

# Cahier des charges version 1.0 - 15/11/2021

## Implémentation d'un système d'aide à la vision sur des lunettes de réalité virtuelle

- **Client** : Olivier Bodini
  - **Équipe de suivi** : Thierry Hamon et Sophie Toulouse
  - **Groupe** : YACOUBI Mohamed Ali, HAMELIN Flavien, KASSOUS Safa,  
CHERIF Farah et AMMARI Saad
-



## Sommaire

<b>1. Introduction</b>	<b>2</b>
<b>2. Contexte général du projet</b>	<b>2</b>
<b>3. Analyse du contexte</b>	<b>2</b>
3.1. Spécification des besoins fonctionnels	3
3.2. Spécification des besoins non fonctionnels	4
3.3. Analyse de l'existant	4
<b>4. Solution proposée</b>	<b>5</b>
4.1. Architecture de la solution proposée	6
4.2. Environnement de travail	6
4.2.1. Environnement matériel	6
4.2.2. Environnement logiciel	7
4.3. Ressources humaines	8
<b>5. Spécification de la vue conceptuelle</b>	<b>9</b>
<b>6. Techniques de détection de contour</b>	<b>10</b>
6.1. Masque de Sobel	10
6.2. Les contours actifs (snakes)	12
<b>7. Techniques de segmentations</b>	<b>13</b>
7.1. Split and merge	14
<b>8. Organisation et planning</b>	<b>15</b>

## 1. Introduction

Dans le cadre de notre formation en informatique à l'école d'ingénieurs Sup Galilée, et en particulier, dans le cadre de notre cours de gestion de projet, nous sommes amenés à réaliser un système d'aide à la vision sur des lunettes de réalité virtuelle.

Au début, nous présentons le cadre général du projet. Puis, nous constituons une analyse du contexte en présentant la phase de détermination des besoins fonctionnels et non fonctionnels que notre système doit accomplir. Ensuite, nous exposons une étude et une analyse de l'existant. Finalement, nous nous consacrons à proposer notre solution ainsi que l'environnement matériel et logiciel qui sera utilisé pour réaliser notre projet.


Ce document sert à suivre une bonne méthodologie et à clarifier les différentes tâches de notre système tout en assurant une bonne organisation au sein de l'équipe et une transversalité et transparence de nos actions avec notre client.

## 2. Contexte général du projet

La dégénérescence maculaire liée à l'âge (DMLA) est la première cause de handicap visuel chez les personnes de plus de 50 ans. La fréquence de cette maladie augmente largement avec l'âge. Elle touche 20% de la population de plus de 65 ans ce qui en fait donc un problème majeur en santé publique. Non traitée, cette maladie peut aboutir à la destruction irréversible de la zone de vision centrale. Il existe plusieurs thérapies qui permettent de réduire la propagation des lésions. Mais de nos jours, il n'existe pas une thérapie qui permet de reconstruire les zones lésées. C'est dans ce contexte que notre client a décidé de réaliser un système d'aide à la vision sur des lunettes de réalité virtuelle qui permettra d'améliorer le quotidien des personnes atteintes de DMLA en diminuant leur handicap.

## 3. Analyse du contexte

Ce projet vise à concevoir et à mettre en œuvre un système connecté d'aide à la vision sur des lunettes de réalité virtuelle qui assure la détection des différentes zones lésées de la rétine, afin d'extraire toutes les informations utiles du monde réel pour



construire une image cohérente et complète à l'utilisateur. En effet, le principe est de créer des points lumineux fugaces en moins d'une seconde à différent endroit d'une façon aléatoire (mais intelligemment) sur les verres des lunettes, le patient doit à chaque fois qu'il voit le point, le signaler en utilisant un téléphone portable connecté aux lunettes.

