UNIVERSITÉ MOHAMED KHIDER – BISKRA

# DETECTION DE LA SAILLANCE DES IMAGES 2D

5 JUILLET 2022


| *auteur* | *Professeur* |
|---|---|
| Moussi Ismail | Dr.Bahi Naima |

# Table des matières

# Table des figures

# Remerciements

*Je tiens à exprimer toute ma reconnaissance à ma directrice de mémoire, Bahi Naima. Je la remercie de m'avoir encadré, orienté, aidé et conseillé.*

*J'adresse mes sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions.*

*Je remercie mes parents, pour leur soutien constant et leurs encouragements.*

*À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.*

# Résumé

La détection de la saillance visuelle est un domaine de recherche très actif dans le domaine de la vision par ordinateur et du traitement d'images ces dernières années. Son objectif est de trouvé rapidement les régions saillantes dans des images comme l'œil humain sur un ordinateur. Il peut être largement utilisé dans la reconnaissance d'objets, l'édition d'images et le traitement d'images. recherche, etc.
Le modèle de détection de saillance visuelle est le processus de prédiction des informations d'une image ou d'une vidéo qui sont les plus visuellement perceptibles grâce à des algorithmes de vision par ordinateur.

L'attention peut être classée en deux fonctions distinctes : l'attention ascendante "bottom-up" et l'attention descendante "Top-down".L'attention visuelle ascendante commence par un traitement visuel de base le long des voies corticales visuelles.L'attention ascendante "Buttom-up" est rapide, involontaire et très probablement rétroactive.Dans le traitement descendant : les perceptions commencent par les plus générales et évoluent vers les plus spécifiques.Mettre tout simplement le cerveau applique ce qu'il sait pour remplir les blancs et anticiper la suite.

La saillance visuelle fait référence au fait que face à une scène, les humains traitent automatiquement les régions saillantes et ignorent sélectivement les régions non-saillantes .Ces zones d'intérêt sont appelées régions saillantes. Pour atteindre cet objectif ,Les différentes caractéristiques sont extraites et représentées sur différentes échelles dont les différences produisent des cartes de discontinuités. Les cartes sont ensuite fusionnées dans une seule carte appelée carte de saillance.

# Abstract

Visual salience detection has been a very active research domain in computer vision and image processing in recent years. Its goal is to quickly find salient regions in images like the human eye on a computer. It can be widely used in object recognition, image editing and image processing. research, etc.
The visual saliency detection model is the process of predicting which information in an image or video is most visually noticeable through computer vision algorithms.

Attention can be categorized into two distinct functions : bottom-up attention and top-down attention. Bottom-up visual attention begins with basic visual processing along visual cortical pathways. Buttom-up attention is rapid, involuntary, and most likely retroactive. In top-down processing : Perceptions begin with the most general and progress to the most specific. Simply putting the brain applies what it knows to fulfill the blanks and anticipate what's next.

Visual salience refers to the fact that when faced with a scene, humans automatically process salient regions and selectively ignore non-salient regions. These areas of interest are called salient regions. To achieve this objective, the different featurs are extracted and represented on different scales whose differences produce maps of discontinuities. The maps are then merged into a single map called a salience map

# Introduction générale

La perception est l'ensemble des mécanismes et procédures qui nous permettent de prendre connaissance du monde qui nous entoure sur la base des informations élaborées par nos différents sens [17].la visualisation est passée de la fonction d'inventaire et de communication à une fonction de traitement et d'analyse de l'information [18].Le besoin d'informatique basée sur la perception apparaît, par exemple, dans les problèmes d'analyse de processus complexes qui résultent de l'interaction de nombreux processus composants et du contrôle de ces processus.

L'attention visuelle est un processus qui dirige une infime fraction de l'information arrivant au cortex visuel primaire vers des centres de haut niveau impliqués dans la mémoire de travail visuelle et la reconnaissance des formes [19] .L'attention sert au moins quatre objectifs différents dans le système visuel, y compris la réduction des données/la sélection des stimuli, l'amélioration des stimuli, la liaison des caractéristiques et la reconnaissance[20].

L'attention visuelle facilite notre capacité à localiser rapidement les informations les plus importantes dans une scène [21] [22] . De telles régions d'image sont dites saillantes car on suppose qu'elles attirent plus l'attention du système visuel que d'autres parties de l'image. On s'attend à ce que ces régions saillantes possèdent des caractéristiques distinctives par rapport aux autres dans l'image. L'étude de la détection de la saillance peut révéler les mécanismes attentionnels des systèmes visuels biologiques, ainsi que modéliser leur comportement de sélection de fixation.D'autre part, en tant que composant du traitement de la vision artificielle de bas niveau, il facilite les traitements ultérieurs tels que la détection ou la reconnaissance d'objets en réduisant les coûts de calcul, ce qui est un élément clé dans les applications en temps réel. Pour la détection d'objets, cela serait toujours plus efficace qu'un échantillonnage dense, à condition de pouvoir garantir l'exactitude des mécanismes attentionnels des systèmes visuels biologiques, ainsi que de modéliser leur comportement de sélection de fixation.

D'autre part, en tant que composant du traitement de la vision artificielle de bas niveau, il facilite les traitements ultérieurs tels que la détection ou la reconnaissance d'objets en réduisant les coûts de calcul, ce qui est un élément clé dans les applications en temps réel. Pour la détection d'objets, cela serait toujours plus efficace qu'un échantillonnage dense, à condition de pouvoir garantir la précision du mécanisme attentionnel.

La détection de la saillance visuelle a reçu beaucoup d'attention de la part des psychologues et des chercheurs en vision par ordinateur [11],[23],[24].

Le système visuel humain (SVH) reçoit une quantité considérable d'informations bien au-delà de sa capacité à les traiter toutes. Pour faire face à de grandes quantités d'informations, l'attention visuelle est l'un des mécanismes les plus importants déployés dans le SVH pour réduire la complexité de l'analyse de scène . Grâce à l'attention visuelle, les spectateurs peuvent concentrer leur attention de manière sélective sur des domaines d'intérêt spécifiques de la scène.

Ce travail est divisé en quatre chapitres, Le premier chapitre est une étude théorique du système visuel humain et la perception visuelle en informatique et l'attention visuelle et

les modéles (cognitifs, bayésiens, théorie de l'information,décision, analyse spectrale , les modèles de classification,graphique),les processus attentionnels : ascendant "Bottom-up" et descendant "Top-down" .A la fin nous présenstons la carte de saillance et les différentes étapes de construction.

Dans le deuxième chapitre nous présenstons brièvement le premier modèle d'attention visuelle proposé par Kosh et Ulmann [16] d'obtenir la saillance.Ensuite nous avons détaillé le modèle de la saillance visuelle Laurent Itti, Christof Koch, and Ernst Niebur ,un modèle d'attention visuelle basée sur la saillance [11].

Le troisième chapitre représente le travail pratique qui détermine l'algorithme principal et les fonctions traitant.Alors que le quatrième chapitre contient les codes et résultat de notre application.Nous présenstons appliqué notre travail sur la l'environnement Visual studio code avec le langage python et la bibliothèque OpenCv.

CHAPITRE 1

# INTRODUCTION A LA SAILLANCE VISUELLE

## 1.1 Introduction

Dans cette chapitre théorique du projet, nous discutons de la perception visuelle et de sa théorie, de la cécité attentionnelle et de la saillance de l'information visuelle, et spécifions les repères visuels graphiques. Nous travaillons à mettre en évidence la relation entre la perception visuelle et l'infographie et la relation entre la saillance de l'information visuelle et l'infographie,ensuit nous discutons de L'attention visuelle et ses modèles, les processus attentionnels : ascendant et descendant.Ensuit nous présentons la carte de saillance.Ensuit nous présentons les mesures d'évaluation et les bases de données qui vont nous permettre d'analyser et de comparer les différents modèles de la saillance visuelle.

## 1.2 Perception visuelle du point de vue de l'infographie

La perception visuelle est le processus par lequel nous construisons une représentation mentale du monde environnant suite à la stimulation de la rétine par des rayons lumineux. cela dépend non seulement de la nature du capteur (l'œil) qui collecte ces informations lumineuses, mais également de tous les traitements ultérieurs dans le cerveau, qui peuvent être influencés par une expérience antérieure, telle que la connaissance de la courbure naturelle des objets ou d'autres informations sensorielles.Par exemple, notre mouvement, par le déplacement de l'image du monde sur notre rétine, nous renseigne sur sa structure spatiale[25] [26].

### 1.2.1 Le système visuel humain

Le système humain traite l'information de son environnement au travers d'organes sensoriels dédiés.Le système visuel comprend à la fois les yeux et le cerveau [27]. La lumière pénètre dans votre œil où elle frappe la rétine, ce qui déclenche des récepteurs de lumière pour envoyer des signaux électriques à travers votre nerf optique, qui se déplacent vers l'arrière de votre cerveau où se déroulent les premières étapes de la perception visuelle.Mais la quantité d'informations qui parviennent à nos yeux à chaque instant est si importante qu'elle ne peut être traitée dans son intégralité,ce qui nous permet de percevoir plus de détails,C'est ce qu'on appelle l'attention "ouverte",Ce déploiement de l'attention visuelle fait intervenir deux mécanismes :"Bottom-Up", "Top-Down".Au niveau du cortex visuel l'information visuelle est traitée ces et ceci grâce à des cellules spécifiques (figure 1.1).

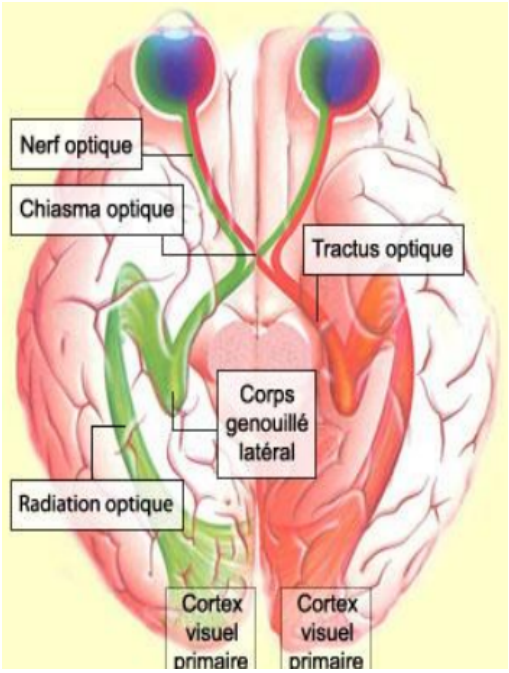



FIGURE 1.1 – Système visuel humain [1].

Le globe oculaire en forme de sphère formée par plusieurs couches classées de l'extérieur vers l'intérieur comme suite (figure 1.2) :

1. Sclérotique.
2. Choroïde.
3. Cornée :barrière protectrice transparente entre le milieu extérieur et l'œil à proprement parler.
4. Iris :qui permettent la modulation de la quantité de lumière entrante
5. Corps ciliaire
6. Cristallin
7. Rétine (couche qui tapisse l'intérieur du globe oculaire à l'exception de deux points la fovéa et le lieu d'émergeante du nerf optique.

On trouve deux types de photo-récepteurs :

1. Les cônes :sont des photorécepteurs situés dans la structure de la rétine.Permettent l'extraction des couleurs (figure 1.3).
2. Les bâtonnets :sont des cellules réceptrices situées au fond de l'oeil et composant, avec les cônes, les cellules photosensibles de la rétine.

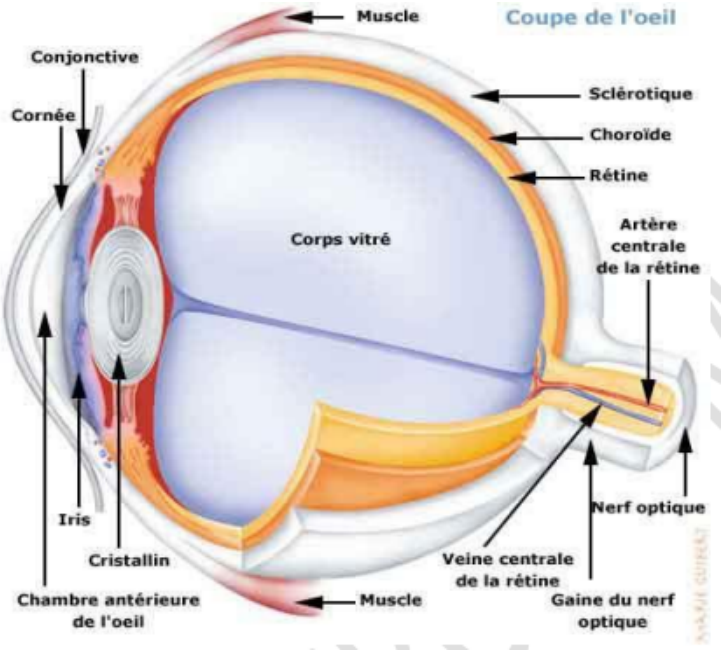



FIGURE 1.2 – L'œil humain [44].

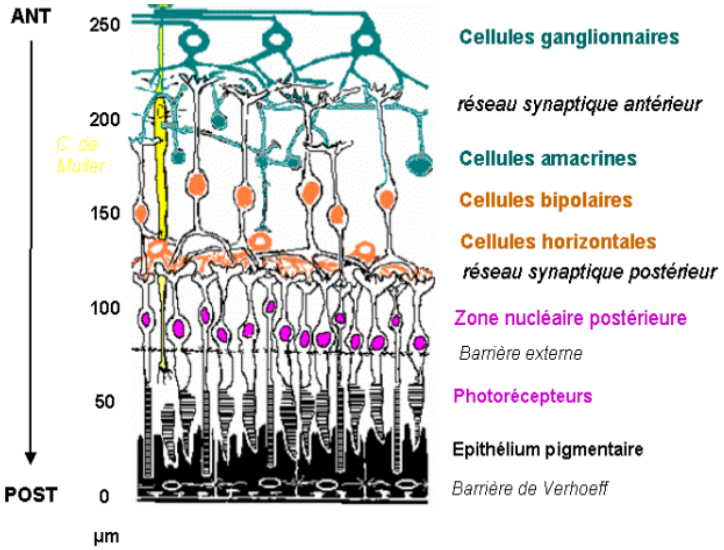



FIGURE 1.3 – Organitation de la rétrine sensorielle[44]

### 1.2.2 Infographie

L'infographie a pour but de générer une ou plusieurs images à partir d'une description d'une scène. Ces descriptions de scènes (modèles) peuvent correspondre à des parties existantes du monde physique, telles que des simulateurs de vol et de conduite ; des mondes physiques hypothétiques, tels que des simulations de construction ; ou des mondes virtuels fantastiques, tels que des jeux informatiques. Les images résultantes peuvent être des approximations photographiques de la réalité physique correspondant à la description de

la scène, ou elles peuvent être des représentations plus stylisées de la scène.

### 1.2.3 Sciences de la vision

La vision est un processus qui produit à partir d'images du monde extérieur une description utile au spectateur et non encombrée d'informations non pertinentes.
Le milieu des années 1900 a vu le début des neurosciences modernes, qui étudient à la fois le fonctionnement à petite échelle des neurones individuels et l'organisation architecturale à grande échelle du cerveau et du système nerveux.Une partie importante de la recherche en neurosciences s'est concentrée sur la vision.Plus récemment, l'informatique a contribué à la compréhension de la perception visuelle en fournissant des outils pour décrire avec précision des modèles hypothétiques de calculs visuels [26].Le terme science de la vision a été inventé pour désigner l'étude multidisciplinaire de la perception visuelle impliquant la psychologie perceptive, les neurosciences et l'analyse informatique.

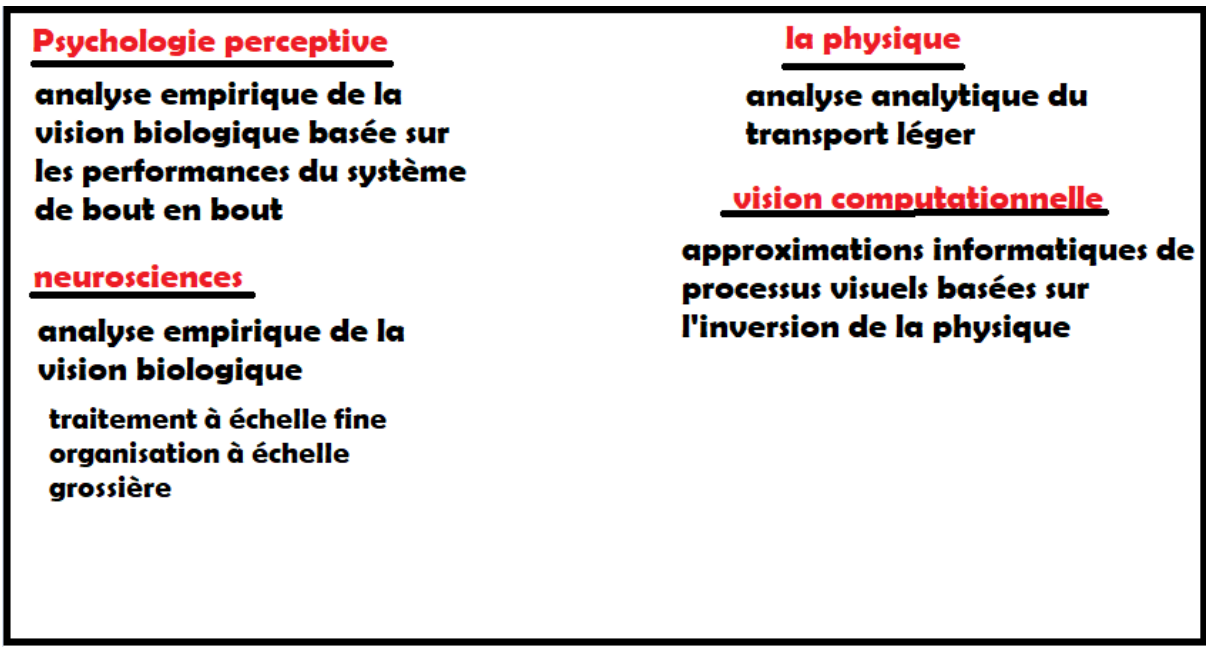



FIGURE 1.4 – Comprendre la vision nécessite de s'appuyer sur des outils issus de multiples disciplines.

### 1.2.4 Les théories de la perception

Les théories de la perception sont les études concernant les règles de la vision. Elles décrivent comment le cerveau perçoit et analyse l'environnement qui l'entoure pour élaborer des images cohérentes.

#### 1.2.4.1 Réalisme direct :

On dit que le réalisme direct implique la prise de conscience "directe" des objets physiques sans la nécessité d'une prise de conscience des représentations intermédiaires des objets [28](figure 1.5,a).

#### 1.2.4.2 réalisme indirect :

Le réalisme indirect est une théorie de la perception, plus étroitement associée à Descartes et Locke, mais implicite dans la pensée de nombreux scientifiques [29](figure 1.5,b)

#### 1.2.4.3 Voile de perception :

Tout ce que nous recevons en fait est le voile qui couvre le monde, un voile qui consiste en nos données sensorielles. La vieille inquiétude du « voile de la perception » surgit dans le débat contemporain sur la conscience et la représentation, et la meilleure articulation de l'inquiétude est en termes de contrainte de contenu[30].

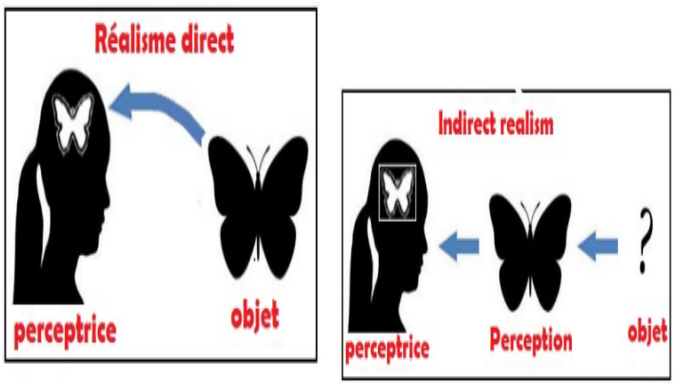
### 1.2.5 Comment s'organisent les théories de la Perception ? :

Les théories de la perception sont organisées en fonction de leurs réponses à trois questions bipolaires principales :

— La question de l'inné et de l'acquis :
Les psychologues sont partagés entre les partisans de l'acquis, qui n'accordaient d'intérêt qu'aux conditionnements, et ceux qui reconnaissaient l'innéité des instincts mais les considéraient comme résultant d'une force inexplicable [31] .

— La question du rationalisme ou de l'empirisme.
Bref, il n'y a pas de rationalité vide, pas d'empirisme disjoint, ce sont les deux obligations philosophiques de la synthèse étroite et précise de la théorie et de l'expérience dans la physique contemporaine[32] .

— La question du globalisme ou de l'élémentarisme.
Par exemple, il y a L'élémentarisme de Biederrman qui développe concept de géons en 1987 alors que le globalisme de la Gestalt où la perception globale de la forme précède celle de ses éléments[33].




(a) Réalisme direct. (b) Réalisme indirect.

FIGURE 1.5 – Les théories de la perception
[34]

### 1.2.6 Vision par ordinateur

La Computer Vision ou vision par ordinateur est une technologie permettant aux ordinateurs de voir de la même façon que les êtres humains.La vision par ordinateur permet aux ordinateurs d'effectuer une grande variété de tâches,son spectre d'applications est large, allant de la détection / suivi d'objet, à la recherche d'images par le contenu [26]. La recherche a longtemps été influencée par les travaux de David Marr [2], dont le paradigme visuel fournit une analyse "bottom up" unique centrée sur les données (Figure 1.6 ).
Les informations délivrées sont de plusieurs ordres : taille, position, orientation, mouvement, force, structure, interprétation des objets, ... Elles peuvent êtreutilisées pour une infinité d'applications : surveillance, suivi, commande, reconnaissance de forme, contrôle qualité

### 1.2.7 MÉTHODES D'ILLUSTRATION BASÉES SUR LA PERCEPTION

Un fait fondamental de la compréhension des phénomènes visuels du système de vision humain est que notre perception de la façon dont nous "voyons" les objets n'est pas

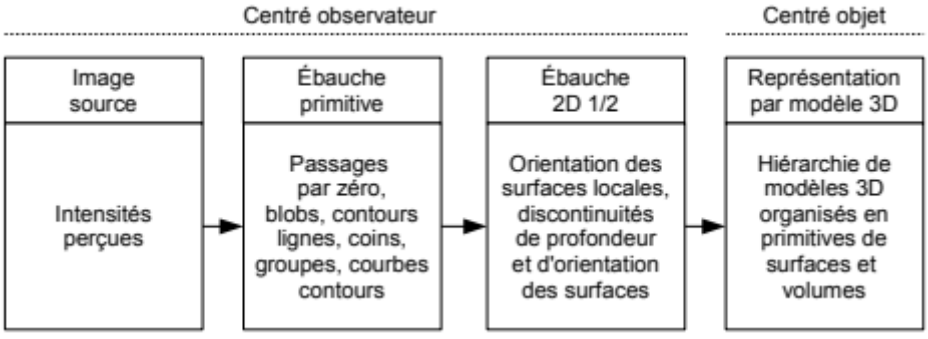



FIGURE 1.6 – Le paradigme de David Marr [2].

seulement déterminée par les rayons lumineux qui atteignent nos yeux, de plus, des objets tridimensionnels sont construits dans notre cerveau en effectuant un traitement neuronal supplémentaire.

Certains des phénomènes visuels du système de vision humaine.

#### 1.2.7.1 Objets tridimensionnels ambigus

Figures ambiguës : une image donnée peut créer des perceptions multiples distinctes, comme le montre le cube de Necker de la (figure 1.7), où l'on peut percevoir deux orientations différentes. Le changement de perception implique que l'interprétation de l'interposition entre les lignes leurs relations de profondeur change également[7].

**1.2.7.1.a Le cube de Necker :** Un exemple très célèbre d'objet tridimensionnel ambigu est le cube de Necker.Le cube de Necker est une figure réversible, qui est souvent utilisée pour évaluer l'attention ou la perception de la profondeur.[35]

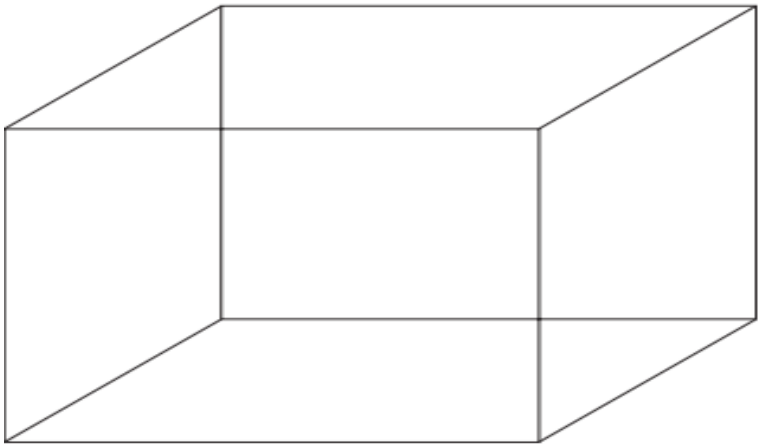


FIGURE 1.7 – Un cube Necker qui peut, avec une attention soutenue, alterner entre deux interprétations s'excluant mutuellement : Un cube tourné vers le bas et vers la gauche, ou un cube tourné vers le haut et vers la droite.[3]

**1.2.7.1.b Les cubes de Köfermann :** Un autre exemple de figures ambiguës sont les cubes de Kopfermann Bien qu'il soit évident que les deux cubes de Kopfermann sont des projections possibles de cubes, ils apparaissent généralement comme des figures plates et il faut beaucoup plus de temps et d'efforts pour les voir comme des cubes par rapport au cube de Necker.

