INF4710
Introduction aux technologies multimédia

A2017 - Travail pratique #3
Indexation de fichiers multimédia : décomposition en prises de vue

---

Objectifs :

- Permettre à l'étudiant de se familiariser avec 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 1er décembre, 15h00 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 :

- Le code lui-même n'est pas évalué, mais il doit être remis au COMPLET, question de quand même pouvoir vérifier les cas de plagiat...
- Aucun code source de base vous est fourni pour ce TP; à vous de tout développer!

Présentation

L'objectif de ce travail pratique est d'implémenter une méthode de détection de changements entre scènes dans une vidéo par détection et mise en correspondance d'arêtes. Pour ce faire, il est nécessaire d'extraire l'image de gradient de chacune des trames de la séquence et d'y effectuer un seuillage pour y identifier les arêtes présentes.

Contrairement à ce qui vous était demandé aux deux premiers TPs, pour le TP3, tout le poids de la note est accordé à votre rapport et à vos résultats. Traitez votre rapport comme un compte-rendu d'implémentation avec résultats de tests **que vous enverriez à un client**. Notez toutefois que ce dernier **n'y connaît absolument RIEN en programmation**, alors ne recopiez pas simplement votre code! Vous devrez aussi y discuter vos choix stratégiques, et les avantages/inconvénients de votre méthode (**voir barème à la fin pour tous les détails**).

Tel que décrit à travers les pages 23 à 33 du chapitre sur l'indexation du contenu pictural provenant des notes de cours, vous aurez d'abord à implémenter une fonction de convolution d'image à l'aide d'un filtre de Sobel. Par la suite, vous aurez à développer une fonction de seuillage

des images de gradient obtenues pour y détecter les arêtes principales de la scène. Ces arêtes devront ensuite être dilatées à l'aide d'une opération morphologique. Finalement, vous aurez à calculer les ratios d'arêtes partagées par des paires d'images à travers une séquence vidéo (ρ_{in} et ρ_{out}), et déterminer quelle stratégie utiliser pour maximiser vos chances de détecter les changements de scènes dans n'importe quelle vidéo. Le travail demandé pour chacune de ces étapes est décrit dans les sections suivantes.

Convolution d'images

Vous devez d'abord implémenter une fonction qui calcule **la carte des forces de gradient normalisée** à l'intérieur d'une image couleur. (Remarque: on ne traite pas d'images en tons de gris dans ce TP.) Pour ce faire, vous devrez implémenter une sous-fonction qui calcule la réponse de convolution d'une image à l'aide d'un noyau quelconque fourni en paramètre (celui-ci sera le noyau de Sobel en X ou en Y). Vous devez suivre la stratégie énoncée aux pages 53 à 55 du même chapitre des notes de cours dans le but d'obtenir une carte des forces calculées en X et en Y pour chaque canal de l'image RGB fournie à la fonction. La sous-fonction qui calcule la réponse de convolution devrait donc être appelée six fois pour traiter chaque image. Les noyaux de Sobel à utiliser sont de taille 3x3 et sont décrits ci-dessous :

-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

La formule à utiliser pour obtenir la norme (i.e. la « force ») du gradient en un point (i,j) dans l'image (et pour un seul canal à la fois) à partir des deux réponses de gradients en X et en Y est la suivante :

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

...où G_X et G_Y sont les réponses de gradient en X et Y pour le pixel (i,j).

Image couleur originale (entrée)

Carte des forces de gradient (sortie)

La visualisation de la carte de forces de gradient produite à cette étape devrait vous donner une image largement noire, mais avec des couleurs visibles près des contours des objets (voir l'exemple ci-dessus). Notez par contre que la normalisation **min-max** ou **0-1** (i.e. où la valeur minimale devient 0, et la valeur maximale 1) de la matrice obtenue est nécessaire pour visualiser ses valeurs. Allez voir https://en.wikipedia.org/wiki/Feature_scaling pour une description!

