



---

# PROJET DE MATHEMATIQUES

---

$\otimes \trianglelefteq \text{Trifocal Tensor} \triangleright \otimes$

---

Mathematics Project realized in the IMAC engineering course  
by Guillaume Segado & Océane Rennesson sous la direction  
de M. NOZICK dans le cadre du cours de mathématiques

Décembre 2012 - Janvier 2013

# SOMMAIRE :

## INTRODUCTION

### I - Le Tenseur

1. Modélisation
2. La Matrice A
3. Calcul du Tenseur

### II - Le Transfert

1. Les Cas du Transfert
2. Les équations de transfert
3. Le calcul du point

### III - Résultats et Bilan

1. Résultats tests
2. Checking des objectifs
3. Améliorations possibles
4. Appréciation du projet

# INTRODUCTION :

## Résumé du projet :

Ce projet traite du domaine de la vision par ordinateur et de la géométrie projective. Nous disposons de trois images d'une même scène mais prises avec des points de vue différents, grâce à l'outil mathématique qu'est le tenseur, nous pouvons, à partir de la position des deux points similaires sur deux images, retrouver la position du point correspondant sur la troisième image. Pour cela, on doit dans un premier temps calculer le tenseur trifocal entre trois images, puis réaliser le transfert de point en utilisant le tenseur.

## Applications possibles ?

Nous prenons l'exemple particulier de trois images avec un tenseur trifocal, mais nous pouvons généraliser cette étude avec plusieurs images et ainsi reconstruire en 3D un objet à partir d'images 2D. Nous pouvons aussi prendre des scènes dynamiques avec 3 caméras par exemple.

Notre objectif est donc de coder en C++ (avec la SDL) un tenseur trifocal entre 3 images et de réaliser le transfert de points.

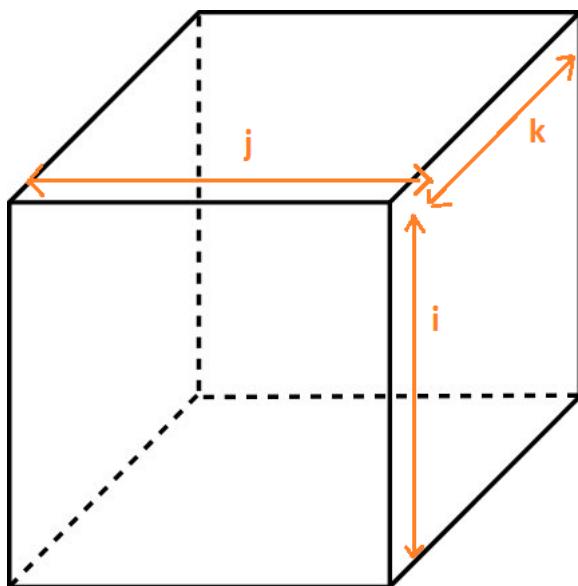
*Pour voir le checking des objectifs demandés, se référer au bilan.*