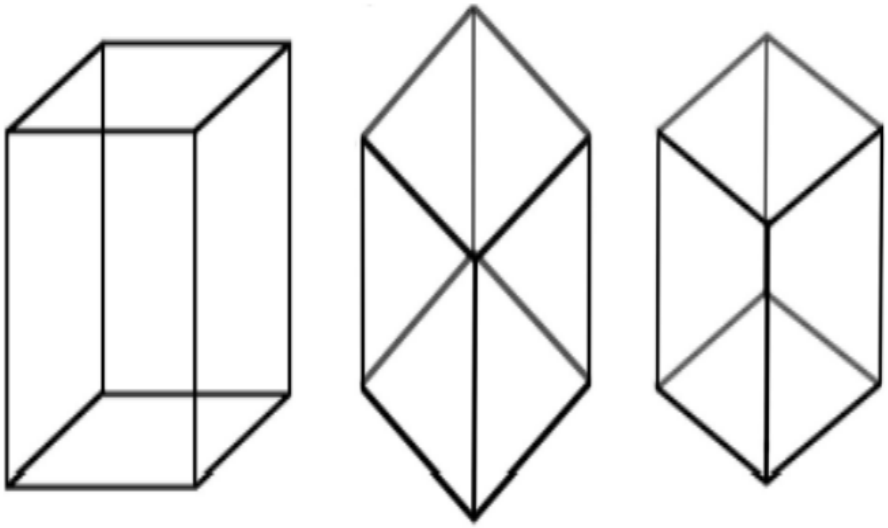


FIGURE 1.8 – Cubes de Kopfermann et Cube de Necker (à gauche) ,la profondeur est plus facilement perçue dans le cube de Necker (à gauche) que dans les deux cubes de Kopfermann [4] .

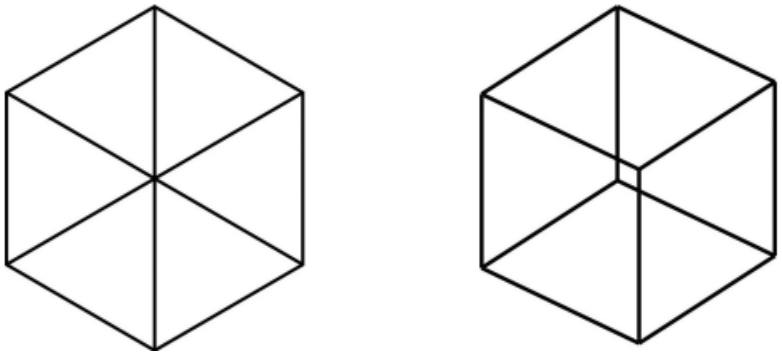


FIGURE 1.9 – kopfermann Cube, Kopfermann Cube avec vue différente[5]

En appliquant les deux règles ci-dessous, nous pouvons expliquer pourquoi le dessin de droite permet de créer si facilement un objet 3D.

**RÈGLE 1** : Interprétez toujours une ligne droite dans une image comme une ligne droite en 3D.
**RÈGLE 2** : Si les pointes de deux lignes coïncident dans une image, alors interprétez-les toujours comme coïncidant en 3D.

#### 1.2.7.2 Objets bidimensionnels créant une perception de profondeur 3d et 2d

La différence entre les formes 2D et 3D est qu'une forme 2D comprend deux dimensions qui sont la longueur et la largeur. Par contre, une forme 3D intègre trois dimensions qui sont la longueur, la largeur et la hauteur(fig 1.10)

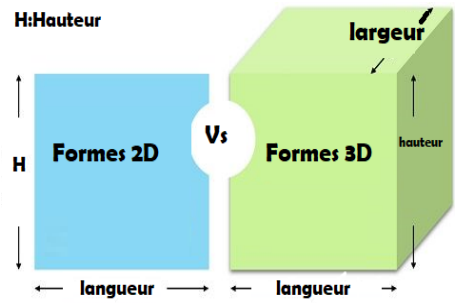



FIGURE 1.10 – Différence entre les formes 2D et 3D [6]

**Voici les notions suivantes** :

- TDS(x,y,z) :Trois dimensions sont là, X, Y et Z.
- CSC :Cube, sphère, cône, cuboïde,etc.
- FIO :Formes isométriques et orthogonales.

TABLE 1.1 – Tableau de comparaison [6]

| ***BASE DE COMPARAISON*** | **FORMES 2D** | **FORMES 3D** |
|---|---|---|
| **De base** | Il n'y a que 2 dimensions qui sont X et Y | TDS(x,y,z) |
| ***Constructions*** | Carré, cercle, triangle, rectangle,etc | CSC |
| ***Represents*** | Vue de dessus,côté,dessous,face | FIO |
| ***Implique*** | Longueur et largeur | largeur et hauteur |
| ***Bords*** | Sont entièrement visibles sur les dessins | Non visible ou caché height |

Les figures suivantes dcrivent certains phénomènes visuels d'objets bidimensionnels qui créeront une certaine impression de profondeur qui leur est associée.

FIGURE 1.11 – Deux rectangles avec différentes nuances de gris sur fond blanc .le rectangle avec la nuance de gris la plus foncée semble être plus proche de l'observateur. Cet effet peut être inversé en utilisant un fond noir, comme illustré ci-dessous [5]

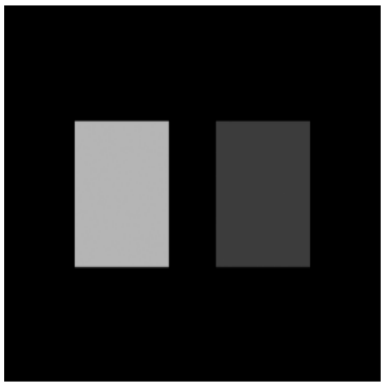


FIGURE 1.12 – Les mêmes rectangles gris sur fond noir [5]

Maintenant le rectangle plus clair semble être plus proche de l'observateur. Un autre exemple d'illusions visuelles figure(1.13).

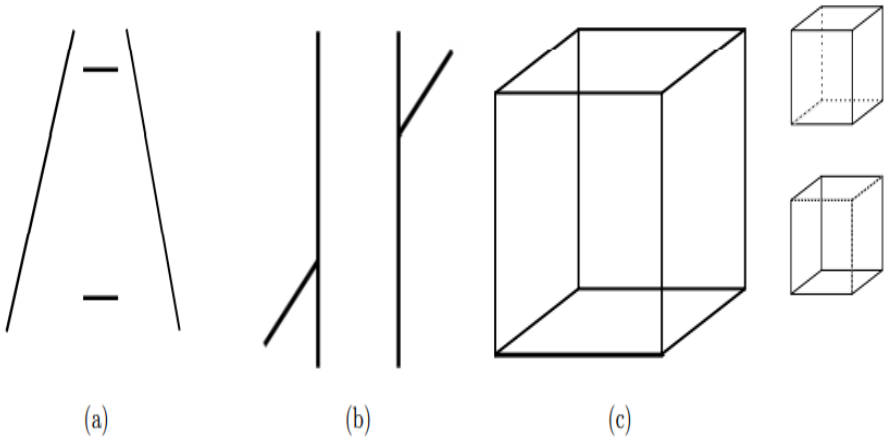



FIGURE 1.13 – L'illusion de Ponzo dans la figure (a) : les deux lignes horizontales partagent la même longueur mais ne semblent pas l'être. L'illusion de Poggendorf sur la figure (b) : les lignes obliques apparaissent décalées. Le cube de Necker de la figure (c) peut être soit un cube vu de dessus (cube du haut à droite) soit de dessous (cube du bas à droite)[7]

### 1.2.8 Cécité attentionnelle

La fonction première de l'attention est de choisir quelles informations prioriser,alors qu'arrive-t-il aux choses volontairement ignorées ou non.Sont-ils bien que tout soit accessible avant conscience ?.
En raison d'un grand nombre d'informations qui arrivent en même temps, une incapacité à voir les grands changements dans la scène visuelle peut se produire, et c'est ce qu'on appelle la cécité changeante [36].Alors il existe une soi-disant cécité intentionnelle, qui se manifeste par l'incapacité de percevoir des objets autour de la scène auxquels nous ne prêtons pas directement attention.

### 1.2.9 Détection de la saillance visuelle à l'aide de la pondération du contenu de l'information

La saillance visuelle est un attribut qui mesure la probabilité que les yeux humains se fixent sur cette zone.La saillance de toute chose est qu'elle attire l'attention. Plus précisément, le degré auquel il attire l'attention par rapport à d'autres éléments de son environnement [8].Dans le système de vision par ordinateur, ce mécanisme est reconnu comme une détection de saillance, où seules les informations pertinentes sont examinées et atteignent le niveau de sensibilisation tandis que les informations non pertinentes sont rejetées.de nombreuses méthodes ont été proposées pour l'estimation de la saillance visuelle, mais globalement classées en deux catégories . L'approche descendante, axée sur les objectifs et lente, nécessite d'énormes connaissances préalables et un traitement de données de haut niveau.De bas en haut , qui est rapide, pré-attentif et basé sur les données. Ces dernières années, visuel de bas en haut.

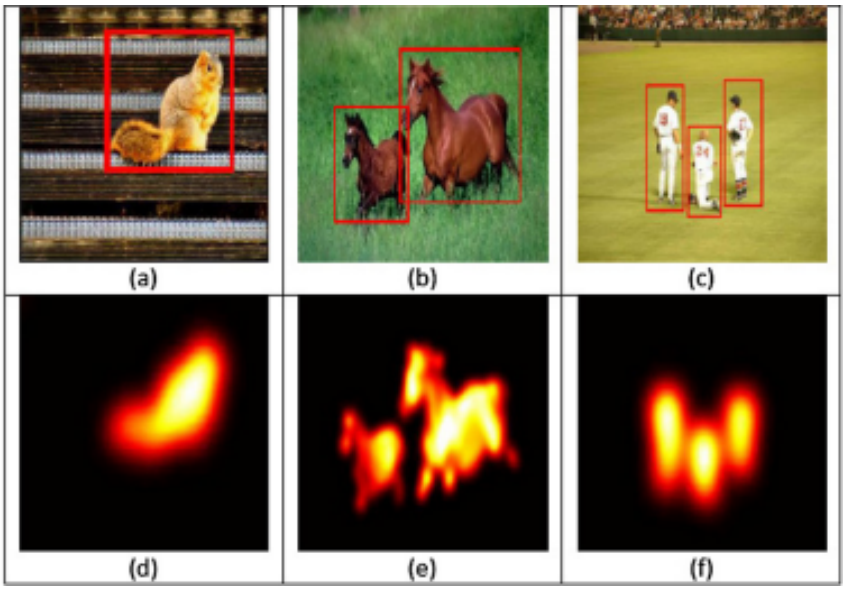



FIGURE 1.14 – Approche de saillance proposée pour calculer la carte de saillance haute résolution (Bottom)[8].

## 1.3 Processus attentionnels

L'attention est un processus de sélection et de synchronisation d'informations visuelles basé sur la saillance de l'image elle-même (ascendant) et sur la connaissance préalable de la scène (descendant)[37]. L'attention peut être classée en deux fonctions distinctes : l'attention ascendante (bottom-up) et l'attention descendante(Top-down)[38] .

### 1.3.1 Modèles à filtres

#### 1.3.1.1 Modèles Top-down

le traitement descendant(top-down), les perceptions commencent par les plus générales et évoluent vers les plus spécifiques. Ces perceptions sont fortement influencées par nos attentes et nos connaissances antérieures [39].Le modèle d'attention sélective descendante, qui est un modèle de traitement piloté par un concept basé sur des informations pré-apprises, effectue une tâche spécifique tels que la détection et la reconnaissance d'objets, l'attention préférable, etc.

Mettre tout simplement le cerveau applique ce qu'il sait pour remplir les blancs et anticiper la suite.

#### 1.3.1.2 Modèles de Buttom-up

Les processus bottom-up appelés également processus ascendants (de la rétine vers le cerveau)sont des mécanismes exogènes dépendants des propriétés intrinsèques du stimuli ou du signal visuel comme le contraste, la texture, la forme, etc [40]. L'attention visuelle ascendante commence par un traitement visuel de base le long des voies corticales visuelles.L'attention ascendante (Buttom-up) est rapide, involontaire et très probablement rétroactive [41]. Le composant ascendant calcule la saillance visuelle des emplacements de scène dans différentes cartes de caractéristiques extraites à plusieurs échelles spatiales

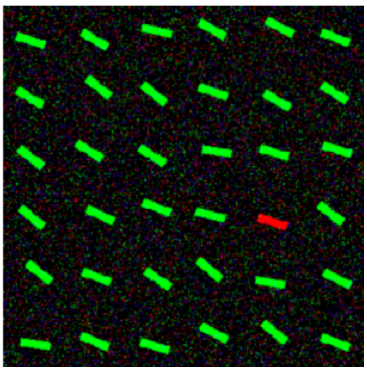


FIGURE 1.15 – Un seul objet rouge dans un champ vert sera saillant et attire notre attention d'une manière "Buttom-up".

Les études ont montré que les mécanismes Buttom-up sont plus rapide et qu'ils précèdent les influences Top-down qui sont plus longues à mettre en oeuvre et qui vont durer dans le temps.

## 1.4 La Carte de Saillance

Un concept connexe concernant la répartition de l'attention est celui d'une carte de saillance.

Le modèle de saillance prédit ce qui attire l'attention. Les résultats de ces modèles sont des cartes de saillance. Une carte de saillance est une représentation topographique de la saillance qui fait référence à des emplacements visuellement dominants et ainsi d'attirer l'attention visuelle de l'observateur. Par conséquent, une carte de saillance est une carte topographique d'un arrangement représentant la saillance Vision,elle est combine différents éléments visuels qui contribuent à la sélection attentive d'un stimulus (couleur, orientation, mouvement, etc)[42].

Il est créé en utilisant les étapes suivantes.

— Nous avons une image et les caractéristiques de base comme la couleur, l'orientation, l'intensité sont extraites de l'image. Figure(1.16) .

— Ces images traitées sont utilisées pour créer des pyramides gaussiennes afin de créer des caractéristiques Map.

— La carte de saillance est créée en prenant la moyenne de toutes les carte de caractéristiques.

La carte de saillance comprends les éléments suivants :

— Une représentation anticipée composée d'un ensemble des cartes de caractéristiques calculées en parallèle.

— Une carte de saillance topographique ou chaque emplacement encode la combinaison des propriétés sur tous les cartes de caractéristiques.

— Une cartographie sélective des propriétés d'un seul emplacement visuel représentée d'une manière non-topographique centrale, à partir de la carte de saillance.

— Inhibition de l'emplacement choisi, qui provoque un décalage automatique vers l'emplacement prochain le plus visible

L'article conceptuel de Koch et Ulman [16] décrit la carte de saillance comme une représentation topographique qui pourrait expliquer l'attention sélective d'une manière plausible compte tenu de la neurophysiologie des primates. Il est important de noter que la carte de saillance était considérée comme une représentation visuelle «précoce» basée principalement sur de simples caractéristiques visuelles.Ce modèle est inspiré par le comportement et l'architecture neuronale du système visuel des primates. Il devient une référence à des nombreux modèles par la suite.

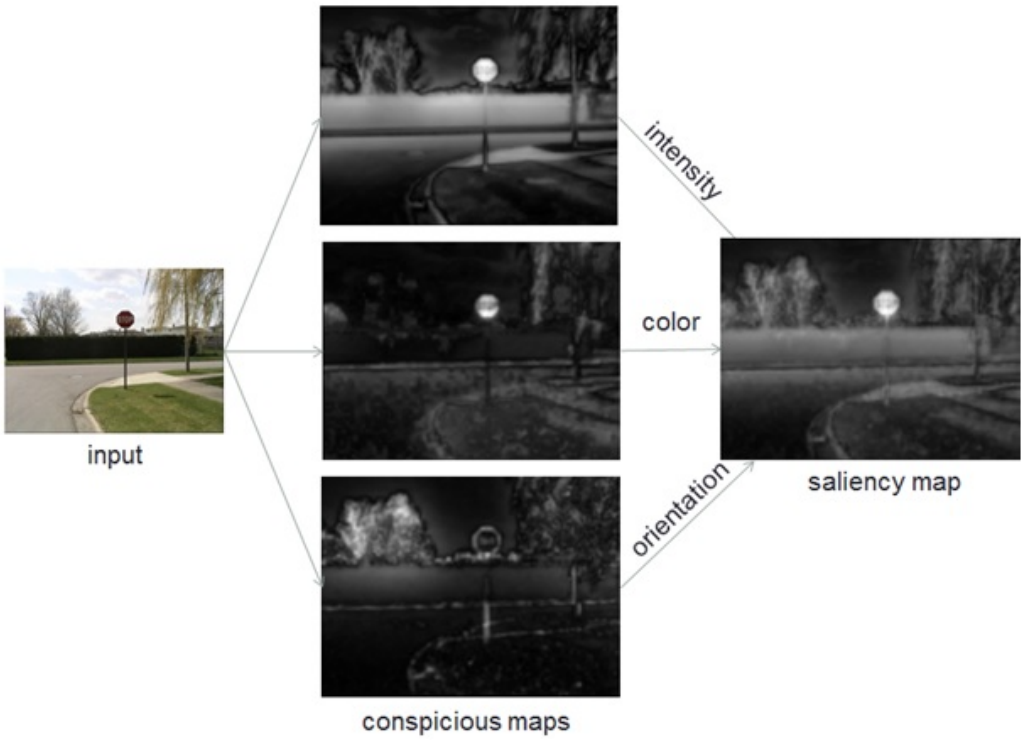



FIGURE 1.16 – Obtenir une carte de saillance en fusion les 3 cartes de vislibilités.

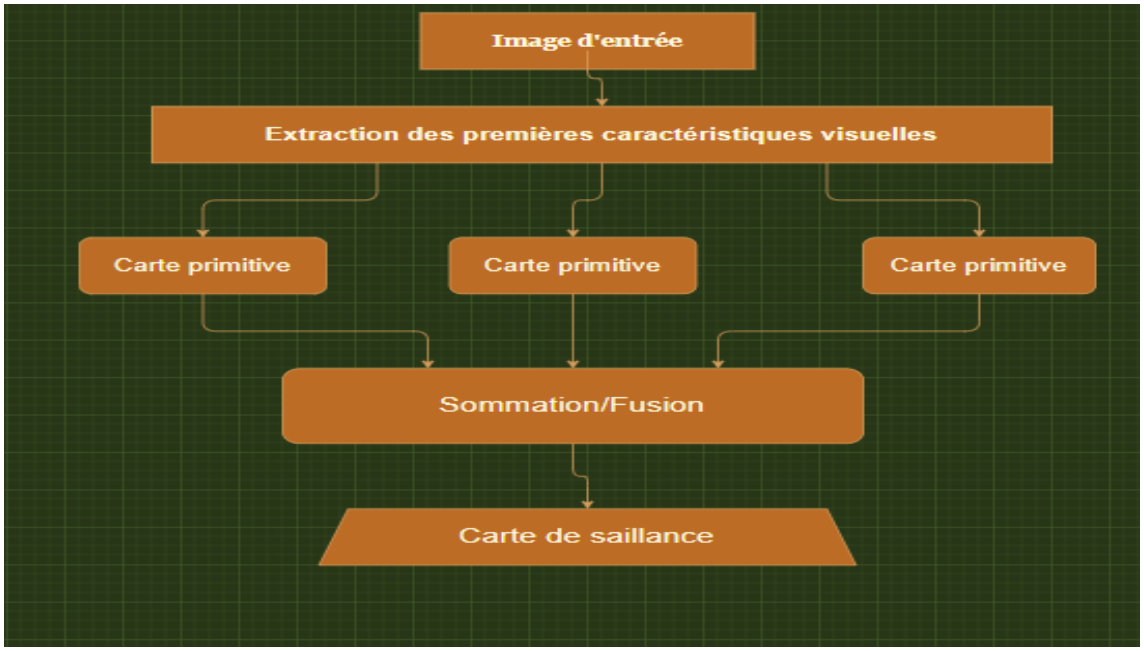



FIGURE 1.17 – Cadre proposé par Koch et Ullman. Les premières caractéristiques visuelles sont extraites de l'entrée visuelle dans plusieurs canaux parallèles séparés. Après cette extraction et un traitement particulier, une carte de caractéristiques est obtenue pour chaque canal. Ensuite, la carte de saillance est construite en fusionnant toutes ces cartes.

## 1.5 Les bases de données (Databases)

Il existe de nombreuses bases de données d'images et de vidéos pour étudier respectivement l'attention statique et l'attention dynamique.Les bases de données qui nous intéressent sont principalement utilisés pour évaluer et comparer les modèles d'attention.Les chercheurs ont également utilisé le "mouse tracking"pour estimer l'attention [43] ont montré que les modèles de mouvement de souris sont proches de ceux de mouvement de l'oeil.

### 1.5.1 Suivi des yeux et suivi de la souris ( Eye-tracking and Mouse-tracking)

1. **Eye-tracking** (Suivi de l'oeil ) :est le processus de mesurer soit
   — Le point de regard.

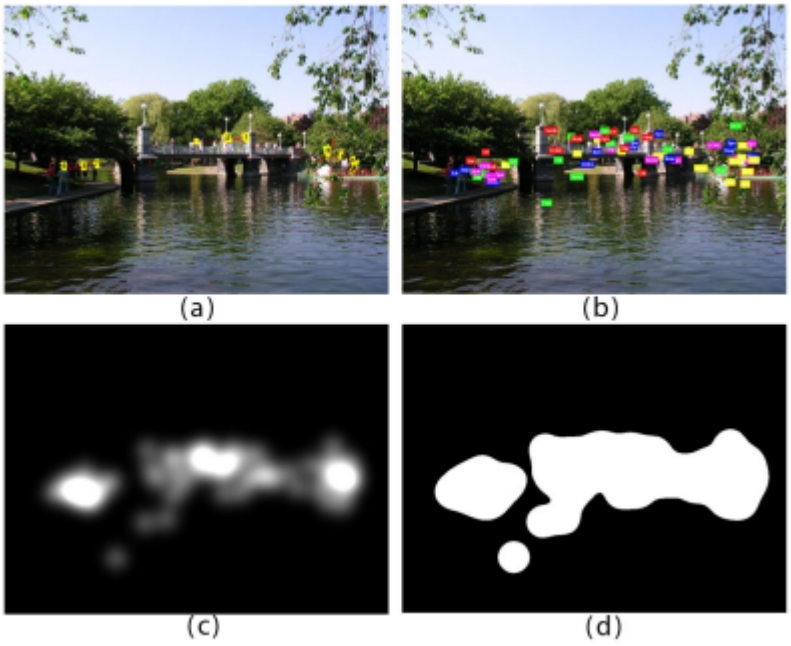
   — Le mouvement de l'oeil par rapport à la tête.
   Le "eye tracker" est un appareil utilisé pour mesurer la position et le mouvement des yeux.

2. **Mouse-tracking (suivi de souris)** : est l'utilisation de logiciels pour collecter les positions de curseur de l'utilisateur sur l'ordinateur. Le bute est de collecter automatiquement les informations les plus riches sur ce que l'utilisateur entrain de faire.

### 1.5.2 Quelques bases de donnees pour la saillance visuelle

**al et Judd** [9] Ont collectés 1003 images aléatoires , et les données "eye tracking" sont enregistrées par 15 utilisateurss.Les dimensions des images vont de 405 à 1024 avec la majorité à 768 pixels.Il y'avait 779 images de paysages et 228 images en mode portrait. ils ont collecté 1003 images aléatoires de la création Flickr commons et LabelMe [44] (Fig 1.23) et ont enregistré les données de "eye tracking" de 15 utilisateurs qui ont visionné gratuitement ces images.

