Rapport de Projet :
Validation et restitution d'orbite

Delphine Roubinet - Etienne Bresciani - Nicolas Bonneel
4ème année GMM, Insa Toulouse

17 mai 2005

Table des matières

Résumé

La validation et la restitution d'une orbite de transfert géostationnaire (GTO) de microsatellites par la seule utilisation des images acquises par théodolite permettrait d'éviter l'usage de moyens complexes et coûteux tels que le radar ou le système GPS.

Nous proposons un algorithme permettant à partir de n'importe quel téléscope amateur de 30cm de diamètre de valider les coordonnées d'un satellite en basse altitude sur une orbite fortement elliptique à partir d'une image et d'un catalogue d'étoiles. Le problème essentiellement traité ici étant alors la reconnaissance d'étoiles afin d'en déduire le positionnement du satellite. Notre méthode permet de déterminer la position du satellite à moins d'un millidegré près.

Les mesures obtenues dans la problématique précédente permet la restitution de l'orbite du satellite avec des erreurs de 4E-6 sur le paramètre d'excentricité de l'ellipse et de 300m environ sur le demi grand axe de l'ellipse dans le cas 2D.

1 Présentation de la problématique

La première partie de ce projet consiste en la validation d'une orbitographie déterminée par radar. Nous devons vérifier l'orbite d'un satellite (c'est à dire la position du satellite en un temps donné) à l'aide uniquement des images acquises par théodolite. Dans une seconde partie, le but est de montrer que l'on peut obtenir une orbitographie autonome par la méthode déterminée précédemment.

Cette méthode de validation et à terme de restitution d'orbite permettrait de remplacer les méthodes actuelles comportant différents problèmes. Le système de détermination par radar est très complexe et des moyens importants doivent être mis en oeuvre (quantité d'énergie d'environ 4kW et antennes de plus de 60 mètres). Et le système GPS nécessitant d'être implémenté à bord du satellite n'est pas utilisable pour des microsatellites.

L'évaluation de la position du satellite se fait à partir d'une image où le temps de prise implique que le satellite sera vu comme un segment. Il faut alors déterminer à quelle partie du ciel correspond notre image ; nous savons d'abord vers où pointe le télescope, de sorte que la zone à étudier est énormément réduite. Ensuite en utilisant le catalogue Hipparcos, on détermine la zone de manière précise en comparant les différentes configurations d'étoiles ; on obtient ainsi les coordonnées des étoiles autour du satellite et on peut donc déterminer la position du satellite.

Dans un second temps, on détermine les paramètres de l'orbite en utilisant les mesures obtenues par la méthode de la première partie. Cette méthode nous permet donc d'obtenir une restitution d'orbite à partir d'images de théodolite ; l'erreur d'orbitographie portant sur la restitution du demi grand axe de l'ellipse est typiquement d'environ 300 mètres.

Une étape de simulation est nécessaire afin de tester les algorithmes et de pouvoir déterminer les erreurs commises.

Cette manière de procéder nécessite la mise au point de méthodes de reconnaissance d'un segment et d'étoiles sur une image par un système de boite englobante, de méthodes de comparaison d'images d'étoiles par corrélation et de détermination des coordonnées d'un satellite à partir des coordonnées des étoiles qui l'entourent.

2 Simulation de l'image

2.1 Modélisation de l'image

Nous avons essayé de simuler au mieux l'image acquise par le téléscope, image numérisée en niveaux de gris (supposés échelonnés de 0 à 255) stockée sous forme matricielle.

Le téléscope nous fournit une image carrée de côté un degré, correspondant donc à un degré carré sur la sphère céleste. Le repérage sur la sphère céleste se fait en coordonnées célestes, c'est à dire en Ascension-Droite / Déclinaison. Pour des raisons de simplification de simulation, nous avons assimilé le repérage en coordonnées célestes à un repérage plan en coordonées cartésiennes, l'ascension-droite correspondant aux abscisses et la déclinaison aux ordonnées. Ceci n'enlève rien à la validité des algorithmes développés par la suite (qu'il suffira d'adapter au cas réel).

Sur Terre, pendant le temps d'intégration de l'image (quelques secondes) le téléscope se déplace à la vitesse de rotation de la Terre dans le sens inverse de cette rotation, ce qui permet de viser la sphère céleste de façon immobile. De cette manière il n'y a que le satellite qui se déplace pendant le temps d'intégration de l'image, ce qui laissera une trainée blanche sur l'image, assimilable à une portion de segment.

Le remplissage de l'image se fera alors par des objets de trois natures :
- simulation des étoiles correspondant au fond céleste
- simulation d'un segment
- simulation de bruits générés par le téléscope lors de l'acquisition.

2.2 Prise des données

Nous avons tout d'abord créé une base de données des étoiles existantes dans le ciel céleste à partir du catalogue Hipparcos récupéré sur internet. Les informations nous intéressant sont les coordonnées

(ascension-droite et déclinaison) et la magnitude des étoiles.

Dans ce catalogue répertoriant la totalité de la sphère céleste, on détermine une première zone qui correspond à la zone dans laquelle pointe le télescope. Cette zone qui fait quelques degrés de côté (typiquement un carré de 2 degré de côté) est dimensionnée par les erreurs de pointage éventuelles du télescope et par la longueur du segment satellite.

Dans cette zone nous déterminons aléatoirement un champ de vision de un degré carré tourné d'un angle aléatoire :

- choix aléatoire du centre de l'image, d'ascension droite comprise entre 0 et 360 degrés et de déclinaison comprise entre -90 et +90 degrés
- détermination des 4 coins caractéristiques de l'image à partir des coordonnées du centre ± 0.5 degré auxquels on applique une rotation d'angle θ, θ étant choisi aléatoirement entre 0 et $\frac{\pi}{2}$.

Puis nous extrayons les étoiles contenues dans ce champ de vision, en testant 4 produits scalaires. En effet, soient le carré ABCD correspondant au champ de vision et E une étoile, alors :

$$E \in ABCD \Leftrightarrow \begin{cases} \vec{AB}.\vec{AE} \geq 0 \\ \vec{BC}.\vec{BE} \geq 0 \\ \vec{CD}.\vec{CE} \geq 0 \\ \vec{DA}.\vec{DE} \geq 0 \end{cases}$$

On récupère ainsi les coordonnées et la magnitude des étoiles contenues dans ce champ de vision, il reste ensuite à effectuer un changement de repère afin d'obtenir les coordonnées des étoiles dans un repère basé sur le coin haut-gauche du champ de vision, ce qui nous fournit les bonnes coordonnées pour placer les éléments dans la matrice. Cela revient à effectuer :

- une translation d'un vecteur formé par le coin haut-gauche et l'origine $(0, 0)$, et
- une rotation d'angle $-\theta$.