# Chapitre 1

## LE TENSEUR :

### 1.1 Modélisation :

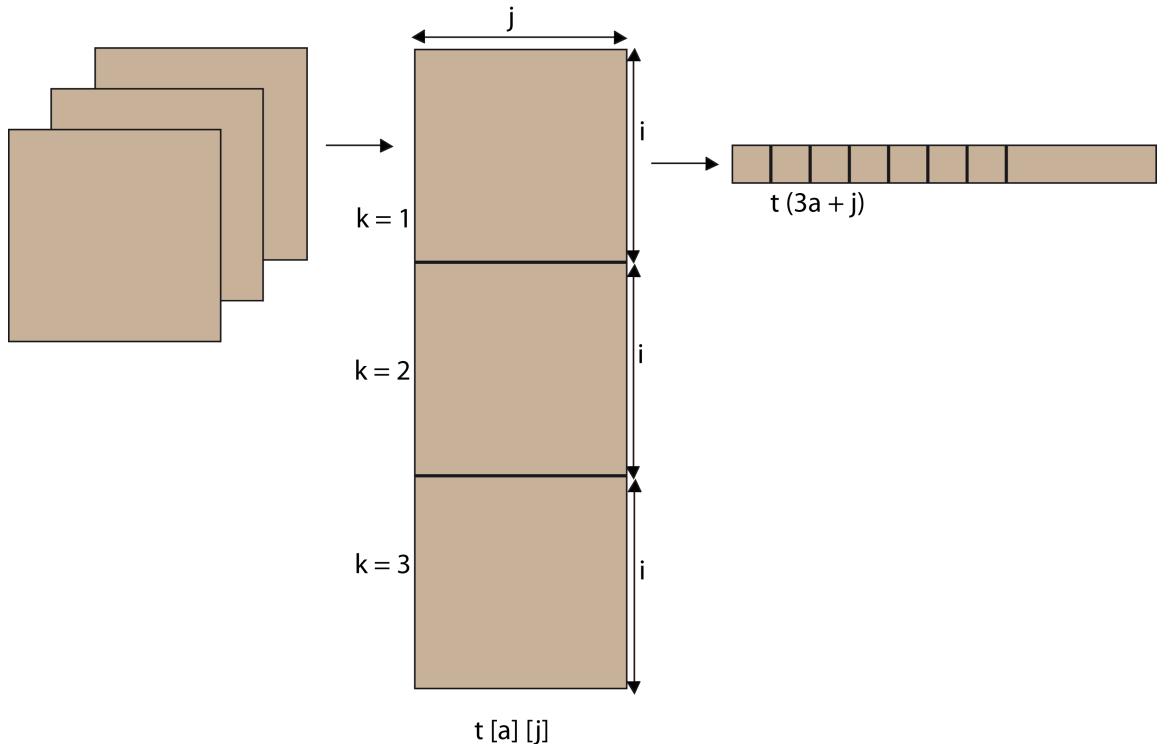
Un tenseur trifocal est une matrice  $3 \times 3 \times 3$ . Il possède donc trois indices pour pouvoir repérer un élément stocké à l'intérieur. Dans tout le reste de l'étude, le tenseur trifocal sera noté  $T_k^{ij}$  et aura cette forme :



Nous constatons que le tenseur comprend 27 composantes et pour y accéder, nous faisons varier  $i, j, k$  de 1 à 3 :  
 $T_1^{22}, T_3^{31}, T_1^{11}, T_1^{31} ou T_2^{32}$  par exemple.

Le premier problème qui survient est donc de modéliser le tenseur. Nous avons choisi de représenter notre objet tenseur sous la forme d'un vecteur

de 27 composantes pour simplifier l'approche en informatique et pour la suite avec les différentes équations de correspondances. Le problème est donc de passer d'un objet avec 3 'degrés' de liberté en un objet avec 1 'degré' de liberté. Voici donc l'approche mathématique que nous avons eu :



On passe de 3 'degrés' de liberté à 2 'degrés' de liberté (matrice) puis 1 'degré' de liberté (vecteur) 'a' varie dans l'intervalle [0,8]. Il faut donc une relation pour  $a = f(i,k)$ . En décrivant les valeurs de  $a$ , nous avons pu trouver une relation empirique qui lie  $a$  avec  $i$  et  $j$ .

$$a = 0 \rightarrow i = 1, k = 1$$

$$a = 1 \rightarrow i = 2, k = 1$$

$$a = 2 \rightarrow i = 3, k = 1$$

$$a = 3 \rightarrow i = 1, k = 2$$

$$a = 4 \rightarrow i = 2, k = 2$$

$$a = 5 \rightarrow i = 3, k = 2$$

$$a = 6 \rightarrow i = 1, k = 3$$

$$a = 7 \rightarrow i = 2, k = 3$$

$$a = 8 \rightarrow i = 3, k = 3$$

Nous n'avons pas pu trouver une relation unique pour  $a$ , car il y a une sorte de dualité, nous avons une relation qui fonctionne pour les 6 premiers termes ( $a = k^2 + i - 2$ ) de  $a$  et une relation qui fonctionne pour les 6 derniers termes

$(a = k^2 + i - 4)$  de a.

Nous pouvons donc accéder à un élément du tenseur avec cette algorithme, en surchargeant l'opérateur () pour le tenseur :

accès à un élément du tenseur (i,j,k)

Si  $k > 2$

On renvoie  $t[3(k^2 + i - 4) + j]$  (-1 si  $t[0]$  est défini)

Sinon  $k < 3$

On renvoie  $t[3(k^2 + i - 2) + j]$  (-1 si  $t[0]$  est défini)

Comme nous allons être amenés à appeler l'accès au tenseur assez souvent, il faut donc que la relation soit la plus simple possible et la moins complexe, avec cette relation, nous n'appellerons qu'un seul if() à chaque fois.