**Tsotsos et Bruce** [24] La base de données est dérivée d'expériences de suivi de l'oeil "eye tracking" menées par des sujets qui ont observé 120 images de différentes couleurs. Les images sont présentées dans un ordre aléatoire pendant 4 secondes par image, avec un masque entre chaque paire d'images.
Les données ont été recueillies auprès de 20 sujets pour un ensemble de 120 images. La figure 1.24 offre une comparaison qualitative de la sortie du modèle proposé par [Bruce et Tsotsos] avec les données expérimentales de suivi des yeux pour une variété d'images. La sortie de l'algorithme Itti et Koch est également représentée à des fins de comparaison.




[!h]

FIGURE 1.18 – Eye tracking data [9]

Ils ont collecté des données de suivi oculaire sur 1003 images de 15 téléspectateurs à utiliser comme données de vérité terrain pour former un modèle de saillance à l'aide de l'apprentissage automatique.

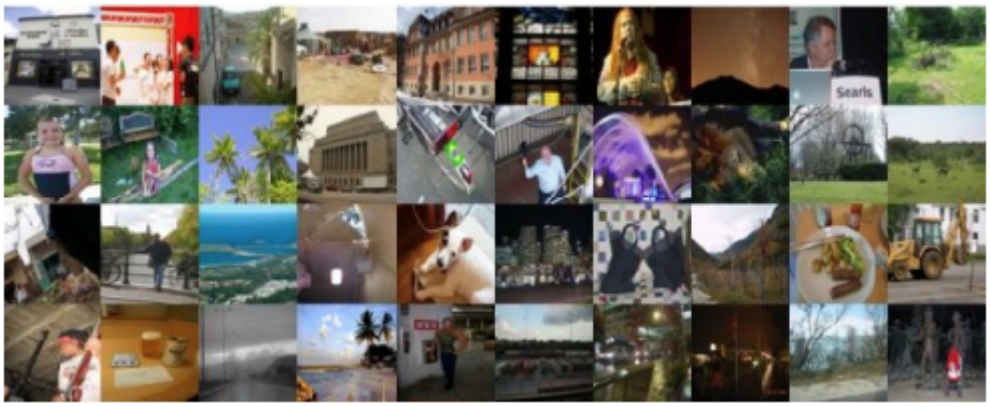


FIGURE 1.19 – échantillon des 1003 images qu'ils ont recueillies sur Flickr et LabelMe.

**al et Kootstra** [45]Les données de fixations humain sont enregistrés au cours d'une expérience de suivi de l'oeil "eye-tracking".L'expérience a été réalisée par 31 étudiants .Dans l'expérience, un ensemble d'images a été présenté :

— 99 images dans cinq catégories différentes

— 12 images d'animaux dans un cadre naturel

— 12 images de scènes de rue

FIGURE 1.20 – Résultats pour la comparaison qualitative

— 16 images de bâtiments

— 40 images de milieux naturels

— 19 images de symétries naturelles des fleurs

— des plantes et des papillons

**Base de données Microsoft** [46] cette base contient 5000 images en couleurs, il y'a aussi des vérités de terrain pour les images dans la base de données : les régions de l'attention humaine étiquetés mis en évidence avec un cadre de sélection créé par 9 sujets.Ces boîtes englobantes représentent les régions de l'attention (figure 1.26)

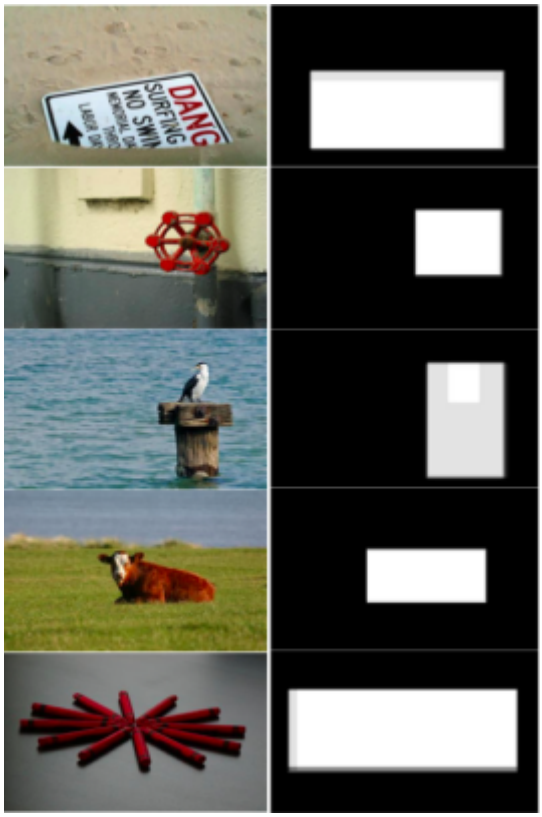


FIGURE 1.21 – Quelques Image de la base de données Microsoft

Colonne 1 : images couleur, Colonne 2 : les régions de l'attention.

## 1.6 Modèles d'attention visuelle

Jusqu'à présent, de nombreux modèles de saillance ont été proposés et ils se sont avérés utiles pour diverses applications, y compris la vision par ordinateur (par exemple, la détection d'objets [47] et la reconnaissance d'objets), la robotique (par exemple, l'interaction homme-machine) et le traitement du signal visuel (par exemple, la région compression basée sur les intérêts et redimensionnement d'image [48]).
Les modèles de saillance peuvent être classés en modèles basés sur les pixels et en modèles basés sur les objets.

— Les modèles basés sur les pixels visent à mettre en évidence les emplacements des pixels où les fixations sont susceptibles de se produire.

— Les modèles basés sur des objets se concentrent sur la détection d'objets saillants dans une scène visuelle.

La majorité des modèles de saillance dans la littérature sont des modèles de saillance basés sur des pixels, tels que Itti [11].

Les modèles de salinité sont divisés en huit catégories[16] en fonction de l'approche de modélisation utilisée, y compris les modèles de

— les modèles cognitifs

— Modèles Bayésiens
— Modèles de la théorie de décision
— Modèles se basant sur la théorie de l'information
— Modèles graphiques
— Modèles d'analyse Spectrale
— Modèles de Classification

#### 1.6.0.1 Modèles Cognitifs

le plupart des modèles d'attention sont inspirés par le concept cognitif. Itti est peut-être le premier travail notable dans le domaine de la modélisation informatique de la saillance, qui combine des caractéristiques d'image multi-échelles en une seule carte de saillance topographique [11],Cette modéle utilise trois caractéristiques : la couleur, l'intensité et l'orientation. C'est un modèle proposé en 1998 par Itti, Kosh et Niebur basé sur 'architecture du premier modèle d'attention visuelle proposé par Kosh et Ulmann (Nous le décrirons en détail dans une section ultérieure )

#### 1.6.0.2 modèles bayésiens

Dans ces modèles, les connaissances préalables et les informations sensorielles sont combinées de manière probabiliste selon les règles de Bayes pour détecter les objets d'intérêt (Les modèles bayésiens combinent une connaissance préalable des scènes visuelles (par exemple, le contexte de la scène) en utilisant la règle de Bayes. ).

#### 1.6.0.3 Les modèles théoriques de décision

Les modèles théoriques de décision traitent le déploiement de l'attention comme un processus de prise de décision dans lequel l'attention est déterminée par l'optimalité (probabilité minimale d'erreur)

#### 1.6.0.4 Les modèles de théorie de l'information

Ces modèles traitent les yeux humains comme des sélecteurs d'informations et réussir à sélectionner les régions les plus informatives d'une scène visuelle.ces modèles sont basés sur l'hypothèse que le calcul de la saillance localisée.

#### 1.6.0.5 Les modèles de classification

Les modèles de cette catégorie utilisent des techniques d'apprentissage automatique pour apprendre les mappages de «stimuli-saillance» des caractéristiques de l'image aux fixations oculaires. Ils estiment la saillance s, p(s|f) où f est un vecteur caractéristique qui pourrait être le contraste d'un emplacement et de son voisinage environnant.

#### 1.6.0.6 Les modèles graphiques

Les modèles graphiques sont des modèles bayésiens généralisés qui ont été utilisés pour modéliser des mécanismes d'attention complexes dans l'espace et dans le temps.

## 1.7 Conclusion

Nous recevons une grande quantité d'informations visuelles qui atteignent nos yeux à tout moment, nous concentrons donc notre attention sur des régions particulières. Ces zones sont choisies par des processus attentionnels :ascendant et descendante.les processus descendante dépendent de la tâche ainsi que l'observateur,tandis que ascendant guidés uniquement par les stimuli et sont donc communs aux observateurs.L'attention visuelle est un sujet actif dans les domaines de la vision par ordinateur.
Dans ce chapitre, nous avons défini la perception visuelle puis nous avons discuté des modélisation de l'attention visuelle en mettant l'accent sur les modèles de saillance ascendants.Nous avons ensuite présenté la carte de saillance et les différentes étapes de construction en se basant sur le premier modèle de saillance de Koch et Ulman.

CHAPITRE 2

# La saillance visuelle :modéle de Itti et al

## 2.1 Introduction

Un grand nombre d'études propose des modèles qui permettent de prédire des fixations oculaires.Les modéles d'attention visuelle sont divisés en deux catégories principales : modéle "Bottom-Up" et modéle "Top-Down".
Dans cette chapitre , nous nous intéresserons au modèle ascendant qui est itti et al [11],nous vous expliquerons toutes les étapes pour obtenir la carte de saillance dans ce modéle.

## 2.2 La saillance visuelle

La saillance visuelle est la qualité unique qui donne lieu à la perception subjective d'un objet Extraordinaire par rapport à ses voisins, il a tout de suite retenu notre attention.

La plupart des modèles de recherche visuelle, qu'ils impliquent des mouvements De l'œil manifestes ou des changements d'attention cachés, sont basés sur le concept d'une carte de saillance (une carte bidimensionnelle explicite qui encode la proéminence ou la visibilité des objets dans l'environnement visuel).

La compétition entre les neurones de cette carte donne lieu à un seul emplacement gagnant qui correspond à la prochaine cible visitée. L'inhibition de cet emplacement permet automatiquement au système de s'occuper de l'emplacement le plus saillant suivant.

Le premier modèle bionique pour calculer la saillance a été développé par Koch et Ullman [16] [8]( figure 2.1),Ils ont utilisé (kochet ullman) un réseau de neurones pour combiner plusieurs cartes topographiques à partir de différents indicateurs d'image.

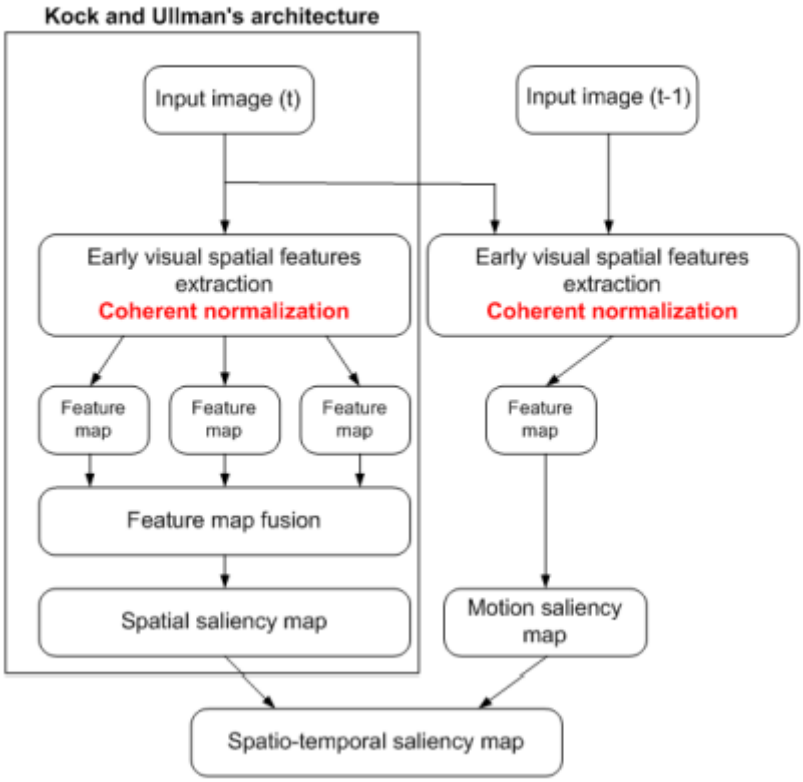



FIGURE 2.1 – modéle proposé basé sur Koch et Ullman [10]

La littérature sur l'attention visuelle s'est fortement développée depuis 1998, avec la publication de Itti et al. Le premier modèle de cartes de saillance [38].

## 2.3 modéle de Itti et al

Le modèle s'inspire du comportement et de la structure neuronale du système visuel des primates. Elle est depuis devenue une référence pour de nombreux modèles,c'est le système le plus populaire et le plus populaire.Qui est l'un des premiers modèles de calcul de l'attention qui a été améliorée constamment depuis sa création.

Le modèle de saillance visuelle ITTI est un modèle d'attention visuelle conçu sur la base du système nerveux visuel des premiers primates[11].Il se base sur la théorie de la saillance énoncée par Duncan et Humphreys[49]qui démontrent que le système visuel analyse les stimuli en les décomposant en caractéristiques visuelles : luminosité, couleur, orientation. Le modèle utilise d'abord la méthode d'échantillonnage gaussienne pour construire la pyramide gaussienne de la couleur, de la luminosité et de la direction de l'image, puis utilise la pyramide gaussienne pour calculer la carte caractéristique de luminosité, la carte des caractéristiques de couleur et la carte des caractéristiques de direction de l'image, et

enfin combine la carte des caractéristiques de différentes échelles pour obtenir la carte de visibilité d'intensité, de la couleur et de la orientation.

Le modèle combine les caractéristiques multi-échelles d'une image en une seule carte de saillance topographique. Il se concentre uniquement sur les processus ascendants " bottom-up".

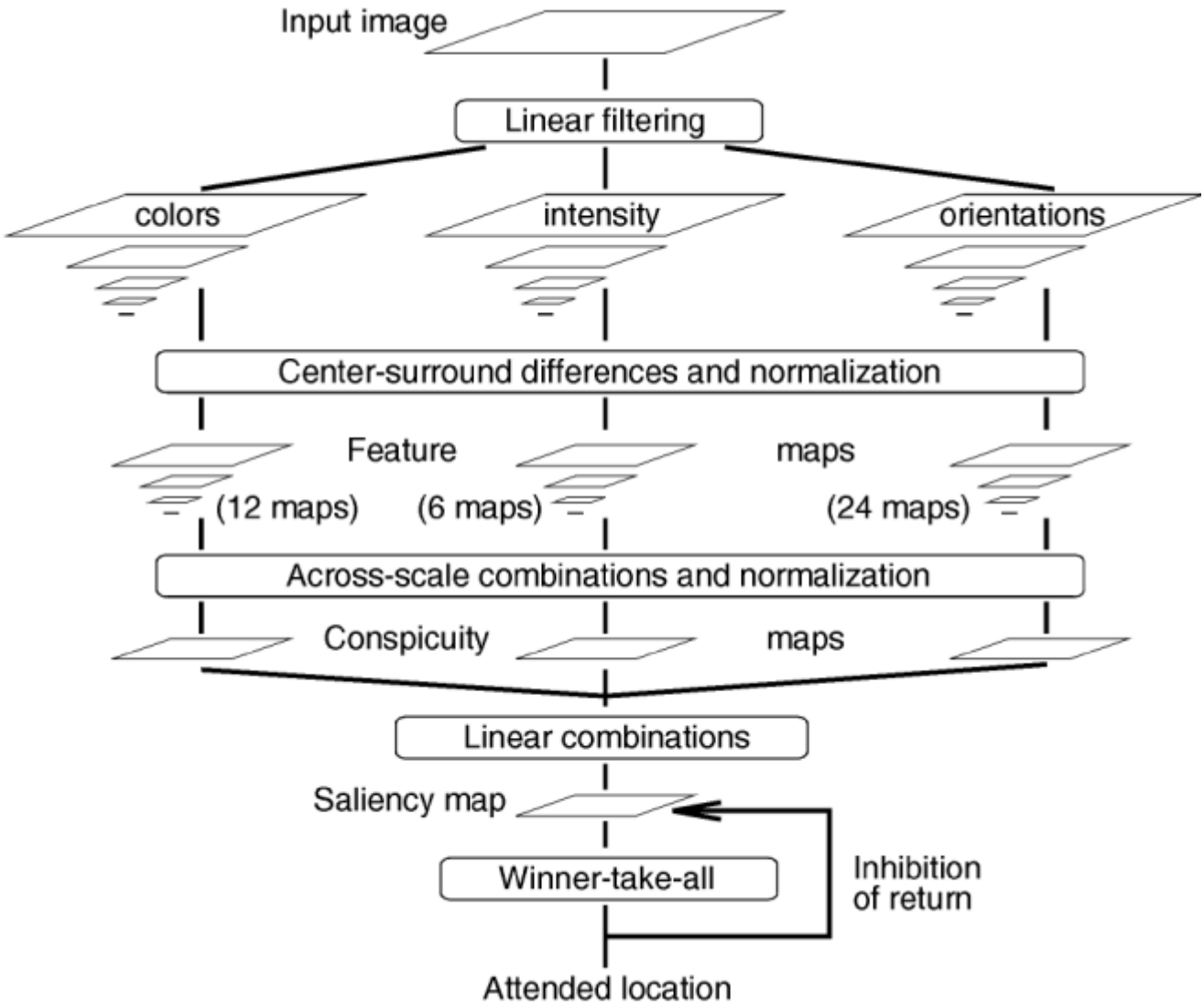



FIGURE 2.2 – Le modèle hiérarchique centralisé de Itti et al [11]

la carte de saillance du modèle est dotée de dynamiques internes qui génèrent des déplacements attentionnels. Ce modèle représente par conséquent un compte rendu complet de la saillance "bottom-up" et ne nécessite aucune orientation "top-down" pour déplacer l'attention[11].

Ce modèle est la base des modèles modernes et la référence la plus citée( (cité plus de 13199 fois),c'est pourquoi nous allons le décrire d'une manière détaillée dans cette chapitre.

Ce modéle Il extrait des caractéristiques visuelles regroupées en grandes familles de cartes d'attribut( l'intensité, la couleur, l'orientation).Une carte de visibilité est obtenue pour chaque famille à l'aide d'un opérateur "centre surround" [50]. Ces quatre cartes sont ensuite fusionnées en une carte maîtresse appelée carte de saillance .

**Récapitulatif de l'algorithme [51]** : L'architecture de ce modèle de bas en haut est basée sur les étapes suivantes :

— Calculer les pyramides gaussiennes pour :
  1. Intensité
  2. Couleur
  3. Orientation

— Combinez des images de différents niveaux des pyramides gaussiennes pour créer des cartes de visibilité

— Fusionner des cartes de visibilité pour créer une carte de saillance

— Supprimer la région la plus saillante, identifier l'autre région la plus saillante.

— Retour à l'étape 4.

### 2.3.1 Mode de fonctionnement

premièrement , plusieurs caractéristiques fondamentales telles que l'intensité, les couleurs opposées et les directions des bords sont extraites d'une image d'entrée, et une pyramide gaussienne est construite pour chaque caractéristique.En prenant des différences au niveau des pixels entre les échelles "center and surround" de la pyramide gaussienne, nous pouvons calculer des contrastes spatiaux multi-échelles pour des combinaisons de trois échelles centrales et deux différences d'échelle centre-voisins "center and surround".
Chaque différence centre-entourage est normalisée pour obtenir la carte de caractéristiques [52].

Les cartes des caractéristiquess avec le même canal caractéristiques sont intégrées dans une seule carte et à nouveau normalisées pour obtenir une carte de visibilité.La carte de saillance finale est une moyenne des cartes des caractiristiques.Enfin, Et au final, le mécanisme du gagnant rafle tout "winner-takes-all" est appliqué,pour sélectionne l'emplacement où la valeur de pixel de la carte de saillance est supérieure à tout autre emplacement.

Le modèle représente la création d'une carte de saillance en passant par plusieurs étapes, que nous détaillerons :

— Extraction des Premières Caractéristiques Visuelles.

— Cartes des caractéristiques (feature maps).

— Différence centre-voisins et normalisation.

— Cartes de visibilité ( Conspicuity Map) .

— "Across-scale" et normalisation.

— "Winner take all".

— Carte de saillance (map saliency).

#### 2.3.1.1 Les canaux de couleurs et leurs pyramides gaussiennes

Avec r, g et b étant les canaux rouge, vert et bleu de l'image d'entrée, une image d'intensité I est obtenu comme **I** = (r + g + b)/3. I est utilisé pour créer une pyramide gaussienne I($\sigma$) où $\sigma$ [0..8]. Quatre canaux de couleur sont créés[11] :

1. R = r (g + b)/2 :pour le rouge.
2. G = g (r + b)/2 :pour le vert.
3. B = b (r + g)/2 : pour le bleu .
4. Y = (r + g)/2 - |r −g |/2-b pour le jaune.

Quatre pyramides gaussiennes G($\sigma$),R($\sigma$),B($\sigma$),Y($\sigma$) sont créés à partir de ces canaux de couleur.

#### 2.3.1.2 les cartes de caractéristiques et "Center-surround"

les différences centre-voisins, qui jouent un rôle central dans la modélisation de l'attention ascendante [53].Les différences "centre-voisins" sont implementées dans cette modéle comme une différence entre des échelles fines (centre) et des échelles grossières (voisins). Atteindre la différence centre-voisins grâce à la différence à travers l'échelle [12].

Le centre est le pixel d'échelle **C** ∈ 2, 3, 4 et les voisins sont les pixels correspondants à l'échelle s = c + $\delta$ , [3, 4].

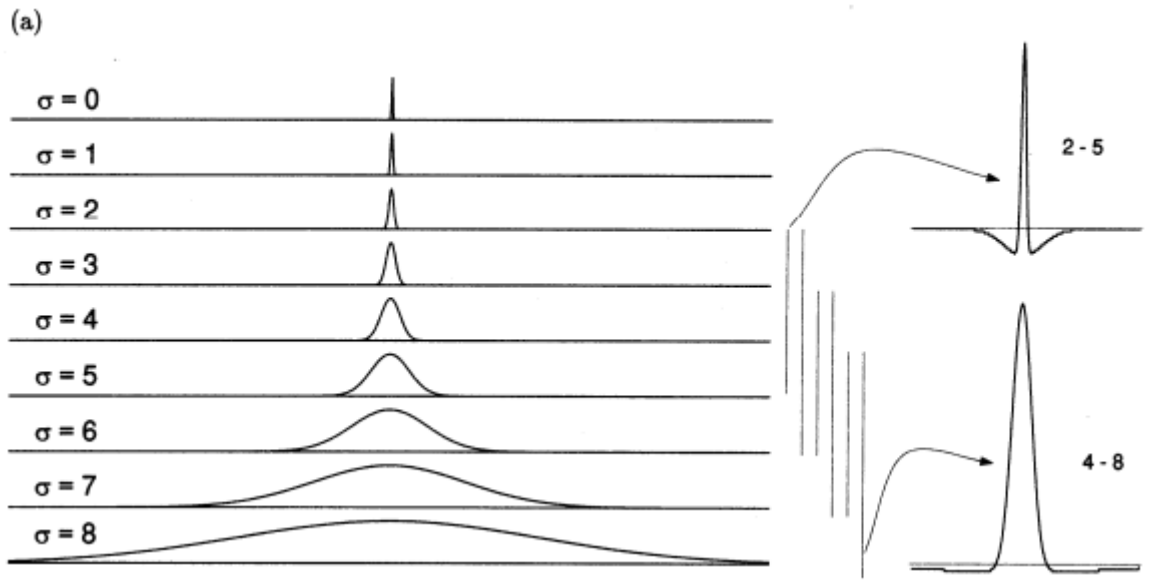



FIGURE 2.3 – Largeurs de pixel gaussiennes pour les neuf échelles utilisées dans le modèle. L'échelle s=0 correspond à l'image d'origine, et chaque échelle suivante est plus grossière d'un facteur 2. Deux exemples des six types de champs récepteurs centre-surround sont présentés, pour les paires d'échelles 2–5 et 4–8 [12]

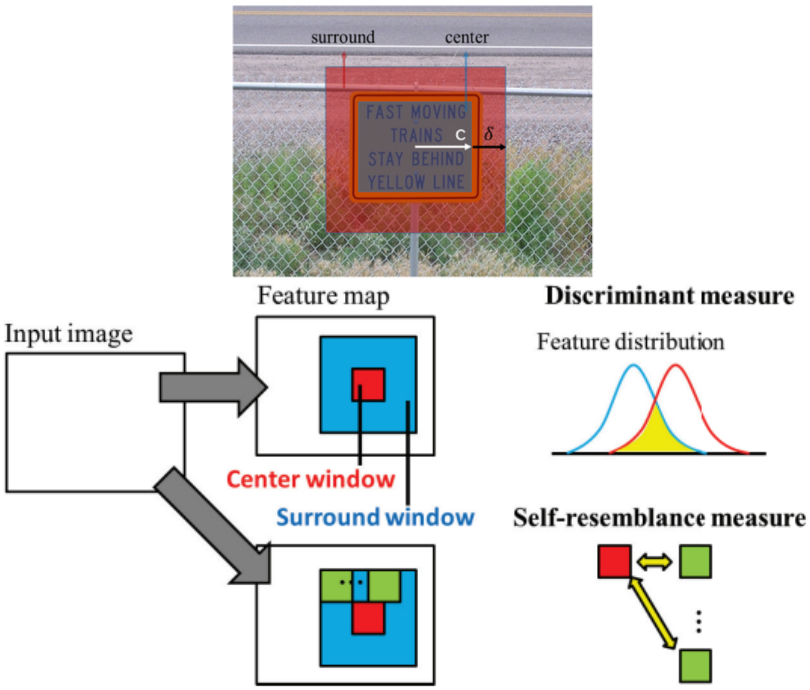



FIGURE 2.4 – les mesures centre-entourage proposées par Gao et Vasconcelos [13] [14]

"⊖" est obtenu par interpolation à l'échelle la plus fine et soustraction point par point.