### 3.1. Spécification des besoins fonctionnels

Notre système doit répondre à des fonctionnalités qui se résument dans les points suivants :

- **Analyser la première partie déjà prête** : cela consiste à analyser les tâches suivantes:
  - **Afficher en temps réel des points lumineux dans le champ visuel du patient** : l'unité de traitement des lunettes projette en temps réel des points lumineux de différentes positions, en commençant par le centre, car le patient en général ne voit plus au centre de son champ de vision (la taille de la tâche dépend de la sévérité de la maladie).
  - **Assurer que l'utilisateur soit capable de transmettre ses réponses en temps réel aux lunettes à travers un téléphone connecté** : l'utilisateur appuie sur un bouton sur son téléphone s'il voit le point lumineux, cela permet d'analyser les zones lésées du patient. En effet, nous allons exploiter les zones non lésées, pour déduire ce que le patient ne peut pas voir.
  - **Faire en sorte que l'application crée une cartographie des zones lésées grâce au réponse de l'utilisateur** : l'unité de traitement des lunettes crée et sauvegarde la cartographie dans le stockage système.
- **Proposer un système de traitement d'image afin de détailler les différentes informations de l'image**: l'unité de traitement applique au début, des filtres de détection de contour à l'image capturée pour la rendre plus lisible. Ensuite, retravailler l'image en la déformant et en faisant en sorte que le maximum d'information soit visible sur les zones saines de l'œil.

### 3.2. Spécification des besoins non fonctionnels

Les besoins non fonctionnels résument les exigences qui ne sont pas liées spécifiquement au comportement du système mais plutôt liées à la performance et la fiabilité du système. Ce sont les contraintes internes et externes du système.

Notre système doit répondre aux exigences suivantes :

- **La consommation d'énergie** : la consommation d'énergie du système doit être faible.
- **La performance** : le temps d'exécution du système doit être minimal.
- **Le fonctionnement en temps réel** : le système doit garantir une réponse en temps réel, en particulier lors de la projection des points lumineux, de la communication avec le téléphone et lors du traitement des images.
- **La disponibilité** : les différentes informations cartographiées doivent être disponibles à la demande de l'utilisateur.
- **L'utilisabilité** : la simplicité et la clarté des interfaces est demandée.

### 3.3. Analyse de l'existant

Dans cette section nous allons recueillir les informations des différentes parties existantes tels que l'environnement technique et logiciel afin de proposer notre solution. Le système d'exploitation open source que nous allons traiter est Moverio OS des lunettes de réalité virtuelle Moverio BT 300.

Le système assure déjà les fonctionnalités suivantes :

#### Générer une cartographie

- **Affichage des points lumineux** : l'unité de traitement fait appel à une primitive permettant de trouver une zone non traitée, une fois cette zone récupérée, l'application affiche un point sur l'écran.
- **Réponse du patient** : l'unité de traitement reste en attente d'une réponse de l'utilisateur pendant un court laps de temps, dès que l'utilisateur répond en tapant sur oui, l'unité de traitement marque la zone comme visible jusqu'au prochain test.
- **Mémorisation des réponses de l'utilisateur** : l'unité de traitement enregistre, dans une structure de données, les coordonnées du point et si celui-ci est visible ou non par l'utilisateur.

- Délimitation des zones lésées : l'unité de traitement utilise la structure de données créée à travers les réponses de l'utilisateur, un algorithme de détection de contour délimite les zones lésées à partir des informations contenues dans la structure de données et ensuite l'unité de traitement génère la cartographie.
- Sauvegarde de la cartographie : sauvegarde de la cartographie de l'utilisateur dans l'espace de stockage des lunettes

### Détecter les zones lésées

En ce qui concerne la détection des zones lésées, plusieurs techniques ont été mises en œuvre.

- Watershed : Algorithme qui permet de partitionner les pixels en un ensemble de régions connexes séparées par un contour fermé.
- DICOM (Digital imaging and communications in medicine) : Sauvegarder le sous-format de fichier DICOM qui est un standard pour la gestion informatique des données issues de l'imagerie médicale. Indépendant des technologies (scanner, IRM, etc.)
- Amsler Grid : C'est une grille de lignes horizontales et verticales, utilisée pour évaluer la partie centrale du champ visuel du patient.

