Rapport algorithme de localisation des ordres et de l'inter-voie

Arthur Fezard

19/06/2017

1 Introduction

Dans ce rapport, nous allons voir ensemble ce qu'il faut savoir sur le programme qui permet de localiser les différents ordres sur le CCD. A l'heure actuelle, ce programme a été développé et testé sur Narval. Il sera nécessaire de le modifier pour qu'il puisse fonctionner sur Neo-Narval.

Le code est suffisamment commenté pour s'y retrouver et savoir ce qu'il se passe mais ce rapport a pour but d'expliquer pourquoi les différents choix ont été faits. Il comporte 4 principales fonctions :

fonction chelli_shift : Basé sur l'article de Chelli, elle va nous permettre de localiser les ordres avec la précision recherchée.

fonction Ref_Localisation : permet de trouver ou se situe chaque ordre sur la ligne centrale.

fonction Order_localisation : permet de trouver l'enveloppe globale de chaque ordre.

fonctions lanes : permet de trouver l'inter_voie pour chaque ordre

Il y a également plusieurs fonctions annexes qui permettent de chercher les différents fichiers et de gérer la création et l'utilisation des différents pickles. Ces dernières seront détaillées à la fin du document.

Petite précision, les limites des ordres (et également des inter voies) sont définies par les minimums locaux en intensité.

2 Fonction Ref_Localisation.

2.1 Utilité et préalable.

Cette première fonction permet de localiser ou se situe l'ordre sur la ligne centrale pour chaque ordre. Elle renvoie ensuite un tableau contenant tout les pixels identifiés.

Par exemple, sur la fonction de la figure (1), le tableau que renvoie la fonction Ref_Localisation est $[5, 36]$ Plusieurs problèmes se posent. Où est ce que l'on commence à chercher nos références ? Comment être surs que nous ne ratons pas d'ordre ? Comment ne pas confondre le bord de l'ordre avec un inter ordre ? ...

IMPORTANT : Pour cette fonction (et donc les suivantes), il est important que le CCD se présente sous la forme de la figure (2).

