ALVES Vincent
HYACINTHE Clément
TESTIER Marc-Antoine

Projet d'informatique différencié: Réalisation d'un panorama d'images

25 novembre 2013

Table des matières

1 Objectif

Dans ce second livrable, nous allons parler des fonctions de traitement d'image demandées et expliquer leur implémentation en C.
Nous allons également proposer une réflexion sur comment automatiser la création de panorama.

2 Fonctions de prétraitement

Comme demandé par le client, nous avons codé quelques fonctions de prétraitement d'image, afin de les utiliser dans le programme final et d'améliorer notre maîtrise du language C.
Toutes ces fonctions ont été conçues pour travailler avec le format .ppm(portable pixmap), qui est un format "raw" d'image, sans aucune compression.
De ce fait, les fichiers résultant sont très lourds, mais faciles à modifier, car il suffit de les ouvrir dans un éditeur de texte pour directement voir les codes RGB de tous les pixels de l'image.

Pour chaque fonction de pré-traitement, nous avons stocké les données de l'image dans un tableau de Pixels de la dimension de l'image. Chaque case du tableau contient 3 données : les couleurs Red, Green et Blue. Cela nous permet une manipulation facile du tableau.

2.1 Fonction de niveau de gris

Le passage d'une image en couleur vers une image en niveau de gris est une fonction simple à réaliser, chaque pixel d'une image étant décomposé en 3 couleurs : rouge, vert bleu, il suffit d'appliquer une formule simple pour obtenir un niveau de gris :

$$Gris = 0{,}299.Rouge + 0{,}587.Vert + 0{,}114.Bleu$$

On applique alors cette formule à chaque Pixel, en remplaçant les 3 éléments du Pixel (Rouge, Vert, Bleu) par le résultat de l'équation.
Pour cette fonction, nous regardons chaque case du tableau à l'aide de 2 boucles 'for', calculons le nouveau pixel à l'aide de l'équation et écrivons 3 fois le résultat (une fois pour chaque 'couleurs') dans un nouveau fichier récoltant la nouvelle image.

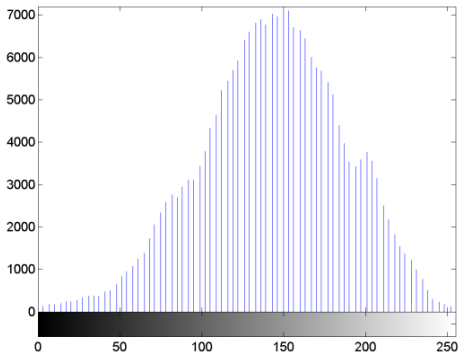

2.2 Fonction de binarisation

La fonction de binarisation s'applique sur une image en niveau de gris et consiste à rendre l'image uniquement en noir et blanc. L'image étant en gris, chaque « couleur » d'un pixel possède la même valeur, on regarde si cette valeur est supérieur ou inférieur à un seuil, donné au préalable par l'utilisateur, puis on transforme le pixel en noir ou en blanc.

Dans notre programme, nous recevons une image en gris et un seuil, on regarde alors pour chaque Pixel s'il est supérieur ou inférieur au seuil.

2.3 Fonction d'histogramme

L'histogramme en niveaux de gris d'une image consiste à indiquer, pour chaque valeur entre le noir et le blanc, combien de pixels de cette valeur existent. On peut alors mettre toutes ces données dans un tableau et créer l'histogramme.



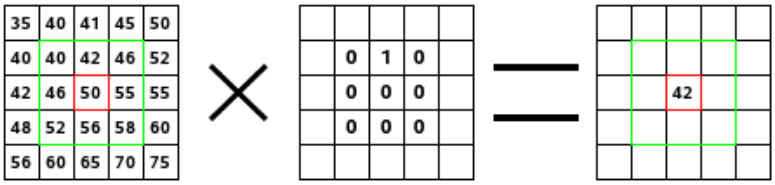
Dans notre programme, nous créeons un tableau unidimensionnel de 256 cases, puis regardons le tableau contenant l'image.
A chaque pixel, nous prenons l'intensité de la couleur grise de ce pixel(de 0 à 255) et incrémentons la case du tableau correspondante.

Les données sont alors placées dans un fichier "gris" contenant 256 lignes avec, pour chaque ligne, la valeur de l'intensité suivie du nombre d'occurrence de cette couleur dans l'image.

2.4 Fonction de convolution

La convolution consiste à transformer un pixel d'une image en se basant sur les pixels environnants. On utilise pour cela des masques de convolution. Un masque de convolution est un tableau regroupant des coefficients permettant de calculer le nouveau pixel.