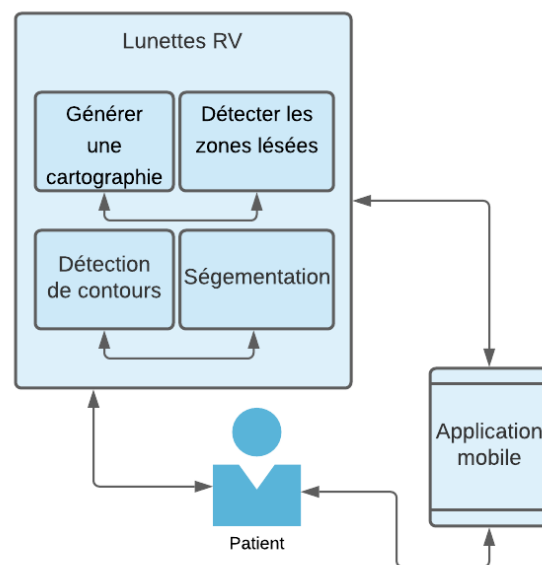
## 4. Solution proposée

Notre solution consiste à traiter les images issues de la caméra des lunettes afin d'aider les malvoyants en appliquant en premier temps des filtres à l'image pour la rendre plus "lisible" à travers plusieurs algorithmes. En effet pour la rendre "plus lisible" nous pouvons augmenter l'image avec la délimitation des objets soulignés. Ensuite, nous pourrions complètement retravailler l'image en la déformant et en faisant en sorte que le maximum d'information soit visible sur les zones saines de l'œil. Nous présentons un système qui répond aux spécifications mentionnées précédemment.

## 4.1. Architecture de la solution proposée

Nous présentons un système qui répond aux spécifications mentionnées précédemment. Ce système renferme les composants principaux suivants :

Des lunettes de réalité virtuelle qui permettent de générer une cartographie, détecter les zones lésées, les contours, et faire une segmentation pour bien mettre en lumière les objets. Les lunettes doivent communiquer avec une application Android pour achever ces différentes tâches.



## 4.2. Environnement de travail

Cette section fera l'objet d'une analyse des outils de développement utilisés, et les composants qui ont servi à réaliser notre projet.

### 4.2.1. Environnement matériel

Nous commençons cette partie par la définition de l'ensemble des équipements matériels choisis pour la réalisation de notre système.

**Epson Moverio BT-300:** représente la troisième génération de lunettes intelligentes à réalité augmentée de Epson. Elle est jusqu'à 20 % plus légère que les lunettes BT-200. Cependant, malgré la réduction de poids, de nombreux autres aspects des lunettes se

sont améliorés, notamment l'écran OLED à base de silicium et le nouveau processeur Atom X5.



Cinq ordinateurs portables pour le développement :

Propriétaire	Processeur	Carte graphique	Mémoire	SSD (Oui/Non)
Mohamed Ali YACOUBI	Intel i5-5200U @2.20 GHz	AMD Radeon R5 M330 - 2 Go	12 Go	Oui
Flavien HAMELIN	Intel i7-6700HQ @3.5 GHz	NVIDIA GeForce GTX 960M	16 Go	Oui
Safa KASSOUS	Intel i7-7700HQ @2.80GHz	NVIDIA GeForce GTX	12 Go	Non
Farah CHERIF	Intel i7-10750H @2.60 GHz	NVIDIA GeForce GTX	16 Go	Oui
Saad AMMARI	Intel i7-8700K @3.70 GHz	NVIDIA GeForce GTX 1050 Ti	12 Go	Non

#### 4.2.2. Environnement logiciel

Après avoir présenté notre choix du matériel à utiliser, cette partie va être consacrée à la description des ressources logicielles utilisées tout le long de notre projet.



