Projet Tutoré - Classification par la théorie des jeux (Segmentation d'images)

Première année de Master Informatique

Adam GOUGUET 21710788
Aurélien THOREL 21605454

Université de Caen Normandie - Campus 2

30 mars 2021

Table des matières

1 Introduction

"Dans ce projet, on souhaite appliquer des méthodes issues de la théorie des jeux à l'optimisation de fonctions d'énergies. La minimisation de fonctions d'énergies trouve un grand succès dans plusieurs domaines.

Le but est d'implémenter une approche d'optimisation de fonctions d'énergies indépendamment de leurs propriétés de convexité. Le problème sera formulé sous la forme d'un jeu stratégique avec des applications diverses en classification et clustering multicritères."

Pour répondre à cette problématique, nous allons nous intéresser à un domaine en particulier, la segmentation d'image. Nous allons voir si il existe un moyen d'implémenter la théorie des jeux a ce domaine pour améliorer et/ou optimiser les résultats.

Dans ce rapport, nous proposons de tirer parti de la théorie des jeux dans la segmentation d'images par fusion de résultats. Ainsi, le jeu présenté est coopératif de telle sorte que les deux joueurs représentés par les deux modules de segmentation (basé sur la région et basé sur les bords) essaient une coalition pour améliorer la valeur d'une fonction caractéristique commune.

Les pixels impliqués sont ceux générés par la coopération par fusion des résultats entre le détecteur de bord (contour actif) et le détecteur de région (région croissante) posant un problème de prise de décision. L'ajout ou la suppression d'un pixel (vers / depuis) la région d'intérêt dépend fortement de la valeur de la fonction caractéristique.

Les procédures et les théories qui sont présenter dans cette article sont basé sur le document [6] ainsi que la théorie des jeux proposée par Chakraborty et Duncan [2].

Ensuite, et pour étudier l'efficacité et la robustesse de notre approche, nous avons proposé de généraliser nos expérimentations, en appliquant cette technique sur une variété d'images de différents types.

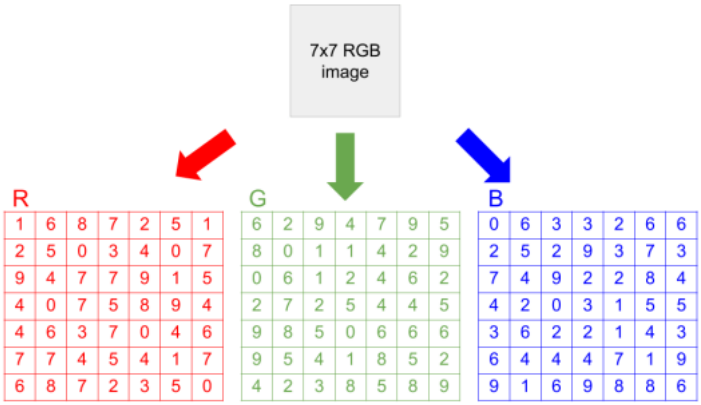
Mots clés : Image, Pixel, Graphe, Segmentation par approche de régions, Segmentation par approche de frontières, Segmentation par graphe, Théorie des jeux, Équilibre de Nash, Fonction caractéristique.

2 Données : Images, Graphes

2.1 Image

Une image peut être décomposée de deux manières différentes.

— Une matrice 2D pour une image en noir en blanc, avec chaque pixels allant de 0 à 255.
— Une matrice 3D pour une image de couleurs, en 3 ou 4 couches, avec chaque pixels de chaque couche allant de 0 à 255.
  - Rouge
  - Vert
  - Bleu
  - Alpha (optionnelle, sert a la transparence du pixel)



RGB Image vers Matrix en 3 couches

2.2 Réseau de flux

Un réseau de flux $G = (V, E)$ est un graphe où chaque arête a une capacité et un flux. Deux sommets du flux réseau sont désignés comme étant respectivement le sommet source s et le puit t. Le but est de trouver le débit maximum qui pourrait être délivré de s à t, tout en satisfaisant les contraintes suivantes. [4]

— **Conservation du débit** : Chaque arêtes $(u, v) \in E$ a une capacité non négative $c(u, v)$ qui doit être supérieure ou égale à son débit $f(u, v)$.

$$0 \leq f(u, v) \leq c(u, v$$

— **Contrainte de capacité** : Pour tous les sommets $u \in V - s, t$, nous avons besoin que l'entrée équivaux à sa sortie.

$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$$

Mais s peut avoir une entrée illimité, et t peut avoir un débit illimité.

Le flux du réseau est le flux qui peut être envoyé par un chemin de s à t, qui, par les contrainte du flux, est égal à l'entrée de s ou à la sortie de t. Une coupe s/t est un partitionnement des sommets en deux sous-ensembles disjoints tels que l'un contient s et l'autre t. La valeur d'une coupe s/t est la somme des flux des arêtes traversant la coupe.

2.3 Pour le projet

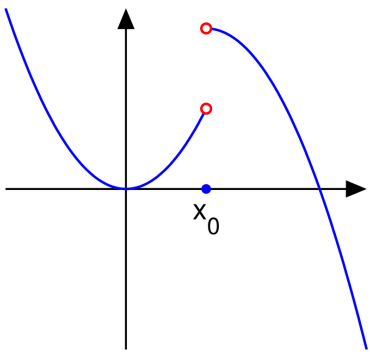
Concernant nos données, nous avons décider de travailler sur des graphes car la structure ressemblent à des images, nous pouvons considérer les pixels comme des noeuds et nous pouvons lier les noeuds à ces quatre voisin comme sur une image. Nous avons décider cela car nos recherche se porteront, à terme, sur plusieurs type de données différentes (images, texte, son, ...), pour cela les graphes sont une bonne forme de données car on peut rapporter n'importe quelle données à un graphe grâce notamment aux bases de données orientée graphe. Donc dans notre application nous ne travaillons pas sur les images mais sur des graphes, pour ce faire nous avons des méthodes pour transformer une image en graphe ou inversement un graphe en image.

3 Segmentation d'image

La segmentation d'image joue un rôle clé dans l'analyse d'image. Il existe principalement deux approches de segmentation. L'approche de segmentation basée sur les bords qui localise les limites des objets et l'approche de segmentation basée sur les régions qui partitionne l'image en un ensemble de régions. Chaque région définit un ou plusieurs objets connectés. Une segmentation d'image est la partition d'une image en un ensemble de régions/pixels distinctes (qui ne se chevauchent pas) et dont l'union est forme l'image entière.