Par exemple :



Pour notre programme, nous utilisons un fichier contenant le masque, que nous récupérons. Il nous reste alors à l'appliquer sur chaque case du tableau contenant l'image.
On multiplie les cases autour du Pixel initial par la valeur du masque correspondant, puis on additionne toutes les valeurs. Pour éviter tout dépassement de valeur, on divise le résultat par la somme des valeurs du masque.
(dans le cas plus haut, 1) Si cette somme vaut 0, on la met à 1.

Quand le pixel est sur un bord, une partie du masque est en dehors de l'image, il existe alors plusieurs possibilités pour gérer ce cas :

1. Nous pouvons étendre l'image, c'est à dire ne pas prendre en compte la partie du noyau qui n'est pas dans l'image et remplacer la ligne libérée par une extension de la ligne suivante.
2. Nous pouvons rogner l'image, c'est à dire supprimer les pixels du bord de l'image et ainsi réduire l'image.
3. Nous pouvons ignorer les pixels étant sur le bord, c'est à dire appliquer la convolution uniquement sur les pixels du centre, et recopier les pixels en bordure tel quels.
4. Ou alors, nous pouvons enrouler l'image, c'est à dire appliquer la partie du masque qui n'est pas dans l'image sur le bord opposé à l'image.

Dans notre cas, nous avons choisis d'enrouler l'image et d'utiliser des masques de taille 3x3.

Sur cette image, nous avons appliqué le masque :

$$M = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

Qui permet de faire ressortir les bords de l'image.

2.5 Fonctions d'érosion et dilatation

La dilatation consiste à éliminer les pixels noirs isolés d'une image. Pour cela, on parcourt tout le tableau de pixel et on regarde pour chaque pixel, les pixels environnants et on remplace la valeur du pixel initial par la valeur du pixel le plus grand l'entourant (255 étant blanc pour les fichiers ppm).
Dans notre cas, nous regardons pour chaque pixel, les 8 pixels environnants, mais cette opération peut aussi être fait sur uniquement les pixels Nord, Sud, Est et Ouest ou même sur une plus grande zone.

L'érosion est le principe inverse, il consiste à éliminer les pixels blancs isolés selon le même principe, on parcourt le tableau et on remplace la valeur de chaque pixel par la valeur du pixel le plus petit l'entourant (0 étant le noir).

Dans la morphologie d'image, nous utilisons plutôt la dilatation et l'érosion l'une après l'autre, c'est pourquoi nous avons décidé d'implémenter deux nouvelles fonctions d'ouverture et de fermeture.
L'ouverture consiste à réaliser une érosion puis une dilatation.


Tandis que la fermeture consiste en l'inverse, c'est à dire en une dilatation puis une érosion.


2.6 Améliorations possibles

Nous avons réalisé toutes ces fonctions, mais nous pouvons toujours ajouter quelques détails, qui seront implémentés au prochain livrable.
On pourra par exemple choisir de laisser à l'ordinateur le choix du seuil de binarisation selon les niveaux de gris, afin d'avoir par exemple une moitié des pixels noir et une autre moitié blanche.
On pourra aussi créer un graphique pour l'histogramme des niveaux de gris, afin d'avoir un résultat plus visuel.
En ce qui concerne la convolution, on pourrait proposer à l'utilisateur de rentrer une matrice de convolution dans la ligne de commande. On pourrait aussi proposer l'utilisation des matrices de taille 5x5.
Pour l'érosion et la dilatation, on pourrait proposer à l'utilisateur de rentrer le nombre d'érosions et de dilatations qu'il souhaite faire afin qu'il n'aie pas à répéter la même opération. On pourra aussi laisser à l'utilisateur le choix de la taille de l'érosion ou de la dilatation.

3 Comment automatiser la création de panorama ?

Lors du dernier livrable, nous avons expliqué la meilleure méthode pour créer un panorama, la matrice homographique.
Cependant, cet outil, bien que très puissant, demande à l'utilisateur de pointer lui-même les points en commun entre les deux images.
Pour automatiser ce processus, il nous faut utiliser deux outils qui se complémentent : Un algorithme de recherche de points d'intérêt, et l'algorithme RANSAC. Nous avions déjà recherché et mentionné deux détecteurs : Moravec et Harris. Moravec étant assez limité, nous avons essayé de trouver d'autres détecteurs qui seraient peut-être plus simples à implémenter que Harris.

