
PROJET SYNTHÈSE

420-B65-VM – Automne 2011

TABLE DES MATIÈRES

Présentation générale	5
Mise en contexte	5
Déroulement général du projet	5
Contraintes générales.....	5
Contraintes associées à la démarche du projet	5
Contraintes techniques générales	6
Les 11 livrables du projet.....	7
1 ^{ier} livrable – Document de conception.....	7
2 ^{ième} livrable – Document de planification générale	8
3 ^{ième} , 4 ^{ième} et 5 ^{ième} livrables – Rapports des <i>Sprints 1, 2 et 3 (3 livrables)</i>	9
6 ^{ième} , 7 ^{ième} et 8 ^{ième} livrables – Votre projet (<i>3 livrables</i>)	9
9 ^{ième} livrable – Manuel de l’usager	10
10 ^{ième} livrable – Site Web	10
11 ^{ième} livrable – Présentation	11
Projet A – Jeu de type « <i>Tower Defense</i> »	12
Présentation générale	12
Objectifs de réalisation.....	13
Contraintes	13
Interface usager.....	13
Déroulement du jeu.....	13
Types de créatures	14
Types de tours	14
Approche technologique	15
Évaluation.....	15
Projet B – Jeu de type « <i>RPG</i> ».....	16
Présentation générale	16
Objectifs de réalisation.....	16
Contraintes	17
Interface usager.....	17
Déroulement du jeu.....	17
Approche technologique	18
Évaluation.....	19
Projet C – Simulateur d’un écosystème	20
Présentation générale	20
Objectifs de réalisation.....	20
Contraintes	21
Interface usager.....	21
Définition des entités	21

Le simulateur	22
Approche technologique	22
Évaluation.....	23
Projet D – Gestion de feuilles de temps	24
Présentation générale	24
Objectifs de réalisation.....	24
Contraintes	25
Données à manipuler.....	25
Base de données.....	26
Rapport à présenter.....	26
les Interfaces usager	27
Approche technologique	27
Évaluation.....	27
Projet E – Traitement d'image.....	28
Présentation générale	28
Objectifs de réalisation.....	28
Précisions.....	28
Contraintes	29
Interface usager et corps du programme	29
Filtres génériques à programmer	29
Approche technologique	39
Évaluation.....	39
Projet F – Vision artificielle	40
Présentation générale	40
Objectifs de réalisation.....	40
Logiciel à réaliser	41
Mise en contexte	41
Approche algorithmique proposée.....	42
Approche technologique	48
Évaluation.....	48
Projet G – Simulateur d'un réseau informatique	49
Présentation générale	49
Objectifs deréalisation.....	49
Logiciel à réaliser	49
Contraintes	50
Approche technologique	51
Évaluation.....	51
Annexe 1 : UML - Diagramme de classes.....	52
Qu'est-ce que UML.....	52
Diagramme de classes	52

La classe.....	53
L'héritage.....	53
L'association	53
L'agrégation et la composition	53
Exemple complet.....	54
Annexe 2 : Espaces de couleur	56
La couleur	56
Les espaces colorimétriques.....	57
Algorithmes de transformation entre les espaces RVB et TSV.....	59
Application des transformations présentées	60
Exemple de transformation d'images.....	60
Exemple pour l'interprétation d'images	60
Annexe 3 : Filtres gaussiens 1D et 2D	61
Sommaire	61
Espace 1D	61
Espace 2D	64
Annexe 4 : Filtre directionnel de Sobel.....	71
Sommaire	71
Images de l'exemple en pleine résolution	74
Annexe 5 : Uniformisation d'éclairage.....	77
Sommaire	77
Concept	77
Réalisation	77
Étape 1 – Filtre de convolution	78
Étape 1b – Moyenne des images	79
Étape 2 – Filtre maximum.....	79
Étape 3 – Filtre de convolution	79
Étape 4 – Uniformisation	79
Étape 5 – Normalisation	80
Optimisation.....	80
Annexe 6 : Algorithmes de remplissage.....	84
Sommaire	84
Concept	84
Algorithme.....	85
Paramètres d'entrée.....	85
Résultats de sortie	85
Algorithme récursif à 4 voisins.....	85
Algorithme récursif à 8 voisins.....	85
Algorithme itératif à 4 voisins	86
Algorithme itératif à 8 voisins.....	87

Annexe 7 : Remplissage de trous d'une images binaires	88
Sommaire	88
Concept	88
Étape 1 – Identification d'un pixel appartenant au fond de l'image	88
Étape 2 – Remplissage du fond de l'image	88
Étape 3 – Inversion de l'image obtenu	89
Étape 4 – Construction de l'image finale	89
Exemple.....	89
Annexe 8 : Étiquetage d'une image binaire	92
Sommaire	92
Concept	92
Exemple.....	93
Annexe 9 : Extraction de blobs	95
Sommaire	95
Avant d'aller plus loin	96
Concept	97
Algorithme.....	98
Paramètres d'entrée.....	98
Résultats de sortie	98
Algorithme général.....	98
Algorithme d'extraction du contour	100

PRÉSENTATION GÉNÉRALE

MISE EN CONTEXTE

Tel que stipulé dans le plan de cours, ce projet vous donne l'opportunité de réaliser entièrement une application informatique et d'en assurer la mise en œuvre complète. Ainsi, à partir d'un devis présentant une idée générale et certaines contraintes spécifiques, vous allez devoir implémenter une application fonctionnelle en passant par toutes les étapes de production qui vous sont demandées.

Comme l'indique le titre du cours, ce projet se veut une synthèse de tous les apprentissages que vous avez accumulés pendant votre DEC. Il est donc essentiel de mettre à profit tous les aspects techniques que vous avez acquis afin de rendre hommage à l'informaticien(ne) que vous êtes rendu(e)!

DÉROULEMENT GÉNÉRAL DU PROJET

D'abord, vous devrez choisir un projet parmi les devis qui vous seront présentés. Dès ce choix effectué, vous aborderez le développement de votre application. Pour donner suite aux notions que vous avez acquises lors cours précédents, la gestion du projet s'effectuera selon la méthode [Scrum](#) qui est une des méthodes [Agile](#).

Chacun des devis qui vous seront proposés comportent des contraintes qui leur sont propres et qui seront abordées à même leur devis respectifs. Néanmoins, chaque projet présente des éléments communs et des contraintes d'ordre générale qui sont abordées un peu plus loin dans ce chapitre.

Les contraintes les plus significatives du projet sont sans contredit les méthodes de travail et le respect des échéanciers qui vous sont demandées. Tout au long du projet, vous devez suivre une démarche serrée pour laquelle vous aurez des livrables précis et des documents à produire. Outre les documents conventionnels que vous aurez à produire, vous allez devoir monter un site web et faire une présentation orale de votre projet.

Le projet qui vous est demandé est exigeant pour le temps que vous avez de disponible. Ainsi, vous allez devoir gérer vos activités très efficacement afin d'éviter les pièges typiques de tels projets. D'ailleurs, vous devez vous y mettre dès la première semaine pour ne pas accumuler de retard. Finalement, ne jamais hésiter à poser des questions sur la gestion du projet ou tout autre aspect technique.

CONTRAINTE GÉNÉRALES

CONTRAINTE ASSOCIÉES À LA DÉMARCHE DU PROJET

Le projet est divisé en 4 *Sprints* distincts :

- ***Sprint 0 – Conception et planification*** : Cette phase vise à réaliser la mise sur pied du projet. Après s'être assuré de bien comprendre le devis, vous devez faire la conception technique du projet ainsi que la planification des activités (les trois *sprints* restants).
- ***Sprint 1 – Implémentation des éléments essentiels*** : Cette phase permet de produire les assises techniques du projet. À la fin de cette étape, vous devez avoir une version préliminaire **fondionnelle** de votre projet. En vous basant sur les parties essentielles, vous devez mettre en

place l'infrastructure de votre application. De plus, votre application doit déjà offrir quelques fonctionnalités de bases.

- **Sprint 2 – Augmentation fonctionnelle** : Cette phase consiste à atteindre un avancement intermédiaire du projet. Comme pour le premier *sprint*, vous devez déterminer des objectifs intermédiaires pour lesquels vous aurez de nouvelles fonctionnalités. Généralement, votre application devrait déjà contenir l'essentiel de tous les modules.
- **Sprint 3 – Livraison finale** : C'est ici que vous peaufinez votre application, faites les tests de fonctionnalité et apporter certaines modifications. Aussi, vous allez devoir terminer la réalisation de tous les documents finaux qui sont à remettre (incluant le site web et votre exposé).

Il est important de bien planifier votre projet afin qu'il soit réalisable dans le temps que vous avez de disponible. Le calendrier des activités donne une excellente vue d'ensemble du déroulement global du projet.

Semaine	Nombre de semaines	Description des activités		Remises et évaluations	Date de remise des travaux Toujours au début du cours sans faute!
1	3	Sprint 0 Conception et planification	20 %	• Document de conception	15 %
2				• Document de planification générale	5 %
3					16 septembre
4	4	Sprint 1 Implémentation des éléments essentiels	15 %	• Projet : Implémentation initiale	10 %
5				• Rapport du <i>Sprint 1</i>	5 %
6					7 octobre
7	3	Sprint 2 Augmentation fonctionnelle	15 %	• Projet : Augmentation fonctionnelle	10 %
8				• Rapport du <i>Sprint 2</i>	5 %
9					11 novembre
10					
11	5	Sprint 3 Livraison finale	50 %	• Version finale de l'application	30 %
12				• Rapport du <i>Sprint 3</i>	5 %
13				• Manuel de l'usager	5 %
14				• Site Web	5 %
15				• Présentation du projet	5 %

CONTRAINTE TECHNIQUE GÉNÉRALE

- Les projets se font seul et aucune équipe n'est permise. Néanmoins, l'entre aide constructive est toujours encouragée.
- Outre le respect des contraintes spécifiées dans les devis, vous avez pleine liberté pour réaliser le projet qui vous intéresse.
- Les outils de développement sont laissés à votre entière discréction en autant qu'ils soient disponibles sur les plateforme du cégep (*IDE*, langage de programmation, environnement de développement, outils de documentation et autres). Néanmoins, malgré la flexibilité qui vous est offerte, vous devez comprendre que le contexte de ce projet se prête mal à l'apprentissage de nouveaux outils informatiques. Le temps que vous avez à votre disposition est déjà très serré

- pour le projet à réaliser. Il est fortement suggéré d'utiliser les outils pour lesquels vous avez déjà certaines connaissances et pour lesquels vous désirez développer une plus grande expertise.
- La validation de votre approche avec l'enseignant est essentielle. Lorsque vous aurez identifié les détails de votre projet, discutez-en avec lui.
 - Les échéanciers ne sont pas négociables.
 - La réalisation de ce projet repose beaucoup sur vos aptitudes à solutionner par vous-même les problèmes que vous rencontrerez. D'ailleurs, être capable d'identifier les problématiques et savoir où trouver les solutions sont des aptitudes essentielles que vous devez développer afin de mieux vous préparer pour le marché du travail.
 - Si au cours de la session vous désirez changer de projet, il sera permis de le faire sous certains conditions :
 - Vous devrez refaire toute la démarche de conception et produire les documents demandés à cet effet.
 - Vous aurez une pénalité croissante pour chaque semaine de la session selon cette formule :

$$\text{Pénalité} = 2.5\% \times \text{semaine} - 2.5\%$$

LES 11 LIVRABLES DU PROJET

1^{ER} LIVRABLE – DOCUMENT DE CONCEPTION

Les trois premières semaines de votre projet doivent vous permettre de faire une conception étayée, approfondie et relativement détaillée de votre projet. Le document de conception que vous allez produire vous serviront de référence pendant la production de votre application et d'élément de validation technique de votre approche.

Ainsi, le document de conception devient le devis technique de votre développement. L'usage d'un tel document joins à la méthode de gestion de développement *Agile*, vous permettra de mettre toutes les chances de votre côté afin d'atteindre vos objectifs. De plus, dans le cycle de production d'un projet informatique, il est important de valider l'interprétation que vous faites du devis qui vous est donné. Cette validation avec les personnes qui l'ont rédigés vous permet d'éviter dès l'amorce du projet la production d'une application qui diffère du devis initial.

Voici les sections que votre rapport de conception doit présenter :

• Présentation générale de votre projet	1.0 %
• Présentation des caractéristiques particulières – Votre touche personnelle!	3.0 %
• Aspects techniques de votre conception :	
○ Cas d'usage	3.0 %
○ Diagramme CRC étendu ¹ (C lasses- R esponsabilités- C ollaboration) ou diagramme de classe UML	5.0 %
○ Les croquis de vos interfaces usagers	3.0 %
	15.0 %

¹ Au diagramme que vous êtes habitués de faire, ajouter une description précise des types de données pour chaque attributs (variables) et la signature complète de vos méthodes (fonctions). De plus, vous devez inclure une légende claire.

Ce projet de fin d'étude est une excellente occasion de mettre de l'avant les notions de conception orientée objet. D'ailleurs, on vous demande de faire un effort particulier pour bien utiliser les notions d'encapsulation, d'héritage et de polymorphisme. Le niveau d'abstraction que vous aurez créé sera l'un des points techniques évalués les plus importants (qui se reflétera dans votre diagramme CRC ou UML).

Vous devez nommer ce document : B65_S0_Conception_VotreNom.docx.

2^{ME} LIVRABLE – DOCUMENT DE PLANIFICATION GÉNÉRALE

Ce document vise la documentation de la planification détaillée de chacun des *sprints*. Votre planification doit inclure une liste de toutes les tâches des modules principaux que vous avez défini. Ces tâches doivent viser un découpage modulaire du projet de façon telle à ce que chaque module amène une fonctionnalité particulière. Pour chacune de ces tâches, vous devez indiquer :

- À quel objectif général il est relié (conception, interface usager, algorithmie, ...) 0.5 %
 - Un numéro de tâche unique 0.5 %
 - Une description technique de haut niveau de la tâche 1.0 %
vous pouvez faire référence à certains éléments de votre document de conception et vous devez ajouter les précisions techniques nécessaires (avec Excel, utilisez l'outil Nouveau commentaire disponible dans l'onglet Révision)
 - Une liste des tâches préalables à la tâche courante (tâches prédecesseurs); 0.5 %
 - Une cote de priorité 0.75 %
1 à 3 où 1 étant le plus prioritaire – cette cote doit vous permettre d'établir quels éléments seront évincés si vous manquez de temps
 - Une cote associé à la difficulté technique de réalisation 0.75 %
1 à 3 où 1 représente une plus grande difficulté pour vous
 - Un estimation du temps requis 1.0 %
- 5.0 %**

N'oubliez pas que vous servirez de ce document pour faire le suivi de vos activités tout au long de la session. Voici un exemple de fichier Excel présentant une partie sommaire d'un tel outil de planification.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	Planification et suivi du projet																						
2																							
3																							
4																							
5																							
6	Objectif spécifique	Description technique de la tâche																					
7	N° de tâche	Description générale de la tâche																					
8																							
9																							
10																							
11																							
12																							
13																							
14																							
15																							
16																							
17																							
18																							
19																							
20																							
21																							
22																							
23																							
24																							
25																							
26																							
27																							
28																							
29																							



Vous devez nommer ce document `B65_0_Planif_VotreNom.xlsx`.

3^{ÈME}, 4^{ÈME} ET 5^{ÈME} LIVRABLES – RAPPORTS DES SPRINTS 1, 2 ET 3 (3 LIVRABLES)

Lorsqu'un sprint est terminé, vous devez faire le bilan des activités et produire un petit rapport d'avancement. Ce document est plutôt simple et reprend en grande partie les éléments que vous avez défini dans votre document de planification.

Pour chaque tâche, on doit retrouver :

- | | |
|---|-------|
| • La somme du temps investi | 1.5 % |
| • L'avancement en % de la tâche dans la colonne du Sprint courant | 1.5 % |
| <i>Vous devez aussi définir un code de couleur pour rehausser votre document et bien montrer l'état de la tâche</i> | |
| • Si une tâche n'est pas terminée, expliquer quelles ont été les raisons du retard
ainsi que les moyens qui seront pris pour la terminer | 2.0 % |
| Pour chaque livrable | |
| 5.0 % | |
| Totalisant pour les 3 livrables | |
| 15.0 % | |

Vous devez remettre le même document mis à jour à chaque remise demandée. Le document doit être nommé comme suit :

- `B65_X_Sprint_VotreNom.xlsx`

6^{ÈME}, 7^{ÈME} ET 8^{ÈME} LIVRABLES – VOTRE PROJET (3 LIVRABLES)

Pour chaque *sprint*, vous devrez démontrer l'avancement de votre projet. Pour les Sprints 1 et 2, vous aurez une évaluation personnelle avec l'enseignant. Pour le Sprint 3, vous aurez à remettre la version fonctionnelle de votre application.

Pour les deux premiers *Sprints* vous aurez en début de session un calendrier sur lequel vous retrouverez deux rendez-vous avec votre enseignant. Vous devrez obligatoirement vous présenter aux heures convenus pour faire ces évaluations qui valent chacun 10% de la session. Ces évaluations sont courtes et visent une estimation de l'avancement de votre projet. Vous serez évalué sur l'avancement, le suivi que vous en faites et la compréhension du problème que vous avez et non sur la qualité de votre code ou les éléments d'apparence graphique. Ces évaluations sont d'environ 10 minutes seulement et se font seul à seul.

Pour le dernier *Sprint*, on vous demande de mettre à l'intérieur d'un fichier *zip*, un dossier contenant tous les éléments de votre projet :

- Un dossier contenant votre projet (le code source et tous les fichiers de projet);
- Un dossier contenant toutes les librairies externes que vous avez utilisées;
- Un dossier contenant un exécutable fonctionnel de votre application (lorsque applicable);
- Dans la racine de votre fichier *zip*, ajouter un fichier `Lisez_moi.txt` expliquant la procédure complète d'installation pour votre projet. La procédure d'installation décrite doit être exhaustive comme si vous vous adressiez à une personne qui a aucune connaissance informatique. Assurez-vous de remettre un fichier `Lisez_moi.txt` clair et complet car sans ce fichier votre travail ne sera pas corrigé et vous aurez automatiquement zéro.

Ce fichier doit se nommer : B65_3_Projet_VotreNom.ZIP

La pondération de ces trois évaluations est la suivante :

• Sprint 1 – Évaluation individuelle 1	10.0 %
• Sprint 2 – Évaluation individuelle 2	10.0 %
• Sprint 3 – Version finale	30.0 %
Totalisant pour les 3 livrables	50.0 %

9^{ÈME} LIVRABLE – MANUEL DE L’USAGER

Vous devez produire un document d’utilisation de votre application. Ce document vise une utilisation simple de toutes les fonctions disponibles par un usager non technique. Ce que doit contenir votre manuel :

• Présentation générale de votre application	0.5 %
• Présentation des fonctionnalités et caractéristiques de votre application <i>À ces listes, vous devez inclure les limitations et contraintes s'il y en a</i>	1.5 %
• Présentation d'un cas d'utilisation simple	1.0 %
• Présentation des interfaces usagers ne présentez que les principaux si vous en avez plusieurs	1.5 %
	5.0 %

Vous devez nommer votre document : B65_3_Manuel_VotreNom.docx

10^{ÈME} LIVRABLE – SITE WEB

Vous avez à monter un petit site web, accessible directement à partir de votre page personnelle du CVM. Ce site doit servir de plateforme pour faire la promotion de votre projet et potentiellement la remise de vos documents. Sans vous lancer dans une élaboration trop ardue, vous serez évalué sur la pertinence des informations qui y seront disponibles et sur la présentation de ces dernières.

En arrivant sur votre site, on doit y retrouver ces informations :

• Présentation sommaire de votre projet	0.5 %
• Dans un but de promotion, quelques captures d’écran avec explications (au moins 3)	1.5 %
• Présentation de tous les outils technologiques utilisés pour le développement <i>/IDE, langages de programmation, librairies, moteur de base de données, etc.</i>	1.0 %
• Temps total investi (réaliste)	0.5 %
• Vos coordonnées afin de vous contacter	0.5 %
• La présentation générale du site sera aussi évaluée	1.0 %
	5.0 %

Sans être obligatoire, vous pouvez mettre l’exécutable de votre projet disponible pour tous. Dans ce cas, n’oubliez pas de joindre à votre projet, les deux fichiers de documentation (*Manuel de l’usager* et *Lisez_moi.txt*).

Finalement, vous devez rendre disponible un fichier compressé contenant tous les fichiers de votre site. Le nom de remise est `B65_3_SiteWeb_VotreNom.ZIP`.

11^{ÈME} LIVRABLE – PRÉSENTATION

Lors de la dernière période, vous allez devoir présenter votre projet à l'ensemble de la classe et possiblement d'autres auditeurs. Plusieurs objectifs sont visés par cet exercice :

- vous pratiquer à présenter un projet à un groupe de personnes;
- faire profiter aux autres votre point de vue et l'approche que vous avez adoptés face au devis;
- présenter quels sont les outils technologiques que vous avez utilisés et discuter de l'expérience que vous en tirez.

Partant de ces objectifs, votre présentation doit respecter les contraintes suivantes :

• Votre exposé doit durer entre 5 et 7 minutes.	0.25 %
• Vous devez présenter tous les aspects suivants de votre projet :	
○ l'approche que vous avez préconisée et les adaptations personnels que vous avez faites du devis initial;	0.5 %
○ l'interface usager principal;	1.0 %
○ un exemple complet d'utilisation;	1.0 %
○ l'environnement techniques que vous avez utilisé;	1.0 %
○ les apprentissages importants que vous avez tirés de ce projet.	0.5 %
• Vous n'avez droit qu'à 2 minutes pour vous installer afin de faire votre présentation.	0.25 %
• Vous allez avoir une période de question d'environ 2 minutes mise à votre disposition.	0.5 %
	5.0 %

Vous n'avez aucune contrainte pour ce qui est du support de votre présentation. Vous pouvez utiliser votre site Web, une présentation PowerPoint ou n'importe quel outil qui peut vous être utile. Néanmoins, vous devez penser aux nombreuses contraintes que nous avons (par exemple, vous n'avez qu'à penser aux 2 minutes mises à votre disposition pour vous installer).

Pour faire un exemple complet d'utilisation de votre application, vous n'êtes pas obligé de faire fonctionner votre application. Vous pouvez tout simplement utiliser des captures d'écran et les afficher les unes après les autres pour illustrer vos propos.

Si vous utilisez un support pour vous aider, vous devez le remettre au moment de la remise. Le nom de votre fichier doit être : `B65_3_Presentation_VotreNom.xyz`.

PROJET A – JEU DE TYPE « *TOWER DEFENSE* »

PRÉSENTATION GÉNÉRALE

Ce projet vous propose de créer un jeu de type « *Tower Defense* ». Ce type de jeu fait partie de la famille des **RTS** (« *Real Time Strategy* » – jeu de stratégie en temps réel).

Le concept du jeu est simple et consiste à empêcher certaines créatures d'atteindre un objectif particulier. Les créatures se déplacent à partir d'un point de départ et se dirigent vers l'objectif que vous devez défendre à tout prix. Votre mécanisme de défense consiste à positionner des tours armées qui élimineront ces créatures.

La particularité de ce type de jeu est que vous avez la liberté de positionner vos tours là où vous le jugez pertinent. Plus le jeu avance et plus les créatures sont puissantes. Par ailleurs, vous pouvez améliorer vos tours en les rendant plus performantes selon divers critères.

L'objectif du jeu peut être simplement de résister le plus longtemps possible ou de réussir certains scénarios prédefinis.

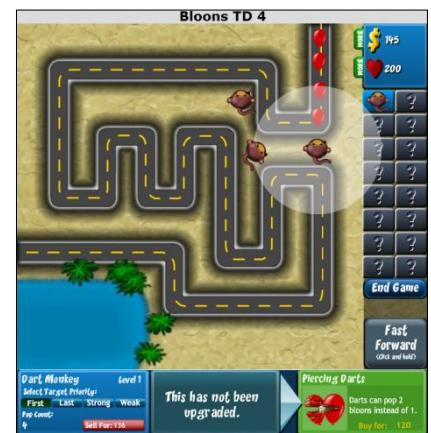
Le site www.towerdefence.net est un des sites les plus populaires présentant une grande collection de ce type de jeu. À titre de référence, essayez au moins ces jeux gratuits qui sont des classiques et des références dans le genre :



[Desktop Tower Defense Pro](#)



[Onslaught](#)



[Bloons TD 4](#)

OBJECTIFS DE RÉALISATION

Ce projet vous permettra de développer une application présentant plusieurs caractéristiques très intéressantes:

- Une interface usager élaborée
- Une interaction temps réel avec la scène de jeu
- Un simulateur permettant :
 - une gestion temporelle d'évènement impliquant la synchronisation graphique;
 - mise en place d'une infrastructure orienté objet pertinente;
 - développement d'algorithme divers;
 - gestion des données de l'évolution du jeu.
- Application ludique généralement très appréciée.
- Un défi multidisciplinaire vous permettant de nombreuses possibilités.

N'oubliez pas de définir un projet réaliste respectant les contraintes de temps disponible et vos connaissances technologiques.

CONTRAINTE

Interface usager

À ce niveau, vous pouvez faire à peu près ce que vous voulez mais vous devez au moins respecter ces quelques règles de base :

- offrir une page d'accueil (« *splash screen* »)
- pendant le déroulement de la partie, vous devez offrir :
 - la plus grande surface de jeu possible;
 - une zone de contrôle vous permettant d'interagir avec les options principales de jeu (par exemple, pause et tous les autres contrôles)
 - une zone de contrôle pour les outils de construction des tours de défense.
- vous devez aussi proposer à partir de la page d'accueil une fenêtre expliquant les grandes lignes de votre jeu.

Déroulement du jeu

- Au démarrage du logiciel, vous devez présenter la page d'accueil (« *splash screen* ») offrant le choix d'amorce que vous voulez (par exemple, démarrer immédiatement une partie, aller dans la fenêtre des options).
- Vous devez offrir au moins 2 modes de jeu :
 - partie standard;
 - partie « *sandbox* » (un mode « *sandbox* » très facile à créer est de démarrer une partie standard avec beaucoup d'argent)
- La progression du jeu doit faire en sorte que les créatures qui se présentent sont de plus en plus fortes et résistantes.
- Vous avez le choix de définition de sentier pour les créatures :
 - soit un sentier prédéfini;
 - soit un sentier définis aléatoirement (avec ou sans critères);

- soit aucun sentier (attention, cette approche est beaucoup plus complexe à cause de l'aspect de la planification de trajectoire à gérer)
- Concernant l'aspect multimédia de votre jeu, vous devez inclure :
 - un graphique différent pour chaque famille de créature;
 - un graphique différent pour chaque type de tour;
 - un graphique différent pour chaque type de projectile;
 - la gestion d'effets sonores n'est pas obligatoire;
 - sans être obligatoire, vous pouvez inclure des petites animations pour les moments particuliers (comme lors d'un impact d'un projectile avec une créature).
- Interaction pendant la partie :
 - Vous devez pouvoir créer des tours à n'importe quel endroit sur la surface de jeu (sauf sur le sentier ou d'autres contraintes que vous pouvez ajouter).
 - Vous pouvez connaître les caractéristiques d'une unité simplement en cliquant sur cette dernière (créature et tour). Cet aspect doit aussi inclure une façon de rehausser l'unité sélectionnée.
 - Vous pouvez modifier les caractéristiques de vos tours à n'importe quel moment (faire les « *upgrade* » ou même vendre vos tours).
 - Vous devez offrir plusieurs contrôle sur le temps pendant une partie:
 - fin de la partie (incluant une confirmation);
 - redémarrer une nouvelle partie (incluant une confirmation);
 - pause et reprise de la partie;
 - différents niveaux de vitesse de la partie (au moins : 1x, 2x et 3x)
 - lancer immédiatement la prochaine vague de créatures*.
- Vous devez définir les critères qui définissent la fin d'une partie. Par exemple, lorsque 5 créatures ont atteints la base centrale.

Types de créatures

- Vous devez présenter au moins trois familles de créatures. Chacune doit posséder des caractéristiques différentes.
- Vous devez considérer au moins tous ces paramètres pour vos familles de créatures :
 - Niveau d'énergie initial
 - Niveaux de résistance en fonction des *n* types de tir
 - Vitesse d'avance

Types de tours

- Vous devez présenter au moins trois familles de tours pour lesquelles chacune possède des effets différents sur le jeu (types de projectile, effet général comme un ralentissement, etc.). Chacune doit posséder des caractéristiques différentes.
- Vous devez considérer au moins tous ces paramètres pour vos familles de tours :
 - Temps de fabrication
 - Types de projectile (lorsque ce critère s'applique)
 - Vitesse d'avance
 - Mode de suivi (par exemple) :
 - direct comme un laser
 - vers l'avant à l'aveugle comme un projectile d'arme à feu
 - vers l'avant avec guidage comme un missile

- Force d'impact
- Fréquence de tir
- Portée de tir
- Prix et scénarii pour les « *upgrades* » (le premier « *upgrade* » fait quoi et coûte combien par exemple)
- Prix de revente
- Facultativement, type de cible
 - plus près / plus loin
 - moins fort / plus fort
 - moins rapide / plus rapide

Approche technologique

- Vous devez obligatoirement mettre de l'avant le concept de la programmation orienté objets pour la création de vos unités. Les notions d'encapsulation, d'héritage et de polymorphisme doivent être mis de l'avant. Ce sera un critère d'évaluation particulièrement important.

