MPixels
  - (c)  $1084 \times 1488$
  - (d)  $1280 \times 1024$  pixels

- (e) WVGA
- (f) VGA
- (g) 5 MPixels

10. La part des applications non industrielles du traitement d'images représente :

- (a) 10 à 15%
- (b) 25 %
- (c) plus de 45 %
- (d) moins de 5 %
- 11. La croissance dans le secteur de la vision est due surtout :
  - (a) à l'évolution de la standardisation
  - (b) à l'évolution des technologies
  - (c) aux applications industrielles
  - (d) aux applications non industrielles
- 12. Par rapport aux besoins industriels, les applications non industrielles du traitement d'images sont techniquement :
  - (a) plus exigeantes
  - (b) moins exigeantes
  - (c) aussi exigeantes
- 13. Les applications non-industrielles du traitement d'images concernent :
  - (a) le médical, science de la vie et microscopie
  - (b) le solaire thermique
  - (c) l'aéronautique
  - (d) la cartographie et l'agriculture
  - (e) le sport
  - (f) la sécurité, surveillance et gestion du trafic
  - (g) le divertissement et la publicité
  - (h) le photovoltaïque
- 14. Parmi les applications industrielles, le secteur en plus forte croissance est :
  - (a) la recherche
  - (b) la fabrication des semi-conducteurs
  - (c) le photovoltaïque
  - (d) la branche automobile
  - (e) le médical
- 15. Les traitements non réalisables par des caméras embarquées sont :
  - (a) l'étude du séchage
  - (b) l'inspection d'éraflures
  - (c) la localisation de petits défauts de bordure
  - (d) le ravitaillement des avions en vol
  - (e) la lecture temps-réel de codes-barres 1D ou 2D
  - (f) l'élaboration de profilés creux
  - (g) la détection d'objets volumineux
  - (h) la détection 3D
  - (i) la détection des couleurs
  - (j) les mesures géométriques (forme et position)
  - (k) les applications multimédia interactives
  - (l) l'analyse spectrale
- 16. La plus grosse capacité mémoire actuelle est de :

- (a) 32 Mo flash + 128Mo DDRAM
- (b) équvalente à 165 minutes de vidéos
- (c) 8 Mo RAM
- 17. Une caméra embarquée peut comporter :
  - (a) une connexion USB 2.0
  - (b) un PC intégré
  - (c) une liaison série
  - (d) un processeur DSP
  - (e) plusieurs capteurs CCD
  - (f) un capteur CMOS
- 18. Les plus gros goulots d'étranglement en traitement d'images sont :
  - (a) la taille mémoire nécessaire
  - (b) la transmission des images
  - (c) la modularité
  - (d) la linéarité des capteurs
  - (e) la compression
  - (f) la qualité de la couleur
- 19. Le meilleur débit de liaison d'une caméra est de :
  - (a) 5600 MIPS
  - (b) 1,6 gigaHertz
  - (c) 1 Go/s
  - (d) 100 Mo-Ethernet
  - (e) 240 Mo/s
  - (f) 2 Gigabit-Ethernet (dual-GigaE)

## 4.68 Annexe numérique

Les tableaux Tab. 4.2 et Tab. 4.3 donnent les valeurs de quelques racines carrées arrondies, et de logarithmes binaires, permettant de résoudre les exercices sans l'aide d'une calculatrice.

TABLE $4.2$	2 – Table de r	acines carrées	arrondies.
$\sqrt{265} \approx 16$	$\sqrt{221} \approx 15$	$\sqrt{125} \approx 11$	$\sqrt{101} \approx 10$

TABLE $4.3 - 1$ able de logarithmes à base 2.												
x	1	2	3	4	5	6	7	8	9	10	11	12
$\log_2(x)$	0	1	1.585	2	2.322	2.585	2.807	3	3.17	3.322	3.459	3.585

TABLE 4.3 – Table de logarithmes à base 2

## Chapitre 5

## **Travaux** Pratiques

## Préambule

On propose ci-après une douzaine de sujets de TP de traitement d'images, dont la mise en œuvre peut être réalisée avec trois langages de programmation différents : C, Java ou Matlab. Selon les cas, l'implantation des algorithmes utilise ou non la bibliothèque OpenCV (ou son portage JavaCV).

## 5.1 Filtres numériques RII

#### 5.1.1 Filtre de lissage : opérateur de flou

On veut implanter le filtre numérique de réponse impulsionnelle h(k):

$$h(k) = c.e^{-\alpha|k|} \tag{5.1}$$

Le paramètre de lissage  $\alpha$  est un réel positif. La constante de normalisation c est telle que le gain du filtre vaut 1 pour une entrée constante unitaire. On démontre que cette condition implique :  $c = \frac{\alpha}{2}$ . Les équations de récurrence pour l'implantation en cascade sont :

$$y^{+}(k) = x(k) + e^{-\alpha} [y^{+}(k-1) - x(k)]$$
(5.2)

$$y(k) = y^{+}(k) + e^{-\alpha}[y(k+1) - y^{+}(k)]$$
(5.3)

Ce filtre RII, dit de Shen et Castan, est donc représenté par deux équations :

- une équation causale, Eq. (5.2), qui ne fait intervenir que les valeurs présentes et passées (indice k-1) de l'entrée et de la sortie,
- une équation non causale, Eq. (5.3), qui fait intervenir les valeurs futures (indice k + 1) de la sortie.

Le filtrage d'une image consiste simplement à effectuer un premier filtrage 1D pour chaque ligne de l'image, suivi d'un deuxième filtrage 1D pour toutes les colonnes (propriété de séparabilité). On obtient ainsi l'image lissée (*i.e.* floue).

## 5.1.2 Implantation en C du flou réglable

1. **Programme** : écrire une fonction USER\_lissage() qui réalise le lissage d'une image chargée en mémoire (où USER est votre nom d'utilisateur).

On utilisera la structure **image** définie ci-dessous (cf. fichier **bibima.h**) pour stocker les données relatives à une image en NdG (et l'on écrira les routines d'allocation et désallocation de cette structure image).

```
/*** definition du type de donnees (8 bits non signes) ***/
typedef unsigned char Pixtyp;
/******* definition de la structure image ******/
typedef struct image {
    int lin; /*nombre de lignes*/
```

On aura besoin, pour les calculs, du même genre de structure en type double :

On écrira la fonction USER\_lissage() selon le squelette ci-dessous :

```
/****************************/
PImage USER_lissage(PImage img, Datatyp alpha)
{
  PImage out;
  b=exp(-alpha);
  for(l=0;l<nbli;l++) /*pour chaque ligne...*/</pre>
  { /*lissage horizontal*/
    for(c=0;c<nbco;c++) {...}</pre>
    for(c=nbco-1;c>=0;c--) {...}
  }
  for(c=0;c<nbco;c++) /*pour chaque colonne...*/</pre>
  { /*lissage vertical*/
    for(l=0;l<nbli;l++){...}</pre>
    for(l=nbli-1;l>=0;l--){...out->pix[l][c]=...}
  }
  return out;
}
/*********************/
```

où out->pix[1][c] représente le pixel destination de coordonnées (l, c) dans l'image résultat.

N.B. : Bien penser à la gestion des effets de bord et au type des données.

2. **Tests** : lancer l'exécutable et tester le filtrage sur diverses images. Commenter l'influence de  $\alpha$  (typ.  $0 < \alpha \leq 5$ ).

#### 5.1.3 Filtre de dérivation : détecteur de contours

La détection de contour utilise souvent un lissage de l'image dans une direction, suivi d'une dérivation dans l'autre direction. La réponse impulsionnelle du filtre de dérivation est alors la dérivée de la réponse impulsionnelle du filtre de lissage. Ainsi, la dérivée de la réponse impulsionnelle précédente peut s'exprimer comme la somme d'un terme causal et d'un terme anti-causal :

$$g(k) = g^+(k) + g^-(k)$$
 avec (5.4)

 $g^+(k) = -c\alpha \cdot e^{-\alpha k} \quad \text{pour } k > 0$  (5.5)

$$g^{-}(k) = c\alpha e^{\alpha k} \qquad \text{pour } k < 0 \tag{5.6}$$

ce qui conduit tout naturellement à une implantation parallèle. Après normalisation, les équations de récurrence du filtre s'écrivent :

$$y^{+}(k) = -x(k) + e^{-\alpha}[y^{+}(k-1) + x(k)]$$
(5.7)

$$y^{-}(k) = x(k) + e^{-\alpha} [y^{-}(k+1) - x(k)]$$
(5.8)

$$y(k) = y^{+}(k) + y^{-}(k)$$
(5.9)

En appliquant ce filtre dans chacune des directions, on obtient les deux dérivées partielles horizontale et verticale. On peut alors mettre en œuvre une détection de contours. Pour cela, on calculera d'abord le module et l'argument du gradient. On fera ensuite un seuillage pour exhiber les maxima du module.

#### 5.1.4 Implantation en C du détecteur de contour réglable

- 1. Ecrire deux fonctions de dérivation USER\_derivX() et USER\_derivY() pour calculer chacune des deux composantes du gradient.
- 2. Tester sur diverses images : afficher les deux dérivées sous forme d'images. Interpréter les résultats : où obtient-on les fortes valeurs de dérivée ? Commenter l'influence du paramètre  $\alpha$ .
- 3. Ecrire une fonction USER\_detecontour() qui calcule le module et la phase du gradient, et qui réalise le seuillage du module pour fournir en sortie une image binaire.
- 4. Tester le détecteur de contours sur diverses images. Commenter l'influence du seuil  $\theta$  (typ.  $5 \le \theta \le 20$ ). Proposer un jeu de paramètres  $(\alpha, \theta)$  qui convient pour une bonne détection de contours.

