INF555 TD2
Image 2D et Matrices

Frank NIELSEN
nielsen@lix.polytechnique.fr

21 Septembre 2011

1 Détection de caractéristiques dans l'image : points de Harris-Stephens

- Écrire une applet/application qui lit l'image couleur `polytechnique.png` et l'affiche en niveau de gris. On mappera les pixels (r, g, b) en niveau d'intensité suivant la formule: $I(r, g, b) = 0.3r + 0.59g + 0.11b$.

- Calculer le gradient *discret* de l'image

$$\nabla I = \left[G_x = \frac{\partial I(x,y)}{\partial x} \quad G_y = \frac{\partial I(x,y)}{\partial y} \right].$$

  (On sortira le gradient discret en x, puis en y dans une image, puis on visualisera son intensité: $||\nabla I|| = \sqrt{G_x^2 + G_y^2}$)

- Écrire une fonction statique qui prend en argument une image et un masque de convolution, et retourne en sortie l'image résultat. Appliquer ensuite les masques 2×2 suivant:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

  Commentez les résultats obtenus.

- La méthode d'Harris-Stephens procède en calculant dans une région où se trouve un "feature": $w(x, y)$ (disons $5x5$ par exemple).

$$M = \sum w(x,y) \begin{bmatrix} G_x^2 & G_x G_y \\ G_x G_y & G_y^2 \end{bmatrix}$$

  On calcule ensuite $R = \det M - k(\mathrm{Trace} M)^2$ avec $k \simeq [0.04, 0.06]$. On seuille ensuite les pixels en retenant uniquement ceux pour lesquels $R >$ Threshold. Enfin, on termine en ne sélectionnant que les maxima locaux. (Cf. `http://www.cim.mcgill.ca/~dparks/CornerDetector/mainApplet.htm`)

Pour mesurer le temps de calcul, on utilisera `System.currentTimeMillis();` qui retourne un `long`: le temps écoulé depuis le 01/01/1970 (La différence entre deux appels mesure donc la durée). (*** A finir chez soi ***)

2 Utilisation de la bibliothèque JAMA pour les matrices en Java

- Télécharger JAMA (*A Java Matrix Package*) (copie locale disponible sur la page du TD). Installer ce fichier dans votre *class path* Java. Par exemple, sous Windows dans l'invite de commandes:

  ```
  C:> echo %classpath%
  C:> set classpath=%classpath%;Jama-1.0.2.jar
  C:> echo %classpath%
  ```

  On peut aussi definir le chemin des bibliothèques explicitement dans le compilateur Java:

  ```
  javac -cp Jama-1.0.2.jar  filename.java
  ```

  (Vous pouvez aussi unzipper le fichier jar. Cela créé un repertoire Jama contenant tous les bytecodes, etc.)

- Compiler et tester le programme `TestJama.java`. Regarder rapidement la documentation des différentes décompositions de matrices disponibles: LU(D), Cholesky, QR, valeurs propres et SVD (singular value decomposition).

- Ecrire un programme qui tire aéatoirement une matrice carrée M de dimension arbitraire d, et indique ci celle-ci est positive définie ou pas ($\forall x \neq 0,\ x^T M x > 0$). Une condition équivalente est que la matrice ait ces valeurs propres toutes positives. Vérifier que pour une matrice A inversible, $A^T A$ est positive définie. En déduire une méthode simple pour générer aléatoirement des matrices positives définies.

(A titre du curiosité, regarder aussi `http://math.nist.gov/javanumerics/`)