ÉVALUATION

• Gestion des évènements de la simulation	35 %
○ Déroulement général du jeu (appréciation du « <i>game play</i> »)	10 %
○ Gestion des options de temps (stop, pause et vitesse de jeu)	5 %
○ Mouvement adéquat des créatures	5 %
○ Mouvement adéquat des projectiles	5 %
○ Évolution de la partie	5 %
○ Définition et gestion des sentiers pour les créatures	5 %
• Interface usager et interaction	25 %
○ « <i>Splash screen</i> »	2 %
○ Page explicative du jeu	2 %
○ Offrir deux modes de jeu (standard et « <i>sandbox</i> »)	1 %
○ Aire de jeu bien conçu (ergonomie et qualité visuelle)	10 %
○ Insertion des tours sur l'aire de jeu	5 %
○ Sélection des tours et ajustement des paramètres (« <i>upgrade</i> » et autres)	5 %
• Respect des contraintes générales	25 %
○ Choix judicieux de la structure objet	10 %
○ Respect et bonne gestion des paramètres de base pour les créatures	5 %
○ Respect et bonne gestion des paramètres de base pour les tours	5 %
○ Respect des éléments multimédia	5 %
• Autres	15 %
○ Originalité et aspect soigné	5 %
○ Niveau de complexité technique du projet	5 %
○ Aspect soigné du code	5 %

PROJET B – JEU DE TYPE « *RPG* »

PRÉSENTATION GÉNÉRALE

Ce projet vous propose de créer un jeu de type « *RPG* » (« *Role-Playing Game* » – jeu de rôle). Généralement, ce type de jeu consiste à réaliser des quêtes diverses tout en faisant évoluer un personnage principal que le joueur incarne.

Ce type de jeu est certainement l'un des plus joué et en une quantité impressionnante de variantes. Principalement, vous incarnez un personnage possédant certains attributs qui évolueront au fil de la partie. Vous pouvez avoir plusieurs objectifs à réaliser allant de la simple survie à des quêtes complexes. Évidemment, plus le jeu avance, plus les défis sont difficiles. Ce sont les différentes évolutions du personnage qui lui permettront de vaincre les différentes épreuves.

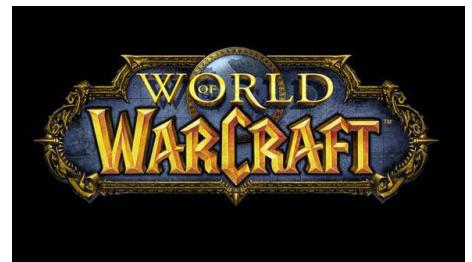
Ce type de jeu vidéo a débuté avec Dragon Quest et suivi de près par Final Fantasy créés en 1986 et 1987 respectivement. Maintenant, c'est l'une des formes de jeu les plus joué par des millions de joueurs passionnés. L'un des succès les plus impressionnantes est le jeu World of Warcraft avec plus de 12 millions de joueurs actifs chaque mois (janvier 2011).



[Dragon Quest](#)



[Final Fantasy](#)



[World of Warcraft](#)

OBJECTIFS DE RÉALISATION

Ce projet vous permettra de développer une application présentant plusieurs caractéristiques très intéressantes:

- Une interface usager élaborée
- Une interaction temps réel avec la scène de jeu
- Un simulateur permettant :
 - une gestion temporelle d'évènement impliquant la synchronisation graphique;
 - la mise en place d'une infrastructure orientée objet pertinente;
 - le développement d'algorithme divers;
 - la gestion des données selon l'évolution du jeu.
- Application ludique généralement très appréciée.
- Un défi multidisciplinaire vous permettant de nombreuses possibilités.
- Utiliser votre potentiel créateur et artistique.

N'oubliez pas de définir un projet réaliste respectant les contraintes de temps disponible et vos connaissances technologiques.

CONTRAINTE

Interface usager

À ce niveau, vous pouvez faire à peu près ce que vous voulez mais vous devez au moins respecter ces quelques règles de base :

- offrir une page d'accueil (« *splash screen* »)
- pendant le déroulement de la partie, vous devez offrir :
 - la plus grande surface de jeu possible;
 - une zone de contrôle vous permettant d'interagir avec les options principales de jeu (par exemple, pause et tous les autres contrôles comme les menus).
- vous devez aussi proposer à partir de la page d'accueil une fenêtre expliquant les grandes lignes de votre jeu.

Déroulement du jeu

- Au démarrage du logiciel, vous devez présenter la page d'accueil (« *splash screen* ») offrant le choix d'amorce que vous voulez (par exemple, démarrer immédiatement une partie, aller dans la fenêtre des options, etc.).
- Vous n'avez aucune contrainte au niveau du scénario général. Vous pouvez créer un jeu comme il vous tente : que ce soit à l'âge de pierre ou dans le futur, qu'il soit réaliste ou à saveur fantastique.
- Le jeu que vous devez mettre sur pied doit être basé sur l'attaque de créatures qui arriveront par vague.
 - La progression du jeu doit faire en sorte que les créatures se présentant sont de plus en plus fortes et résistantes.
 - Vous n'avez pas de contrainte sur les modes de déplacement et points de départ des créatures.
 - Néanmoins, les créatures doivent éventuellement se diriger vers votre personnage afin de l'attaquer.
 - Vous devez implémenter au moins 3 sortes de créatures différentes.
- Les améliorations que vous pouvez apporter à votre personnage peuvent être très variés mais vous devez en implémenter au moins 3 à votre guise. Par exemple :
 - Vitesse de déplacement
 - Vitesse de frappe
 - Force de frappe
 - Résistance
 - Pouvoir particulier
 - Armes différentes
 - ...
- Ces améliorations doivent être offertes selon deux évènements précis :
 - Lorsque vous avez terminé une vague d'attaque, un menu vous offre un « upgrade » possible.
 - Lorsque vous tuez un ennemi, il peut laisser tomber un article que vous ramassez.
- L'environnement dans lequel évolue la scène est laissé à votre entière discréction. Cependant, vous devez implémenter au moins un élément qui interagit avec le personnage principal et les

- créatures. Par exemple : le vent, la nuit et le jour, des zones d'eau, des zones marécageuses où les passant sont ralenti.
- Concernant l'aspect multimédia de votre jeu, vous devez inclure :
 - un graphique différent pour chaque type de créature et votre personnage principal;
 - la gestion d'effets sonores n'est pas obligatoire;
 - sans être obligatoire, vous pouvez inclure des petites animations pour les moments particuliers (comme lors des déplacements de votre personnage ou lors de combat entre votre personnage et une créature).
 - Interaction pendant la partie :
 - Vous devez pouvoir attaquer les créatures de façon cohérente avec votre scénario.
 - Vous pouvez connaître les caractéristiques d'une unité simplement en cliquant sur cette dernière (personnage principal et créatures).
 - Vous pouvez modifier les caractéristiques de votre personnage lorsque votre scénario le permet.
 - Vous devez offrir plusieurs contrôle sur le temps pendant une partie:
 - fin de la partie (incluant une confirmation);
 - redémarrer une nouvelle partie (incluant une confirmation);
 - pause et reprise de la partie;
 - lancer immédiatement la prochaine vague de créatures*.
 - Sans être obligatoire, un mode de sauvegarde de l'état actuel d'une partie sera un ajout intéressant.

Approche technologique

- Vous devez obligatoirement profiter de cette occasion pour pousser le concept de la programmation orienté objets pour la création de vos unités. Les notions d'encapsulation, d'héritage et de polymorphisme doivent être mises de l'avant. Ce sera un critère d'évaluation particulièrement important.

ÉVALUATION

• Gestión des évènements de la simulation	40 %
○ Déroulement général du jeu (appréciation du « <i>game play</i> »)	10 %
○ Gestion des options de temps (stop, pause et redémarrage)	5 %
○ Mouvement adéquat du personnage principal	5 %
○ Mouvement adéquat des créatures	5 %
○ Interaction avec l'environnement	5 %
○ Évolution de la partie	5 %
○ Gestion adéquate des améliorations du personnage principal	5 %
• Interface usager et interaction	20 %
○ « <i>Splash screen</i> »	2 %
○ Page explicative du jeu	2 %
○ Aire de jeu bien conçu (ergonomie et qualité visuelle)	10 %
○ Sélection des éléments du jeu et visibilité des paramètres	6 %
• Respect des contraintes générales	25 %
○ Choix judicieux de la structure objet	10 %
○ Respect et bonne gestion des paramètres pour le personnage principal	5 %
○ Respect et bonne gestion des paramètres pour les créatures	5 %
○ Respect des éléments multimédia	5 %
• Autres	15 %
○ Originalité et aspect soigné	5 %
○ Niveau de complexité technique du projet	5 %
○ Aspect soigné du code	5 %

PROJET C – SIMULATEUR D’UN ÉCOSYSTÈME

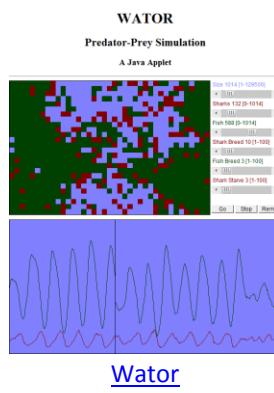
PRÉSENTATION GÉNÉRALE

Ce projet vous propose de créer un simulateur d’écosystème. Ce type de simulateur est parfois appelé un jeu à 0 joueur dont les paramètres initiaux permet de voir différents scénarios d’évolution.

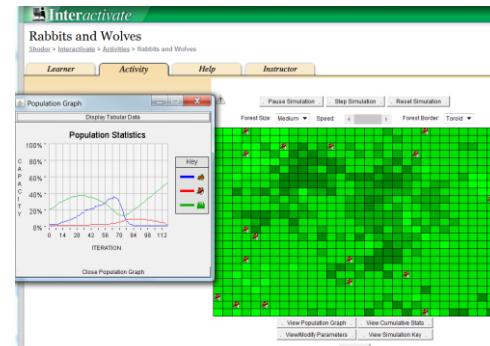
Le concept de ce simulateur consiste à créer une maquette virtuelle d’un environnement réduit et simpliste. Vous devrez créer de petits êtres virtuels ayant des caractéristiques différentes et devant interagir dans l’environnement où ils seront insérés. Principalement, il devra y avoir des proies et des prédateurs interagissant les uns avec les autres. L’équilibre de ce petit écosystème sera important pour que toutes les races puissent vivre ensemble.

La particularité de ce type de simulateur est de pouvoir observer quels sont les impacts de la simulation lorsque vous changez les paramètres initiaux.

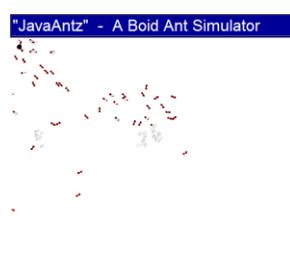
Il existe beaucoup de simulateurs de ce type dont certains présentent d’excellente description technique. En voilà quelques uns :



[Wator](#)

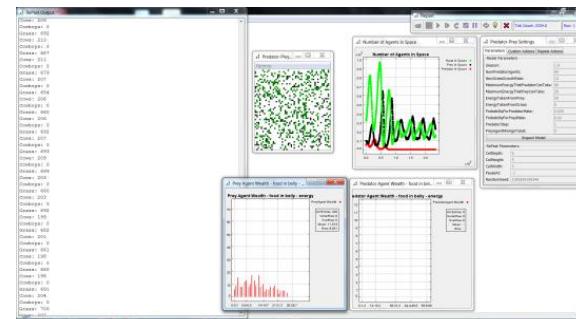


[Interactivate](#)



[JavaAntz](#)

On peut interagir simplement en cliquant sur la surface



[Simulateur proies/prédateurs](#)

Simplement télécharger l’application Java.

OBJECTIFS DE RÉALISATION

Ce projet vous permettra de développer une application présentant plusieurs caractéristiques très intéressantes:

- Une interface graphique présentant une simulation en temps réel.
- La définition d'entités virtuelles devant interagir de façon autonome.
- Un simulateur permettant :
 - une gestion temporelle d'évènement impliquant la synchronisation graphique;
 - mise en place d'une infrastructure orienté objet pertinente;
 - développement d'algorithme divers;
 - gestion des données de l'évolution de la simulation.
- Un défi multidisciplinaire vous permettant de nombreuses possibilités.

N'oubliez pas de définir un projet réaliste respectant les contraintes de temps disponible et vos connaissances technologiques.

CONTRAINTE

Interface usager

À ce niveau, vous pouvez faire à peu près ce que vous voulez mais vous devez au moins respecter ces quelques règles de base :

- offrir une page d'accueil (« *splash screen* »)
- pendant le déroulement de la simulation, vous devez offrir :
 - une vue en temps réel de votre simulation;
 - quelques contrôle de gestion du temps (démarrer/pause/plus rapide/...)
 - au moins un graphique statistique en temps réel du nombre de chaque type d'entité.
- une interface permettant de définir de nouvelles entités et de modifier tous les paramètres des entités virtuels.
- vous devez aussi proposer, à partir de la page d'accueil, une fenêtre expliquant les grandes lignes de votre simulateur.

Définition des entités

- Vous devez définir des entités qui devront interagir entre elles. Ils devront partager plusieurs paramètres parmi ceux-ci :
 - âge maximum pouvant vivre;
 - âge de la maturité sexuelle (à partir de quel âge peut-elle se reproduire);
 - quel est le rythme de reproduction sexuelle;
 - délais maximum entre chaque repas;
 - délais après un repas avant d'avoir faim;
 - vitesse maximum de déplacement;
 - facteur de proximité pour une même race (à quel point ces entités tentent à se regrouper);
 - type de nourriture (carnivore, herbivore, cannibalisme);
 - distance de vue (jusqu'à quelle distance peut voir l'entité, pratique pour se sauver des prédateur et se rapprocher des semblables)
 - ...
- N'oubliez pas que plus vous aurez de paramètres et d'entités, plus votre simulateur sera intéressant mais en même temps plus il sera difficile à équilibrer.
- En plus des entités, vous devez ajouter 1 élément variable de l'environnement comme :

- l'ajout systématique de certaines ressources (comme la pluie ou de la végétation);
 - le cycle du jour et de la nuit;
 - le cycle des saisons;
 - le vent;
 - ...
- Vous devez être capable d'enregistrer et de charger des entités. Dans le même ordre d'idée, vous devez être capable d'enregistrer et de charger des scénarios.
- On vous demande de proposer au moins :
 - un scénario à 2 entités pouvant évoluer plus de 2 minutes;
 - un scénario à 3 entités pouvant évoluer plus de 1 minute.
- Vous allez devoir définir des règles comportementales telles que :
 - quels sont les actions à prioriser selon l'état actuel de l'entité;
 - quels sont les mécanismes de déplacement;
 - quel sera le mécanisme de reproduction;
 - ...
- Même si ça pourrait être très intéressant, aucune interaction pendant la simulation vous est imposé.

Le simulateur

- Pour chaque itération de la simulation, chaque entité est appelée et détermine son état pour l'itération suivante.
- Vous devez être capable de contrôler le déroulement du temps de votre simulateur directement sur l'interface usager pendant la simulation par des contrôle faisant :
 - Démarrer/Pause/Redémarrer/Stop
 - Vitesse 1x, 2x et 3x.
- L'aspect graphique du simulateur est laissé à votre entière discréction. Cependant, faites attention de ne pas trop y investir de temps.

Approche technologique

- Vous devez obligatoirement mettre de l'avant le concept de la programmation orienté objets pour la création de vos unités. Les notions d'encapsulation, d'héritage et de polymorphisme doivent être mis de l'avant. Ce sera un critère d'évaluation particulièrement important.

ÉVALUATION

• Interface usager	40 %
○ « <i>Splash screen</i> »	2 %
○ Page explicative du jeu	2 %
○ Fenêtre de définition et de modification des entités	5 %
○ Enregistrement et chargement d'entités	2 %
○ Zone graphique du simulateur	12 %
○ Contrôles de déroulement temporel	5 %
○ Graphique statistique	10 %
○ Enregistrement et chargement de scénarios	2 %
• Respect des contraintes générales	45 %
○ Choix judicieux de la structure objet	10 %
○ Définition adéquate des paramètres des entités	10 %
○ Définition adéquate du paramètre variable de l'environnement	5 %
○ Définition adéquate des règles comportementales	10 %
○ Robustesse du simulateur	10 %
• Autres	15 %
○ Originalité et aspect soigné	5 %
○ Niveau de complexité technique du projet	5 %
○ Aspect soigné du code	5 %

PROJET D – GESTION DE FEUILLES DE TEMPS

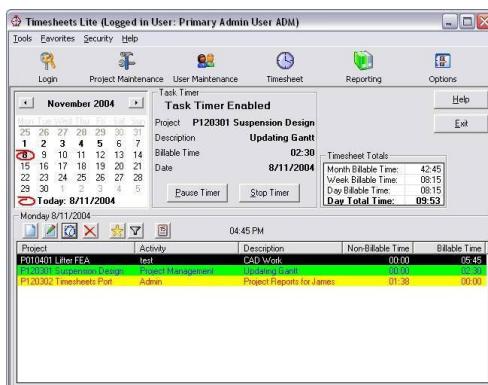
PRÉSENTATION GÉNÉRALE

Ce projet vous propose de créer une application de gestion de données typique des entreprises. Vous aurez à concevoir et créer entièrement une application de gestion des feuilles de temps.

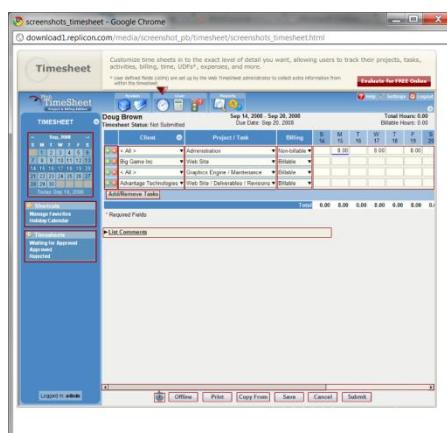
Toutes les entreprises d'une certaine taille utilisent certains outils logiciels pour comptabiliser leur temps. La culture de travail de certaines entreprises ainsi que leurs besoins particuliers fait en sorte qu'il est fréquent de voir le développement d'outils de gestion personnalisés. Ce projet vous donnera l'occasion de développer un outil simple et adapté à sa clientèle cible : une entreprise de développement de projets en ingénierie.

La particularité principale de votre système est qu'il devra faire les liens entre les activités des employés, les tâches typiques de l'entreprise et les projets en cours.

Il existe sur le marché plusieurs produits présentant plusieurs solutions intéressantes et performantes tels que :



Logiciel gratuit ayant une grande popularité
[TimeSheets Lite](#)



Produit commercial basé sur le Web
[Web TimeSheet](#)



Plateforme « open source », performante
et base sur le Web
[Timesheet.php](#)

Avant d'amorcer votre travail, il serait pertinent de faire un survol de quelques applications existantes afin de vous inspirer des approches les plus intéressantes.

OBJECTIFS DE RÉALISATION

Ce projet vous permettra de développer une application présentant plusieurs caractéristiques très intéressantes:

- Une interface usager adaptée au besoin défini.
- Une méthode de gestion des données (insertion, suppression et édition).
- Des outils de recherche d'informations diverses.
- Des outils de présentation des résultats obtenus.

Ce projet requiert l'utilisation d'une base de données quelconque.

CONTRAINTE

Données à manipuler

Vous aurez au moins quatre familles d'information à gérer :

- Les employés de l'entreprise. Voici les champs obligatoires à inclure :
 - Nom
 - Prénom
 - Coordonnées personnelles (à votre choix)
 - Date d'embauche
 - Salaire annuel
 - Titre
- Les tâches communes. Ce sont des tâches génériques qui s'appliquent à l'ensemble des projets et auxquelles vous ferez références ultérieurement. Cette liste de tâche sera définie et ne sera pas appelé à changer souvent. En fait, même si vous devez offrir des outils de gestion tels que l'insertion, la suppression et l'édition, il est probable que l'usager n'ait jamais à la changer. Voici quelques tâches suggérées :
 - Absence maladie
 - Absence familiale
 - Rencontre client
 - Rédaction de devis
 - Réunion de projet
 - Réunion technique
 - Gestion de projet
 - Recherche technique
 - Conception
 - Développement
 - Tests des développements
 - Tests de fonctionnalité
 - Déverminage
 - Documentation
 - Suivre une formation
 - Donner une formation
 - Installation
 - Autres
 - ...
- Les projets en cours. Voici les champs que vous devez inclure :
 - Code interne du projet
 - Nom du projet
 - Description technique du projet
 - L'employé gestionnaire principal du projet
- Toutes les entrées de temps passé par chaque employé. C'est ici que vous faites la gestion des feuilles de temps. Vous devez faire la saisie de toutes les informations pertinentes pour chaque entrée des temps saisis. Vous devez considérer que chaque entrée correspond à la réalisation d'une tâche. Voici les champs obligatoires à saisir :
 - L'employé.
 - La date et le nombre d'heures passés sur une tâche.
 - Le type de tâche fait
 - Le projet sur lequel a tâche a été réalisé
 - Une description de la tâche

Toutes les informations présentées précédemment sont les éléments de base obligatoires du projet. Vous pouvez ajouter à ces éléments tout autre attribut ou élément de gestion que vous désirez.

La gestion de toutes ces informations doit être facile pour l'utilisateur final de votre logiciel. C'est pourquoi vous devrez offrir un mode de gestion simple qui permettra :

- insérer un enregistrement
- supprimer un enregistrement
- modifier un enregistrement

Pour les éléments 1 à 3, vous pouvez faire des interfaces usager très simples et plutôt fonctionnel. En fait, ce sont des éléments qui ne changeront pas souvent et dont la gestion pourrait être gérée par un utilisateur unique.

Par contre, la gestion de l'élément 4 (la gestion du temps) est l'élément central de votre application. C'est elle que chacun utilisera à tous les jours. C'est pourquoi vous devrez offrir un mode de gestion plus élaboré. Tous les logiciels de gestion du temps sont confrontés à un élément commun : les employés ont de la difficulté à les utiliser car ils y perçoivent une perte de temps. C'est pourquoi vous devrez offrir différents mode de saisie efficace de l'information. Voici les modes de saisie obligatoire :

- Mode d'insertion détaillée. Ici l'usager doit insérer tous les éléments requis un à la fois.
- Mode d'insertion en vrac. Ici l'usager peut entrer une même tâche pour plusieurs entrée de temps différent. Par exemple, l'usager pourrait vouloir saisir le fait qu'il a fait du développement logiciel sur le même projet toute la semaine. Il entre donc les informations pertinentes et peut déterminer toutes les périodes de temps désirées.
- Mode d'insertion par temps compté. Ce mode permet à l'usager de ne pas se soucier de compter le temps. Il utilise le logiciel pour entrer un « début de tâche ». Il fait sa tâche et revient sur le logiciel pour identifier la fin de sa tâche. Il saisie les informations techniques de sa tâche et le temps est calculé automatiquement.

Base de données

Vous pouvez utiliser le moteur de la base de données que vous désiré. Aussi, vous devez réaliser le maximum de traitement par la base de données. Par exemple, lorsque vous faites la recherche d'information pour un rapport, vous devez tenter de la faire par le SGBD.

Rapport à présenter

Un outil de gestion semblable présente très peu d'intérêt s'il ne donne pas accès à des rapports synthèses sur les informations saisies.

Vous allez devoir offrir à l'utilisateur plusieurs types de rapports. Voici les rapports obligatoires à réaliser :

1. La liste des activités faites par un employé pour un intervalle de temps donné. À la fin, inclure le nombre total d'heures travaillées.
2. La liste des employés ayant travaillé sur un projet spécifique. Pour chacun d'eux, la sommes des heures investies sur le projet.
3. La liste des employés n'ayant pas fait 40 heures pour une semaine spécifique. Vous devez inclure le temps manquant pour chaque employé identifié. Par exemple, si un employé a indiqué seulement 32 heures, on devrait voir apparaître les 8 heures manquantes. Cet outil est pratique pour identifier ceux qui n'ont pas terminé de saisir leur feuille de temps.
4. Le nombre d'heures totales fait par un employé sur un projet.
5. Au moins 2 autres recherches que vous trouverez pertinentes.

les Interfaces usager

À ce niveau, vous pouvez faire à peu près ce que vous voulez mais vous devez penser au fait que les parties gestion du temps et des rapports sont importantes. Offrir des interfaces usager efficaces et d'apparence soignée est la clé du succès d'une telle application.

Approche technologique

- Dépendamment de l'approche technologique que vous préconiserais, l'utilisation d'une approche par objet est plus ou moins pertinente. Malgré tout, vous devez la considérer lorsque possible.
- La création de vos tables ainsi que les requêtes que vous ferez doivent être méticuleuses et minimiser la duplication des données.

ÉVALUATION

• Définition adéquate des données	10 %
○ Structure des tables adéquates (aucune redondance inutile)	2.5 %
○ Respect des champs demandés	2.5 %
○ Définition pertinentes des types de données	2.5 %
○ Intégrité des données en tout temps	2.5 %
• Gestion adéquate des données	30 %
○ Permet de faire l'insertion, la suppression et l'édition des employés	4 %
○ Permet de faire l'insertion, la suppression et l'édition des tâches	4 %
○ Permet de faire l'insertion, la suppression et l'édition des projets	4 %
○ Permet de faire la suppression et l'édition des champs de gestion du temps	4 %
○ Permet de faire l'insertion des entrées de gestion du temps par :	
■ Mode d'insertion détaillée	4 %
■ Mode d'insertion en vrac	5 %
■ Mode d'insertion par temps compté	5 %
• Rapports	45 %
○ Qualité des rapports - les critères sont :	
■ interface usagers pour la saisie des paramètres d'entrées	
■ bonne utilisation des requêtes	
■ justesse des résultats	
■ interface usagers pour la présentation des résultats	
○ Rapport 1	7.5 %
○ Rapport 2	7.5 %
○ Rapport 3	7.5 %
○ Rapport 4	7.5 %
○ Rapport 5	7.5 %
○ Rapport 6	7.5 %
• Interface usager	5 %
○ Présentation générale pertinente (visibilité des données et champs de saisie)	5 %
• Autres	10 %
○ Originalité de l'interface usager et aspect soigné	5 %
○ Aspect soigné du code	5 %

PROJET E – TRAITEMENT D’IMAGE

PRÉSENTATION GÉNÉRALE

Ce projet vous propose de créer une application vous permettant d’appliquer certains filtres sur des images.

La notion de filtre ici est générique et les opérations que vous allez devoir implémenter seront de nature très différentes du point de vue du traitement d’image :

- filtres linéaire et non linéaire par pixel;
- filtres spatial de type statistique;
- filtres spatial de type convolution;
- manipulations d’image.

Il existe une très grande panoplie de logiciels offrant des outils de manipulation d’images tels que [PhotoShop](#), [Gimp](#) ou [ImageMagick](#). Votre logiciel vous permettra de comprendre certains algorithmes fréquemment utilisés et surtout d’appliquer certaines particularités intéressantes. Aussi, vous devez savoir que la manipulation d’image ne sert pas seulement dans le monde du multimédia mais aussi dans le monde industriel avec les différentes applications de vision par ordinateur.

Outre l’application de filtre sur des images, vous devrez créer un outil de gestion des filtres. Cet outil de gestion vous permettra de définir des filtres personnalisés à partir des filtres génériques. De plus, vous pourrez créer des filtres qui seront une combinaison de filtres spécifiques. Cette gestion vous permettra de créer une bibliothèque de filtres prédéfinis puissants.

De plus, un mode de traitement en « *batch* » vous permettra d’automatiser certaines opérations.

OBJECTIFS DE RÉALISATION

Ce projet vous permettra de développer une application présentant plusieurs caractéristiques très intéressantes:

- Une interface usager graphique vous permettant de voir immédiatement les résultats de vos filtres.
- Une introduction à la manipulation d’image.
- Le développement d’algorithmes professionnels.
- Faire la gestion d’une bibliothèque dynamique de filtres.

Prenez note que l’aspect théorique du traitement d’image ne sera pas abordé dans ce projet mais plutôt une courte présentation des notions de base incluant une vulgarisation des algorithmes à développer.

N’oubliez pas de définir un projet réaliste respectant les contraintes de temps disponible et vos connaissances technologiques.

PRÉCISIONS

Pour ce projet, nous faisons référence à trois types de filtre qui correspondent en fait à leur niveau de spécialisation. Plus précisément :

- Les **filtres génériques** : ce sont les filtres sans définition préalable des paramètres. Par exemple, on pourrait vouloir rendre plus sombre et utiliser un filtre en particulier. Par contre, ce sera à l'usager de définir exactement les paramètres d'assombrissement désiré.
- Les **filtres spécifiques** : ce sont les filtres dont les paramètres sont déjà définis. Par exemple, on veut assombrir une image à 25 % et il existe déjà un filtre dont les paramètres sont définis réalisant cette tâche.
- Les **super filtres** : ce sont des filtres pour lesquels on applique plusieurs filtres de base en cascade. Par exemple, vous pourrez créer un super filtre faisant de rehaussement, suivi d'un rognage et finalement d'un rendu plus flou. Aussi, pour les super filtres, chaque paramètre doit être défini.

Pour ce projet, on considérera toutes les images d'entrée comme étant des images couleurs de type RGB (qui veux dire **Red/Green/Blue** – Rouge/Verte/Bleu).