On pourra au choix :

- Insérer les routines de traitement dans une application utilisant la bibliothèque OpenCV. Ceci permet de disposer d'une interface graphique pour la gestion et l'affichage des images (fichiers au format propriétaire TRF).
- Construire une application autonome qui intègre les routines de traitement et les entrées/sorties fichier. Il faudra alors écrire un programme principal en C appelant trois fonctions :
  - 1. lecture d'un fichier image (chargement en mémoire) au format JPEP ou PNG,
  - 2. traitement de l'image chargée en mémoire,
  - 3. sauvegarde du résultat dans un fichier image JPEG ou PNG.

On utilisera dans ce cas l'application  $\mathbf{xv}$  (commande  $\mathbf{xv}$ ) pour la visualisation des fichiers images au format PGM.

## 5.2 Estimateur de mouvement

#### 5.2.1 Algorithme de Horn et Schunck

On rappelle (cf. section 2.7.1) que l'algorithme itératif est défini par :

$$u^{k+1} = \overline{u}^k - I_x \frac{I_x \overline{u}^k + I_y \overline{v}^k + I_t}{\alpha^2 + I_x^2 + I_y^2}$$

$$(5.10)$$

$$v^{k+1} = \overline{v}^k - I_y \frac{I_x \overline{u}^k + I_y \overline{v}^k + I_t}{\alpha^2 + I_x^2 + I_y^2}$$

$$(5.11)$$

où :

- $I_x, I_y, I_t$  sont les dérivées partielles par rapport à x, y, t calculées dans le cube  $2 \times 2 \times 2$  (Fig. 2.20a, p. 52),
- $\alpha$  est un coefficient de lissage (typ.  $\alpha = 40$ ),
- k est l'indice d'itération (typ.  $k_{max} = 50$ ),
- $u^k, v^k$  sont les estimées des composantes de vitesse u, v à l'itération k (condition initiale :  $u^0 = 0, v^0 = 0$ )

—  $\overline{u}, \overline{v}$  sont des moyennes pondérées sur un voisinage  $3 \times 3$  (cf. Fig. 2.20b p. 52). Le critère d'arrêt des itérations portera sur la variation relative  $\left|\frac{\Delta u}{u}\right| + \left|\frac{\Delta v}{v}\right|$  comparée à un seuil de précision p (typ. p = 0.01%).

N.B. : En pratique, un léger filtrage passe-bas initial de la séquence est souvent souhaitable pour réduire l'influence du bruit sur les calculs de dérivées (on pourra utiliser pour cela le programme de la section 5.1.1).

- 1. Donner les équations pour le calcul de  $I_x, I_y, I_t$  et  $\overline{u}, \overline{v}$  (cf. exercice 4.23).
- 2. Donner l'organigramme de l'algorithme complet (cf. exercice 4.57), en séparant le prétraitement (filtrage passe-bas optionnel et calcul des gradients), le traitement principal (calcul itératif des vitesses) et l'affichage des résultats (images de gradients et vecteurs-vitesses).

#### 5.2.2 Implantation en C de l'estimateur de vitesse

#### 1. Calcul des gradients :

Ecrire une routine USER\_gradient() qui calcule les gradients spatio-temporels  $I_x, I_y, I_t$  (3 tableaux de valeurs de type double) à partir de deux images successives (2 tableaux de pixels de type unsigned char).

Pour pouvoir visualiser les résultats, on utilisera la routine data2ima() qui transfère et normalise un tableau de valeurs double dans une structure image, en vue de l'affichage des gradients :

```
PData data; /*pointeur memoire 2D type double en entree*/
PImage img; /*structure image normalisee en sortie*/
/*-----*/
/* copie normalisee de structure calcul vers image*/
/*-----*/
void data2ima(PData data,PImage img)
{
  int ligne=img->lin; /*hauteur image*/
  int colonne=img->col; /*largeur image*/
 register int l,c;
 Pixtyp *pout;
 Datatyp *pi;
 Datatyp min=data->pix[0][0];
 Datatyp max=min;
 Datatyp x;
  for(l=0;l<ligne;l++) {</pre>
   for(c=0;c<colonne;c++) {</pre>
     if(data->pix[1][c]>max)
       max=data->pix[l][c];
     else if(data->pix[l][c]<min)</pre>
       min=data->pix[l][c];
   }
 }
 for(l=0;l<ligne;l++) {</pre>
   pout=img->pix[1];/*ligne image normalisee en sortie*/
   pi=data->pix[1];/*ligne donnees de calcul en entree*/
   for(c=0;c<colonne;c++) {</pre>
     x=255.0*((*pi++)-min)/(max-min);
     *pout++=(Pixtyp)x;
   }
 }
}
```

2. Estimation itérative des vitesses :

Ecrire une routine USER\_vitesse() qui calcule les composantes (u, v) des vitesses, avec pour paramètres d'entrée :  $\alpha$ ,  $k_{max}$ , p.

3. Tests : On testera l'algorithme sur quelques séquences synthétiques simples, par exemple un carré en translation diagonale, ou une séquence contenant un disque se déplaçant à la vitesse de 3 pixels/image. Interpréter les résultats et faire tout commentaire pertinent.

Tester d'abord sur la séquence **carre2.trf** (carré en translation diagonale) et interpréter les résultats. Puis tester sur **cercle.trf** contenant un objet se déplaçant à la vitesse de 3 pixels/image. Quel commentaire peut-on faire?

#### N.B. : Utilisation de l'interface

L'utilisateur **USERx** dispose d'un makefile (lire les informations en en-tête) et d'un point d'entrée sous forme d'un fichier **USERx\_interface.c** qui crée une fenêtre avec deux boutons (apply et reset) que l'on affectera aux deux procédures à écrire (USERx\_gradient et USERx\_vitesse). On pourra si nécessaire ajouter d'autres objets graphiques (bouton, entrée numérique ···).

NB : Les données de la séquence sont chargées dans une structure de type *sequence* pointée par **seq**.

Les informations sur l'utilisation des menus de l'interface graphique (load, processing, display) seront données en TP.

## 5.3 Correction d'histogramme

Ce TP concerne la programmation d'une LUT de correction d'histogramme.

Une LUT (*Look-Up Table*) est un tableau de transformation 1D : les adresses de la table sont les niveaux de gris en entrée; les données pointées par les adresses sont les niveaux de gris en sortie. L'intérêt de cette technique est sa rapidité d'exécution (simple adressage sans aucune opération de calcul, puisque tout est déja précalculé dans le tableau). A titre d'exemple, le Tab. 5.1 montre le cas particulier de l'inversion des niveaux de gris, qui permet d'obtenir le négatif d'une image.

TABLE 5.1 Timespe de la Le I.									
adresse $i$	donnée $T(i)$	Exemple : négatif							
0	T(0)	255							
1	T(1)	254							
•									
254	T(254)	1							
255	T(255)	0							

TABLE 5.1 – Principe de la LUT.

- 1. Programmer différentes LUT linéaires par morceaux pour modifier l'histogramme d'une image et tester l'effet sur diverses images.
- 2. Programmer l'étalement d'histogramme (Eq. 1.10). Ceci requiert la détermination préalable de  $i_{min}$  et  $i_{max}$ , minimum et maximum des NdG présents dans l'image.
- 3. Programmer enfin une égalisation d'histogramme (Eq. 1.9). Comparer la complexité de calcul.

## 5.4 Transformation de Fourier rapide

Ce TP vise l'implantation de l'algorithme de FFT (*Fast Fourier Transform*) pour l'analyse fréquentielle d'une image.

On donne ci-dessous l'algorithme en FORTRAN de la FFT 1D (sur 1024 points) de Cooley-Tukey [60].

Implanter en C la FFT 2D pour afficher le spectre d'une image (en module et en phase).

```
SUBROUTINE FFT(F,LN)
  COMPLEX F(1024), U, W, T, CMPLX
  PI=3.141593
  N=2**LN
  NV2=N/2
  NM1=N-1
  I=1
  DO 3 I=1,NM1
    IF(I.GE.J) GO TO 1
    T=F(J)
    F(J)=F(I)
    F(I)=T
    K=NV2
1
    IF(K.GE.J) GO TO 3
2
    J=J-K
    K=K/2
    GO TO 2
3 J=J+K
  DO 5 L=1.LN
    LE=2**L
    LE1=LE/2
    U=(1.0,0.0)
    W=CMPLX(COS(PI/LE1),-SIN(PI/LE1))
    DO 5 J=1,LE1
      DO 4 I=J,N,LE
        IP=I+LE1
        T=F(IP)*U
        F(IP)=F(I)-T
4
      F(I)=F(I)+T
5 U=U*W
  DO 6 I=1,N
6 F(I) = F(I) / FLOAT(N)
  RETURN
  END
```

## 5.5 Filtre de Shen-Castan : implantation Java

## 5.5.1 Présentation

Ce TP montre la possibilité d'implantation d'algorithmes de traitement d'images n'utilisant aucune bibliothèque spécifique, autre que les packages logiciels standards du langage Java de base.