**Cartes de caractéristiques du contrast d'intensité** :

Le premier ensemble de cartes de caractéristiques concerne le contraste d'intensité, qui, chez les mammifères, est détecté par des neurones sensibles soit aux centres sombres sur les environnements lumineux, soit aux centres lumineux sur les environnements sombres [54].

Les deux types de sensibilités sont simultanément calculé en un ensemble de six cartes I(c, s) [11] ,avec c ∈ 2, 3, 4 et s = c + $\delta, \delta \in 3,4$ :

$$I(c, s) = |I(c) - I(s)| \qquad (2.1).$$

**Cartes de caractéristiques de couleurs** :

Semblable à la carte d'intensité, mais utilisant des canaux de couleur différents Le deuxième ensemble des cartes de caractéristiques est construit,les neurones sont excités par une couleur et inhibés par une autre le contraire est vrai pour les voisins. Les adversaires spatiaux et chromatiques existent dans le cortex visuel humain pour vert/rouge , bleu/jaune, rouge/vert , jaune/bleu (figure 2.5). Donc les cartes RG(c, s) et BY (c, s) sont crées :

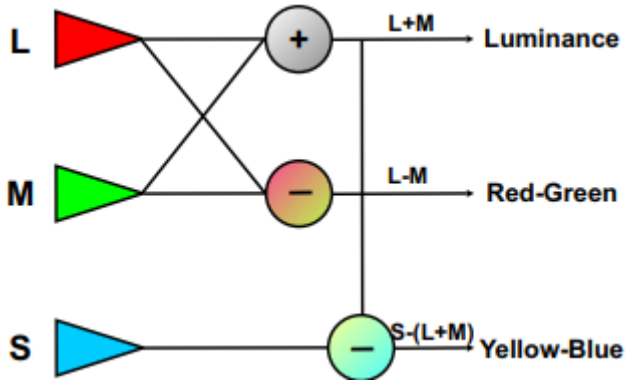



FIGURE 2.5 – Schéma des interconnexions des cônes dans la rétine, conduisant à des signaux de type adversaire [15]

$$BY(c,s) = |(B(c) - Y(c)) \ominus (Y(s) - B(s))| \qquad (2.2)$$

$$RG(c,s) = |(R(c) - G(c)) \ominus (G(s) - R(s))| \qquad (2.3).$$

**Cartes de caractéristiques de l'intensité** :

Le troisième ensemble de cartes de caractéristiques est obtenu par intensité I en utilisant filtres linéaire (filtre Gabor) O $(\sigma, \theta)$,pour les orientations :
$\theta \in 0\circ, 45\circ, 90\circ, 135\circ$

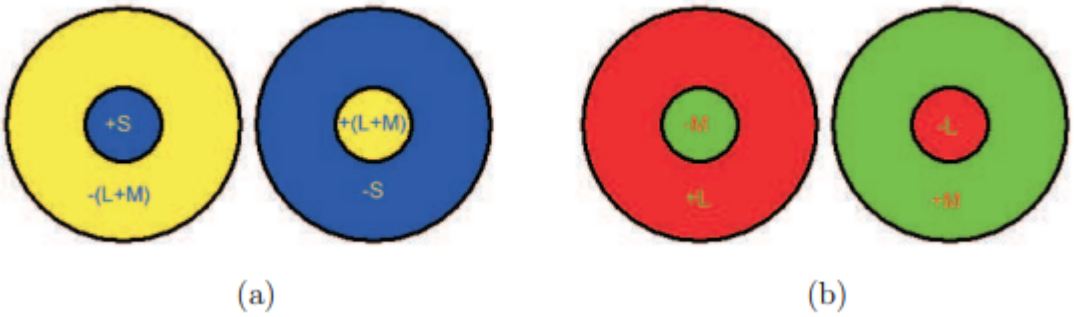



FIGURE 2.6 – Exemple de champs récepteurs antagonistes centre-voisines typiques : (a) champs récepteurs jaune-bleu centrés ; (b) champs récepteurs rouge-vert décentrés [15] .


Les cartes de caractéristiques d'orientation, **O**(c, s, $\theta$), encodent, en tant que groupe, le contraste d'orientation local entre les échelles "center surround" [11] :

$$O(c,s,\theta) = |O(c,\theta) \ominus O(s,\theta)|. \qquad (2.4)$$

c et s à nouveau similaires à l'intensité.

Au total, 42 cartes de caractéristiques sont calculées : six pour l'intensité, 12 pour la couleur(6 pour **RG** +6 pour **BY**)et 24 pour l'orientation.

#### 2.3.1.3 Les cartes de visibilite

Le modèle crée trois cartes de visibilites pour l'intensité, la couleur et l'orientation combinant les cartes de caractéristiques créées,les cartes de caractéristiques sont regroupées en trois cartes de visibilité à l'échelle 4 ($\sigma = 4$).

1. $\bar{I}$ pour l'intensité.
2. $\bar{C}$ pour la couleur
3. $\bar{O}$ pour l'orientation

Ces cartes sont obtenus par une addition across-scale "$\bigoplus$" qui consiste a réduire chaque carte a l'échelle 4 et une addition point par point :

$$\bar{I} = \bigoplus_{c=2}^{4} \bigoplus_{c+4}^{s=c+3}. \qquad (2.5)$$

$$\bar{C} = \bigoplus_{c=2}^{4} \bigoplus_{c+4}^{s=c+3} [N(RG(c,s)) + N(BY(c,s))]. \qquad (2.6)$$

Pour l'orientation, quatre cartes intermédiaires sont d'abord créées en combinant les six cartes de caractéristiques pour un q donné et sont ensuite combinées en une seule carte de visibilité d'orientation :

$$\bar{O} = \sum_{\theta \in 0đ,45đ,90đ,135đ} \cdot \bigoplus_{c+4}^{s=c+3} N(O(c,s,\theta))). \qquad (2.6)$$

#### 2.3.1.4 La carte de saillance

Les trois cartes de visibilité sont normalisées et additionnées pour obtenir la carte de saillance finale S :

$$S = \frac{1}{3}((N(\bar{I}) + (N\bar{C}) + (N\bar{O})) \qquad (2.7)$$

La carte de saillance finale est une moyenne des cartes de visibilité.

**Opérateur de normalisation** [55]
Sur la base du modèle d'activité dans la carte de saillance, dans laquelle pratiquement chaque barre provoque un fort pic d'activité, il serait difficile de choisir un emplacement comme étant clairement plus intéressant et digne d'attention que n'importe lequel des autres. Intuitivement, on pourrait donc vouloir appliquer un opérateur de normalisation N(.) qui donnerait un poids global très faible à la contribution de cette carte à la carte de saillance finale [11][12][56].

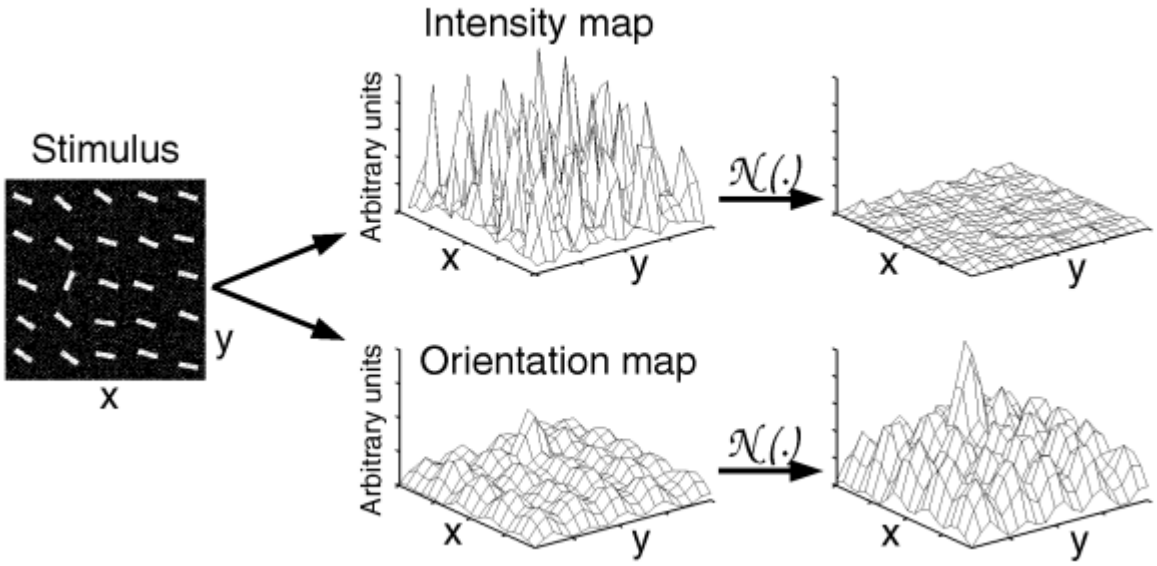



FIGURE 2.7 – L'opérateur de normalisation N(.)[11]

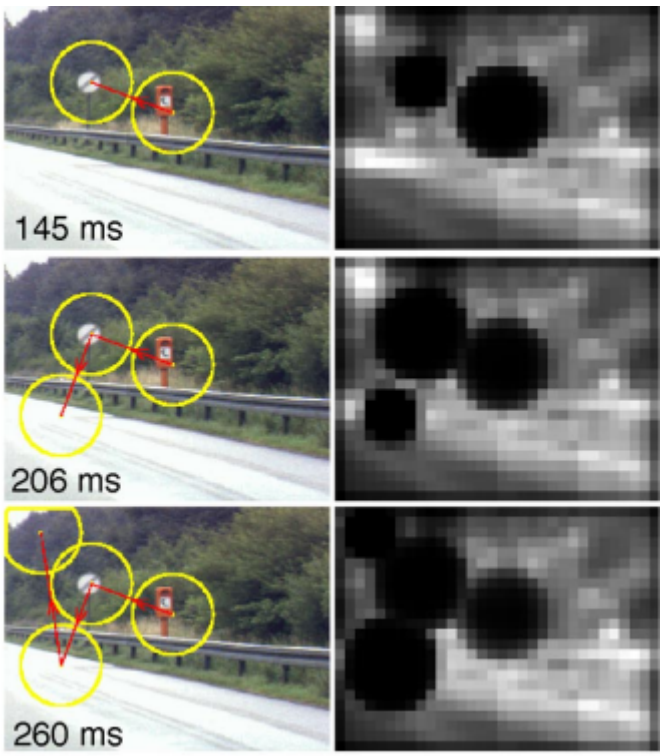



FIGURE 2.8 – détourner l'attention [11]

#### 2.3.1.5 Winner-take-all

A un moment donné, un seul emplacement est sélectionné dans la première représentation et copié dans la représentation centrale.

Le FOA (focus of attention) est déplacé vers l'emplacement du neurone gagnant. L'inhibition globale du WTA est déclenchée et inhibe complètement (réinitialise) tous les neurones WTA. L'inhibition locale est activée de manière transitoire dans le CS(Carte de saillance), dans une zone de la taille et du nouvel emplacement du FOA [57][58].

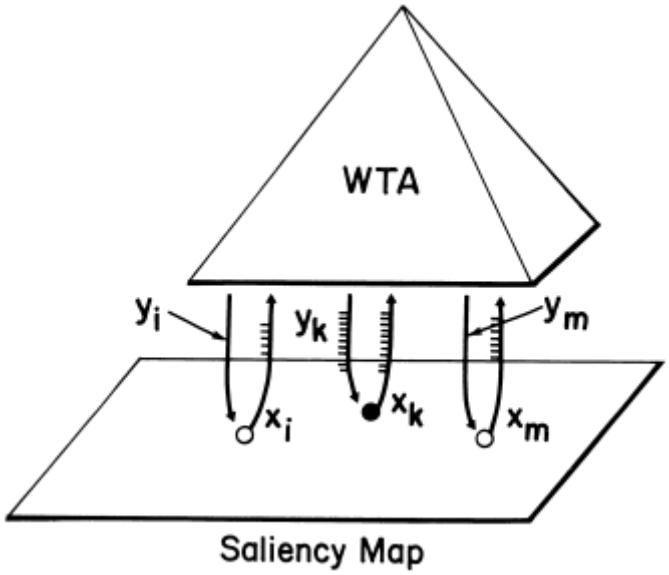



FIGURE 2.9 – Un dessin schématique illustrant le Winner-Take-All (WTA)[16]

Le réseau "winner-take-all" (WTA) détecte l'emplacement le plus saillant et dirige l'attention vers celui-ci, de sorte que seules les caractéristiques de cet emplacement atteignent une représentation plus centrale pour une analyse plus approfondie[12].

## 2.4 Conclusion

Dans ce chapitre, le modèle de saillance d'tti est expliqué en fonction de son mécanisme d'obtenir la carte de saillance.Nous parlons sur la conception et les algotithmes de ce modèle dans le chapitre suivant.

# CHAPITRE 3 Conception

Itti et al[16].Ont proposé un modèle similaire d'attention visuelle ascendante ("Bottom-Up") qui a montré de bons résultats pour simuler la focalisation humaine de l'attention visuelle. Ils ont utilisé le calcul des caractéristiques centre-voisins et une nouvelle stratégie de combinaison. Plus tard, ils ont proposé une nouvelle stratégie de combinaison pour améliorer leur modèle de carte de saillance[59].

Ce modéle computationnel repose sur le calcul à différentes échelles spatiales de variations d'intensite , de couleurs et d'orientation. Les cartes résultantes sont combinées et normalisees d'abord entre échelles puis entre caractéristiques pour donner la carte de saillance complete [60].

Dans ce chapitre, nous allons présenter l'architecture générale et détaillée de l'application basé sur le modéle d'itti al grâce à un ensemble d'algorithmes et de schémas.

## 3.1 Schéma général de notre application :

Le schéma représente les différentes étapes d'une méthode pour le calcul des cartes de saillance. Tout d'abord, nous parcourons le chemin de l'image souhaitée. Si l'image n'est pas trouvée, un message sera affiché informant l'utilisateur que l'image n'existe pas, sinon la carte de saillance sera calculé.

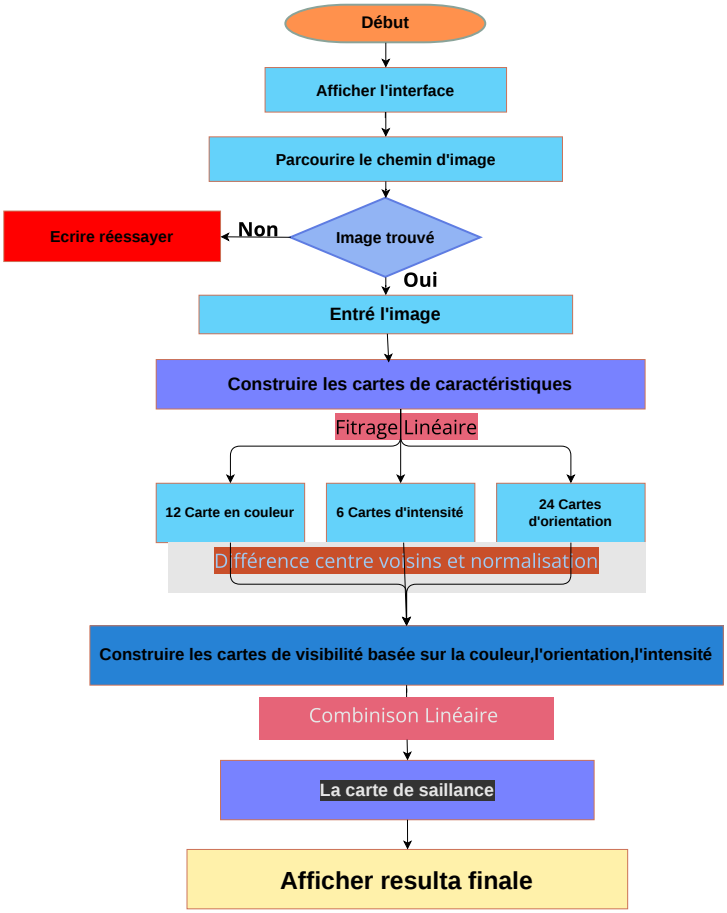



FIGURE 3.1 – Schéma général de construction des cartes saillances

L'entrée de l'algorithme est une image couleur statique de comprenant trois canaux de r, g et b. Tout d'abord, les trois canaux de r, g et b doivent être sous échantillonnés pour obtenir une image à trois canaux à neuf échelles.Les pyramides gaussienne de l'intesité,de couleur peut alors être construite, calculée à neuf échelles, où elle doit être normalisée.Enfin, une pyramide de direction de Gabor est construite à l'aide d'un filtre de Gabor.Après avoir obtenu les pyramides gaussienne , la méthode center-voisins pour calculer la carte de caractéristique correspondante.Ainsi, les cartes de caractéristiques résultante totale a des 46 cartes caractéristiques. L'opération de normalisation de la carte des caractéristiques,ce qui peut améliorer la carte des caractéristique.Pour chaque caractéristique (orientation,intensité ,couleur) toutes les cartes de trait sont fusionnées pour créer une carte de visibilité . Cette carte est aussi normalisée par la normalisation . Enfin, la carte de saillance finale est construite en fusionnant les trois cartes de visibilité.

## 3.2 Conception détaillé de l'application

Nous allons présenter la conception détaillée, le niveau de détail de la conception d'illustrer sur les schéma tous les éléments de conception nécessaires à la l'implementation du l'application.

### 3.2.1 construire les pyramides de Gaussien

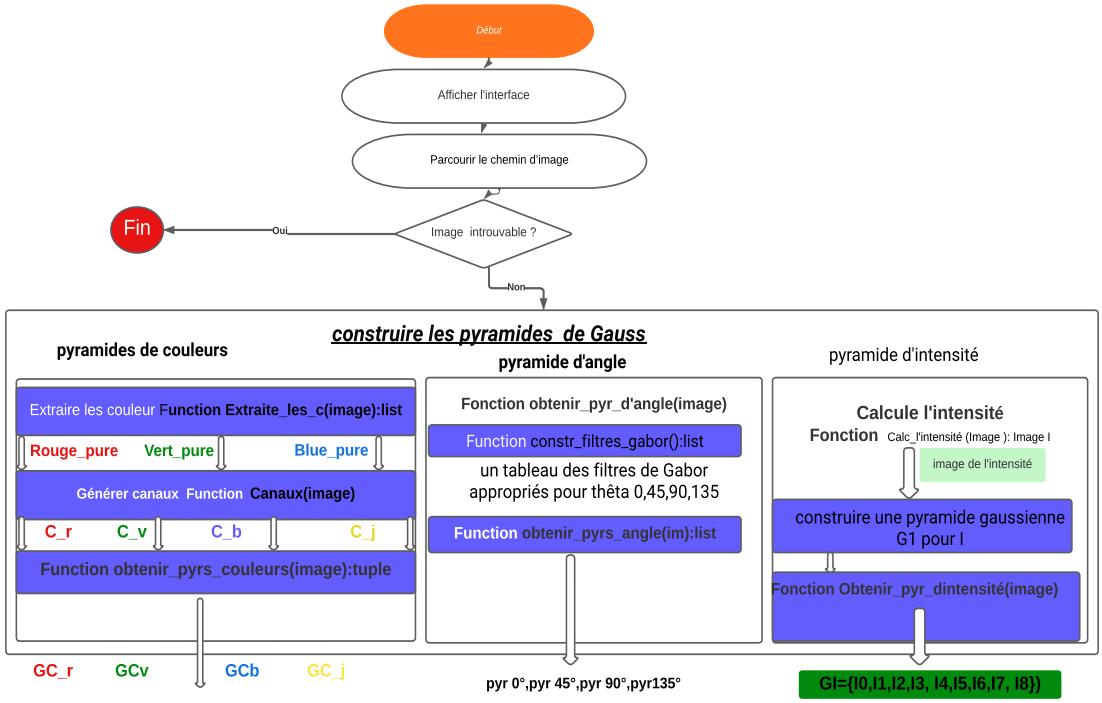



FIGURE 3.2 – Création des pyramides gaussiennes

**Les sorties de cette étape** :

— fonction obtenir_pyrs_couleurs :un tableau des 4 pyramides de couleurs, rouge, vert, bleu, jaune.
— fonction obtenir_pyrs_d'intensité :renvoie la pyramide gaussienne d'intensité.
— fonction btenir_pyr_d'angle : un tableau des filtres de Gabor appropriés pour thêta = 0,45,90,135

### 3.2.2 Obtenir les cartes de caractéristiques

**Les entrée de cette étape** :

— La pyramide des angles pour 0,45,90,135.
— La pyramide en couleur r,v,b,j.
— La pyramid d'intensité.

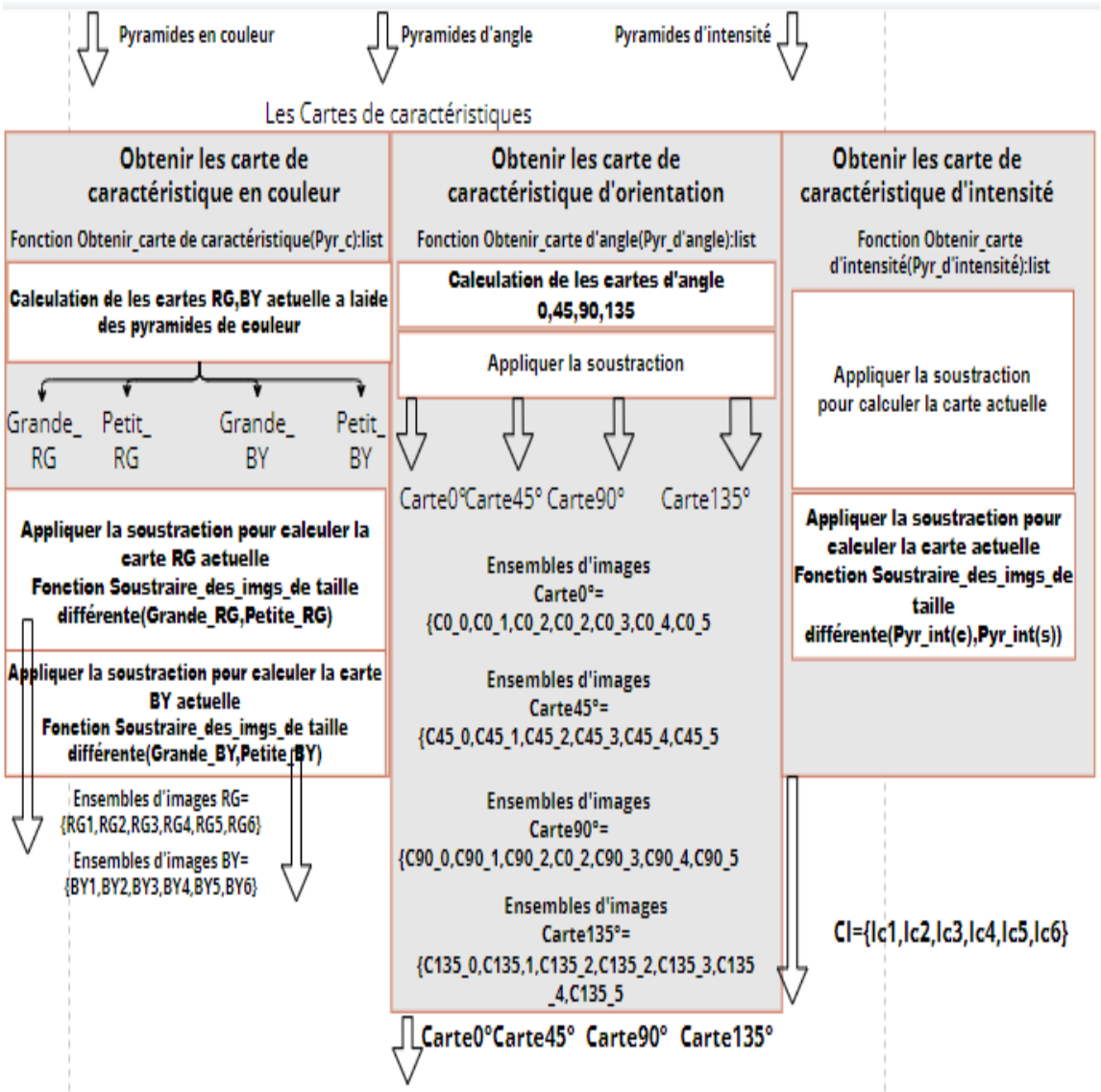



FIGURE 3.3 – Création des cartes de caractéristique

**Les sorties de cette étape** :
— les 12 cartes de caractéristiques de couleurs sous forme de tableau, le premier tableau est de 6 entrées sont la carte RG et les 6 autres sont les cartes BY.
— 0,45,90,135 cartes de caractéristiques(6*4 cartes).
— cartes de caractéristiques d'intensité.

