Travaux dirigés Matrices n°5

Mathématiques pour l'informatique

—IMAC 2—

▸ **Exercice 1.** ***Stabilisation d'images***

1. *Télécharger le code sur la page de l'enseignant.*
2. *Compiler et exécuter le programme :*
   - *avec une vidéo comme argument.*
   - *sans argument, mais avec une webcam branchée.*
   - *jeter un oeil à* `\dev\camera*`
3. *lire brièvement le code.*
4. *Créez une matrice zoom (Eigen::Matrix3d) de la forme :*

$$zoom = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

   avec $\alpha = 0.7$.

5. *Créez une matrice de translation* `Hcentering` *qui translate le milieu de l'image en* $(0, 0, 1)^\top$, *ainsi que sa matrice inverse* `HcenteringInv`.
6. *Dans la partie "image stabilization", vous avez accès à une matrice H (de Eigen) correspondant à une transformation 2D en coordonnées homogènes. Cette matrice est ensuite convertie en matrice de OpenCV pour être appliquée à l'ensemble des pixels de l'image. Changez le code pour que cette matrice de OpenCV devienne*

```
Hocv = HcenteringInv * zoom * Hcentering * H
```

   Que constatez-vous?

7. *On cherche à stabliliser la vidéo, pour cela, nous trackons un ensemble de points d'une image à l'autre :* `initialPts(composante,i)` $\rightarrow$ `currentPts(composante,i)`, *où* `composante` *vaut 0 pour* x, *1 pour* y *et 2 pour* w, *et* i *correspond à l'indice du point tracké. Trouvez le système surdéterminé permettant de modéliser une transformation rigide entre les point initiaux et les points trackés. Une transformation rigide (isométrie) est une combinaison d'une rotation et d'une translation.*

8. *Coder et résoudre ce système.*

9. *Pour fonctionner correctement, ce système doit utiliser des données déjà centrées. Appliquez la matrice* `Hcentering` *aux données.*

10. *Extraire les données de rotation et de translation du résultat de ce système et les mettre sous forme d'une matrice de transformation M.*

11. *Remplir la matrice H telle que* `H = HcenteringInv * M * Hcenterin`*. C'est la matrice de stabilisation d'image. Il faut l'appliquer à toute l'image (ce qui est fait par défaut dans le programme), mais aussi aux points initiaux : on cherche la transformation de stabilisation qui opère sur nos points déjà stabilisés.*

12. *A présent, les translations et les rotations sont stabilisées, mais quand on se rapproche d'un objet, la stabilisation a tendance à faire un scale ... pour stabiliser l'image. Comment expliquez-vous ce problème?*

13. *Pour remédier à ce problème, nous pouvons renforcer notre matrice de rotation en faisant en sorte qu'elle soit bien une matrice orthogonale et non une matrice de rotation "approximative", comme celle que l'on calcule. Pour celà, il suffit de lui appliquer une SVD :*

$$M = UDV^\top$$

et de forcer toutes les valeurs singulières à prendre la valeur 1 :

$$M = UV^\top$$

14. *Une fois arrivé là, notre système stabilise bien, mais peut-être un peu trop. Lorsque la caméra bouge et quitte le champs de vision qu'elle avait au départ, l'image stabilisée a tendance à sortir de la fenêtre d'affichage. Pour celà, nous allons limiter la portée de réctification de la matrice M en bornant l'angle de rotation ainsi que la translation. L'angle de rotation se récupère grace à la fonction* $\arcsin(\sin(\alpha))$ *(qui fonctionne bien pour un angle entre -90 et 90 degrès). On bornera l'angle à 60 degrés et la translation à 100 pixels par composante.*

15. *Pour finir, on rajoutera à la matrice H la contribution d'une matrice identité tendant à réduire l'impact de la stabilisation dans le temps. Si la caméra ne bouge plus, l'image stabilisée se recentre lentement au milieu de l'image :*

```
H = 0.01*Matrix3d::Identity() + 0.99*HcenteringInv * M * Hcentering;
```