L'objectif du TP est d'implanter en Java les filtres exponentiels classiques de Shen et Castan (déjà présentés en section 5.1 pour l'implantation en C) qui permettent de réaliser un flou ou une détection de contours réglables.

On dispose pour cela d'un fichier d'exemple ChangeTailleOuCouleur. Java utilisant le PixelGrabber pour accéder au tableau de pixels d'une image. Ce programme simple permet de réduire la taille d'une image d'un facteur 2, ou de remplacer la couleur de l'arrière-plan (voir section 5.5.3).

Les classes Java utiles et leurs méthodes sont données dans le Tab. 5.2.

## 5.5.2 Travail demandé

- 1. Implanter le filtre de lissage exponentiel selon les équations Eq. (5.2) et (5.3) p. 119. Il requiert un paramètre réglant la force du filtrage :  $\alpha > 0$  (typ.  $\alpha \in [0...1]$ ).
- 2. Tester le filtre sur différentes images. Commenter l'influence de  $\alpha$ .

```
Classes Java utiles.
                    TABLE 5.2 -
ImageIcon
                                                 PixelGrabber
  .getIconWidth()
                                                   .grabPixels()
  .getIconHeight()
                        JFrame
  .getImage()
                           .setSize()
                                                 Color
                           .setVisible()
                                                   .getRGB()
Image
                           .add()
                                                   .getGreen()
                           .repaint()
                                                   .equals()
MemoryImageSource
                           .createImage()
```

3. Implanter le filtre de dérivation pour l'extraction des contours : Eq. (5.9).

Cela requiert deux paramètres :  $\alpha$  et le seuil  $\theta$  appliqué sur le module du gradient.

- 4. Tester l'extracteur de contours, notamment sur une image-test de damier, pour voir le phénomène visuel perceptible aux angles des cases.
- 5. Concevoir une interface utilisateur (IHM) : en plus du choix du fichier image source, et de l'affichage des images (avant et après traitement), ajouter deux entrées numériques sur l'interface graphique permettant à l'utilisateur d'ajuster les 2 paramètres du traitement : exposant de lissage  $\alpha$  et seuil  $\theta$ .

#### 5.5.3 Programme exemple

```
import java.awt.*;
import java.awt.image.*;
import javax.swing.*;
public class ChangeTailleOuCouleur {
 static public void main(String argv[]) {
  //Chargement de l'image initiale
  String rep="C:/Users/luthon/Pictures/";
  String fic="Planetes.jpg";//"femme.jpg";//"damier.jpg";//
  ImageIcon fond=new ImageIcon(rep+fic);
  int wi=fond.getIconWidth();
  int he=fond.getIconHeight();
  System.out.println(wi+"x"+he);
  //Affichage de cette image dans une fenetre temoin
  JFrame initiale=new JFrame("Image initiale");
  initiale.setSize(wi+20,he+45);
  initiale.setVisible(true);
  initiale.add(new JLabel(fond));
  initiale.repaint();
  //Transformation de cette image en tableau de pixels
  int[] tabPix=new int[wi*he];
  //un pixel est codé par 4 octets: alpha,rouge,vert,bleu
  //alpha dose la transparence: 0=transparent...255=opaque
  PixelGrabber pg = new PixelGrabber(fond.getImage(), 0, 0,
         wi, he, tabPix, 0, wi);
  try { pg.grabPixels(); }
  catch (InterruptedException e) {
   System.err.println(
   "interruption pendant la recuperation des pixels de l'image!");
  }
  //Traitement de ce tableau de pixels
  int[] tabOut=null;
```

```
int large=wi;
 int haut=he;
 int choix=3; //choix du traitement
 switch(choix) {
  case 0://Diminue Taille(copie 1 pixel sur 2 dans petit tableau)
   //NB:Pour ce traitement, l'image doit avoir une taille paire
   //en largeur et hauteur(car on divise les dimensions par 2)
   int[] tableauReduit=new int[wi*he/4];
   int indice=0;
   for (int l = 0; l < he; l=l+2)
    for (int c = 0; c < wi; c=c+2) {
     tableauReduit[indice]=tabPix[l*wi + c];
     indice++;
    }
   large=wi/2;
   haut=he/2;
   tabOut=tableauReduit;
  break;
   case 1://Change Couleur du Fond (ici en couleur caca d'oie)
   Color couleurDuFond=new Color(tabPix[0]);//1er pixel=fond
   Color modifie=new Color(200,150,50); //nouvelle couleur
   for (int l = 0; l < he; l++)
    for (int c = 0; c < wi; c++) {
     Color coul=new Color(tabPix[l*wi+c]);//couleur pixel courant
     if (coul.equals(couleurDuFond)) { //s'il est couleur fond
      tabPix[l*wi + c]=modifie.getRGB(); //on le change
     }
    }
   tabOut=tabPix;
   break;
   default: //Traitement par defaut
   //Recuperation du plan VERT uniquement
   Color coul=null;
   for (int l = 0; l < he; l++)
    for (int c = 0; c < wi; c++) {</pre>
     coul=new Color(tabPix[l*wi+c]);//couleur pixel courant
     tabPix[l*wi+c]=coul.getGreen();//recuperation du vert
    }
//======= INSERTION DES TRAITEMENTS =======//
     TraitementFL.algoNegatif(tabPix, tabPix, wi); //Negatif
//Conversion du tableau resultat en image NdG
    int pixel;
   for (int l = 0; l < he; l++)
    for (int c = 0; c < wi; c++) {
     pixel=tabPix[l*wi+c];
     coul=new Color(pixel,pixel,pixel); //pour Image NdG
     tabPix[l*wi+c]=coul.getRGB();
    }
   tabOut=tabPix;
  break;
 }
 //Conversion du tableau de pixels en objet de classe Image
 Image resultat=initiale.createImage(new MemoryImageSource(large,
```

```
haut, tabOut, 0, large));
//Conversion du tableau de pixels en objet de classe ImageIcon
ImageIcon resultatIcone=new ImageIcon(resultat);
//Affichage du resultat dans une autre fenetre
JFrame finale=new JFrame("Image traitée");
finale.setSize(resultatIcone.getIconWidth()+20,
        resultatIcone.getIconHeight()+45);
finale.setVisible(true);
finale.add(new JLabel(resultatIcone));
finale.repaint();
}
```



FIGURE 5.1 - Résultats visuels du programme ChangeTailleOuCouleur.

## 5.6 OpenCV : implantation Java dans Eclipse

## 5.6.1 Présentation

L'objectif de ce TP est d'apprendre à utiliser la bibliothèque JavaCV, riche et pratique, qui simplifie la programmation d'algorithmes de traitement d'images. Elle résulte du portage en langage Java de la fameuse bibliothèque OpenCV, développée à l'origine en langage C. On dispose pour ce TP de :

- 3 bibliothèques (préalablement téléchargées) placées à la racine  $\texttt{C:\del}$  de l'ordinateur :
  - **opencv** : 275 Mo (exécutable disponible sur **opencv**.org)

```
javacv-bin : 13 Mo
```

- javacv-cppjars : 89 Mo (zip disponibles sur code.google.com)
- une documentation sur OpenCV [24, 74] dans le parcours eLearn "Image Processing & Computer Vision" (rubrique "Supports de cours").
- plusieurs vidéo-tutoriels en ligne : engineervisions.blogspot.com/
- un site expliquant l'installation de JavaCV et la configuration de l'environnement Eclipse : opencvlover.blogspot.com/
- un fichier d'exemple JavaCV1.java.

## 5.6.2 Installation et test de JavaCV

- 1. Créer un nouveau projet (nommé avec votre nom d'utilisateur  $\tt USER)$  :
  - File> New Java Project> Project Name 'USER\_Proj'> Next>
     Add project 'USER\_Proj' to build path> Finish;
- 2. Charger les 3 archives JAR dans l'environnement de programmation :
  - Project> Properties> Java Build Path> Add External JARS>
    - ... > Apply> Order&Export> Select All

Si tout va bien, on obtient 9 fichiers pour le système Windows (Fig. 5.2).

3. Importer un premier programme pour tester le bon fonctionnement :

File Import> General> File System> Browse 'CheminRepertoire'

choisir le fichier-source JavaCV1. java et le ranger dans le sous-répertoire du projet 'USER\_Proj/src'; choisir les fichiers de données (image ou vidéo) et les mettre dans le répertoire racine du projet 'USER\_Proj/'.

4. Lancer le programme test JavaCV1 pour vérifier le bon fonctionnement.



FIGURE 5.2 – Copie d'écran des bibliothèques dans l'environnement Eclipse.

#### 5.6.3 Programmation d'un traitement d'image au choix

En s'aidant de JavaCV1. java, de la documentation sur OpenCV, et des tutoriels en ligne, implanter un traitement parmi les suivants sur un fichier d'image fixe JPEG ou PNG :

- Détection de contours dans une ROI (algorithme de Canny)
- Pyramide et détection de contours (algorithme de Sobel)
- Morphologie mathématique : lissage, Top Hat, Black Hat
- Transformée de Fourier discrète en cosinus (DCT).

#### 5.6.4 Listing du programme exemple

Le programme JavaCV1.java charge un fichier image (cvLoadImage), fait des conversions couleur (cvCvtColor), affiche les résultats (cvShowImage), attend une entrée de touche au clavier (cvWaitKey), pour finalement enregistrer les résultats dans des fichiers (cvSaveImage).