CONTRAINTE

Interface usager et corps du programme

À ce niveau, vous pouvez faire à peu près ce que vous voulez mais vous devez offrir trois fenêtres différentes qui auront des rôles bien spécifiques :

- Gestion et définition de filtre. Vous devrez offrir les mécanismes nécessaires afin de :
 - définir les nouveaux filtres :
 - pour les filtres spécifiques, vous devez offrir le filtre générique sur lequel s'appuie le filtre spécifique et tous les paramètres nécessaires;
 - pour les super filtres, vous devez offrir une séquence de filtres génériques et tous les paramètres de chaque filtre générique.
 - modifier les filtres existants (filtres spécifiques et super filtres);
 - supprimer les filtres obsolètes.
- Traitement d'une image :
 - charger une image et l'afficher;
 - choisir un filtre (si c'est un filtre générique, on doit pouvoir préciser les valeurs de chaque paramètre);
 - lancer le traitement d'image et afficher l'image après traitement;
 - vous devez offrir le choix d'enregistrer l'image après le traitement.
- Traitement de plusieurs images en « *batch* » :
 - sélectionner un répertoire qui contient toutes les images à traiter;
 - choisir un filtre (si c'est un filtre générique, on doit pouvoir préciser les valeurs de chaque paramètre);
 - déterminer la règle de création des noms de fichiers de sortie;
 - lancer le traitement et pour chaque image traitées faire :
 - la mise à jour d'une barre de progression à l'écran;
 - l'enregistrer sur le disque avec le nouveau nom de fichier.

Filtres génériques à programmer

Cette section détail les différents filtres à programmer. Certaines explications seront plutôt intuitives alors que d'autres présenteront une définition mathématique plus formelle.

Voici le sens des variables utilisés :

p	un pixel quelconque de l'image d'entrée
$p_{x,y}$	le pixel à la position (x, y) de l'image d'entrée
$(R, G \text{ et } B)$ ou $(p_1, p_2 \text{ et } p_3)$	les trois composantes du pixel p (généralement de 0 à 255)
p'	un pixel quelconque de l'image de sortie
$p'_{x,y}$	le pixel à la position (x, y) de l'image de sortie
$(R', G' \text{ et } B')$ ou $(p'_1, p'_2 \text{ et } p'_3)$	les trois composantes du pixel p' (généralement de 0 à 255)
$\alpha_1, \alpha_2, \alpha_3 \dots \alpha_n$	les n paramètres du filtre

TRANSFORMATIONS LINÉAIRES

Ces transformations consistent à calculer, pour chaque pixel, une somme pondérée des trois composantes couleurs (**RGB**).

$$\begin{aligned} R' &= \alpha_1 + \alpha_2 R + \alpha_3 G + \alpha_4 B \\ G' &= \alpha_5 + \alpha_6 R + \alpha_7 G + \alpha_8 B \\ B' &= \alpha_9 + \alpha_{10} R + \alpha_{11} G + \alpha_{12} B \end{aligned}$$

Attention, vous devez vous assurer que la valeur de sortie soit limitée à l'intervalle [0, 255].

Aussi, vous remarquerez que si les paramètres de chaque bande de couleur sont les mêmes, vous effectuerez une transformation de votre image en niveaux de gris (image monochrome).

Ce qui est intéressant avec cette forme générique, c'est qu'il est possible de créer simplement plusieurs type de filtres spécifiques performants et surtout visuellement très intéressants.

Description	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}	α_{11}	α_{12}
Filtres monochromatiques												
- Négatif monochrome	255.0	-0.333	-0.333	-0.333	255.0	-0.333	-0.333	-0.333	255.0	-0.333	-0.333	-0.333
- Extraction du rouge	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
- Extraction du vert	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
- Extraction du bleu	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
- Extraction du jaune	0.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0
- Extraction du cyan	0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0
- Extraction du magenta	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0
- Moyenne arithmétique	0.0	0.333	0.333	0.333	0.0	0.333	0.333	0.333	0.0	0.333	0.333	0.333
- Moyenne anthropomorphique	0.0	0.30	0.59	0.11	0.0	0.30	0.59	0.11	0.0	0.30	0.59	0.11
- Moyenne avec luminosité	0.0	0.21	0.71	0.08	0.0	0.21	0.71	0.08	0.0	0.21	0.71	0.08
- Rehaussement absolu 12.5 %	32.0	0.333	0.333	0.333	32.0	0.333	0.333	0.333	32.0	0.333	0.333	0.333
- Rehaussement absolu 25.0 %	64.0	0.333	0.333	0.333	64.0	0.333	0.333	0.333	64.0	0.333	0.333	0.333
- Assombrissement absolu 12.5 %	-32.0	0.333	0.333	0.333	-32.0	0.333	0.333	0.333	-32.0	0.333	0.333	0.333
- Assombrissement absolu 25.0 %	-64.0	0.333	0.333	0.333	-64.0	0.333	0.333	0.333	-64.0	0.333	0.333	0.333
- Rehaussement relatif 12.5 %	0.0	0.375	0.375	0.375	0.0	0.375	0.375	0.375	0.0	0.375	0.375	0.375
- Rehaussement relatif 25.0 %	0.0	0.416	0.416	0.416	0.0	0.416	0.416	0.416	0.0	0.416	0.416	0.416
- Assombrissement relatif 12.5 %	0.0	0.291	0.291	0.291	0.0	0.291	0.291	0.291	0.0	0.291	0.291	0.291
- Assombrissement relatif 25.0 %	0.0	0.250	0.250	0.250	0.0	0.250	0.250	0.250	0.0	0.250	0.250	0.250
- ...												
Filtres couleurs												
- Négatif couleur	255.0	-1.0	0.0	0.0	255.0	0.0	-1.0	0.0	255.0	0.0	0.0	-1.0
- Rehaussement de 25 %	0.0	1.25	0.0	0.0	0.0	0.0	1.25	0.0	0.0	0.0	0.0	1.25
- Rehaussement du rouge de 25 %	0.0	1.25	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
- Rehaussement du vert de 25 %	0.0	1.0	0.0	0.0	0.0	0.0	1.25	0.0	0.0	0.0	0.0	1.0
- Rehaussement du bleu de 25 %	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.25
- Rehaussement du jaune de 25 %	0.0	1.25	0.0	0.0	0.0	0.0	1.25	0.0	0.0	0.0	0.0	1.0
- Rehaussement du cyan de 25 %	0.0	1.0	0.0	0.0	0.0	0.0	1.25	0.0	0.0	0.0	0.0	1.25
- Assombrissement de 25 %	0.0	1.25	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.75
- Assombrissement du rouge de 25 %	0.0	0.75	0.0	0.0	0.0	0.0	0.75	0.0	0.0	0.0	0.0	1.0
- Assombrissement du vert de 25 %	0.0	0.75	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
- Assombrissement du bleu de 25 %	0.0	0.75	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.75
- Assombrissement du jaune de 25 %	0.0	0.75	0.0	0.0	0.0	0.0	0.75	0.0	0.0	0.0	0.0	1.0
- Assombrissement du cyan de 25 %	0.0	0.75	0.0	0.0	0.0	0.0	0.75	0.0	0.0	0.0	0.0	0.75
- Assombrissement du magenta de 25 %	0.0	0.75	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.75
- Rehaussement : pigment de type Sepia	0.0	0.393	0.769	0.189	0.0	0.349	0.686	0.168	0.0	0.272	0.534	0.131
- ...												

TRANSFORMATIONS NON LINÉAIRES DE TYPE GAMMA

Il existe plusieurs façons d'appliquer une correction gamma. L'équation qui vous est suggérée ici correspond à une technique générique qui vous donnera pleine flexibilité. Encore une fois, vous devrez vous assurer que la sortie reste dans l'intervalle [0, 255] car selon les paramètres entrés, le résultat peut diverger rapidement.

$$p_i' = \alpha_1 + 255 \cdot \alpha_2 \cdot \left(\frac{p_i}{255} \right)^{\alpha_3}$$

L'interprétation des paramètres est relativement simple et vous permet plusieurs opérations intéressantes. Par contre, vous devez vous assurer que les valeurs saisies respectent les intervalles données.

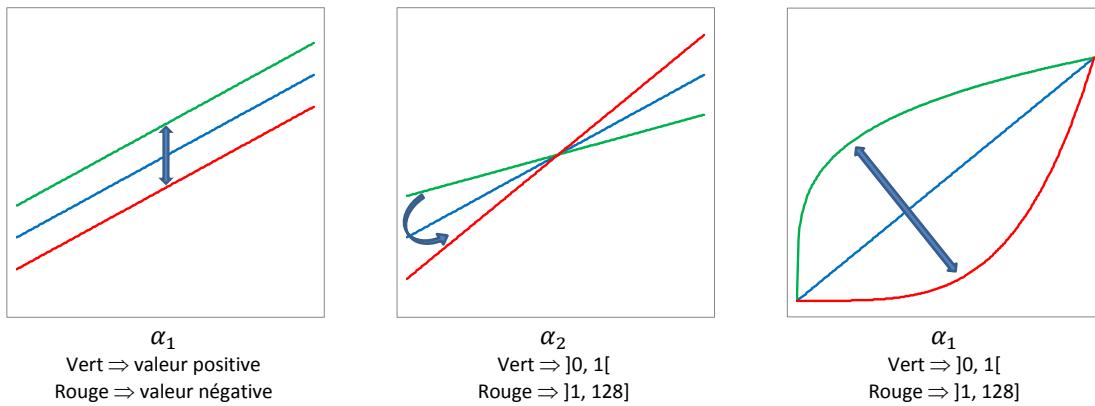
α_1 facteur de décalage de l'intensité : [-128, 128]

α_2 facteur de croissance linéaire de l'intensité : [-128, 0[\cap]0, 128]

α_3

facteur de croissance exponentielle de l'intensité :]0, 128]

En regardant les graphiques suivants, vous verrez comment s'applique l'effet de chaque paramètre. Ces graphiques montrent, sur les abscisses, l'intensité initiale d'un pixel et, sur les ordonnées, l'intensité de sortie d'un pixel. Donc, les courbes correspondent à la réponse du filtre. La courbe bleue est la courbe de référence.



Voici quelques filtres spécifiques que vous pourriez créer :

Description	α_1	α_2	α_3
- Correction gamma de rehaussement niveau 1	0.0	1.0	0.75
- Correction gamma de rehaussement niveau 2	0.0	1.0	0.50
- Correction gamma de rehaussement niveau 3	0.0	1.0	0.25
- Correction gamma d'assombrissement niveau 1	0.0	1.0	1.50
- Correction gamma d'assombrissement niveau 2	0.0	1.0	2.00
- Correction gamma d'assombrissement niveau 3	0.0	1.0	4.00
- Négatif de l'image (une autre façon de le faire)	255.0	-1.0	1.0
- Correction gamma de rehaussement niveau 1 avec négatif	255.0	-1.0	0.75
- Correction gamma de rehaussement niveau 2 avec négatif	255.0	-1.0	0.50
- Correction gamma de rehaussement niveau 3 avec négatif	255.0	-1.0	0.25
- Correction gamma d'assombrissement niveau 1 avec négatif	255.0	-1.0	1.50
- Correction gamma d'assombrissement niveau 2 avec négatif	255.0	-1.0	2.00
- Correction gamma d'assombrissement niveau 3 avec négatif	255.0	-1.0	4.00
- ...			

SEGMENTATION

Les algorithmes de segmentation consistent à produire une image binaire (en noir et blanc) à partir d'une image monochrome ou couleur. On doit établir un seuil à partir duquel il est possible d'établir la classe d'appartenance d'un pixel (pixel noir ou pixel blanc). Pour le projet, le seul paramètre que vous aurez à déterminer sera ce fameux seuil (1 paramètre pour une image monochrome et 3 paramètres pour une image couleur).

La pertinence des résultats obtenus restent souvent tributaire du seuil de segmentation déterminé. C'est pourquoi il existe plusieurs méthodes permettant de faire une détection automatisée du seuil en fonction de certains critères de l'image. Évidemment, la réalisation d'une méthode automatisée n'est pas demandée pour le projet.

SEGMENTATION MONOCHROME

$$p' = R' = G' = B' = \begin{cases} 0 & \text{si } p \leq \alpha_1 \\ 255 & \text{si } p > \alpha_1 \end{cases}$$

Le paramètre α_1 doit être inclus dans l'intervalle [0, 255].

Le résultat d'une segmentation

SEGMENTATION COULEUR

$$\begin{aligned} R' &= \begin{cases} 0 & \text{si } R \leq \alpha_1 \\ 255 & \text{si } R > \alpha_1 \end{cases} \\ G' &= \begin{cases} 0 & \text{si } G \leq \alpha_2 \\ 255 & \text{si } G > \alpha_2 \end{cases} \\ B' &= \begin{cases} 0 & \text{si } B \leq \alpha_3 \\ 255 & \text{si } B > \alpha_3 \end{cases} \end{aligned}$$

Les trois paramètres doivent être inclus dans l'intervalle [0, 255].

FILTRES SPATIAL

Ces filtres sont basés sur l'analyse des pixels voisins afin d'estimer la valeur du pixel central. Le secret ici est la façon de faire ce traitement car il peut être très long.

Tous ces filtres utilisent le concept de masque qui permet de déterminer la zone d'intérêt. Même s'il est possible de faire des masques complexes, on se contentera ici par des masques carrés dont la dimension pourra varier entre 3 x 3 pixels jusqu'à 13 x 13 pixels. Nous limiterons aussi à des masques centrés, ce qui impliquera que la dimension des masques devra être de taille impaire.

L'application des masques sera discutée selon le type de filtre. Néanmoins, ces types de filtre présentent une problématique commune, la gestion des effets de bords. En effet, ces algorithmes nécessitent l'utilisation des pixels voisins pour établir la nouvelle valeur du pixel d'intérêt. Mais qu'arrive-t-il aux pixels qui se trouvent sur les bords, ceux qui ne peuvent réellement être traités comme les autres puisqu'ils n'ont pas tous les voisins requis. En fait, il existe une multitude d'approche pour palier à ce problème dont ceux-ci :

1. On réduit la taille de l'image de sortie pour qu'elle corresponde à tous les pixels valides.
2. On ne traite pas les pixels sur les bords et on insère dans l'image de sortie directement une copie des pixels de l'image d'entrée.
3. On traite les pixels sur les bords avec des algorithmes plus ou moins élaborés afin d'insérer des valeurs qui auront un maximum de vraisemblance. Certaines méthodes peuvent être plutôt simpliste comme des techniques d'interpolation ou d'extrapolation alors que d'autres peuvent devenir réellement complexes comme des réseaux de neurones ou des méthodes d'analyse statistique sophistiquée.

Pour ce projet, on vous demande de mettre en œuvre la deuxième technique.

FILTRES MIN, MAX, ORDRE N ET MÉDIAN

Cette gamme de filtre est particulièrement exigeante en terme de temps de calcul mais les résultats obtenus en valent vraiment la peine. Même si le concept de masque tel que démontré plus haut peut

s'appliquer, on simplifiera l'algorithme en appliquant uniquement une sélection de pixel autour du pixel d'intérêt (toujours en respectant les contraintes de masque déjà énoncées).

Ainsi, pour chaque pixel, on considère tous les pixels inclus dans le masque de taille défini en les plaçant dans une liste qu'on va finalement trié en ordre croissant de valeur. De cette liste, on va prendre une seule valeur selon le type de filtre choisi :

- filtre minimum : la première valeur
- filtre maximum : la dernière valeur
- filtre d'ordre n : la $n^{\text{ième}}$ valeur (implique l'usage d'un paramètre supplémentaire)
- filtre médian : on prend la valeur se trouvant au centre de la liste (par exemple, pour un filtre 9×9 on aura 81 pixels et on choisira la $41^{\text{ième}}$ valeur d'intensité)

Pour ce filtre, vous aurez à utiliser un paramètre pour déterminer la taille du masque et un paramètre supplémentaire dans le cas du filtre d'ordre N .

CONVOLUTION

La convolution est l'un des outils les plus puissants du traitement d'image. Sans entrer dans les détails théorique, elle permet de créer certains des outils les plus efficaces et pertinents en traitement d'image. Malgré tout, elle demande une quantité de traitement très important.

Formellement, la convolution peut s'exprimer par :

$$p'_{x,y} = \sum_{i=-k}^k \sum_{j=-k}^k \alpha_{i,j} \cdot p_{x+i,y+j}$$

D'un point de vue plus pratique, il suffit de faire la somme pondérée de chaque pixels considérées par le masque à une position spécifique. Ce qui nécessite un masque dont chaque élément possède une valeur numérique.

α_1	α_2	α_3	α_4	α_5
α_6	α_7	α_8	α_9	α_{10}
α_{11}	α_{12}	α_{13}	α_{14}	α_{15}
α_{16}	α_{17}	α_{18}	α_{19}	α_{20}
α_{21}	α_{22}	α_{23}	α_{24}	α_{25}

Donc, un masque de 5×5 nécessite 25 paramètres et un masque de 9×9 pas moins de 81 paramètres.

Voici un exemple vulgarisant le concept de la convolution. Vous avez l'image et le masque suivant :

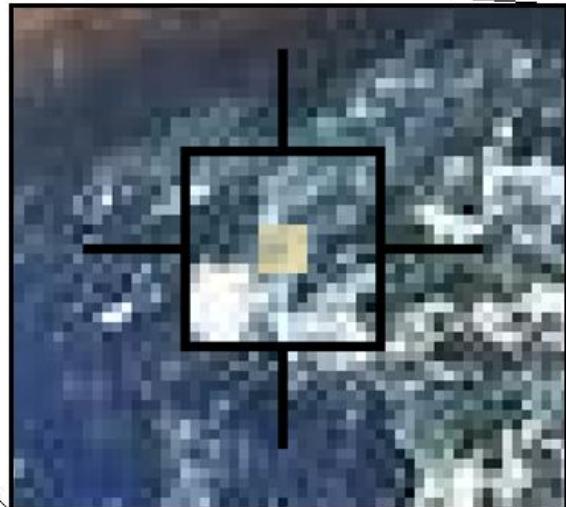


Image d'entrée

0.0037	0.0147	0.0256	0.0147	0.0037
0.0147	0.0586	0.0952	0.0586	0.0147
0.0256	0.0952	0.1502	0.0952	0.0256
0.0147	0.0586	0.0952	0.0586	0.0147
0.0037	0.0147	0.0256	0.0147	0.0037

Masque de 5 x 4

Prenons comme exemple le pixel (110, 110) :



La région mise en surbrillance représente la zone de 5 x 5 pixels centrée sur le pixel à la position (110, 110). En regardant de plus près les valeurs numériques de ces pixels nous avons :

	108	109	110	111	112
108	R = 131 G = 154 B = 168	R = 146 G = 169 B = 183	R = 84 G = 107 B = 121	R = 67 G = 90 B = 106	R = 76 G = 102 B = 119
109	R = 123 G = 146 B = 160	R = 122 G = 145 B = 159	R = 91 G = 114 B = 128	R = 122 G = 145 B = 159	R = 45 G = 71 B = 88
110	R = 85 G = 108 B = 122	R = 76 G = 96 B = 110	R = 67 G = 90 B = 104	R = 91 G = 114 B = 130	R = 111 G = 134 B = 152
111	R = 55 G = 77 B = 90	R = 141 G = 167 B = 180	R = 88 G = 111 B = 125	R = 128 G = 154 B = 169	R = 132 G = 155 B = 173
112	R = 47 G = 64 B = 72	R = 62 G = 84 B = 95	R = 141 G = 167 B = 180	R = 164 G = 192 B = 206	R = 78 G = 104 B = 119

Ayant toutes les informations requises pour le traitement, il est maintenant possible de faire la convolution pour le pixel de la position (110, 110). On calcule pour chaque bande la somme pondérée suivante :

$$R_{110,110}' = \alpha_1 \cdot R_{108,108} + \alpha_2 \cdot R_{109,108} + \alpha_3 \cdot R_{110,108} + \alpha_4 \cdot R_{111,108} + \alpha_5 \cdot R_{112,108} + \alpha_6 \cdot R_{108,109} + \alpha_7 \cdot R_{109,109} + \alpha_8 \cdot R_{110,109} + \alpha_9 \cdot R_{111,109} + \alpha_{10} \cdot R_{112,109} + \alpha_{11} \cdot R_{108,110} + \alpha_{12} \cdot R_{109,110} + \alpha_{13} \cdot R_{110,110} + \alpha_{14} \cdot R_{111,110} + \alpha_{15} \cdot R_{112,110} + \alpha_{16} \cdot R_{108,111} + \alpha_{17} \cdot R_{109,111} + \alpha_{18} \cdot R_{110,111} + \alpha_{19} \cdot R_{111,111} + \alpha_{20} \cdot R_{112,111} + \alpha_{21} \cdot R_{108,112} + \alpha_{22} \cdot R_{109,112} + \alpha_{23} \cdot R_{110,112} + \alpha_{24} \cdot R_{111,112} + \alpha_{25} \cdot R_{112,112}$$

$$G_{110,110}' = \alpha_1 \cdot G_{108,108} + \alpha_2 \cdot G_{109,108} + \alpha_3 \cdot G_{110,108} + \alpha_4 \cdot G_{111,108} + \alpha_5 \cdot G_{112,108} + \alpha_6 \cdot G_{108,109} + \alpha_7 \cdot G_{109,109} + \alpha_8 \cdot G_{110,109} + \alpha_9 \cdot G_{111,109} + \alpha_{10} \cdot G_{112,109} + \alpha_{11} \cdot G_{108,110} + \alpha_{12} \cdot G_{109,110} + \alpha_{13} \cdot G_{110,110} + \alpha_{14} \cdot G_{111,110} + \alpha_{15} \cdot G_{112,110} + \alpha_{16} \cdot G_{108,111} + \alpha_{17} \cdot G_{109,111} + \alpha_{18} \cdot G_{110,111} + \alpha_{19} \cdot G_{111,111} + \alpha_{20} \cdot G_{112,111} + \alpha_{21} \cdot G_{108,112} + \alpha_{22} \cdot G_{109,112} + \alpha_{23} \cdot G_{110,112} + \alpha_{24} \cdot G_{111,112} + \alpha_{25} \cdot G_{112,112}$$

$$B_{110,110}' = \alpha_1 \cdot B_{108,108} + \alpha_2 \cdot B_{109,108} + \alpha_3 \cdot B_{110,108} + \alpha_4 \cdot B_{111,108} + \alpha_5 \cdot B_{112,108} + \alpha_6 \cdot B_{108,109} + \alpha_7 \cdot B_{109,109} + \alpha_8 \cdot B_{110,109} + \alpha_9 \cdot B_{111,109} + \alpha_{10} \cdot B_{112,109} + \alpha_{11} \cdot B_{108,110} + \alpha_{12} \cdot B_{109,110} + \alpha_{13} \cdot B_{110,110} + \alpha_{14} \cdot B_{111,110} + \alpha_{15} \cdot B_{112,110} + \alpha_{16} \cdot B_{108,111} + \alpha_{17} \cdot B_{109,111} + \alpha_{18} \cdot B_{110,111} + \alpha_{19} \cdot B_{111,111} + \alpha_{20} \cdot B_{112,111} + \alpha_{21} \cdot B_{108,112} + \alpha_{22} \cdot B_{109,112} + \alpha_{23} \cdot B_{110,112} + \alpha_{24} \cdot B_{111,112} + \alpha_{25} \cdot B_{112,112}$$

Prenons un exemple concret, le résultat du pixel pour la bande des rouges sera :

$$R_{110,110}' = 0.0037 \cdot 131 + 0.0147 \cdot 146 + 0.0256 \cdot 84 + 0.0147 \cdot 67 + 0.0037 \cdot 76 + 0.0147 \cdot 123 + 0.0586 \cdot 122 + 0.0952 \cdot 91 + 0.0586 \cdot 122 + 0.0147 \cdot 45 + 0.0256 \cdot 85 + 0.0952 \cdot 76 + 0.1502 \cdot 67 + 0.0952 \cdot 91 + 0.0256 \cdot 111 + 0.0147 \cdot 55 + 0.0586 \cdot 141 + 0.0952 \cdot 88 + 0.0586 \cdot 128 + 0.0147 \cdot 132 + 0.0037 \cdot 47 + 0.0147 \cdot 62 + 0.0256 \cdot 141 + 0.0147 \cdot 164 + 0.0037 \cdot 78$$

Il existe beaucoup de théorie sur la convolution et certains aspects sont très intéressants. Sans entrer dans les détails théoriques, voici plusieurs noyaux de convolution que vous pouvez prédéfinir :

Les filtres directionnels Sobel (3 x 3)

Les filtres directionnels simples (3 x 3)

Les filtres moyenneurs gaussiens

Les filtres moyenneurs uniformes

-1	0	1
-2	0	2
-1	0	1

Direction \Rightarrow

1	0	-1
2	0	-2
1	0	-1

Direction \Leftarrow

-2	-1	0
-1	0	1
0	1	2

Direction \triangleleft

2	1	0
1	0	-1
0	-1	-2

Direction \triangleright

-1	-2	-1
0	0	0
1	2	1

Direction \Downarrow

1	2	1
0	0	0
-1	-2	-1

Direction \Updownarrow

0	-1	-2
1	0	-1
2	1	0

Direction $\triangleleft\triangleright$

0	1	2
-1	0	1
-2	-1	0

Direction $\triangleleft\triangleright$

0.111	0.111	0.111
0.111	0.111	0.111
0.111	0.111	0.111

3 x 3

0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04

5 x 5

0.020	0.020	0.020	0.020	0.020	0.020	0.020
0.020	0.020	0.020	0.020	0.020	0.020	0.020
0.020	0.020	0.020	0.020	0.020	0.020	0.020
0.020	0.020	0.020	0.020	0.020	0.020	0.020
0.020	0.020	0.020	0.020	0.020	0.020	0.020
0.020	0.020	0.020	0.020	0.020	0.020	0.020
0.020	0.012	0.020	0.020	0.020	0.020	0.020

7 x 7

0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012
0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012
0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012
0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012
0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012
0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012
0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012
0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012
0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012

9 x 9

0.0037	0.0147	0.0256	0.0147	0.0037
0.0147	0.0586	0.0952	0.0586	0.0147
0.0256	0.0952	0.1502	0.0952	0.0256
0.0147	0.0586	0.0952	0.0586	0.0147
0.0037	0.0147	0.0256	0.0147	0.0037

7 x 7

5 x 5

Direction \Rightarrow

0	0	0
1	-1	0
0	0	0

Direction \triangleleft

1	0	0
0	-1	0
0	0	0

Direction \Downarrow

0	1	0
0	-1	0
0	0	0

Direction \Updownarrow

0	0	1
0	-1	0
1	0	0

Direction $\triangleleft\triangleright$

1	0	-1
2	0	-2
1	0	-1

2	1	0
1	0	-1
0	-1	-2

1	2	1
0	0	0
-1	-2	-1

Direction \Updownarrow

0	1	2
-1	0	1
-2	-1	0

Direction $\triangleleft\triangleright$

<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>-1</td><td>-1</td><td>2</td></tr> <tr><td>0</td><td>0</td><td>0</td></tr> </table>	0	0	0	-1	-1	2	0	0	0	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>-1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>-1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>2</td></tr> </table>	-1	0	0	0	-1	0	0	0	2	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>-1</td><td>0</td></tr> <tr><td>0</td><td>-1</td><td>0</td></tr> <tr><td>0</td><td>2</td><td>0</td></tr> </table>	0	-1	0	0	-1	0	0	2	0	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>0</td><td>-1</td></tr> <tr><td>0</td><td>-1</td><td>0</td></tr> <tr><td>2</td><td>0</td><td>0</td></tr> </table>	0	0	-1	0	-1	0	2	0	0
0	0	0																																					
-1	-1	2																																					
0	0	0																																					
-1	0	0																																					
0	-1	0																																					
0	0	2																																					
0	-1	0																																					
0	-1	0																																					
0	2	0																																					
0	0	-1																																					
0	-1	0																																					
2	0	0																																					
Direction \Rightarrow	Direction \triangleleft	Direction \Downarrow	Direction $\triangleleft\triangleright$																																				
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>2</td><td>-1</td><td>-1</td></tr> <tr><td>0</td><td>0</td><td>0</td></tr> </table>	0	0	0	2	-1	-1	0	0	0	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>2</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>-1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>-1</td></tr> </table>	2	0	0	0	-1	0	0	0	-1	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>2</td><td>0</td></tr> <tr><td>0</td><td>-1</td><td>0</td></tr> <tr><td>0</td><td>-1</td><td>0</td></tr> </table>	0	2	0	0	-1	0	0	-1	0	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>0</td><td>2</td></tr> <tr><td>0</td><td>-1</td><td>0</td></tr> <tr><td>-1</td><td>0</td><td>0</td></tr> </table>	0	0	2	0	-1	0	-1	0	0
0	0	0																																					
2	-1	-1																																					
0	0	0																																					
2	0	0																																					
0	-1	0																																					
0	0	-1																																					
0	2	0																																					
0	-1	0																																					
0	-1	0																																					
0	0	2																																					
0	-1	0																																					
-1	0	0																																					
Direction \Leftarrow	Direction \nwarrow	Direction \Uparrow	Direction \nearrow																																				
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>-4</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> </table>	0	1	0	1	-4	1	0	1	0	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>-1</td><td>-1</td><td>-1</td></tr> <tr><td>-1</td><td>9</td><td>-1</td></tr> <tr><td>-1</td><td>-1</td><td>-1</td></tr> </table>	-1	-1	-1	-1	9	-1	-1	-1	-1	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>-8</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	1	1	1	1	-8	1	1	1	1	Embossage uniforme									
0	1	0																																					
1	-4	1																																					
0	1	0																																					
-1	-1	-1																																					
-1	9	-1																																					
-1	-1	-1																																					
1	1	1																																					
1	-8	1																																					
1	1	1																																					
Laplacien (``edge detection``)	Acuité (``sharpness``)																																						

MANIPULATION D'IMAGE

ROGNAGE (``CROP``)

Cette opération consiste à construire une nouvelle à partir d'une section de l'image d'entrée. Ce filtre est très simple à réaliser et consiste simplement à définir la nouvelle image selon la taille définie et de copier pixel par pixel le contenu de la première image.