Ces deux méthodes sont distinctes se reposent sur deux propriétés entre pixels voisins : la discontinuité et la similitude. La discontinuité est utilisée par l'approche de segmentation basée sur les bords (frontière), tandis que la similitude des pixels est utilisée par l'approche de segmentation basée sur la région.

Similitude : les objets de même couleurs sont semblable (*wikipédia*)



Discontinuétés : ici la fonction est discontinue au point x_0 (*wikipédia*)

Pour l'approche de segmentation basée sur la région, nous allons nous servir des valeurs des pixels pour différencier les différentes régions présente dans l'image. Quand à l'approche basée sur les bords, on va chercher dans l'image, comme sur une fonction, les points où les valeurs des pixels de l'image sont discontinue pour trouver les bords.

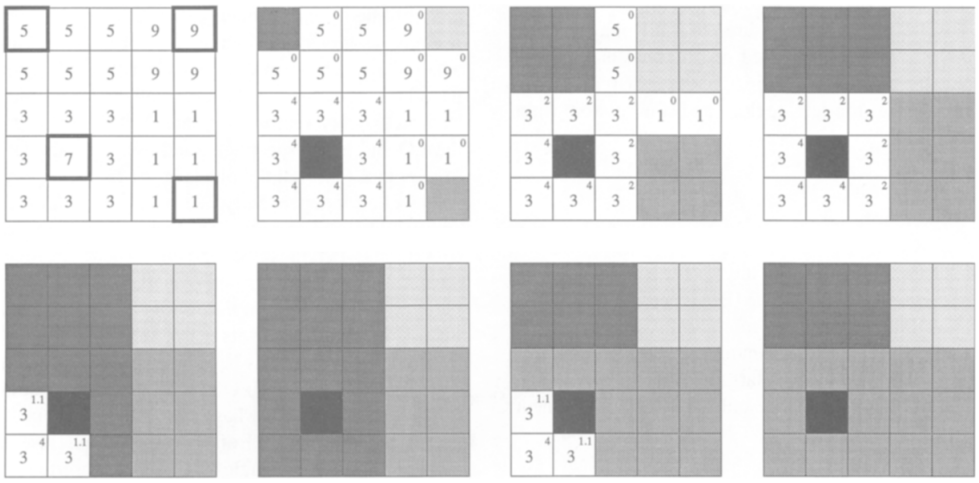
3.1 Segmentation par approche régions

Comme dit précédemment, la méthodes par approche de régions consiste à diviser une image en plusieurs régions distinctes. Contrairement à l'approche par frontières cette approche prend en compte le contenue de la région concerné. Il existe de multiple façon d'aborder cette méthode, voici les plus courantes : *Region growing*, *Region Splitting*, *Region Merging* et *Region Splitting and Merging*

Nous avons étudier pour la problématique qui nous intéresse la technique Region growing.

3.1.1 Region growing

Region growing est une technique qui se base et qui néccesite une "seed", c'est à dire, un ensemble de pixels comme point de départ pour construire la segmentation, ce sont les points de départ qui vont servir à notre algorithme. Ensuite, chaque pixels adjacent se voient attribuer une valeurs en fonction de leurs homogénéité avec leurs voisins (niveaux de gris, couleurs similaire, ...), ainsi on propage ce calcul sur toute l'image.



Exemple de la technique Region Growing (*An improved seeded region growing algorithm*)

Le processus s'arrête lorsque tout les pixels ont été attribuer à une région (ont reçu un label/une valeur en fonction de la région). Nous avons choisit cette méthode car elle simple d'utilisation, facile à comprendre et rapide. Nos recherche ne nécessitant pas de méthode précise à ce niveaux ici, nous avons choisit la simplicité. Voici le pseudo code permettant une segmentation grâce à la technique de Region growing :

Algorithme 1 : Region Growing

Entrées : Image I, valeur *seuil*
Output : Une segmentation de I (image)
// On initialise l'image résultat;
$R \leftarrow$ nouvelle image se basant sur I (taille);
// On choisit une méthode de sélection pour les premiers pixels;
pixels $\leftarrow$ méthode de sélection des pixels initiales;
pour *tous les pixels p dans pixels et qui n'a pas encore été visité* **faire**
  $R \leftarrow$ propagation_de_region(I, R, *seuil*, p);
retourner R;

On créer une région pour chaque pixels initials (Algorithme 2 : propagation_de_region), si il ont la même valeurs (exemple) alors on fusion les régions pour en créer une seule. A la fin de l'algorithme, tous les pixels de l'image sont doter d'une région.

Algorithme 2 : propagation_de_region

Entrées : Image I, Image R, valeur *seuil*, pixel p
Output : Image R avec une nouvelle région
$Q \leftarrow$ nouvelle pile;
$s \leftarrow$ nouvelle pile;
insérez p dans Q;
tant que *la pile Q n'est pas vide* **faire**
  pixel $p \leftarrow$ premier élement de Q;
  pour *tous les voisins v de p* **faire**
    si *valeur absolue de (v - p)* $\leq$ *seuil* **alors**
      insérez v dans Q;
  si *p n'est pas dans la pile s* **alors**
    insérez p dans s;
pour *chaque pixel p dans s* **faire**
  Attribuer la valeur $VRAI$ au pixel p dans l'image R
retourner R;

3.2 Segmentation par approche frontières

L'approche de frontières cherche à identifier les changements entre les régions. Un pixel appartenant à la frontière est un pixel appartenant à la limite entre deux ou plusieurs objets ayant des valeurs différentes. On cherche donc exploiter le fait qu'il existe une transition entre deux régions connexes. Il existe aussi plusieurs méthodes de segmentation utilisant les frontières : modèles déformables à l'aide de courbes paramétriques (courbe de Bézier, spline...) et les méthodes de dérivée (l'approche par gradient, l'approche laplacienne).

3.2.1 Munford-shah

On va s'intéresser à la conjecture de Mumford-shah car celle-ci est la base du model Chan-Vese qui nous sera nécessaire par la suite.