N.B. : Les importations de bibliothèques figurant en début de ce programme sont à mettre **systématiquement** dans tout en-tête de fichier qui utilise les fonctions de JavaCV.

```
import com.googlecode.javacv.*;
import static com.googlecode.javacv.cpp.opencv_calib3d.*;
import static com.googlecode.javacv.cpp.opencv_objdetect.*;
import static com.googlecode.javacv.cpp.opencv_legacy.*;
import static com.googlecode.javacv.cpp.opencv_core.*;
import static com.googlecode.javacv.cpp.opencv_imgproc.*;
import static com.googlecode.javacv.cpp.opencv_highgui.*;
public class JavaCV1 {
 public static void main(String[] args) {
  IplImage img=cvLoadImage("Penguins.jpg");
  IplImage hsvimg =cvCreateImage(cvGetSize(img),IPL_DEPTH_8U,3);
  IplImage graying =cvCreateImage(cvGetSize(img), IPL_DEPTH_8U, 1);
  //==== INSERTION DES TRAITEMENTS ====//
  cvCvtColor(img,hsvimg,CV_BGR2HSV);
  cvCvtColor(img,grayimg,CV_BGR2GRAY);
  //===========//
  cvShowImage("Original",img);
  cvShowImage("HSV",hsvimg);
  cvShowImage("GRAY",grayimg);
  cvWaitKey();
  cvSaveImage("HSV.jpg",hsvimg);
  cvSaveImage("GRAY.jpg",grayimg);
  cvReleaseImage(img);
  cvReleaseImage(hsvimg);
  cvReleaseImage(grayimg);
 }
}
```

## 5.6.5 Structure d'image

La structure d'image **IplImage** utilisée dans OpenCV ou JavaCV hérite d'une structure de matrice, elle-même héritant d'une structure de tableau (Fig. 5.3) [24].



FIGURE 5.3 – Structures-mères de IplImage.

La définition de cette structure est donnée ci-après :

```
typedef struct _IplImage {
    int nSize;
    int ID;
    int nChannels;
```

F.LUTHON,2025

```
int alphaChannel;
  int depth;
  char colorModel[4];
  char channelSeq[4];
  int dataOrder;
  int origin;
  int align;
  int width;
  int height;
  struct _IplROI* roi;
  struct _IplImage* maskROI;
  void* imageId;
  struct _IplTileInfo* tileInfo;
  int imageSize;
  char* imageData;
  int widthStep;
  int BorderModel[4];
  int BorderConst[4];
  char* imageDataOrigin;
} IplImage;
```

```
Parmi toutes les variables de cette structure complexe, les principales sont bien sûr width, height
```

et nChannels qui donnent respectivement la largeur, la hauteur et le nombre de plans couleur de l'image (par exemple nChannels = 3 pour une image RVB).

Enfin, le pointeur ImageData donne accès aux données élémentaires. La syntaxe d'adressage d'une valeur dans l'image img à la ligne l, à la colonne c et dans le plan couleur i est la suivante :

```
img->imageData[(l*width+c)*nChannels +i];
```

N.B. Pour une image en NdG (où nChannels = 1 et donc i = 0), l'adressage d'un pixel se réduit simplement à :

img->imageData[l\*width+c]

## 5.7 Détecteur de mouvement

#### 5.7.1 Présentation

On dispose de :

- une séquence vidéo présentant une scène de rue comportant un piéton marchant sur un trottoir,
- une image fixe de plage de même dimension (pour incrustation),
- un fichier exemple TP3. java permettant de visionner une séquence d'images.

## 5.7.2 Programmation en Java

- 1. Calculer et visualiser les observations de mouvement  $o_s$  (simples différences d'images).
- 2. Implanter un seuillage pour obtenir la séquence binaire des changements temporels.
- 3. Ajouter un post-traitement de morphologie mathématique (lissage morphologique) pour améliorer la détection et visualiser les masques mobiles.
- 4. Effet de trucage : remplacer finalement le fond fixe de la scène de rue par un autre fond obtenu avec l'image de plage, pour donner l'impression que le piéton marche sur la plage.
- 5. Concevoir une interface utilisateur qui offre les fonctions suivantes :
  - application d'un flou sur une séquence d'images JPEG (cf.  $\S5.5$ ),
  - détection de contours et visualisation dynamique (cf.  $\S5.5$ ),
  - choix de la séquence vidéo grâce à un explorateur de fichiers,

- détection de mouvement avec réglage des paramètres,
- changement du fond de la séquence (incrustation du mobile dans une nouvelle scène) avec possibilité de choisir l'image de fond sur lequel on souhaite incruster l'objet en mouvement.

#### 5.7.3 Listing du programme exemple

Le programme TP3. java calcule le négatif de chaque image de la vidéo choisie, et stocke le résultat dans le fichier Out.AVI. Il permet de charger les images d'une séquence vidéo ou celles issues d'une webcam, de visionner la séquence initiale et la séquence traitée, et d'enregistrer la vidéo résultat.

```
public class TP3 {
```

```
public static void main(String[] args) throws Exception {
 int wi=320; //largeur par defaut
 int he=240; //hauteur par defaut
 CvCapture capture; //source video
 int choixSource=0; //Camera=0;sinonFichier//
 switch(choixSource) {
  case 0:
   capture=cvCreateCameraCapture(CV_CAP_ANY);//ou (-1);//
   cvSetCaptureProperty(capture,CV_CAP_PROP_FRAME_WIDTH,wi);
   cvSetCaptureProperty(capture,CV_CAP_PROP_FRAME_HEIGHT,he);
  break;
  default:
   String rep="C:/Users/luthon/Videos/";
   String file="Pieton.AVI"; //"video_intro.mpg";//
   file=rep+file;
   capture=cvCreateFileCapture(file);
 }
 //INITIALISATION 1ere image//
 IplImage frame=cvQueryFrame(capture);
 if(frame == null) {
  System.out.println("ERR=No video file"); }
 CvSize dim=cvGetSize(frame);
 wi=dim.width(); he=dim.height();
 System.out.println(wi+"x"+he);
 FrameRecorder recorder=new OpenCVFrameRecorder("Out.AVI",wi,he);
 recorder.setVideoCodec(CV_FOURCC('M','J','P','G'));
 recorder.setFrameRate(15);
 recorder.setPixelFormat(0);//ATT!:0=NdG;1=couleur//
 recorder.start();
 cvNamedWindow("VideoIn", CV_WINDOW_AUTOSIZE);
 cvNamedWindow("VideoOut", CV_WINDOW_AUTOSIZE);
 IplImage grayimg=cvCreateImage(dim,IPL_DEPTH_8U,1);
 CvMat matrice= cvCreateMat(he,wi,CV_8UC1);
 //BOUCLE TEMPORELLE//
 for (;;) {
  frame=cvQueryFrame(capture);
  if(frame == null) {
   System.out.println("ERR=No video file");
  break;
  }
  cvShowImage("VideoIn",frame);
  //==== INSERTION DES TRAITEMENTS ====//
  for (int 1=0; 1<he; 1++)
   for (int c=0; c<wi; c++) {</pre>
```

```
CvScalar pixel = cvGet2D(frame.asCvMat(), 1, c);
     double pix=255-pixel.green(); //Negatif
     matrice.put((l*wi + c), pix);
    }
   grayimg = matrice.asIplImage();
   //=======
                                      =====//
   cvShowImage("VideoOut",grayimg);
   recorder.record(grayimg); //Enregistrement
   char c=(char) cvWaitKey(1000/25);
   if(c=='q') break;
  }
  cvWaitKey();
  recorder.stop();
  cvReleaseCapture(capture);
  cvDestroyWindow("VideoIn");
  cvDestroyWindow("VideoOut");
  matrice.release();
 }
}
```

Les sorties visuelles de TP3. java sont illustrées sur la Fig. 5.4.



FIGURE 5.4 – Une image de la séquence Pieton et son négatif.

## 5.8 Mini-projet : interface de traitement vidéo

## 5.8.1 Objectifs

Réaliser une application en Java qui offre les fonctionnalités suivantes :

— acquisition et restitution vidéo (webcam),

— traitement d'images en temps-réel (*live*) ou différé (*batch*).

## 5.8.2 Cahier des charges

Une interface graphique conviviale permettra à l'utilisateur de :

- choisir la source de données à traiter : flux vidéo capté par la webcam, séquence d'images AVI, vidéo MPEG, ou images fixes JPEG stockées sur disque,
- régler les différents paramètres de traitement (via des entrées numériques à glissière ou *sliders*) : seuils, caractéristiques de filtre,
- visualiser simultanément la source vidéo et le résultat du traitement,
- sauvegarder les données sur disque (séquence acquise, image résultat...)

## 5.8.3 Consignes de travail

- Utiliser la bibliothèque JavaCV (Java Computer Vision).
- Programmer en terme d'objets : définir les classes, interfaces et méthodes *ad hoc*; fournir le diagramme de classes.
- Tester et caractériser l'application : performances, limitations, bugs, rapidité, efficacité, qualités et défauts, perspectives d'évolution.
- Documenter avec toutes les explications utiles, organigrammes, illustrations de résultats, objets graphiques et fonctionnalités de l'interface, listings des fichiers sources Java.
- On travaillera en binôme.