- JAVA : Java est un langage de programmation orienté objet de haut niveau, basé sur des classes, conçu pour avoir le moins de dépendances d'implémentation possible.
- Android : Android est un système d'exploitation mobile basé sur une version modifiée du noyau Linux et d'autres logiciels open source, conçu principalement pour les appareils mobiles à écran tactile tels que les smartphones et les tablettes.
- XML : c'est un langage de balisage qui définit un ensemble de règles pour l'encodage de documents dans un format à la fois lisible par l'homme et par machine.
- Android Studio: c'est l'environnement de développement intégré officiel pour le système d'exploitation Android de Google, construit sur le logiciel IntelliJ IDEA de JetBrains et conçu spécifiquement pour le développement Android.
- OpenCV : OpenCV est une bibliothèque de fonctions de programmation principalement destinées à la vision par ordinateur en temps réel.

### 4.3. Ressources humaines

Le tableau ci-dessous présente notre équipe de développement :

Nom et prénom	Role
Mohamed Ali YACOUBI	Développeur
Flavien HAMELIN	Développeur et chef d'équipe
Saad AMMARI	Développeur
Safa KASSOUS	Développeur
Farah CHERIF	Développeur

## 5. Spécification de la vue conceptuelle

Les cas d'utilisation de notre projet sont classés par ordre de priorité :

### 1- Générer une cartographie :

- Afficher un point lumineux.
- Réponse utilisateur.
- Mémoriser les réponses de l'utilisateur.
- Délimiter les zones lésées.
- Sauvegarder la cartographie.

### 2- Vision en temps réel :

- Prendre une photo.
- Déformer l'image.
- Afficher l'image déformée.

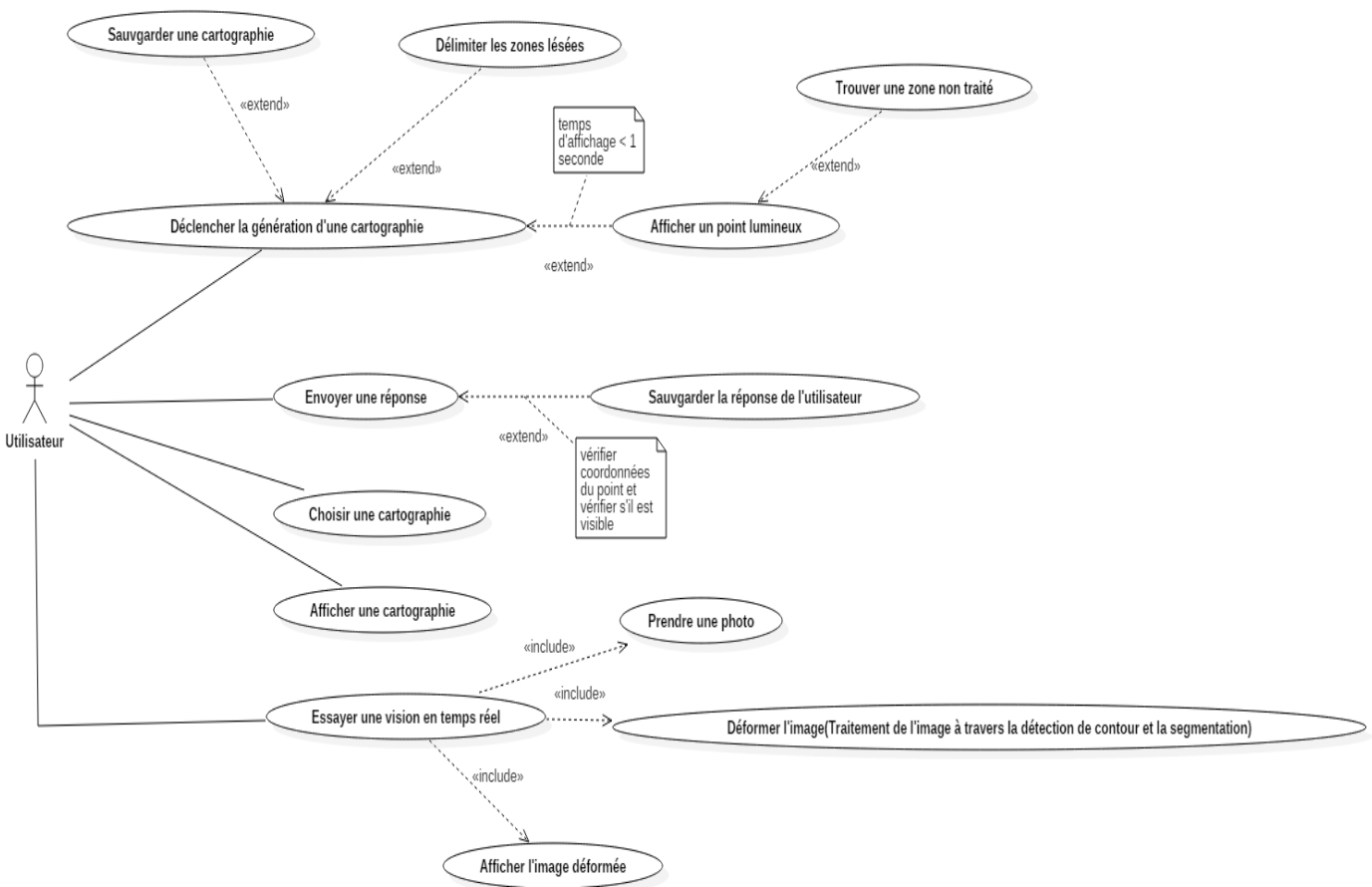
### 3- Choisir une cartographie :