### 3.2.3 Obtenir les cartes de visibilites

**Les entrée de cette étape** :
— cartes de caractéristiques d'intensité.
— cartes de caractéristiques d'ongle.
— cartes de caractéristiques en couleur.

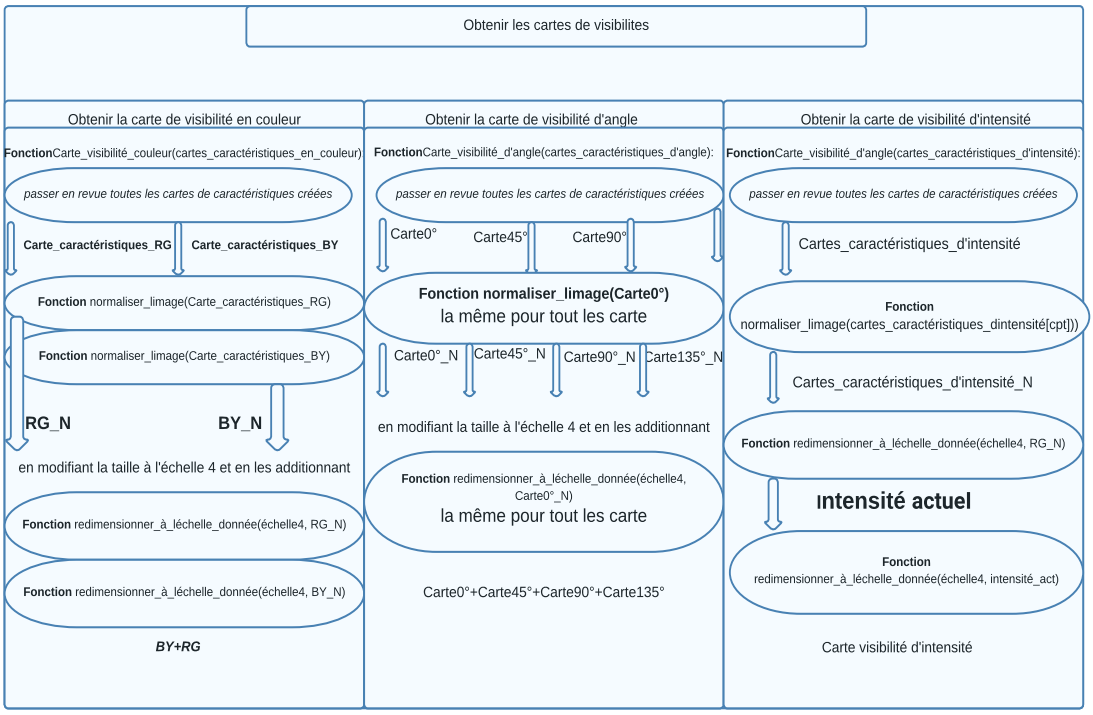



FIGURE 3.4 – Création des cartes de visibilité

**Les sorties de cette étape** :
— Les cartes de visibilité(en couleur,d'angle,d'intensité).

### 3.2.4 Obtenir la carte de saillance

**Les entrée de cette étape** :Les trois cartes de visibilité.

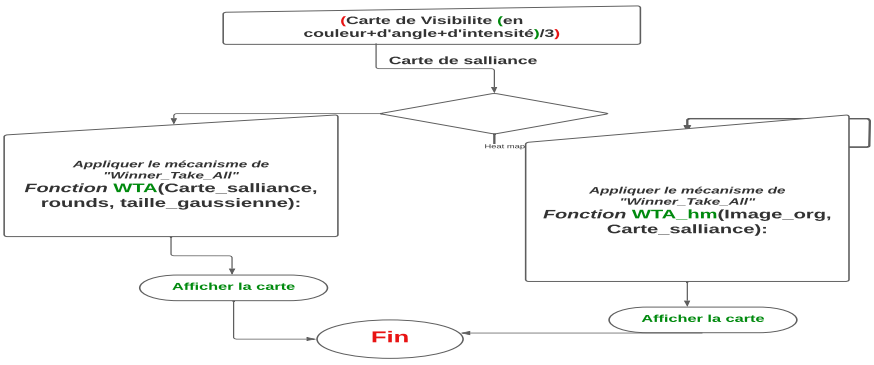



FIGURE 3.5 – Le mécanisme pour obtenir la carte de saillance finale

**Le sortie de cette étape** :La carte de saillance.

Tout d'abord, l'utilisateur entre le chemin de l'image ou appuie sur le bouton afin de récupérer une image de l'appareil,Nous supposons que l'image choisie par l'utilisateur est .Ensuite, nous redimensionnons l'image à une certaine échelle(640 x 480). Extraire les couleurs de l'image comme suit :

1. rouge_pur = Extraire_Rouge(image).
2. vert_pur = Extraire_Vert(image).
3. bleu_pur = Extraire_blue(image).

Pour l'extraction des couleurs, l'image est convertie en espace colorimétrique rouge-vert-bleu-jaune,créant les canaux de couleur largement accordés :

1. canal_rouge = rouge_pur - (vert_pur + bleu_pur) / 2.
2. canal_vert = vert_pur-(rouge_pur + bleu_pur) / 2
3. canal_bleu = bleu_pur - (rouge_pur + vert_pur) / 2.
4. canal_jaune = (rouge_pur + vert_pur) / 2 - abs(rouge_pur - vert_pur)/2 - bleu_pur.

Les informations sur l'orientation locale sont extraites à l'aide du filtre de Gabor sous quatre angles.On obtient un tableau des filtres de Gabor appropriés pour thêta = 0,45,90,135.

La phase suivante consiste à la création de pyramides gaussienne pour : :

— Intensité(figure 4.2).

— Color (figures(4.7,4.8,4.9,4.10)).

— Orientation(figure(4.3,4.4,4.5,4.6))

C'est à l'aide de la fonction construire_pyr_g(im, niveaux max, filtre gaussien) : qui construire une pyramide d'image gaussienne ,qui permet de une pyramide, à partir de l'image I, à 8 échelles et de renvoyer 8 images I0,I1,I2,I3,I4,I5 ,I6,I7.

1. paramètre im :l'image d'entrée.
2. paramètre niveaux max : à quelle profondeur la pyramide doit-elle aller.
3. paramètre filtre gaussien :le filtre à utiliser pour le flou .

L'utilisation de l'échantillonnage supérieur ou inférieur détermine ce qu'on appelle le sens de l'opération. Ainsi la direction de réduction diminue la résolution de l'image (down-sample), tandis que l'expansion agit dans le sens inverse.
En pratique, la première étape lors de l'obtention de pyramides gaussiennes est de calculer un filtre discret pour représenter le noyau gaussien.Compte tenu de la séparabilité de ce noyau, tout le processus est considérablement simplifié en ne considérant qu'une seule dimension obtenir_un_filtre_gaussien(Taille) :
La fonction obtient une taille et renvoie le filtre gaussien 1D correspondant.

Imaginer la pyramide comme un ensemble de couches dans lesquelles plus la couche est haute, plus la taille est petite(level max).

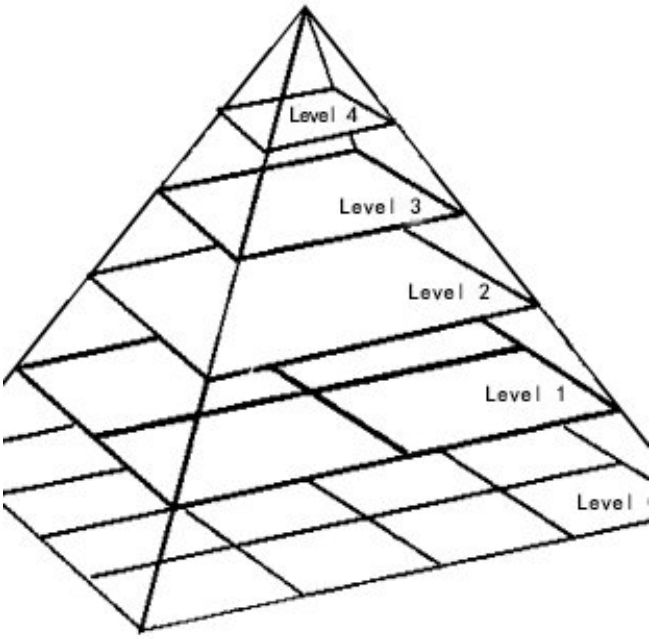



FIGURE 3.6 – Pyramide Gaussienne

La pyramide gaussienne se compose d'images filtrées passe-bas et à densité réduite (c'est-à-dire sous-échantillonnées) du niveau précédent de la pyramide, où le niveau de base est défini comme l'image d'origine.
l'image originale bidimensionnelle soit notée I(x, y).

Pour obtenir les pyramides dans une dimension, nous utilisons un filtre à 5 robinets. Les cinq poids du filtre G5 sont représentés sur la 5ème ligne du triangle. Pour éviter un débordement dans la convolution, ces coefficients sont normalisés par la somme totale (16).Ainsi le filtre est donné par
$G_5 = \frac{1}{16}[1\ 4\ 6\ 4\ 1]$.
G5 sont représentés sur la 5ème ligne du triangle(Triangle Pascal).
**2ème extension :**

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

La pyramide gaussienne est définie récursivement comme suit [61],

G_o(x, y) = I(x, y), pour le niveau, l = 0.

$$G_l(x,y) = \sum_{m=-2}^{2}\sum_{n=-2}^{2} w(m, G_{l-1}(2x+m, 2y+n), sinon.$$

Étant donné que les niveaux diffèrent dans leur densité d'échantillon, il est nécessaire d'interpoler de nouvelles valeurs d'échantillon entre celles d'un niveau donné avant que ce niveau ne soit soustrait du niveau immédiatement inférieur.

**Interpolation d'images** :
Étant donné les échantillons F d'une image f, la tâche d'interpolation consiste à calculer une valeur pour f(x,y) même pour ceux (x,y) qui ne sont pas des points d'échantillonnage. L'interpolation fonctionne en utilisant des données connues pour estimer des valeurs à des points inconnus. Par exemple : si vous vouliez comprendre l'intensité des pixels d'une image à un emplacement sélectionné dans la grille (disons la coordonnée (x, y), mais seulement (x-1,y-1) et (x+1,y+1 ) sont connues, vous estimerez la valeur en (x, y) à l'aide d'une interpolation linéaire. Plus la quantité de valeurs déjà connues est élevée, plus la précision de la valeur de pixel estimée sera élevée.
Les algorithmes d'interpolation sont principalement utilisés pour redimensionner et déformer une image haute résolution en une image haute résolution. Il existe différents algorithmes d'interpolation, l'un d'eux est l'interpolation bicubique. Il existe différents algorithmes d'interpolation, l'un d'eux est Interpolation linéaire.

**Interpolation linéaire** [62] :
Reliez les points. Entre les points d'échantillonnage adjacents k et k+1, nous supposons que la fonction est une fonction linéaire et donc dans cet intervalle [k, k+1] nous avons :

k≤x≤k+1 : $\hat{f}\longrightarrow(x) = (1-(x-sk))F(k) + (x-k)F(k+1).$

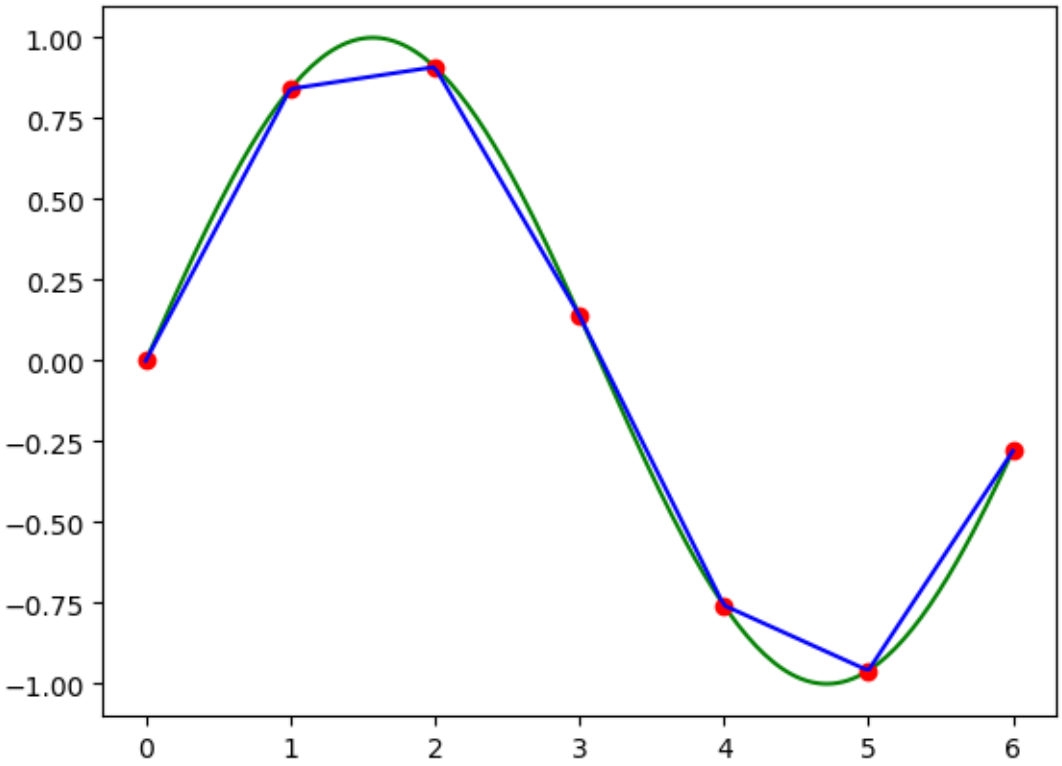
Bien qu'il s'agisse d'une méthode d'interpolation simple, l'interpolation linéaire est très souvent utilisée dans la pratique. Il est simple à mettre en œuvre et très rapide.

D'abord, on applique la fonction expand_im(im,taille_dest) qui permet de retourner l'image agrandie avec une convolution du filtre gaussien.

Ensuite, on passe à l'application de la fonction soustraire_des_taille_différente(grande_im,petit_im) :

Routerne : la différence entre les deux images pixel par pixel, après mise à l'échelle 'petit_image' par interpolation.
appeler la fonction (expand_im).
La phase suivante consiste en la création de cartes de cartes de caractéristiques :
L'organisation centre-voisins " du champ récepteur des neurones ganglionnaires est mise en œuvre sous forme de différence de gaussien "Gaussian difference" entre les échelles plus fines et plus grossières d'une pyramide appelée carte de caractéristiques.

1. obtenir_des_cartes_caractéristiques_en_couleur() :
   Calculée en soustrayant tous les combos entre le niveau (1,2,3) et (6,7),c'est-à-dire (7e et 8e niveaux).




Routerne : "12 cartes de caractéristique en couleur sous forme de tableau, le premier tableau est de 6 entrées sont la carte RG(figure 4.11) et les 6 autres sont les cartes BY(figure 4.12).

**L'algorithme** :

(a) Différence de canaux de couleur à l'échelle **c**

(b) Différence de canaux de couleur à l'échelle **s**

(c) Suréchantillonner pour effectuer une soustraction entre différentes échelles appeler la fonction. soustraire_des_taille_différente(Grand RG,petit RG). soustraire_des_taille_différente(Grand BY,petit BY).

```
Fonction centre_surround() :list
  for (i dans c) do
    for (j dans s) do
        redimensionner(pyramid[i + j], pyr_s, pyr_c.taille());
        pyr_c ← pyramid[i];
        pyr_s ← pyramid[j];
  soustraire_des_imgs_de_taille_différente(diff(pyr_c, pyr_s, diff))
        différences←Ajouter(diff);
    end for
  end for
```

2. obtenir_des_cartes_de_caractéristiques_d'intensité(pyramide d'intensité)(figure 4.1) : paramètre image.
3. obtenir_la_carte_des_caractéristiques_d'angle() : return : 0,45,90,135 cartes de caractéristiques(figure).

Le modèle crée trois cartes de visibilité pour l'intensité, la couleur et l'orientation combinant les cartes de caractéristiques créées.

La phase suivante consiste en la création de les cartes carte de visibilité :

1. en couleur(figure 4.17).
2. d'angle (figure 4.18).
3. d'intensité(figure 4.16).

Normalisez et ajoutez les cartes de visibilité pour obtenir la carte de saillance :

**La Normalisation** :
On applique la fonction normaliser_l'image(image) :

1. normalisation des valeurs d'image entre 0 et l'image max.
2. trouver des maximums globaux et m maximums locaux
3. Retourne l'image normalisée.

La carte de saillance finale est une moyenne des cartes visibilité.

L'objectif de la première étape est d'extraire trois caractéristiques de bas niveau, l'intensité, la couleur et l'orientation, puis d'utiliser la différence de gaussienne (DOG) pour former un total de quarante cartes d'activation. Enfin, un opérateur linéaire est utilisé pour normaliser ces cartes, dans lequel l'emplacement le plus saillant est sélectionné par le "winner-take-all" pour générer une carte de saillance.

### 3.2.5 Algorithmes

#### 3.2.5.1 les algorithmes pour créer les pyramide

L'entrée visuelle fournie sous la forme d'images couleur statiques est d'abord calculée par un ensemble d'opérations linéaires "center-surround" apparentées aux champs visuels réceptifs. Il se décompose en trois canaux : couleurs, intensité, orientations. Ce processus produit un ensemble des cartes de caractéristiques :

**Fonction** Oobtenir_un_filtre_gaussien(Taille)
**var** base,g_filtre ;tableaux d'entiers dimension
▷ créer la base pour commencer à **convoluer** [1,1] pour obtenir le filtre gaussien.
**if** $taille = 1$ **then**
**Retourne**1.
**end if**
$base[0] \leftarrow 1$
$base[1] \leftarrow 1$
$g\_filtre[0] \leftarrow 1$
$g\_filtre[1] \leftarrow 1$
$i \leftarrow 0$
▷ quantité convolante de fois pour obtenir un filtre de bonne taille.
**while** $i < (taille - 2)$ **do**
g_filtre $\leftarrow convoluer(base, g\_filtre)$
$i++$
**end while**
▷ Retourne : normalize to sum ←1 g_filtre, et remodelé('reshape') pour être utilisé pour la convolution 2d
**Retourne** g_filter/sum(g_filter).remodeler(1,taille).

L'opérateur de convolution dans le traitement du signal, où il modélise l'effet d'un système linéaire invariant dans le temps sur un signal . En théorie des probabilités, la somme de deux variables aléatoires indépendantes est distribuée selon la convolution de leurs distributions individuelles.

Si _filtre est plus long que (base), les tableaux sont échangés avant le calcul.

**Remarques** :
L'opération de convolution discrète est définie comme :

$$(g\_filtre * base) = \sum_{m=-\infty}^{\infty} base_m * g\_filtre_{n-m}. \tag{3.1}$$

**Fonction** obtenir_pyramides_de_couleurs(image)

▷ la fonction crée un tuple de 4 pyramides une pour chaque canal de couleur - ROUGE, VERT, BLEU, JAUNE

▷ isoler les couleurs de l'image

rouge pur ← Extraire_Rouge(image);
vert pur ← Extraire_Vert(image);
blue pur ← Extraire_Blue(image);

▷ créant les canaux de couleur largement réglés.

canal rouge ← rouge pur - (vert pur + bleu pur) / 2.
canal vert ← vert pur-(rouge pur + bleu pur) / 2.
canal bleu ← bleu pur - (rouge pur + vert pur) / 2.
canal jaune ← (rouge pur + vert pur) / 2 - abs(rougepur - vert pur)/2 - bleu pur.

▷ créer les pyramides de couleurs.

$pyr_rouge \leftarrow construire_pyr(canalrouge, Profondeur_$pyr,Taille_G)pyr_vert← $construire\_pyr(canalvert$,Profondeur_pyr,Taille_G)
$pyr\_blue \leftarrow construire\_pyr(canalblue$,Profondeur_pyr,Taille_G) pyr_Jaune← $construire_pyr(canalJaune, Profondeur_$pyr,Taille_G)

**Retourne** $pyr\_rouge, pyr\_blue, pyr\_Jaune, pyr\_vert$

Les quatre couleurs V (vert), Y (jaune) ,R (rouge),B (bleu) sont extraites selon les équations suivantes :

$$R = r - \frac{v+b}{2}. \tag{3.2}$$

$$V = v - \frac{r+b}{2}. \tag{3.3}$$

$$B = b - \frac{r+v}{2}. \tag{3.4}$$

$$Y = \frac{r+v}{2} - \frac{|r-v|}{2} - b. \tag{3.5}$$

Ensuite, chaque couleur est décomposée par une pyramide passe-bas. Donc, pour 4 couleurs, il y a 4 pyramides R($\sigma$), V($\sigma$), B($\sigma$), Y ($\sigma$).

**Fonction** obtenir_pyramides_d'ntensité(image) :la pyramide gaussienne d'intensité

▷ **Retourne** :la pyramide gaussienne d'intensité.

im_d'intensité←Extraire_Rouge(image)+Extraire_Vert(image)+Extraire_Blue(image);
im_d'intensité ← im_d'intensité/3;
**Retourne** im_d'intensité;

L'intensité correspond à la moyenne des trois canaux r, v, b représentés dans l'espace RvB .

$$I = \frac{r+v+b}{3}. \tag{3.6}$$

L'intensité **I** est ensuite décomposée par une pyramide passe-bas multirésolution a 8 niveaux . Ainsi, on obtient une pyramide I($\sigma$) où  représente la résolution, $\sigma$  [0..8].

**Fonction** obtenir_pyramides_d'angle(image) :la pyramide des angles pour 0,45,90,135
  **var** pyramid_dg :list ;
  filter_gabor← construire_filtre_gabor() ;
  **while** kernel_thêta in filter_gabor **do**
    filter_im_ang ←filter2D(im_d'intensité, kernel_thêta) ;
    pyramid_dg←Ajouter(filter_im_ang,Profondeur_pyr,Taille_G)
  **end while**
**Retourne** pyramid_dg

Les caractéristiques de direction sont principalement obtenues en utilisant des filtres gabor pour filtrer les caractéristiques des couleurs de l'image dans quatre directions principales de 0 °, 45 °, 90 °, 135 °. Par conséquent, le filtre Gabor peut bien simuler les caractéristiques de traitement du signal de cellules simples dans le cortex visuel humain.

#### 3.2.5.2 les algorithmes pour créer les cartes de caractéristique

le contraste est extrait en effectuant la différence entre les valeurs a différents niveaux d'une pyramide.

Commençons par l'algorithme d'agrandissement d'image

Fonction agrandir_limage(image,dest$_taille$) : $list = 0$
▷ la Fnc obtient une image Routeurnel ' image agrandie avec une convolution du filtre gaussien .

algorithme spécifique de soustraction à échelle croisée est le suivant : en interpolant linéairement l'image représentant la plus petite échelle de l'information d'arrière-plan environnante, de sorte qu'elle ait la même taille que l'image de la plus grande échelle représentant l'information centrale, puis en effectuant une opération de soustraction point à point, c'est-à-dire l'opération de différence périphérique centrale, une telle opération de soustraction à échelle transversale est représentée par le symbole .