2.3 Création de l'image

2.3.1 Simulation des étoiles

Bien qu'étant ponctuelle, une étoile nous apparaît comme une tache lumineuse, ce qui est dû aux altérations des rayons lumineux par l'atmosphère terrestre. Chaque étoile sera donc modélisée par une gaussienne en deux dimensions, dont le centre aura pour valeur sa magnitude M et pour coordonnées les coordonnées expliquées ci-dessus. Il reste à calculer l'intensité pixellique I sur l'image correspondant à la magnitude M. On suppose connaître le couple $(M0, I0)$ caractéristique du télescope donnant l'intensité pixellique $I0$ correspondant à une magnitude de référence $M0$. On a alors la formule :

$$I = I0 * exp\frac{M0 - M}{2.5} log(10)$$

et l'intensité de l'étoile autour de son centre est :

$$intensite(x, y) = I * exp - \frac{1}{2}\frac{(x^2 + y^2)}{\sigma^2}.$$

La taille de la gaussienne est déterminable en fonction de l'écart-type σ choisi, sachant que la gaussienne s'évanouie vite après quelques σ. Dans notre programme nous avons simulé des gaussiennes carrées de 5 ou 6 pixels de côté. L'écart-type σ est choisi en conséquences.

2.3.2 Simulation du segment satellite

Vu de la Terre, un satellite ressemble beaucoup à une étoile. Le segment sera donc modélisé par une trace de gaussienne (la même que pour les étoiles). Le tracé du segment est effectué de la sorte : on choisit aléatoirement deux points correspondant aux extrémités du segment, puis on utilise l'algorythme de Bresenham, qui trace une ligne entre les deux points, à la différence près qu'à chaque fois que l'on doit tracer un point, on dessine une gaussienne de centre ce point.
Grossièrement l'algorythme de Bresenham fonctionne de la façon suivante : soient $P1(x1, y1)$ et $P2(x2, y2)$ les deux extrémités du segment, on pose : $\Delta_x = |x2 - x1|$ et $\Delta_y = |y2 - y1|$. On part du point $P1$. Tant que $Delta_x \geq Delta_y$ on trace un pixel horizontalement, sinon verticalement. On se décale en conséquences et on recommence jusqu'à atteindre le point $P2$.

2.3.3 Suréchantillonnage

L'image acquise par le téléscope sera stockée dans une matrice de taille 1000*1000. Dans cette image, les centres des étoiles ne sont pas forcément centrés sur un pixel (les étoiles ne paraissent pas parfaitement symétriques). Pour représenter ce phénomène on utilise un procédé de suréchantillonage : la simulation se fera tout d'abord sur une matrice de taille 5000*5000, que l'on ramènera ensuite à la taille normale en moyennant sur des pavés de 5*5 pixels. Il en est de même pour le segment. Voici les résultats obtenus lors de la simulation d'une étoile :

- Fig.1 : exemple d'étoile parfaitement centrée sur un pixel
- Fig.2 : exemple d'étoile centrée sur le côté d'un pixel

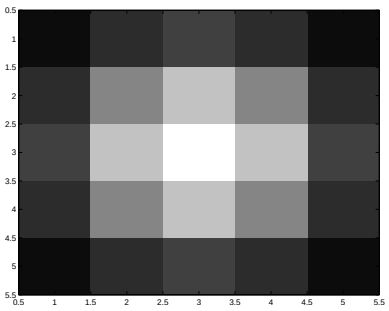


Fig.1

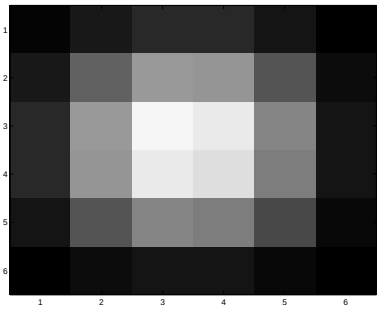


Fig.2

2.3.4 Simulation du bruit

Une fois que l'image 5000*5000 contient les étoiles et le segment, on procède donc à la réduction de l'image à sa taille normale 1000*1000 pour rajouter le bruit car celui-ci est créé par le téléscope, donc sur l'image réellement acquise. Le bruit engendré par le téléscope est de deux natures : additif (chaque pixel se voit rajouter une valeur) et multiplicatif (chaque pixel est multiplié par une valeur). Nous nous contentons de simuler le bruit additif, dont l'intensité suit une distribution gaussienne. On prend une gaussienne de moyenne nulle (l'image réelle pourrait facilement être traitée pour que cela soit le cas), c'est donc la variance σ qui déterminera l'intensité du bruit : plus σ est grand plus l'intensité est importante car la densité de probabilité f s'écrit :

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp \frac{1}{2} \frac{x^2}{\sigma^2} ////s : valeurdel'intensitedubruit$$

.

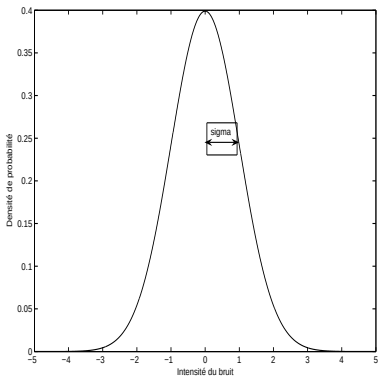
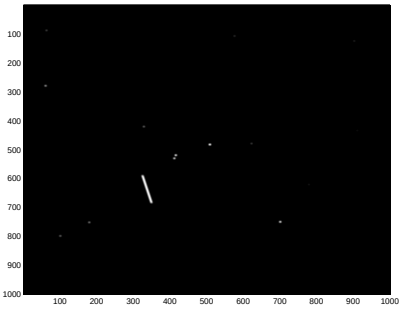


FIG. 1 – Importance du bruit selon sigma

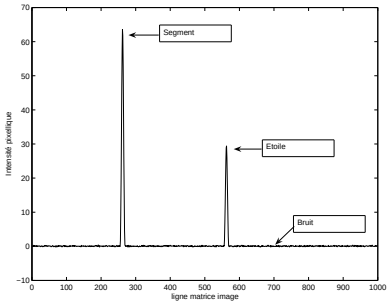
Le σ choisi est fonction des caractéristiques du téléscope. Supposons que le téléscope nous fournisse une valeur d'intensité $I0$ pour une valeur de magnitude 10, alors on supposera que $\sigma = \frac{1}{10} I0$. (Mais cela reste à préciser par le constructeur du téléscope.) Le bruit est alors simulé de la façon suivante : à chaque pixel on rajoute une valeur tirée aléatoirement suivant cette distribution.