Vous aurez besoin de quatre paramètres pour ce filtre :

- α_1 la position du haut
- α_2 la position du bas
- α_3 la position à gauche
- α_4 la position à droite

ROTATION PAR INCRÉMENT DE $\pm 90^\circ$

Ce filtre consiste à appliquer une rotation de l'image par incrément de 90° uniquement. Cette opération est pratique pour mettre dans le bon sens des images prises par un appareil photo.

Vous aurez à créer une image de sortie de la bonne taille et copier les pixels adéquatement un à un. Vous n'aurez qu'un paramètre indiquant le nombre d'incrément de la rotation. Seul les valeurs de 1, 2 & 3 sont permises.

Approche technologique

- Vous devez obligatoirement considérer le concept de la programmation orienté objets pour la création des filtres. Les notions d'encapsulation, d'héritage et de polymorphisme doivent être mis de l'avant. Ce sera un critère d'évaluation particulièrement important.

ÉVALUATION

• Interface de gestion des filtres	15 %
○ Définition de nouveaux filtre spécifiques	5 %
○ Définition de nouveaux super filtres	5 %
○ Modification de filtres existants (filtres spécifiques et super filtres)	4 %
○ Suppression de filtres	1 %
• Interface de traitement d'une seule image	20 %
○ Charger une image et l'afficher adéquatement	2 %
○ Outil de sélection de filtre	2 %
○ Définition des paramètres pour les filtres génériques	6 %
○ Traitement d'image	6 %
○ Affichage de l'image finale	2 %
○ Sauvegarde de l'image finale	2 %
• Interface de traitement d'images en « <i>batch</i> »	20 %
○ Outil de sélection d'un répertoire de départ	2 %
○ Outil de sélection de filtre	2 %
○ Définition des paramètres pour les filtres génériques	6 %
○ Outils permettant de déterminer la règle de création de nom	5 %
○ Traitement en « <i>batch</i> » des images (incluant rétroaction et sauvegarde)	5 %
• Développement des filtres	35 %
○ Transformation linéaire	5 %
○ Transformation non linéaire	5 %
○ Segmentation monochrome	2 %
○ Segmentation couleur	2 %
○ Filtre spatial min, max, d'ordre N et médian	8 %
○ Convolution	10 %
○ Rognage	1 %
○ Rotation par incrément de 90°	2 %
• Autres	10 %
○ Originalité de l'interface usager et aspect soigné	5 %
○ Aspect soigné du code	5 %

PROJET F – VISION ARTIFICIELLE

PRÉSENTATION GÉNÉRALE

Ce projet vous propose de créer une application de vision artificielle en passant par toutes les étapes typiques d'un tel projet (excepté l'acquisition des images).

Ainsi, vous devrez résoudre un vrai problème de vision dans un contexte de contrôle de qualité pour la production d'un produit spécifique. Ce projet vous permettra d'aborder plusieurs notions intéressantes telles que :

- manipulation d'images :
 - chargement en mémoire;
 - affichage;
- manipulation des images pixel par pixel pour réaliser divers traitements :
 - filtrage;
 - uniformisation d'éclairage;
 - segmentation couleur;
- développement d'autres algorithmes spécifiques;
 - extraction de caractéristiques;
 - classification;

La vision par ordinateur est un domaine complexe qui touche plusieurs disciplines à la fois. Elle met en jeu autant les phénomènes physiques d'acquisition d'images, que les défis d'implantation logicielle ainsi que des problèmes mathématiques de résolution de problèmes (notamment les problématiques de segmentation d'images et de classification). Sans voir la théorie de façon formelle, vous aborderez un cas simple avec des outils de base. À la fin du projet, vous comprendrez que même ce projet « simple » présente des difficultés importantes et peu devenir rapidement un défi de taille.

En somme, ce projet vous permettra de réaliser une application complète et abordant presque la totalité de la chaîne de traitement d'une application de vision artificielle.

OBJECTIFS DE RÉALISATION

Ce projet vous permettra de développer une application présentant plusieurs caractéristiques très intéressantes:

- Une interface usager graphique vous permettant de voir immédiatement les résultats de vos algorithme de traitement.
- Une introduction à la manipulation d'image.
- Le développement d'algorithmes professionnels.
- La résolution d'un problème relativement difficile.

Prenez note que l'aspect théorique du traitement d'image et de la vision ne sera pas abordé dans ce projet mais plutôt une courte présentation des notions de base incluant une vulgarisation des algorithmes à développer.

N'oubliez pas de définir un projet réaliste respectant les contraintes de temps disponible et vos connaissances technologiques.

LOGICIEL À RÉALISER

Vous devez mettre sur pied une application permettant la résolution du problème présenté. La partie la plus significative de ce projet est définitivement la partie algorithmique qui permettra de faire l'analyse des images. Malgré tout, vous devez monter une application présentant certaines caractéristiques spécifiques :

- Interface usager :
 - Avoir un menu offrant certaines commandes :
 - Ouverture d'une image (qui sera l'image analysée)
 - Sauvegarde de l'image affichée
 - Démarrage d'une analyse
 - Une zone d'affichage de l'image courante. Cette zone d'affichage doit être en mesure de montrer toutes les images intermédiaires du processus d'analyse. Vous devez offrir le choix de l'image à afficher :
 - Image d'entrée
 - Image après l'étape 1 – Filtration
 - Image après l'étape 2 – Uniformisation
 - Image après l'étape 3 – Segmentation (selon un choix de couleur spécifique)
 - Image après l'étape 4 – Remplissage (selon un choix de couleur spécifique)
 - Image après l'étape 5 – Descripteur de forme (attention, cette option doit afficher un tableau de données avec les résultats des calculs et non une image)
 - Image après l'étape 6 – Image finale avec annotation
 - De plus, la zone d'affichage de l'image doit afficher, selon la position de la souris, la position relative en pixel sur l'image et la valeur du pixel (les valeurs *RGB* pour les images couleurs ou simplement la valeur d'intensité pour les images monochromatiques).
- Pendant tout le processus d'analyse d'image, vous devez afficher une barre de progression et l'étape courante.
- Lors du démarrage du processus d'analyse, vous devez demander quel type d'analyse doit être fait (selon le type d'image d'entrée).

MISE EN CONTEXTE

Vous devez développer un logiciel qui fera le contrôle de qualité d'une production de friandise. Vous aurez à votre disposition des images de différentes produits et vous devrez, pour chaque image :

- Compter le nombre de friandises de chaque catégorie et de chaque couleur.
- Identifier les friandises présentant des défauts de fabrication.

Vous avez à votre disposition plusieurs images de 5 produits différents :

- Smarties
- M&M
- M&M Almond

- Reeses Pieces
- Whoppers

Vous avez ensuite plusieurs images de mélange qui sont faites.

- Mix01 – Smarties + M&M
- Mix02 – Smarties + M&M + Reeses Pieces
- Mix03 – Smarties + M&M + Reeses Pieces + M&M Almond
- Mix04 – Smarties + M&M + Reeses Pieces + M&M Almond + Whoppers

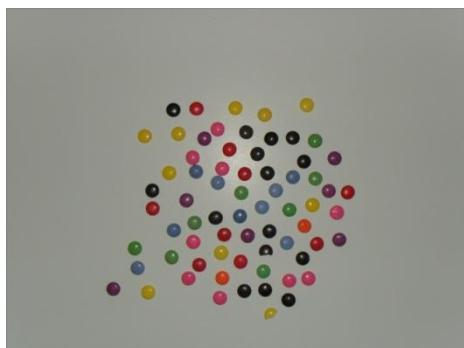
Chacune des images est classée selon l'une des trois catégories suivantes :

- Level1 – Images plus faciles à résoudre – Aucun des produits ne se touchent.
- Level2 – Images moyennement difficiles à résoudre – Uniquement les produits de couleurs différentes peuvent se toucher.
- Level3 – Images plus difficiles à résoudre – Tous les produits peuvent se toucher, peut importe leur couleur.

Vous avez aussi 5 images de référence de la scène de fond.

Le but du projet n'est pas de résoudre toutes les images mises à votre disposition. Par contre, il est attendu que vous soyez capable de résoudre les images sans mélange du niveau 1 pour au moins deux familles de produit. Le niveau 2, tant qu'à lui est un peu plus difficile et devrait vous être accessible sans de gros effort. Finalement, les images du niveau 3 présentent des difficultés beaucoup plus importantes. Si le défi vous intéresse, ces images vous montreront comment un problème en apparence simple devient rapidement complexe.

Voici quelques images :



Smarties – Level 1



M&M Almond – Level 2



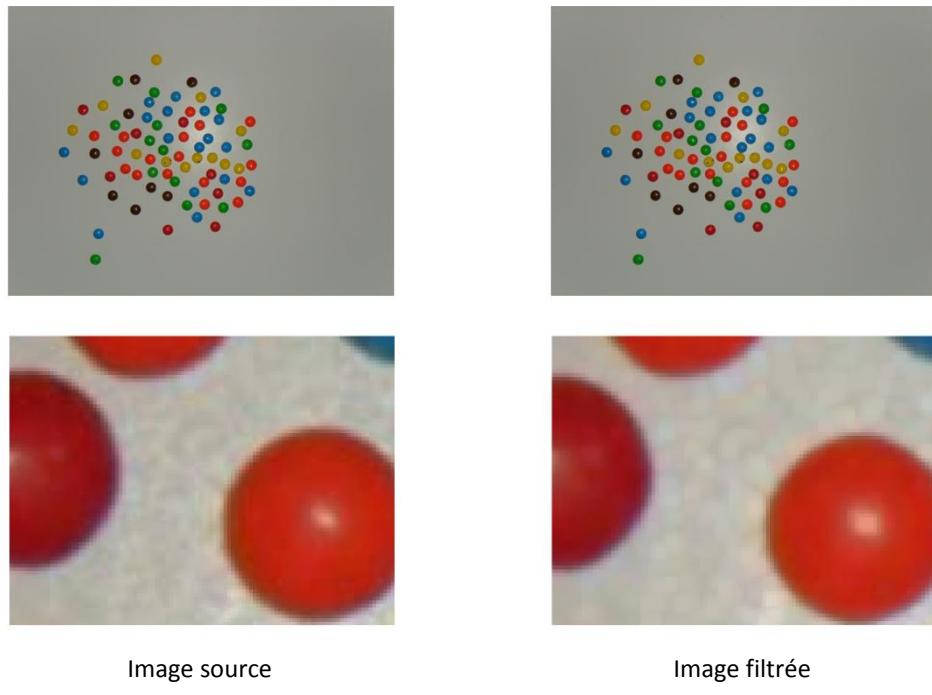
Mix03 – Level 3

APPROCHE ALGORITHMIQUE PROPOSÉE

La résolution d'un tel problème peut se faire de plusieurs façons différentes. On vous propose ici une approche très simple utilisant des notions de base du traitement d'image et de la vision artificielle. Vous trouverez en annexe l'explication détaillée des différents algorithmes nécessaires à la résolution du problème.

Cette solution ne permet pas de solutionner tous les cas présentés. Cependant, elle vous donne un excellent point de départ pour réaliser ce projet. Voici une description sommaire de chacune des 6 étapes :

1. **Filtrage de l'image d'entrée** – Généralement, les images acquises présentent toujours différentes formes de bruit. On voudra souvent réduire ces bruits en appliquant divers filtres qui seront sélectionnés selon les problèmes d'acquisition présents. Les images qui vous sont présentées ne font pas exception et présentent plusieurs problématiques d'acquisition. Nous proposons ici d'utiliser deux filtres en cascade. D'abord, un filtre médian 3×3 et ensuite, une convolution avec un noyau de dimension 3×3 et une distribution gaussienne.



2. **Uniformisation de l'éclairage** – Lors de l'acquisition des images, il est très difficile d'obtenir une répartition uniforme de l'intensité d'éclairage sur toute la surface d'observation. Les images que vous avez présentent d'ailleurs une forte intensité facilement observable au centre. Cette forte intensité s'estompe rapidement vers les extrémités. L'interprétation de telle images reste possible mais plus difficile pour les opérations de segmentation et de classification. On vous propose ici de réaliser une correction de cette variation d'éclairage. Encore une fois, plusieurs façon permettent d'obtenir un résultat intéressant, on vous propose ici une méthode facile qui donne généralement de bons résultats. Après cette opération, vous pourrez observer que l'image est plus uniforme en termes d'éclairage et que les couleurs sont plus uniformes sur toute l'image. En se référant à l'annexe d'uniformisation d'éclairage, voici les paramètres suggérés pour chaque étape :

Étape 1 – Filtre de convolution avec un noyau de taille 9×9 . La distribution utilisée était de type gaussienne mais une distribution uniforme serait tout aussi efficace.

Étape 1b – Toutes les images de référence ont été utilisées.

Étape 2 – Filtre maximum avec un noyau de taille 9 x 9.

Étape 3 – Filtre de convolution avec un noyau de taille 151 x 151. Encore une fois, la distribution utilisée était de type gaussienne mais une distribution uniforme serait toujours tout aussi efficace.

Étape 4 – Uniformisation sans paramètres.

Étape 5 – Une normalisation donnant une plage [0, 1] a été utilisée. Ici la plage de sortie n'a aucune incidence réel en autant que vous restez conséquent avec le type de données.

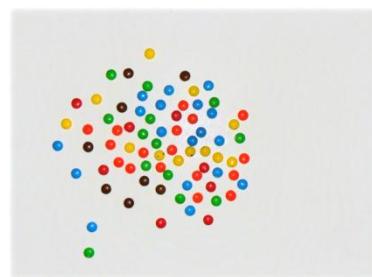
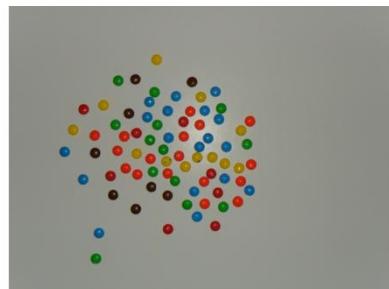


Image filtrée

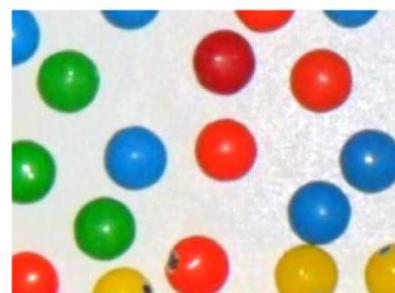
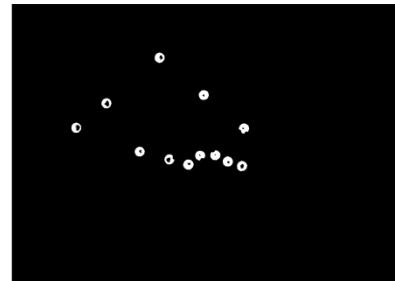
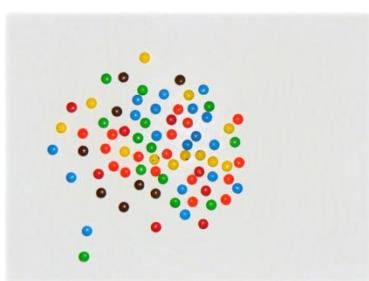


Image uniformisée

3. **Segmentation des couleurs** – La segmentation d'une image consiste à identifier quelle partie de l'image est intéressante et quelle autre partie ne l'est pas. Cette opération reste l'une des plus complexes dans le domaine de la vision par ordinateur et présente parfois des défis déroutant étant donné leur apparente simplicité. Nous proposons ici une approche fondamentale qui consiste simplement à identifier directement sur l'image les pixels dont les valeurs d'intensité rouge-vert-bleu sont dans des intervalles donnés. L'idée consiste à segmenter autant d'images différentes que de produits différents à identifier. Ainsi, une image pour les produits rouges, une image pour les produits verts, une image pour les produits bleus et ainsi de suite.



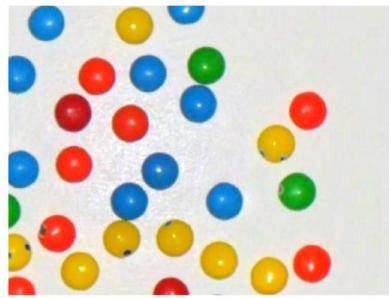


Image uniformisée

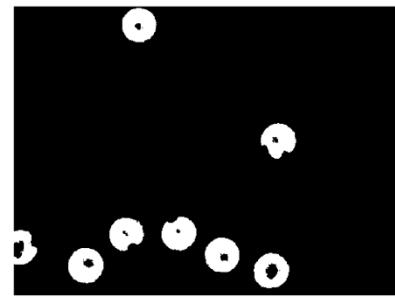


Image segmentée

4. **Remplissage des zones internes** – Afin d'identifier toutes les zones intéressantes de l'image, nous proposons ici de remplir tous les trous présents dans l'image. Cette opération est relativement simple et peut permettre certaines simplifications ultérieures. Vous trouverez dans l'annexe associé une description détaillée de l'algorithme. On vous propose de réaliser une implémentation considérant un voisinage de 4 pixels.

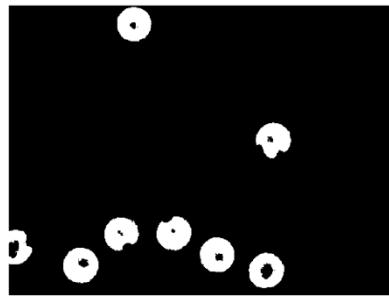
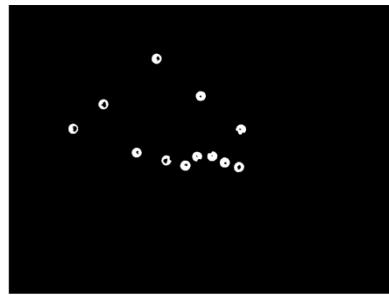


Image binaire source

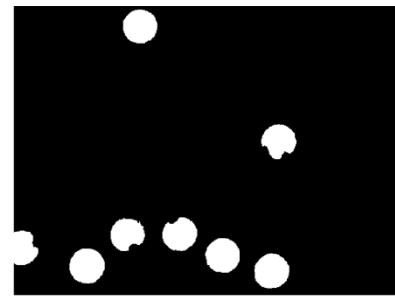
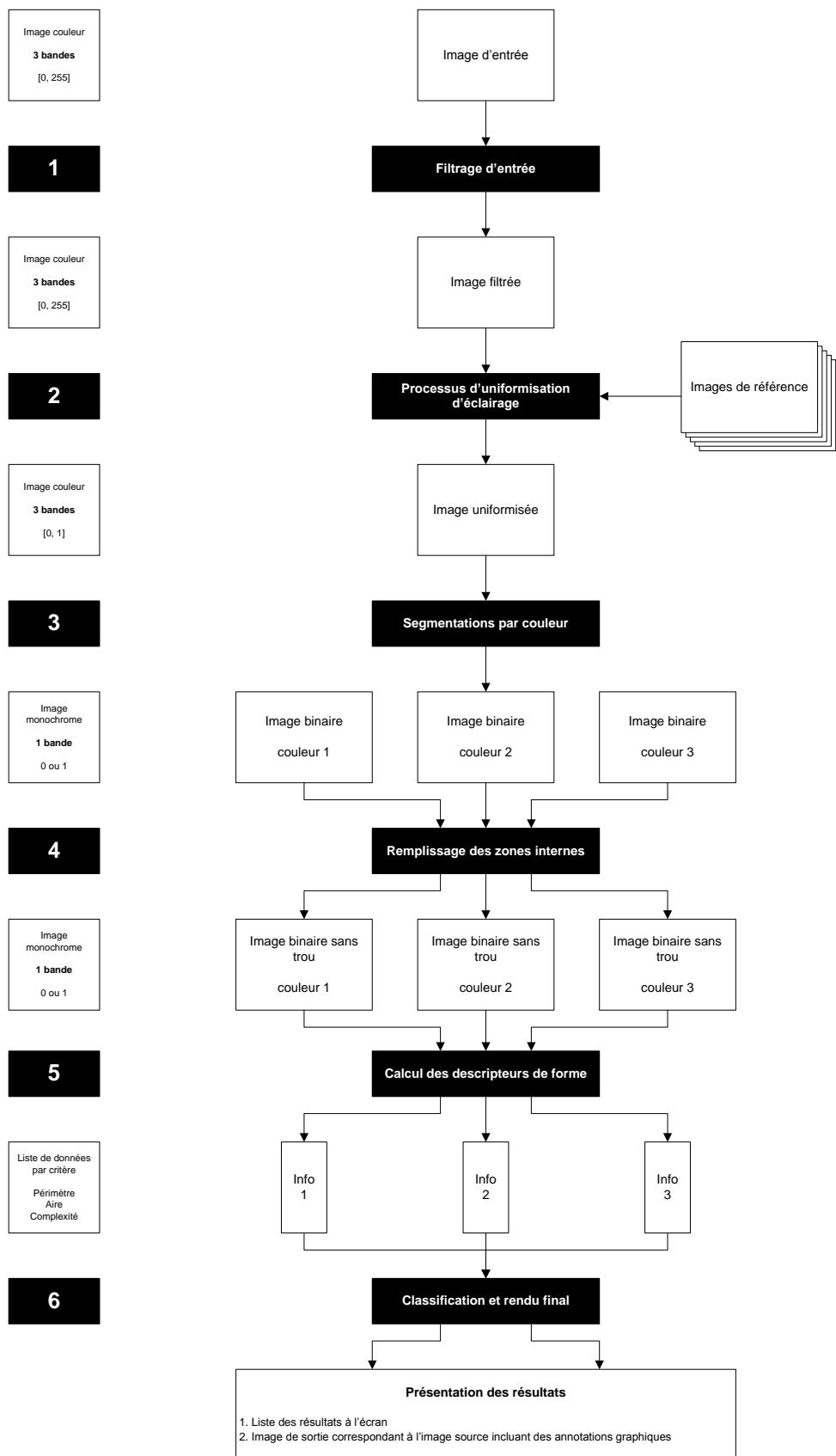


Image dont les trous sont remplis

5. **Calcul des descripteurs de formes** – Nous pouvons considérer chaque région contiguë d'une image binaire comme étant un objet à part entière. La présente étape consiste à identifier chacune de ces régions et de calculer certaines caractéristiques pour chacune d'entre elle. Pour commencer, il est suggéré de calculer le périmètre, l'aire et la complexité de chaque objet.
6. **Classification et rendu final** – À partir des caractéristiques précédemment calculées, il sera possible d'identifier les objets pertinents et finalement déterminer lesquels correspondent à certaines classes (la classe des bons produits ou celle des produits défectueux). Finalement, il sera important de produire deux formes de résultat : un tableau avec les produits identifiés et une image finale sur laquelle sera surimposée des étiquettes d'identification.



On remarque que chaque friandise est reconnue et encerclée par un contour vert pour les bons produits et rouge pour les produits défectueux. Cette façon d'annoter l'image n'est qu'une suggestion.



Approche technologique

- Vous devez obligatoirement considérer le concept de la programmation orienté objets pour la création de votre solution. Les notions d'encapsulation, d'héritage et de polymorphisme doivent être mis de l'avant. Ce sera un critère d'évaluation particulièrement important.
- Vous devez apporter au moins une amélioration au processus d'analyse qui vous est proposé. Si vous manquez d'idée, n'hésitez pas à consulter l'enseignant pour avoir des suggestions.

ÉVALUATION

• Interface usager	35 %
○ Ouverture et sauvegarde d'une image	4 %
○ Affichage de l'image	4 %
○ Sélection du type d'image à afficher	10 %
○ Affichage de la valeur du pixel courant selon la position de la souris	6 %
○ La présence d'une barre de progression pendant l'analyse	6 %
○ Demande du type d'analyse	5 %
• Développement algorithmique	55 %
○ Aspect modulaire de chaque filtre (incluant la paramétrisation)	5 %
○ Filtre médian	6 %
○ Filtre de convolution	8 %
○ Uniformisation d'éclairage	8 %
○ Segmentation couleur	2 %
○ Remplissage des zones internes	5 %
○ Extraction des objets et descripteurs de formes	12 %
○ Classification	2 %
○ Superposition d'annotations sur l'image final	2 %
○ Ajout personnel par rapport à l'énoncé	5 %
• Autres	10 %
○ Originalité de l'interface usager et aspect soigné	5 %
○ Aspect soigné du code	5 %

PROJET G – SIMULATEUR D’UN RÉSEAU INFORMATIQUE

PRÉSENTATION GÉNÉRALE

Ce projet vous propose de créer une application illustrant le comportement d'un réseau informatique pour quelques usages simples. L'objectif de cette application est uniquement pédagogique.

Il est maintenant très fréquent de trouver différentes applications informatiques vulgarisant et illustrant divers concepts de toute sorte. Ces applications sont de plus en plus axées vers des plateformes multimédias interactives et permettent ainsi d'expliquer concrètement des concepts parfois abstraits et souvent difficiles à se représenter. Ces outils deviennent ainsi des outils pédagogiques importants et de plus en plus sollicités.

Le tableau suivant présente quelques exemples de ce genre :

...

OBJECTIFS DERÉALISATION

Ce projet vous permettra de développer une application présentant plusieurs caractéristiques très intéressantes:

- Une interface usager graphique avancée vous permettant :
 - de définir la configuration d'un réseau;
 - de faire l'appel de commandes spécifiques;
 - de voir, à l'aide de pictogramme dynamique, une simulation du transfert d'information dans tout le réseau.
- La gestion d'une structure de données avancées.
- Le développement d'algorithmes divers.
- D'utiliser abondamment les notions de programmation orientée objet.

N'oubliez pas de définir un projet réaliste respectant les contraintes de temps disponible et vos connaissances technologiques.

LOGICIEL À RÉALISER

Vous devez mettre sur pied une application permettant la résolution du problème présenté. Ce projet requiert une bonne coordination car deux parties sont à produire :

- un environnement permettant de définir la structure d'un réseau;
- un outil de simulation représentant la transmission des données suite à l'appel d'une commande

Voici les éléments les plus importants à réaliser :

- Ce projet requiert une interface usager assez élaborée étant donné son aspect pédagogique. Elle doit être agréable à regarder et simple d'utilisation.
- La partie « *Définition de la structure d'un réseau* » doit inclure :
 - L'ajout et la suppression des constituants du réseau (ordinateur, router, etc.).
 - La définition des paramètres réseau pour chacun de ses constituants (il doit être possible de définir et de modifier ces paramètres).
 - Chaque constituant du réseau est représenté par un pictogramme clair de sa classe et la position sur la surface de simulation doit être défini par l'usager.
 - L'ajout et la suppression des liens entre chacun des constituants. Ces liens doivent rester visibles en tout temps.
 - Vous devez pouvoir enregistrer l'état courant du réseau et charger en mémoire une configuration préalablement sauvegardée.
- La partie « *Simulation* » doit inclure :
 - La visualisation de la scène déjà définie.
 - Un panneau de contrôle offrant plusieurs contrôles qui permettent :
 - de pousser une commande dans le réseau à partir d'un point de départ;
 - de mettre en pause et de reprendre la simulation;
 - d'arrêter la simulation;
 - de déterminer la vitesse de simulation (x_1, x_2, x_3 , etc.).
 - Pendant la simulation, il doit être possible de voir le flux des données sur le réseau par le déplacement progressif de pictogrammes illustrant les paquets qui sont échangés.
 - Une légende des différents pictogrammes doit aussi être visible.

CONTRAINTE

Les outils permettant de définir le réseau pour la simulation doit offrir les constituants suivants :

- Ordinateurs de types serveur
- Ordinateurs
- Routeurs
- Hub
- Switch

Les éléments de configuration à considérer pour chaque constituant sont les suivants (selon si elles s'appliquent ou non) :

- Nom du constituant (par défaut, un nom à valeur incrémentale tel que *Ordinateur1*, *Ordinateur2*, etc.)
- Adresse IP
- Masque
- Passerelle

Aussi, les commandes devant être implémentées sont :

- ping

- le transfert d'une trame fictive entre deux applications (par exemple, *Skype* sur l'ordinateur 1 envoi un paquet à *Skype* sur l'ordinateur 2);
- vous devez ajouter au moins une autre commande de votre choix.

La simulation que vous présentez doit être sans faute et tenir compte de la variabilité des configurations possibles.

Approche technologique

- Vous devez obligatoirement considérer le concept de la programmation orienté objets pour la création de votre solution. Les notions d'encapsulation, d'héritage et de polymorphisme doivent être mis de l'avant. Ce sera un critère d'évaluation particulièrement important.

