0.1 Section bidule

0.1.1 Calcul du tenseur

A partir de l'équation dégagée et indiquée dans l'énoncé :

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

qui correspond en fait à un système de plusieurs équations, nous devons pour calculer le vecteur t retranscrire ce système sous forme matricielle, de la forme At = 0, et donc construire la matrice A. Chaque élément de A correspond aux coefficients présents dans les équations devant les éléments de t. Différentes boucles permettront donc d'insérer les bons coefficients au bon endroit dans la matrice.

La première étape consiste à initialiser chaque élément de A à 0, puisque dans chacune des équations du système, seuls 12 éléments de t sur 27 apparaissent, les autres ont donc un coefficient nul. Eigen nous fournit des outils pour initialiser la matrice à 0 directement.

La deuxième étape est le remplissage de A. En étudiant l'équation (1), on comprend que ce remplissage sera possible grâce à 4 boucles : 3 boucles sur les p, i, et l, permettant de se positionner sur la bonne ligne de la matrice puisque ce sont les variables qui génèrent de nouvelles équations, puis une boucle sur k qui permet de se positionner sur la bonne colonne en fonction de la coordonnées de t sur laquelle on travaille.

Dans l'énoncé, i et l varient de 1 à 2, p de 1 au nombre de points de correspondance, et k de 1 à 3. Puisque les lignes et colonnes de nos matrices débutent à 0, nous aurons donc i et l variant de 0 à 1, k de 0 à 2 et p de 0 au nombre de points -1.

Les coefficients que l'on va insérer dans A vont dépendre des coordonnées des points x, x' et x". Or ces coordonnées sont stockées dans nos listes de points de correspondance, donc sous forme matricielle. Il faut donc être également capable de les retrouver. Par exemple, si l'on considère que les points de la première, deuxième et troisième image sont respectivement stockés dans les matrices list1, list2 et list3, alors la coordonnée k du point de correspondance p de la première image correspond au point list1(p, k).

La dernière étape consiste à calculer t grâce à la décomposition SVD fournie par Eigen. La matrice A est alors décomposée en un produit de 3 matrices U*D*V et l'on peut alors stocker les valeurs de notre tenseur qui correspondent à la dernière colonne de V (la colonne 26).

MODELE POUR ALGORYTHME LATEX : http://www.math-linux.com/spip.php?article130

Ainsi, toujours d'après l'équation (1), nous arrivons au pseudo-code suivant :

Algorithm 1 Calcul du tenseur

```
Initiate A to 0
for p = 0 to nbPoints − 1 do
  for i = 0 to 1 do
    for l = 0 to 1 do
      for k = 0 to 2 do
        A(4*p + 2*i + l, 9*2 + 3*l + k) += list1(p,k) * list2(p,i) * list3(p,2)
        A(4*p + 2*i + l, 9*i + 3*l + k) -= list1(p,k) * list2(p,2) * list3(p,2)
        A(4*p + 2*i + l, 9*2 + 3*2 + k) -= list1(p,k) * list2(p,i) * list3(p,l)
        A(4*p + 2*i + l, 9*i + 3*2 + k) += list1(p,k) * list2(p,2) * list3(p,l)
      end for
    end for
  end for
end for
A = UDV {SVD decomposition}
tensor ← V.col(26)
```

0.1.2 Transfert

t étant maintenant connu, nous cherchons à retrouver l'un des 3 points à partir des 2 autres. Le principe est le même : retranscrire le système sous forme matricielle et donc remplir une matrice B.

Avant de se lancer tête baissée dans le calcul d'un vecteur à 3 coordonnées comme à l'étape précédente, il convient de remarquer que la 3ème coordonnée du point que nous cherchons est connue et vaut 1 puisqu'il s'agit de sa coordonnée homogène. Nous n'avons donc que 2 inconnues, et devons cette fois-ci résoudre une équation matricielle de la forme Bv = b, v étant le vecteur à 2 coordonnées.

Il convient de réarranger l'équation (1) en considérant cette dernière remarque. Pour être capable de retrouver un point de n'importe quelle image par rapport aux 2 autres, nous devons traiter cette équation pour chacun des 3 cas.

Si l'on cherche le point x' alors x'3 vaut 1 ce qui nous amène à l'équation suivante :

$$x^k x'^i x''^3 T_k^{3l} - x^k x'^i x''^l T_k^{33} = x^k x''^3 T_k^{il} - x^k x''^l T_k^{i3}$$

Si l'on cherche le point x" alors x"3 vaut 1 ce qui nous amène à l'équation suivante :

$$x^k x'^3 x''^l T_k^{i3} - x^k x'^i x''^l T_k^{33} = x^k x'^3 T_k^{il} - x^k x'^i T_k^{3l}$$

Enfin si l'on cherche le point x alors x3 vaut 1 ce qui nous amène aux équations suivante :

si k=3 :

$$x'^i x''^3 T_k^{3l} - x'^3 x''^3 T_k^{il} - x'^i x''^l T_k^{33} + x'^3 x''^l T_k^{i3} = 0^{il}$$

sinon :

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

Ainsi il nous faut cette fois-ci remplir la matrice B et le vecteur b. Pour plus de clarté nous utiliserons MatB et Vecb plutôt que B et b.

Algorithm 2 Transfert

```
Require: x1 and x2 are known
  Initiate MatB to 0
  Initiate Vecb to 0
  for i = 0 to 1 do
    for l = 0 to 1 do
      for k = 0 to 2 do
        if we search the point x then
          factor = x1(i) * x2(2) * tensor(2, l, k) - x1(2) * x2(2) * tensor(i, l,
          k) - x1(i) * x2(l) * tensor(2, 2, k) + x1(2) * x2(l) * tensor(i, 2, k)
          if k=2 then
            Vecb(2*i + l) -= factor
          else
            MatB(2*i + l, k) += factor
          end if
        else if we search the point x' then
          MatB(2*i + l, i) += x1(k) * ( x2(2) * tensor(2, l, k) - x2(i) *
          tensor(2, 2, k) )
          Vecb(2*i + l) -= x1(k) * ( x2(l) * tensor(i, 2, k) - x2(2) * tensor(i,
          l, k) )
        else if we search the point x" then
          MatB(2*i + l, l) += x1(k) * ( x2(2) * tensor(i, 2, k) - x2(i) *
          tensor(2, 2, k) )
          Vecb(2*i + l) -= x1(k) * ( x2(i) * tensor(2, l, k) - x2(2) * tensor(i,
          l, k) )
        end if
      end for
    end for
  end for
  MatB = UDV {SVD decomposition}
  solution ← SVD.solve(Vecb)
```