## 1.2 La Matrice A :

Nous avons plusieurs équations de correspondance qui mettent en jeu les différentes composantes du tenseur. Comme nous disposons de trois images, c'est une relation faisant intervenir trois points des trois images et le tenseur qui s'écrit de cette forme là :

$$x^i x'^j x''^k \epsilon_{jqs} \epsilon_{krt} T_i^{qr} = 0_{st}$$

Après simplification, nous utiliserons cette équation :

$$\sum_{k=1}^3 x_p^k (x'_p{}^i x''^3 T_k^{3l} - x'_p{}^3 x''^3 T_k^{il} - x'_p{}^i x''^l T_k^{33} + x'_p{}^3 x''^l T_k^{i3}) = 0^{il}$$

p représente la correspondance (il y en a n en tout), il nous faut donc au minimum 27 équations car le tenseur comporte 27 composantes, qui nous sont inconnues pour le moment. i et l varient de 1 à 2, ce qui génère 4 équations pour chaque correspondance. Il faut donc 7 correspondances de points au minimum  $4 \times 7 = 28$ , nous pouvons donc résoudre et le système est surdéterminé.

t est notre tenseur sous la forme d'un vecteur rempli de 27 inconnues dont de taille 27, nous devons donc résoudre  $A\vec{t} = \vec{0}$  avec  $\vec{0}$  est le vecteur nul de dimension 4n et A une matrice de taille 4n x 27

Notre approche pour remplir A a été la suivante :

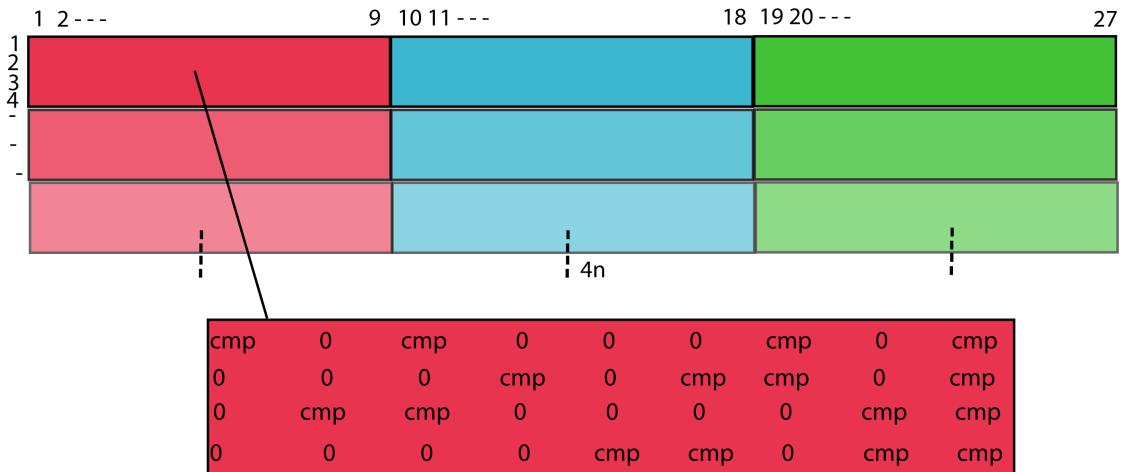
Nous avons écrit sur papier dans l'ordre les 27 éléments de  $t$ , c'est à dire :  $T_1^{11} T_1^{12} T_1^{13} T_1^{21}$  Nous avons fait varier  $i$ ,  $l$  et  $k$  et nous avons regardé les différentes composantes qu'il pouvait y avoir. Nous avons constaté que défois pour une inconnue, il y avait  $n$  composantes, défois  $2n$  composantes et défois  $4n$  composantes.

Nous avons ensuite étudié les répétition qu'ils pouvaient y avoir. Nous en avons trouvé deux grandes :

pour chaque  $p$  allant de 0 à  $n$

et pour  $k$  allant de 1 à 3.

Le schéma à coder qui nous a paru le plus simple est de remplir par 'bloc' la grande matrice  $A$ . voici nos blocs déterminés :



'cmp' correspond à 'composante'.

Nous pouvons remplir une à une les colonnes  $\rightarrow_{k=1} 9$  et après répéter le même schéma  $\rightarrow_{k=2} 18$  puis  $\rightarrow_{k=3} 27$ . Pour remplir chaque colonne, on peut boucler 4 par 4  $n$  fois, selon le nombre de correspondances. Donc en général, on remplit par bloc de  $4 \times 9 = 36$  composantes. Etant donné que nous avons analysé un bloc, on remarque que chaque bloc est composé de 20 zeros donc nous avons 16 composantes à rentrer par bloc. Nous avons choisi de faire deux boucles, une première sur  $p$  de 1 à  $n$  et une deuxième à l'intérieur sur  $k$  allant de 1 à 3.