Les explications sont simplement théorique pour comprendre l'origine du model Chan-Vese et sont tirer du document *"Chan–Vese Segmentation"* par Pascal Getreuer [5], où l'on peut retrouver les explications sur la fonctionnelle de Mumford et Shah :

Pour voir où se situe la discontinuité au sein de l'image, et donc résoudre le problème de segmentation d'image, Mumford et Shah ont l'idée d'introduire une fonctionnelle qui, par minimisation, va chercher les points de l'image les plus discontinus. Mumford et Shah approxime l'image f par une régularité par morceaux de la fonction u comme solution du problème de minimisation :

$$arg \min_{(u,C))} \mu Length(C) + \lambda \int_{\Omega} (f(x) - u(x))^2 \, dx + \int_{\Omega \setminus C} ||\nabla u(x)||^2 \, dx \tag{1}$$

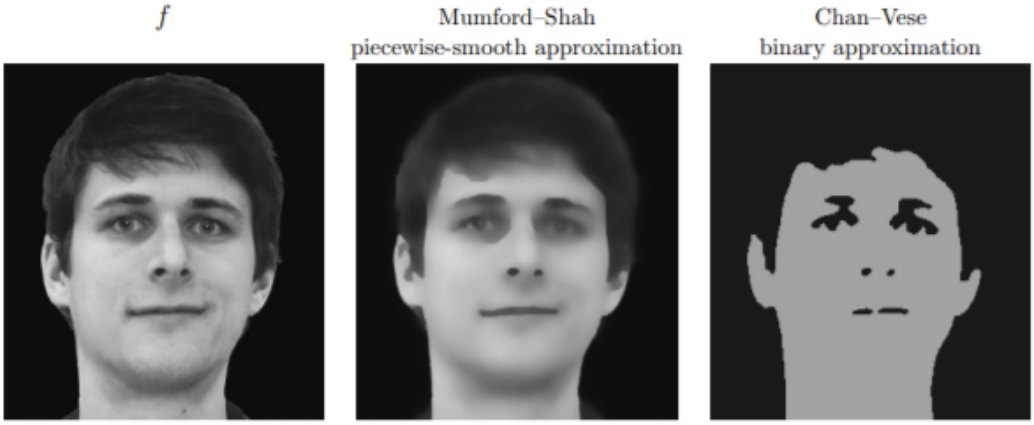
où C est un contour et où u est supposé être discontinue. Le premier terme existe pour assurer la régularité de C, le deuxième encourage u à être proche de f et enfin le troisième terme assure que u est différentiable sur $\Omega \setminus$ C. Ici, ∇u est le vecteur gradient de u et $\|.\|$ est la norme euclidienne.

3.2.2 Chan-Vese

Le modèle Chan-Vese est donc inspiré du modèle Mumford-Shah. Pour réaliser une segmentation à l'aide du modèle Chan-Vese, l'utilisateur à besoin de définir un contour initial C. L'objectif de l'algorithme de Chan-Vese est de minimiser la fonction d'énergie F(C) définie par :

$$\begin{aligned} F(C) =& \mu.Lenght(C) + v.Area(Inside(C)) + \lambda_1 \int_{Inside(C)} |u_0(x,y) - c_1|^2 \, dxdy \\ &+ \lambda_2 \int_{Outside(C)} |u_0(x,y) - c_2|^2 \, dxdy \end{aligned} \tag{2}$$

Où $\mu \geq 0$, v ≥ 0 et λ_1, $\lambda_2 > 0$ sont des paramètres fixée par l'utilisateur. Communément, on fixe les paramètres v $= 0$ et $\lambda_1 = \lambda_2 = 1$. Les variables c1 et c2 représente les moyennes des valeurs des points dans les régions intérieur et extérieur au contour, respectivement. Le premier terme contrôle la régularité en pénalisant la longueur. Le second pénalise la zone fermée de C pour contrôler sa taille. Les troisième et quatrième termes pénalisent la discordance entre le modèle constant par morceaux u et l'image d'entrée f. En trouvant un minimum local de ce problème, une segmentation est obtenue comme la meilleure approximation constante de régularisation par morceaux à deux phases de u sur l'image f.



Exemple de segmentation avec Mumford-Shah et Chan-Vese (*Chan–Vese Segmentation par Pascal Getreuer*)

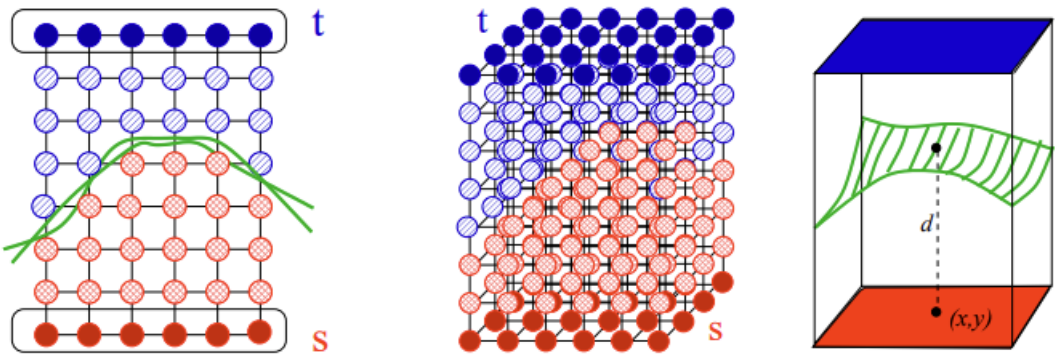
3.3 Segmentation par graphe

3.3.1 Graph Cut

Theorem 1 *Théorème flot-max/coupe-min — Pour tout graphe orienté G , tout couple (s,t) de sommets, et pour tout vecteur de capacités positives, la valeur maximale du flot de s à t est égale à la capacité d'une coupe minimale séparant s de t.*

Tout d'abord, un graphe (réseaux de flux) est construit sur la base de l'image d'entrée. Ensuite, un algorithme max-flow est exécuté sur le graphe afin de trouver le min-cut, qui produit la segmentation optimale.

Cette méthode se concentre sur l'utilisation de coupes graphiques pour diviser une image en segments d'arrière-plan et de premier plan (l'objet). Basée sur le papier de Yuri Boykov et Olga Veksler [3].



(a) a cut on a 2D grid (b) a cut and a separating hypersurface in 3D

Exemple de Graph Cut (2D et 3D)

3.3.2 Superpixels

Similaire au Graph Cut, l'image est converti en graphe. Mais contrairement au graph cut, son objectif est diffèrent, il cherche a réduire la taille du graphe crée [1].

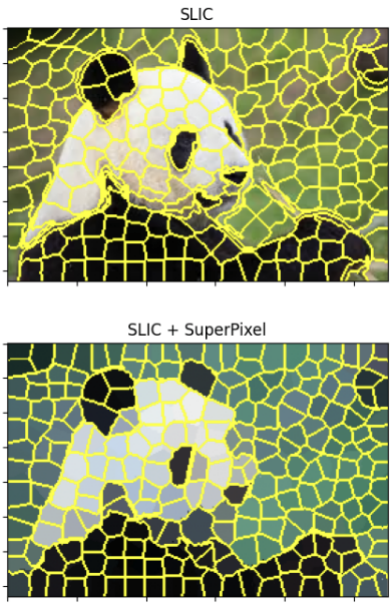
Les techniques de superpixel segmentent une image en régions en tenant compte des mesures de similitude définies à l'aide de caractéristiques perceptuelles. En d'autres termes, contrairement à watersheds et MSER, les techniques de superpixels créent des groupes de pixels qui se ressemblent. La but est d'obtenir des régions qui représentent des descriptions significatives avec beaucoup moins de données que ce n'est le cas lors de l'utilisation de tous les pixels d'une image.