Bien entendu, nous nous attendons à ce que vous implémentiez ces deux fonctions au complet, sans utiliser les fonctions de convolution déjà disponibles dans Matlab/OpenCV – ça fait partie de l'apprentissage! Ceux qui utilisent les fonctions « built-in » de leur environnement seront pénalisés dans le rapport. Vous pouvez toutefois tester le fonctionnement de vos méthodes en comparant manuellement vos résultats à ceux de 'imgradient' (Matlab), ou à ceux de cet exemple (OpenCV). Comme d'habitude, faites attention aux types/intervalles lors de vos comparaisons!

Seuillage des images de gradients

Dans cette partie, vous devez vous-même choisir une stratégie pour transformer l'image des forces de gradient (qui devrait encore contenir trois canaux) en une image binaire de un canal contenant une valeur non-nulle uniquement là où une arête « **importante** » est présente. Il est clair qu'il faut utiliser une opération de seuillage pour y arriver, mais c'est à vous de choisir l'opération ainsi que le (ou les) seuils à utiliser, au besoin. Dans votre rapport, décrivez bien votre approche ainsi que les paramètres configurables qu'elle introduit dans votre méthode finale. Ultimement, vous devriez être capable d'obtenir une transformation semblable à celle de l'image ci-dessous.

Carte des forces de gradient (entrée)

Image binaire des arêtes détectées (sortie)

Dilatation des arêtes trouvées

Pour dilater les arêtes des images binaires obtenues précédemment, vous devez implémenter une opération morphologique recevant l'image à traiter (binaire, d'un seul canal), ainsi que la taille du noyau de dilatation à utiliser (typiquement impaire), qui retourne une nouvelle image binaire d'un seul canal. L'opération de dilatation est relativement simple : vous n'avez qu'à

parcourir tous les pixels de l'image, et pour chaque pixel ayant une valeur non nulle, vous devez copier cette valeur à tous les pixels présent dans un voisinage de taille N centré sur le pixel original (ici, N est la taille du noyau de dilatation). Cette approche est en fait une simplification de celle expliquée dans le cours (i.e. dilatation avec un cercle ou un losange). Le résultat que vous devriez obtenir, une fois cette opération complétée, devrait ressembler au suivant (pour N=5) :

Image binaire originale

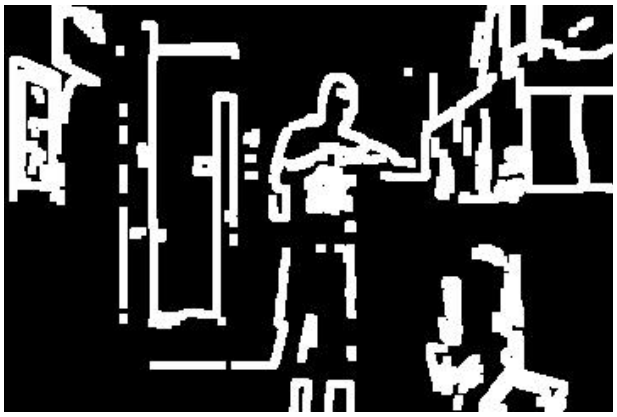

Image binaire dilatée

Encore une fois, n'utilisez pas les fonctions « built-in » de votre environnement (e.g. cv::dilate ou imdilate) dans cette fonction, sous peine de pénalité! Vous pouvez par contre les utiliser pour comparer avec leur résultat...

Calcul du ratio des arêtes partagées

Une fois que vous disposez d'images d'arêtes dilatées et non dilatées, vous pouvez enfin calculer les valeurs de ρ_{in} et ρ_{out} en utilisant les formules suivantes (tirées des notes de cours) :

$$\rho_{in} = 1 - \frac{\sum_{x,y} D(x,y,t)E(x,y,t+1)}{\sum_{x,y} E(x,y,t+1)}$$

$$\rho_{out} = 1 - \frac{\sum_{x,y} E(x,y,t)D(x,y,t+1)}{\sum_{x,y} E(x,y,t)}$$

...où E est une image binaire d'arêtes non dilatée, et D est une image binaire d'arêtes dilatée. Vous devez implémenter le calcul de ces deux métriques pour les images binaires obtenues dans les étapes précédentes.