C'est un choix de ne pas avoir fait de boucle faisant varier  $i$  et  $l$ , car  $i$  et  $l$  ne prennent que deux valeurs, 1 ou 2 et faire une boucle pour deux valeurs selon moi n'est pas assez pour faire une boucle, c'est un choix, j'ai

préféré rentrer en dur les valeurs de i et l dans le bloc qui se répetera selon n et selon k.

On crée A avec le nombre de correspondance  $4n$  en ligne et 27 en colonne. A est une matrice de Eigen. Nous initialisons A avec des zeros pour chaque composante puis nous la remplissons par bloc. Nous avons aussi au préalable stocké dans 3 matrices de Eigen les points  $x_p$  de l'image 1,  $x'_p$  de l'image 2 et  $x''_p$  de l'image 3 triés par correspondances (l'utilisateur charge ses points ou il les rentre manuellement en faisant les correspondances au clique).

### **1.3 Le Calcul du Tenseur :**

Nous disposons de A maintenant, pour résoudre  $A\vec{t} = \vec{0}$ , nous pouvons utiliser la SVD sur A. Eigen nous fournit un outil de SVD. La SVD va décomposer A en  $A = UDV^T$ . Nous avons vérifié que les éléments de D soient bien triés par valeur décroissante. Nous trouvons la solution du système, c'est à dire les 27 inconnues de t sur la dernière colonne de  $V^T$ ,  $V^T$  a 27 colonnes, nous prenons donc la 27<sup>ème</sup>. Le choix s'est porté sur la SVD, car c'est une méthode robuste, rapide et fiable. En effet nous obtenons des résultats satisfaisants plus les correspondances sont précises (se référer à Bilan et Résultats).

Nous conservons la solution dans le tenseur instancié dans le main tout simplement en rentrant les valeurs de la solution dans l'ordre dans le tenseur sous forme de vecteur. Nous n'avons pas besoin de réarranger sous forme de tenseur car nous avons une surcharge d'opérateur

## Chapitre 2

# LE TRANSFERT :

### 2.1 Les cas du transfert :

Il y a trois cas pour le transfert de point.

Si l'utilisateur veut transférer un point vers l'image 3, il a donc cliqué sur deux points correspondants dans l'image 1 2. Nous connaissons donc  $x$  et  $x'$  (par convention) et nous nous proposons de lui trouver la position du point  $x''$  sur l'image 3.

Si l'utilisateur veut transférer un point vers l'image 2, il a donc cliqué sur deux points correspondants dans l'image 1 3. Nous connaissons donc  $x$  et  $x''$  (par convention) et nous nous proposons de lui trouver la position du point  $x'$  sur l'image 2.

Si l'utilisateur veut transférer un point vers l'image 1, il a donc cliqué sur deux points correspondants dans l'image 2 3. Nous connaissons donc  $x'$  et  $x''$  (par convention) et nous nous proposons de lui trouver la position du point  $x$  sur l'image 1.

Pour faciliter la visualisation, quand l'utilisateur choisit ses deux points pour le transfert, au moment où il clique sur un pixel, un disque en couleur apparaît (son centre est le pixel choisi). Puis nous lui montrons son résultat du transfert sur la troisième image par un cercle non plein. Nous choisissons de ne pas effacer ces cercles, ainsi, l'utilisateur peut tester plusieurs combinaisons, et réaliser les trois types de transfert comme il le souhaite, dans l'ordre voulu.

## 2.2 Les équations de transfert :

Comment transférer un point sur une autre image ?

Cette fois ci, on dispose du tenseur trifocal rempli ainsi que de deux points correspondants sur deux images. Nous cherchons les coordonnées du troisième point, c'est à dire par exemple :  $(x''^1, x''^2)$  ou  $(x'^1, x'^2)$  ou  $(x^1, x^2)$ . Mais dans  $P$  l'espace projectif, nous ne devons pas négliger la coordonnée homogène. En effet, elle va nous être très utile lorsque nous devrons repasser en coordonnées cartésiennes.

Nous utilisons donc les mêmes équations que celle du calcul du tenseur, mais nous n'avons plus que 3 inconnus : Les 3 coordonnées du point à transférer.

$$\sum_{k=1}^3 x^k (x'^i x''^3 T_k^{3l} - x'^3 x''^3 T_k^{il} - x'^i x''^l T_k^{33} + x'^3 x''^l T_k^{i3}) = 0^{il}$$

Nous avons donc procédé avec la même démarche, nous avons fait varier  $i$  et  $l$  de 1 à 2 et étudié les différentes composantes pour chacune des trois coordonnées. Nous avons fait varier  $i$  et  $l$  dans cette ordre là.

$$i = 1 \ k = 1$$

$$i = 2 \ k = 1$$

$$i = 1 \ k = 2$$

$$i = 2 \ k = 2$$

Celà nous a donc généré 4 équations pour 3 inconnues, nous pouvons donc résoudre avec les mêmes outils, entre autre, la SVD. Nous allons donc créer avec une matrice  $G$  de taille  $4 \times 3$  et résoudre le système  $G\vec{x''} = \vec{0}$  par exemple.