2.4 Simulation effective

Nous avons ainsi simulé des images théodolites prises à plusieurs endroits de l'univers, afin de pouvoir tester les algorithmes suivants dans différentes situations, c'est à dire avec un nombre d'étoiles plus ou moins grand (on voit beaucoup plus d'étoiles au niveau de la voie lactée par exemple). En voici un exemple, ainsi qu'une coupe transversale représentant l'intensité pixellique suivant une ligne de la matrice image :



Exemple d'image simulée



Coupe transversale

2.5 Paramètres

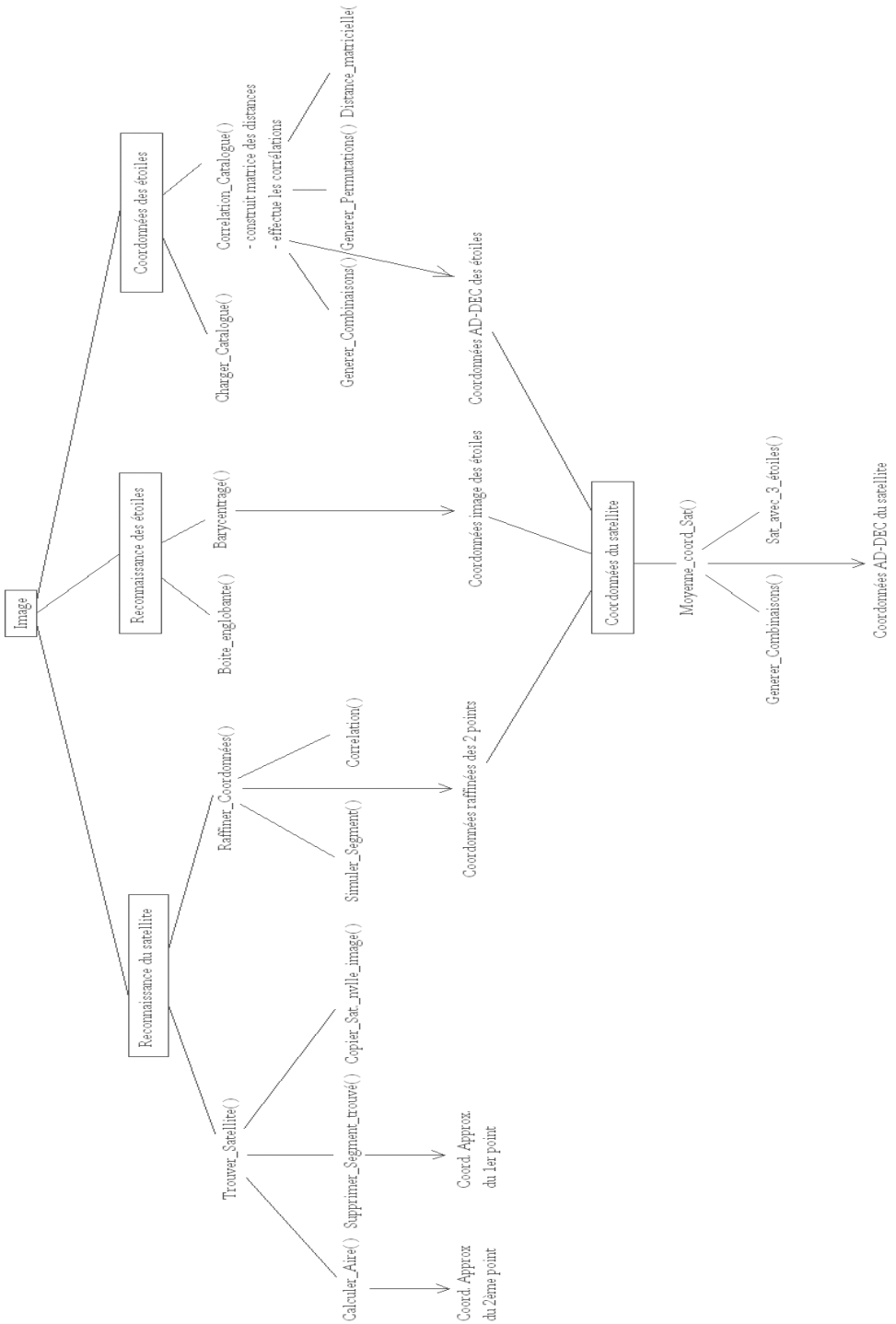
Voici un récapitulatif des paramètres intégrés dans la simulation de l'image :

- image correspondant à un champ de vision de un degré carré codée dans une matrice 1000*1000 (un pixel correspond donc à un millième de degré)
- précision de la zone de pointage du télescope : typiquement un carré de 2 degrés de côté
- nombre d'étoiles dans le champ de vision d'un degré carré : dépend de la zone du ciel pointée, typiquement entre 3 et 10 étoiles
- $(M0, I0)$ le couple (Magnitude,Intensité pixellique) caractéristique du télescope, typiquement nous avons pris $M0 = 10$ et $I0 = 50$
- rapport signal à bruit : $RSB = \sigma_{bruit} = \frac{1}{10} I0$

Remarque : tous les paramètres de simulation sont modifiables afin de se rapprocher le plus possible de la réalité.

3 Reconnaissance

Nous essayons maintenant de retrouver le satellite dans notre image simulée. Voici le diagramme des appels aux fonctions :



3.1 Segment

Il s'agit de retrouver les coordonnées dans l'image des deux extrémités du segment composant la trajectoire du satellite le plus précisément possible.
En effet, ce sont ces données dont les erreurs influeront le plus sur les coordonnées réelles du satellite. On cherchera donc à obtenir une erreure inférieure à 1/5 de pixel sur la restitution de ces coordonnées (soit

1/5000ème de degré).