On veillera à programmer proprement en respectant les 3 principes de base du *Clean Code* [18] : un **nommage des variables** parlant, avec des noms prononçables et cherchables (par CTRL F), des **fonctions simples** implantant un seul traitement, avec peu de paramètres d'entrée et nommées par un verbe, des **commentaires utiles**, dont les aspects de licence du code (le cas échéant) toujours à mettre en début des fichiers.

#### 5.8.4 Liste de traitements proposée

On implantera l'un des traitements suivants en s'inspirant du fichier exemple TP3. java qui manipule au choix une séquence vidéo enregistrée ou le flux issu d'une webcam, et enregistre la vidéo résultat dans un fichier AVI (section 5.7.3).

- 1. Flou de force réglable sur les NdG ou la couleur (filtre de Shen-Castan).
- 2. Détection de contours sur les NdG et incrustation sur la couleur.
- 3. Détection de teinte chair [123] (algorithme décrit page 135).
- 4. Rehaussement de détails [132] (algorithme en section 5.8.5).
- 5. **Rehaussement automatique de contraste** par égalisation ou étalement de l'histogramme des NdG.
- 6. Transformée couleur LUX (Eq. (1.80) p. 32).
- 7. **Trucage d'image** par manipulation d'histogramme : affichage d'histogramme et choix par l'utilisateur d'une transformation non-linéaire (linéaire par morceaux).
- 8. Histogramme et seuillage pour binarisation d'image.
- 9. Filtre morphologique adaptatif (algorithme fourni en section 4.48).
- 10. Seuillage entropique pour la détection de mouvement (section 4.62).
- 11. Etiquetage en composantes connexes (algorithme en section 4.49).
- 12. Détection de mouvement avec régularisation par MRF (section 2.2.4).
- 13. Estimation de mouvement (algorithme Horn-Schunck, section 4.23).
- 14. Squelettisation d'image binaire (morphologie mathématique § 4.15).
- 15. Détection de présence basée sur la détection de mouvement et déclenchement d'une alarme visuelle ou sonore sur le poste distant.
- 16. Apprentissage et classification d'objets par reconnaissance de forme et/ou de teinte.

#### 5.8.5 Algorithme de rehaussement de détails

ALGORITHME DIT DU MASQUE FLOU Données:

```
-image I de taille C*L
-entier N impair (typ. N=5)
-matrice de convolution H de taille N*N
-pondération A dans [0;1]
```

Voisinage de chaque pixel (x,y) limité aux bords:

```
d = (N-1)/2
xmin(x) = max(0, x-d)
xmax(x)=min(C-1,x+d)
ymin(y) = max(0, y-d)
ymax(y)=min(L-1,y+d)
Coefficient du voisin (i,j) dans matrice centree sur pixel (x,y):
vx(i)=i-x+d
vy(j)=j-y+d
Calcul du masque flou: pour chaque pixel (x,y)
pour i allant de xmin(x) à xmax(x) et j de ymin(y) à ymax(y)
  MFlou[x,y]=Somme(I[i,j]*H[vx(i),vy(j)])/Somme(H[vx(i),vy(j)])
Calcul de l'image rehaussee: pour chaque pixel (x,y)
MDetail[x,y]=I[x,y]-MFlou[x,y]
ImFinale[x,y]=A*I[x,y]+(1-A)*MDetail[x,y]
ou
ImFinale[x,y]=A*I[x,y]+(1-A)*log(MDetail[x,y])
 011
ImFinale[x,y]=A*I[x,y]+(1-A)*exp(MDetail[x,y])
```

## 5.9 Initiation au traitement d'image avec Matlab

## 5.9.1 Introduction

Matlab (pour *Matrix Laboratory*) est un environnement logiciel de **calcul et visualisation scientifique**. Il repose sur un langage **interprété**. La donnée de base est la **matrice** ou tableau (une image, un vecteur ou un scalaire étant des cas particuliers). L'**interactivité** de Matlab permet de programmer et simuler très rapidement. Ses inconvénients éventuels sont la lenteur d'exécution et le coût en mémoire.

Matlab est fourni avec de nombreuses boîtes à outils (*toolboxes*) dédiées à des applications spécifiques ; ce sont des bibliothèques de fonctions (fichiers ASCII d'extension .M), dont notamment :

— Signal Processing : traitement de signal (filtrage, analyse spectrale),

— Image Processing : traitement d'image [61].

Pour connaître la syntaxe d'une fonction, on dispose d'une aide en ligne très utile, simplement en tapant dans la fenêtre de commandes Matlab :

help <nom\_de\_la\_fonction>

L'équivalent de Matlab en logiciel libre est scilab, téléchargeable sur le site : https://www.scilab.org/

#### 5.9.2 Programmes démonstratifs

La boîte à outils *Image Processing* contient de nombreuses démonstrations dans le sous-répertoire : <Matlab>\toolbox\images\imdemos\

Lancer quelques démonstrations, par exemple :

edgedemo, firdemo, dctdemo, nrfiltdemo.

#### 5.9.3 Lecture-écriture, affichage, manipulation d'image

Ecrire un programme qui lit et affiche une image, et donne la taille mémoire.

help: imread(), imshow(), image(), whos

Ajouter diverses manipulations telles que : découpage, rotation, ré-échantillonnage.

help: imcrop(), imrotate(), imresize()

Sauvegarder le résultat d'un des traitements dans un fichier au format JPEG et afficher ses informations.

help: imwrite(), imfinfo.

#### 5.9.4 Modification d'histogramme et seuillage

Afficher et égaliser l'histogramme d'une image à faible contraste, par exemple l'image **auto\_00.bmp** ou l'image **pout.tif** fournie dans Matlab.

```
help: imhist(), histeq(), imadjust()
```

Ecrire une fonction d'étalement d'histogramme et la tester dans le programme principal. Comparer à la fonction prédéfinie dans Matlab (... trouver laquelle!)

Compléter le programme en ajoutant un seuillage des NdG de l'histogramme pour binariser l'image. Le seuil sera soit choisi par l'utilisateur, soit déterminé automatiquement.

```
help: im2bw(), graythresh().
```

#### 5.9.5 Filtrage linéaire : contours, gradient, laplacien, lissage

Tester un détecteur de contour proposé dans Matlab.

Implanter par ailleurs un autre détecteur de contour par calcul du gradient avec le masque le plus simple  $\begin{bmatrix} -1 & 1 \end{bmatrix}$  (cf. section 4.5). Implanter aussi un laplacien et un lisseur de taille 5 × 5. Visualiser les images résultats.

```
help: edge(),filter2(), imfilter(), im2double(), uint8(),
            sqrt(), ones(), colorbar
```

#### 5.9.6 Transformée de Fourier et fonction de transfert

Calculer et afficher le spectre d'une image. Visualiser la fonction de transfert d'un filtre gaussien.

help: fft2(), fftshift(), log(), abs(), fspecial(), freqz2().

## 5.10 Traitement d'image statique avec Matlab

#### 5.10.1 Morphologie mathématique

Tester quelques-uns des opérateurs de morphologie mathématique disponibles dans Matlab, à la fois sur des images en NdG :

```
help: strel(), imerode(),imdilate(),imclose(),imopen(),imtophat()
```

et sur des images en noir et blanc (obtenues après un seuillage par exemple) :

```
help: bwhitmiss(), bwmorph().
```

#### 5.10.2 Détection de teinte chair

On fournit le principe d'un algorithme de détection de teinte chair pour le suivi de visage [123]. Il travaille sur les composantes normalisées  $\in [0; 1]$ :

$$r = \frac{R}{R+G+B} \quad \text{et} \quad g = \frac{G}{R+G+B}$$
(5.12)

et recherche les pixels dont le couple (r, g) est compris à l'intérieur d'un croissant de lune du plan (r, g)limité par deux paraboles dont les coefficients sont fournis ci-après (les auteurs en proposent deux jeux différents). Un simple test d'appartenance au croissant de lune donne les pixels candidats pour la teinte chair (skin S): Eq. (5.13).

$$S = \begin{cases} 1 & \text{si } (Q_{-} < g < Q_{+}) \& (W > 0.004) \& r \in [0.2; 0.6] \\ 0 & \text{sinon} \end{cases}$$
(5.13)

où : 
$$\begin{cases} Q_{+} = A_{u}r^{2} + b_{u}r + c_{u} \\ Q_{-} = A_{d}r^{2} + b_{d}r + c_{d} \\ W = (r - 0.33)^{2} + (g - 0.33)^{2} \end{cases}$$
(5.14)

$$\begin{cases} A_u = -1.3767 & A_d = -0.776 \\ b_u = 1.0743 & b_d = 0.5601 \\ c_u = 0.1452 & c_d = 0.1766 \end{cases} \quad \text{ou} \begin{cases} A_u = -1.8423 & A_d = -0.7279 \\ b_u = 1.5294 & b_d = 0.6066 \\ c_u = 0.0422 & c_d = 0.1766 \end{cases}$$

Programmer cet algorithme et le tester sur diverses images de visage.

## 5.11 Simulation des exercices avec Matlab

#### 5.11.1 Filtrage linéaire

Tracer toutes les fonctions de transfert des filtres vus au chapitre 4. Les appliquer sur quelques images et visualiser les résultats.

help: freqz2(), imfilter().

#### 5.11.2 Visualisation scientifique : tracé de courbes

Tracer les histogrammes rencontrés au chapitre 4. Tracer également l'allure du signal vidéo TV Secam, ainsi que les courbes des estimateurs robustes rencontrés au chapitre 4.

help: bar(), stairs().