Nous obtenons ces trois systèmes pour les trois cas différents (cmp pour composante) :

Le transfert pour  $x''$  (vers la troisième image)

$$\begin{pmatrix} cmp & 0 & cmp \\ cmp & 0 & cmp \\ 0 & cmp & cmp \\ 0 & cmp & cmp \end{pmatrix} \begin{pmatrix} x''^1 \\ x''^2 \\ x''^3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Le transfert pour  $x'$  (vers la deuxième image)

$$\begin{pmatrix} cmp & 0 & cmp \\ 0 & cmp & cmp \\ cmp & 0 & cmp \\ 0 & cmp & cmp \end{pmatrix} \begin{pmatrix} x'^1 \\ x'^2 \\ x'^3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Le transfert pour x (vers la première image)

$$\begin{pmatrix} cmp & cmp & cmp \\ cmp & cmp & cmp \\ cmp & cmp & cmp \\ cmp & cmp & cmp \end{pmatrix} \begin{pmatrix} x^1 \\ x^2 \\ x^3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Après avoir développé les équations de transfert, nous avons remarqué que pour chaque cas, dans les composantes des matrices G, étaient présents tous les élément du tenseur, il n'en manquait pas un.

Pour remplir les matrices G, nous faisons une boucle sur k qui varie de 1 à 3, car les composantes sont des sommes sur k, donc des morceaux se répètent. Encore une fois, par choix, nous ne faisons pas de boucle sur i et j qui varient seulement de 1 à 2.

Pour gérer les trois cas, la fonction 'transfert' est adaptée pour remplir la matrice G correctement.

### 2.3 Le Calcul du point :

Pour calculer le point transféré, il nous faut résoudre le système :  $G\vec{x}'' = \vec{0}$ . Nous allons utiliser les mêmes outils de la matrice A pour trouver la solution. On pratique la SVD sur G. et nous récupérons la solution sur la dernière colonne : la 3<sup>ème</sup>.

Nous obtenons un vecteur solution avec des valeurs très petite. En effet, la SVD calcule le vecteur solution  $\vec{x}$  tel que  $\|\vec{x}\| = 1$ . De plus pour pouvoir repasser dans l'espace euclidien et ainsi avoir des résultats cohérents, il faut diviser les coordonnées solutions par la coordonnées homogène. En effet, on obtient des résultat concluant, le transfert est effectué. Par exemple, nous obtenons :

$$\begin{pmatrix} 0,8786 \\ 0,4776 \\ 0,005351 \end{pmatrix} \doteq \begin{pmatrix} 164,19 \\ 89,25 \\ 1 \end{pmatrix}$$