3.1.1 Détection approximative

Dans un premier temps, on cherche les coordonnées (i,j) approximatives du satellite. Pour cela on parcourt les pixels de l'image jusqu'à dépassement du seuil de bruit. Si ce seuil est dépassé, on calcule l'aire de la zone qui dépasse le seuil. Si cette aire est sufisamment grande, on aura à faire au satellite et on retiendra les coordonnées minimales et maximales du satellite (= les deux extremités). Dans le cas contraire, il s'agira simplement d'une étoile ou d'un bruit ponctuel important.
De plus, le segment doit être supprimé de l'image.

Algorithme :

```
Trouver_Satellite()
  Pour chaque ligne j de l'image faire
    Pour chaque colonne i de l'image faire
      aire = 0
      Si  pixel(i,j) > seuil_bruit  alors
        aire = Calculer_Aire(i,j)
      Fin Si
      Si aire > aire_minimale alors
        Supprimer_segment_trouvé()
        retourner i et j
// première extremité trouvée approximativement
      Fin Si
    Fin Pour i
  Fin Pour j
Fin Trouver_Satellite()
```

Par commodité de programmation, c'est la fonction calculant l'aire qui retournera les coordonnées de la seconde extremité. De plus cette fonction a été codée récursivement :

```
Calculer_Aire(i,j)
  aire = 0

// on parcourt la ligne j vers les i decroissants
  k = 0
  Tant que pixel(i-k, j) > seuil_bruit
   incrementer(aire)
   incrementer(k)
  Fin Tant que

// on parcourt la ligne j vers les i croissants
  l = 0
  Tant que pixel(i+l, j) > seuil_bruit
   incrementer(aire)
   incrementer(l)
  Fin Tant que

// on appelle recursivement avec i le milieu des extremités
// gauches et droites de la ligne courante, et j la ligne
// suivante
   Si pixel(i+(l-k)/2 , j+1) > seuil_bruit alors
    aire = aire + Calculer_Aire( i+(l-k)/2 , j+1)
   Sinon
```

```
    retourner i+1 et j    // deuxième extremité trouvée
   Fin Si

   retourner aire
Fin Calculer_Aire
```

La fonction "Supprimer_segment_trouvé" est la même que la fonction qui calcule l'aire du segment, à part que chaque pixel au dessus du seuil est fixé à zéro. Par ailleur, cette fonction copie le segment dans une toute nouvelle image ne contenant que lui pour la suite des opérations.

3.1.2 Raffinement des coordonnées

A ce point, nous connaissons approximativement les coordonnées des extremités du segment (à plus ou moins 2 ou 3 pixels...). On utilisera une méthode de maximum de corrélation entre ce segment, et un segment théorique dont on fera varier les extremités de l'ordre du cinquième de pixel (= précision souhaitée).

Pour des raisons de temps de calcul, on commencera par un maximum de correlation en faisant varier les extremités de 1 pixel avant de passer à une echelle inférieure de 1/5 de pixel.

Le segment théorique utilisé ici est la même fonction que la fonction qui nous a permis de tracer ce segment lors de la phase de simulation (segment avec PSF gaussienne) : Simuler_segment(point1_i, point1_j, point2_i, point2_j). Il sera ensuite rétréci d'un facteur 5 de la même manière que l'image du théodolite a été construite.

Le segment étant "suffisament grand" nous connaissons sa pente assez précisément. On se permettra donc de n'effectuer cette correlation qu'avec un morceau de ce segment situé à chaque extrémité de celui-ci. Cela nous permettra de réduire le temps de calcul puisqu'il n'y aura que deux paramètres à correler (les coordonnées d'une extremité) deux fois, la pente étant connue, au lieu de quatre paramètres (les coordonnées des deux extremités) si la pente avait été considérée inconnue.

Nous considererons que nous connaissons les coordonnées du segment à 3 pixels près par exemple. On fera attention à la direction du segment qui selon comment est calculé la pente pourrait être inversée, ainsi qu'à décaler d'un demi "gros" pixel (3 "petits" pixels) le résultat à cause du centrage du segment théorique sur un "gros" pixel.

Algorithme pour raffiner les coordonnées du premier point

```
Raffiner_coordonnées(point1_i,point1_j, point2_i, point2_j)
 pente = Calculer_pente_segment(point1_i,point1_j,point2_i,point2_j)
 largeur = 300
 hauteur = 300
 image_theorique = Nouvelle_image(largeur, hauteur)

 maximum_correlation = 0
 Pour k de -3 à 3 faire
  Pour l de -3 à 3 faire
   centre_i = largeur/2 + 5*k
   centre_j = hauteur/2 + 5*l

   Si pente <= 0 alors
    Simuler_segment(centre_i,centre_j,centre_i+1000,centre_j+1000*pente)
   Sinon
    Simuler_segment(centre_i,centre_j,centre_i-1000,centre_j+1000*pente)
   Fin Si

   Reduire_image_facteur_5(image_theorique)
```

```
   correlation = Correlation(image_theodolite, image_theorique)
   Si correlation > maximum_correlation alors
    maximum_correlation = correlation
    k_raffiné = k
       l_raffiné = l
     Fin Si
   Fin Pour l
  Fin Pour k


 maximum_correlation_2 = maximum_correlation
 Pour k de -5 à 5 faire
  Pour l de -5 à 5 faire
   centre_i = largeur/2 + 5*k_raffiné + k
   centre_j = hauteur/2 + 5*l_raffiné + l

   Si pente <= 0 alors
    Simuler_segment(centre_i,centre_j,centre_i+1000,centre_j+1000*pente)
   Sinon
    Simuler_segment(centre_i,centre_j,centre_i-1000,centre_j+1000*pente)
   Fin Si

   Reduire_image_facteur_5(image_theorique)

   correlation = Correlation(image_theodolite, image_theorique)
   Si correlation > maximum_correlation_2
    maximum_correlation_2 = correlation
    k_raffiné_2 = k
    l_raffiné_2 = l
   Fin Si
  Fin Pour l
 Fin Pour k

retourner 5*k_raffiné+k_raffiné_2+3 et 5*l_raffiné+l_raffiné_2+3

Fin Raffiner_coordonnées
```

La fonction Correlation(f, g) calcule le coefficient de correlation entre deux images f et g par la formule :

$$\frac{\sum_{i=1}^{n} f_i . g_i}{\sqrt{\sum_{i=1}^{n} f_i^2 . \sum_{i=1}^{n} g_i^2}}$$

On cherchera donc à avoir un coefficient de correlation proche de 1 (donc le plus grand possible puisqu'il est forcément inférieur ou égal à 1).