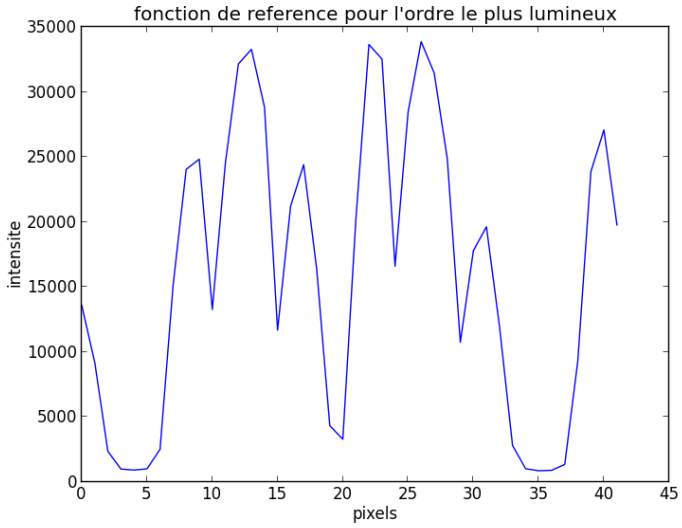

FIGURE 1 – exemple de fonction de reference

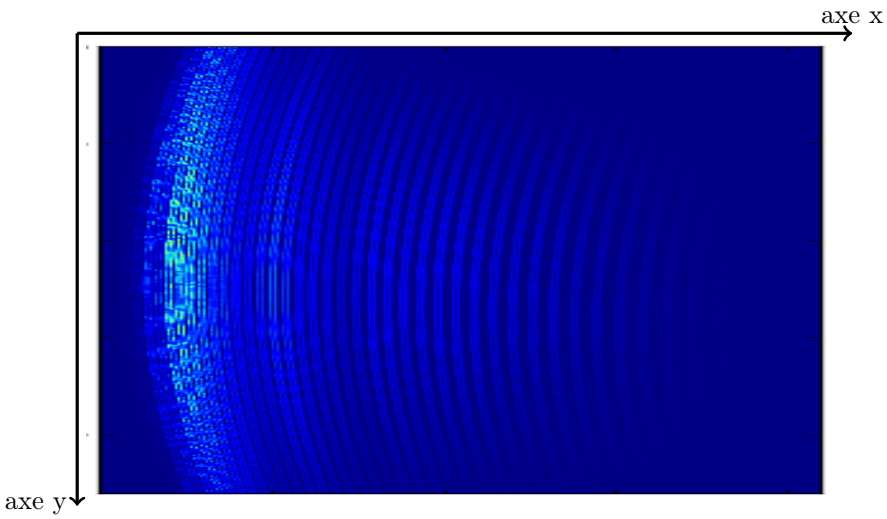

FIGURE 2 – CCD vierge

2.2 Principe de fonctionnement.

Tout d'abord, il faut savoir ou commence notre recherche. Plutôt que de fixer une valeur minimale au delà de laquelle nous considérons que nous sommes sur un ordre, nous allons partir de l'ordre le plus lumineux. En effet, sur la ligne centrale, il existe toujours un pic d'intensité correspondant à un ordre. Nous allons donc partir de ce pic d'intensité.

Nous identifions donc le pic d'intensité. Ensuite, nous définissons une fenêtre de recherche (de l'ordre de 20 pixels au début) et nous cherchons le minimum à droite et à gauche du pic d'intensité, au sein de cette fenêtre de recherche. Ainsi, on cherche le minimum à droite du pic entre : le pixel du pic et le pixel du pic + la longueur de la fenêtre de recherche. Nous venons donc de délimiter l'ordre le plus lumineux au niveau de la ligne centrale.

L'idée est d'ensuite prend le résultat de l'opération précédente et de l'utiliser comme fonction de référence pour faire une auto-corrélation de cette fonction avec l'ensemble de la ligne centrale. Comme chaque ordre a la même forme, les pics de l'auto-corrélation nous permettent de localiser

les autres ordres, en théorie. Seulement, près de l'ordre lumineux, les ordres sont trop proches et les pics correspondent à la fois à la localisation des ordres et des inter ordres comme le montre la figure (3). Cependant, lorsque l'on s'éloigne des ordres les plus lumineux, les pics de l'auto-corrélation correspondent aux ordres, comme le montre la figure (4).

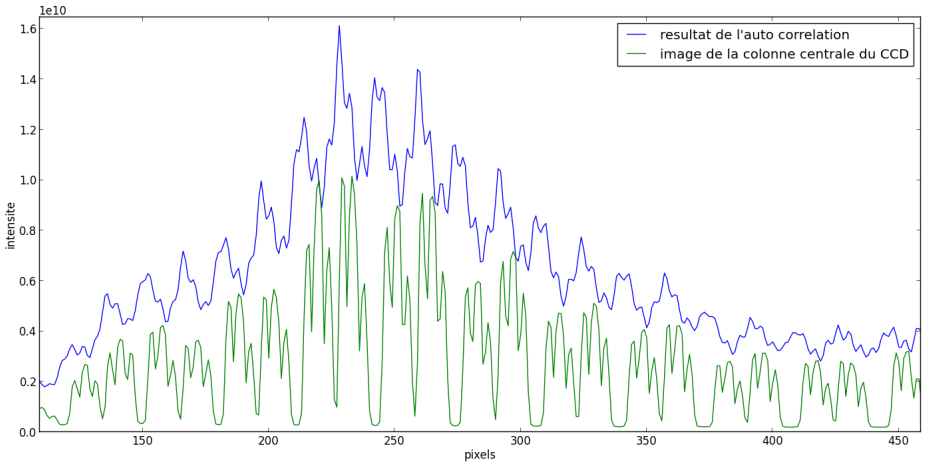

FIGURE 3 – résultat de la corrélation au niveaux de l'ordre le plus lumineux.

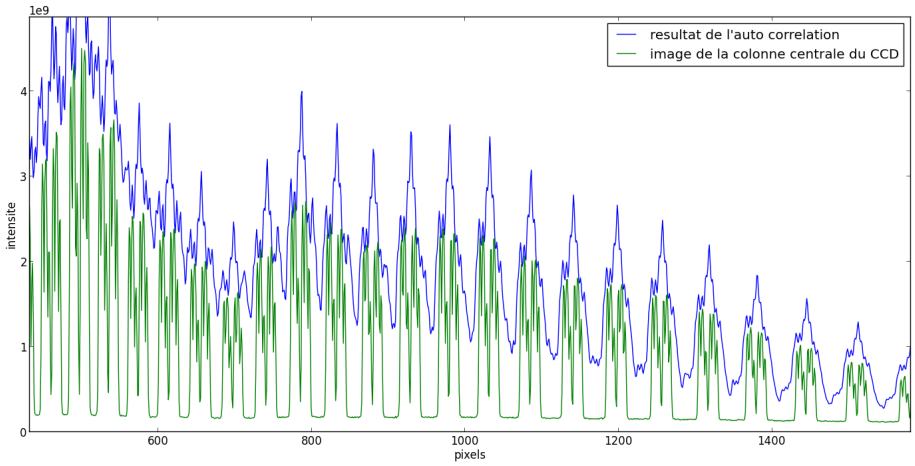

FIGURE 4 – résultat de la corrélation pour les derniers ordres.

