Rapport sur le filtrage et détection de sources astronomiques faibles

Antonin JUQUEL, Raj Porus HIRUTHAYRAJ
E4FI, ESIEE Paris

1 Synthèse de l'article

Dans cette section, nous allons approfondir notre compréhension du document en examinant en détail chaque élément de celui-ci. Nous allons prendre le temps de bien comprendre chaque aspect du papier afin de pouvoir en discuter de manière éclairée et argumentée.

1.1 La problématique de l'article

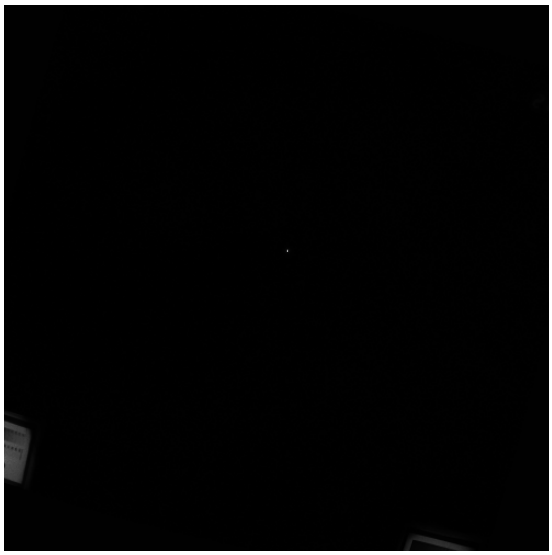
La problématique traité par ce papier découle du fait qu'en astronomie, les images contiennent une énorme quantié de sources lumineuses. L'extraction manuelle de chacun de ces sources n'est pas envisageable. De plus, beaucoup de ces sources ont une intensité faible, ils se rapprochent quasiment au niveau de bruit.

Dans ce contexte, il devient essentiel de mettre en place des méthodes automatisées permettant de détecter efficacement ces sources lumineuses faibles et étendues. C'est là que le filtrage d'attributs statistiques entre en jeu.

En utilisant des techniques statistiques avancées, ce filtrage permet de mettre en évidence ces sources faibles qui pourraient autrement être masquées par le bruit de fond. Cette méthode est particulièrement utile pour les observations à grand champ, où il y a un nombre élevé de sources dans l'image et où il est difficile de les détecter de manière visuelle.

Le papier que nous allons étudier en détail dans cette section présente une approche novatrice de filtrage d'attributs statistiques pour la détection de sources lumineuses faibles et étendues en astronomie. Nous allons examiner en profondeur les méthodes utilisées par les auteurs, ainsi que les résultats obtenus et leur pertinence pour la communauté scientifique. Enfin, nous allons discuter des limites et des perspectives de cette approche, ainsi que de son potentiel pour l'avenir de la recherche en astronomie.

Et après avoir appliqué l'algorithme MTObjects, on peut y voir des formes que l'on ne pouvait pas distinguer précédement.



(a) Avant MTObject

(b) Après MTObject

Figure 1: Astro1 Ultraviolet Imaging Telescope (512 x 512) primary array image

1.2 Méthode proposée dans l'article

Cet article propose une méthode pour extraire de manière automatique sur des photographies, ces sources lumineuses, que l'on a mentionné plus haut. Cette méthode s'appelle MTObjects.

Au cours de cet article, cette méthode sera comparé à la méthode SExtractor (Source Extractor) qui est une méthode état de l'art dans ce domaine. Ce dernier effectue son filtrage en utilisant un seuil de détection fixe, tandis que MTObjects propose d'utiliser un seuillage dynamique.

L'algorithme mtObjects prend en entrée une image I, une fonction nodeTest, un niveau de significance α, un gain g et un facteur de déplacement λ. Il renvoie un arbre Max-Tree M. Les nœuds de M correspondant aux objets sont marqués. L'algorithme mtObjects est décrit en détail dans l'article. Dans un premier temps on éstime la moyenne et la variance de l'arrière plan de l'image. On applique ensuite un seuillage à l'image I, en soustrayant la moyenne de l'arrière plan. On crée ensuite une représentation Max-Tree de l'image. On applique ensuite la fonction SignificantNodes sur l'arbre Max-Tree. On trouve ensuite les objets dans l'arbre Max-Tree. On déplace ensuite les objets vers le haut de l'arbre Max-Tree.