Détection de transitions dans une séquence vidéo

Pour cette partie, vous devez créer une fonction d'analyse capable de traiter la séquence vidéo fournie avec l'énoncé ('INF4710_A2017_TP3_video.avi' disponible sur Moodle) pour y détecter les transitions (**coupures et fondus**). Votre analyse devra faire usage de toutes les fonction implémentées précédemment. La stratégie d'analyse à suivre est libre à vous; vous avez quelques exemples dans les notes de cours de ce qui peut être fait avec ρ_{in} *et* ρ_{out} en fonction du temps, mais vous pouvez décider d'y ajouter d'autres composantes (au besoin). Notez que votre stratégie d'analyse devrait idéalement pouvoir être appliquée à n'importe quelle séquence, et pas seulement celle fournie avec l'énoncé! Bref, n'allez pas 'hard-coder' des indices propres à la séquence de test dans votre analyse. Enfin, gardez en tête que votre méthode serait « livrée » à un client suite à votre rapport, alors c'est important de bien justifier ce que vous avez développé!

D'autre part, dans votre rapport, vous devez (minimalement) présenter un graphique des valeurs de ρ_{in} *et* ρ_{out} en fonction du temps pour 'TP3_video.avi', et les numéros de trames où des changements ont pu être détectés (utilisez par exemple des barres verticales dans votre graphe). Notez qu'il est possible que vous n'arriviez pas à détecter toutes les transitions de la séquence de test, ou que vous en détectiez trop, **si votre stratégie est trop simple, ou si vos paramètres sont mal choisis.** Dans ce cas, vous devez **obligatoirement** en détailler la cause dans votre rapport (voir le barème).

Notez que pour éviter d'avoir à relancer l'analyse de la séquence vidéo pendant que vous testez/ajustez votre algorithme de détection de transitions, vous pouvez sauvegarder sur le disque toutes les valeurs de ρ_{in} *et* ρ_{out} obtenues lors d'une première analyse, et les recharger par la suite. Cela ne sera probablement pas nécessaire en C++, mais en Matlab, ça pourrait vous sauver pas mal de temps de calcul...

Barème : (rapport sur 20 pts)

- Présentation du problème, survol de la méthode (i.e. détection par arêtes) = 2 pts (~ ½ à 1 page)
- Présentation modules de convolutions, seuillage, dilatation, calcul ratio = 4 pts (~1 à 2 pages)
  - Expliquez tout dans vos mots; ne faites pas simplement recopier des notes de cours!
  - Si votre méthode inclut des paramètres qui doivent être ajustés, vous devez spécifier leur rôle dans cette section de votre rapport!
- Discussion avantages/inconvénients de la méthode = 3 pts (~ ½ à 1 page)
  - Comparez ici votre implémentation à la méthode par histogrammes vue dans le cours! (expliquez dans quels cas une méthode est meilleure que l'autre...)
- Présentation résultats de détections pour la séquence de test fournie = 4 pts (~1 page)
  - Illustrez ρ_{in} et ρ_{out} avec un graphe, et notez les numéros de trames avec transitions détectées.
- Discussion amélioration possibles = 3 pts (~½ à 1 page)
  - Si vous obtenez des faux positifs/négatifs dans vos détections, détaillez la/les cause(s) derrière ces mauvaises détections, et proposez une approche pour arriver à de meilleurs résultats (sans l'implémenter).
  - Si vos résultats sont bons (bonnes détections à peu près au bon moment pour toutes les transitions, et sans détections inutiles), décrivez uniquement une amélioration qui pourrait être apportée à votre méthode pour la rendre encore plus efficace (sans l'implémenter).
- Propreté, formatage, lisibilité = 4 pts
  - Page titre, numéros de page, langue — prenez le tout au sérieux!
- Autres pénalités possibles = -4 pts max
  - -2 pts si vous utilisez une fonction de convolution d'OpenCV (e.g. filter2D, Sobel, Scharr, ...) ou de Matlab (e.g. conv, conv2, edge, imgradient, ...) dans votre code
  - -2 pts si vous utilisez une opération morphologique d'OpenCV (e.g. dilate) ou de Matlab (e.g. imdilate) dans votre code