Nous allons donc fixer arbitrairement une limite au delà de laquelle nous utiliserons la fonction d'auto corrélation. Cette limite dans le cas de Narval a été fixé au pixel 700. Pour trouver les ordres en dessous de ce pixel limite, il faut donc utiliser une autre méthode.

Pour localiser les fonctions de référence des ordres les plus lumineux (et les plus serrés), nous allons simplement utiliser une recherche de minimum local. Je vais me focaliser sur la recherche d'ordre à droite de l'ordre le plus lumineux, mais la démarche à gauche de ce dernier reste la même. De plus nous supposons que le bord droit de l'ordre m et le bord gauche de l'ordre $m+1$ coïncident.

En partant de l'ordre le plus lumineux, nous allons donc faire une recherche de minimum à droite du bord droit de l'ordre précédent. Nous recherchons ce minimum entre le pixel du bord droit + une marge de sécurité et le pixel du bord droit + cette marge + notre fenêtre de recherche. Cette marge de sécurité, fixée arbitrairement à 5 pixels, permet de s'assurer que nous commençons

notre recherche au niveau de l'ordre suivant. De plus, comme les ordres s'espacent de plus en plus, on augmente la fenêtre de recherche de 1 pixel à chaque itération.

Pour finir, au delà du pixel 700, nous utilisons la fonction d'auto corrélation calculée préalablement. Tout d'abord, nous avons le bord droit du dernier ordre calculé. Or, nous savons que les numéros de pixels concordent entre le CCD et la fonction d'auto corrélation calculée. Ainsi, nous allons utilisé le dernier bord droit calculé sur le CCD dans la fonction d'auto corrélation.
Dans la fonction d'auto corrélation, nous faisons une recherche de maximum dans une fenêtre de recherche à droite du bord droit (calculé préalablement sur le CCD), en prenant en compte une marge de sécurité. Le pixel qui nous est donnée correspond à la position de l'ordre sur le CCD. Enfin, nous trouvons le bord gauche sur le CCD en cherchant dans une fenêtre de recherche à droite du maximum précédemment localisé. Cette fonction est ainsi capable de nous renvoyer la position des ordres sur la ligne centrale.

2.3 Réserves et compléments.

J'ai limité au maximum les paramètres choisis arbitrairement afin que l'algorithme soit efficace quelques soit la situation. Le seul petit problème qui peut se présenter se situe pour les derniers ordres. En effet, l'inter ordre augmente de plus en plus et la recherche de minimum dans cet espace peut se situer plus ou moins loin de l'ordre recherché. Il se peut que sur un ordre il y ait 9 pixels entre le bord gauche et le début de l'ordre , et seulement 2 pixels pour l'ordre suivant. Mais dans ce cas nous sommes au moins surs de ne manquer aucun ordre. Il faut en plus également lui spécifier dans le programme combien d'ordres il doit chercher à gauche et à droite de l'ordre le plus lumineux.
De plus, il est prévu une option qui permet de prendre en compte s'il y a deux ou trois voies par ordre. Cela affecte seulement la taille initiale de la fenêtre de recherche et le seuil en pixels au delà duquel on utilise l'auto-corrélation.

3 Fonction Order_localisation

3.1 Utilisation

Cette fonction est le cœur de l'algorithme. En utilisant le résultat de la fonction précédente, nous allons délimiter chaque ordre sur toutes les lignes du CCD. Cet algorithme se base sur Chelli afin d'avoir la précision souhaitée. Cette fonction prend en entrée le fichier fit et un entier correspondant au nombre de voies par ordre. Elle renvoie une matrice dont les lignes correspondent aux lignes du CCD et donc les colonnes contiennent le numéro du pixels correspondant aux diverses enveloppes. Elle renvoie également un tableau contenant les épaisseurs des ordres aux centres, qui est supposée constante lors du calcul.Enfin, elle renvoie un tableau contenant les fonctions polynomiales qui ont servi pour calculer les bords des différents ordres.