MTObjects utilise une représentation hiérarchique de l'image appelée arbre de composants ou arbre max pour détecter les sources, en s'appuyant sur des attributs associés à chaque nœud de l'arbre pour le filtrage ou la segmentation de l'image.

Pour détecter les sources, MTObjects parcourt l'arbre de haut en bas, en commençant par les nœuds les plus intenses et en descendant vers les nœuds de faible intensité. Les sources sont identifiées en tant que groupes de pixels connectés qui satisfont certaines conditions de seuil. Ensuite, la méthode utilise un processus de découpage quantifié pour séparer les sources imbriquées ou superposées.

Il semble que cette méthode a été conçue pour améliorer la détection de sources de faible intensité dans les images, en particulier dans les images de galaxies en interaction ou superposées. Elle vise également à améliorer la méthode de seuil fixe utilisée par SExtractor en utilisant une approche plus adaptative qui tient compte des propriétés des objets détectés.

En utilisant une représentation hiérarchique de l'image appelée arbre de composants ou arbre max, et en associant des attributs à chaque nœud de l'arbre pour le filtrage ou la segmentation de l'image, MTObjects peut détecter les sources d'image de manière plus précise en se basant sur des caractéristiques de l'image telles que l'intensité, la forme et la position des sources. Cette méthode est particulièrement utile pour la détection de sources de faible intensité dans les images, comme celles présentes dans les images de galaxies en interaction ou superposées.

1.3 Validation et résultat

Ils ont réalisé des tests sur différents types d'objets celestes, sur une image d'une simulation de 25 étoiles pour tester la détéction du background et du bruit avec la segmentation de SExtractor, sur des galaxies et autres objets avec des régions extérieures plus pâles et des objets imbriqués, sur une portion d'image représentant la fragmentation d'une fine lamentation faible entre deux galaxies, sur des voies de poussières et des artefacts.

Les tests précédent révèlent que de manière générale, MTObjects conserve mieux les contours de faible intensité des objets et les objets imbriqués. MTObjects est plus rapide que SExtractor. MTObjects est plus robuste que SExtractor face au bruit et aux artefacts. MTObjects est plus fiable que SExtractor face à la fragmentation d'objets. MTObjects gère mieux que la présence de poussières que SExtractor.

2 Étude de l'implémentation

2.1 Inspection du projet

Le code fourni permet de reproduire les expériences présentées dans l'article, cependant les images utilisées ne sont pas fournies. Il est donc nécessaire de les trouver sur internet ou de les générer soi-même.

Les fichiers nécessaires à l'installation du code sont fournis, mais il est nécessaire de les compiler soi-même en lançant le fichier 'recompile.sh' et d'installer les dépendances nécessaires à Python décrites dans le 'README.rst'.

Il n'y a pas d'interface graphique. L'utilisation du code se fait donc uniquement en ligne de commande et est donc limitée à des utilisateurs avertis.

Le code est bien structuré, il est facile à comprendre et à modifier, chaque fonction est commentée et expliquée. Plus particulièrement, le code est divisé en plusieurs fichiers, ce qui facilite la compréhension et la modification du code. De plus, il est bien documenté par un README qui contient des exemples d'utilisation.

Nous pensons que le code peut être réutilisé dans un autre contexte, mais il faudrait le modifier pour qu'il puisse être utilisé avec d'autres types d'images que les FITS, en effet la détéction de sources d'image est un problème récurrent dans de nombreux domaines de l'astrophysique, et il est donc possible que ce code puisse être réutilisé dans d'autres domaines.

2.2 Comprendre les algorithmes

- Algorithm 1: **IsFlat(T, α)** du paper se trouve dans **background.py** sous le nom de fonction **check_tile_is_flat**, elle vérifie si une tuile (c'est-à-dire une portion de l'image) est "plate", c'est-à-dire qu'elle respecte certaines conditions de normalité et d'homogénéité.