Algorithme de clustering itératif linéaire simple

$$G(x,y) = ||I(x+1,y) - I(x-1,y)||^2 + ||I(x,y+1) - I(x,y-1)||^2 \tag{3}$$

Algorithme 3 : Efficient superpixel segmentation

Initialiser les centres de cluster $Ck = [lk, ak, bk, xk, yk]$ en échantillonnant des pixels sur une grille régulière de zone S;
répéter
  pour *Chaque centre de cluster* Ck **faire**
    Attribuez les meilleurs pixels correspondants à partir d'un voisinage carré 2S × 2S autour le centre du cluster en fonction de la mesure de distance (Eq. du dessus);
    Calculez les nouveaux centres de cluster et l'erreur résiduelle E (distance L1 entre les centres et centres recalculés);
jusqu'à $E \leq seuil$;
Appliquer la connectivité;

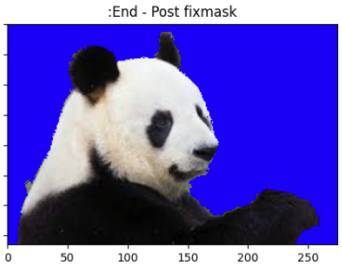


Exemple de segmentation avec Superpixels (Sans en haut, avec en bas)

3.4 Contrôle de la segmentation par méthodes des patches

Cette étape consiste a réduire la marge d'inconnue des éléments qui ne sont étiquetés en tant que objet ou fond. Pour cela, pour chaque pixels non labialisée, on crée un patch (une zone de 3x3 pixels) dans son voisinage, si un patch similaire (selon un certain dégrée de similitude/distance) existe et si il est labialisée. Si il est, alors on assigne le label du patch similaire au patch actuel.

Avant le contrôle de la segmentation par méthodes des patches



Après le contrôle de la segmentation par méthodes des patches

4 Théorie des jeux

Définition :La théorie des jeux est un domaine des mathématiques qui vise à étudier le comportement planifié, réel ou a posteriori justifié des agents face à différentes situations, et cherche à mettre en évidence des stratégies optimales. Il est basé sur le concept de jeu défini par un ensemble de joueurs (considérés comme des agents rationnels), toutes les stratégies possibles pour chaque joueur, et la spécification des gains des joueurs pour chaque combinaison de stratégies.

Il existe plusieurs type de jeux avec des caractéristique différentes, on peut voir les principales :

- Jeu coopératif ou non coopératif ;
- Jeu fini ou infini ;
- Jeu synchrone ou asynchrone.
- Jeu à somme nulle ou jeu à somme non nulle
- Jeu avec des informations incomplètes ou avec des informations complètes

4.1 Équilibre de nash

L'équilibre de Nash est un type de solution, une situation où aucun joueur ne regrette son choix. Proposé par John Forbes Nash en 1950, celle-ci est couramment utilisé en théorie des jeux. Un équilibre de Nash est, en effet, une combinaison de décisions individuelles, appelées « stratégies », où chacun prévoit correctement le choix des autres et maximise son gain, compte tenu de cette prévision. En fait, la grande et seule question

que se pose un joueur au moment de faire son choix est, en théorie des jeux : que va faire l'autre ? Ses croyances concernant le comportement des autres ont donc un rôle essentiel au moment de la décision. En fonction des croyances du joueur, les situations résultantes seront différentes.

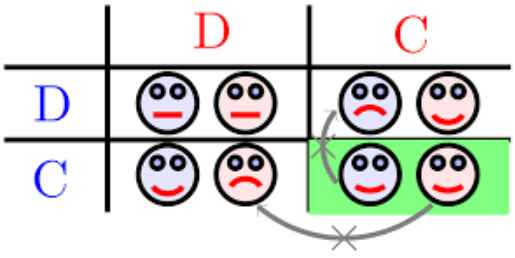
En théorie, un équilibre de Nash est un ensemble de stratégie $s* = ((s*_i)_{i\in[1,n]})$ où chaque joueur i joue sa stratégie optimal en prenant en considération les choix des autres joueur j et de leurs stratégies $s*_j$ et en maximisant son gain/minimisant ses pertes :

$$\forall(i,j) \in [1,n]^2, \forall s_i \in ((s*_i)_{i\in[1,n]}), \pi(s*_i, s*_j) \geq \pi(s_i, s*_j) \tag{4}$$

Donc un équilibre de nash existe si tous les joueurs n'ont aucun regret concernant leur choix sachant que les choix se font toujours simultanément dans la théorie des jeux. Pour visualiser et trouver un équilibre de Nash dans un jeu, on présente une matrice de coût qui résume toutes les situations possible au cours de ce même jeu.

4.1.1 Exemple connue : dilemme du prisonnier

Prenons un exemple bien connue pour expliquer comme trouve t-on un équilibre de Nash. Tout d'abord, le jeu implique deux joueur, deux prisonnier qui ont deux choix à faire, deux stratégies. La première stratégies consiste à dénoncer l'autre joueur, dans ce cas l'autre joueur est condamné a plus d'année de prison et le joueur qui dénonce a une peine réduite. Le deuxième choix est de se taire, dans ce cas la, la peine n'est pas réduite et l'autre joueur n'a pas une peine allongé. Donc si on compare les choix de chacun, on peut voir qu'il existe un équilibre de Nash, effectivement si chacun dénonce l'autre alors aucun des deux joueur n'aura de regret, car s'il avait choisi de se taire, alors que l'autre a opté pour la dénonciation, sa « tristesse » (perte) aurait augmenté. Cependant il existe un optimum qui est de se taire pour chacun des joueurs, mais en considèrent les choix, l'optimum n'est pas un équilibre de Nash. Voici deux matrice de coût qui représente la situation :



Matrice de coût symbolique de l'exemple du dilemme du prisonnier (*wikipédia*)

		Suspect A	
		Se tait	Dénonce
Suspect B	Se tait	6 mois pour A et B optimum	B est condamné à 10 ans A est relâché
	Dénonce	A et condamné à 10 ans B est relâché	5 ans pour A et B équilibre de Nash

Même matrice de coût mais cette fois ci avec de vrai valeurs (*quantmetry*)

4.2 Fonction caractéristique

La fonction caractéristique est une notion de la théorie des jeux coopératifs, qui se concentre sur la prédiction des coalitions qui se formeront, les actions conjointes des groupes et les retombées collectives qui en résultent. Il s'oppose à la théorie traditionnelle des jeux non coopératifs qui se concentre sur la prédiction des actions et des gains des joueurs individuels et sur l'analyse des équilibres de Nash.