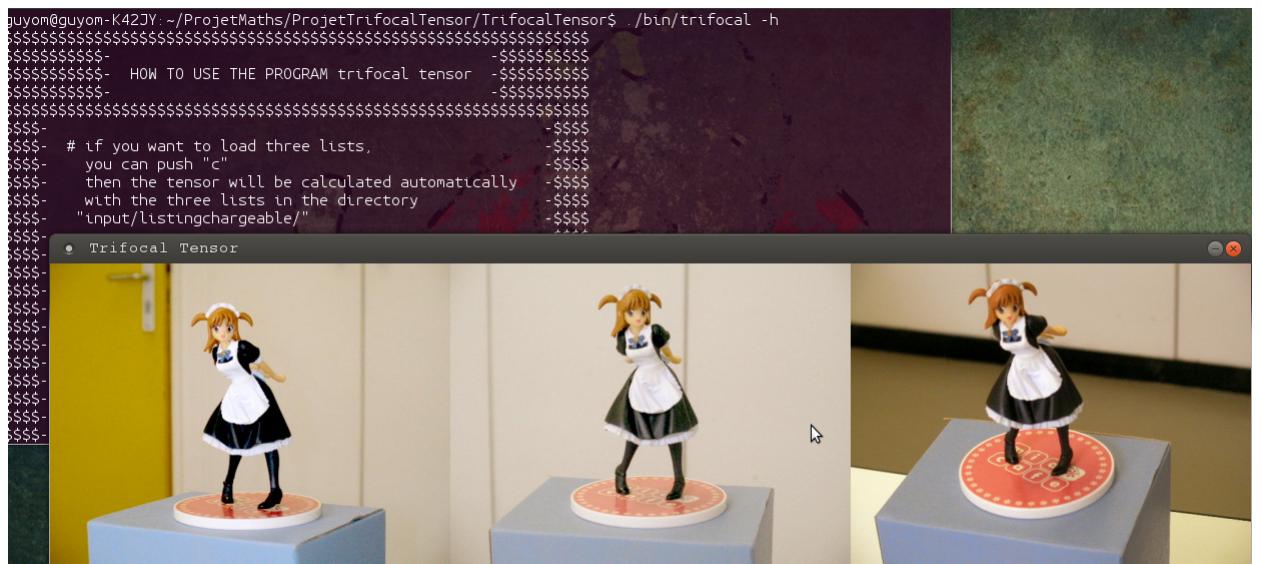
## Chapitre 3

# RESULTATS ET BILAN :

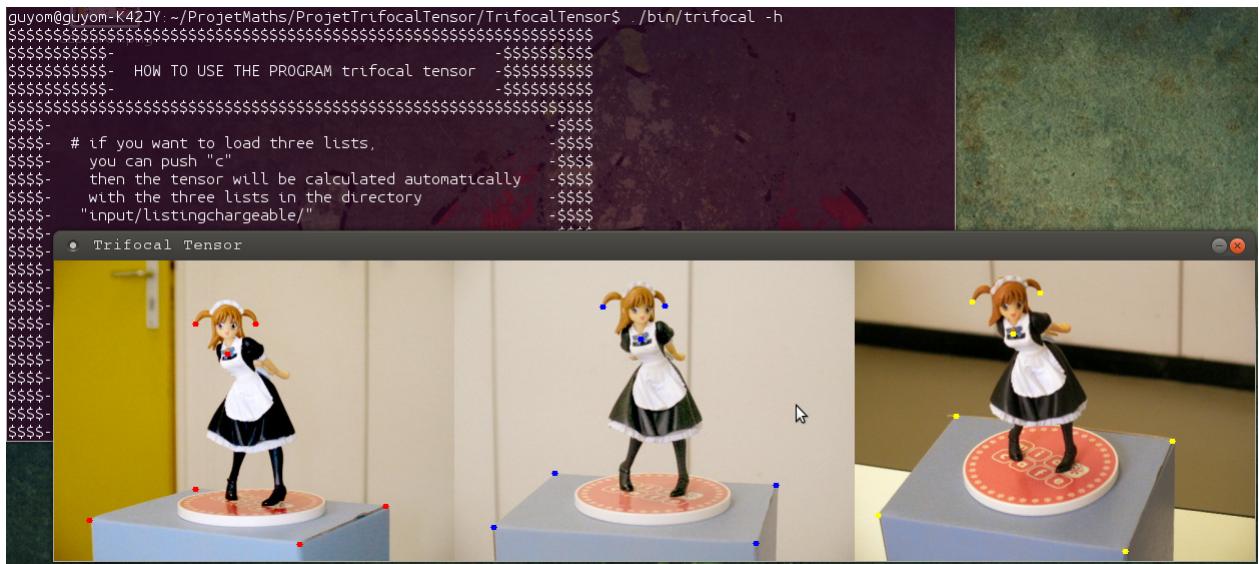
### 3.1 Les Résultats :