On effectue une procédure similaire pour le second point (le sens du segment theorique est juste inversé).

On obtient ainsi les coordonnées (i,j) du satellite au cinquième de pixel près.

3.2 Etoiles

3.2.1 Boite englobante

Pour effectuer la reconnaissance d'étoiles sur une image, nous déterminons pour chaque étoile la boite qui la contient ; c'est à dire la position du pixel en haut à gauche et celle du pixel en bas à droite du rectangle entourant l'étoile.

Afin de différencier les étoiles du bruit présent sur l'image, un seuil est choisi (ce seuil est un paramètre modifiable fixé à 20 dans les simulations qui suivent) tel que tout pixel d'intensité inférieure à ce seuil est considéré comme du bruit et donc les valeurs supérieures correspondent à des étoiles.

Ainsi le sous-programme rep_etoile renvoie à partir de la matrice des pixels (appelée image) le tableau coord des coordonnées caractéristiques des boites déterminées et le nombre d'étoiles repérées. Pour cela, on balaie la matrice image jusqu'à ce que l'on trouve une valeur supérieure au seuil ; ce point sera le premier point de la première ligne de la boite à déterminer. En balayant la première ligne, on détermine les valeurs extrêmes à gauche et à droite de la boite ; on effectue ensuite un balayage des lignes jusqu'à que l'on soit sur une ligne où il n'y a plus une seule valeur supérieure au seuil, on détermine ainsi les valeurs extrêmes en haut et en bas de la boite. Pour chaque nouvelle ligne étudiée, il faut bien vérifier que les valeurs extrêmes gauche droite de la boite ne sont pas à modifier par un balayage des colonnes (la valeur de gauche peut être diminuer d'un ou plusieurs pixels et celle de droite augmentée).

Le cas particulier d'une boite d'un pixel sera considéré comme du bruit.

3.2.2 Barycentrage

Pour déterminer les coordonnées en pixels des étoiles on effectue un barycentrage sur chaque boite englobante. Il nous suffit de sommer les intensités des pixels pondérées par les coordonnées en ligne et en colonne et de diviser chacune de ces deux sommes par la somme des intensités sur toute la boite.

Le sous-programme calc_bary nous renvoie ainsi à partir des tableaux de pixels image et de coordonnées des boites englobantes coord le tableau bary des coordonnées des barycentres de chaque boite.

3.3 Restitution des coordonnées réélles des étoiles

Nous connaissons les coordonnées (i,j) des étoiles. Par ailleurs, nous avons en notre possession le catalogue Hipparcos contenant les coordonnées de 118 000 étoiles en ascension droite et déclinaison. Le but est d'associer à chaque étoile de l'image, une étoile du catalogue.

Nous procedons en plusieurs étapes :

(a) Sélection des étoiles du catalogue → Connaissant très approximativement la position du satellite (puisque le théodolite pointe le satellite !), nous n'avons pas besoin d'avoir recours à tout le catalogue Hipparcos. Nous sélectionnons donc les étoiles présentes dans une zone de 1 ou 2° (distance angulaire) autours de la position que le téléscope pointe.

(b) Construction de matrices de distances → Calcul des distances entre chaque étoile et formation d'une matrice $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. Nous formons ainsi deux matrices : l'une $D1$ contenant les distances entre chaque étoile de l'image dont on connait les coordonnées (i,j) en pixels, l'autre $D2$ contenant les distances entre chaque étoile du catalogue dont on connait les ascensions droites et déclinaisons. Ces matrices ayant des échelles différentes (l'une étant exprimée en pixels, l'autre en distance angulaire), nous les normalisons donc. Nous avons choisi la norme matricielle : $\|D\| = \sqrt{\sum_{i,j} d_{i,j}^2}$

(c) Combinaisons des étoiles → Nous possédons une image de téléscope contenant p étoiles, et nous connaissons un catalogue de n étoiles. Nous cherchons donc parmis tous les arrangements de p étoiles parmis n du catalogue celui rendant la distance matricielle $d(M,N) = \sqrt{\sum_{i,j}(m_{i,j} - n_{i,j})^2}$ minimale.

On génère donc toutes les combinaisons de p indices d'étoiles parmis n, puis toutes les permutations de ces indices.

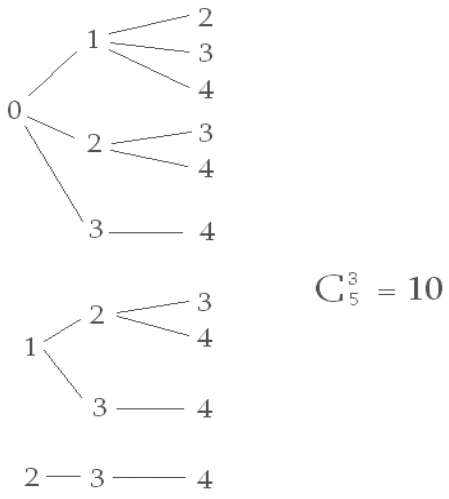
C'est l'opération la plus couteuse en calcul : il y a $C_n^p * p! = \frac{n!}{(n-p)!}$ calculs de distances matricielles (nous pourrons déjà minimiser $\sum_{i,j}(m_{i,j} - n_{i,j})^2$ au lieu de $\sqrt{\sum_{i,j}(m_{i,j} - n_{i,j})^2}$ pour éliminer le calcul couteux d'une racine carrée inutile) .

Remarque : Nous avons ici considéré qu'il y avait plus d'étoiles dans le catalogue que dans l'image. Il est très probable que cela soit l'inverse puisque le catalogue Hipparcos contient des étoiles de magnitude maximale de 12.4 et le théodolite utilisé sera plus puissant. Dans ce cas, on devra considérer aussi un certain nombre de combinaisons d'étoiles de l'image du théodolite Par exemple, si nous avons détecté 20

étoiles, il est possible qu'une dizaine soient des étoiles non répértoriées par Hipparcos : nous pourrons donc chercher toutes les combinaisons de 3 étoiles parmis 20, effectuer la procédure précédente pour obtenir une distance matricielle minimale parmis ces combinaisons, puis recommencer avec toutes les combinaisons de 4 étoiles parmis 20 etc... jusqu'à ce que la distance minimale soit trop grande. Par ailleurs, par souci du temps de calcul, il est inutile de connaitre la position de toutes les étoiles de l'image : nous pouvons nous restreindre à 5 ou 6 étoiles (quitte à perdre en précision), car le temps de calcul augmente beaucoup avec le nombre d'étoiles de l'image.