ÉVALUATION

• Module de définition d'un réseau	40 %
○ Capacité d'introduire de nouveaux composants principaux	
● (serveurs, ordinateurs, routeurs, etc.)	4 %
○ Capacité d'introduire un lien entre deux composants	4 %
○ Capacité de retirer des composants sur la surface de travail	2 %
○ Capacité de retirer des liens sur la surface de travail	2 %
○ Capacité de définir les paramètres des composants principaux	10 %
○ Représentation adéquate des composants par des pictogrammes pertinents	4 %
○ Représentation adéquate des liens entre les différents composants	2 %
○ Capacité d'enregistrer la configuration courante du réseau	4 %
○ Capacité d'ouvrir la configuration présente dans un fichier	
● préalablement sauvégarde	4 %
○ Présence des cinq composants de base	4 %
• Module de simulation	40 %
○ Affichage du réseau	2 %
○ Présence du panneau de contrôle et de ses contrôles :	
■ Introduire une commande dans le réseau	4 %
■ Mettre en pause et reprendre la simulation	2 %
■ Arrêter la simulation	2 %
■ Déterminer la vitesse de simulation	2 %
○ Affichage clair du flux réseautique par des pictogrammes adéquats	
● pendant la simulation	10 %
○ Présence de la légende pour les différents symboles utilisés	2 %
○ Commande à simuler :	
■ ping	4 %
■ transfert de trame	4 %
■ commande personnelle	4 %
■ exactitude du comportement pour chaque commande	4 %
• Autres	20 %
○ Structure orientée objet pertinente	10 %
○ Originalité de l'interface usager et aspect soigné	5 %
○ Aspect soigné du code	5 %

ANNEXE 1 : UML - DIAGRAMME DE CLASSES

QU'EST-CE QUE UML

L'acronyme UML vient de « *Unified Modeling Language* » qui veut dire langage de modélisation unifié. Ce langage a été créé afin de normaliser les nombreux outils de conception de projet existants auparavant. Il consiste à créer des graphiques à base de pictogrammes illustrant différents éléments conceptuels d'un système. Initialement développé par des chercheurs du domaine du génie logiciel et destiné aux éléments de conception orientée objet, cette méthode de conception est maintenant utilisée par plusieurs branches du génie (électrique, mécanique, industriel, etc).

La dernière révision d'UML (datant de mai 2010) propose 13 types de diagrammes permettant d'élaborer la conception d'un projet. Ces 13 diagrammes sont dépendants hiérarchiquement et se complètent mutuellement de façon à permettre la modélisation d'un projet tout au long de son cycle de vie (que ce soit de sa conception, sa réalisation, son implémentation, son déploiement et même sa maintenance). À titre de référence, voici la liste des 13 diagrammes :

- Les 6 diagrammes structurels (statique)
 - Diagramme de classes
 - Diagramme d'objets
 - Diagramme de composants
 - Diagramme de déploiement
 - Diagramme des paquetages
 - Diagramme de structure composite
- Les 3 diagrammes comportementaux
 - Diagramme des cas d'utilisation
 - Diagramme état-transition
 - Diagramme d'activité
- Les 4 diagrammes d'interaction (dynamique)
 - Diagramme de séquence
 - Diagramme de communication
 - Diagramme global d'interaction
 - Diagramme de temps

Il est important de comprendre que l'utilisation du langage UML n'est pas une méthode. Ce qui implique que son utilisation est laissé au besoin de son utilisateur. Par exemple, plusieurs méthodologies utilisent partiellement le langage. D'ailleurs, il est courant de retrouver en industrie seulement les diagrammes les plus utilisés : le diagramme des cas d'utilisation et le diagramme de classes notamment.

Dans le cadre du projet, vous n'aurez qu'à fournir le diagramme de classes.

DIAGRAMME DE CLASSES

Le diagramme de classes est un schéma utilisé pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci. Une classe, comme vous le savez certainement, décrit les responsabilités, les attributs et le comportement d'un ensemble d'objet de même type. De cette

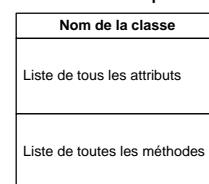
façon, elles permettent de modéliser un programme pour ainsi découper une tâche complexe en plusieurs sous-tâches plus simples.

Les classes peuvent être liées entre elles grâce au mécanisme d'héritage qui permet de mettre en évidence les relations de parenté. Plusieurs autres types de relations sont possibles entre des classes. Chacune de ces relations est représentée par un pictogramme spécifique dans le diagramme de classes.

Dans le cadre du cours, nous n'aborderons qu'une partie de ce qu'offre le diagramme de classes. Nous mettrons l'emphase sur trois types de relations simplifiées : l'héritage, l'association et l'agrégation.

La classe

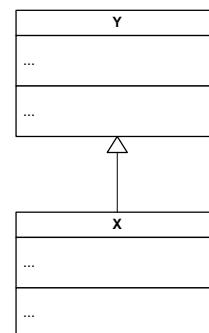
La classe est l'élément de base et est représentée par un pictogramme rectangulaire divisé en trois parties distinctes. La première partie possède le nom de la classe, la deuxième partie possède la liste de tous les attributs (incluant les types) et la troisième partie possède la liste de toutes les méthodes (incluant la signature détaillée de la fonction). Habituellement, on ajoute un code pour indiquer si chaque attribut et méthode est privé, protégé ou public.



L'héritage

C'est le principe de division par généralisation et spécialisation. D'ailleurs, parfois on nomme le concept d'héritage comme étant le concept de généralisation ou de spécialisation dépendamment du point de vue. Néanmoins, ce type de relation implique toujours une classe parent et une classe enfant pour laquelle, la classe enfant hérite des attributs et méthodes (publics et protégés) de la classe parent.

Le pictogramme pour indiquer cette relation est la flèche vide pointant vers le parent. De cette façon, le pictogramme se lit ainsi, la classe enfant X hérite de la classe parent Y.

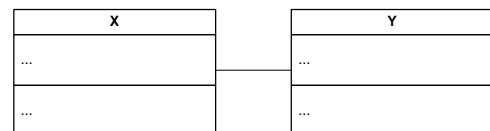


L'association

Ce type de relation implique une connexion sémantique entre deux classes. Par exemple, une première classe peut modifier le contenu d'une deuxième classe simplement en passant cette dernière comme paramètre dans une de ses méthodes. Un autre cas, une méthode de la première classe retourne une nouvelle instance d'une deuxième classe.

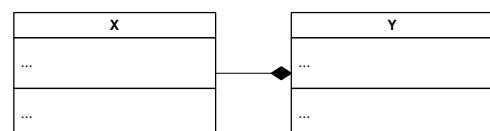
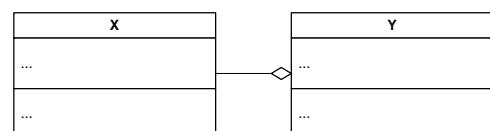
Dans tous ces cas, il y a association entre ces deux classes.

Mais ni l'une ni l'autre va obligatoirement utiliser ou nécessiter l'autre pour exister. Le pictogramme utilisé ici est simplement une ligne.



L'agrégation et la composition

C'est une relation de subordination entre deux classes. On dit qu'une première classe regroupe d'autres classes ou que l'instance de la première classe utilise une instance d'une deuxième classe. On peut spécialiser la notion d'agrégation lorsque le cycle de vie de la deuxième classe dépend de la première classe. On parle alors de composition.

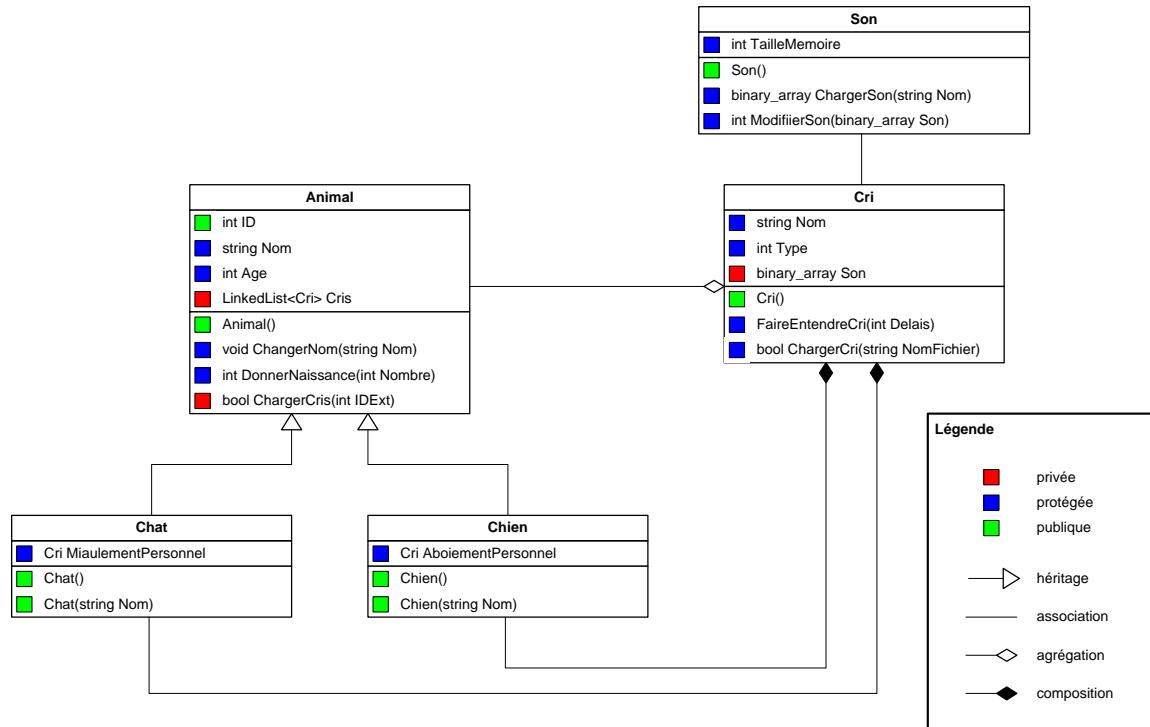


Le pictogramme utilisé est une ligne terminé par un losange vide (pour l'agrégation) ou plein (pour la composition).

EXEMPLE COMPLET

Cet exemple montre une petite conception incomplète de classe reliée à une application faisant l'usage de différents animaux. Cet exemple présente 5 classes distinctes toutes reliées d'une façon ou d'une autre.

Voici le diagramme de classes associé :



Les classes *Chat* et *Chien* hérite de la classe *Animal* de façon évidente.

La composition entre la classe *Cri* et les classes *Chat* et *Chien* vient du fait que chacune des classes possesseurs (*Chat* et *Chien*) possèdent une variable de type *Cri* (*MiaulementPersonnel* et *AboiementPersonnel*). On remarque que linstanciation d'un objet *Cri* se fera en même temps que linstanciation de la classe possesseur. Ainsi, la classe *Cri* « naîtra » et « mourra » en même temps que la classe *Chat* par exemple.

Lagrégation entre la classe *Animal* et *Cri* représente le fait que la classe possesseur (*Animal*) peut posséder une ou plusieurs instances de la classe *Cri*. Lorsque la classe *Animal* est instanciée, il est possible quaucune classe *Cri* ne soit instanciée. Il est même possible que les objets de la classe *Cri* incluses dans la liste de *Animal* ne soient pas libérées lorsque linstance de la classe *Animal* soit libérée.

Lassociation entre la classe *Cri* et *Son* vient du fait que la classe *Cri* utilise dans une de ses fonctions (*ChargerSon*) une classe externe pour faire le travail. Cestàdire que la classe *Son* sera utilisée par la

classe *Cri* pour faire une tâche spécifique. Néanmoins, aucune instance de la classe *Son* n'est possédée par la classe *Cri*.

Voici maintenant un exemple de code illustrant ces propos :

```
public class Animal
{
    public int ID;
    protected String Nom;
    protected int Age;
    private LinkedList<Cri> Cris = new LinkedList<Cri>();           // Agrégation

    public Animal() {};
    public void ChangerNom(String Nom) { this.Nom = Nom };
    protected int DonnerNaissance(int Nombre) { ... }
    private boolean ChargerCris(int IDExt) { ... }
}

public class Cri
{
    protected String Nom;
    protected int Type;
    private binary_array Son;

    public Cri() {};
    protected void FaireEntendreCri(int Delais) { ... };
    protected boolean ChargerCri(String NomFichier) { ... }           // Association
}

public class Son
{
    protected int TailleMemoire;

    public Son() {};
    protected binary_array ChargerSon(String Nom) { ... };
    protected int ModifierSon(binary_array Son) { ... }
}

public class Chat extends Animal                                // Héritage
{
    protected Cri MiaulementPersonnel;                         // Composition

    public Chat() {};
    public Chat(String Nom) { ... };
}

public class Chien extends Animal                                // Héritage
{
    protected Cri AboiementPersonnel;                          // Composition

    public Chat() {};
    public Chat(String Nom) { ... };
}
```

ANNEXE 2 : ESPACES DE COULEUR

LA COULEUR

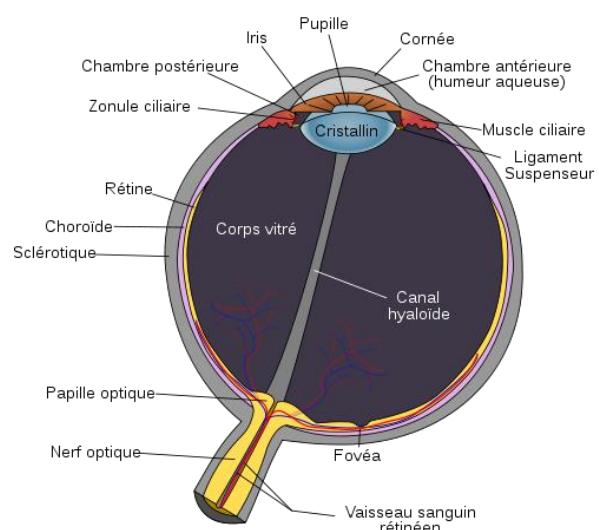
Comprendre comment est formée la couleur n'est pas une mince tâche. On peut considérer plusieurs aspects différents afin de caractériser la couleur tels que :

- La nature physique du phénomène lumineux incluant les caractéristiques de :
 - la source lumineuse;
 - le milieu de propagation;
 - le corps réfléchissant;
 - le(s) récepteur(s).
- L'interprétation psychologique que chaque individu fait du signal reçu.
- le contexte d'application dans lequel s'inscrit cette interprétation :
 - l'affichage sur un écran;
 - l'impression d'un document;
 - la quantification de la couleur;
 - l'estimation de la différence entre deux couleurs;
 - qui sont les individus destinés à cette interprétation :
 - humain dit normal;
 - humain présentant quelques caractéristiques particulières telles que le daltonisme;
 - autres être vivants.

Pour chacun des aspects nommé il existe plusieurs aspects théoriques et pratiques qui font varier la perception de la couleur, par exemple, l'œil humain possède plusieurs caractéristiques intéressantes. En plus des nombreuses composantes anatomiques permettant de produire une image nette sur la rétine, il possède deux familles de capteurs répartis sur la rétine : les bâtonnets et les cônes.

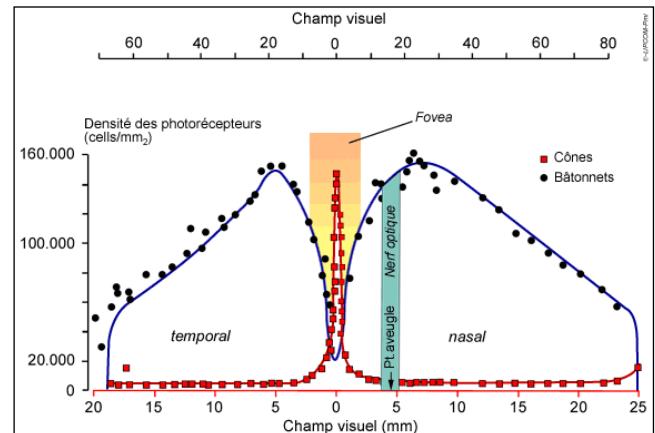
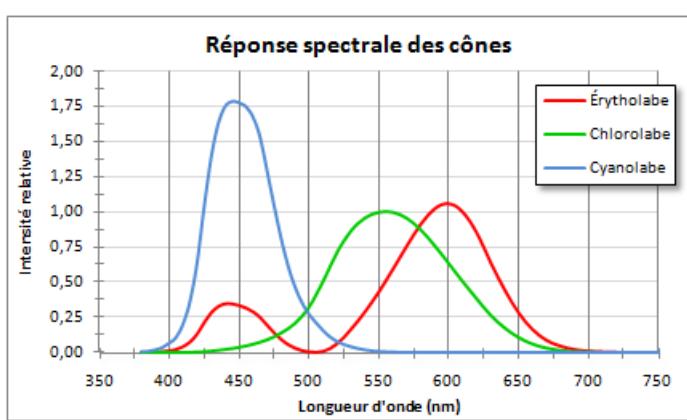
Les bâtonnets, au nombre approximatif de 120 millions par œil, sont spécialisés dans la perception de l'intensité de la lumière. Ils sont 10 000 fois plus sensibles que les cônes et sont répartis plutôt sur la périphérie de l'œil. Ces cellules ont une réponse en fréquence plus rapide que les autres à raison de 75 à 80 Hz et sont très sensible à l'interprétation du mouvement. Néanmoins, ils sont sensibles seulement de 510 à 555 nm.

Les cônes, au nombre de cinq à sept millions sont spécialisés dans la perception de la fréquence de la lumière et par conséquent l'interprétation de la couleur. Ils sont concentrés vers l'intérieur du champ de vision de la rétine. Leur persistance est d'environ 40 Hz. Il en existe trois types :



Type	Nom	Sensibilité	Couleur
L	Érytholabe	460 à 700 nm	Rouge
M	Chlorolabe	530 à 546 nm	Vert
S	Cyanolabe	424 à 436 nm	Bleu

Les graphiques suivant, montrent la réponse spectrale des différents cônes (à gauche) et la répartition de la densité des différents types de capteurs sur la rétine (à droite).



L'interprétation de la couleur, est une combinaison complexe de tous ces capteurs et de biens d'autres facteurs. Il est intéressant de savoir qu'il existe des animaux possédant un seul type de capteur alors que d'autre peuvent en posséder jusqu'à 16. Ce qui leur donne une capacité de résolution colorimétrique loin devant tout autre être vivant.

LES ESPACES COLORIMÉTRIQUES

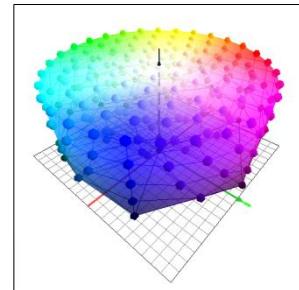
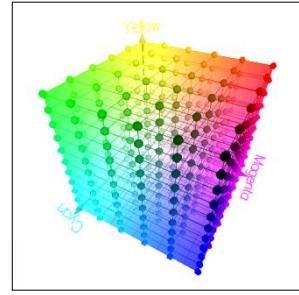
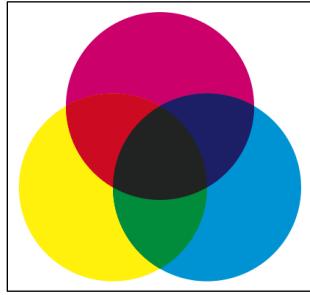
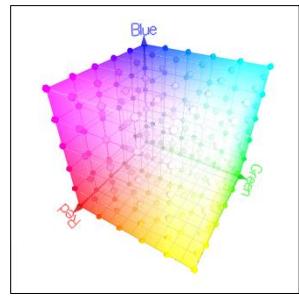
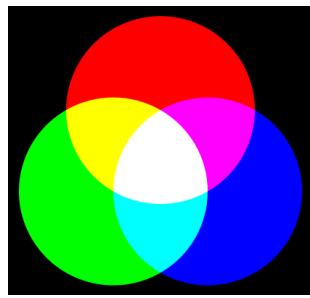
Les espaces colorimétriques (ou souvent appelé espace de couleurs) sont des outils mathématiques permettant de quantifier la couleur selon différents critères d'estimation. On y détermine plusieurs dimensions dans lesquelles chacune quantifie un attribut spécifique. Chaque couleur se caractérise donc ainsi par un point à n dimensions pour chaque couleur.

Le nombre de dimensions utilisées dépend de plusieurs facteurs à la fois :

- lorsqu'ils s'appliquent, les contraintes et raisons des capteurs mis en cause (réponse spectrale des caméras – par exemple, les caméras communes ont une réponse spectrale sur trois bandes de couleurs alors que certains en possèdent 1024);
- l'usage à laquelle est destinée l'information de couleur (rendu à l'écran, impression, comparaison pour peindre un mur, analyse théorique, vision artificielle, etc.);
- la facilité d'interprétation désirée par l'utilisateur (certains espaces colorimétriques utilisent des dimensions très abstraites pour un néophyte alors que d'autres utilisent des notions beaucoup plus intuitives).

À titre introductif, voici quelques espaces de couleurs assez simples à saisir :

- **L'espace RVB (Rouge-Vert-Bleu | communément appelé *RGB* en anglais).** Cet espace est basé sur les trois couleurs primaires en synthèse additive (synthèse de la lumière à la source). Cet espace est basé sur les trois cônes de l'œil humain et permet d'établir facilement toutes les couleurs perçues par l'homme. Cet espace de couleur est facile à comprendre et abondamment utilisé notamment pour l'affichage sur les écrans d'ordinateur. Il est important de savoir qu'il existe plusieurs variantes de cet espace de couleur et que celle utilisée pour générer des images sur un écran ne permet pas de générer toutes les couleurs que l'œil perçoit.
- **L'espace CMJN (Cyan-Magenta-Jaune-Noir | communément appelé *CMYK* en anglais).** Cet espace est basé sur les trois couleurs primaires en synthèse soustractive (synthèse de la lumière par réflexion sur une surface réfléchissante). Cet espace est très utilisé en imprimerie et c'est pourquoi une quatrième dimension y est ajoutée. En effet, dans l'espace CMJ, le noir se fait par l'addition en quantité égale des trois couleurs primaires mais il est très difficile d'atteindre un résultat satisfaisant. Trop souvent le résultat donne plutôt une couleur noire sale. Ainsi, on a ajouté un pigment supplémentaire noir pour l'imprimerie et créer un espace de couleur à quatre dimensions permettant des impressions de très haute qualité.
- **L'espace TSV (Teinte-Saturation-Valeur | communément appelé *HSV* en anglais).** Cet espace est en fait un réarrangement de l'espace RVB de façon telle à ce qu'il corresponde davantage à la perception humaine. Même si cet espace est loin d'être fidèle à la perception humaine, il permet de s'en rapprocher beaucoup par une opération simple. Ainsi, cet espace est plus près du langage et permet de nommer chaque caractéristiques de façon plus naturel. Par exemple, il est plus facile de dire qu'une couleur est rouge et saturée. Il est important de mentionner qu'il existe plusieurs formes de cet espace mais ils sont tous basés sur la même approche et présentes uniquement quelques variantes parfois subtiles des paramètres S et V.



Il existe un très grand nombre d'espaces colorimétriques selon différents modèles. Voici les plus communs avec leur nom anglais :

Famille des espaces de couleur basés sur	Noms
...	
les couleurs primaires	RGB, sRGB, Adobe RGB, Adobe Wide Gamut RGB, CMY, CMYK, ...
la luminance et la chrominance	YIQ, YUV, YDbDr, YPbPr, YCbCr, xvYCC, ...
la teinte et la saturation	HSV, HSL, HCL', ...
les études de la perception humaine (CIE)	CIE 1931 XYZ, CIELUV, CIELAB, CIEUVW, ...
autres	Modèle physique de la couleur, ...

Il ne faut pas croire que la science de la couleur en est une simple. En fait, elle est très complexe et met en relation plusieurs autres sciences. C'est pourquoi il est difficile d'avoir une compréhension complète du phénomène.

Le reste de ce document se concentre sur les applications informatiques des espaces de couleurs et met l'emphasis sur les transformation des espaces RVB et TSL. Au besoin, d'autres transformations pourraient être documentées.

ALGORITHMES DE TRANSFORMATION ENTRE LES ESPACES RVB ET TSV

Transformation de RVB vers TSV

```
H, S, V = RGB2HSV(R, G, B)

// Plage des valeurs d'entrée :   RGB = [0, 255]
// Plage des valeurs de sortie : HSV = [0, 1]

// Normalisation
R /= 255
G /= 255
B /= 255

// Maximum, minimum et étendu
M = max(R, G, B)
m = min(R, G, B)
Span = M - m

V = M

if (Span = 0)
    H = 0
    S = 0
else
    S = Span / M

dR = ((M - R)/6 + Span/2) / Span
dG = ((M - G)/6 + Span/2) / Span
dB = ((M - B)/6 + Span/2) / Span

if (M = R)
    H = dB - dG
else if (M = G)
    H = 1/3 + dR - dB
else if (M = B)
    H = 1/3 + dG - dR

if (H < 0)
    H += 1
else if (H > 0)
    H -= 1
```

Transformation de TSL vers RVB

```
R, G, B = HSV2RGB(H, S, V)

// Plage des valeurs d'entrée : HSV = [0, 1]
// Plage des valeurs de sortie : RGB = [0, 255]

if (S = 0)
    R = V * 255
    G = V * 255
    B = V * 255
else
    if (H = 1)
        H = 0                         // H doit être < 1

    dH = H * 6
    dI = floor(dH)

    v1 = V * (1 - S)
    v2 = V * (1 - S*(dH - dI))
    v3 = V * (1 - S*(1 - (dH - dI)))

    if      (dI = 0)      R = V      G = v3      B = v1
    else if (dI = 1)      R = v2      G = V      B = v1
    else if (dI = 2)      R = v1      G = V      B = v3
    else if (dI = 3)      R = v1      G = v2      B = V
    else if (dI = 4)      R = v3      G = v1      B = V
    else                  R = V      G = v1      B = v2

    R *= 255
    G *= 255
    B *= 255
```

APPLICATION DES TRANSFORMATIONS PRÉSENTÉES

Exemple de transformation d'images

À venir...

Exemple pour l'interprétation d'images

À venir...

ANNEXE 3 : FILTRES GAUSSIENS 1D ET 2D

SOMMAIRE

Ce document présente la notion de filtre gaussien appliqué à un signal donné. On présente d'abord la fonction de distribution gaussienne dans l'espace 1D afin de bien saisir les concepts inhérents à ce type de fonction pour poursuivre la définition dans l'espace 2D.

Il est important de comprendre que le filtre gaussien tel qu'utilisé ici correspond à un noyau² spécifique dans le contexte de l'opération de convolution sur un signal d'entrée. La convolution est un opérateur mathématique faisant la transformation linéaire de deux fonctions initiales afin d'en créer une troisième. Ici, la première fonction est le signal d'entrée et la deuxième est le noyau de convolution.

L'opérateur de convolution correspond à :

$$S'_i = \sum_{\forall j \in k} S_{i+j} \cdot k_j \quad \forall i \in S$$

où :
 S est le signal d'entrée
 S' est le signal de sortie (le résultat de la convolution)
 k est le noyau de convolution

Le noyau de convolution k peut être de n'importe quelle nature et cet article présente comment utiliser un noyau de convolution de type gaussien sur un signal 1D et 2D.

ESPACE 1D

D'abord, la fonction gaussienne dans l'espace 1D se définit par :

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

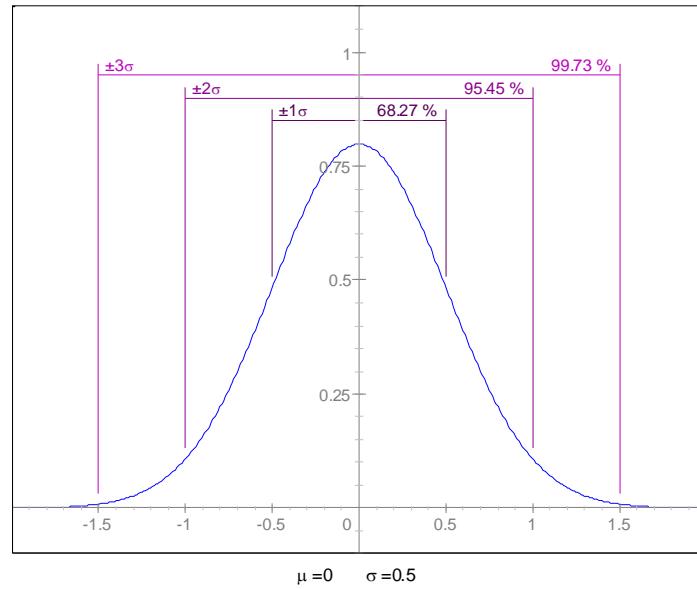
où :
 μ correspond à la moyenne de la distribution
 σ correspond à l'écart type de cette dernière

Plusieurs éléments de discussion peuvent être abordés concernant cette fonction. Rappelons seulement la signification de μ et σ . Dans un premier temps, la moyenne permet de positionner la valeur de crête là où le filtre doit être le plus pertinent. Quant à lui, l'écart type représente la densité autour de la moyenne. Sans entrer trop dans la théorie, il est important de comprendre qu'à $\pm n \times \sigma$, nous retrouvons une proportion spécifique de l'échantillonnage (l'aire sous la courbe). Par exemple :

- $\pm 1\sigma$ donne 68.27 % de l'aire sous la courbe
- $\pm 2\sigma$ donne 95.45 % de l'aire sous la courbe
- $\pm 3\sigma$ donne 99.73 % de l'aire sous la courbe

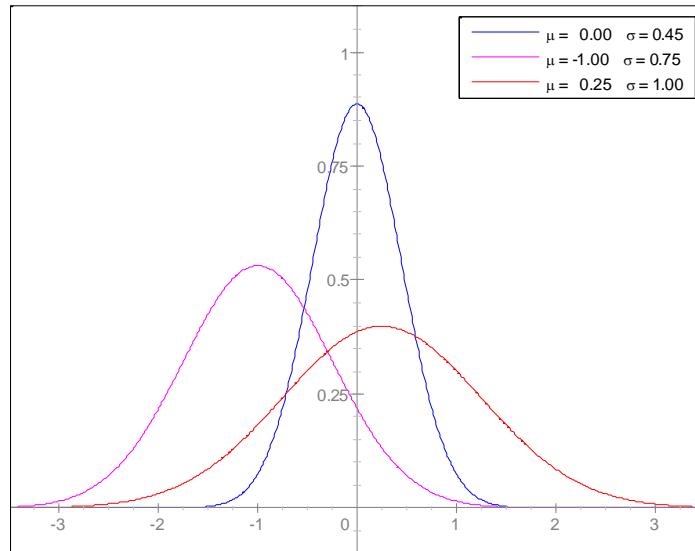
² Noyau de convolution est souvent appelé masque de convolution.