- Afficher la cartographie.

L'acteur principal de notre application est :

- L'utilisateur : c'est le déclencheur de notre application. Il voit à travers la caméra des lunettes pour avoir une vision corrigée. De plus, il peut envoyer des réponses lorsqu'il voit un point lumineux afin de générer par la suite une cartographie qui sera affichée. Et finalement, il peut choisir une autre cartographie.

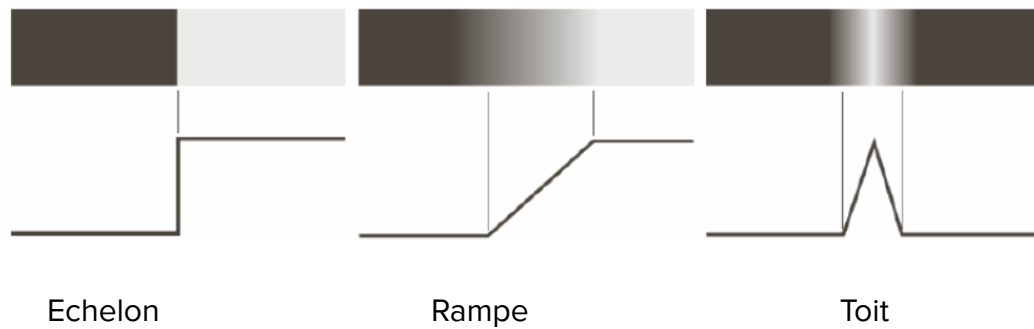
La figure suivante montre notre diagramme de cas d'utilisation global :



## 6. Techniques de détection de contour

Un but principal de notre système est le fait d'augmenter l'image avec la délimitation des objets soulignée. Dans cette section, nous présentons, en premier lieu, les techniques que nous comptons utiliser pour la détection des contours.

Un contour peut approximativement être défini comme une zone de l'image où l'intensité des pixels change brusquement. Cette discontinuité dans l'image est le passage d'un niveau de gris à un autre, de manière plus ou moins rapide, donnant lieu à des interprétations d'échelon, de rampe, de toit ou de ligne comme le montre la figure suivante.