En général, les fonctions de carte de caractéristique sont écrites comme suit :
L'organisation 'Center-surrounde' du champ récepteur des neurones ganglionnaires est mise en œuvre sous forme de différence de gaussien entre les échelles plus fines et plus grossières d'une pyramide appelée carte de caractéristiques.

```
for (i dans centerScale) do
    for ( j dans surroundScale) do
        redimensionner(pyramid[i + j], pyr_s, pyr_c.taille());
        pyr_c ← pyramid[i];
        pyr_s ← pyramid[j];
        absdiff(pyr_c, pyr_s, diff);
        différences←Ajouter(diff);
    end for
end for
```

```
Fonction obtenir_des_ cartes_caractéristique _en_couleur(couleur_pyr)
  ▷ cette carte des caractéristique est calculée en soustrayant tous les combos entre
  les niveaux (1,2,3) et (6,7), c'est-à-dire (7e et 8e niveaux).
  var Cartes_de_couleurs_RG,Cartes_de_couleurs_BY :list;
  pyr_rouge, pyr_vert, pyr_bleu, pyr_jaune ← pyr_de_couleur;
  for C in range (1,4) do
      for S in range (6,8) do
                  ▷ calcul de la carte RG actuelle à l'aide des pyramides de couleurs.
          RG_map_actuel ← soustraire_des_imgs_de_taille_différente(grand_RG,petir_RG)
                  ▷ calcul de la carte By actuelle à l'aide des pyramides de couleurs.
          RY_map_actuel ← soustraire_des_imgs_de_taille_différente(grand_RY,petir_RY)
          Cartes_de_couleurs_RG← Ajouter(RG_map_actuel);
          Cartes_de_couleurs_BY← Ajoute(BY_map_actuel);
      end for
  end for
Retourne Cartes_de_couleurs_BY,Cartes_de_couleurs_RG;
```

les cartes RG(c, s) et les cartes ,BY (c, s) sont créées :

$$RG(c,s) = |(R(c) - G(c)) \ominus (G(s) - R(s))|. \tag{3.7}$$

$$BY(c,s) = |(B(c) - Y(c)) \ominus (Y(s) - B(s))|. \tag{3.8}$$

```
Fonctionobtenir_des_cartes_de caractéristiques_d'intensité(intensité_pyr)
  var cartes_d'intensité :list;
  for C in range (1,4) do
      for S in range (6,8) do
          ▷ calcul de la carte d'intensité actuelle à l'aide de la pyramide d'intensité;
          I_actuel←soustraire_des_imgs_de_taille_différente(intensité_pyr[c],intensité_pyr[s]);
      end for
  end for
       =0
Retourne I_actuel;
```

Pour l'intensité, les valeurs des niveaux de résolution plus fine c sont soustraites aux valeurs des niveaux de résolution plus grossiére s :

$$I(c,s) = |I(c) \ominus I(s)|. \tag{3.9}$$

cette soustraction nécessite une interpolation de la carte I(s) pour qu'elle puisse avoir la méme taille que I(c). On obtient 6 des cartes de caractéristiques d'intensité ("feature maps") I(c, s).
Où c représente "center" et s représente "surround".

#### 3.2.5.3 les algorithmes pour créer les carte de visibilité

Pour chaque caractéristique (intensité, orientation ou couleur) toutes les cartes de trait sont fusionnées pour créer une carte de visibilité (" conspicuity map"). Cette carte est aussi normalisée par la normalisation . Enfin, la carte de saillance finale est construite en fusionnant les trois cartes de visibilité.

**Fonction** carte_de_visibilité_couleur(carte_de_ caractéristique_couleur)
**Var** Carte_RG,Carte_BY←carte_de_ caractéristique_couleur;
▷ le calcul sur l'échelle 4 de la pyramide.
**Var** échelle_4;
**i**←0;
▷ en modifiant la taille de toutes les cartes caractéristique à l'échelle 4 et en les additionnant.
**Var** Carte;
**for** `C in range (1,4)` **do**
**for** `S in range (6,8)` **do**

RG ← normaliser_image(Carte_RG[i]);
▷ changer la taille à l'échelle 4;
RG ← changer_léchelle(RG,échelle_4);
BY ← normaliser_image(Carte_BY[i]);
BY ← changer_léchelle(BY,échelle_4);
la Carte = $laCarte + (RG + BY)$;
**end for**
**end for**
=0
**Retourne** normaliser_image(la Carte);

Pour les trois canaux, les cartes des caractéristiques sont normalisées et combinées à travers les échelles et les orientations dans des "cartes de visibilité" pour chaque canal. Enfin, une carte globale de saillance est obtenue en combinant linéairement les canaux.

Les cartes de trait seront sommées en vue de la création de la carte de saillance.

la carte de saillance est combinée avec le mécanisme de WTA (“Winner-Take-All”) pour choisir les fixations au cours du temps.La premiére fixation est choisie comme le maximum de la carte de saillance. Cette position est ensuite masquée avant de chercher le maximum suivant pour la deuxiéme fixation.

## 3.3 Conclusion

Dans cette chapitre, nous avons présenté l’architecture générale et détaillée de l’application à travers un ensemble d’algorithmes et de schémas.

# CHAPITRE 4 Mise en oeuvre et résultats

## 4.1 Introduction

Après avoir élaboré une étude détaillée sur le principe du modèle de l'attention visuelle d'itti. Nous consacrons le chapitre courant aux aspects techniques de notre travail.
Le domaine du traitement d'images est riche en fonctions, nous en avons utilisé selon les besoins du programme, nous avons choisi la bibliothèque opencv et une langage de programmation interprété, multi- paradigme et multiplateformes (Python) avec le programme Visual Studio Code

## 4.2 Outils de développement

Visual Studio Code est un éditeur de code en termes simples. Visual Studio Code est "un éditeur gratuit qui aide le programmeur à écrire du code, aide au débogage et corrige le code en utilisant la méthode intelli-sense".S'exécute sur votre bureau et est disponible pour Windows, macOS et Linux. Il est livré avec un support intégré pour JavaScript, TypeScript et Node.js et dispose d'un riche écosystème d'extensions pour d'autres langages (tels que C++, C, Java, Python, PHP, Go) et des runtimes (tels que .NET et Unity) . Commencez votre voyage avec VS Code avec ces

## 4.3 Langages de programmation

Python est un langage de programmation interprété, orienté objet et de haut niveau avec une sémantique dynamique. Ses structures de données intégrées de haut niveau, combinées au typage dynamique et à la liaison dynamique, le rendent très attrayant pour le développement rapide d'applications.

## 4.4 Description de OpenCv

Initialement développé par Intel, OpenCV (Open Source Computer Vision) est une bibliothèque multiplateforme gratuite pour le traitement d'images en temps réel qui est devenue un outil standard de facto pour tout ce qui concerne la vision par ordinateur. La première version est sortie en 2000 sous licence BSD et depuis, ses fonctionnalités ont été très largement enrichies par la communauté scientifique. En 2012, la fondation à but non lucratif OpenCV.org s'est chargée de maintenir un site de support pour les développeurs et les utilisateurs [63]. OpenCV a été conçu pour fournir une infrastructure commune pour les applications de vision par ordinateur et pour accélérer l'utilisation de la perception par machine dans les produits commerciaux. En tant que produit sous licence BSD, OpenCV permet aux entreprises d'utiliser et de modifier facilement le code.

## 4.5 Les Codes

### 4.5.1 Prétraitement et construire les pyramides

```
class pretraitement:
    def obtenir_filtre_gaussien(self,Taille):
        if Taille == 1:
            return [1]
        base = np.array([1,1])
        filter_g = np.array([1,1])

        for i in range(Taille - 2):
             filter_g = np.convolve(base,filter_g)

        return (filter_g/sum(filter_g)).reshape(1,Taille)
    def construire_pyr_g_rec_(self,image,Niveau_max,g_filter):
        height, width = image.shape
        if(Niveau_max == 1 or height < Taille_min_image or width <
   Taille_min_image ):
            return[image]
        image_floue = convolve(image,Taille_filter,mode="reflect")
        image_floue = convolve(image_floue,Taille_filter.T,mode="
   reflect")

        image_lechelle_down = image_floue[::2,::2]

        pyramide = self.construire_pyr_g_rec_(image_lechelle_down,
   Niveau_max - 1, Taille_filter)

        pyramide.insert(0,image)
        return pyramide
    def construire_pyr_g(self,image, Niveau_max, Taille_filter):
        filtre_gaussien = self.obtenir_filtre_gaussien(filter_size)
        return self.construire_pyr_g_rec_(image,Niveau_max,
   filtre_gaussien)
```

Cette classe contient plusieurs fonctions :

— La fonction "obtenir _filtre_gaussien :"obtient une taille comme paramétre et renvoie le filtre gaussien 1D correspondant.,la fonction créer le tableau "base"("numpy array") pour commencer à convoluer pour obtenir le filtre gaussien.

— Construire_pyr_g_rec :fonction récursive, crée une pyramide gaussienne d'une image donnée jusqu'à des niveaux maximum.La fonction prend les variables suivantes comme paramétre :

  — Image :l'image d'entrée
  — niveaux max : à quelle profondeur la pyramide doit-elle aller.
  — g_filter, le filtre à utiliser pour le flou.

  Retourne : une tableaux python de toutes les images de la pyramide et le vecteur utilisé pour faire covoluer l'image

— La fonction construire_pyr_g :crée une pyramide d'image gaussienne et la renvoie sous forme de tableau python.La fonction prend les variables suivantes comme paramétre :

— image :une image en niveaux de gris.
— Niveau_max :à quelle profondeur la pyramide doit-elle aller.
— Taille_filter quelle taille de filtre gaussien utiliser.

### 4.5.2 Obtenir les pyramides

```
def extrait_les_c(image):
     rouge_pure = image[:,:,Rouge]
     vert_pure = image[:,:,Vert]
     blue_pure = image[:,:,BLUE]
     return rouge_pure,vert_pure,blue_pure
def  Canaux(image):
    rouge,vert,bleu = extrait_les_c(image)
    Canal_rouge  = rouge - (vert + bleu) / 2
    Canal_vert  = vert-(rouge + bleu) / 2
    Canal_blue = bleu  - (rouge + vert) / 2
    Canal_j = (rouge + vert) / 2 - abs(rouge - vert)/2 - bleu
    return Canal_rouge,Canal_vert,Canal_blue,Canal_j
def obtenir_pyrs_couleurs(image):
    pyr=pr traitement()
    Canal_rouge,Canal_vert,Canal_bleu,Canak_j = Canaux(image)

    pyr_rouge = pyr.construire_pyr_g(Canal_rouge,PROFONDEUR_PYRAMIDE,
   TAILLE_GAUSSIENNE)
    pyr_vert = pyr.construire_pyr_g(Canal_vert,PROFONDEUR_PYRAMIDE,
   TAILLE_GAUSSIENNE)
    pyr_blue = pyr.construire_pyr_g(Canal_bleu,PROFONDEUR_PYRAMIDE,
   TAILLE_GAUSSIENNE)
    pyr_jaune = pyr.construire_pyr_g(Canak_j,PROFONDEUR_PYRAMIDE,
   TAILLE_GAUSSIENNE)

    carte_caractiristique.obtenir_carte_caract ristique_en_couleur([
   pyr_rouge,pyr_vert,pyr_blue,pyr_jaune])
    return
def calcule_lintensit (image):
    image_intensit  = image[:,:,Rouge] + image[:,:,Vert] + image[:,:,
   BLUE]
    image_intensit  = image_intensit  / 3
    return image_intensit

def obtenir_pyramide_intensit (image):
    pyr=pr traitement()

    image_intensit  = calcule_lintensit (image)



    return  carte_caractiristique.
   obtenir_carte_caract ristique_intensit (pyr.construire_pyr_g(
   image_intensit ,PROFONDEUR_PYRAMIDE,TAILLE_GAUSSIENNE))

def filtres_Gabor():
    filters = []
    TailleK = 31
```

```
    for theta in np.arange(0, np.pi, np.pi/4 ):
        kern = cv2.getGaborKernel((TailleK, TailleK), 4.0, theta, 7.0,
   0.5, 0, ktype=cv2.CV_32F)
        kern /= 1.5 * kern.sum()
        filters.append(kern)
    return filters
def obtenir_pyrs_angle(image):
    pyr=pr traitement()
    pyramide_angle = []
    image_intensit  = calcule_lintensit (image)
    filtres_gb = filtres_Gabor()
    for ker_theta in filtres_gb:
        im_filtr e_angle = cv2.filter2D(image_intensit , cv2.CV_32F,
   ker_theta)
        pyramide_angle.append(pyr.construire_pyr_g(im_filtr e_angle ,
   PROFONDEUR_PYRAMIDE,TAILLE_GAUSSIENNE))

    carte_caractiristique.obtenir_carte_caract ristique_angle(
   pyramide_angle)
    return
```

— extraite_les_c :La fonction prend une image puis isoler ses couleurs dans des tableaux.
— Canaux :La fonction prend une image puis créer les canaux de couleur largement réglés.
— obtenir_pyrs_couleur :
  — param image r,g,b.
  — la fonction crée un tuple de 4 pyramides une pour chaque canal de couleur pyr_rouge,pyr_vert, pyr_blue,pyr_jaune.
  — Retourne :un tableau des 4 pyramides de couleurs, r,v,b,j.
— obtenir_pyramide_intensité :
  — paramétre :image.
  — Retourne :la pyramide gaussienne d'intensité
— obtenir_pyrs_angle :
  — paramétre image.
  — obtenir l'image d'intensité, les pyramides d'angle.
  — Retourne :la pyramide des angles pour 0,45,90,135 dans une tableau (pyramide_angle).

```
class image_operation:
    def agrandir_limage(self,im,Taille_dest):
        return cv2.resize(im, (Taille_dest[1],Taille_dest[0]),
   interpolation = cv2.INTER_LINEAR)

    def soustraire_des_imgs_de_taille_diff rente(self,grand_image,
   petite_image):
        image_agrandie = self.agrandir_limage(petite_image, grand_image
   .shape)
        return grand_image - image_agrandie
```

### 4.5.3 Obtenir les cartes de caractéristiques

```
def obtenir_carte_caract ristique_en_couleur(Pyr_couleur):
    obj=image_operation()
    Carte_RG = []
```

```
    Carte_BY = []
    Pyr_rouge,Pyr_vert,Pyr_blue,Pyr_jaune = Pyr_couleur
    for C in range (1,4):
        for S in range (6,8):
            grand_RG = Pyr_rouge[C] - Pyr_vert[C]
            petit_RG = Pyr_vert[S] - Pyr_rouge[S]
            RG = abs(obj.soustraire_des_imgs_de_taille_diff rente(
   grand_RG,petit_RG))
            grand_BY = Pyr_blue[C] - Pyr_jaune[C]
            petit_BY = Pyr_jaune[S] - Pyr_blue[S]
            BY = abs(obj.soustraire_des_imgs_de_taille_diff rente(
   grand_BY,petit_BY))
            Carte_RG.append(RG)
            Carte_BY.append(BY)
    C_visibi.Carte_visibilit _couleur([Carte_RG, Carte_BY])
def obtenir_carte_caract ristique_intensit (Pyr_intensit ):
    obj=image_operation()
    Carte_intensit  = []
    for C in range(1, 4):
        for S in range(6, 8):
            La_carte_dintensit _actuelle = abs(obj.
   soustraire_des_imgs_de_taille_diff rente(Pyr_intensit [C] ,
   Pyr_intensit [S]))
            Carte_intensit .append(La_carte_dintensit _actuelle)
    return    C_visibi.Carte_visibilit _dintensit (Carte_intensit )

def obtenir_carte_caract ristique_angle(Pyr_angle):
    obj=image_operation()
    p0,p45,p90,p135 = Pyr_angle
    Carte0  = []
    Carte135 = []
    Carte90 = []
    Carte45 = []
    for C in range(1, 4):
        for S in range(6, 8):
            Carte_dintensit _act = abs(obj.
   soustraire_des_imgs_de_taille_diff rente(p0[C] , p0[S]))
            Carte0.append(Carte_dintensit _act)
            Carte_dintensit _act = abs(obj.
   soustraire_des_imgs_de_taille_diff rente(p45[C] , p45[S]))
            Carte45.append(Carte_dintensit _act)
            Carte_dintensit _act = abs(obj.
   soustraire_des_imgs_de_taille_diff rente(p90[C] , p90[S]))
            Carte90.append(Carte_dintensit _act)
            Carte_dintensit _act = abs(obj.
   soustraire_des_imgs_de_taille_diff rente(p135[C] , p135[S]))

            Carte135.append(Carte_dintensit _act)
    C_visibi.Carte_visibilit _dangle([Carte0,Carte45,Carte90,Carte135
   ])
```

— la fonction obtenir_carte_caractristique_en_couleur :
  — paramétre couleur_pyr renvoyé par la fonction obtenir_pyrs_couleur(image)
  — les variables : grand_RG,grand_BY,petite_RG,petite_BY pour la caluclation de la carte RG,BY.

— Retourne :les 12 couleurs des cartes sous forme de tableau, le premier tableau(Carte_RG)est de 6 entrées sont la carte RG et les 6 autres sont les cartes BY(Carte_RG).

— La fonction obtenir_carte_caractéristique_intensité
  — paramétre :intensité_pyr renvoyé par la fonction obtenir_pyrs_intensité(image).
  — Variables La_carte_dintensit_actuelle pour Calcul de la carte d'intensité actuelle.
  — ajouter la carte au tableau (Carte_intensité).

### 4.5.4 Normalisation et interpolation

```
class Normaliser_et_Interpolation:
    def Etirer(self,image):
        val_min = np.amin(image)
        val_max = np.amax(image)
        if(val_min!=val_max):
            image =  (image - val_min) * (val_max/(val_max-val_min))
        return image
    #*******************************************************
    def normaliser_limage(self,image):
        image = self.Etirer(image)
        max_val = np.amax(image)
        max_local = local_maxima(image,min_distance = 9 , num_peaks =
   200)
        max_local = np.transpose(max_local)
        vals_max_local = image[max_local[0],max_local[1]]
        moy_max  = np.average(vals_max_local)
        facteur = (max_val - moy_max) ** 2
        return image * facteur
    #*******************************************************
     def redimensionner_ _l chelle_donn e(self,echelle,image):
        return cv2.resize(image, (echelle[1] , echelle[0]),
   interpolation = cv2.INTER_LINEAR)
```

Cette classe contient les fonctions suivantes :

— Etirer(image) :
  — variable val_min :retourne le minimum du tableau.
  — variable val_max :retourne le maximum du tableau.
  — étire les valeurs d'image données entre 0 et la valeur maximale de l'image.

— La fonction normaliser_limage (image) :
  — normalisation des valeurs d'image entre 0 et l'image max.
  — trouver des maxima globaux et m maximas locaux.
  — Retoure : l'image normalisée.

### 4.5.5 Obtenir les cartes de visibilites

```
def Carte_visibilit _couleur(cartes_caract ristiques_en_couleur):
    N_I_objet=Normaliser_et_Interpolation()
    Carte_caract ristiques_RG , Carte_caract ristiques_BY =
   cartes_caract ristiques_en_couleur
      chelle4   = Carte_caract ristiques_RG[2].shape
    La_carte = np.zeros(  chelle4  )
```

```
    cpt = 0
    for C in range(1, 4):
       for S in range(6, 8):
         #RG_actuelle
         RG = N_I_objet.normaliser_limage(Carte_caract ristiques_RG[cpt])
         RG = N_I_objet.redimensionner_ _l chelle_donn e( chelle4 , RG)
        #######################################
        #RY_actuelle
         BY = N_I_objet.normaliser_limage(Carte_caract ristiques_BY[cpt])
         BY = N_I_objet.redimensionner_ _l chelle_donn e( chelle4 , BY)

         La_carte = La_carte + (BY + RG)
         cpt+=1
    carte_saillence.passer_carte_visibilit _couleur(N_I_objet.normaliser_limage(La_carte))
    return
def Carte_visibilit _dangle(cartes_caract ristiques_dangle):
    N_I_objet=Normaliser_et_Interpolation()
    cartes_caract0, cartes_caract45, cartes_caract90,cartes_caract135 = cartes_caract ristiques_dangle
     chelle4  = cartes_caract0[2].shape
    La_carte = np.zeros( chelle4 )
    cpt = 0
    for C in range(1, 4):
        for S in range(6, 8):
            _0 = N_I_objet.normaliser_limage(cartes_caract0[cpt])
            _0 = N_I_objet.redimensionner_ _l chelle_donn e( chelle4 , _0)
            _45 = N_I_objet.normaliser_limage(cartes_caract45[cpt])
            _45 = N_I_objet.redimensionner_ _l chelle_donn e( chelle4 , _45)
            _90 = N_I_objet.normaliser_limage(cartes_caract90[cpt])
            _90 = N_I_objet.redimensionner_ _l chelle_donn e( chelle4 , _90)
            _135 = N_I_objet.normaliser_limage(cartes_caract135[cpt])
            _135 = N_I_objet.redimensionner_ _l chelle_donn e( chelle4 , _135)
            La_carte = La_carte + (_0 + _45 + _90 + _135)
            cpt += 1
    carte_saillence. passer_carte_visibilit _angles(La_carte)
    return
def Carte_visibilit _dintensit (cartes_caract ristiques_dintensit ):
    N_I_objet=Normaliser_et_Interpolation()
     chelle4  = cartes_caract ristiques_dintensit [2].shape
    La_carte = np.zeros( chelle4 )
    cpt = 0
    for C in range(1, 4):
        for S in range(6, 8):
            intensit _act = N_I_objet.normaliser_limage(cartes_caract ristiques_dintensit [cpt])
```

```
49              intensit _act = N_I_objet.
    redimensionner_ _l chelle_donn e( chelle4 , intensit _act)
50              La_carte = La_carte + intensit _act
51              cpt += 1
52      return carte_saillence.passer_carte_visibilit _intensit (
    N_I_objet.normaliser_limage(La_carte))
```

— Carte_ visibilit_couleur(cartes_caractristiques_e n_couleur) :
  — paramétre :cartes_caractiristiques_en_c :calculé en la fonction obtenir_carte_caractristique_en_couleur
  — variable échelle4 :pour calculé la pyramide sur l'échelle 4
  — définir une tableau(La_cart) vide pour commencer à ajouter de la valeur.
  — L'ajout de toutes les cartes de caractéristique créées.
  — passant, la carte normalisée qui a été calculée.

```
1 Carte_couleur = 0
2 Carte_intensit  = 0
3 Carte_dangle = 0
```

déclaration de les variables (Carte_couleur,Carte_intensité,Carte_dangle) comme des variables globale.

### 4.5.6 Créer la carte de saillance

Enregistre les données dans ces trois fonctions.

```
1  def passer_carte_visibilit _couleur(Carte_visibilit _couleur):
2      global Carte_couleur
3      Carte_couleur = Carte_visibilit _couleur
4
5
6  def  passer_carte_visibilit _angles(Carte_visibilit _dangle):
7      global Carte_dangle
8      Carte_dangle = Carte_visibilit _dangle
9
10
11 def passer_carte_visibilit _intensit (Carte_visibilit _dintensit ):
12     global Carte_intensit
13     Carte_intensit  = Carte_visibilit _dintensit
14     return cr er_carte_de_saillance()
```

Calcule la carte de saillance.

```
1  def cr er_carte_de_saillance():
2      carte_saillance = Carte_dangle + Carte_intensit  + Carte_couleur
3      carte_saillance = carte_saillance / 3
4      Carte_couleur = 0
5      Carte_intensit  = 0
6      Carte_dangle = 0
7      return  resulta.Carte_s_finale(carte_saillance)
```

### 4.5.7 Appliquer le mécanisme WTA

```
def WTA(Carte_salliance, rounds, taille_gaussienne):
    #le facteur
    Taille_demi = int(taille_gaussienne/2)
    Taille_cart_s_x = Carte_salliance.shape[0] + taille_gaussienne
    Taille_cart_s_y = Carte_salliance.shape[1] + taille_gaussienne

    carte_s_wta = np.zeros((Taille_cart_s_x,Taille_cart_s_y))

    #creating the gaussian to be the center

    gaussien = cv2.getGaussianKernel(ksize=taille_gaussienne, sigma =
    20)
    gaussien  = gaussien * np.transpose(gaussien)

    #trouver l'endroit le plus saillant de l'image.
    coordonn es_max = local_maxima(Carte_salliance, min_distance = 15
    , num_peaks = rounds)

    #cr er la carte en pla ant des noyaux gaussiens aux points
    saillants.
    for i in range(rounds):
        if(coordonn es_max.shape[0] -1 < i):
            break
        coordonn e_actuelle = coordonn es_max[i]
        deb_ligne  = coordonn e_actuelle[0]
        fin_ligne    = coordonn e_actuelle[0] + taille_gaussienne

        d b_colonne  = coordonn e_actuelle[1]
        fine_colonne    = coordonn e_actuelle[1] + taille_gaussienne


        carte_s_wta[int(deb_ligne):int(fin_ligne), int(d b_colonne):
    int(fine_colonne)] += gaussien
        carte_s_wta[np.where(carte_s_wta < 0)] = 0

    #redimensionner la carte     sa taille d'origine.

    carte_s_wta = carte_s_wta[Taille_demi:Carte_salliance.shape[0] +
    Taille_demi,
                    Taille_demi:Carte_salliance.shape[1] + Taille_demi]
    return Etirer(carte_s_wta)
```

— Les paramétres :
  — paramétre Carte_saillance La carte de saillance calculé.
  — paramétre rounds :combien de rounds le vainqueur prend pour courir.
  — taille_gaussienne :quelle est la taille de la gaussienne à utiliser pour la carte thermique.
— les variables :
  — Taille_demi :calculer le facteur nécessaire pour créer la gaussienne et la carte thermique finale.
  — gaussien :créant la gaussienne pour être le centre des points de chaleur.
  — coordonnes_max :pour trouver l'endroit le plus saillant de l'image.