#### Test avec les images du sujet

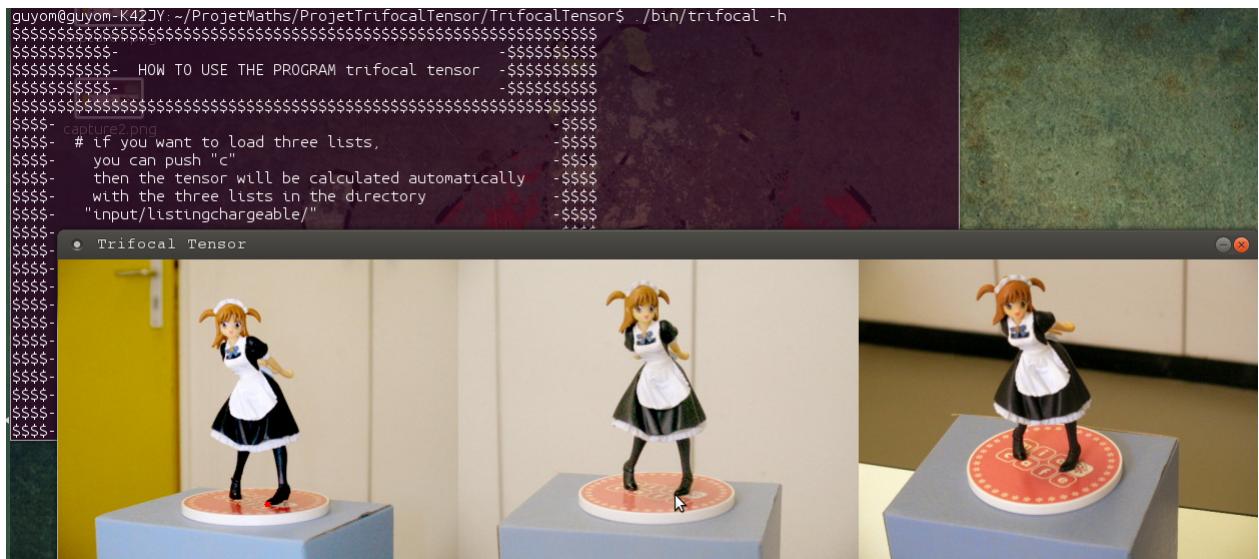
Le programme se lance (-h pour activer l'aide). Les images sont chargées. L'utilisateur lance alors le chargement de listes pré-construites, il appuie alors sur la touche 'c' pour le mode de chargement de listes.



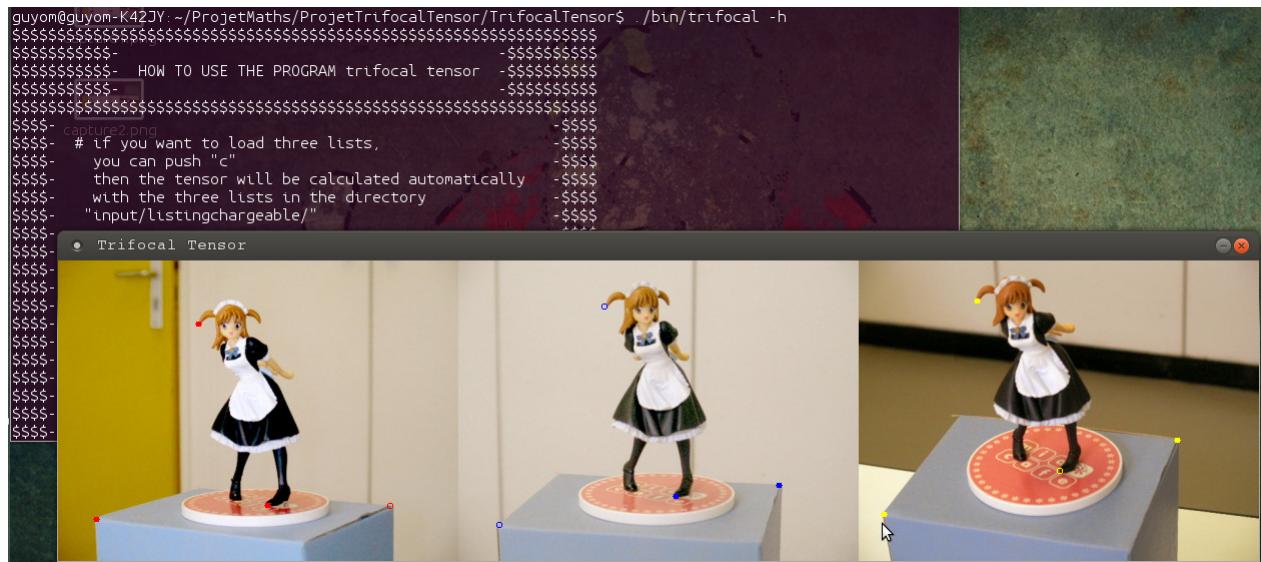
Les points sont chargés, et des disques de couleur apparaissent sur les images pour montrer à l'utilisateur les points qu'il vient de charger pour le calcul du tenseur. Le programme attend 2 secondes pour passer ensuite au mode de transfert après avoir calculé le tenseur et effacé des images les points de départs.



L'utilisateur passe en mode transfert, il clique sur deux points correspondants sur deux images.



Puis un cercle de couleur non rempli se dessine sur la troisième image afin de montrer la position du point transféré. A chaque nouveau test, les cercles ne sont pas effacés pour laisser à l'utilisateur le choix d'apprécier ses résultats.