3.1 Détecteur de Harris

3.1.1 Rappel des limites du détecteur de Moravec

Harris et Stephen ont identifié certaines limitations du détecteur de Moravec et, en les corrigeant, en ont déduit un détecteur de coins très populaire : le détecteur de Harris.
Les limitations du détecteur de Moravec prises en compte étaient :

1. La réponse du détecteur est anisotropique en raison du caractère discret des directions de changement que l'on peut effectuer (des pas de 45 degrés). Pour améliorer cet aspect, il sufdit de considérer le developpement de Taylor de la fonction d'intensité I au voisinage du pixel (u, v) :

   $$I(u + x, v + y) \approx I(u, v) + I_x(u, v)x + I_y(u, v)y$$

   D'ou

   $$S(x, y) \approx \sum_u \sum_v w(u, v)\ (I_x(u, v)x + I_y(u, v)y)^2$$

2. La réponse du détecteur de Moravec est bruitée en raison du voisinage considéré. Le filtre w utilisé est en effet binaire (valeur 0 ou 1) et est appliqué sur un voisinage rectangulaire. Pour améliorer cela, Harris et Stephen proposent d'utiliser un filtre Gaussien :

   $$w(u, v) = exp - (u^2 + v^2)/2\theta^2$$

3. Enfin, le détecteur de Moravec repond de manière trop forte aux contours en raison du fait que seul le minimum de E est pris en compte en chaque pixel. Pour prendre en compte le comportement général de la fonction E localement, on écrit :

   $$E(x, y) = (x, y)M(x, y)^t$$

   Avec M une matrice caractérisant le comportement local de la fonction E.

   $$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

   Les valeurs propres de cette matrice correspondant aux courbures principales associées à E.
   Si les deux courbures sont de faibles valeurs, alors la région considérée a une intensité approximativement constante.
   Si une des courbures est de forte valeur alors que l'autre est de faible valeur alors la région contient un contour.

Si les deux courbures sont de fortes valeurs alors l'intensite varit fortement dans toutes les directions, ce qui caractérise un coin.
Par voie de conséquence, Harris et Stephen propose l'opérateur suivant pour détecter les coins dans une image :

$$R = Det(M) - kTrace(M)^2$$
$$\text{avec } Det(M) = AB - C^2 \text{ et } Trace(M) = A + B.$$

3.1.2 Comment implémenter le détecteur de Harris ?

On prend une image $I(x, y)$ dans laquelle nous commençons par prendre les dérivés de l'image en ligne et en colonne :

$$I_x(x, y) = \delta I/\delta x(x, y)$$
$$I_y(x, y) = \delta I/\delta y(x, y)$$

Pour les calculs, nous aurons besoin de trois matrices : Les dérivées au carré, et leurs multiplications.

$$A(x, y) = {I_x}^2(x, y)$$
$$B(x, y) = {I_y}^2(x, y)$$
$$C(x, y) = I_x(x, y).I_y(x, y)$$

Ces trois matrices doivent être lissées par un filtre passe bas gaussien, qui est une des matrices de convolution que nous avons étudiées :

$$k(x, y) = k(x, y) * Hgauss$$
Avec k = A, B ou C et Hgauss le filtre gaussien.

Nous pouvons maintenant calculer la force de coin Q pour chaque pixel. elle se calcule ainsi :

$$Q(x, y) = (AB - C)^2 - \alpha.(A + B)^2$$

Ou α est la sensibilité du détecteur. La force de coin Q retourne des fortes valeurs à la détection d'un coin.
Plus α est grand, plus la sensibilité est faible et moins le nombre de coins sera grand. Typiquement, α est compris entre 0.04 et 0.06, mais ne dépasse pas 0.25.
On effectue par la suite un seuillage pour éliminer les faibles valeurs de la force de coin :

$$Q(x, y) < th = 0$$

Généralement, ce seuil (th) est entre 10^4 et 10^7.
On peut ensuite ranger les coins détectés par ordre décroissant, et procéder à une détection de maxima locaux pour éliminer les coins trop proches.
Les maxima locaux doivent être faits à partir du coin de plus fort valeur, et éliminer les coins de faible valeur.