### 4.5.8 Obtenir le "Heat map"

```
def carte_thermique(image_or, Carte_salliance):
    alpha = 0.4
    image_or = cv2.resize(image_or, (Carte_salliance.shape[1],
   Carte_salliance.shape[0]),
                                  interpolation=cv2.INTER_LINEAR)
    saillant_max = np.max(Carte_salliance)
     tape    = saillant_max /7
    thermique_couleur = np.zeros((image_or.shape[0],image_or.shape
   [1],4))

    taches_de_couleurs(thermique_couleur)

    if (image_or.shape[2] == 3):
        im_temp = np.ones((image_or.shape[0], image_or.shape[1], 4))
        im_temp[:, :, 0:3] = image_or
        image_or = im_temp

    cv2.addWeighted(thermique_couleur, alpha, image_or, 1 - alpha,
                    0, thermique_couleur)
    return  thermique_couleur
```

## 4.6 Résultats

FIGURE 4.1 – Image 1

FIGURE 4.2 – la pyramide d'intensité

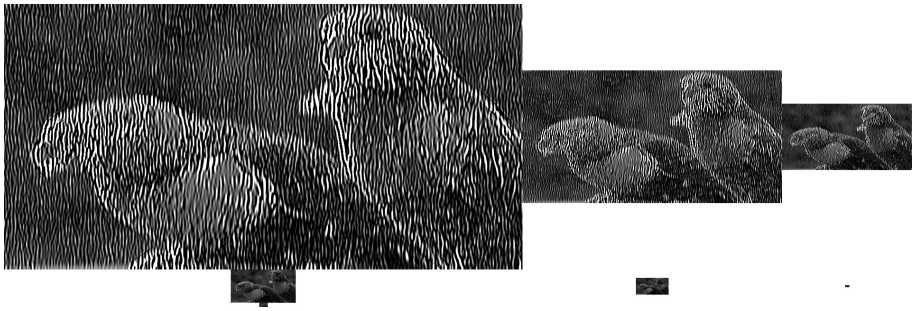


FIGURE 4.3 – la pyramide des angles pour 0

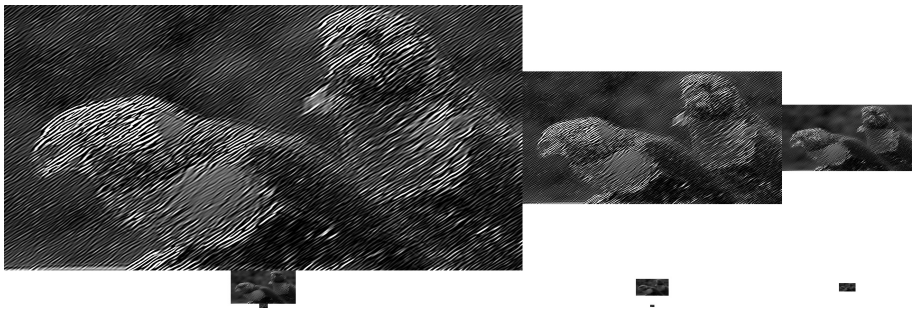


FIGURE 4.4 – la pyramide des angles pour 45

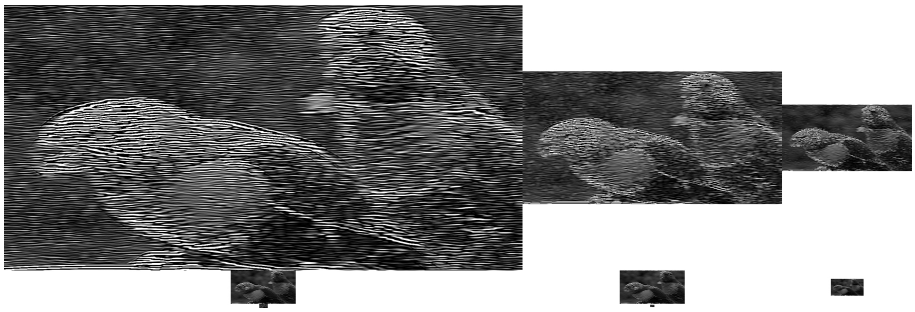


FIGURE 4.5 – la pyramide des angles pour 90

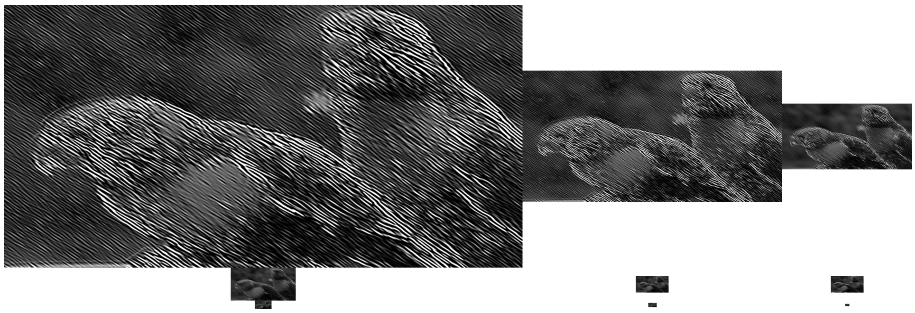


FIGURE 4.6 – la pyramide des angles pour 135

FIGURE 4.7 – pyramides bleues

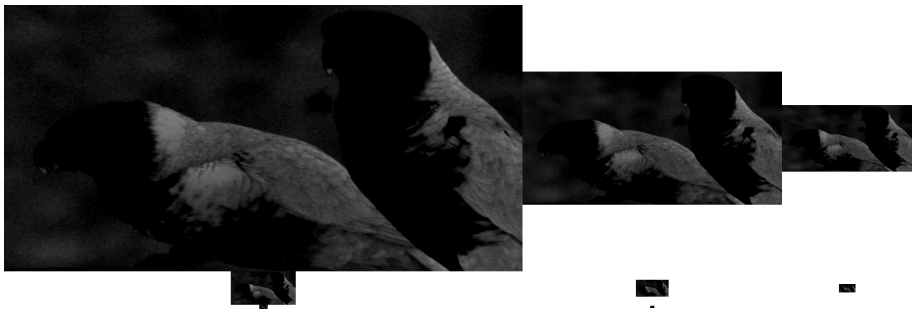


FIGURE 4.8 – pyramides vertes

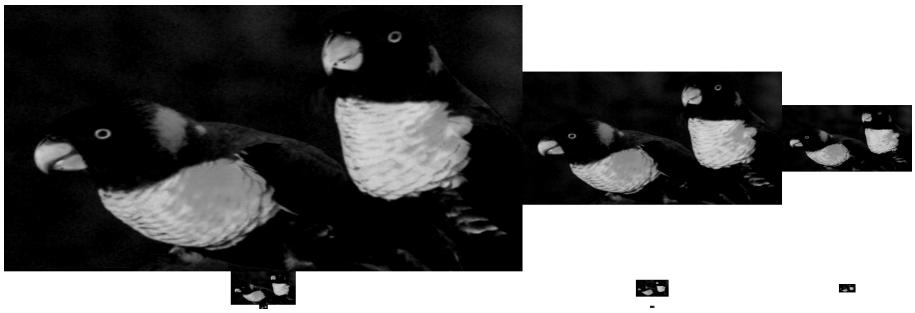


FIGURE 4.9 – pyramides rouges

FIGURE 4.10 – pyramides jaunes

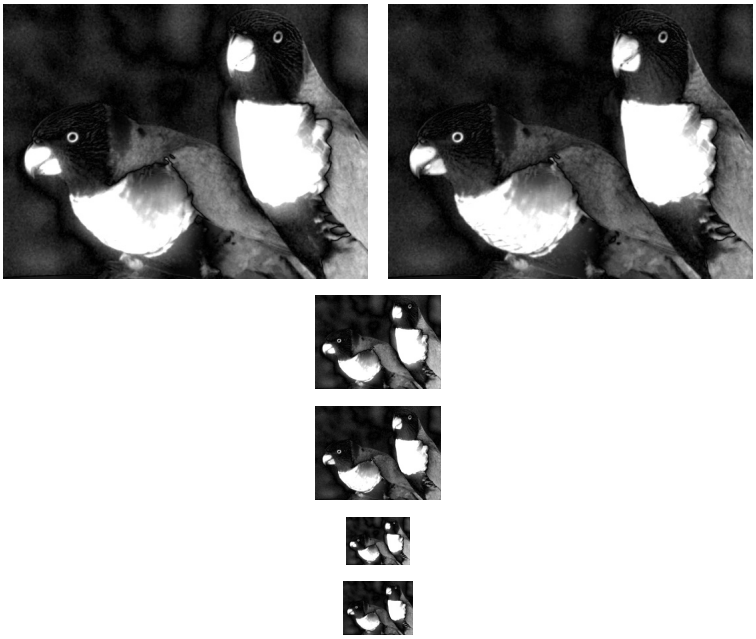


FIGURE 4.11 – carte RG

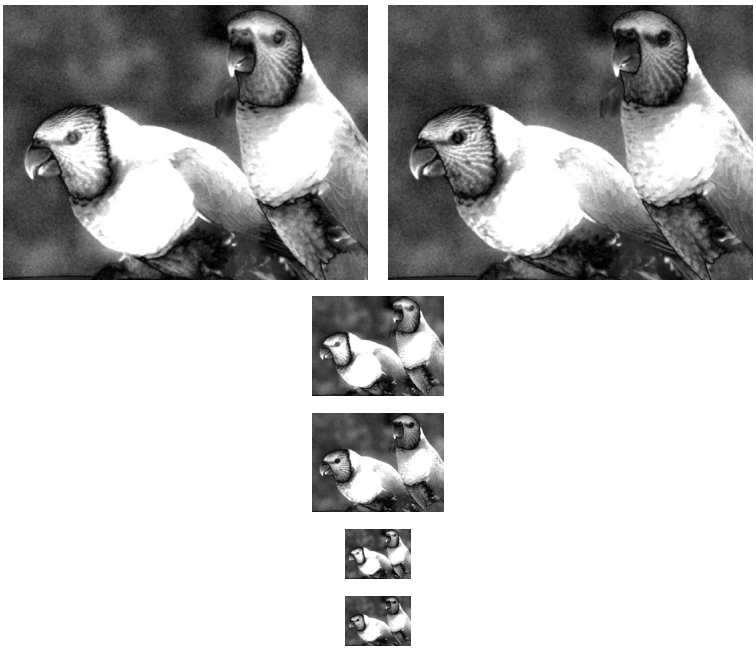


FIGURE 4.12 – carte BY

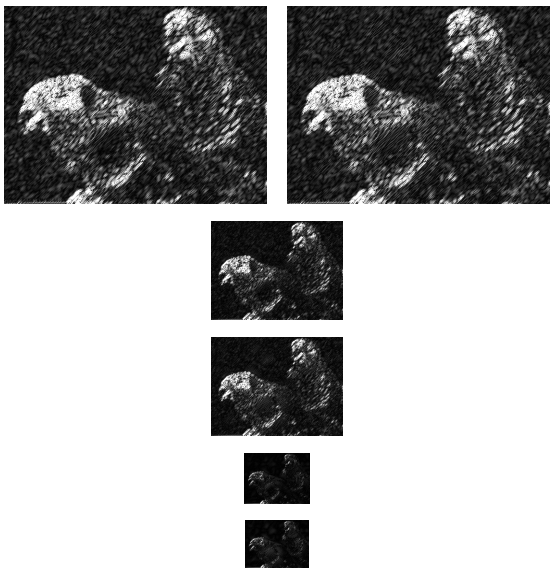


FIGURE 4.13 – cartes de caractéristiques d'angle(45°)

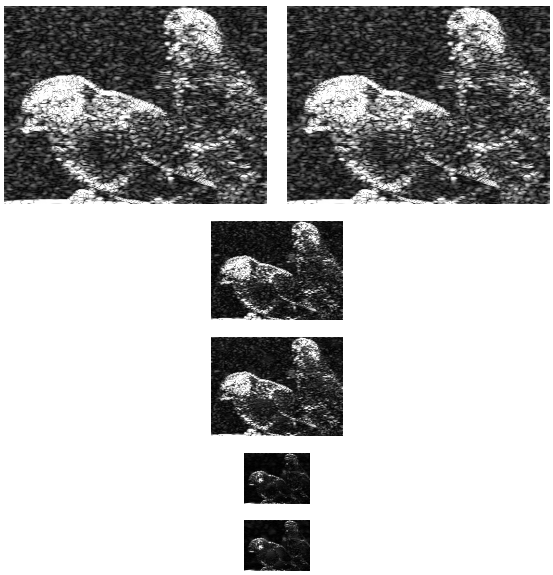


FIGURE 4.14 – cartes de caractéristiques d'angle(90°)

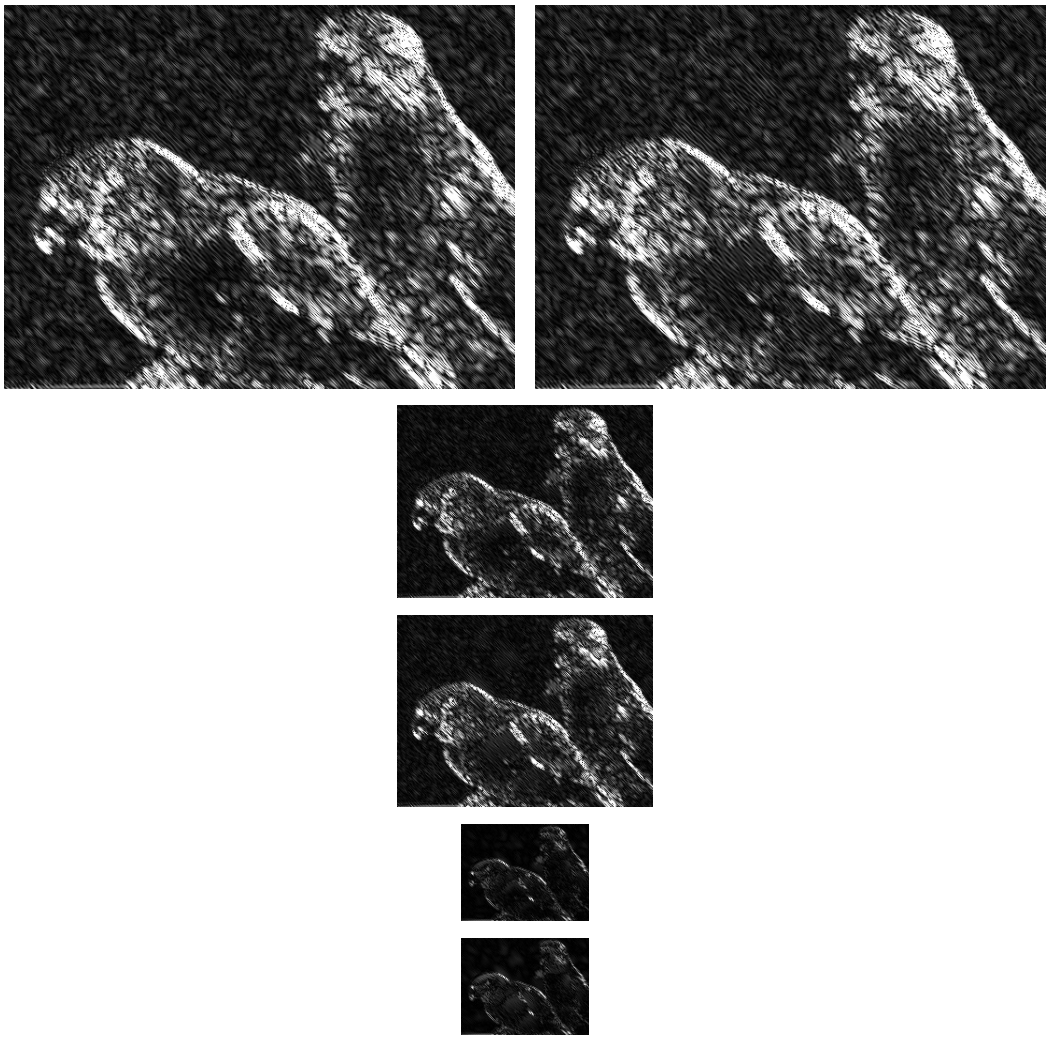


FIGURE 4.15 – cartes de caractéristiques d'angle(135°)

FIGURE 4.16 – carte de visibilité d'intensité

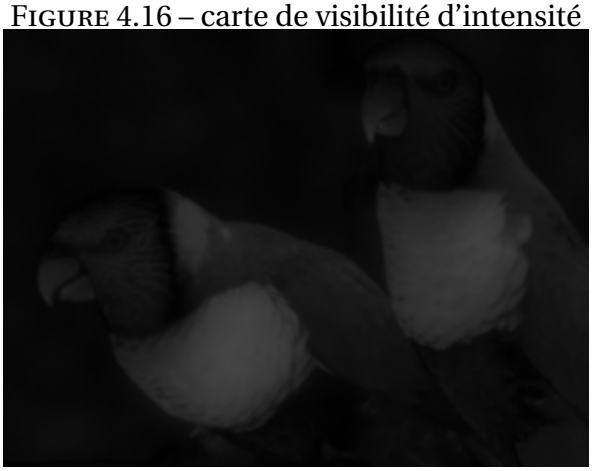


FIGURE 4.17 – carte de visibilité en couleur

FIGURE 4.18 – carte de visibilité d'angle

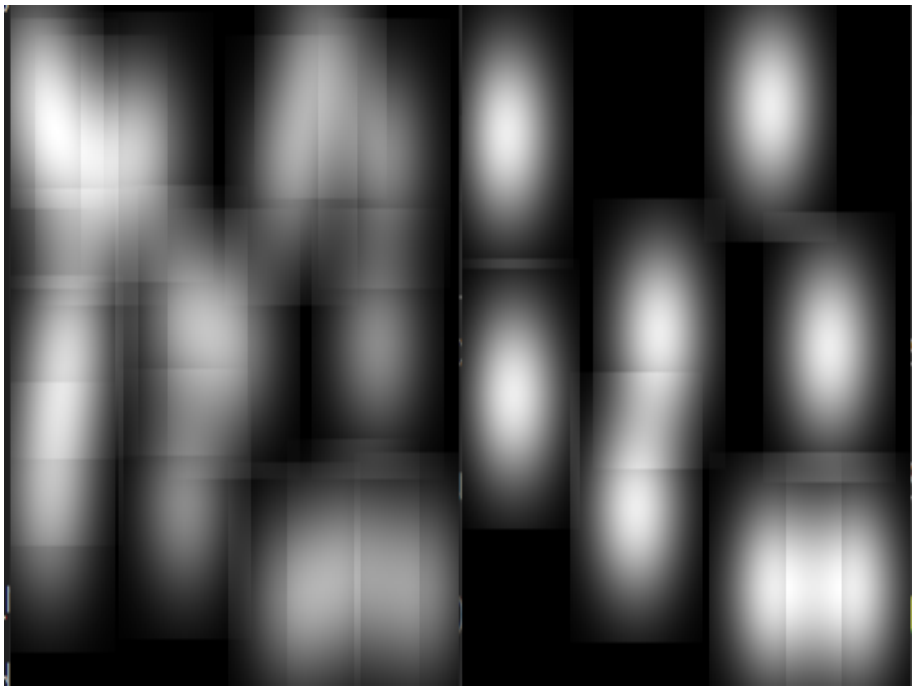


FIGURE 4.19 – La carte de saillance(La photo de gauche est le résultat avant le WTA)

FIGURE 4.20 – La carte de saillance

FIGURE 4.21 – Image2

FIGURE 4.22 – la pyramide d'intensité

FIGURE 4.23 – la pyramide des angles pour 0

FIGURE 4.24 – la pyramide des angles pour 45

FIGURE 4.25 – la pyramide des angles pour 90

FIGURE 4.26 – la pyramide des angles pour 135

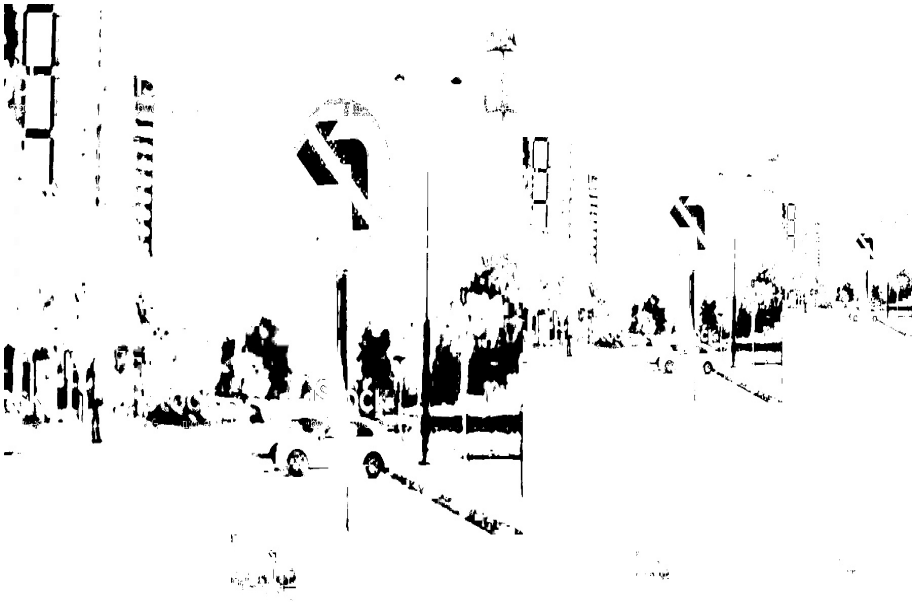


FIGURE 4.27 – pyramides bleues

FIGURE 4.28 – pyramides vertes

FIGURE 4.29 – pyramides rouges

FIGURE 4.30 – pyramides jaunes

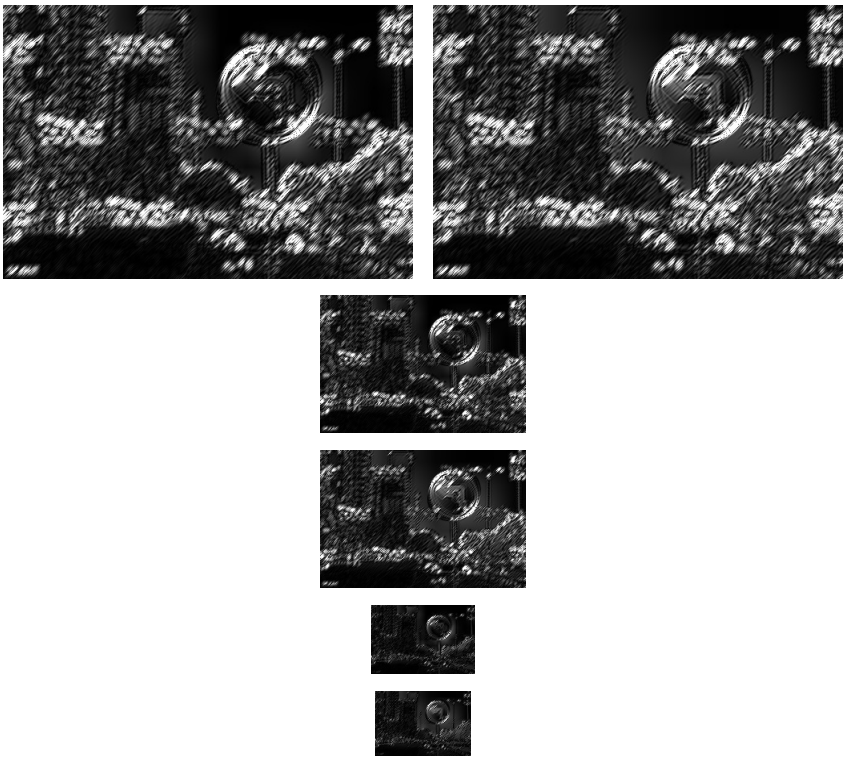


FIGURE 4.31 – cartes de caractéristiques d'angle(45°)

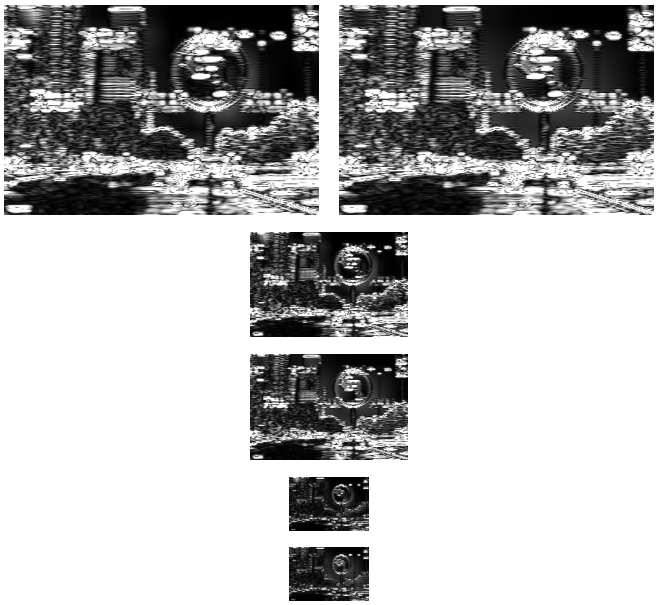


FIGURE 4.32 – cartes de caractéristiques d'angle(90°)