3.2 Principe de fonctionnement

Tout d'abord, on récupère la liste des ordres sur la ligne centrale via la fonction précédente. Pour chaque ordre, l'image de l'ordre au centre va nous servir de vérité (ou de référence) pour le calcul de décalage. Pour chaque ordre, nous utilisons Chelli pour chaque ligne en la comparant avec la ligne centrale pour ainsi obtenir le décalage.
Plus en détails, en comparant la ligne centrale et la ligne en dessous. Les bords de l'ordre sur la ligne centrale nous donnent une idée d'où chercher l'ordre sur la ligne suivante, étant donné que la courbure est faible. Chelli nous donne ensuite le décalage précis entre la ligne centrale et la ligne adjacente. En ajoutant ce décalage au bord de la ligne centrale, nous obtenons l'enveloppe pour la ligne en dessous.
Enfin, la subtilité réside dans la donnée suivante : où faut-il chercher l'ordre dans la ligne suivante ? Pour se faire, nous ajoutons au bord de la ligne précédente (la ligne centrale ici donc) la somme du décalage trouvé pour la ligne actuelle et le décalage trouvé pour la ligne précédente. Cette somme est ensuite convertie en entier pour servir d'indice de recherche dans une matrice. Cette

fenêtre de recherche servira pour la ligne suivante. Cela permet de décaler la fenêtre de recherche suffisamment pour pouvoir suivre l'ordre sans sauter d'un ordre à un autre.
Pour finir, après avoir effectué cette opération sur toutes les lignes, nous "fittons" par un polynôme d'ordre 4 les données calculées pour chaque bord. Cela nous permet de nous affranchir des problèmes de bruit au bord du CCD pour les ordres peu lumineux. Ces données sont ensuite stockées dans une matrice où les lignes correspondent aux lignes du CCD et les colonnes correspondent aux bords des ordres. Une subtilité subsiste pour les derniers ordres les moins lumineux. Pour les 5 derniers ordres (et donc les 6 dernières limites), les coefficients du polynômes de fit sont calculés par rapport aux 2 précédents en effectuant une régression linéaire.

3.3 Résultat et précision

Cette section va surtout se concentrer sur le choix de la fonction fit et la précision associée. Tout d'abord, voici la différence entre les données sans fit et les données avec fit. Sur ces figures, les courbes en rouge sont les courbes données par la fonction en question. Les courbes en bleu sont celles données par la fonction lanes2 qui sera détaillé dans la prochaine section. On remarque surtout l'intérêt du fit entre les figures (7) et (8). En effet, sur la première le bruit est visible et fausse le résultat alors que sur la seconde ce problème est éliminé.