Distribution gaussienne



Voici quelques exemples spécifiques de courbes :

Exemples de courbe gaussienne



Voici maintenant, les étapes de construction d'un noyau pouvant être appliquée à l'opérateur de convolution :

1. définir la longueur du noyau

- L – La longueur du noyau de convolution a pour effet d'amplifier l'effet de filtrage et d'étendre à une grande surface le calcul de moyenne pondérée effectué. Même s'il est possible de créer des noyaux de n'importe quelle dimension, on privilégie des tailles de dimension impaire facilitant ainsi le positionnement sur chaque échantillon. Dans le but d'alléger le document, on prendra pour acquis que L est impair à partir de ce point.

2. déterminer les paramètres de la distribution gaussienne (μ et σ)

- μ – La moyenne permet de définir la position de la valeur crête de la distribution. De façon plus concrète, on doit interpréter ce paramètre comme étant la partie du signal qui doit être considérée la plus significative pour chaque itération de la convolution. Un débordement par rapport au centre aura pour effet de déplacer vers la gauche ou vers la droite le signal. On souhaite généralement centrer la distribution sur le noyau et pour ces cas on utilisera la formule suivante :

$$\mu = 1 + \frac{L - 1}{2}$$

- σ – L'écart type permet de définir l'amplitude des valeurs autour de la moyenne. Concrètement c'est l'étendue du filtre et la pondération relative des données autour de la moyenne. Généralement, on souhaite avoir une distribution uniforme des données sur le domaine disponible et l'équation suivante permet de déterminer un σ intéressant indépendant de la dimension du noyau :

$$\sigma = \frac{L - 1}{5}$$

3. construire le noyau de convolution

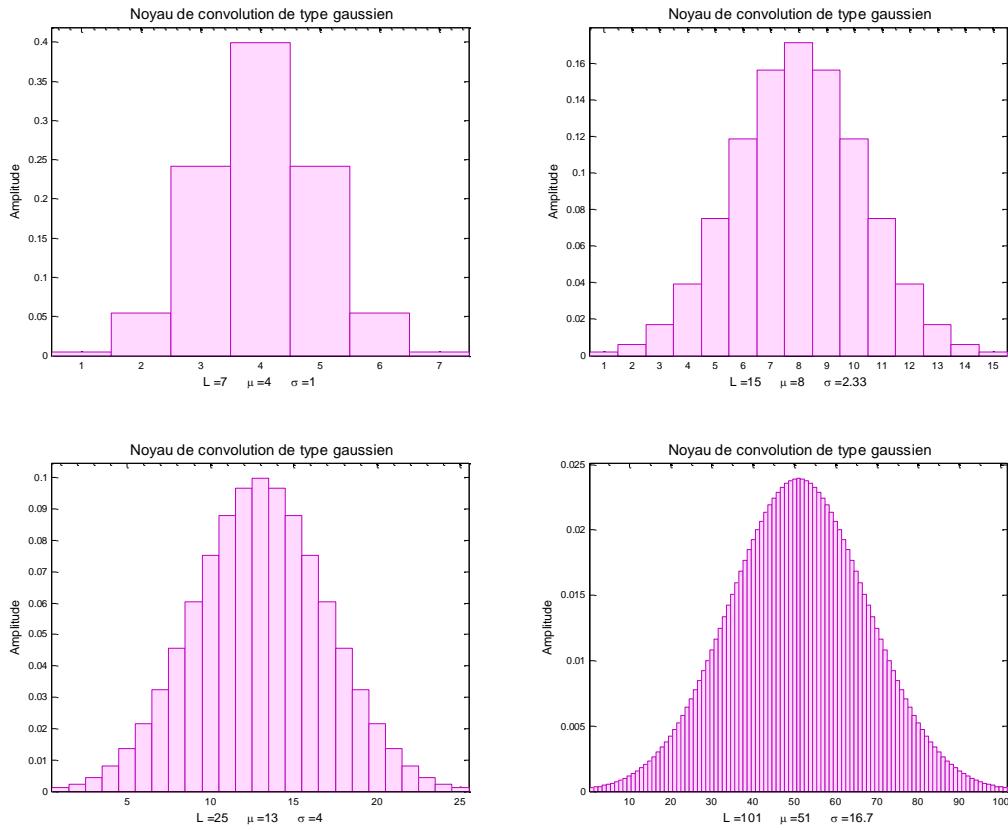
- k – Cette étape est très simple et consiste simplement à appliquer l'équation plus haut en fonction des paramètres établis.

4. normaliser le filtre en amplitude si nécessaire

- L'application d'une convolution avec un noyau de convolution arbitraire a pour effet de changer la plage dynamique de la réponse. Par exemple, si le signal était borné dans l'intervalle $[0, 1]$, l'application d'un filtre quelconque peut modifier l'amplitude initiale dans un autre intervalle (par exemple $[-10, 10]$). Or, dans de nombreux cas, le signal doit rester borné dans l'intervalle initial. Au lieu de renormaliser le signal après l'application du filtre, il est plus pertinent et performant de normaliser le filtre lui-même et ainsi s'assurer que le signal de sortie respecte la même plage que le signal d'entrée. Voici la méthode très simple pour y arriver :

$$k_{norm} = \frac{k}{\sum_{\forall i \in k} k_i}$$

Exemple de noyau de convolution gaussien :



ESPACE 2D

L'introduction dans l'espace 1D permet de comprendre plus facilement les concepts généraux qui s'appliquent de la même façon dans un espace de dimensions supérieures. Plus spécifiquement, dans l'espace 2D, le concept reste identique mais s'applique légèrement différemment. Dans un espace bidimensionnel, la fonction gaussienne se définit par :

$$f(x, y) = \frac{1}{\sqrt{2\pi}\sigma_x\sigma_y} e^{-\left(\frac{(x-\mu_x)^2}{2\sigma_x^2} + \frac{(y-\mu_y)^2}{2\sigma_y^2}\right)}$$

où : μ_x et μ_y correspondent aux moyennes de chacun des axes
 σ_x et σ_y correspondent aux écarts type de chacun des axes

Cette formulation en est une parmi plusieurs autres et elle a le mérite d'être plutôt simple et pratique. Néanmoins, elle ne permet pas d'appliquer une rotation de la distribution par rapport à l'axe Z (axe vertical). Le lecteur intéressé pourra se référer à cette autre formulation :

$$f(x, y) = Ae^{-\left(a(x-\mu_x)^2 + 2b(x-\mu_x)(y-\mu_y) + c(y-\mu_y)^2\right)}$$

Cette formulation utilise une définition polynomiale stricte dont les paramètres (A , a , b et c) sont définitivement plus abstrait (sauf A étant un facteur d'amplitude). Néanmoins, il est possible d'établir ces paramètres selon d'autres paramètres que nous connaissons davantage (σ et θ).

$$a = \frac{\cos^2 \theta}{2\sigma_x^2} + \frac{\sin^2 \theta}{2\sigma_y^2}$$

$$b = -\frac{\sin 2\theta}{4\sigma_x^2} + \frac{\sin 2\theta}{4\sigma_y^2}$$

$$a = \frac{\sin^2 \theta}{2\sigma_x^2} + \frac{\cos^2 \theta}{2\sigma_y^2}$$

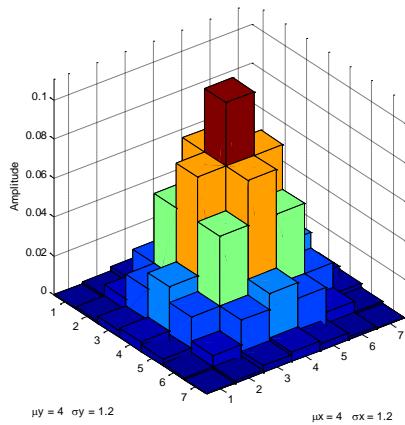
Cette deuxième formulation est laissée au lecteur chevronné et nous retournons à la première formulation plus standard.

Tous les concepts que nous avons vu pour la construction du noyau de convolution 1D s'appliquent à celui de l'espace 2D avec les mêmes astuces et stratégies.

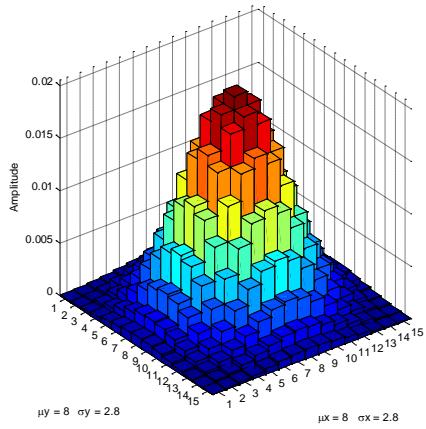
1. définir la taille du noyau (L_x et L_y)
2. déterminer les paramètres de la distribution gaussienne (μ_x , μ_y , σ_x et σ_y)
3. construire le noyau de convolution
4. normaliser le filtre en amplitude si nécessaire

Voici plusieurs exemples :

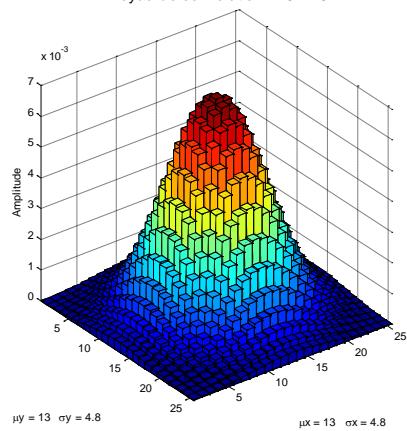
Noyau de convolution : 7×7



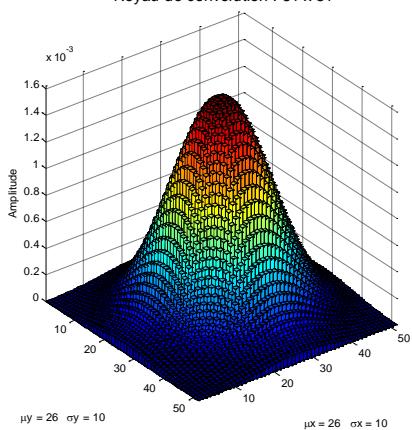
Noyau de convolution : 15×15



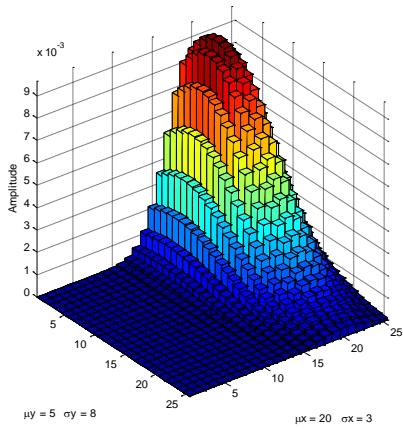
Noyau de convolution : 25×25



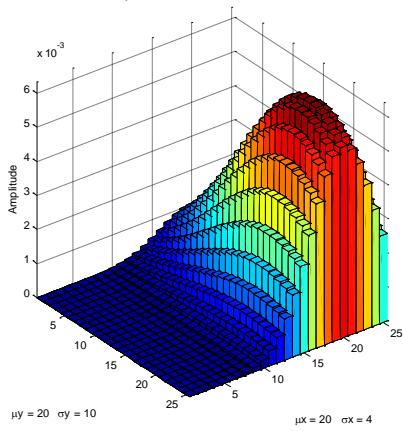
Noyau de convolution : 51×51



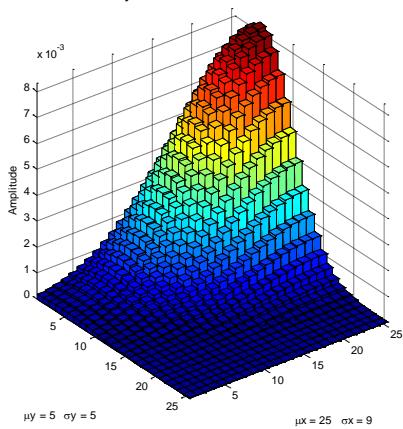
Noyau de convolution : 25×25



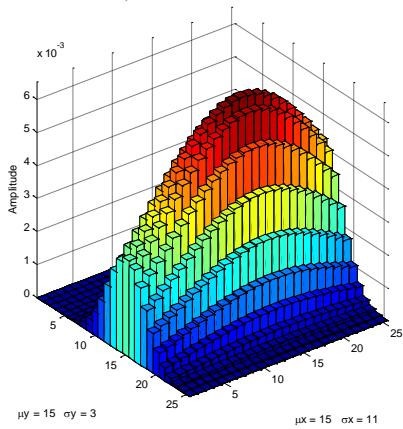
Noyau de convolution : 25×25



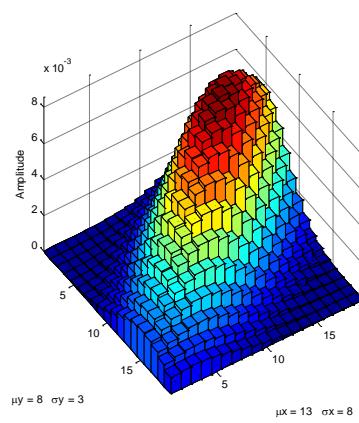
Noyau de convolution : 25×25



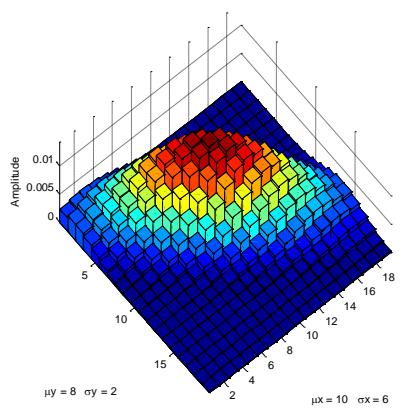
Noyau de convolution : 25×25



Noyau de convolution : 19×19 à 36°



Noyau de convolution : 19×19 à 144°



ANNEXE 4 : FILTRE DIRECTIONNEL DE SOBEL

SOMMAIRE

Dans le contexte d'analyse automatisée d'image il est fréquent de vouloir connaître l'amplitude et l'orientation des arêtes dans les images (parfois appelé les paramètres de la dérivée directionnelle).

Évidemment, il existe beaucoup de théorie sur le sujet et plusieurs façons d'estimer les paramètres de cette dérivée. Néanmoins, l'opérateur Sobel est un opérateur relativement simple qui permet de faire une approximation intéressante de ces paramètres.

Cet opérateur est généralement classé dans la catégorie des rehausseurs ou détecteurs d'arêtes (« *edges detection* »). Si on compare ce filtre aux autres du projet, ce dernier présente quelques caractéristiques différentes :

- d'abord il prends une seule image en entrée alors qu'il génère deux images de sorties
 - l'amplitude de la dérivée directionnelle
 - l'orientation de la dérivée directionnelle
- de plus, il requiert quatre opérations différentes pour obtenir les résultats

Voici comment réaliser ce filtre :

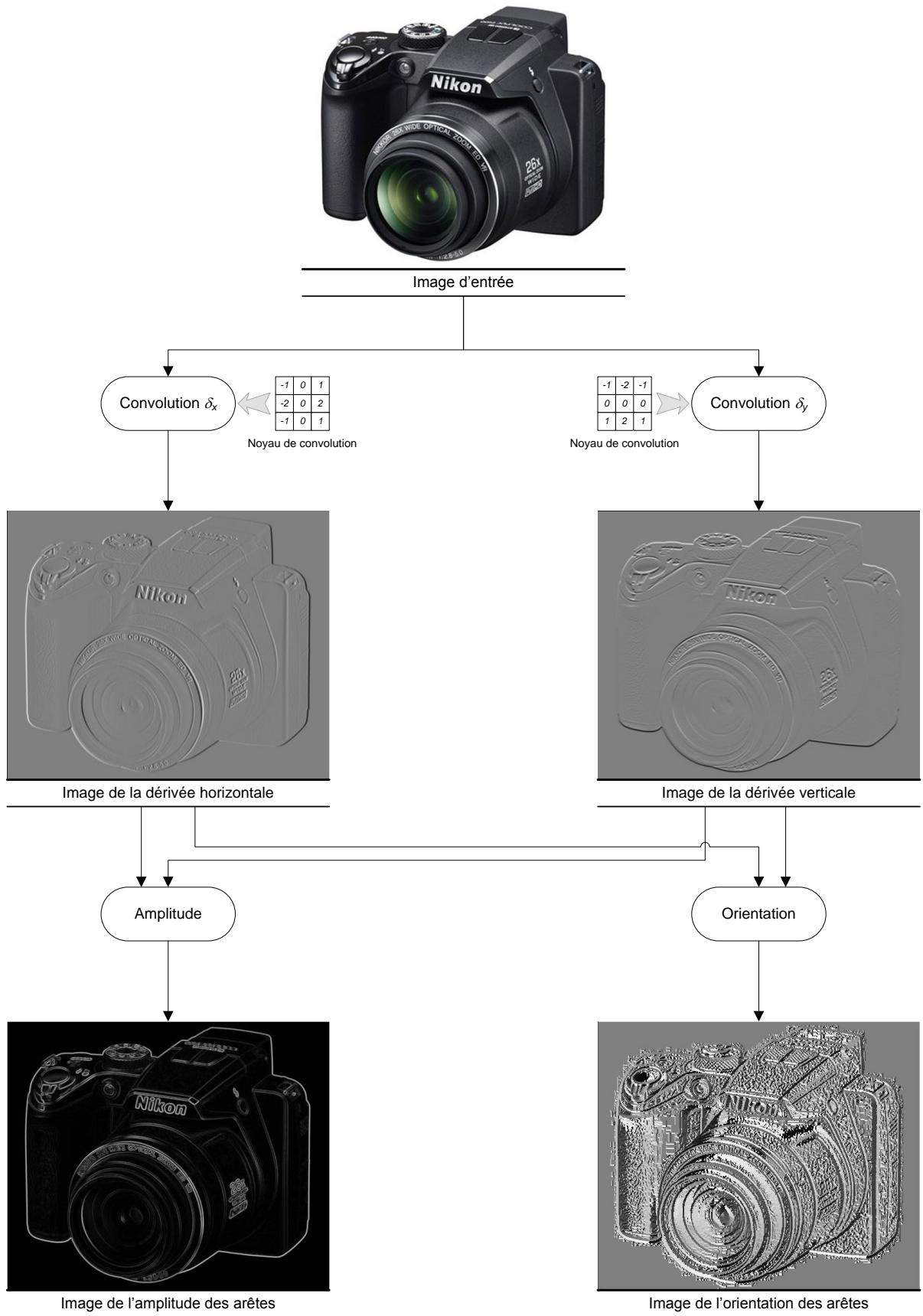
1. On calcule la dérivée verticale (δ_x) pour détecter les arêtes horizontales en appliquant une convolution avec le masque de *Sobel* vertical.
2. On calcule la dérivée horizontale (δ_y) pour détecter les arêtes verticales en appliquant une convolution avec le masque de *Sobel* horizontal.
3. On calcule l'amplitude de la dérivée totale (A) par :

$$A_{i,j} = \sqrt{{\delta_{x_{i,j}}}^2 + {\delta_{y_{i,j}}}^2}$$

4. On calcule l'orientation de la dérivée totale (O) par :

$$O_{i,j} = \tan^{-1} \left(\frac{\delta_{y_{i,j}}}{\delta_{x_{i,j}}} \right)$$

Il est conseillé d'utiliser la fonction `atan2` qui permet de définir l'arc tangente selon toute la plage possible $[-\pi, \pi]$. Aussi, contrairement à la fonction `atan`, l'usage de cette fonction permet d'éviter la gestion des cas limites engendrés par les divisions par 0 (lorsque δ_x vaut 0).



Les images générées par cet opérateur sont particulières et doivent être traitées en conséquence. Les plages de sorties ne sont pas standards et il importe de les normaliser si vous voulez avoir un aperçut adéquat des résultats.

En effet, dans un contexte d'analyse d'images, la normalisation n'est pas nécessaire. Par contre, il est essentiel de le faire pour permettre leur affichage à l'écran. Ainsi, en utilisant une image monochrome dont chaque pixel varie entre 0 et 255, il sera possible d'illustrer les résultats obtenus (d'autres alternatives couleur peuvent être envisagées).

Donc, voici les plages de résultats possibles :

1. Images des dérivées (δ) : [-255, 255]
2. Image d'amplitude (A) : [0, $\sqrt{2}(2^8 - 1)$]
3. Image d'orientation (O) : [- π , π]

Si on considère que l'objectif de la normalisation ici est simplement d'afficher au mieux possible les résultats obtenus, il est recommandé d'appliquer cette technique.

$$\Delta_{x_{i,j}} = 255 \left(\frac{\delta_{x_{i,j}} - min_{\delta}}{max_{\delta} - min_{\delta}} \right) \quad \text{et} \quad \Delta_{y_{i,j}} = 255 \left(\frac{\delta_{y_{i,j}} - min_{\delta}}{max_{\delta} - min_{\delta}} \right)$$

où : $\Delta_{x_{i,j}}$ et $\Delta_{y_{i,j}}$ correspondent aux pixels normalisés de $\delta_{x_{i,j}}$ et $\delta_{y_{i,j}}$ respectivement
 min_{δ} est la valeur minimum trouvée en parcourant complètement les deux images δ_x et δ_y
 max_{δ} est la valeur maximum trouvée en parcourant complètement les deux images δ_x et δ_y

$$\Lambda_{i,j} = 255 \left(\frac{A_{i,j} - min_A}{max_A - min_A} \right)$$

où : $\Lambda_{i,j}$ correspond au pixel normalisé de $A_{i,j}$
 min_A est la valeur minimum trouvée en parcourant complètement l'image A
 max_A est la valeur maximum trouvée en parcourant complètement l'image A

$$\varphi_{i,j} = 255 \left(\frac{O_{i,j} + \pi}{2\pi} \right)$$

où : $\varphi_{i,j}$ correspond au pixel normalisé de $O_{i,j}$

Attention, il est important de faire la normalisation seulement après avoir fait tous les calculs intermédiaires.

IMAGES DE L'EXEMPLE EN PLEINE RÉSOLUTION



Image d'entrée



Image de la dérivée horizontale

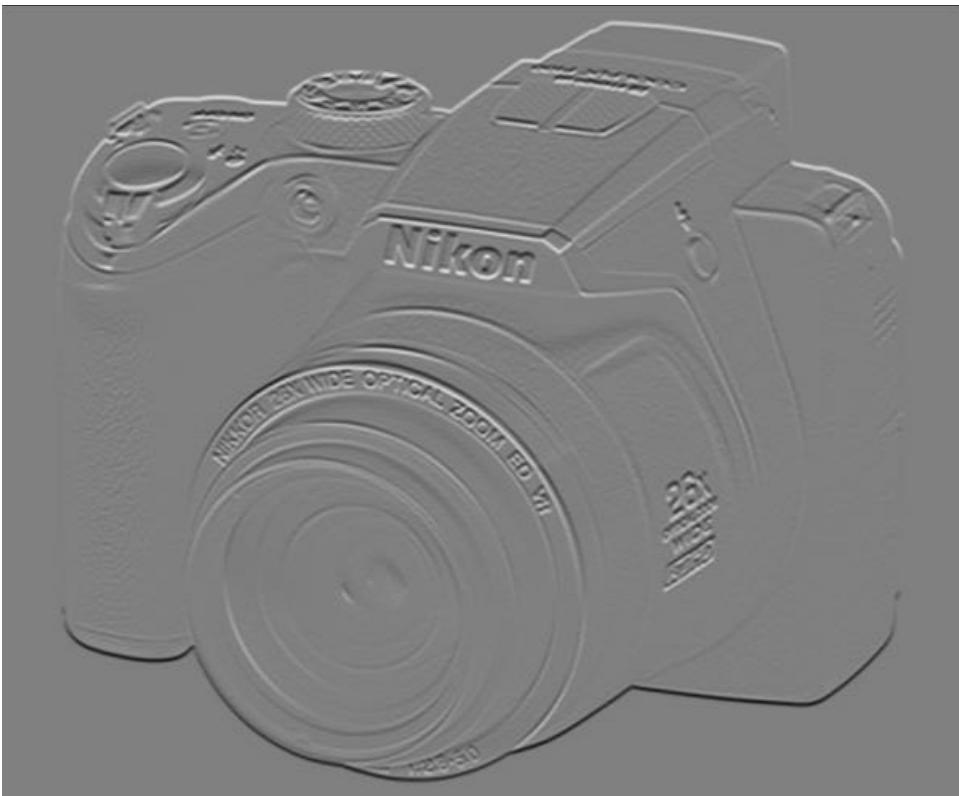


Image de la dérivée verticale



Image d'amplitude

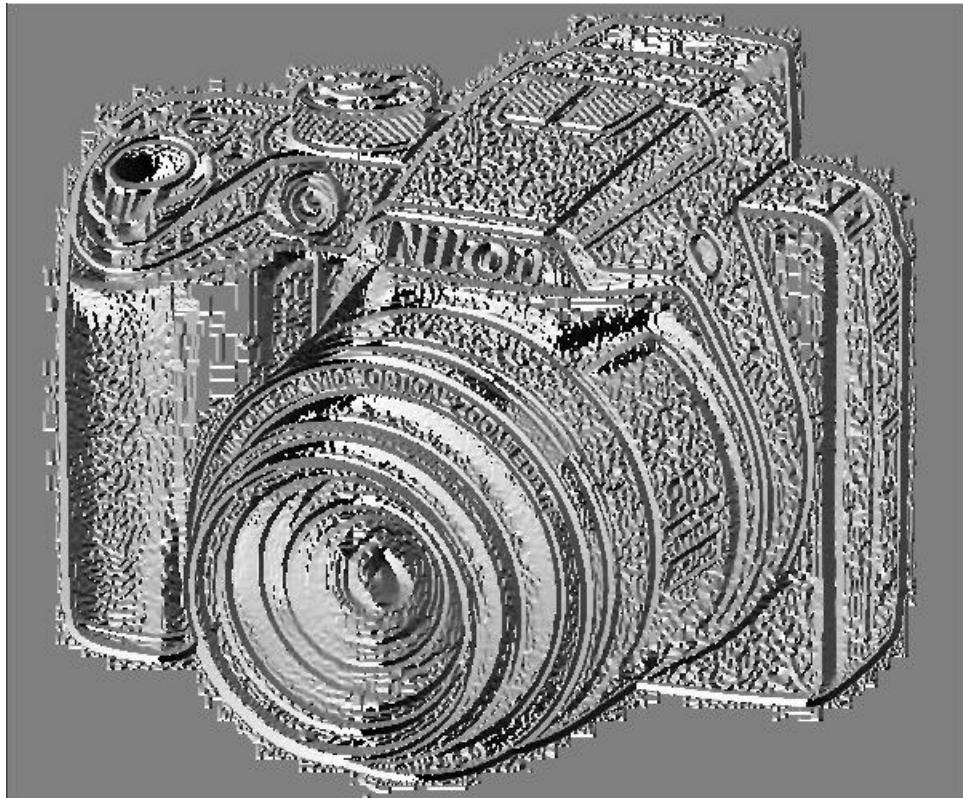


Image d'orientation (comprenez-vous bien la signification de cette image?)

ANNEXE 5 : UNIFORMISATION D'ÉCLAIRAGE

SOMMAIRE

L'uniformisation d'éclairage est l'un des processus les plus utilisés dans les applications de vision artificielle. Il consiste à uniformiser, par l'usage d'algorithme, les variations d'intensité de la source lumineuse. Lorsqu'on fait l'acquisition d'une image, il est généralement très difficile (voire impossible dans la majorité des cas) d'obtenir un éclairage parfaitement uniforme. Ce manque d'uniformité a pour effet d'augmenter la difficulté des algorithmes de traitement sous-jacent, principalement la segmentation.

Pour palier à ce problème important, les fabricants de matériel de vision offrent avec leurs produits (caméras, cartes d'acquisition et bibliothèques de traitement d'images) plusieurs outils faisant une correction approximative de la variation d'éclairage.

Nous allons voir ici une des méthodes existantes permettant de faire cette correction. En fait, il existe plusieurs techniques plus ou moins complexes basées sur des approches scientifiquement et technologiquement différentes. La méthode présentée ici a l'avantage d'être assez robuste dans plusieurs cas différents :

- Elle donne de bons résultats dans des cas très différents présentant des variations quelconques de l'intensité. Que ce soit pour une seule source lumineuse ou plusieurs sources à la fois.
- Elle permet une correction efficace des images monochromes et souvent satisfaisante des images couleurs. Malgré tout, une adaptation de la solution présentée permet d'améliorer grandement les performances avec des images couleurs.
- Elle peut être utilisée avec et sans image de référence. Évidemment, les résultats restent toujours meilleurs avec une image de référence.