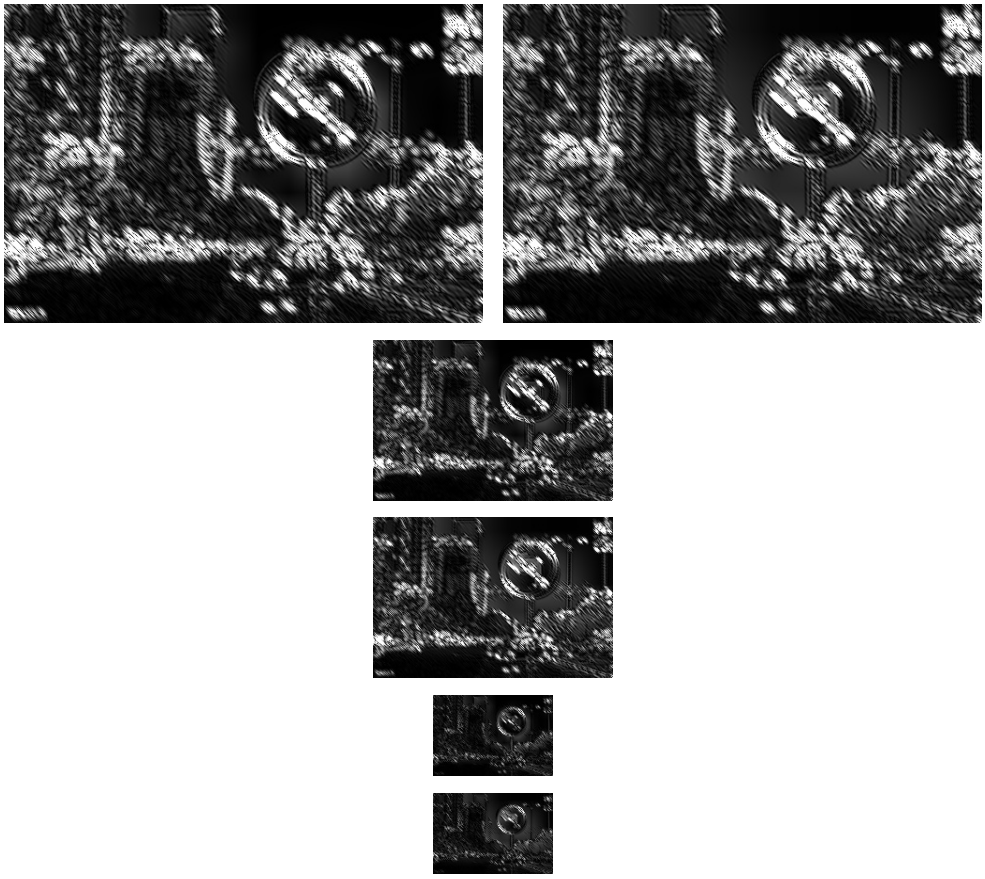


FIGURE 4.33 – cartes de caractéristiques d'angle(135°)

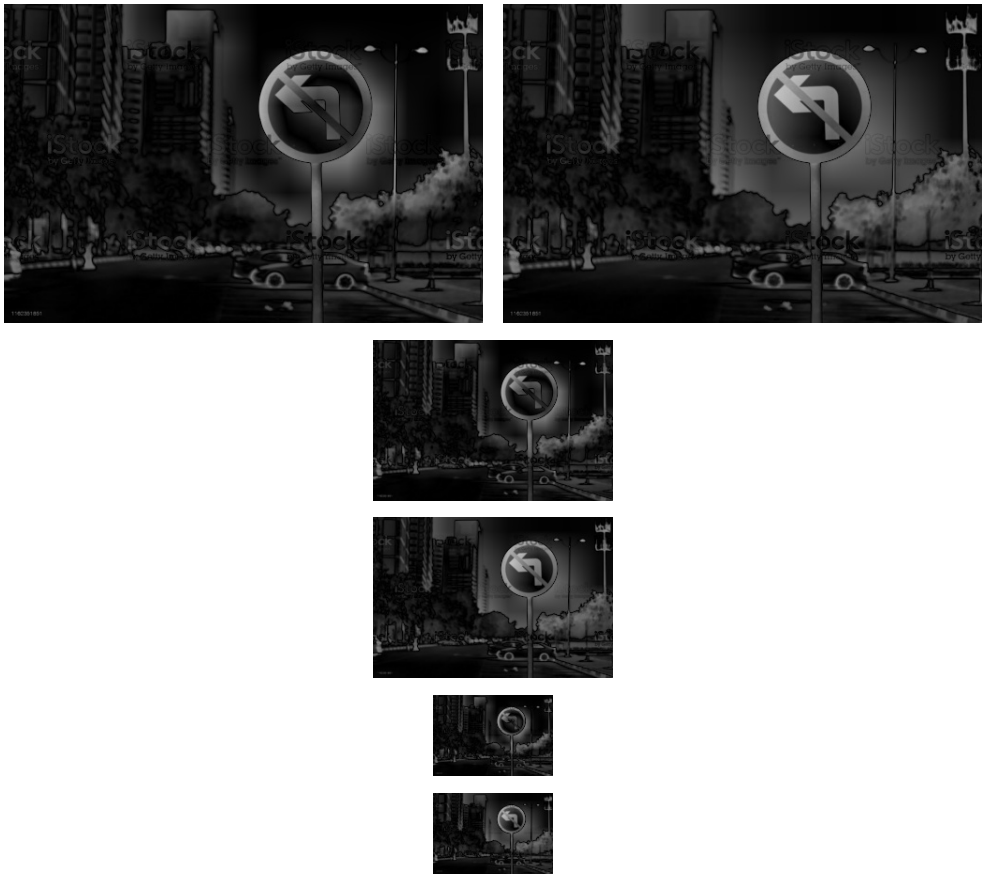


FIGURE 4.34 – cartes de caractéristiques d'intensité

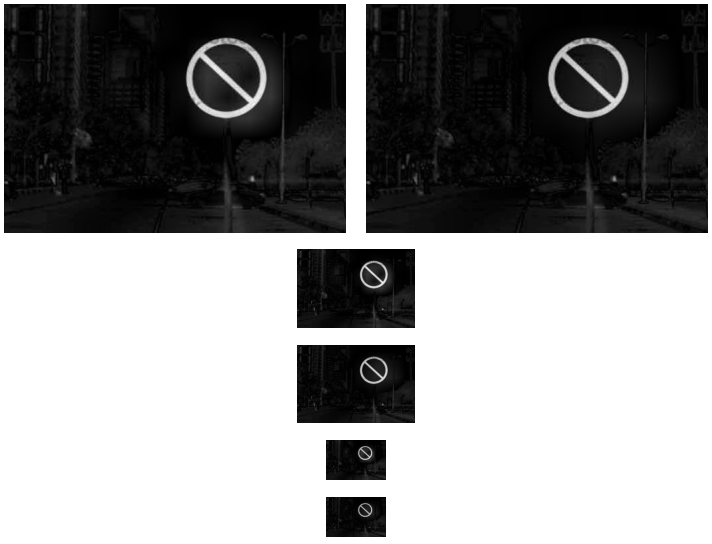


FIGURE 4.35 – carte RG

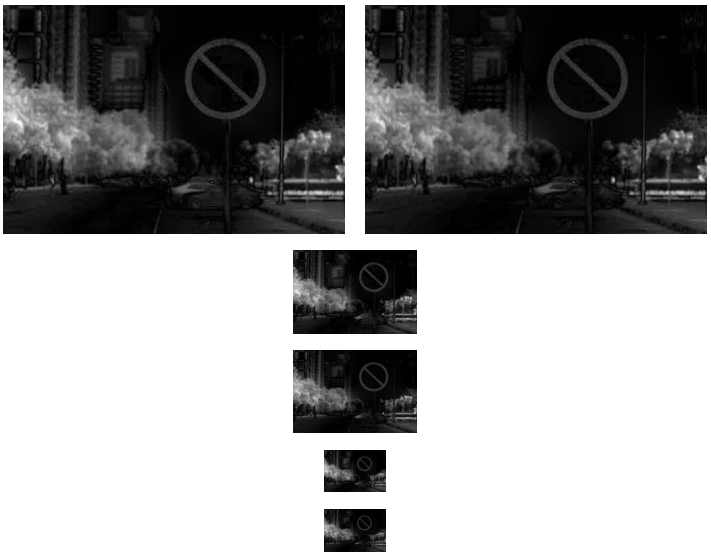


FIGURE 4.36 – carte BY

FIGURE 4.37 – carte de visibilité d'intensité

FIGURE 4.38 – carte de visibilité en couleur

FIGURE 4.39 – carte de visibilité d'angle

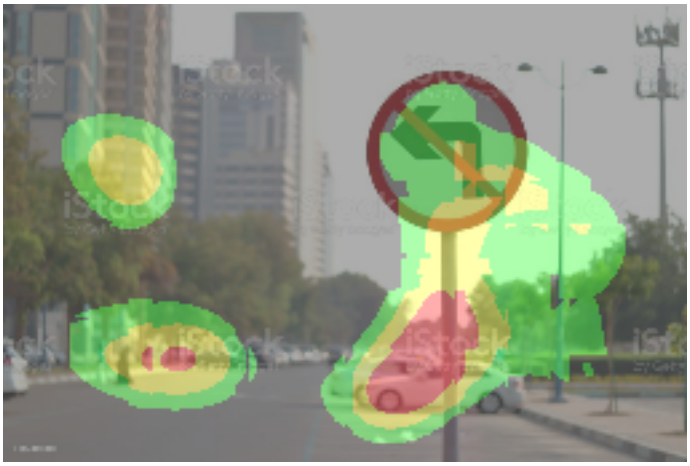


FIGURE 4.40 – La carte de saillance("Heat map")

## 4.7 Temps de calcul

Dans cette section, nous présentons les valeurs de temps de calculs de chacune des étapes de la construction de la carte de la saillance.

| | Les pyramide | Cartes de caractéristiques | Cartes de visibilités | Carte de saillance |
|---|---|---|---|---|
| **Image1** | 0.1575Sec | 0.1695Sec | 0.3221Sec | 1.012Sec |
| **Image2** | 0.1226Sec | 0.1336Sec | 0.4188Sec | 1.2023sec |

FIGURE 4.41 – Temps d'execution des différentes étapes de l'application.

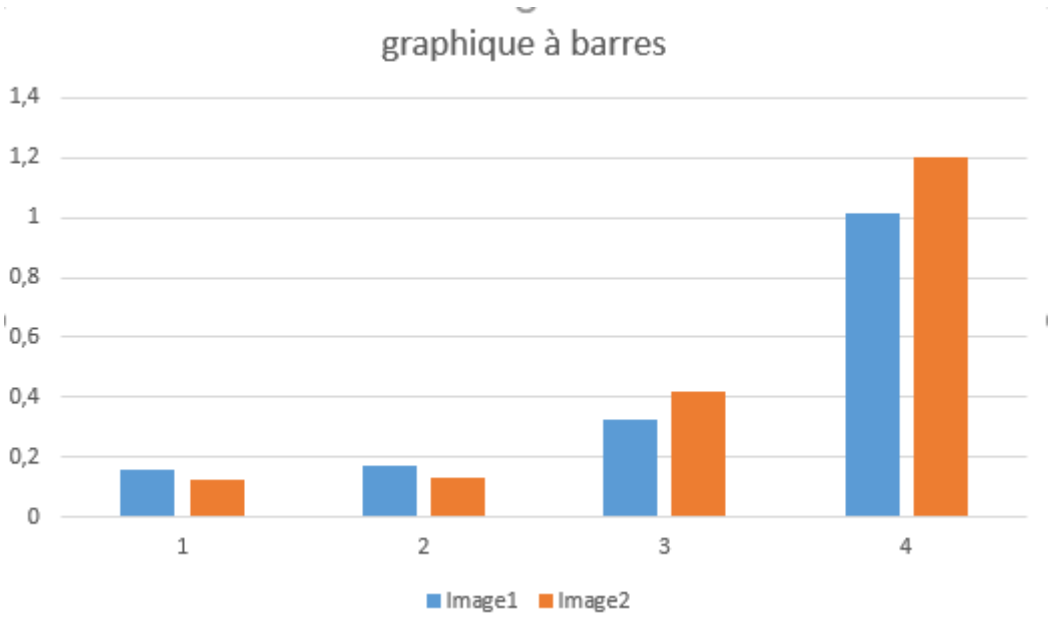



FIGURE 4.42 – Temps d'execution

## 4.8 Conclusion

Nous avons présenté dans ce chapitre les résultats de l'implémentation de la carte de saillance en utilisant la méthode classique d'Itti.Enfin,nous avons comparé les valeurs de temps de calcul pour chaque étape de la construction de la carte de saillance pour deux entrées différentes.

# Conclusion génerale :

Dans ce mémoire, nous nous sommes intéressés à la détection la détection de saillants dans des scènes complexe.L'objectif de ce travail est la détection de la carte saillance pour des images 2d ,qui permet mettre en évidence les zones importantes de l'image et ne traiter que les parties en saillants. Cela contribuera en fait à alléger la charge de calcul.
Nous avons exposé dans la première chapitre du théorique quelques notions de base concernant la perception visuelle.Ensuit Nous avons discuté des modèles et des processus d'attention.Ce chapitre nous avons présenté aussi la carte de saillance et les différentes étapes de construction.
Dons le chapitre 2 nous avons présente une étude bibliographique sur les modèles existants (koch ulman,Itti al, y compris une analyse détaillée du modéle d'itti.
Le triosiém chapitre est consacrée à la Conception d'application.où nous avons montré l'architecture générale et détaillée de l'application grâce à un ensemble d'algorithmes et de schémas.
Finalement, le chapitre 4 est dédié pour la phase de mise en œuvre et résultats de l'application des cartes saillances.Nous avons donnée une description sur le langage de programmation ,Outils de développement,la bibliothèque suivie par une séries de résultats obtenus.

# Bibliographie


[1] Dr Benmezroua Mohammed. Le systeme visuel. . https ://fmed13.weebly.com/uploads/1/0/8/0/108036529/13-le$_s ysteme_v isuel.pdf$.

[2] David Marr. Vision : A computational investigation into the human representation and processing of visual information, henry holt and co. *Inc., New York, NY*, 2(4.2), 1982.

[3] Ruben E Laukkonen and Jason M Tangen. Can observing a necker cube make you more insightful? *Consciousness and cognition*, 48 :198–211, 2017.

[4] Paul Taberham. *Lessons in perception : the avant-garde filmmaker as practical psychologist*. Berghahn Books, 2018.

[5] Peter Ferschin. Perception based illustration methods. 2004.

[6] Différence entre les formes 2d et 3d, 2022. . https ://fr1.suzuran-law-office.com/difference-between-1ad-and-3bbd-shapes-3bb3bb3bb.

[7] Josselin Gautier. *Un modèle d'attention visuelle dynamique pour conditions 2D et 3D; codage de cartes de profondeur et synthèse basée inpainting pour les vidéos multi-vues*. PhD thesis, Université Rennes 1, 2012.

[8] Qichang Duan, Tallha Akram, Pan Duan, and Xiaogang Wang. Visual saliency detection using information contents weighting. *Optik*, 127(19) :7418–7430, 2016.

[9] T Judd. Learning to predict where humans look ieee international conference on computer vision. *Proc. ICCV, 2009*, 2009.

[10] O Le Meur, P Le Callet, D Barba, et al. A spatio-temporal model to predict visual fixation : description and assessment. *Vision research*, 47(19) :2483–2498, 2007.

[11] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11) :1254–1259, 1998.

[12] Laurent Itti and Christof Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision research*, 40(10-12) :1489–1506, 2000.

[13] Dashan Gao, Vijay Mahadevan, and Nuno Vasconcelos. The discriminant center-surround hypothesis for bottom-up saliency. *Advances in neural information processing systems*, 20, 2007.

[14] Dashan Gao and Nuno Vasconcelos. Decision-theoretic saliency : computational principles, biological plausibility, and implications for neurophysiology and psychophysics. *Neural computation*, 21(1) :239–271, 2009.

[15] GIANLUCA Monaci, G Menegaz, S Susstrunk, and K Knoblauch. Color contrast detection in spatial chromatic noise. Technical report, 2002.

[16] Christof Koch and Shimon Ullman. Shifts in selective visual attention : towards the underlying neural circuitry. In *Matters of intelligence*, pages 115–141. Springer, 1987.

[17] Daniel Gaonac'h. *Psychologie cognitive et bases neurophysiologiques du fonctionnement cognitif*. Presses universitaires de France, 2006.

[18] Agnès Saulnier, Jérôme Thievre, and Marie-Luce Viaud. La perception du mouvement dans la visualisation : le cas des graphes. *Revue d'Interaction Homme-Machine*, 7(2), 2006.

[19] Charles H Anderson, David C Van Essen, and Bruno A Olshausen. Directed visual attention and the dynamic control of information flow. In *Neurobiology of attention*, pages 11–17. Elsevier, 2005.

[20] Karla K Evans, Todd S Horowitz, Piers Howe, Roccardo Pedersini, Ester Reijnen, Yair Pinto, Yoana Kuzmova, and Jeremy M Wolfe. Visual attention. *Wiley Interdisciplinary Reviews : Cognitive Science*, 2(5) :503–514, 2011.

[21] AL Yarbus. Eye movements and vision'plenum press. *New York*, 1967.

[22] U Neisser. Cognitive psychology. appleton-century-crofts.[aac] nelson, k.(2003) self and social functions : Individual autobiographical memory and collective narrative. *Memory*, 11(2) :12536, 1967.

[23] John K Tsotsos. What roles can attention play in recognition? In *2008 7th IEEE International Conference on Development and Learning*, pages 55–60. IEEE, 2008.

[24] Neil Bruce and John Tsotsos. Saliency based on information maximization. *Advances in neural information processing systems*, 18, 2005.

[25] Matthieu Perreira Da Silva. *Modèle computationnel d'attention pour la vision adaptative*. PhD thesis, Université de La Rochelle, 2010.

[26] William Thompson, Roland Fleming, Sarah Creem-Regehr, and Jeanine Kelly Stefanucci. *Visual perception from a computer graphics perspective*. CRC press, 2011.

[27] Alan C Bovik. *Handbook of image and video processing*. Academic press, 2010.

[28] John Smythies and Robert E French. *Direct versus indirect realism : A neurophilosophical debate on consciousness*. Academic Press, 2018.

[29] Derek H Brown and Fiona Macpherson. *The Routledge Handbook of Philosophy of Colour*. Routledge, 2020.

[30] Nicholas Silins. Seeing through the 'veil of perception'. *Mind*, 120(478) :329–367, 2011.

[31] Jenny Litzelmann. 13. redéfinition des notions d'instinct, d'inné et d'acquis chez konrad lorenz. In *Penser le comportement animal*, pages 305–318. Éditions Quæ, 2010.

[32] Gaston Bachelard. *Le rationalisme appliqué*, volume 43. Presses universitaires de France Paris, 1949.

[33] Samira Khettab. *LE SENS DU LIEU DANS LA GESTION DU PAYSAGE URBAIN : CAS DE TIPAZA*. PhD thesis, EPAU, 2019.

[34] TOK 17estankeviciute. Theory of perception, 2015.

[35] Yoichi Shimada, Kenichi Meguro, Mari Kasai, Masumi Shimada, Hiroshi Ishii, Satoshi Yamaguchi, and Atsushi Yamadori. Necker cube copying ability in normal elderly and alzheimer's disease. a community-based study : The tajiri project. *Psychogeriatrics*, 6(1) :4–9, 2006.

[36] James Cavanaugh and Robert H Wurtz. Subcortical modulation of attention counters change blindness. *Journal of Neuroscience*, 24(50) :11236–11243, 2004.

[37] Ueli Rutishauser, Dirk Walther, Christof Koch, and Pietro Perona. Is bottom-up attention useful for object recognition ? In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE, 2004.

[38] Josselin Petit, Roland Bremond, and Jean-Philippe Tarel. Une carte de saillance adaptée aux images hdr. *Revue Electronique Francophone d'Informatique Graphique*, 3(2), 2009.

[39] Kendra Cherry and Paul G Mattiuzzi. *The Everything Psychology Book : Explore the human psyche and understand why we do the things we do*. Simon and Schuster, 2010.

[40] Anass Nouri. *Cartes de saillance et évaluation de la qualité des maillages 3D*. PhD thesis, Normandie Université, France, 2016.

[41] Sarra BABAHENINI. *L'apport de la perception/l'attention visuelle à l'amélioration de la fusion d'images multifocales*. PhD thesis, Université de mohamed kheider biskra, 2021.

[42] Vincent Ricordel, Junle Wang, Josselin Gautier, Olivier Le Meur, and Emilie Bosc. Perceptual modelling for 2d and 3d. 2010.

[43] Christian Scheier and Steffen Egner. Visual attention in a mobile robot. In *ISIE'97 Proceeding of the IEEE International Symposium on Industrial Electronics*, volume 1, pages SS48–SS52. IEEE, 1997.

[44] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme : a database and web-based tool for image annotation. *International journal of computer vision*, 77(1) :157–173, 2008.

[45] Gert Kootstra, Arco Nederveen, and Bart De Boer. Paying attention to symmetry. In *British Machine Vision Conference (BMVC2008)*, pages 1115–1125. The British Machine Vision Association and Society for Pattern Recognition, 2008.

[46] Tie Liu, Zejian Yuan, Jian Sun, Jingdong Wang, Nanning Zheng, Xiaoou Tang, and Heung-Yeung Shum. Learning to detect a salient object. *IEEE Transactions on Pattern analysis and machine intelligence*, 33(2) :353–367, 2010.

[47] Vidhya Navalpakkam and Laurent Itti. An integrated model of top-down and bottom-up attention for optimizing detection speed. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2049–2056. IEEE, 2006.

[48] Cynthia Breazeal and Brian Scassellati. A context-dependent attention system for a social robot. *rn*, 255(3), 1999.

[49] John Duncan and Glyn W Humphreys. Visual search and stimulus similarity. *Psychological review*, 96(3) :433, 1989.

[50] Sophie Marat. *Modèles de saillance visuelle par fusion d'informations sur la luminance, le mouvement et les visages pour la prédiction de mouvements oculaires lors de l'exploration de vidéos*. PhD thesis, Université Joseph-Fourier-Grenoble I, 2010.

[51] Russell Reinhart. A model of saliency, 2018. . https ://pdfs.semanticscholar.org/4816/f0b6f0d05da3901441bfa5cc7be044b4da8b.pdf.

[52] Akisato Kimura, Ryo Yonetani, and Takatsugu Hirayama. Computational models of human visual attention and their implementations : A survey. *IEICE TRANSACTIONS on Information and Systems*, 96(3) :562–578, 2013.

[53] Gunhee Kim, Daniel Huber, and Martial Hebert. Segmentation of salient regions in outdoor scenes using imagery and 3-d data. In *2008 IEEE Workshop on Applications of Computer Vision*, pages 1–8. IEEE, 2008.

[54] Audie G Leventhal et al. *Neurological Basis of Visual Function*, volume 4. Boca Raton : CRC Press, 1991.

[55] Laurent Itti and Ali Borji. Computational models : Bottom-up and top-down aspects. *arXiv preprint arXiv :1510.07748*, 2015.

[56] Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature reviews neuroscience*, 2(3) :194–203, 2001.

[57] Laurent Itti. *Models of bottom-up and top-down visual attention*. California Institute of Technology, 2000.

[58] John K Tsotsos, Scan M Culhane, Winky Yan Kei Wai, Yuzhong Lai, Neal Davis, and Fernando Nuflo. Modeling visual attention via selective tuning. *Artificial intelligence*, 78(1-2) :507–545, 1995.

[59] Laurent Itti and Christof Koch. Comparison of feature combination strategies for saliency-based visual attention systems. In *Human vision and electronic imaging IV*, volume 3644, pages 473–482. SPIE, 1999.

[60] Christian Boucheny. *Visualisation scientifique de grands volumes de données : Pour une approche perceptive*. PhD thesis, Université Joseph-Fourier-Grenoble I, 2009.

[61] Konstantinos G Derpanis. The gaussian pyramid. 2005.

[62] Rein van den Boomgaard. Image interpolation, 2017. . https ://staff.fnwi.uva.nl/r.vandenboomgaard/IPCV20172018/LectureNotes/IP/Images/ImageInterpolation.h

[63] Gloria Bueno García, Oscar Deniz Suarez, José Luis Espinosa Aranda, Jesus Salido Tercero, Ismael Serrano Gracia, and Noelia Vállez Enano. *Learning image processing with OpenCV*. Packt Publishing Birmingham, 2015.