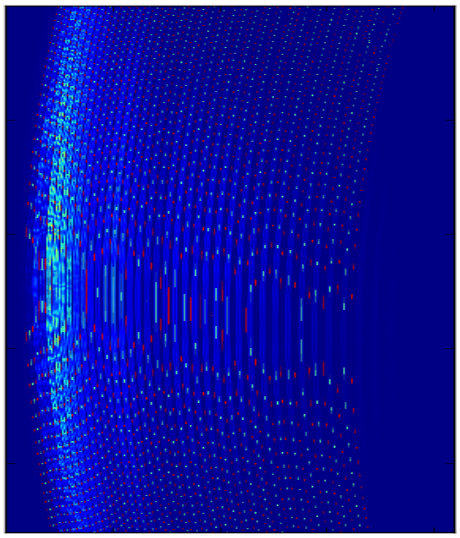

FIGURE 5 – CCD sans fit

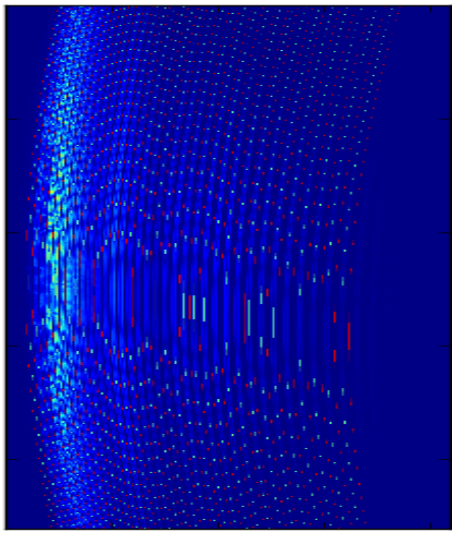

FIGURE 6 – CCD avec fit

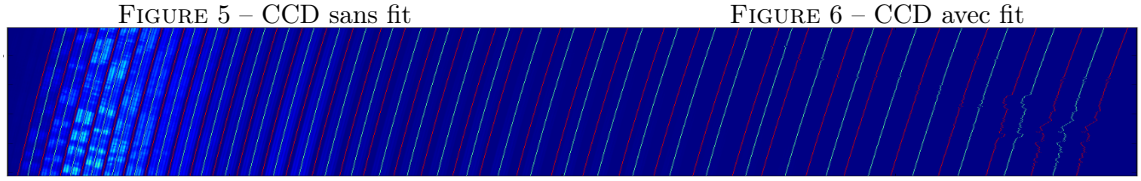

FIGURE 7 – bord haut sans fit

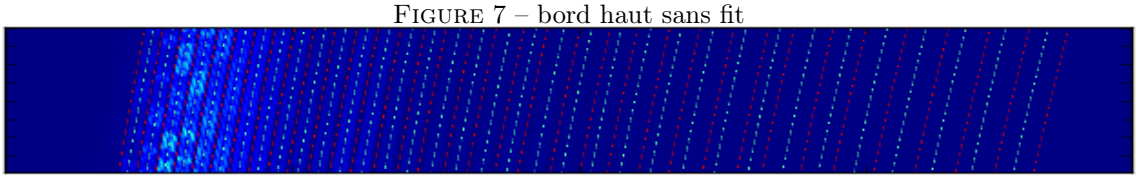

FIGURE 8 – bord haut après fit

Nous avons vu l'intérêt du fit, maintenant nous allons voir pourquoi nous utilisons ce fit. Pour ce faire, après plusieurs test pour plusieurs degrés de polynômes, on effectue pour chaque ordre la différence entre les valeurs calculées et les valeurs fittées pour chaque ligne. Ce calcul a été fait avec des polynômes d'ordre 2, 4, 6 et 10. La figure (9) représente le résultat de ce calcul pour l'ordre

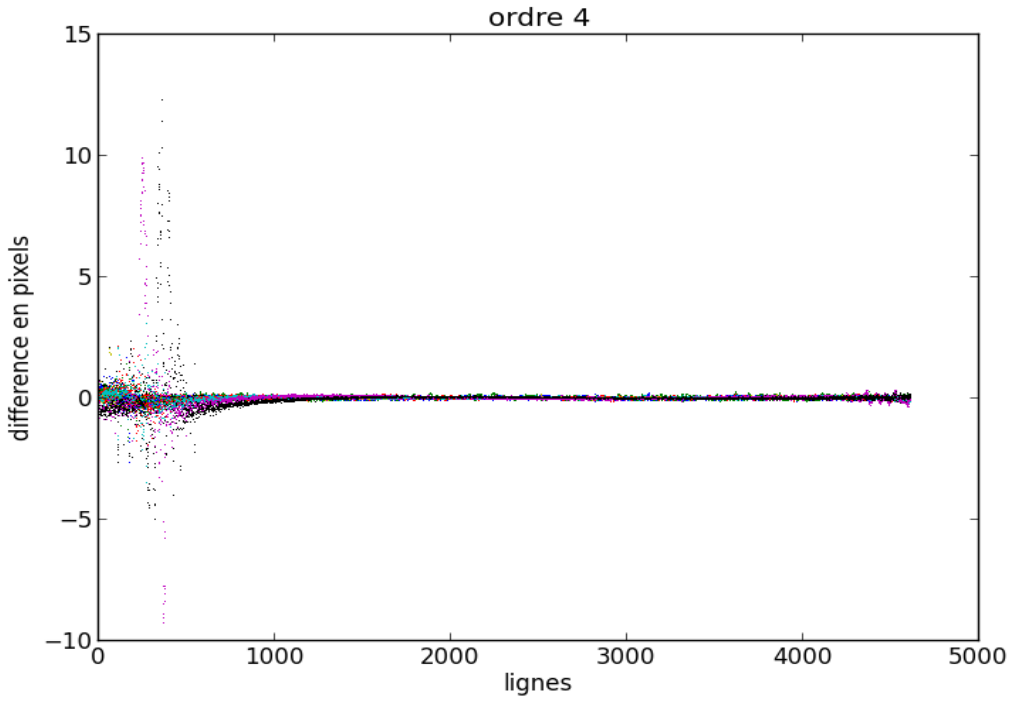

FIGURE 9 – différence pour l'ordre 4

4 uniquement. Les autres ordres donnent des résultats similaires pour les premières lignes. Cette forte amplitude est due à la présence du bruit pour les ordres peu lumineux.

En éliminant les premières lignes pour les 4 ordres cités précédemment, on obtient la figure (10). Sur cette dernière, on peux voir qu'à partir de l'ordre 6, une certaine structure commence à apparaître, et l'on remarque que cette structure est clairement visible pour l'ordre 10. Ceci est sûrement dû à une sur-évaluation du degré du polynôme de fit.

