# INF4710
# Introduction aux technologies multimédia

## A2016 - Travail pratique #3
## Indexation de fichiers multimédia : décomposition en prises de vue

---

**Objectifs :**

- Permettre à l'étudiant de se familiariser avec l'histogramme d'une image, la convolution d'un signal 2D, ainsi qu'avec les algorithmes de détection de transitions dans une séquence vidéo

**Remise du travail :**

- Au plus tard, le 28 novembre, 14h00 sur Moodle – **aucun retard accepté**

**Références :**

- Voir les notes de cours sur Moodle (indexation contenu pictural)

**Documents à remettre :**

- Tous vos scripts/fonctions/sources ainsi qu'un rapport (**.pdf**) dans une archive (.zip/.7z/...) nommée convenablement

**Autres directives :**

- Les en-têtes et les commentaires sont fortement suggérés dans le code, mais non obligatoires (c'est alors plus facile de vous donner des points si rien ne fonctionne!)

---

## Présentation

L'objectif de ce travail pratique est d'implémenter un algorithme de détection de transitions entre scènes dans une vidéo par différences d'histogrammes et/ou détection et mise en correspondance d'arêtes. Contrairement à ce qui vous était demandé aux deux premiers TPs, pour le TP3, vous aurez à développer votre propre algorithme, et un poids beaucoup plus important de la notre finale est accordé à votre rapport. Traitez ce dernier comme un compte-rendu de développement (avec résultats de tests) **que vous enverriez à un client**. Notez toutefois que ce dernier **n'y connaît RIEN en programmation**, alors ne discutez pas de votre code, et n'utilisez pas de pseudocode! Celui-ci est plus intéressé par une description haut niveau de votre algorithme; vous devrez donc discuter principalement de vos choix stratégiques, de vos paramètres, et des avantages/inconvénients de votre approche (**voir barème à la fin**). Notez aussi qu'un plus grand poids est accordé à la présentation, alors c'est important d'avoir une page titre et des figures!

Tel que décrit à travers les pages 11 à 33 du chapitre sur l'indexation du contenu pictural provenant des notes de cours, vous aurez d'abord à implémenter deux fonctions de base qui serviront à transformer les trames d'une séquence vidéo pour simplifier l'analyse des variations de leur contenu en fonction du temps. La première fonction servira à calculer l'histogramme d'une image, et la deuxième servira à détecter les contours/arêtes d'une image par convolution. Vous aurez par la suite à développer et tester votre propre logique de détection de transitions en utilisant les résultats de ces deux fonctions; les deux stratégies présentées dans les notes de cours (et données ici comme pistes de solution) peuvent être fusionnées et

utilisées comme référence si vous êtes à court d'idées. Notez par contre que vous pouvez aussi complètement ignorer les deux fonctions de base, et développer votre algorithme comme bon vous semble. Toutefois, l'implémentation de ces deux fonctions doit quand même se retrouver avec votre rapport lors de la remise (voir le barème pour les détails) – ce sont les deux seuls fichiers de code qui seront corrigés. Le travail demandé est expliqué plus en détail dans les sections qui suivent. Comme d'habitude, vous ne pouvez pas utiliser les fonctions déjà programmées de Matlab ou d'OpenCV dans vos implémentations du calcul d'histogramme ou de la convolution, mais vous pouvez vous en servir pour valider vos résultats.

## Histogrammes

La première fonction de base que vous aurez à compléter est celle qui calcule l'histogramme de couleurs d'une image donnée, et qui possède la signature suivante (dans 'histo.m/cpp') :

```
function [ hist ] = histo( image, N )                                    (Matlab)

cv::Mat histo(const cv::Mat& image, size_t N) { ... }                    (C++)
```

Où '*image*' possède toujours trois canaux (on ignore les images en noir et blanc dans le cadre de ce TP), et où la matrice retournée ('*hist*') est de taille 3xN, et chaque élément qu'elle contient représente le nombre d'occurrences d'une certaine intensité provenant d'un canal, divisé par le nombre total de pixels dans l'image. En d'autres mots, il s'agit de l'histogramme normalisé des trois canaux de l'image traités indépendamment, et qui contient donc uniquement des valeurs en point flottant entre 0 et 1. Le paramètre '*N*' sert à déterminer le niveau de quantification à utiliser dans l'histogramme – vous aurez vous-même à déterminer quelle valeur utiliser pour optimiser votre algorithme plus tard.

Pistes de solution pour détection de transitions : La comparaison d'histogrammes de trames consécutives dans la vidéo (e.g. par distance euclidienne) peut vous donner une idée de la variation de couleur sans se soucier de l'organisation spatiale du contenu. Ceci peut être vu comme un avantage ici, car un objet en mouvement dans la scène ne causera normalement pas de variation très marquée entre deux histogrammes de trames consécutives. Vous pourrez donc profiter de cette propriété pour masquer les fausses transitions détectées à cause du mouvement rapide d'objets dans la scène.