La fonction caractéristique v(C) intervient donc lorsqu'un jeu est coopératif, cette fonction est la fonction pour laquelle la valeur d'une coalition entre les joueurs est maximum. La valeur de cette fonction est le gain collectifs de l'ensemble des joueurs qui participe à la coalition. Par exemple, si une coalition entre deux joueur obtient une valeur de 20 alors, on note v(j_1, j_2) = 20.

On peut donc dire qu'un jeu coopératif est décrit par toutes les valeurs des combinaisons de coalitions possibles. Dans un jeu à n joueurs alors il existe $2^n - 1$ coalitions non vide et autant de valeurs pour la fonction caractéristique. On définit généralement la valeur à 0 pour une coalition vide.

4.3 Formulation de notre recherche

Concernant notre problématique, on va étudier une méthode de segmentation en fusionnant deux ou plusieurs résultats de technique déjà existante. On souhaite donc que les différents modules de segmentation que l'on va utiliser coopère pour créer un gain comparer aux résultats des segmentations déjà obtenu.

On va donc s'intéresser à un jeu coopératif entre deux joueurs, représenter par des modules de segmentation. L'un sera une approche de régions (Region growing) quant à l'autre nous avons choisit une approche par frontières (Chan-Vese) pour essayer de conserver les avantages des deux techniques en réduisant leurs défauts.

Nous somme donc dans le cas d'une coalition entre les deux module de segmentation, c'est à dire que nous allons essayer d'améliorer les résultats d'une fonction caractéristique commune.

4.3.1 Notre jeu pour la segmentation d'image

Notre jeu va utiliser et traiter comme donnée des images, qui sont des données finis et les joueurs n'ont aucune possibilité de rester bloquer dans le jeu. On va considérer les joueurs comme étant les module de segmentation Region growing j_1 et Chan-Vese j_2. Le but étant d'améliorer le gain finale de la segmentation d'image, les gains sont forcement non nulle. Nous avons donc au final un jeu coopératif, fini, à somme non nulle et avec des informations complètes.

Il nous faut donc définir une fonction caractéristique pour notre jeu. Pour ce faire nous allons proposer une fonction objective différente pour chaque joueur prenant en compte les deux joueurs, nous aurons donc deux fonctions objectives $F1$ et $F2$.

Concernant le joueur 1, pour le module à l'approche de région, on définit sa fonction objective comme étant :

$$\begin{aligned} F1(j_1, j_2) =& min_X[\sum_{i,j}[y_{i,j} - x_{i,j}]^2 + \lambda^2(\sum_{i,j}(x_{i,j} - x_{i-1,j})^2 + \sum_{i,j}(x_{i,j} - x_{i,j+1})^2)] \\ &+ \alpha[\sum_{(i,j)\in A_p}(x_{i,j} - u)^2 + \sum_{(i,j)\in \bar{A}_p}(x_{i,j} - v)^2] \end{aligned} \tag{5}$$

Où, A_p correspond aux points qui se trouve à l'intérieur du contour p et inversement, $\bar{A}_p$ sont les points qui sont à l'extérieur du contour p. Finalement, $A_p \cup \bar{A}_p$ = l'image entière. La variable y est l'intensité du point dans l'image d'origine et x est l'intensité du point dans l'image segmenter par j_1. u et v sont les moyennes des valeurs des points présent à l'intérieur et l'extérieur du contour dans l'image segmenter par j_2. Les variables λ et α sont de simple paramètre. Le premier terme de cette fonction permet de minimiser la différence entre les valeurs des pixels trouvé dans la région et donc de renforcé la continuité dans l'image. Alors que le deuxième terme essaye de faire correspondre la région de j_1 et le contour trouvé par j_2.

Il nous reste à définir la seconde fonction objective pour le joueur 2 j_2 :

$$F2(j_1, j_2) = argmax_p[M_{gradient}(I_g, p) + \beta M_{region}(I_r, p)] \tag{6}$$

Où p est le contour proposé par j_2, I_g est le gradient de l'image et I_r est l'image segmenté par j_1. $M_{gradient}$ représente la correspondance entre le gradient de l'image et le contour trouvé par j_1. Et M_{region} est la mesure entre l'image segmenté de j_1 et le contour p, β est son poids.

Pour calculer cette fonction, nous avons étudier le document [**deformale˙boudary**] qui utilise le théorème de Green pour obtenir :

$$argmax_p(\int_{C_p}\left\{K_1 I_g(x,y) + K_2\left[N_r(x,y)\frac{\partial x}{\partial t} + M_r(x,y)\frac{\partial y}{\partial t}\right]\right\}dt) \tag{7}$$

Où, C_p est le contour, K_1 et K_2 sont des paramètres et que M_r et N_r sont définie par :

$$\begin{aligned} M_r(x,y) &= \int_0^x I_r(z,y)dz \\ N_r(x,y) &= -\int_0^y I_r(x,z)dz \end{aligned} \tag{8}$$

I_r étant l'image segmenté par approche de région. Au final les fonction M_r et N_r sont simplement des sommes des valeurs des pixels de l'image I_r dans une direction (x pour M_r et y pour N_r). Les fonction M_r et N_r ne sont calculer qu'une fois au début de l 'algorithme.

Maintenant que nous avons nos deux fonction objective, une par joueur, il nous faut réaliser une fonction caractéristique pour le jeu. Donc on unifie nos deux fonction pour trouver celle-ci :

$$F = \frac{F^1_{i-1}(j_1, j_2) - F^1_i(j_1, j_2)}{F^1_{i-1}(j_1, j_2)} + \frac{F^2_i(j_1, j_2) - F^2_{i-1}(j_1, j_2)}{F^2_{i-1}(j_1, j_2)} \tag{9}$$

Voila donc notre fonction caractéristique, ajouter ou enlever un pixel de la région d'intérêt va fortement dépendre de la valeur de cette fonction. L'index i permet de savoir si l'on à pris en compte le pixel i (i-1 pour non et i pour oui). Cette fonction permet de prendre en compte les deux fonctions objectives définie ci-dessus mais elle permet aussi de normaliser les valeurs d'amélioration et de dégradation car on sait que les variations de la fontion F^1 sont toujours supérieur à la fonction F^2, ces fonctions sont incommensurables.