Enfin, en calculant l'erreur rms associée pour chaque ordre et pour chaque degré du polynôme, nous obtenons la figure (11). Sur cette dernière, nous remarquons que l'ordre 2 est trop peu précis et que les ordres 4, 6 et 10 ont une précision équivalente pour les ordres les plus lumineux. Cette précision est inférieure à $0,05$ pixels (estimée à 0.03 pixels). Elle se dégrade énormément pour les ordres peu lumineux encore une fois. Nous considérons donc que, pour tout les ordres, l'enveloppe fittée est précise à 0.03 pixels.

Ainsi, nous pouvons donner l'enveloppe des ordres à 0.03 en fittant grâce à un polynôme d'ordre 4.

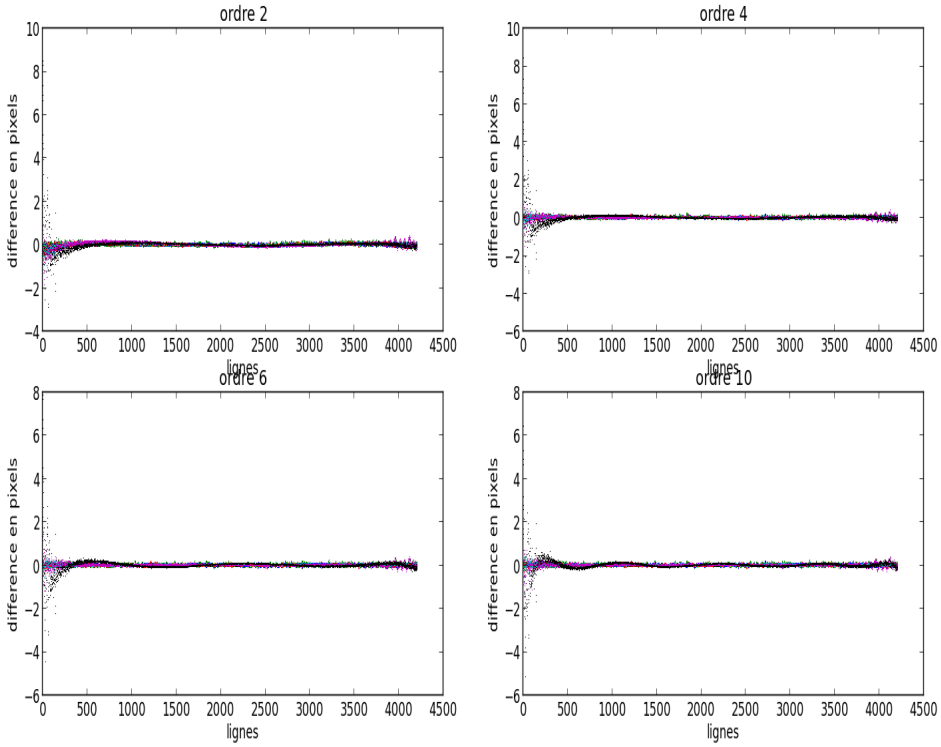

FIGURE 10 – différence en fonction des ordres du polynôme de fit

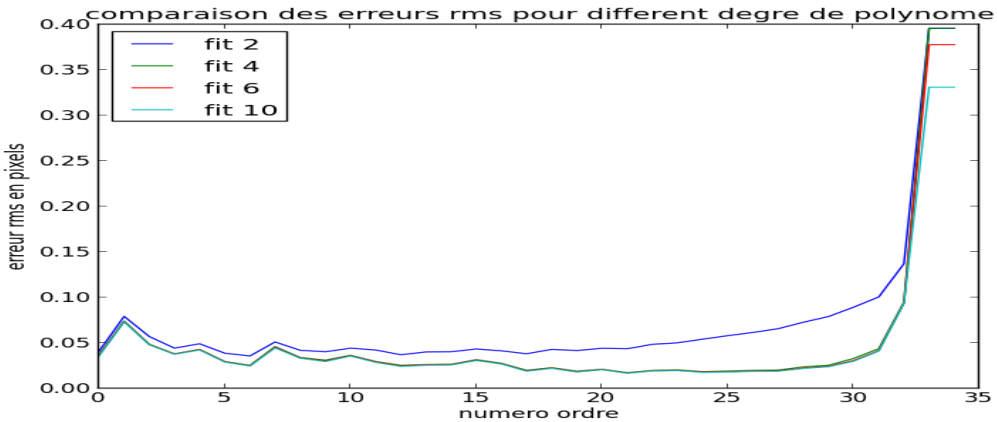

FIGURE 11 – précision

4 fonctions lanes

4.1 Utilisation

Il existe 2 versions de cette fonction : la version Lanes2 qui permet de récupérer l'inter-voie lorsqu'il n'y a que 2 voies pas ordres. La version Lanes3 qui permet de récupérer l'inter-voie lorsqu'il y a 3 voies par ordres. Ces deux fonctions renvoient une matrice dont les lignes correspondent au lignes du CCD et les colonnes contiennent les numéros de pixels correspondant aux différentes enveloppes. Elles prennent en argument le fichier contenant le CCD ainsi que l'enveloppe et l'épaisseur calculées précédemment.

4.2 Principe de fonctionnement

Tout d'abord, il nous faut récupérer la matrice calculée grâce à la fonction précédente. De plus, il nous faut également récupérer le tableau que renvoie la fonction Ref_localisation. Cette dernière donnée permet d'avoir directement la position des ordres sur la ligne centrale au pixel prêt car ces derniers vont servir d'indice de recherche dans des tableaux dans la suite. Tout l'algorithme va de nouveau travailler sur la ligne centrale.