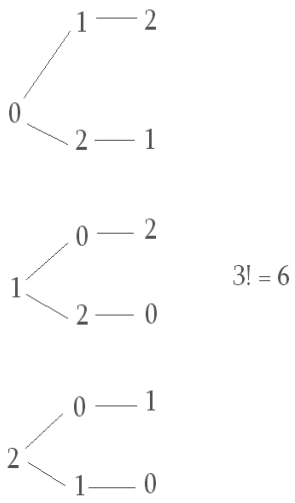
Les fonctions renvoyant les combinaisons et les arrangements étant les plus délicates à coder, en voici les schémas explicatifs et les algorithmes récursifs de parcours de l'arbre de tous les arrangements et combinaisons pour les stocker :

Arbre des combinaisons pour 3 éléments parmis 5 :



$$C_5^3 = 10$$

Arbre des arrangements de 3 éléments



3! = 6

```
// Retourne le nombre de combinaisons de p parmis n elements
// selon la formule de récurrence du triangle de Pascal
C(n,p)
  Si p = 0 alors
    retourner 1
  Fin Si
  Si n = 0 alors
    retourner 0
  Fin Si
  retourner C(n-1, p-1)+C(n-1,p)
Fin C


// variable globale
Numéro_combinaison_courante = 0

//Génère les n!/(p!(n-p)!) combinaisons de p parmis n entiers
//de 0 à n-1
Generer_Combinaisons(tableau_combinaisons[C(n,p),p],indice_début,
indice_fin,p,n)

  Si p = 0 alors
    incrémenter(Numéro_combinaison_courante)
  Sinon
    Pour j de indice_début à indice_fin-1
      Pour i de 0 à C(n-j-1, n-1)-1
        tableau_combinaisons[Numéro_combinaison_courante, p-1] = j
      Fin Pour
    Generer_Combinaisons(tableau_combinaisons,j+1,indice_fin+1,p-1,n)
    Fin Pour
  Fin Si
Fin Generer_Combinaisons
```

```
// variable globale
Numéro_permutation_courante = 0

// Génère les n! combinaisons de n entiers de 0 à n-1
Generer_Permutations(tableau_permutations[n!,n],n,numéro_courant)
 Booléen : Numéro_courant_déjà_présent

 Si indice_courant = n alors
  incrémenter(Numéro_permutation_courante)
 Sinon
  Pour i de 0 à n-1
   Numéro_courant_déjà_présent = Faux
   Pour j de 0 à numéro_courant
    Si tableau_permutations[Numéro_permutation_courante,
numéro_courant-j-1]=i
     Numéro_courant_déjà_présent = Vrai
    Fin Si
   Fin Pour
   Si Numéro_courant_déjà_présent = Faux
    Pour j de 0 à (n-numéro_courant-1)!
     tableau_permutations[Numéro_permutation_courante+j,
numéro_courant]=i
    Fin Pour
    Generer_Permutations(tableau_permutations,n,numéro_courant+1)
   Fin Si
  Fin Pour
 Fin Si
Fin Generer_Permutations
```

3.4 Restitution des coordonnées du satellite

Grâce aux étapes précédentes on connait les coordonnées des étoiles proches du satellite sous deux formes : d'abord leur position sur le tableau de pixels de l'image puis précisément en terme d'ascencion droite et déclinaison. De plus, on a déterminé la position du segment en pixel.

A partir de ces valeurs pour trois étoiles, on peut définir par interpolation une correspondance entre les coordonnées sur l'image et les coordonnées célestes. Nous recherchons donc une fonction f tel que z=f(x,y) avec z les coordonnées célestes et x et y celles sur l'image. Ceci revient à résoudre l'équation du plan suivante : z=ax+by+c pour trois étoiles dont toutes les coordonnées sont connues et dans deux cas différents, pour z étant l'ascension droite puis pour z étant la déclinaison. Lorsque les valeurs des variables a,b et c sont déterminées dans les deux cas, on obtient alors les coordonnées célestes du segment en fonction de ses coordonnées sur l'image. Pour cela on résoud deux fois le système suivant par la méthode de Kramer :

$$\begin{pmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} z_0 \\ z_1 \\ z_2 \end{pmatrix}$$

Ainsi, le sous-programme position_segment renvoie le tableau coord_sp_sat des coordonnées spatiales du satellite à partir des tableaux coord_sp, coord_ij et coord_ij_sat étant respectivement les coordonnées spatiales et de l'image des étoiles et les coordonnées du satellite sur l'image.

Le sous-programme moyenne_position_segment permet d'obtenir les coordonnées spatiales du satellite à partir de la position de plusieurs étoiles en appliquant la méthode précédente à toutes les combinaisons posssibles de trois étoiles et en faisant la moyenne des résultats obtenus afin que l'interpolation soit la plus réaliste possible.

4 Résultats obtenus et limites

4.1 Exemples de résultats

Nous effectuons des tests pour différentes zones du catalogue afin d'étudier les résultats obtenus pour différentes quantités d'étoiles à comparer.

1. Parmi 4 étoiles
   Pour la zone prédéterminée suivante :
   - déclinaison haut=60
   - déclinaison bas=58
   - ascencion droite 'gauche'=73
   - ascencion droite 'droite'=75

   4 étoiles sont présentes dans cette zone et dans l'image étudiée ; 3 étoiles sont détectées sur l'image et les erreurs sur la position du satellite (obtenues aux extrémités du segment) sont (-0.0304,0.0144) et (-0.0301,0.0114).

2. Parmi 9 étoiles
   Les paramètres du cadre de départ sont :
   - déclinaison haut=6
   - déclinaison bas=4
   - ascencion droite 'gauche'=73
   - ascencion droite 'droite'=75

   Dans ce cas, il y a 9 étoiles dans la zone prédéterminée et 7 étoiles sur l'image étudiée ; 6 étoiles ont été détectées sur cette image et les erreurs obtenues sur la position du satellite sont (0.0013,-0.0016) et (0.0024,-0.000323).