Néanmoins, la technique présentée reste une approximation et ne prétend à aucune justesse d'un point de vue scientifique.

CONCEPT

Le concept à la base de ce traitement est très simple et consiste à normaliser localement l'amplitude du signal de l'image (l'intensité de chaque pixel). Lorsqu'on parle de normalisation locale, on fait référence au fait d'estimer la valeur d'un pixel selon l'intensité moyenne de la région autour.

Ainsi, l'idée consiste à construire une image de normalisation qu'on appellera « carte d'éclairage ». Cette carte d'éclairage représentera une estimation de l'éclairage moyen d'une région.

RÉALISATION

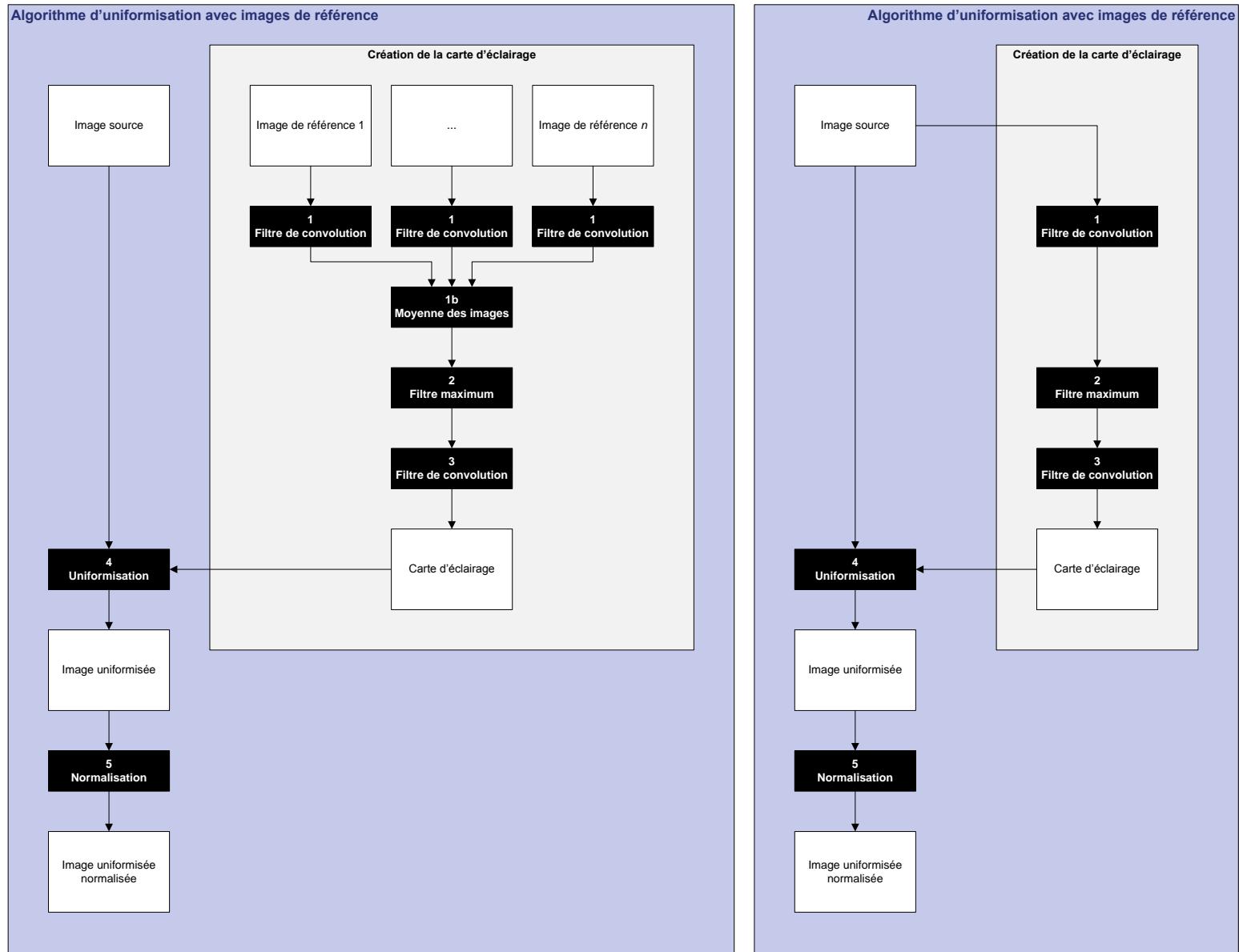
Deux approches pratiquement identiques peuvent être considérées :

- sans image de référence
- avec image(s) de référence

Généralement, lorsqu'on a le choix, on optera toujours pour le processus utilisant plusieurs images de référence. Il faut comprendre que dans ce contexte précis, le terme image de référence, représente une

image de la scène sans l'élément qu'on tente d'observer mais **avec les mêmes conditions d'éclairage**. Si les conditions d'éclairage changent, alors l'image de référence devient inutile.

Les deux schémas suivants représentent les deux algorithmes d'uniformisation d'éclairage :



On remarque bien l'étroite relation entre chacune des méthodes. Seule la façon de construire la carte d'éclairage diffère. Encore là, la méthode est très semblable.

Étape 1 – Filtre de convolution

Ce premier filtre a pour objet de filtrer les images qui serviront à construire la carte d'éclairage. Ce premier filtrage prépare les images pour le filtre maximum qui suivra. Nous tentons d'éviter que le filtre maximum rehausse à outrance certains problèmes d'acquisition tels que le bruit ou tout autre élément perturbant dans la structure de l'image.

La convolution utilise ici un noyau de convolution gaussien ou uniforme (le filtre gaussien donnera de meilleur résultat). La taille du filtre peut varier selon plusieurs critères :

- taille de l'image
- résolution de l'image
- disparité d'éclairage
- complexité de la scène
- objet particulier de la scène
- objectif spécifique de l'uniformisation

Il est fréquent de voir des tailles varier entre 11×11 jusqu'à 33×33 .

Étape 1b – Moyenne des images

Dans le cas où sont utilisées plusieurs images de référence en entrée pour créer la carte d'éclairage, il est important de faire la somme de toutes les images réalisées à l'étape 1. En faisant la moyenne des images, on obtient une image synthèse dans la même plage que l'image source.

$$I' = \frac{1}{n} \sum_{i=1}^n I_i$$

où I' est l'image de sortie, I_i est la $i^{\text{ème}}$ image d'entrée et n est le nombre d'images disponibles.

Étape 2 – Filtre maximum

Ce filtre tente de rehausser les parties les plus visibles de l'image. Pour différentes raisons théoriques, on souhaite augmenter l'intensité locale afin de construire une carte d'éclairage plus efficace.

Encore une fois, la taille du noyau peut varier beaucoup. Mais généralement, il est égale ou légèrement plus grand que celui de l'étape précédente.

Étape 3 – Filtre de convolution

Cette étape est la plus cruciale et la plus longue en terme de temps de calcul. Elle consiste à lisser l'image de façon telle à ne plus voir aucun élément structural de cette dernière. En fait, on recherche une image la plus lisse possible montrant la variation d'intensité moyenne sur une grande surface.

Pour cette raison, on utilisera encore un noyau de type uniforme ou gaussien (préféablement gaussien). Mais cette fois-ci, la taille du noyau sera très grande, entre 5 à 10 fois la taille utilisée à l'étape 1. Il faut faire attention et ne pas choisir une taille inutilement trop grande qui aura comme effet de produire une carte d'éclairage inadéquate.

Étape 4 – Uniformisation

Lorsque la carte d'éclairage est faite, la partie uniformisation est très rapide. Il ne suffit que de diviser l'image source par la carte d'éclairage pixel par pixel. Attention à vérifier la validité de la division (division par zéro interdite).

$$I' = \frac{I}{CE}$$

Où I' est l'image de sortie, I est l'image d'entrée et CE est la carte d'éclairage déjà créée.

Étape 5 – Normalisation

Après l'uniformisation, l'image résultante aura une plage de données qui variera beaucoup selon divers critères. Dans des conditions parfaites, chaque pixel variera dans l'intervalle [0, 1]. Néanmoins, il arrive toujours que des aspérités et artéfacts créent des zones de l'image qui peuvent varier davantage. Il est courant de voir des variations pouvant aller jusqu'à [0, 5]. Ainsi, il importe de normaliser l'image obtenue selon la plage de travail qui est désirée.

On désirera généralement normaliser les pixels soit entre [0, 1] ou entre [0, 255]. Dans les deux cas, plusieurs méthodes peuvent être appliquées, tout dépend du résultat voulu.

La méthode suivante permet de normaliser efficacement dans plusieurs cas.

$$I' = \frac{I - \min}{\max - \min} \times \text{val}$$

Où I' est l'image de sortie, I est l'image d'entrée, \min et \max sont l'intensité minimum et maximum parmi tous les pixels de l'image et val est la valeur de normalisation maximum souhaitée (par exemple 1 ou 255). Le résultat de cette normalisation donne une plage dans l'intervalle [0, val].

OPTIMISATION

L'exécution complète de cet algorithme est très long et prends un temps considérable à exécuter. En tentant de bien comprendre les détails du processus, on remarque que la partie la plus longue est l'estimation de la carte d'éclairage alors que la normalisation elle-même est un processus beaucoup plus rapide.

L'usage d'image(s) de référence permet de créer la carte d'éclairage une seule fois et de l'utiliser par la suite. Tant que les conditions d'acquisition et d'éclairage sont les mêmes, on utilise la même carte d'éclairage afin de faire la normalisation. Le temps sauvé ainsi est plus que significatif.

Aussi, l'opération de division est beaucoup plus lente que l'opération de multiplication et pour cette raison on privilégiera de créer l'inverse de la carte d'éclairage. Ainsi, l'étape de normalisation deviendra :

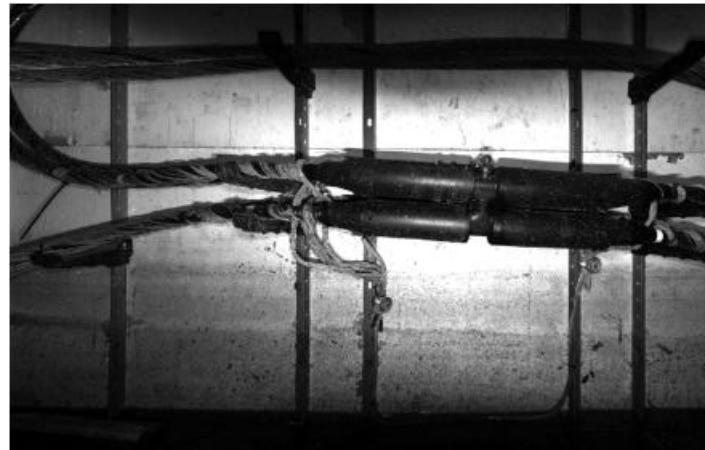
$$I' = I \times CE_{Inv}$$

Où I' est l'image de sortie, I l'image d'entrée et CE_{Inv} est l'inverse de la carte d'éclairage soit : $CE_{Inv} = \frac{1}{CE}$

Premier exemple – Image d'un souterrain

On remarque comment les objets la partie inférieure de l'image deviennent clairs.

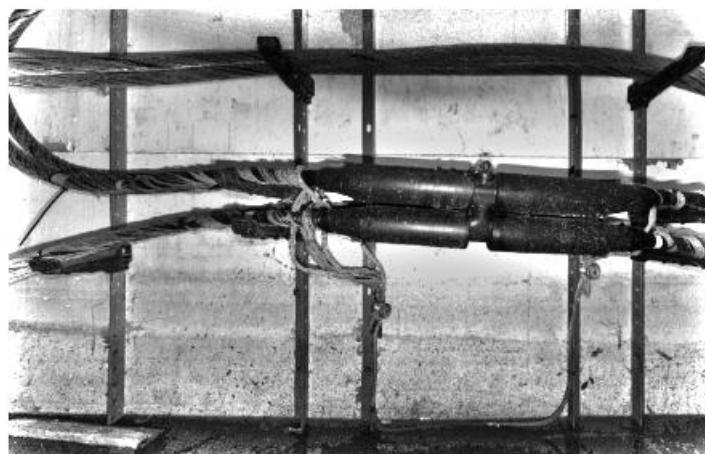
Image source



Carte d'éclairage



Éclairage uniformisé



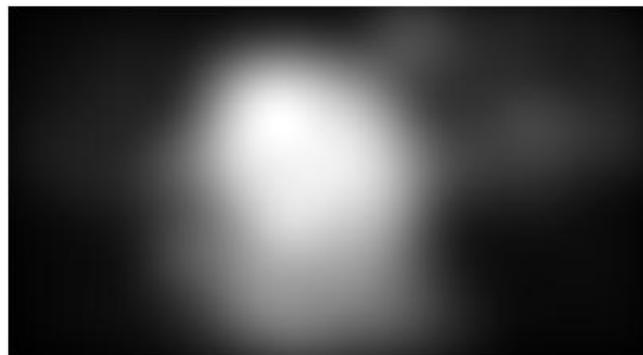
Deuxième exemple – Image d'une jeune fille

On remarque comment le personnage à droite de l'image devient apparent. Aussi, la texture de la pierre à gauche de l'image devient elle aussi très évidente.

Image source



Carte d'éclairage



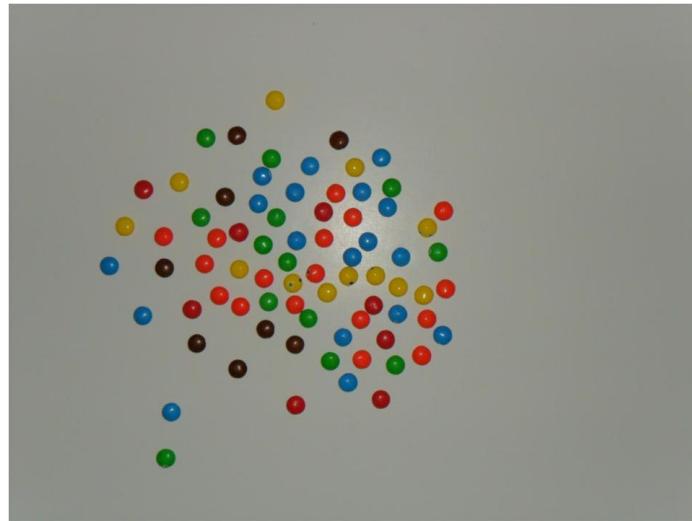
Éclairage uniformisé



Troisième exemple – Image de Smarties

On remarque comment le point chaud au centre de l'image devient plus uniforme. Pour une segmentation couleur, il est souvent difficile de tenir compte des variations d'intensité au centre de l'image par rapport aux bords.

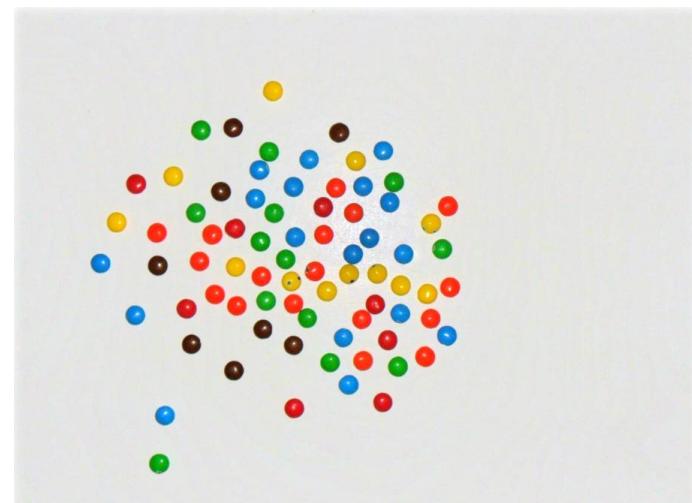
Image source



Carte d'éclairage



Éclairage uniformisé



ANNEXE 6 : ALGORITHMES DE REMPLISSAGE

SOMMAIRE

Le remplissage de région bornée est une tâche récurrente dans plusieurs contextes applicatifs de traitement de données (traitement de signal, graphisme, jeux vidéo, etc.).

Le remplissage consiste à mettre à une valeur nominale toute les éléments définie contiguë d'un ensemble de données référencées spatialement. Dans un espace à deux dimensions, il existe habituellement deux variante de proximité acceptée : à 4 ou à 8 voisins. L'utilisation d'une approche plutôt que l'autre dépend uniquement du domaine d'application et des éléments de conception requis.

Néanmoins, on peut dire qu'il existe une différence fondamentale entre un résultat et l'autre. Le premier, à 4 voisins, donne un remplissage beaucoup moins sévère que celui à 8 voisins.

CONCEPT

D'un point de vue conceptuel, l'implémentation de cette tâche se fait simplement par un algorithme récursif qui consiste à visiter les voisins immédiats et faire le changement de valeur lorsque nécessaire.

Malheureusement, cette approche élégante n'est pas optimale si on considère les contraintes d'implémentation et d'exécution telles que le temps d'exécution requis et surtout le nombre d'appels récursifs engendrés pour des remplissages de grandes tailles.

Pire encore, certains environnements d'exécution limités en ressource ne pourront pas réaliser tous les appels récursifs nécessaires étant donné la limitation de la taille mémoire disponible pour la pile d'exécution. Par exemple, certains langages tels que *JavaScript*, *ActionScript*, *Python*, *Matlab* et même *Java* ont tous une limite qui vont bien au-delà des langages tels que le *C* ou le *C++*.

Ainsi, pour plusieurs raisons, il sera pertinent d'utiliser une version itérative du remplissage. Malgré le fait que ces algorithmes soient moins élégants et moins faciles à comprendre, ils sont souvent préférés pour des raisons de performance en vitesse et de limitation de ressources.

Finalement, il existe plusieurs algorithmes différents permettant de faire ces différentes implémentations. En plus des versions récursives à 4 et 8 voisins, on présente ici deux versions itératives relativement simples. Ces algorithmes ne sont pas nécessairement les plus performant mais ont le mérite d'être assez simple à implémenter, conceptuellement plus accessibles que d'autres et assez performante malgré tout.

Les algorithmes présentés font référence à des données spatiales 2D telle qu'une image.

ALGORITHME

Paramètres d'entrée

- **Data** Données 2D d'entrée qui seront modifiées (l'image par exemple)
 - S'applique peut importe le type
 - S'applique peut importe la taille
- **x, y** Point de départ du remplissage
- **vc** Valeur cible à remplacer
- **vr** Valeur de remplacement

Résultats de sortie

- Aucun paramètre spécifique retourné
- Néanmoins, les données d'entrée sont modifiées selon le remplissage effectué

Algorithme récursif à 4 voisins

```
01 void Remplissage(Data, x, y, vc, vr)
02
03     // Vérifie la condition d'arrêt de l'algorithme récursif
04     if (Data[x][y] != vc) {
05         return;
06     }
07
08     // Applique le remplacement de valeur
09     Data[x][y] = vr;
10
11    // Effectue la recherche des voisins dont la valeur est à remplacer
12    Remplissage(Data, x-1, y, vc, vr)
13    Remplissage(Data, x+1, y, vc, vr)
14    Remplissage(Data, x, y-1, vc, vr)
15    Remplissage(Data, x, y+1, vc, vr)
16
17
```

Algorithme récursif à 8 voisins

```
01 void Remplissage(Data, x, y, vc, vr)
02
03     // Vérifie la condition d'arrêt de l'algorithme récursif
04     if (Data[x][y] != vc) {
05         return;
06     }
07
08     // Applique le remplacement de valeur
09     Data[x][y] = vr;
10
11    // Effectue la recherche des voisins dont la valeur est à remplacer
12    Remplissage(Data, x-1, y-1, vc, vr)
13    Remplissage(Data, x , y-1, vc, vr)
14    Remplissage(Data, x+1, y-1, vc, vr)
15    Remplissage(Data, x-1, y , vc, vr)
16    Remplissage(Data, x+1, y , vc, vr)
17    Remplissage(Data, x-1, y+1, vc, vr)
18    Remplissage(Data, x , y+1, vc, vr)
19    Remplissage(Data, x+1, y+1, vc, vr)
20
```

Algorithme itératif à 4 voisins

```
01  /* Cet algorithme requiert l'usage d'une structure supplémentaire de type file avec ses méthodes
02  courantes d'usage : queue(), add(), remove(). */
03
04  void Remplissage(Data, x, y, vc, vr)
05
06      // Initialiser la file
07      Queue ListeARemplacer();
08
09      // S'assure que la valeur d'entrée soit à remplacer et amorce le remplissage
10      if (Data[x][y] != vc) {
11          return;
12      } else {
13          ListeARemplacer.Add(x, y);
14      }
15
16      // Parcours la liste des données à remplacer
17      for (i = 0; i < ListeARemplacer.length(); i++) {
18          x, y = ListeARemplacer[i];
19          if (Data[x][y] == vc) {
20              // Recherche de l'élément contigu le plus à gauche devant être remplacé
21              G = x;
22              while (G >= 0 && Data[G][y] == vc) {
23                  Data[G][y] = vr;
24                  if (y > 0 && Data[G][y-1] == vc) {                                // Insère le voisin du haut si
25                      adéquat
26                          ListeARemplacer.Add(G, y-1);
27                  }
28                  if (y < Data.Height()-1 && Data[G][y+1] == vc) {        // Insère le voisin du bas si
29                      adéquat
30                          ListeARemplacer.Add(G, y+1);
31                  }
32                  G--;
33              }
34
35              // Recherche de l'élément contigu le plus à droite devant être remplacé
36              D = x + 1;
37              while (D < Data.Width() && Data[D][y] == vc) {
38                  Data[D][y] = vr;
39                  if (y > 0 && Data[D][y-1] == vc) {                                // Insère le voisin du haut si
40                      adéquat
41                          ListeARemplacer.Add(D, y-1);
42                  }
43                  if (y < Data.Height()-1 && Data[D][y+1] == vc) {        // Insère le voisin du bas si
44                      adéquat
45                          ListeARemplacer.Add(D, y+1);
46                  }
47                  D++;
48              }
49          }
50      }
51  }
```

Algorithme itératif à 8 voisins

```
01  /* Cet algorithme requiert l'usage d'une structure supplémentaire de type file avec ses méthodes
02  courantes d'usage : queue(), add(), remove(). */
03
04  void Remplissage(Data, x, y, vc, vr)
05
06      // Initialiser la file
07      Queue ListeARemplacer();
08
09      // S'assure que la valeur d'entrée soit à remplacer et amorce le remplissage
10      if (Data[x][y] != vc) {
11          return;
12      } else {
13          ListeARemplacer.Add(x, y);
14      }
15
16      // Parcours la liste des données à remplacer
17      for (i = 0; i < ListeARemplacer.length(); i++) {
18          x, y = ListeARemplacer[i];
19          if (Data[x][y] == vc) {
20              // Recherche de l'élément contigu le plus à gauche devant être remplacé
21              G = x;
22              while (G >= 0 && Data[G][y] == vc) {
23                  Data[G][y] = vr;
24                  if (y > 0 && Data[G][y-1] == vc) {                                // Insère le voisin du haut si
25                      adéquat
26                          ListeARemplacer.Add(G, y-1);
27                  }
28                  if (y < Data.Height()-1 && Data[G][y+1] == vc) {        // Insère le voisin du bas si
29                      adéquat
30                          ListeARemplacer.Add(G, y+1);
31                  }
32                  G--;
33              }
34              if (G >= 0) {                                         // Recherche les voisins diagonaux à
35                  gauche
36                  if (y > 0 && Data[G][y-1] == vc) {                // Insère le voisin du haut si
37                      adéquat
38                          ListeARemplacer.Add(G, y-1);
39                  }
40                  if (y < Data.Height()-1 && Data[G][y+1] == vc) {    // Insère le voisin du bas si
41                      adéquat
42                          ListeARemplacer.Add(G, y+1);
43                  }
44              }
45
46              // Recherche de l'élément contigu le plus à droite devant être remplacé
47              D = x + 1;
48              while (D < Data.Width() && Data[D][y] == vc) {
49                  Data[D][y] = vr;
50                  if (y > 0 && Data[G][y-1] == vc) {                    // Insère le voisin du haut si
51                      adéquat
52                          ListeARemplacer.Add(D, y-1);
53                  }
54                  if (y < Data.Height()-1 && Data[G][y+1] == vc) {    // Insère le voisin du bas si
55                      adéquat
56                          ListeARemplacer.Add(D, y+1);
57                  }
58                  D++;
59              }
60              if (D < Data.Width()) {                               // Recherche les voisins diagonaux à
61                  droite
62                  if (y > 0 && Data[G][y-1] == vc) {                // Insère le voisin du haut si
63                      adéquat
64                          ListeARemplacer.Add(D, y-1);
65                  }
66                  if (y < Data.Height()-1 && Data[G][y+1] == vc) {    // Insère le voisin du bas si
67                      adéquat
68                          ListeARemplacer.Add(D, y+1);
69                  }
70              }
71      }
72  }
```

ANNEXE 7 : REMPLISSAGE DE TROUS D'UNE IMAGES BINAIRES

SOMMAIRE

Sur des images binaires (noir et blanc, 0 ou 1), il est fréquent de vouloir remplir les trous des objets identifiés. Cette opération permet entre autre d'identifier toute la région d'un objet dont le contour est déjà bien identifié.

CONCEPT

Cet algorithme est relativement simple et requiert seulement 4 opérations :

1. identification d'un pixel appartenant au fond de l'image;
2. remplissage à partir du fond de l'image;
3. inversion de l'image obtenue;
4. addition à l'image source.

Étape 1 – Identification d'un pixel appartenant au fond de l'image

Cette étape est nécessaire à l'algorithme suivant, le remplissage, qui se fait toujours à partir d'un point initial. C'est ce point que nous tentons de trouver ici. Aussi, puisque l'étape suivante tente de remplir le fond de l'image, il est impératif de trouver au moins un pixel appartenant au fond. Même si tous les pixels blancs appartiennent aux objets de la scène, il n'est pas certains que tous les pixels noirs appartiennent au fond. En fait, plusieurs pixels noirs appartiendront aux zones trouées des objets que nous désirons justement remplir.

Cette étape est définitivement la plus difficile de tout ce processus car elle nécessite de prendre une décision arbitraire. En effet, l'identification d'un pixel appartenant au fond de l'image peut être difficile à faire. Sans une analyse contextuelle de chaque pixel et de son lien dans l'espace, comment déterminer sa classe d'appartenance? Pour y arriver, on simplifiera le problème en posant l'hypothèse suivante :

- *Nos images ne présenteront jamais une grande quantité d'objets sur son contour et la dimension la plus grande des trous des objets est connue (nommé d).*

À partir de cette assumption, on pourra rechercher sur le contour, le premier pixel respectant ces conditions. C'est-à-dire, le premier pixel sur le contour où il existe d voisins contigus qui le précèdent. Cette recherche peut se faire en parcourant tous les pixels du contour dans le sens horaire à partir du point $(0, 0)$ de l'image jusqu'à ce que la condition soit respectée. On nommera le pixel identifié p correspondant à la coordonnée (x, y) .

Étape 2 – Remplissage du fond de l'image

Ici, vous devez simplement implémenter un algorithme de remplissage. Référez-vous à l'annexe sur le remplissage pour le détail de son implémentation. Considérant les exemples de remplissage donné, nous aurons un appel de fonction semblable à :

- `Remplissage(Image, x, y, 0, 1)`

On nommera l'image obtenu I_f .

Étape 3 – Inversion de l'image obtenu

L'inversion de l'image obtenue permet d'identifier tous les trous des objets. Cette inversion ce fait simplement par l'opérateur logique d'inversion.

On nommera cette image I_f' .

Étape 4 – Construction de l'image finale

Il suffit maintenant de produire l'image finale en fusionnant l'image source à l'image I_f' obtenu à l'étape précédente.

Cette fusion peut se faire simplement en additionnant chacune des images ou (de façon plus efficace) en appliquant l'opérateur *ou* logique bit à bit.

