Université de Bordeaux
2015/2016

Master S&T Informatique
Spécialité Image, Son, Vidéo
U.E. « Image et Son : Fondements »

Estimation du mouvement dans des séquences d'images

TD 1 : Prise en main d'OpenCV

Résumé

Le but de ce premier TD est de se familiariser avec la bibliothèque OpenCV, en écrivant quelques fonctions utiles pour les TDs suivants.

1 Introduction

La compensation du mouvement peut être divisée en deux parties :

1. **Estimation** On estime un ensemble de vecteurs de déplacements entre une image de référence et l'image courante.
2. **Compensation** On applique ces vecteurs de déplacements à l'image de référence pour obtenir l'image compensée.

2 Travail à réaliser

Afin de se familiariser avec OpenCV, vous allez coder quelques fonctionnalités qui seront nécessaires lors de l'implémentation et de l'évaluation des algorithmes de compensation de mouvements.

Conseils

- ✓ Chargez les images en niveaux de gris (le type des données est alors `unsigned char`)
- ✓ Séparez les coordonnées x et y des champs de vecteurs dans 2 matrices distinctes

2.1 Extraction de la séquence des images d'une vidéo

Écrire une classe `VideoFrames` qui possède 2 fonctionnalités :

- ✓ stocker l'ensemble des trames d'une vidéo dans un vecteur de matrices
- ✓ sauvegarde chaque trames d'une vidéo dans un fichier image au format PNG.

Pour la lecture trame-par-trame d'un vidéo, vous pouvez utilisez la classe `cv::VideoCapture`.
Le stockage peut se faire dans un attribut de classe de type `std::vector<cv::Mat>`.
Prévoyez un « getter » qui retourne la i^{e} trame stockée dans la classe.

Exemple d'exécution : `./monExecutable video.avi repertoireDestinationSequenceTrame`

2.2 Affichage d'un champ de vecteurs

1) Écrire une fonction qui permet de dessiner un champ de vecteurs sur une image. Vous pouvez utiliser les primitives de dessin d'OpenCV telles que `cv::line`.

Attention : si vous dessinez tous les vecteurs, vous risquez de saturer l'image. Pensez à ajouter à la fonction un paramètre permettant de ne dessiner qu'un vecteurs tous les X pixels.

2) Écrire un programme qui utilise cette fonction et affiche l'image résultante à l'écran (en utilisant `cv::namedWindow` et `cv::imshow`). Étant donné que n'avons pas encore de fonction pour estimer ces champs de vecteurs, vous faire des tests avec des vecteurs fixes identiques dans toute l'image, ou bien encore tirer les valeurs aléatoirement.

Exemple d'exécution : `./monExecutable image.png 10`

2.3 Compensation de l'image

Écrire une fonction qui applique un champ de vecteurs de déplacement D à une image I afin de la compenser. La valeur d'un pixel compensé correspond donc à la valeur du pixel atteint lorsque l'on applique le vecteur en ce point.

$$I_C(x, y) = I((x, y) + D(x, y)) = I(x + d_x, y + d_y)$$

2.4 Calcul de l'image d'erreur

Afin de visualiser les erreurs de recalage, écrire une fonction qui calcule l'image d'erreur E entre l'image courante I et l'image compensée I_C, centrée en 128.

$$E(x, y) = min(255, max(0, I(x, y) - I_C(x, y) + 128))$$

2.5 Calcul de l'EQM

Afin d'évaluer la qualité d'une compensation de mouvement, écrire une fonction qui calcule l'Erreur Quadratique Moyenne (EQM) entre l'image courante I et l'image compensée I_C.

$$EQM = \frac{1}{w * h} \sum_{x,y} (I(x, y) - I_C(x, y))^2$$

Exemple d'exécution : `./monExecutable image.png`

Liens

Estimation du mouvement

http://fr.slideshare.net/mmv-lab-univalle/slides7-ccc-v8

https://cagnazzo.wp.mines-telecom.fr/files/2013/05/poly_me.pdf

API OpenCV

http://docs.opencv.org/2.4