3. Parmi 12 étoiles
   Les paramètres du cadre de départ sont :
   - déclinaison haut=12
   - déclinaison bas=10
   - ascencion droite 'gauche'=73
   - ascencion droite 'droite'=75

   Dans ce cas, il y a 12 étoiles dans la zone prédéterminée et 9 étoiles sur l'image étudiée ; 8 étoiles ont été détectées sur cette image et les erreurs obtenues sur la position du satellite sont (0.000524,-0.0056) et (0.000903,-0.0026).

   Pour des cadres de taille identique on observe que plus le nombre d'étoiles dans l'image étudiée est important, plus la position du satellite est précise ; ce qui est logique avec la méthode de détermination

4. Parmi 16 étoiles
   Les paramètres du cadre de départ sont :
   - déclinaison haut=50
   - déclinaison bas=45
   - ascencion droite 'gauche'=73
   - ascencion droite 'droite'=75

   On a ici 16 étoiles dans la zone prédéterminée et 7 étoiles sur l'image étudiée ; 6 étoiles ont été détectées sur cette image et les erreurs obtenues sur la position du satellite sont (-0.0029,-0.0034) et (-0.000854,-0.0061).

   On observe ainsi que la précision de la position du satellite ne dépend pas de la taille du cadre mais seulement du nombre d'étoiles présentes sur l'image étudiée ; seul le temps de calcul pouurrait être augmenté (puisqu'il y a alors plus de combinaisons à étudier) mais la différence avec le cas 'Parmi 9 étoiles' n'est pas significative ici.

4.2 Erreurs calculées

Les erreurs produites durant le fonctionnement du programme concernent la reconnaissance des étoiles et du segment.

Le maximum d'erreur obtenu sur la détermination des centres d'étoiles en traitant tous les cas possibles est de un demi pixel sur l'image finale ; la moyenne des erreurs est de (0.11068 ;0.113778) et l'écart type de
(0.17787 ;0.175543).

Pour la recherche de la position du segment satellite, l'erreur est de 1/5 de pixel au maximum.

4.3 Limites et perspectives

Les limites de l'algorithme sont d'abord au niveau de la reconnaissance des étoiles. En effet, deux étoiles côte à côte ne seront pas distinguées l'une de l'autre ; c'est à dire que si les pixels les séparant sont inexistant (dans le cas d'étoiles jumelles) ou ont une valeur supérieure à celle du seuil, l'algorithme les considérera comme une seule étoile.

Le temps de calcul peut aussi être un problème car dans le cas où la zone étudiée comporte plus d'une vingtaine d'étoiles, le nombre de comparaison à faire étant important, le temps de calcul peut être de plusieurs heures.

Ceci peut être résolu en ne conservant que les étoiles les plus importantes de la carte et en adaptant alors la valeur du seuil pour ne reconnaître que les étoiles qui nous intéressent.

Quelques modifications à apporter en vue d'un programme plus opérationnel :

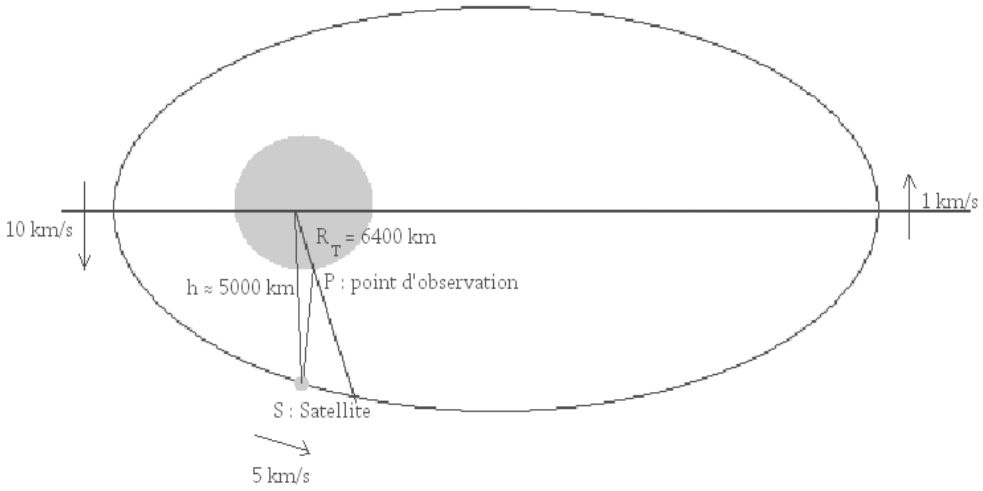
1. Une autre limite du programme est que le cas du passage de 360 à 0°C pour l'ascencion droite n'est pas traité ; ces cas particuliers seraient à prendre en compte dans le calcul de distances pour une image qui serait située dans cette zone limite.
2. Le même problème se pose pour la déclinaison au niveau des pôles célestes mais il ne paraît pas util de traiter ce cas puisque le télescope ne pointera pas vers ces zones.

3. Erreurs dûes à l'approximation plane : il faut noter que lorsqu'on se rapproche des pôles, un degré carré correspond à une surface plus grande qu'un degré par un degré en coordonnées célestes. Pour la simulation de l'image nous n'en n'avons pas tenu compte puisque nous avons assimilé les coordonnées célestes à des coordonnées cartésiennes. Ceci n'est pas sans conséquences : en effet, un degré en ascension-droite représente une distance réelle plus petite lorsqu'on se rapproche des pôles alors que l'image observée par le téléscope reste de la même taille (1 degré sphère $\leq$ degré observé). Le calcul de distances devrait donc se faire autrement.

   Les raisons ayant pourtant motivé ce choix sont :
   - Simulation beaucoup plus simple : un degré en coordonnées célestes sera toujours assimilé à un degré observé par le téléscope, ce qui rend beaucoup plus facile la sélections des étoiles à simuler. Un degré en coordonnées célestes sera "étalé" sur un degré carré. La simulation en sera par contre légèrement faussée, mais nos résultats sont bons puisque les distances dans le catalogue sont calculées en coordonnées célestes. Cela n'enlève donc rien à la validité des algorithmes.
   - Les erreurs sont faibles si l'on reste près de l'équateur céleste ; or c'est le cas des satellites observés.
   - Si l'on veut être cohérant dans le calcul des distances (ce qui se révélera util lors de l'acquisition de vraies images) il suffira de faire la correspondance dans le catalogue entre distances en coordonnées célestes et distances angulaires visuelles (comme utilisées pour faire pointer le télescope).