# Bibliographie

- T. Aach, A. Kaup, and R. Mester. Statistical model-based change detection in moving video. Signal Processing, 31(2):165–180, March 1993.
- [2] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. International Journal of Computer Vision, 2:283–310, 1989.
- [3] N. Baaziz. <u>Approches d'estimation et de compensation de mouvement multirésolutions pour le</u> codage de séquences d'images. PhD thesis, Université de Rennes 1, October 1991.
- [4] Cristina Barroca. <u>Traitement d'images : le livre des secrets</u>. Dunod, mars 2005.
- [5] J.L. Barron, D.J. Fleet, and S.S. Beauchemin. Performance of optical flow technics. <u>International</u> Journal of Computer Vision, 12(1):43–77, 1994.
- [6] B. Beaumesnil and F. Luthon. Real Time Tracking for 3D Realistic Lip Animation. In <u>18th Int.</u> Conf. Pattern Recognition, ICPR'06, pages 219–222, Hong Kong, 20-24 Aug 2006. Vol.1.
- [7] B. Beaumesnil, F. Luthon, and M. Chaumont. Extraction temps-réel de contours labiaux par segmentation vidéo robuste en vue d'animation 3D. In <u>20e colloque GRETSI</u>, pages 489–492, UCL Louvain-la-Neuve, Belgique, 6-9 Sept. 2005.
- [8] B. Beaumesnil, F. Luthon, and M. Chaumont. Liptracking and MPEG-4 Animation with Feedback Control. In <u>IEEE Int. Conf. Acoustics Speech Signal Processing</u>, ICASSP'06, pages 677–680, Toulouse, France, 14-19 May 2006. Vol.II.
- [9] Brice Beaumesnil. Suivi labial couleur pour analyse-synthèse vidéo et communication temps réel. PhD thesis, Université de Pau et des Pays de l'Adour, Bayonne, France, December 2006.
- [10] A. Beghdadi and A. Le Négrate. Contrast enhancement technique based on local detection of edges. Computer Vision, Graphics, and Image Processing, 46(2) :162–174, 1989.
- [11] A. Belaïd and Y. Belaïd. <u>Reconnaissance des formes</u> : Méthodes et applications. InterEditions, Paris, 1992.
- [12] B. Benmiloud and W. Pieczynski. Estimation des paramètres dans les chaînes de Markov cachées et segmentation d'images. Traitement du Signal, 12(5):433–454, 1995.
- [13] Laurent Berger. <u>Traitement d'images et de vidéos avec OpenCV3 (en C++)</u>. D-BookeR, déc. 2017.
- [14] Laurent Berger. <u>Traitement d'images et de vidéos avec OpenCV4 (en Python)</u>. D-BookeR, jan. 2020.
- [15] Maïtine Bergounioux. <u>Introduction au traitement mathématique des images Méthodes</u> déterministes, volume 76 of Mathématiques et Applications. Springer, Berlin, 2015.
- [16] J. Besag. On the statistical analysis of dirty pictures. J. R. Statist. Soc. B, 48(3):259–302, 1986.
- [17] J. Biemond, L. Looijenga, D.E. Boekee, and R.H.J.M. Plompen. A pel-recursive Wiener-based displacement estimation algorithm. <u>Signal Processing</u>, 13:399–412, 1987.
- [18] Xavier Blanc. Coder proprement. Programmez !, 214 :29–32, Janvier 2018.
- [19] I. Bloch, Y. Gousseau, H. Maître, D. Matignon, B. Pesquet-Popescu, F. Schmitt, M. Sigelle, and F. Tupin. <u>Le traitement des images Tome 1</u>. Département TSI, Télécom, Paris, version 5.0 edition, septembre 2004. Polycopié de cours, 226 p.
- [20] René Bouillot. Cours de traitement numérique de l'image. Dunod, mai 2005.

- [21] Emmanuel Bouix. Modèles de flux et de composants pour applications multimédias distribuées dynamiquement reconfigurables. PhD thesis, Université de Pau et des Pays de l'Adour, Bayonne, France, November 2007. co-encadrant 50% P. Roose, MC27.
- [22] P. Bouthémy and P. Lalande. Recovery of moving object masks in an image sequence using local spatiotemporal contextual information. <u>Optical Engineering</u>, 32(6) :1205–1212, June 1993.
- [23] A. Bovik, editor. <u>Handbook of Image & Video Processing</u>. Elsevier, USA, 2nd edition, 2005.
- [24] Gary Bradski and Adrian Kaehler. Learning OpenCV. O'Reilly, USA, September 2008.
- [25] Stéphane Bres, J-Michel Jolion, and Frank Lebourgeois. <u>Traitement et analyse des images</u> <u>numériques</u>. Lavoisier-Hermès, juillet 2003.
- [26] W. Brown and B. Shepherd. <u>Graphics File Formats reference & guide</u>. Manning Publications Co., Greenwich, USA, 1995.
- [27] Gilles Burel. <u>Introduction au traitement d'images</u>; simulation sous Matlab. Hermès. Lavoisier, octobre 2001.
- [28] J. Canny. A Computational Approach to Edge Detection. <u>IEEE Trans. on Pattern Analysis and</u> Machine Intelligence, 8(6) :679–698, November 1986.
- [29] A. Caplier, C. Dumontier, F. Luthon, and P. Y. Coulon. Algorithme de détection de mouvement par modélisation markovienne. Mise en oeuvre sur DSP. <u>Traitement du Signal</u>, 13(2):177–190, 1996. https://www.iieta.org/pdf-viewer/7585.
- [30] A. Caplier and F. Luthon. Approche spatio-temporelle pour l'analyse de séquences d'images. Application en détection de mouvement. <u>Traitement du Signal</u>, 14(2) :195-208, 1997. https://www.iieta.org/pdf-viewer/7141.
- [31] A. Caplier, F. Luthon, and C. Dumontier. Real-time implementations of an MRFbased motion detection algorithm. <u>Real-Time Imaging</u>, 4(1) :41–54, February 1998. doi:10.1006/rtim.1996.0062.
- [32] Alice Caplier. Modèles markoviens de détection de mouvement dans les séquences d'images : Approche spatio-temporelle et mises en œuvre temps réel. PhD thesis, Institut National Polytechnique, Grenoble, France, December 1995.
- [33] A. Chéhikian, P. Y. Coulon, and F. Luthon, editors. <u>Ecole des Techniques Avancées en Signal-Image-Parole (ETASIP'97)</u>, Grenoble, 9-12 Septembre 1997. LTIRF-INPG. Session 2, 486 pages.
- [34] R. Chellappa and A. Jain, editors. <u>Markov Random Fields</u> : Theory and Application. Academic Press, Inc., San Diego, 1993.
- [35] G. Chen, H. Cao, J. Conradt, H. Tang, F. Röhrbein, and A. Knoll. Event-based neuromorphic vision for autonomous driving. A paradigm shift for bio-inspired visual sensing and perception. IEEE Signal Processing Magazine, 37(4):34–49, July 2020.
- [36] Leonardo Chiariglione. Delivering standards to industries : The MPEG case. <u>IEEE Signal</u> Processing Magazine, 37(2) :81–88, March 2020.
- [37] S.Y. Chien, Y.W. Huang, C.Y. Chen, H.H. Chen, and L.G. Chen. Hardware architecture design of video compression for multimedia communication systems. <u>IEEE Communications Magazine</u>, 43(8) :123–131, August 2005.
- [38] K. Choi, J. Chen, D. Rusanovskyy, K. P. Choi, and E. S. Jang. An overview of the MPEG-5 essential video coding standard. IEEE Signal Processing Magazine, 37(3):160–167, May 2020.
- [39] J. P. Coquerez and S. Philipp. <u>Analyse d'images : filtrage et segmentation</u>. Masson, Paris, 1995.
- [40] P.Y. Coulon. <u>Traitement des Images</u>. Reprographie ENSERG, INPG, Grenoble, 1998.
- [41] I. Daubechies. <u>Ten lectures on wavelets</u>, volume 61 of <u>CBMS-NSF regional conference series in</u> Applied Mathematics. SIAM, Philadelphia, Pennsylvania, 1992.
- [42] R. Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. International Journal of Computer Vision, 1(2) :167–187, May 1987.