## Convolutions

Vous aurez ensuite à implémenter une fonction qui calcule le résultat de la convolution d'une image couleur avec un noyau 2D quelconque fourni en paramètre. La fonction en question est à compléter dans 'convo.m/cpp', et elle possède la signature suivante :

```
function [ result ] = convo( image, kernel )                            (Matlab)

cv::Mat convo(const cv::Mat& image, const cv::Mat_<T>& kernel) { ... }   (C++)
```

Où l'image donnée en entrée ('*image*') possède encore toujours trois canaux, et où le noyau ('*kernel*') est carré et possède toujours un seul canal. Pour traiter l'image, la convolution devra être appliquée à chaque canal indépendamment, ce qui veut donc dire que la matrice de sortie ('*result*') devra posséder elle aussi trois canaux, et sera de la même taille que l'image d'entrée. Le type de cette matrice de sortie est laissé à votre discrétion, mais notez qu'il doit pouvoir contenir plus de valeurs que le type de l'image d'entrée (ainsi que des valeurs négatives); un type en point flottant est suggéré. Pour gérer le cas spécial où la convolution d'un pixel

a besoin d'accéder à des valeurs à l'extérieur de l'image, vous pouvez les substituer pour celles les plus proches à l'intérieur de l'image, ou bien par des valeurs nulles (zéro).

Pistes de solution pour détection de transitions : La convolution avec les opérateurs (ou noyaux) de Sobel illustrés ci-dessous vous permettra d'obtenir les cartes de gradients en X et en Y de l'image originale.

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

$S_X$ : Noyau Sobel 3x3

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

$S_y$ : Noyau Sobel 3x3

Avec les cartes de gradients en X et en Y (notées $G_X$ et $G_Y$), vous pourrez calculer la norme (ou l'amplitude/la force) du gradient pour chaque pixel (i,j) de l'image à l'aide de la formule suivante :

$$F_G(i,j)=\sqrt{G_x^2(i,j)+G_y^2(i,j)}$$

Par la suite, afin d'effectuer le seuillage d'une carte d'amplitudes (qui devrait toujours posséder à ce moment-là trois canaux) pour obtenir une carte d'arêtes, vous pourrez simplement comparer l'amplitude maximale de chaque pixel (i.e. le max du triplet RGB) à une valeur prédéterminée. La mise en correspondance des cartes d'arêtes ainsi obtenues vous permettra de savoir si les objets principaux de la scène se retrouvent toujours au même endroit, ou s'ils semblent disparaître. Pour minimiser l'effet du bruit lors de cette mise en correspondance, vous pourrez dilater les cartes arêtes à l'aide d'une opération morphologique (e.g. *cv::dilate* ou *imdilate*). Enfin, les ratios de mises en correspondance d'arêtes ($\rho_{in}$ et $\rho_{out}$) pourront être calculés en fonction du temps et utilisés tel que décrits dans les notes de cours; ceux-ci vous permettront de savoir lorsque des objets semblent disparaître (ce qui indiquerait un fondu entre deux plans).

## Détection de transitions dans une séquence vidéo

Tel que brièvement discuté dans les pistes de solution des deux dernières sections, les résultats fournis par les deux fonctions de base peuvent être utilisées dans le but d'obtenir un algorithme de détection de transitions. Leur complémentarité peut être très utile afin d'éliminer des fausses détections causée par une sensibilité trop élevée; ce sera à vous de trouver une bonne stratégie de combinaison, ou encore de développer une toute nouvelle approche plus robuste que celle proposée.

Sur Moodle, vous trouverez une séquence fournie avec l'énoncé ('TP3_video.avi') avec laquelle vous pourrez tester votre algorithme. Notez toutefois que votre stratégie d'analyse devrait pouvoir être appliquée sur n'importe quelle séquence vidéo; on cherche un algorithme possédant une bonne universalité/généralité, ce qui devrait être reflété dans votre rapport lors de la discussion des paramètres choisis.

D'autre part, veuillez joindre à votre rapport tout le code nécessaire pour que votre algorithme soit testé lors de la remise; l'implémentation ne sera pas corrigée, mais celle-ci devrait fonctionner (et elle sera comparée à celle des autres pour éliminer les risques de plagiat).

## Barème : (total sur 20 pts)

1. **Implémentation/fonctionnement** : (sur 5 pts)
    - histo.m/cpp = 2.5 pts
    - convo.m/cpp = 2.5 pts
2. **Rapport** : (sur 15 pts, **~6 à 8 pages** avec page titre et graphiques/images)
    - Présentation des fonctions de base (hist. + conv. + autres, au besoin) = 2.5 pts (~1 page)
    - Présentation logique de détection de transitions = 2.5 pts (~1 page)
    - Présentation des paramètres (rôle et stratégie d'ajustement) = 1 pt (~ ½ page)
    - Discussion avantages/inconvénients pour stratégie proposée = 1 pt (½ à 1 page)
    - Présentation et discussion des résultats de détections pour 'TP3_video.avi' = 2.5 pts (~1 page)
    - Discussion des améliorations possibles, et généralisation pour autres vidéos = 2.5 pts (~½ page)
    - Propreté, formatage, lisibilité = 3 pts

## Références supplémentaires

- Aide-mémoire (« Cheat sheet ») Matlab :
    - http://web.mit.edu/18.06/www/Spring09/matlab-cheatsheet.pdf
- Guide complet Matlab :
    - http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf
- OpenCV
    - http://opencv.org/
    - http://docs.opencv.org/doc/tutorials/tutorials.html