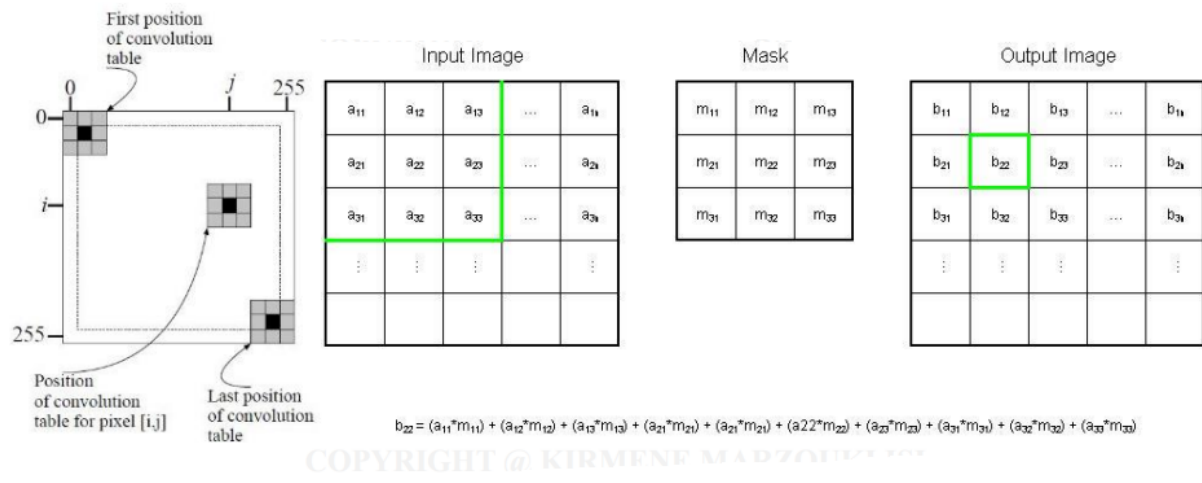
Par définition, un contour est la frontière qui sépare deux objets dans une image, c'est-à-dire une discontinuité de l'image.

### 6.1. Masque de Sobel

Le masque de Sobel mesure le gradient d'une image 2D. Typiquement, il est utilisé pour trouver une approximation du gradient en tout point d'une image au niveau de gris. Une paire de masques à convolution, l'un estimant le gradient suivant l'axe-x et l'autre selon l'axe-y.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = S_x \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = S_y$$

$S_x$  et  $S_y$  doivent être calculés pour chaque pixel de l'image. Tous les pixels auront une réponse aux modèles, mais seuls de très grandes réponses correspondront à des contours.



En effet, Sy détecte les contours verticaux et Sx détecte les contours horizontaux.



Image originale



Application de Sx



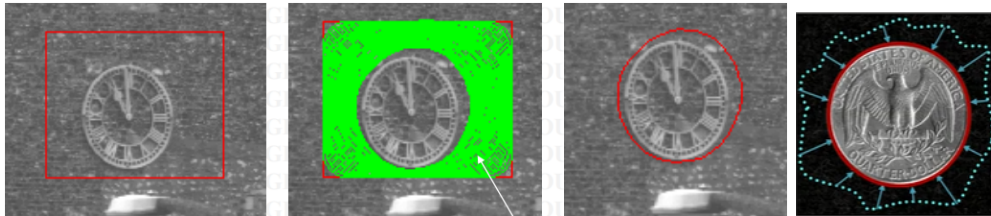
Application de Sy



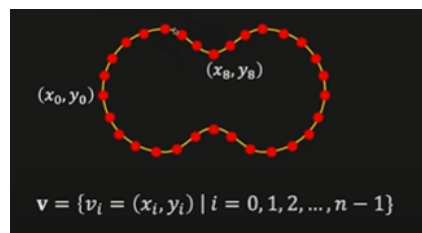
L'image S résultante avec  $S = \sqrt{Sx^2 + Sy^2}$

## 6.2. Les contours actifs (snakes)

Utiliser des courbes déformables qui sont “attirées” par les formes recherchées dans l'image. La figure suivante montre que le “snake” se contracte et s'adapte à la forme de l'horloge, la même chose pour la pièce de monnaie.