- [43] Fadi Dornaika, editor. <u>Advances in Face Image Analysis : Theory and Applications</u>. Bentham Science Publishers, 2016. ISBN 978-1-68108-111-3.
- [44] R.O. Duda and P.E. Hart. <u>Pattern Classification and Scene Analysis</u>. John Wiley & Sons, New York, 1973.
- [45] C. Dumontier, F. Luthon, and J. P. Charras. Real-time DSP implementation for MRF-based video motion detection. <u>IEEE Trans. on Image Processing</u>, 8(10) :1341–1347, October 1999. doi:10.1109/83.791960.
- [46] Christophe Dumontier. <u>Etude et mise en œuvre temps réel d'un algorithme de détection de</u> <u>mouvement par approche markovienne</u>. PhD thesis, Institut National Polytechnique, Grenoble, France, November 1996.
- [47] N. Eldrogi, B. Larroque, V. Bolliet, and F. Luthon. Vision par ordinateur pour suivi automatique de civelles en bassin. In <u>17e Congrès ORASIS</u>, Saint-Dié-des-Vosges, 27-31 Mai 2019. AFRIF.
- [48] N. Eldrogi, F. Luthon, B. Larroque, S. Alqaddafi, and V. Bolliet. Motion estimation of glass eels by differential methods. <u>Int. Science and Technology Journal (ISTJ)</u>, pages 299–315, July 2018. (paper presented at 2nd Conf. on Medical & Technological Sciences (CMTS), Qaminis, Libya, Jul. 2018).
- [49] F. Faux and F. Luthon. Modélisation de la teinte du visage par fusion d'informations couleur dans le cadre de la théorie de Dempster-Shafer. In <u>CORESA'05</u>, pages 85–90, Rennes, 7-8 Nov 2005.
- [50] F. Faux and F. Luthon. Modélisation de visage par fusion d'information couleur dans la cadre de la théorie de l'évidence et suivi par filtrage particulaire. In <u>15e Congrès RFIA'06</u>, Tours, 25-27 Janv. 2006.
- [51] F. Faux and F. Luthon. Robust face tracking using colour Dempster-Shafer fusion and particle filter. In <u>9th Int. Conf. on Information Fusion (ICIF'06)</u>, pages 1–7, Firenze, Italy, 10-13 Jul. 2006. IEEE.
- [52] F. Faux and F. Luthon. Etude de différentes règles de fusion d'information couleur appliquées à la détection d'un visage en temps réel. In <u>Colloque LFA'07 Logique Floue et Applications</u>, Nîmes, Nov. 2007.
- [53] F. Faux and F. Luthon. Théorie de l'évidence pour suivi de visage. <u>Traitement</u> <u>du Signal</u>, 28(5) :515-545, Sept-Oct. 2011. Lavoisier, doi:10.3166/TS.28.515-545, https://www.iieta.org/pdf-viewer/5922.
- [54] F. Faux and F. Luthon. Theory of evidence for face detection and tracking. <u>Int. Journal of</u> Approximate Reasoning, 53(5):728–746, July 2012. Elsevier, doi:10.1016/j.ijar.2012.02.002.
- [55] Francis Faux. <u>Détection et suivi de visage par la théorie de l'évidence</u>. PhD thesis, Université de Pau et des Pays de l'Adour, Anglet, France, October 2009.
- [56] David Forsyth and Jean Ponce. <u>Computer Vision</u> : A Modern Approach. Prentice Hall, 2 edition, 2003.
- [57] Elise Gabarra. De la binarisation de documents vers la reconnaissance de symboles dans l'analyse de schémas électriques. PhD thesis, Université de Pau et des Pays de l'Adour, Bidart, France, December 2008. bourse CIFRE, co-encadrant 50% O. Patrouix, MC61.
- [58] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. <u>IEEE Trans. on Pattern Analysis and Machine Intelligence</u>, 6(6):721–741, November 1984.
- [59] P. Golland and A.M. Bruckstein. Motion from color. <u>Computer Vision and Image Understanding</u>, 68(3) :346–362, December 1997.
- [60] R.C. Gonzalez and R.E. Woods. <u>Digital image processing</u>. Addison-Wesley Publishing Company, Reading, MA., 1993. 4ème réédition en 2017, chez Pearson, London.
- [61] R.C. Gonzalez, R.E. Woods, and S.L. Eddins. <u>Digital Image Processing Using MATLAB</u>. Gatesmark Publishing, 3rd edition, 2020.

- [62] R. Gordon and R.M. Rangayyan. Feature enhancement of film mammograms using fixed and adaptive neighborhoods. Applied Optics, 23(4):560–564, February 1984.
- [63] D. J. Heeger. Model for the extraction of image flow. J. Opt. Soc. Am. A, 4(8) :1455–1471, August 1987.
- [64] B. K. P. Horn and B. G. Schunck. Determining optical flow. <u>Artificial Intelligence</u>, 17:185–203, 1981.
- [65] Y. Z. Hsu, H. H. Nagel, and G. Reckers. New likelihood test methods for change detection in image sequences. <u>Computer Vision</u>, Graphics, and Image Processing, 26:73–106, 1984.
- [66] Olivier Hélénon, Joël Chabriais, Bernard Gibaud, and Denis Mariano-Goulart. <u>Traitement de</u> <u>l'image : de la numérisation à l'archivage et la communication</u>. Imagerie médicale Formation. Elsevier Masson, Issy-les-Moulineaux, avril 2013.
- [67] J.M. Jolion. Les systèmes de vision, Traitement du Signal et de l'Image. Hermès Sciences Europe, Paris, 2001.
- [68] Lin Ke-Fong. Le format JPEG. Login, 113 :52–59, Jan. 2004.
- [69] F. Khattar, F. Dornaika, B. Larroque, and F. Luthon. 3D Object-Camera and 3D Face-Camera Pose Estimation for Quadcopter Control : Application to Remote Labs. In Blanc-Talon J. et al., editor, <u>Advanced Concepts for Intelligent Vision Systems</u>, volume 11182 of <u>LNCS</u>, pages 99–111. Springer, 2018. doi:10.1007/978-3-030-01449-0 9.
- [70] F. Khattar, F. Dornaika, F. Luthon, and B. Larroque. Quadcopter control using onboard monocular camera for enriching remote laboratory facilities. In <u>21th IEEE Int. Conf. on</u> <u>Automation, Quality and Testing, Robotics (AQTR 2018)</u>, Cluj-Napoca, Romania, 24-26 May 2018. doi:10.1109/AQTR.2018.8402730.
- [71] F. Khattar, F. Luthon, B. Larroque, and F. Dornaika. Visual localization and servoing for drone use in indoor remote laboratory environment. <u>Machine Vision and Applications</u>, 32(1), January 2021. doi:10.1007/s00138-020-01161-7.
- [72] Fawzi Khattar. <u>Enriching remote labs with computer vision and drones</u>. PhD thesis, Université de Pau et des Pays de l'Adour, Anglet, France, December 2018. cotutelle avec Espagne (coencadrement 50%).
- [73] R. Köhler. A segmentation system based on thresholding. <u>Computer Graphics and Image</u> Processing GMIP, 15(4) :319–338, April 1981.
- [74] Robert Laganière. <u>OpenCV 2 Computer Vision Application Programming Cookbook</u>. Packt Publishing Ltd., Birmingham, UK, May 2011.
- [75] Sophie Laplace. <u>Conception d'architectures logicielles pour intégrer la qualité de service dans les applications multimédia réparties</u>. PhD thesis, Université de Pau et des Pays de l'Adour, Bayonne, France, May 2006. co-encadrant 50% M. Dalmau, MC27.
- [76] J. Larrieu, F. Clément, B. Held, N. Soulem, F. Luthon, C. Guimon, and H. Martinez. Analysis of microscopic modifications and macroscopic surface properties of polystyrene thin films treated under DC pulsed discharge conditions. <u>Surface and Interface Analysis</u>, 37(6) :544–554, April 2005. doi:10.1002/sia.2047.
- [77] V. Lebugle. Une vision industrielle du traitement de l'image. <u>Le journal de la production</u>, 91 :24–30, septembre/octobre 2008.
- [78] D. Lecomte, D. Cohen, P. de Bellefonds, and J. Barda. Les normes et les standards du multimédia. Dunod, Paris, 2nd edition, 2000.
- [79] M. Liévin and F. Luthon. Nonlinear color space and spatiotemporal MRF for hierarchical segmentation of face features in video. <u>IEEE Trans. on Image Processing</u>, 13(1):63–71, January 2004. doi:10.1109/TIP.2003.818013.
- [80] Marc Liévin. <u>Analyse entropico-logarithmique de séquences vidéo couleur. Application à la segmentation markovienne et au suivi de visages parlants</u>. PhD thesis, Institut National Po-lytechnique, Grenoble, France, September 2000.