- Algorithm 2: **SignificantNodes(M, nodeTest, α, g, $\hat{\sigma}^2_{bg}$)** est présent dans **mt_objects.c**, et répond au nom de la fonction **mt_significant_nodes_up** s'occupe de mettre à jour la branche principale (main branch) d'un noeud donné dans l'arbre maxtree. Elle fait cela en comparant la taille de la branche principale actuelle du noeud avec celle du noeud passé en argument. Si la taille du noeud passé en argument est plus grande, la fonction met à jour la branche principale du noeud avec le noeud passé en argument. Et **mt_significant_nodes_down** s'occupe de mettre à jour la liste des noeuds significatifs descendants (significant descendants) d'un noeud donné dans l'arbre maxtree. Elle parcourt la liste des descendants du noeud et vérifie si leur taille est supérieure à un seuil de signification défini. Si c'est le cas, elle ajoute le descendant à la liste des noeuds significatifs descendants du noeud.

- Algorithm 3: **FindObjects(M)** est implémenté dans **mt_objects.c** sous la fonction **mt_find_objects**, elle est utilisée pour trouver les objets significatifs dans une image à l'aide de l'algorithme de maxtree. Elle prend en entrée une structure de données **mt_object_data** qui contient des informations sur l'image et l'arbre maxtree qui a été construit à partir de cette image.

- Algorithm 4: **MoveUp(M, λ, g, $\hat{\sigma}^2_{bg}$)** est lui aussi dans **mt_objects.c**, et est désormais la fonction **mt_move_up**, elle se charge de mettre à jour la liste des noeuds significatifs descendants (significant descendants) d'un noeud donné dans l'arbre maxtree. Elle parcourt les ancêtres du noeud et vérifie

si leur taille est supérieure à un seuil de signification défini. Si c'est le cas, elle ajoute l'ancêtre à la liste des noeuds significatifs descendants du noeud.

- Algorithm 5: **MTObjects(I, nodeTest, α, g, λ)** de la même manière que le dernier, il se trouve dans **mt_objects.c**, et la fonction est appelé **mt_objects**, il est utilisée pour trouver les objets significatifs dans une image à l'aide de l'algorithme de maxtree. Elle prend en entrée une structure de données **mt_object_data** qui contient des informations sur l'image et l'arbre maxtree qui a été construit à partir de cette image, ainsi que sur les objets significatifs trouvés jusqu'à présent.

3 Expérimentations

Lors des expériences avec le code on a pu le lancer avec plusieurs paramètres différents comme par exemple, la verbosité, qui nous permettais de mieux comprendre ce que le programme fait :

```
python3 mto.py -verbosity 2 astro.fits

---Image dimensions---
Height =  512
Width =  512
Size =  262144

---Estimating background---
Using a tile size of 64 in the background
Number of usable tiles: 3
0.07425 seconds to estimate background.

---Background Estimates---
Background mean:  1.2580513321838228e-16
Background variance:  1.3691925147718795e-31
Gain:  1.4451246971101104e+16  electrons/ADU

---Building Maxtree---
4 neighbors connectivity.
0.07411 seconds to create max tree.

---Finding Objects---
Using significance test 4 (powerAlt given area, 3x3 Gaussian filter with FWHM = 2).
Number of nodes to be tested: 75947.
6410 significant nodes.
Found 24 objects (including 14 nested).
0.05976 seconds to find objects.

---Generating segmentation map---
Saved output to out.png

---Calculating parameters---
Saved parameters to parameters.csv
```

Voici quelques paramètres qui peuvent influencer le traitement d'une image :

- `gain` : un nombre flottant représentant le gain en électrons par unité de valeur ADU (analog-to-digital unit). La valeur par défaut est -1.
- `bg_mean` : un nombre flottant représentant la moyenne de l'arrière-plan. La valeur par défaut est `None`.
- `bg_variance` : un nombre flottant représentant la variance de l'arrière-plan. La valeur par défaut est -1.
- `alpha` : un nombre décimal représentant le niveau de signification. La valeur par défaut est 1e-6.

- `move_factor` : un nombre positif représentant le déplacement du marqueur de l'objet. La valeur par défaut est 0.5.
- `min_distance` : un nombre positif représentant la distance de luminosité minimale entre les objets. La valeur par défaut est 0.0.

Nous avons expérimenté en utilisant des images que nous avons trouvées sur Internet, comme celles que vous pouvez voir au début de notre rapport. Nous avons également modifié certains paramètres pour essayer différentes valeurs autres que celles par défaut, mais les résultats de ces tests n'ont pas été concluants. Il est évident que l'algorithme fonctionne correctement, mais nous regrettons de ne pas avoir eu accès au code de SExtractor afin de pouvoir le tester et le comparer nous-mêmes.