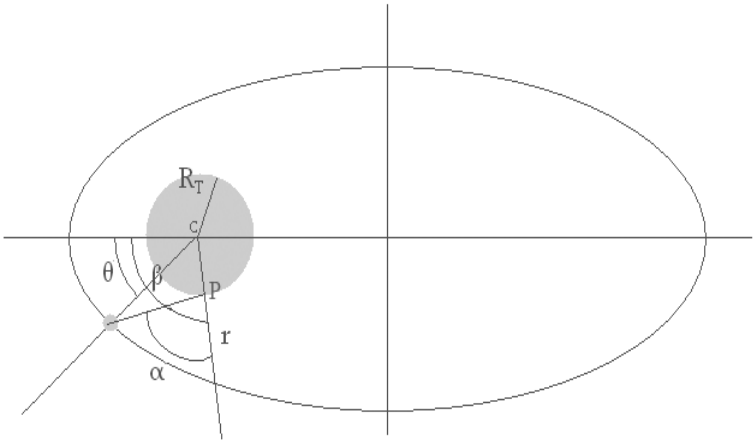
5 Orbitographie autonome

Nous cherchons à savoir si à partir de plusieurs positions du satellite calculé par l'algorithme précédent à partir des images de théodolites, on pouvait obtenir les paramètres orbitaux du satellite. Cela permettrait de réduire énormément les coûts pour prédire la position du satellite puisque cette opération est effectuée actuellement par d'énormes radiotéléscopes, par GPS ou avec un système DORIS qui sont deux systèmes embarqués donc onéreux. Nous pourrions à la place utiliser notre algorithme n'utilisant qu'un simple téléscope amateur de 30cm de diamètre, ce qui fournirait une solution peu couteuse pour détecter le satellite à basse altitude.
L'orbite de notre satellite étant une orbite GTO donc très fortement elliptique, les sytèmes actuels deviennent inopérants : lieu d'observation éclairé rendant les observations par les radiotéléscopes impossible de jour, la basse altitude du satellite et sa vitesse rendant inefficaces le GPS et le DORIS...



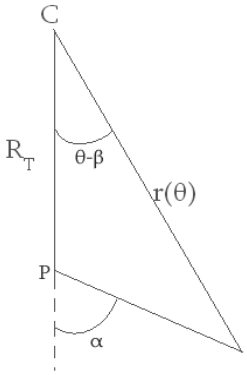
Nous nous sommes intéréssés à la faisabilité de notre méthode de détection du satellite pour prédire sa position, à savoir l'erreur commise sur les paramètres orbitaux en fonction de l'erreur obtenue par la méthode précédente de validation d'orbite sur la position du satellite.

Nous n'avons pu nous intéresser qu'au cas 2D, sans utiliser les équations de Kepler.
On connait l'équation de l'ellipse de paramètre (a, e, α) : $r(\theta) = \dfrac{a(1-e^2)}{1 + e * cos\ \theta}$, et α en fonction de θ



On a : $e = \frac{c}{a}$, $p = \frac{b^2}{a}$,
D'où : $r(\theta, r, e, p) = \frac{p}{1 + e * cos\ \theta}$.

Les relations trigonométriques dans le triangle :



nous donnent :

$$\frac{r(\theta)}{sin\ \alpha} = \frac{R_T}{sin(\alpha - \theta + \beta)}$$

d'où :

$$\frac{p}{(1 + e * cos\ \theta).sin\ \alpha} = \frac{R_T}{sin(\alpha + \beta).cos\ \theta - cos(\alpha + \beta).sin\ \theta)}$$

$$\Rightarrow cos\ \theta.(\frac{sin(\alpha + \beta)}{R_T} - \frac{e.sin\ \alpha}{p}) = sin\ \theta.(\frac{cos(\alpha + \beta)}{R_T}) + \frac{sin\ \alpha}{p}$$

que l'on résoudra itérativement par Newton pour α fixé afin d'obtenir θ

On utilise un développement limité pour linéariser l'équation de l'ellipse :

$$F(X + dX) = F(X) + X.\nabla F(X) + \mathcal{O}(\|X\|)$$

où X est le vecteur (a, e, α) et où F est la fonction renvoyant $(r(\theta), \theta$.
En considérant que l'erreur commise sur θ et r provient uniquement de l'erreur sur α, l'erreur sur r et sur θ est donc donnée par l'expression :

$$(dr \quad d\theta)^T = \nabla F.(da \quad de)$$

soit :

$$Y = A.\bar{X}$$

Nous cherchons les erreurs sur a et e connaissant celles sur r et θ, et nous disposons de plusieurs mesures de ces erreurs (une dizaine de mesures espacées de 150 km environ). On utilise donc Newton pour résoudre ce problème des moindres carrés et on obtient :

$$\bar{X} = ({}^tA.A)^{-1}.{}^tA.Y$$

avec :

$$A = \begin{pmatrix} \begin{pmatrix} \frac{dr}{da} & \frac{d\theta}{da} \\ \frac{dr}{de} & \frac{d\theta}{de} \end{pmatrix}_i \\ \vdots \\ \vdots \end{pmatrix}$$

$$Y = \begin{pmatrix} \begin{pmatrix} dr \\ d\theta \end{pmatrix}_i \\ \vdots \\ \vdots \end{pmatrix}$$

$$\bar{X} = \begin{pmatrix} da \\ de \end{pmatrix}$$

et $dr = \frac{\partial r}{\partial \alpha}.d\alpha$,
$d\theta = \frac{\partial \theta}{\partial \alpha}.d\alpha$

α étant connu : c'est l'erreur commise lors de la detection du satellite sur sa position.

On obtient les dérivées en les discrétisant.

Conclusion

Nous avons ainsi pu obtenir une méthode de validation d'orbite satisfaisante puisqu'on détecte le satellite à moins de 1 millidegré près (environ 0.5 millidegré) et très peu couteuse (ne nécessite qu'un petit téléscope amateur et un cliché photo du satellite). Cette méthode peut être satisfaisante pour faire de l'orbitographie (nous obtenons une précision de 300m environ sur le demi grand axe de l'ellipse) mais nécessite une étude plus approfondie (utilisation des équations de Kepler, version 3D).
Notre algorithme, bien que couteux en temps de calcul lorsqu'il y a beaucoup d'étoiles peut facilement être optimisé.