Les contours actifs sont définis par une courbe continue, fermée ou non, à extrémités fixes ou non. Ils se déforment (analogie aux serpents) à partir d'une position initiale située près de l'objet d'intérêt. Ils sont basés sur une minimisation d'une fonction d'énergie. Donc un contour est une liste de nœuds 2D connectés par des lignes droites de longueurs bien déterminées.



L'énergie des contours actifs est définie par cette formule :

$$E_{snakes} = \alpha . E_{interne} + \beta . E_{externe}$$

$\alpha, \beta$ : contrôlent l'influence de l'élasticité et de la fluidité

**Energie interne** : L'énergie interne gère la cohérence de la courbure. Elle maintient la cohésion des points et la raideur de la courbure. Chaque point va se déplacer au milieu de ses voisins : contraction (lissage) du snake, tout en maintenant une répartition homogène des espaces entre les points de contrôle.

**Energie externe** : L'énergie externe est liée à l'image. Elle correspond à l'adéquation aux données. Cette énergie externe prend en compte les caractéristiques de l'image. Rappelons ici que ce sont les contours de formes qui sont recherchés, donc les points de fort gradient ou des points ayant une propriété de position par rapport à une couleur donnée.

**Exemples d'utilisation :**



## 7. Techniques de segmentations

Pour faire en sorte de complètement retravailler l'image en la déformant et en assurant que le maximum d'information soit visible sur les zones saines de l'œil, nous proposons de faire une segmentation. La segmentation vise à diviser l'image en morceaux, ces morceaux correspondent aux objets de l'image. La segmentation est normalement basée sur les discontinuités : contours, les changements abruptes, et les frontières entre régions. La segmentation est le découpage d'une image en différentes régions et/ou contours.

Il existe plusieurs approches:

- Approches GLOBALES: Histogrammes (seuillage)
- Approches LOCALES: Region growing
- Approches HYBRIDES: Split & Merge, CSC

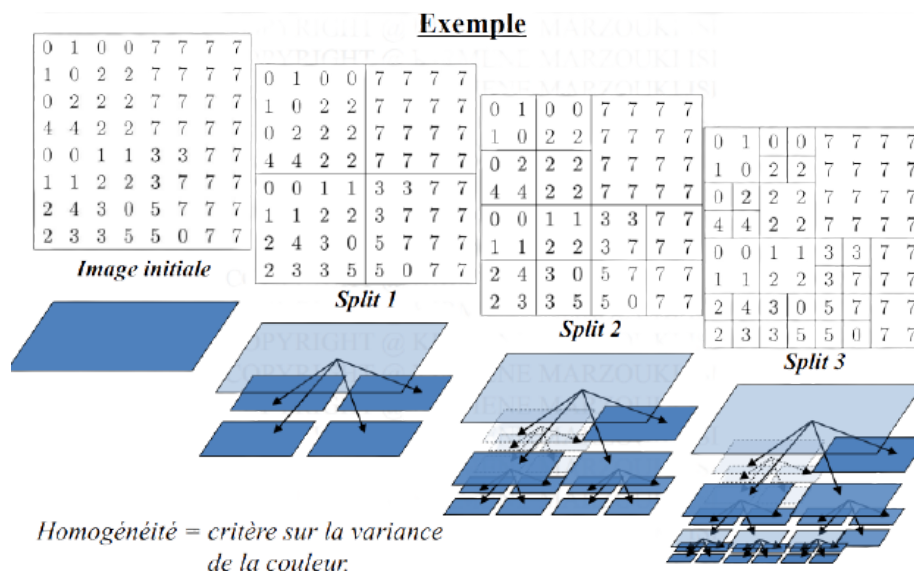
En l'absence de contextes particuliers, les meilleurs résultats sont obtenus avec les méthodes hybrides.