La subtilité de l'algorithme est qu'il doit pallier le manque de précision de la fonction Ref_localisation. En effet, cette fonction détermine où se situe l'inter-ordre pour la ligne centrale. Le problème est que la largeur de l'inter-ordre augmente pour les derniers ordres. De plus, comme dit précédemment, le pixel correspondant à l'inter-ordre est plus au moins proche des différents ordres.

Pour pallier à ce problème, il suffit de se prendre une marge de sécurité en intensité. Comme nous avons déjà l'enveloppe contenant chacun des ordres, nous prenons au sein de cette enveloppe le tiers de la différence entre le maximum et le minimum au sein de l'enveloppe (au niveau de la ligne centrale toujours). Ainsi, lorsque l'intensité dépasse la valeur minimum + cette marge de sécurité, l'algorithme considère que l'ordre commence ici. Par exemple, sur la figure(12), la fonction Ref_localisation renvoie l'enveloppe [10 :80], les fonctions Lanes vont chercher dans [28 :53].

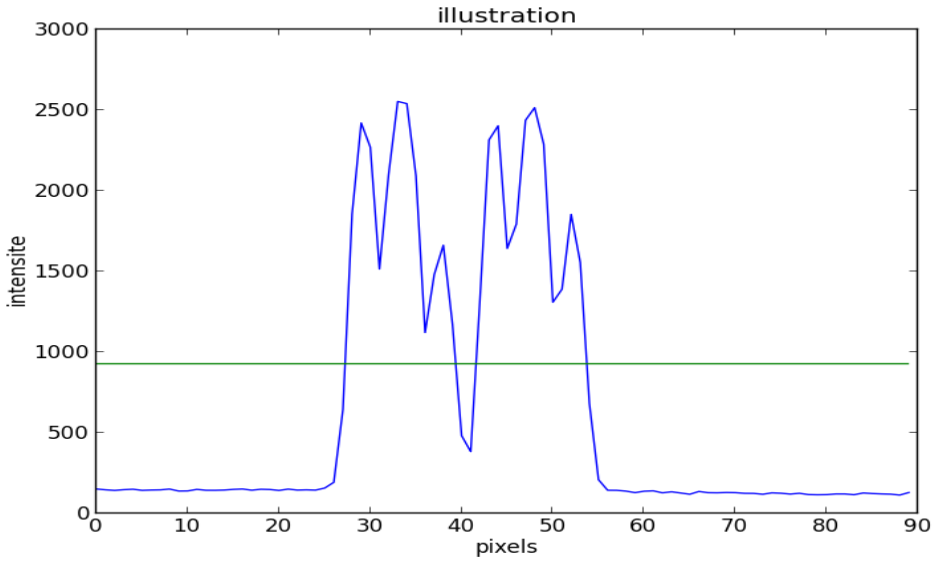

FIGURE 12 – illustration de la fonction lanes

Ainsi, une fois que l'on a délimité précisément notre ordre, la fonction Lanes2 va elle faire un recherche de minimum dans cet ordre. Ce minimum correspond ainsi à l'inter-voie pour cette ordre sur la ligne centrale. Il sera stocké dans la nouvelle matrice entre le minimum à gauche et à droite trouvé par la fonction précédente.