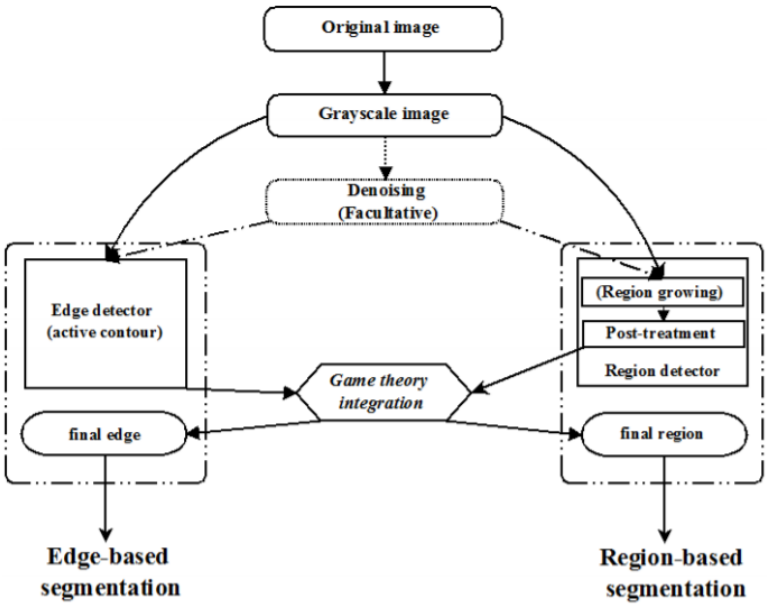


Schéma pour notre application

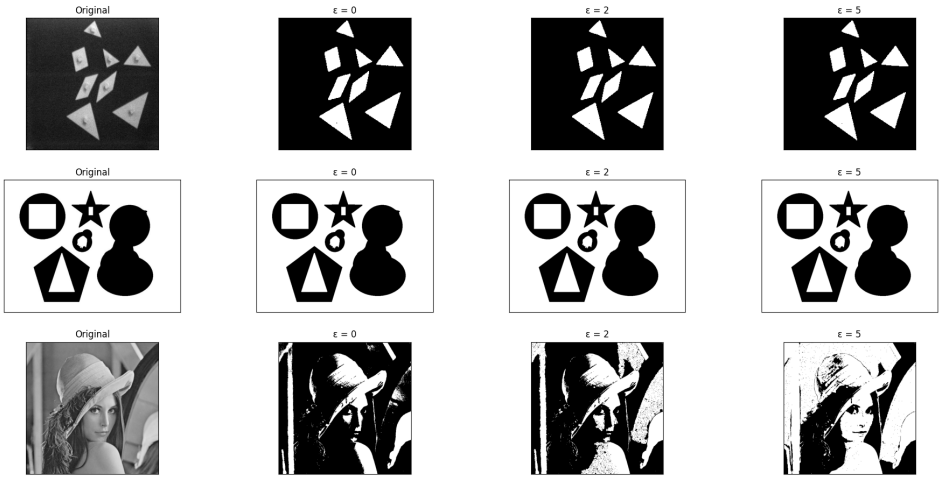
5 Application / En pratique

Pour réaliser l'application, nous avons décider d'utiliser le langage Python car il nous semblait simple d'accès et nous avions déjà travailler avec. Nous savions aussi qu'il exister beaucoup de librairie sur l'imagerie à notre disposition si besoin.

5.1 Segmentations d'image

5.1.1 Region growing

Pour la segmentation à approche de région, nous avons implémenter celle-ci en reprennent l'algorithme 1 de ce rapport mais en prenant des graphes au lieu des images. Voici un des résultats que peut fournir notre programme :



Résultat de notre algorithme Region Growing en fonction de ε(0, 2 et 5)

Où ϵ est le seuil donné à l'algorithme.

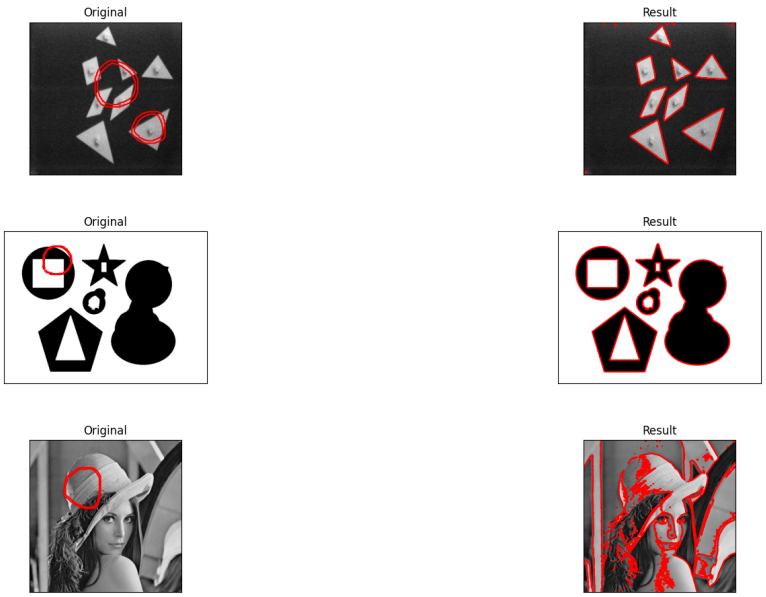
5.1.2 Chan-Vese

L'implémentation de la méthode de Chan-Vese n'est pas compliqué ni très longue mais il existe quelque subtilité. Par exemple si l'on reprend la formule (2) de ce rapport, on peut voir qu'il existe des intégrale mais une "vraie intégration" c'est dans un domaine continu et (souvent) infini. En informatique, le continu n'existe pas, et l'infini encore moins. Les intégrales sont donc approximées par des sommes. Pour l'imagerie, on est aussi dans un domaine discontinu et fini, puisque les images ont des bordures et que le plus petit élément d'une image est le pixel. Donc il est vraisemblable que si en théorie nous avons une intégrale, en pratique nous devons l'obtenir avec une somme.

En quelques itération, un contour se dessine très bien sur des forme simple et nous avons un contour qui est assez grossier sur des images avec des détails.