## 7.1. Split and merge

Dans l'algorithme split and merge, les régions agrégées proviennent d'une première phase (split) de traitement de l'image qui construit de manière récursive des régions carrées de taille variable mais homogènes.

- Phase 1 : Créer les zones homogènes = SPLIT
- Phase 2 : Les regrouper = MERGE

**Split** : La méthode de découpage de l'image utilisée dans cet algorithme est basée sur la notion de "quadtree". Cette structure de données est un arbre quaternaire qui permet de stocker l'image à plusieurs niveaux de résolution.



**Merge** : La solution, qui correspond à la phase merge de l'algorithme, est de procéder à une fusion de régions après le découpage. L'implémentation la plus simple de cette



fusion cherche tous les couples de régions adjacentes dans l'arbre issu du découpage et cherche à les fusionner si leur couleur est assez proche.

## 8. Organisation et planning

N°	Mode	Nom de la tâche	Durée	Début	Fin	18 Oct 21	15 Nov 21	13 Déc 21	10 Jan 22	07 Fév 22	07 Mar 22	04 Avr 22				
	Tâche					M	S	M	D	J	L	V	M	S	M	D
1		<b>Formation</b>	<b>34 jours</b>	<b>Sam 30/10/21</b>	<b>Mer 15/12/21</b>											
2		Formation traitement d'images	28 jours	Sam 30/10/21	Mar 07/12/21											
3		Fomation Android	6 jours	Mer 08/12/21	Mer 15/12/21											
4		Cahier des charges	11 jours	Lun 01/11/21	Dim 14/11/21											
5		<b>Phase Initiation (Conception)</b>	<b>26 jours</b>	<b>Mer 10/11/21</b>	<b>Mer 15/12/21</b>											
6		Diagramme de cas d'utilisation général du pr	4 jours	Mer 10/11/21	Dim 14/11/21											
7		Hierachisation cas d'utilisation	12 jours	Lun 15/11/21	Mar 30/11/21											
8		Description détaillée du projet	11 jours	Mer 01/12/21	Mer 15/12/21											
9		<b>Elaboration 1 (Prototype)</b>	<b>33 jours</b>	<b>Mer 01/12/21</b>	<b>Ven 14/01/22</b>											
10		Description détaillée de l'application	23 jours	Mer 15/12/21	Ven 14/01/22											
11		Création des diagrammes UML de l'applicati	33 jours	Mer 01/12/21	Ven 14/01/22											
12		<b>Elaboration 2 (Intégration)</b>	<b>38 jours</b>	<b>Mer 15/12/21</b>	<b>Ven 04/02/22</b>											
13		Description détaillée des algorithmes utilisés	16 jours	Sam 15/01/22	Ven 04/02/22											
14		Mise en place dans l'environnement de développement	6 jours	Mer 15/12/21	Mer 22/12/21											
15		Mise en oeuvre de l'algorithme de detection de contour	16 jours	Jeu 23/12/21	Jeu 13/01/22											
16		Mise en oeuvre de l'algorithme de segmentation	16 jours	Ven 14/01/22	Ven 04/02/22											
17		Test et intégration	82 jours	Jeu 30/12/21	Ven 22/04/22											
18		<b>Construction 1</b>	<b>31 jours</b>	<b>Sam 05/02/22</b>	<b>Ven 18/03/22</b>											
19		Description détaillée complète du projet	7 jours	Sam 05/02/22	Dim 13/02/22											
20		Amélioration de précision de détection de contour	25 jours	Lun 14/02/22	Ven 18/03/22											