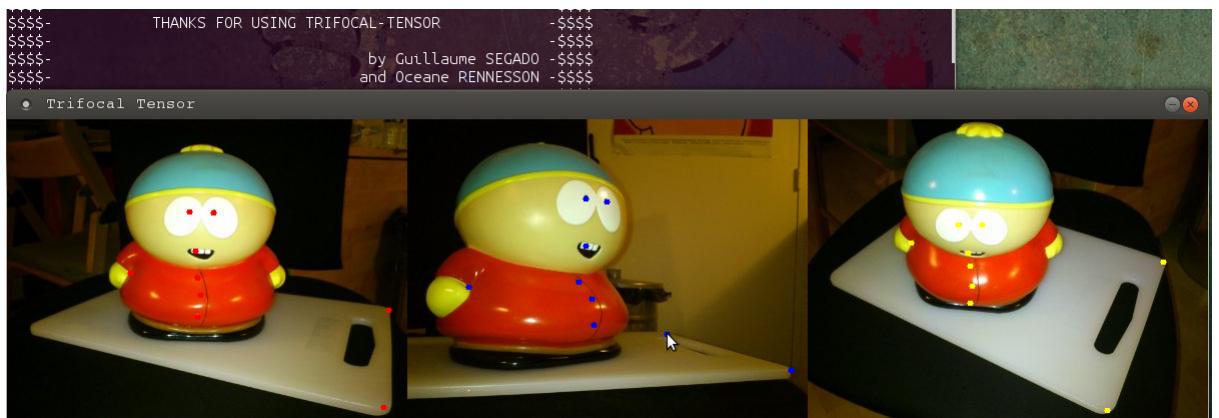
### Test avec nos images

Et oui, Cartman s'est porté volontaire pour participer au projet trifocal tensor, il à même poser pour nous. Là pareil, l'utilisateur va rentrer des points, mais cette fois, il ne va pas charger des listes pré-construites, il va passer en mode manuel et cliquer sur ses correspondances dans l'ordre (au minimum 7). Une fois fini, l'utilisateur doit appuyer sur la touche 'f' pour calculer le tenseur et passer au mode de transfert de points. Les correspondances sont sauvegardées sous forme de fichier.list mis en page de cette façon là :

row 11

col 3

```
104 119 1
262 189 1
173 144 1
161 230 1
99 171 1
223 141 1
269 212 1
163 229 1
81 213 1
388 83 1
69 90 1
```



Il passe ensuite au mode transfert qui fonctionne sur le même principe que ci-dessus.



Merci Cartman

### **3.2 Bilan :**

**Checking des points demandés :**

- ✓ implémenter un programme de gestion de tenseur trifocal.
- ✓ être écrit en C++
- ✓ utiliser un Makefile
- ✓ ne plus afficher de warning lors de la compilation (utilisation du flag -Werror)
- ✓ fonctionne sous Linux
- ✓ Lancement sous console avec une aide en anglais qui s'affiche si l'option -h est ajouté à la commande d'exécution
- ✓ n'utilise que l'interface de la SDL

- ✓ chargement de listes déjà établie et correspondances cliquables ainsi que sauvegarde dans un fichier. la sauvegarde s'effectue sous la bonne forme, mais pour le chargement de liste, ce n'est pas la bonne forme.
- ✓ Calcul d'un Tenseur transfert de points (les 3 cas de transfert)
- ✓ Affichage de cercles ou disques pour les points sélectionnés, chargés ou transférés

#### **Améliorations possibles :**

Il y a beaucoup d'améliorations possibles pour ce programme. Par exemple au niveau de la gestion, au niveau de l'interface. On peut par exemple se tromper en effectuant un clic de point, on pourrait imaginer un 'undo' de la dernière action entreprise par l'utilisateur. On peut aussi par exemple lors du transfert, sauvegarder les résultats pour pouvoir les exploiter ou les réutiliser plus tard.

#### **Appréciation du projet :**

Pour ma part, ce projet m'a beaucoup apporté, tant en compétences informatiques qu'en compétences mathématiques. Le problème au début me paraissait concret, je visualisais l'utilité. Mais la partie mathématique m'a rebuté car complexe à première vue. Une fois que l'on pose bien le problème plusieurs fois, on trouve des résultats, et on trouve du plaisir à voir de la cohérence par la suite selon les choix mathématiques et informatiques que l'on prend. Au niveau de l'informatique, j'ai pu me familiariser avec Eigen, et surtout découvrir la puissance de certains outils informatiques, notamment la SVD. Et pour finir, j'ai de plus développé mes compétences en L<sup>A</sup>T<sub>E</sub>X