Algorithme 4 : Chan-Vese

Entrées : Graphe G, Contour C, valeur *iteration*
Output : Graphe G segmenté
$Ccopy \leftarrow$ copie du contour C d'origine;
$iter \leftarrow 0$;
tant que $iter$!= $iteration$ **faire**
  $C_{inside} \leftarrow$ moyenne des valeurs des points à l'intérieur du contour $Ccopy$;
  $C_{outside} \leftarrow$ moyenne des valeurs des points à l'extérieur du contour $Ccopy$
  pour *tous les noeuds n de G* **faire**
    $Ccopy[n] = Ccopy[n] + ((G[n] - C_{inside})^2 - (G[n] - C_{outside})^2)$
  $iter \leftarrow iter + 1$
retourner $Ccopy$;

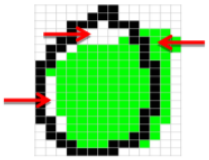


Résultat de l'algorithme Chan-Vese avec 2 itération

5.2 Approche coopérative entre méthodes de segmentations de régions et de frontières

Maintenant, notre but est d'arriver à fusionner ces résultats pour obtenir un résultat avec un gain, c'est à dire, avec des information supplémentaires (des contours plus précis et des détails) qui nous manquent avec les méthodes que nous avons vu. Comme vu dans la partie Théorie des jeux, nous allons implémenter les fonctions objectives pour essayer de créer la fonction caractéristique de notre jeu. Ensuite, nous observerons les résultats que l'on peut en ressortir, voir si cela apporte un réel gain par rapport à ce que cela nous coût en terme de temps de calcul. Pour cela il nous faut pour notre algorithme, le graphe/image d'origine, les deux segmentation obtenu par les modules de régions et de frontières et un nombre d'itération maximum.

Pour ne pas perdre du temps en calcul, nous sélectionnons seulement les noeuds intéressants, c'est à dire les noeuds qui sont soit à l'intérieur du contour de j_2 et pas dans la région de j_1 soit à l'intérieur de j_1 mais pas dans l'intérieur du contour de j_2.



Les noeuds sur lesquelles va s'appliquer l'algorithme se situe aux niveaux des flèches rouge

Nous allons donc faire une boucle sur tout les noeuds qui nous semble intéressant et calculer la fonction caractéristique pour ses noeuds seulement. Nous effectuons cela sur plusieurs itération jusqu'à ce qu'il n'y ai plus de noeuds qui nous semble intéressant.

D'après nos expériences et vu la logique de la coopération entre les deux segmentation, on à trouvé que en prenant les noeuds de valeurs égal au maximum de F et en les intégrant dans le contour de j_2, cela permettait à ce contour de ce rapprocher de la région. Inversement, en prenant les noeuds de valeurs égal au minimum de F et en les enlevant de la région de j_1, cela permettait à la région de ce rapprocher du contour.

Sur notre exemple ci-dessus, nous voyons que la partie en haut à droite de région (en vert) devrait disparaître, une partie au moins, et que la partie haute du contour (en noir) devrait "descendre" pour "coller" la région.

Algorithme 5 : Segmentation coopérative entre région growing et Chan-Vese

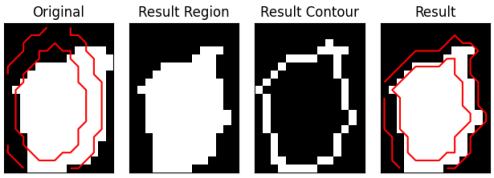
Entrées : Graphe G, Joueur 1 / Graphe segmenté par region growing $S1$,
Joueur 2 / Graphe segmenté par Chan-Vese $S2$
Output : Les Graphes des joueurs segmenté
$iter \leftarrow 0$;
tant que *iter != iteration* **faire**
  $nodes \leftarrow$ sélection des noeuds qui sont intéressant
  pour *tout noeud n dans nodes* **faire**
    On prend en compte le noeud n dans la segmentation;
    $F1_i$, $F2_i \leftarrow$ calcul de la fonction objective de $S1$, $S2$;
    On ne prend pas en compte le noeud n dans la segmentation ;
    $F1_{i-1}$, $F2_{i-1} \leftarrow$ calcul de la fonction objective de $S1$, $S2$;
    $F = \frac{F1_{i-1}(S1,S2)-F1_i(S1,S2)}{F1_{i-1}(S1,S2)} + \frac{F2_i(S1,S2)-F2_{i-1}(S1,S2)}{F2_{i-1}(S1,S2)}$;
  pour *tout noeud n dans F qui est égal au minimum de F* **faire**
    $S2[n] = VRAI$
  pour *tout noeud n dans F qui est égal au maximum de F* **faire**
    $S1[n] = FAUX$
  $iter \leftarrow iter + 1$
retourner $S1$ *et* $S2$;

5.3 Optimisation de l'algorithme

En effet pour diminuer le temps de calcul de cette algorithme nous avons effectuer des optimisation. Si on prend les fonction objectives, au lieu de calculer à chaque fois l'entièreté de l'image/graphe, nous calculons seulement pour le noeud en cours et comparons au résultat précèdent. Par exemple, pour la fonction objective $F1$, nous calculons le min_x du premier terme seulement pour le noeud en cours et ces voisins ensuite on le compare au minimum que nous avions stocker avant ; pour le deuxième terme, on recalcule les moyenne des parties intérieur/extérieur du contour en fonction de la valeur du noeud en cours. Ce genre d'optimisation bout à bout nous à permis de rendre notre algorithme plus rapide. Notre algorithme effectuer en python se réalise en 37 minutes pour l'image "Lina", une image de 512x512 pixels.

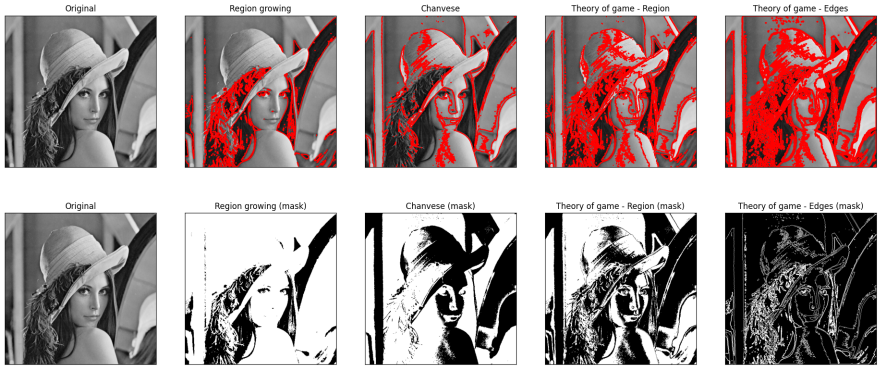
6 Résultats

Pour réaliser nos tests et vérifier que notre programme fonctionne nous nous sommes basé sur l'exemple avec la région verte.