- [81] Diane Lingrand. Introduction au traitement d'images. Vuibert, 2 edition, février 2008.
- [82] F. Lopes and M. Ghanbari. Hierarchical motion estimation with spatial transforms. In <u>7th IEEE</u> <u>Int. Conf. Image Processing (ICIP'2000)</u>, Vancouver, Canada, 2000.
- [83] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In <u>Proc. of the International Joint Conference on Artificial Intelligence</u>, pages 121–131, 1981.
- [84] A.C. Luther. <u>Audio et vidéo numériques. Principes et applications</u>. Eyrolles, Paris, 2001.
- [85] F. Luthon. Du mouvement dans les images, ou de l'algorithme à la cadence vidéo. Mémoire de HDR, LIS-INPG, Grenoble, 19 janvier 1999. 100 pages.
- [86] F. Luthon. Face detection using the theory of evidence. In F. Dornaika, editor, <u>Advances in Face Image Analysis : Theory and Applications</u>, chapter 9, pages 169–200. Bentham Science Publishers, 2015. ISBN : 978-1-68108-111-3, doi:10.2174/9781681081106116010012.
- [87] F. Luthon, P. Arnould, C. Baillot, X. Navarro, and J.M. Coutellier. Couleur et ROI : deux options de JPEG2000. Investigations et simulateur MATLAB. In <u>8èmes Journées Compression</u> <u>et Représentation des Signaux Audiovisuels (CORESA'03)</u>, pages 219–222, Lyon, France, Janv. 16-17 2003.
- [88] F. Luthon and B. Beaumesnil. Color and R.O.I. with JPEG2000 for wireless videosurveillance. In <u>IEEE Int. Conf. on Image Processing (ICIP'04), Vol.5</u>, pages 3205–3208, Singapore, October 24-27 2004.
- [89] F. Luthon and B. Beaumesnil. Real-time liptracking for synthetic face animation with feedback loop. In <u>1st International Conference on Computer Vision Theory and Applications (VISAPP'06)</u>, Vol.2, pages 402–407, Setubal, Portugal, 25-28 Feb 2006.
- [90] F. Luthon, B. Beaumesnil, and N.Dubois. LUX color transform for mosaic image rendering. In <u>17th IEEE Int. Conf. on Automation, Quality and Testing, Robotics (AQTR 2010)</u>, pages 93–98, Cluj-Napoca, Romania, May 28-30 2010. Vol. III.
- [91] F. Luthon, A. Caplier, and M. Liévin. Spatiotemporal MRF approach to video segmentation : Application to motion detection and lip segmentation. <u>Signal Processing</u>, 76(1) :61–80, July 1999. doi:10.1016/S0165-1684(98)00247-3.
- [92] F. Luthon and F. Clément. Macroscopic quality measurement of plasma treated polystyrene through computer vision. In <u>IEEE Int. Conf. on Automation, Quality & Testing, Robotics,</u> <u>AQTR'06</u>, pages 321–326, Cluj-Napoca, Romania, May 25-28 2006. Vol. II.
- [93] F. Luthon and D. Dragomirescu. A cellular analog network for MRF-based video motion detection. <u>IEEE Trans. on Circuits and Systems-I</u>, 46(2) :281–293, February 1999. doi:10.1109/81.747202.
- [94] F. Luthon and F. Faux. Audioslide ScienceDirect présentation d'article scientifique. www.youtube.com/watch?v=AaV51gz1GBU.
- [95] F. Luthon and M. Liévin. Lip motion automatic detection. In <u>10th Scandinavian Conference on</u> <u>Image Analysis (SCIA'97)</u>, pages 253–260, Lappeenranta, Finland, June 9-11, 1997.
- [96] F. Luthon and M. Liévin. Entropy power for thresholding technique in image processing. In <u>XI</u> <u>European Signal Processing Conf. (EUSIPCO'02)</u>, pages 605–608, Toulouse, France, Sept. 3-6 2002. Vol. I.
- [97] F. Luthon, M. Liévin, and F. Faux. On the use of entropy power for threshold selection. <u>Signal Processing</u>, 84(10) :1789–1804, October 2004. doi:10.1016/j.sigpro.2004.06.008.
- [98] F. Luthon, X. Navarro, and M. Liévin. Seuillage entropique en traitement d'images. In <u>18e</u> <u>Colloque sur le traitement du signal et des images (GRETSI'01)</u>, pages 353–356, Toulouse, France, 10-13 Sept. 2001. Vol.2.
- [99] F. Luthon, G. V. Popescu, and A. Caplier. An MRF-based motion detection algorithm implemented on analog resistive network. In J. O. Eklundh, editor, <u>3rd European Conference on Computer</u> <u>Vision (ECCV'94)</u>, pages 167–174, Stockholm, Sweden, May 2-6, 1994. Springer-Verlag. LNCS Vol. 800.

- [100] Franck d'Images : Luthon. Initiation au Traitement Contours, Couleurs, Mouvements Cours, exercices TP corrigés. Références Sciences. \_  $\operatorname{et}$ Ellipses Ed. Marketing, Paris, Sept. 2021.288pages, ISBN : 9782340-056749,https://www.editions-ellipses.fr/recherche?controller=search&s=luthon.
- [101] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In Proc. of the fifth Berkeley symposium on mathematical statistics and probability, volume 1-14, pages 281–297, 1967.
- [102] S. G. Mallat. Multifrequency channel decompositions of images and wavelet models. <u>IEEE Trans.</u> on ASSP, 37(12) :2091–2110, December 1989.
- [103] A. Marion. Introduction aux techniques de traitement d'images. Eyrolles, Paris, 1987.
- [104] D. Marr. Vision. Freeman, 1980.
- [105] Henri Maître. Le traitement des images. Hermès. Lavoisier, janvier 2003.
- [106] J. Murray and W. Van Ryper. Encyclopedia of Graphics File Formats. O'Reilly, 1994.
- [107] H. H. Nagel. On the estimation of optical flow : relations between different approaches and some new results. Artificial Intelligence, 33 :299–324, 1987.
- [108] H. H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. <u>IEEE Trans. on Pattern Analysis and Machine</u> Intelligence, 8(5) :565–593, September 1986.
- [109] Mohamed Najim. <u>Synthèse de filtres numériques en traitement du signal et des images</u>. Hermes, Lavoisier, Paris, 2004.
- [110] A. N. Netravali and J. D. Robbins. Motion-compensated television coding : Part 1. <u>Bell System</u> <u>Technical Journal</u>, 58(3) :631–670, March 1979.
- [111] N. Otsu. A threshold selection method from gray-level histograms. <u>IEEE Trans. on Systems</u>, Man and Cybernetics, 9(1) :62–66, January 1979.
- [112] Nulle part ailleurs. Vision, une chute amortie. Jautomatise, 72:29–33, Septembre-Octobre 2010.
- [113] E. Peli. Contrast in complex images. J. Optical Soc. America A, 7(10) :2032–2040, October 1990.
- [114] W. Pieczynski. Champs de Markov cachés et estimation conditionnelle itérative. <u>Traitement du</u> <u>Signal</u>, 11(2) :141–153, 1994.
- [115] W.K. Pratt. <u>Digital Image Processing</u>. John Wiley, New York, 2nd edition, June 1991. 4ème réédition en 2007, chez Wiley & Sons, New Jersey.
- [116] M. Rabbani and R. Joshi. An overview of the JPEG 2000 still image compression standard. Signal Processing : Image Communication, 17 :3–48, 2002. www.jpeg.org/JPEG2000.htm.
- [117] C.G. Relf. Image Acquisition and Processing with LabVIEW. CRC Press, Boca Raton, 2004.
- [118] A. Rosenfeld and A. Kak. Digital Picture Processing. Academic Press, 1976.
- [119] John C. Russ and F. Brent Neal. <u>The image processing handbook</u>. CRC Press, Boca Raton, 7 edition, September 2018.
- [120] C. E. Shannon. A mathematical theory of communication. <u>The Bell System Technical Journal</u>, 27:379–423,623–656, July, October 1948.
- [121] J. Shen and S. Castan. An optimal linear operator for edge detection. In <u>Proc. CVPR'86</u>, pages 109–114, Miami, USA, June 1986. IEEE.
- [122] K. Skifstad and R. Jain. Illumination independent change detection for real world image sequences. Computer Vision, Graphics, and Image Processing, 46 :387–399, 1989.
- [123] M. Soriano, B. Martinkauppi, S. Huovinen, and M. Lakksonen. Using the skin locus to cope with changing illumination conditions in color-based face tracking. In <u>Proc. IEEE Nordic Signal</u> <u>Processing Symposium (NORSIG'2000)</u>, Kolmaarden, Sweden, June 2000.
- [124] D.S. Taubman and M.W. Marcellin. JPEG 2000 Image Compression Fundamentals, Standards and Practice. Kluwer Academic Publishers, The Netherlands, 2001.

- [125] V. Torre and T. A. Poggio. On edge detection. <u>IEEE Trans. on Pattern Analysis and Machine</u> Intelligence, 8(2) :147–163, March 1986.
- [126] T. Totozafiny, F. Luthon, and O. Patrouix. Pros and Cons of the Nonlinear LUX Color Transform for Wireless Transmission with Motion JPEG2000. In Proc. Image and Vision Computing New Zealand (IVCNZ'06), pages 49–54, Great Barrier Island, New Zealand, November 2006.
- [127] T. Totozafiny, O. Patrouix, F. Luthon, and J.M. Coutellier. Motion reference image JPEG2000 : Road surveillance application with wireless device. In <u>Visual Communications and Image</u> <u>Processing (VCIP'05)</u>, pages 1839–1848, Beijing, China, Jul. 12-15 2005. Proc. SPIE Vol. 5960 <u>Issue 4.</u>
- [128] T. Totozafiny, O. Patrouix, F. Luthon, and J.M. Coutellier. Dynamic Background Segmentation for Remote Reference Image Updating within Motion Detection JPEG2000. In <u>Int. Symposium</u> <u>Industrial Electronics, ISIE'06</u>, pages 505–510, Montréal, Canada, 9-13 Jul 2006. Vol.1.
- [129] Théodore Totozafiny. Compression d'images couleur pour application à la télésurveillance routière par transmission vidéo à très bas débit. PhD thesis, Université de Pau et des Pays de l'Adour, Bidart, France, July 2007. bourse CIFRE, co-encadrant 50% O. Patrouix, MC61.
- [130] A. Trémeau, C. Fernandez-Maloigne, and P. Bonton. <u>Image Numérique Couleur</u>. De l'acquisition <u>au traitement</u>. Sciences Sup. Dunod, Paris, 2004.
- [131] D. R. Walker and K. R. Rao. Improved pel-recursive motion compensation. <u>IEEE Trans. on</u> Communications, 32(10) :1128–1134, October 1984.
- [132] Jakub Zimmermann. Faites ressortir les détails de vos images. Login, 98:52–53, Sept. 2002.