EXEMPLE

Image source couleur

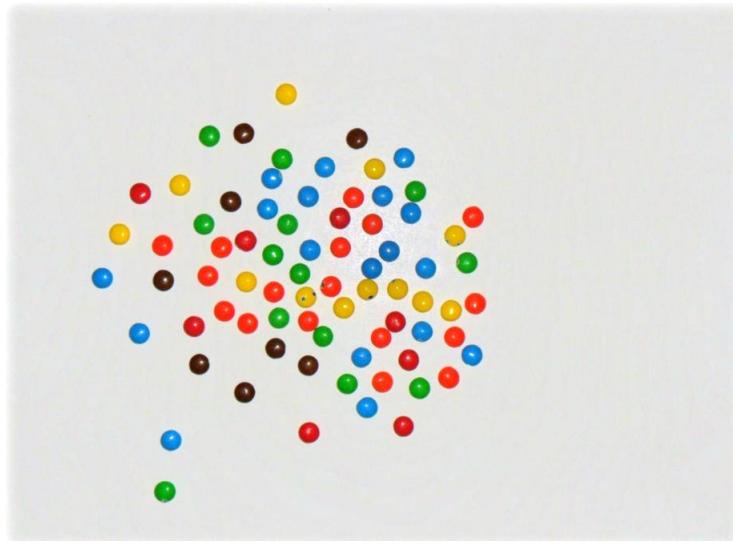


Image segmentée binaire

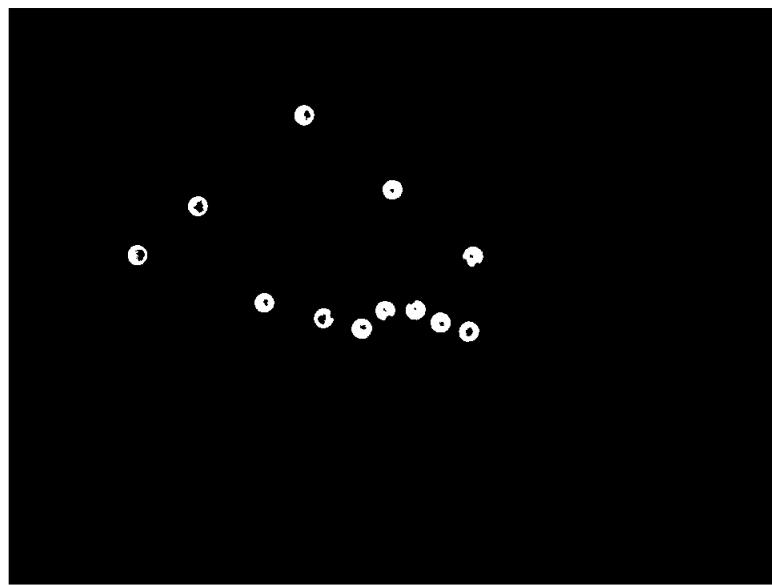


Image avec le fond rempli

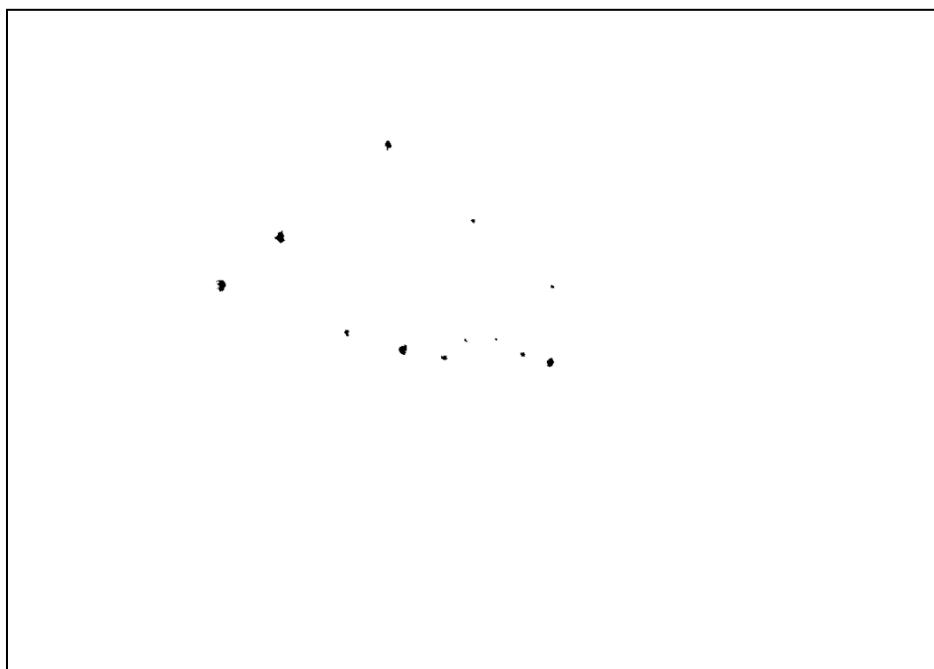


Image inversée

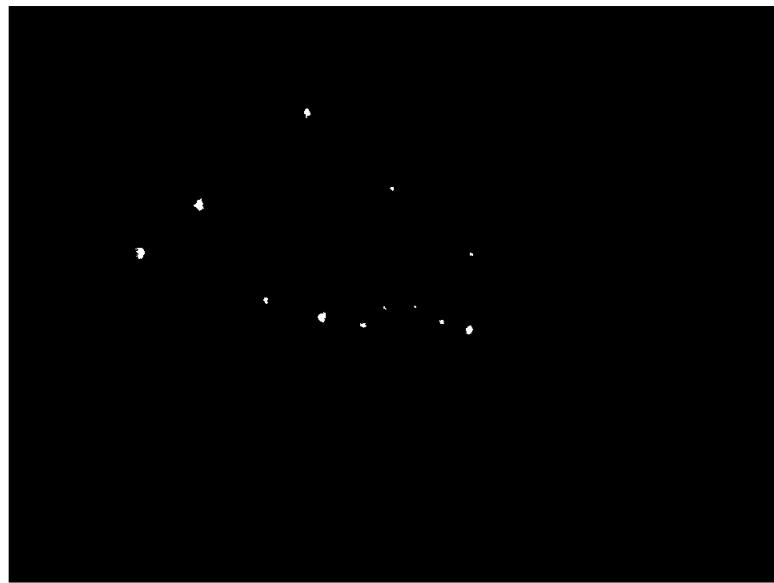
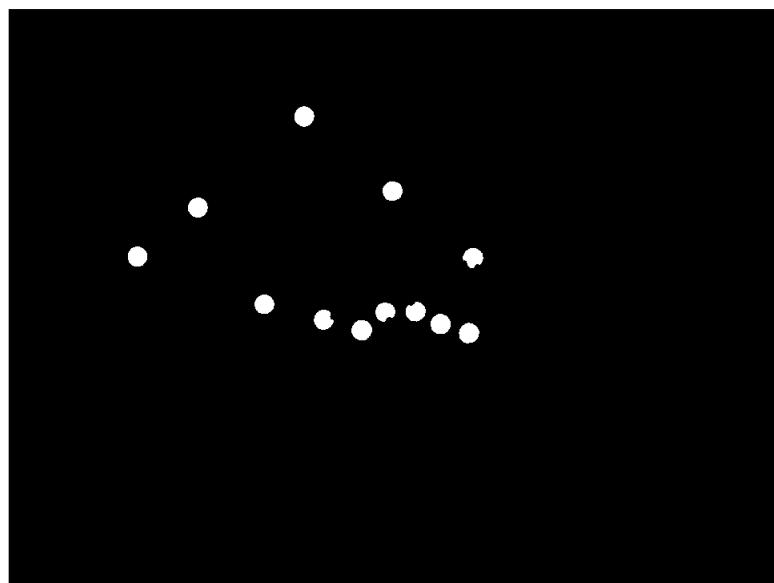


Image finale



ANNEXE 8 : ÉTIQUETAGE D'UNE IMAGE BINAIRE

SOMMAIRE

L'étiquetage d'une image consiste à identifier tous les pixels appartenant à chaque région bien définis et contiguë d'une image. Cette identification est très utile dans plusieurs algorithmes qui font l'analyse de ces régions.

Le résultat est une image où chaque pixels des différentes régions possèdent une étiquette qui lui propre. On nomme souvent ce type d'opération en anglais par du *Labeling*.

CONCEPT

Cet algorithme est relativement simple et permet d'identifier rapidement sans trop d'effort chaque région.

L'idée consiste à parcourir l'image et, pour chaque pixel appartenant à la valeur du signal, amorcer une procédure de remplissage avec un nouvel identificateur unique.

Voici un pseudo code de cet algorithme :

```
01 // Cet algorithme considère les paramètres suivants sur l'image d'entrée :
02 //   - Image binaire de type « integer »
03 //   - Le fond est à 0
04 //   - Les objets sont à 1
05 void Etiquetage(Image)
06
07     Etiquette = 2;
08
09     for (y = 0; y < Image.Height(); y++) {
10         for (x = 0; x < Image.Width(); x++) {
11             if (Image[x][y] == 1) {
12                 Remplissage(Image, x, y, 1, Etiquette)
13                 Etiquette++;
14             }
15         }
16     }
17 }
```

La sortie de cet algorithme donne une image où chaque région est bien définie et unique.

L'algorithme présenté en est un parmi plusieurs autres. Ce n'est pas l'algorithme le plus performant à cause de la nature particulière du remplissage. Néanmoins, il est très simple à utiliser, surtout si vous avez déjà une fonction de remplissage sous la main.

Si vous recherchez un algorithme plus performant, vous pouvez tenter d'implémenter l'algorithme d'étiquetage à deux passes présenté [ici](#).

Finalement et tel qu'expliqué dans l'annexe sur le remplissage, le remplissage peut se faire à 4 ou à 8 voisins selon les besoins de l'application.

EXEMPLE

Image source

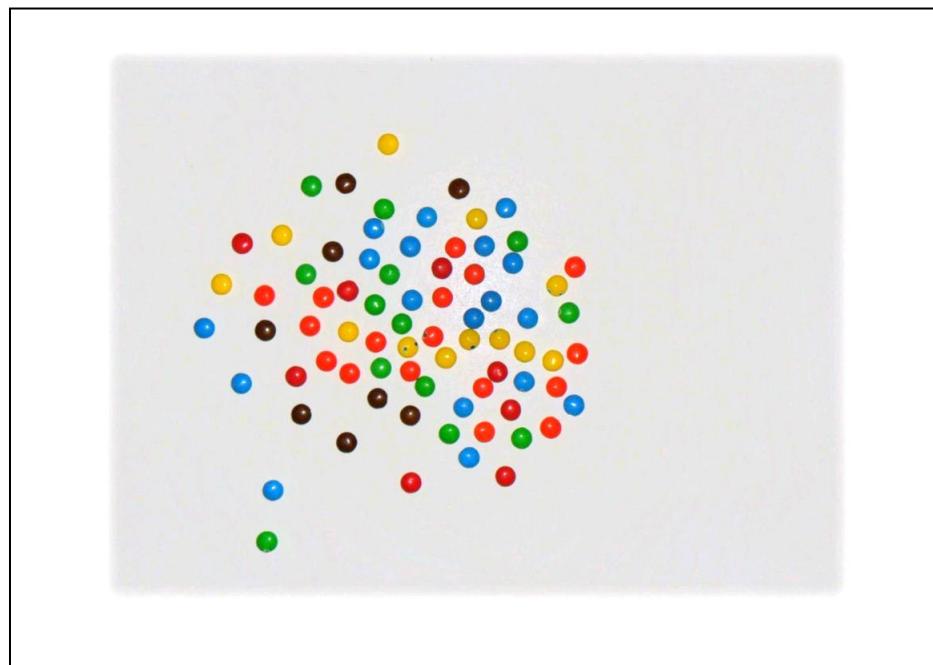


Image segmentée pour les friandises jaunes

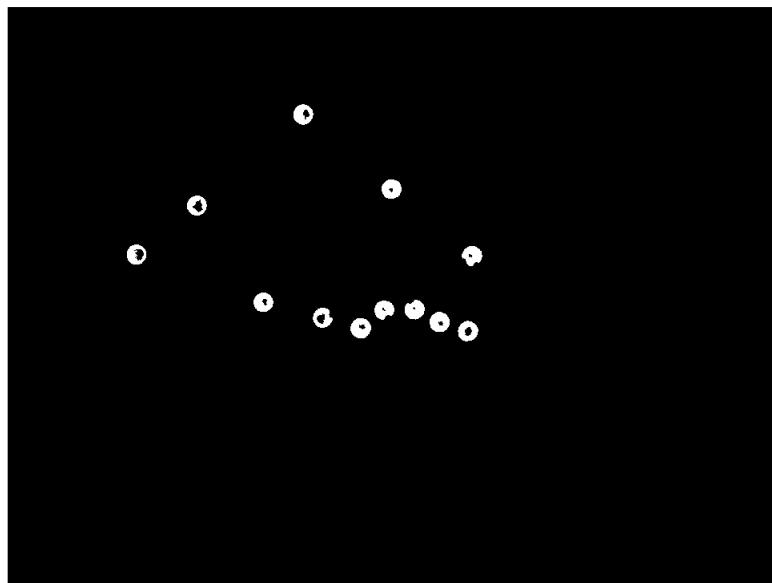
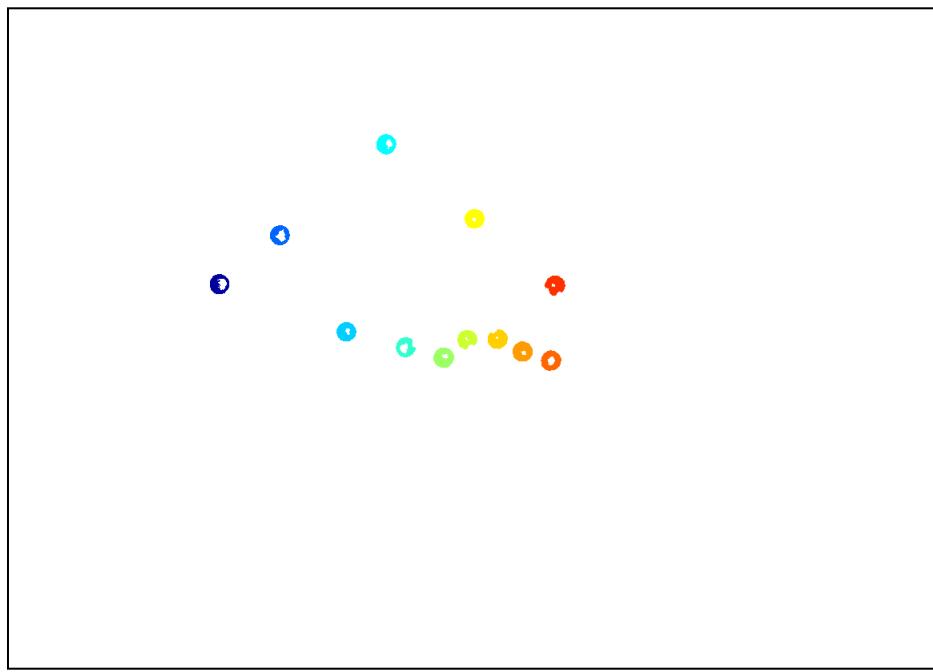


Image d'étiquette (l'exemple donné colore artificiellement les différentes étiquettes pour fin de visualisation seulement, ainsi le cyan correspond à l'étiquette 2, le jaune à l'étiquette 3, le bleu moyen à l'étiquette 4 et ainsi de suite)



ANNEXE 9 : EXTRACTION DE BLOBS

SOMMAIRE

L'extraction de blobs est une tâche fondamentale en vision artificielle. Elle permet d'identifier les formes de l'image en plus de les quantifier par divers descripteurs. Ces descripteurs, souvent appelés descripteurs de forme, sont très variés et peuvent représenter :

- des quantités très concrètes comme :
 - l'aire
 - le périmètre
 - l'intensité moyenne ou la couleur moyenne
- des données plus abstraites comme :
 - la complexité
 - des indicateurs de circularité
 - des indices de corrélations avec des formes prédefinies
- des quantités beaucoup plus abstraites comme :
 - les moments d'inerties de divers ordre
 - les moments de *Zernike*
 - des transformations d'ordre et d'espace

L'extraction de ces informations est un domaine très large en vision et il existe vraiment plusieurs approches permettant l'extraction de ces diverses informations. L'approche présentée ici présente plusieurs aspects intéressants :

- seulement le contour des blobs est parcouru et non toute la partie intérieure
- malgré tout, seulement à partir du parcours extérieur, il sera possible d'extraire ces informations utiles (nous nous limiterons aux descripteurs identifiés par un astérisque) :
 - * périmètre;
 - * aire;
 - * complexité;
 - moment d'inertie d'ordre varié et tous les descripteurs qui peuvent en découler comme les orientations principales et secondaires;
 - etc.
- Malgré tout ces aspects intéressants, les descripteurs possibles en ne parcourant que le contour sont ceux qui décrivent un objet d'intensité uniforme (comme sur une image binaire et non comme ceux sur image en niveaux de gris ou même couleur).

Finalement, même si il est possible de faire la tâche d'extraction directement à partir d'une image couleur, nous nous limiterons à une image déjà étiquetée. Ce type d'image rend les algorithmes sous-jacents plus simples et plus accessibles.

AVANT D'ALLER PLUS LOIN

Il est important de comprendre la nécessité et la pertinence des descripteurs de forme. Il serviront à prendre les décisions finales à savoir si les objets de l'image sont pertinents ou non selon vos critères.

Par exemple, il est facile de comprendre que le périmètre ou l'aire peuvent être de bons descripteurs de forme pour discriminer des objets dont le contour et la superficie sont très différents. Par exemple, si vous avez un objet dont la surface varie entre 250 et 350 pixels, il est clair qu'un objet de 135 pixels est trop petit et qu'il sera rejeté.

Néanmoins, malgré la pertinence de ce type de descripteurs de forme, certains d'entre eux, selon l'application, ont une limitation importante : c'est descripteurs ne sont pas invariants. C'est-à-dire que selon le type de prise d'image ils peuvent varier pour un même objet.

Généralement, on fait référence au fait qu'un descripteur soit invariant soit à la translation, la rotation et la mise à l'échelle. Idéalement, aux trois!

L'invariance à la translation représente le fait qu'un descripteur donnera toujours la même valeur pour un objet peu importe sa position sur l'image. Par exemple, le centroïde d'un objet n'est pas invariant en translation alors que l'aire est invariant en translation (ou presque).

L'invariance à la rotation représente le fait qu'un descripteur donnera toujours la même valeur pour un objet peu importe son orientation sur l'image. Par exemple, la mesure de l'axe principale n'est pas invariante à la rotation alors que le périmètre lui l'est (ou presque).

L'invariance à la mise à l'échelle représente le fait qu'un descripteur donnera toujours la même valeur pour un objet peu importe la taille qu'il a dans l'image. Par exemple, lorsque la caméra s'éloigne un peu du plan d'observation. Les descripteurs tels que le périmètre et la aire ne sont pas invariants à la mise à l'échelle alors que la complexité l'est.

Qu'est-ce que la complexité? C'est un descripteur de forme très simple qui consiste à décrire une forme tout en étant invariant à la translation, rotation et mise à l'échelle à la fois. Ce descripteur se calcule simplement par :

$$c = 1 - \frac{4\pi A}{p^2}$$

où : c est la complexité
 A est la aire
 p est le périmètre

La complexité, telle que normalisée ici, donne virtuellement toujours une valeur comprise entre 0 et 1 où 0 représente un cercle parfait. Donc, plus la forme est complexe, plus la valeur tend vers 1.

Finalement, l'usage de ces descripteurs vous permettra de prendre les décisions éclairées sur la classe d'appartenance de chaque élément. Aussi, vous pourrez utiliser les pixels identifiés sur le contour pour colorer artificiellement l'image source afin d'illustrer quels objets appartient à quelles classes.

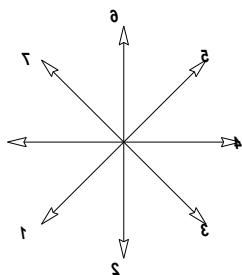
CONCEPT

En considérant une image étiquetée, la tâche d'extraction de blob reste relativement simple. Il suffit de parcourir l'image pour identifier le premier point de contour de chaque objet inclus.

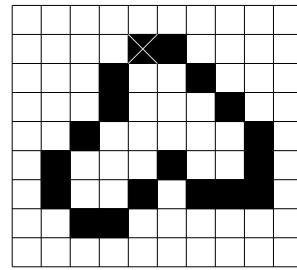
Chacun des points identifiés appartenant inévitablement au contour, on amorce la procédure d'extraction du contour. Lorsque le contour est extrait, on effectue les calculs pour déterminer la valeur des descripteurs de forme désirés.

L'extraction du contour proposée ici utilise une méthode basée sur ce qu'on appelle le code de Freeman. Cette méthode d'extraction permet d'identifier facilement le contour en ne conservant que la direction du prochain élément. Il existe deux implémentations de cet algorithme où le contour est décrit à partir de 4 ou 8 directions. Nous ferons ici une implémentation à 8 directions. Les 8 directions indiquent les 8 voisins immédiats du pixel analysé : 0 à l'ouest, 1 au sud ouest, 2 au sud, 3 au sud est, 4 à l'est, 5 au nord est, 6 au nord et 7 au nord ouest.

Voici un exemple concret de cette approche :



Représentation des 8 directions du code de Freeman



Représentation du contour d'une forme quelconque.

Ainsi l'image illustrée ci-haut à droite possèdera le code de Freeman suivant, le point d'origine étant celui identifié par un X se trouvant en (4, 1) :

$$(4, 1) : 1-2-1-1-2-3-4-5-5-3-4-4-6-6-7-7-7-0$$

Un fait intéressant, les directions établies pour décrire le code de Freeman (de 0 à 7) et la description du contour de la forme sont définis de façon antihoraire.

La gestion des effets de bord peut être fastidieuses et ralentir notablement la performance de l'algorithme. Pour ces raisons et pour simplifier le problème, on évitera la gestion des effets de bord simplement en mettant la bordure de l'image à 0. Ainsi, on parcours tout le contour pour le mettre à la valeur de fond.

ALGORITHME

Paramètres d'entrée

- **Label** L'image d'entrée dont chaque région est déjà étiquetée

Résultats de sortie

- Une liste d'objets possédant tous ces paramètres pour chacun des objets :
 - Point de départ (nommé px, py)
 - Contour de l'objet identifié (nommé Contour[i] où i inclue tous les éléments de contour)
 - Périmètre (nommé Perimetre)
 - Aire (nommé Aire)
 - Complexité (nommé Complexité)

Algorithme général

L'algorithme présenté est très efficace car il permet de calculer plusieurs descripteurs à même le contour. Notamment, il utilise plusieurs propriétés propres au suivi de contour. À titre indicatif. Voici la définition mathématique utilisée pour le calcul de l'aire :

$$A = \sum_{i=1}^{n_F} S_{F_{i-1}, F_i} (y_i + O_{F_{i-1}, F_i}) + U_{F_{i-1}, F_i}$$

où : A est la aire

F_i est le $i^{\text{ème}}$ élément du code de Freeman

y_i est la hauteur du $i^{\text{ème}}$ élément du contour

S , O et U sont les matrices de références nécessaires au suivi de contour. Elles sont définies par :

$$S = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 & -1 & -1 \\ -1 & -1 & 0 & 0 & 0 & 0 & -1 & -1 \\ -1 & -1 & 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \quad O = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 & -1 & -1 \\ -1 & -1 & 0 & 0 & 0 & 0 & -1 & -1 \\ -1 & -1 & 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \quad U = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

L'algorithme proposé commence par chercher le premier éléments de contour pour chaque région étiquetée. Ensuite, pour chaque région, on extrait le contour. Finalement, lorsque tous les contours sont extraits, le calcul de tous les descripteurs de forme est fait.

Cet algorithme peut facilement être amélioré en modifiant l'algorithme d'étiquetage de façon telle à ce qu'il garde en mémoire les premières coordonnées identifiées pour les régions étiquetées. Ainsi, la première passe de recherche pourrait être évitée et le processus général en profiterait.

```

01 /* On défini une table de référence (souvent appelée "Lookup table") permettant d'obtenir
02 rapidement certaines valeurs spécifiques */
03 const float pi      = 3,1415926535897932384626433832795f;
04 const float DiagUnit = sqrt(2.0f);
05 const float Perim[] = { 1, DiagUnit, 1, DiagUnit, 1, DiagUnit, 1, DiagUnit };
06 const int RefX =     { -1, -1, 0, 1, 1, 1, 0, -1 };
07 const int RefY =     { 0, 1, 1, 1, 0, -1, -1, -1 };
08 const int O[][] =    { { -1, -1, 0, 0, 0, 0, -1, -1 },
09                      { -1, -1, 0, 0, 0, 0, -1, -1 },
10                      { -1, -1, 0, 0, 0, 0, -1, -1 },
11                      { 0, 0, 0, 0, 0, 0, 0, 0 },
12                      { 0, 0, 0, 0, 0, 0, 0, 0 },
13                      { 0, 0, 0, 0, 0, 0, 0, 0 },
14                      { 0, 0, 0, 0, 0, 0, 0, 0 },
15                      { -1, -1, 0, 0, 0, 0, -1, -1 } };
16
17 const int S[][] =    { { -1, -1, 0, 0, 0, 0, -1, -1 },
18                      { -1, -1, 0, 0, 0, 0, -1, -1 },
19                      { -1, -1, 0, 0, 0, 0, -1, -1 },
20                      { 0, 0, 1, 1, 1, 1, 0, 0 },
21                      { 0, 0, 1, 1, 1, 1, 0, 0 },
22                      { 0, 0, 1, 1, 1, 1, 0, 0 },
23                      { 0, 0, 1, 1, 1, 1, 0, 0 },
24                      { -1, -1, 0, 0, 0, 0, -1, -1 } };
25
26 const int U[][] =    { { 0, 0, 0, 0, 1, 1, 0, 0 },
27                      { 0, 0, 0, 0, 0, 1, 0, 0 },
28                      { 0, 0, 0, 0, 0, 0, 0, 0 },
29                      { 1, 1, 0, 0, 0, 0, 0, 1 },
30                      { 1, 1, 0, 0, 0, 0, 0, 0 },
31                      { 0, 1, 0, 0, 0, 0, 0, 0 },
32                      { 0, 0, 0, 0, 0, 0, 0, 0 },
33                      { 0, 0, 0, 1, 1, 1, 0, 0 } };
34
35
36 ListeObjets ExtractionBlob(Image)
37     MettreContourImageAZero(Image);
38
39 // Parcours de l'image à la recherche de chaque objet étiqueté pour en extraire le parcours
40 Etiquette = 2;
41 for (y = 0; y < Image.Height(); y++) {
42     for (x = 0; x < Image.Width(); x++) {
43         if (Image[x][y] == Etiquette) {
44             ListeObjets[Etiquette-2].px = x;
45             ListeObjets[Etiquette-2].py = y;
46             Contour(Image, x, y, ListeObjets[Etiquette-2].Contour);
47             Etiquette++;
48         }
49     }
50 }
51 // Pour chaque Objet extrait, on calcule les descripteurs de forme
52 for (i = 0; i < ListeObjets.length(); i++) {
53     ListeObjets[i].Perimetre = ListeObjets[i].Aire = 0;                                // Initialisation
54     y = ListeObjets[i].py;                                                               // Valeur de
55     l'élevation
56     FreemanPrecedant = ListeObjets[i].Contour[ListeObjets[i].Contour.length()-1];
57     // Si ce n'est qu'un seul pixel, il n'a pas de périmètre
58     if (ListeObjets[i].Contour.length() == 0) {
59         ListeObjets[i].Perimetre = 1.0f / (4.0f*pi);
60         ListeObjets[i].Aire = 1;
61     } else {
62         // On fait le calcul pour tous les éléments de contour déjà identifié
63         for (c = 0; c < ListeObjets[i].Contour.length(); c++) {
64             FreemanCourant = ListeObjets[i].Contour[c];                                // Extrait le code
65             Freeman
66             s = S[FreemanPrecedant][FreemanCourant];
67             o = O[FreemanPrecedant][FreemanCourant];
68             u = U[FreemanPrecedant][FreemanCourant];
69
70             ListeObjets[i].Perimetre += Perim[FreemanCourant];                         // Mise à jour du
71             pérимètre
72             ListeObjets[i].Aire += s*(y + o) + u;                                         // Mise à jour de
73             l'aire
74
75             // Mise à jour des variables de suivi
76             FreemanPrecedant = FreemanCourant;
77             y += RefY[FreemanCourant];
78         }
79     }
80     ListeObjets[i].Complexite = 1.0f - (4.0f*pi*ListeObjets[i].Aire) / ...           (ListeObjets[i].Perimetre*ListeObjets[i].Perimetre);
81

```

```

        }
        return ListeObjets;
    }
}

```

Algorithme d'extraction du contour

```

01 void Contour(Image, x, y, Etiquette, C);
02
03     // Initialisation des variables de controle
04     C.clear();
05     Direction = 0;
06     XCourant = x;
07     YCourant = y;
08
09     // S'assure qu'il y a au moins 1 candidat possible (les 4 autres ont implicitement été
10     validés)
11     if ( Image[x - 1][y + 1] != Etiquette &&
12         Image[x      ][y + 1] != Etiquette &&
13         Image[x + 1][y + 1] != Etiquette &&
14         Image[x + 1][y      ] != Etiquette) {
15         return;
16     }

    // Parcours le contour jusqu'au retour
    do {
        Direction = ProchaineDirection(Image, XCourant, YCourant, Direction, Etiquette);
        C.add(Direction);
        XCourant = x + RefX[Direction];
        YCourant = y + RefY[Direction];

    } while (((XCourant != x && YCourant != y) || Direction != C[0]) && C.length() < MaxNPixel) {

    // Les variables NouvDirection, NouvX et NouvY sont des références qui seront modifiées
    // Cette fonction retourne :
    //     - vrai s'il existe au moins un voisin permettant de définir un contour
    //     - faux si aucun voisin ne permet de créer un contour
    int ProchaineDirection(Image, x, y, DirectionPrecedente, Etiquette)

        i = 0;
        DirectionCourante = DirectionPrecedente + 6 % 8;
        XCourant = x + RefX[DirectionCourante];
        YCourant = y + RefY[DirectionCourante];

        while (Image[XCourant][YCourant] != Etiquette && i < 8) {
            DirectionCourante = DirectionCourante + 1 % 8;
            XCourant = x + RefX[DirectionCourante];
            YCourant = y + RefY[DirectionCourante];
            i++;
        }

        if (i < 8) {
            return DirectionCourante;
        } else {
            return -1;
        }
    }
}

```