Exemple simpliste pour tester notre approche (blanc région et rouge le contour)

Notre application fonctionne à priori, seulement sur cette exemple il n'y a pas de réel objet, ce sont un contour et une région défini arbitrairement. Nous avons donc effectuer un test sur l'image "Lina" :



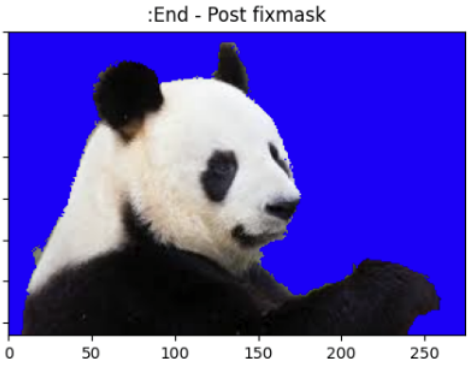
Résultat de notre méthode sur l'image "lina"
Ligne 1 : image original avec le contour
Ligne 2 : masque résultant de la méthode associé

On peut voir que sur la segmentation réaliser par "Region growing", il y a beaucoup de détail (cheveux, yeux) sur certaines partie mais il manque des objets comme le chapeau, le fond, etc ... Au contraire, sur la segmentation par Chan-Vese, on aperçoit tous les objets de la scène mais ils manque des détails que nous trouvons sur "Region growing". Grâce à notre approche nous arrivons à fusionner les deux résultats et faire ressortir le meilleurs des deux méthodes de segmentation utilisé. Les contours sont bien défini, on voit tous les détails, le fond apparaît clairement et on peut même observer des dégrader de couleurs et des textures qui n'exister même pas sur les segmentations avant (chapeau, visage).

Malheureusement, nous n'avons pas eu le temps d'effectuer de réel test sur un ensemble d'images connue (corpus) et de mesurer notre segmentation de manière numérique pour la comparer aux autres méthodes.

Concernant notre méthode de segmentation par graphe nous obtenons un résultat plus que satisfaisant : On obtient une très bonne segmentation sur un objet qui à des couleurs très différentes (valeurs de noeuds très différents) :

L'ours - Image d'origine



Exemple de l'ours via SegGraph (Bleu représente le fond)

7 Amélioration

7.1 Généraliser les méthodes de segmentation

En effet, on n'a vu que notre méthode fonctionne mais seulement avec deux technique spécifique et nous avons aussi vu qu'il existe de nombreuses façon de segmenter une image. Notre application devrait pouvoir fournir une approche de fusion entre toutes les segmentation, malheureusement, cela voudrait dire qu'il faudrait revoir la fonction caractéristique et les fonction objectives pour chaque pair de segmentation possible.

On peut même imaginer une fusion avec plus de deux modules de segmentation.

7.2 Sélection des noeud/pixels les plus intéressant

On n'a vu à plusieurs reprise qu'il y a possiblement dans une segmentation un moment ou l'on choisit certains pixels/noeuds à étudier pour segmenter la données, des point clé/intérêts en somme, il existe plusieurs algorithme qui permette de récupérer/repérée ses points.

7.2.1 Le détecteur de Moravec, le détecteur de Harris et autres méthodes

En récupérant ces point d'intérêts, on peut regarder si dans nos segmentation si il y a eu une segmentation autour de ces point d'intérêts. Cela permettrai d'améliorer un gain supplémentaire en amont de notre coopération, en fait cela révélerai des "zones" prioritaire où les deux segmentation doivent être d'accord ou donner une priorité a l'une des deux si l'autre n'est pas d'accord. Un type de segmentation connue est d'ailleurs appelé segmentation par point d'intérêts.

7.3 Optimisation

Concernant l'optimisation, nous avons vu que notre programme s'exécute en 37 minutes pour une image de seulement 512x512 pixels donc on ne peut pas dire que celui-ci soit très optimiser, il reste des endroit du code à optimiser de manière algorithmique. Malgré cela, si l'on imagine que nous avons fait notre maximum nous pouvons proposer d'utiliser la programmation parallèle pour diminuer le temps de calcul, l'imagerie est un domaine qui est parfait pour l'utiliser.

8 Conclusion

Au final le but de ce projet était d'expérimenter une classification en utilisant la théorie des jeux comme support pour améliorer le gain de la segmentation entre les données. Nous avons donc étudier les différents types de segmentation existante pour nous permettre de savoir ce qui nous serais possible de faire dans le projet. Le projet c'est concrétiser avec notre application où nous avons fusionner deux segmentation dans un jeu coopératif à deux joueurs où les joueur sont les modules de segmentation. Notre application fonctionne, le gain est augmenter et nous avons pu démontrer que les résultats font ressortir des information qui n'était pas dans les données de base.

Notre projet montre que la théorie des jeux permet d'obtenir un gain sur la segmentation d'image. En transformant une image en graphe nous avons vu qu'il est aussi possible de d'améliorer la segmentation sur toutes les données qui ont la possibilité de se rendre au format de graphe. Toutefois nous avons implémenter qu'une parti de la théorie des jeux, les jeux coopératifs, on peut donc se demander si les jeux non coopératif ont un réel gain sur la classification de données.

Au cours de ce projet nous avons étudier, découvert et implémenter en Python les différentes segmentations d'image, la théorie des jeux coopérative et non coopérative. Nous avons pour la première fois eu la possibilité d'effectuer de vrai recherche et d'essayer des choses de notre coté sans d'instruction précise sur la marche à suivre.

Références

[1] Radhakrishna Achanta et al. "SLIC superpixels". In : *Technical report, EPFL* (juin 2010).

[2] James S. Duncan Amit Chakraborty Lawrence H. Staib. "Deformable Boundary Finding in Medical Images by Integrating Gradient and Region Information". In : *IEEE* (1996).

[3] Y. Boykov et O. Veksler. "Graph Cuts in Vision and Graphics : Theories and Applications". In : *Handbook of Mathematical Models in Computer Vision*. Sous la dir. de Nikos Paragios, Yunmei Chen et Olivier Faugeras. Boston, MA : Springer US, 2006, p. 79-96. isbn : 978-0-387-28831-4. doi : 10.1007/0-387-28831-7_5. url : https://doi.org/10.1007/0-387-28831-7_5.

[4] Yuri Boykov et M. Jolly. "Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images". In : (2001).

[5] Pascal Getreuer. "Chan–Vese Segmentation". In : *Yale University* (2012).

[6] Karima Benatchba Omar Boudraa. "Region-Edge Cooperation for Image Segmentation Using Game Theory". In : *5th International Conference on Computer Science and Its Applications (CIIA)* (May 2015), p. 515-526.