Unité 5I-IG4 – Vision par ordinateur
TP1 : points de Harris

Xavier Hilaire
x.hilaire@esiee.fr

28 septembre 2022

1 Préparation

Logez-vous sur Linux. Si ce n'est déjà fait, visitez `https://mvproxy.esiee.fr`, et indiquez qui est votre binôme en le citant une seule fois dans les vœux d'affectation dès le début de la séance. Vous serez appariés automatiquement avant la fin de la première séance.

La version de OpenCV que vous devez utiliser se trouve dans `/nfs/opt/bdr/OpenCV`. Il s'agit d'une version recompilée. Il se trouve que OpenCV est également installé dans une version 3 sur les machines de l'ESIEE, d'où un risque de conflit entre bibliothèques.

Pour l'éviter, votre variable d'environnement `LD_LIBRARY_PATH` doit absolument être positionnée sur `/nfs/opt/bdr/OpenCV/lib`. Voici comment le faire de façon permanente sur une machine donnée :

```
echo "setenv LD_LIBRARY_PATH /nfs/opt/bdr/OpenCV/lib" >> $HOME/.cshrc
source .cshrc
```

Les nouveaux terminaux ouverts définiront automatiquement `LD_LIBRARY_PATH`. Rappelez-vous également que votre HOME Unix se trouve dans `/user/$user/homedir`, en non pas dans `/user/$user` : en changeant de machine d'une séance à l'autre, vous perdrez l'effet des lignes ci-dessus et vous ne retrouverez pas vos fichiers si vous ne les sauvegardez pas dans le bon répertoire.

Les fichiers nécessaires au TP se trouvent ici : `https://perso.esiee.fr/~hilairex/5I-IG4/TP1.tgz`

2 Points de Harris

2.1 Calcul des points d'intérêt

Dans son état actuel, le programme `harris.cpp` ne fait rien de plus que convertir en niveaux de gris, si c'est nécessaire, l'image qui lui est passée en unique argument, et afficher le résultat. La fonction `getDoGX(Mat& K,int w,double sigma)` calcule et stocke dans `K` les valeurs de la première dérivée partielle par rapport à x d'une gaussienne bidimensionelle d'écart-type σ, en les limitant à une fenêtre de taille $(2w+1) \times (2w+1)$.

1. Utiliser filter2D() et `getDoGX()` pour produire des images de gradients en x et y. Afficher le résultat. Tester le programme sur l'image `shapes.png`.
2. Ajouter une fonction `double harris(Mat& Gx, Mat& Gy, int x, int y, int w, Mat& H)`, qui, étant donné des images de gradients Gx, Gy, calcule et stocke dans H la matrice de Harris centrée au point d'intérêt (x, y), en sommant dans une fenêtre de taille $2w +$

$1 \times 2w + 1$, et retourne la valeur du score de Harris correspondant avec $k = 0.1$. Tester la fonction sur 1 ou 2 points remarquables de l'image `shapes.png`. Conseil : si vous utilisez des produits de matrices dans vos calculs, ces dernières doivent être de type `CV_32FC1`, c'est à dire à coefficients `double`, sinon le produit ne fonctionne pas et lance une exception. Vous pouvez avoir intérêt à faire tous vos calculs en `double`.

3. Ajouter du code dans le callback pour afficher :
   — la matrice de Harris
   — ses vecteurs et valeurs propres,
   — et le score Harris & Stephens avec $k = 0.2$
   à chaque point cliqué par l'utilisateur sur l'image source. Vous aurez besoin de la fonction eigen() pour ce faire.

2.2 Tests

2.2.1 Sensibilité à la forme

Testez et reportez les valeurs de H et du score de Harris & Stephens que vous obtenez en différents points :

— un angle saillant, et un angle obtu : comment varie la matrice ? Que vaut le score ?
— une portion de droite : retrouvez-vous bien une valeur propre nulle ? Le vecteur propre associé à la valeur propre dominante est-il bien dans bonne direction ? Comment évolue le score ?
— les angles droits du carré : le score est-il le même d'un coin à l'autre ? Pourquoi ? Comment sont (théoriquement) orientés les vecteurs propres dans ce cas précis ?

2.2.2 Sensibilité au contraste

Divisez les niveaux de gris de l'image source `shapes.png` par 2, et comparez les valeurs propres que vous obtenez.

2.2.3 Sensibilité à σ

Multipliez par 2 la valeur de σ que vous avez utilisée pour calculer les dérivées de gaussienne dans la première partie. Comparez les valeurs de H que vous obtenez avant et après, en un point particulier. Quel est l'effet général d'une augmentation de σ ?

3 Détecteur de Harris-Laplace

Dans `harris-laplace.cpp`, modifiez l'appel à `create()` pour que le résultat soit plus sélectif : les points clés détectés doivent être moins nombreux (de l'ordre de 500 plutôt que 5000) et le seuil de Harris doit être plus sévère (de l'ordre de 0.1 plutot que 0.01).

Essayez le programme sur l'image `shape.png`, et faites varier le nombre d'octaves autorisées. Quels sont les points qui apparaissent lorsque ce nombre croît ? Pourquoi s'écartent-ils des coins effectifs ?

Testez le programme sur une image riche en coins (les vues de Vincennes fonctionnent en principe assez bien). Un réglage correct doit vous permettre de trouver une superposition de points à différentes échelles, que vous ne pourriez pas trouver tous par un seuillage sur le score de Harris à une seule.