La fonction Lanes3 fait également une recherche de minimum dans l'ordre précisément délimité. Ensuite, elle vérifie si ce minimum est celui de l'inter-voie de gauche ou l'inter-voie de droite. Elle

effectue ensuite une nouvelle recherche de minimum dans le nouvelle intervalle.

Une fois que les inter-voies ont été retrouvées pour la ligne centrale, il suffit de les décaler pour les lignes suivantes. Comme nous avons déjà les enveloppes pour chaque ordre, nous avons accès à la courbure. Il nous suffit de prendre le décalage entre l'enveloppe sur la ligne centrale et la ligne actuelle, et d'appliquer ce décalage à l'inter-voie que l'on a trouvé sur la ligne centrale.

Ainsi, nous avons localisé pour chaque ordre l'enveloppe de chacune des voies.

5 Section Run subroutine

Cette section se focalise sur toutes les fonctions annexes du code qui vont permettent de faire fonctionner l'algorithme de façon autonome. Il y à sept fonctions annexes. Les quatre premières gèrent la création et l'utilisation des pickles, les deux suivantes s'occupent des fichiers fits et différent chemins et la dernière est celle qu'il faut appeler pour lancer l'algorithme.

5.1 fonction Run_envelope

Cette fonction permet de lancer la fonction Order_localisation et de gérer les pickles qui sont créés. Elle prend en argument le fichier fits que l'on traite (donc un fichier 10f), le nom (ou le chemin) pour le pickle contenant l'enveloppe et le nom (ou le chemin) pour le pickle contenant l'épaisseur des enveloppes. Elle prend également en argument un entier qui par défaut est égal à 2 et qui correspond au nombre de voies par ordre.

5.2 fonction Open_envelope

Cette fonction permet d'ouvrir les pickles créés par la fonction précédente. Elle prend donc en argument les chemins des pickles contenant l'enveloppe et l'épaisseur des ordres et elle renvoie leur contenu, à savoir une matrice contenant l'enveloppe des ordres et un tableau contenant l'épaisseur des ordres.

5.3 fonction Run_lane

Cette fonction permet de lancer la fonction lanes adéquate et de gérer les pickles qui sont créés. Elle prend en argument le fits sur lequel on travaille (le fichier 10f donc), le chemin du pickle contenant l'enveloppe des voies et le chemin pour le pickle contenant les épaisseurs des voies. Elle prend également en argument les chemins des pickles contenant l'épaisseur des ordres et l'enveloppe des ordres. Elle prend également un nombre entier par défaut égal à 2 correspondant aux nombres de voies par ordres. Si ce nombre est égal à 2, elle lance la fonction lanes2. Si ce nombre est égal à 3, elle lance la fonction lanes3.

5.4 fonction Open_lane

Cette fonction permet d'ouvrier les pickles créés par la fonction précédente. Elle prend donc en argument les chemins des pickles contenant l'enveloppe et l'épaisseur des ordres et elle renvoie leur contenu, à savoir une matrice contenant l'enveloppe des voies et un tableau contenant l'épaisseur des voies.

5.5 fonction find_10flats

Cette fonction permet de trouver tous les fichiers 10f qui vont devoir être traités. Elle prend en argument le chemin du dossier contenant les fichiers 10f. Pour se faire, elle stocke dans un tableau le chemin des différents fichiers 10f. Elle les reconnaît par la terminaison de la forme "10f.fts". Cette fonction renvoie ainsi ce tableau.

5.6 fonction find_path

Cette fonction permet de trouver le chemin du dossier que l'on souhaite, sachant que ce dernier est stocké dans un fichier ".txt". Elle prend donc en argument le chemin du fichier contenant tous les chemins et le nom du dossier. En partant du principe que dans le fichier de chemin, les informations sont stockées sous la forme "Nom : lechemin". La fonction renvoie le chemin lorsqu'elle le trouve et renvoie "not found" sinon.

5.7 fonction Run

Cette fonction est la fonction qui fait tourner l'algorithme. Elle prend en argument un entier qui vaut 2 par défaut correspondant au nombre de voies par ordres.

Tout d'abord, en connaissant le chemin du fichier txt contenant les chemins (que l'on est obligé de coder en dur), elle récupère grâce à la fonction précédente le chemin du fichier "FILES" contenant les fichiers 10f.

Ensuite, tant que le tableau contenant les fichiers 10f n'est pas vide, elle va lancer les fonctions Run_envelope et Run_lanes. Les noms des différents pickles sont normalisés et sont stockés dans le dossier "TEMP". De plus, dès qu'un fichier 10f est traité on le supprime de la liste de départ.