3.2 Le détecteur SUSAN

SUSAN signifie Smallest Univalue Segment Assimilating Nucleus. SUSAN utilise un masque en forme de disque centré sur le pixel à analyser. Pour chaque pixel p compris dans le masque, on effectue la comparaison avec le centre pn du masque grâce à la fonction :

$$c(p) = e^{\frac{-(I(p)-I(pn))^6}{t}}$$

Ou t est le rayon du disque. Puis, nous calculons la somme des comparaisons du masque :

$$n(pn) = \sum_{\forall p} c(p)$$

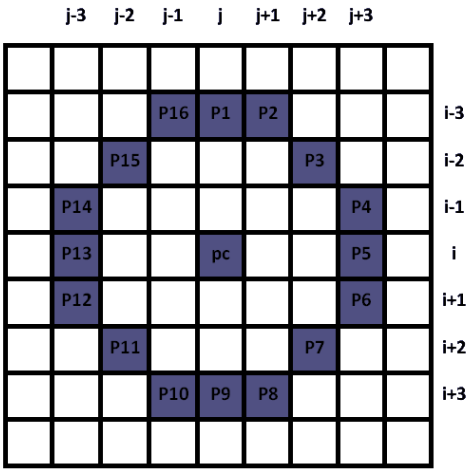
Nous utilisons ensuite la comparaison suivante :

$$R(pn) = \begin{cases} g - n(pn) \, si \, n(pn) < g \\ 0 \qquad\qquad\qquad sinon \end{cases} \tag{1}$$

g est appelé un seuil géometrique.
Si g est assez grand, SUSAN devient un detecteur de contour.

3.3 Le détecteur FAST

FAST signifie Features from Accelerated Segment Test. FAST utilise un cercle autour du pixel central comme base de calcul.
Prenons pour exemple les pixels suivant, ou pc est le pixel central.

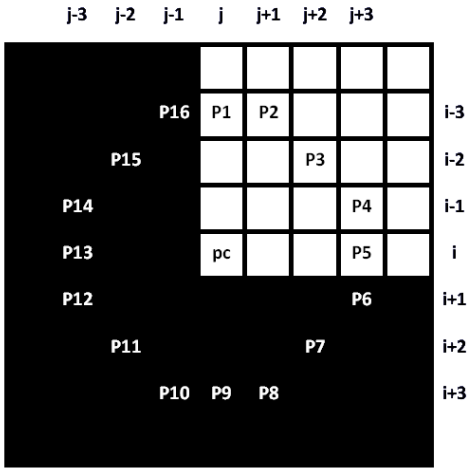


Nous prenons un cercle de taille 3 pour obtenir le vecteur de 16 pixels représenté ci-dessous.

Une fois ce vecteur obtenu, nous allons, pour chaque pixel du vecteur, faire la différence entre lui meme et le pixel central.
On choisit ensuite un seuil haut et un seuil bas autour de la valeur du pixel central. On va definir trois états selon le résultat de la différence :

1. -1 si le résultat est inférieur au seuil bas
2. 0 si le résultat est entre le seuil bas et le seuil haut
3. +1 si le résultat est supérieur au seuil haut

Afin de comprendre l'algorithme, prenons l'exemple suivant :



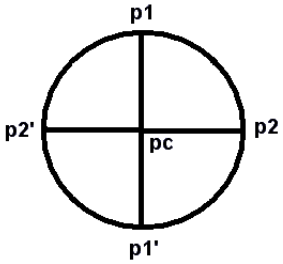
Si on applique l'algorithme précédent, nous obtenons le vecteur suivant :

Nous pouvons remarquer que pour détecter que *pc* est un coin, une succesion de onze 1 doit etre contenue dans le vecteur. Il en va de meme pour une suite de onze -1.

3.4 Le détecteur de Trajkovic

3.4.1 Méthode 4 voisins

Cette détection de coin utilise un cercle pour trouver la variation d'intensité entre les pixels. Prenons le cercle suivant, ou nous voulons déterminer si *pc* est un point d'intérêt :



Dans ce cercle nous faisons le calcul suivant :

$$d(x,y) = min(r1, r2) \text{ Ou :}$$
$$r1 = (p1 - pc)^2 + (p1' - pc)^2$$
$$r2 = (p2 - pc)^2 + (p2' - pc)^2$$

Les coordonnées x et y correspondent au centre du cercle pc. Ce calcul à pour effet de calculer les coins potentiels de l'image.
Les pixels possédant une valeur assez importante sont concidérés comme coins potentiels.
Pour chaque coin potentiel, nous effectuons le calcul de la variation minimum dans toutes les directions grâce à l'algorithme suivant :

$$m(x,y) = \begin{cases} C - \dfrac{B^2}{A} \, si \, B < 0 \, et \, (A+B) > 0 \\ d(x,y) \qquad\qquad\qquad\qquad sinon \end{cases} \tag{2}$$

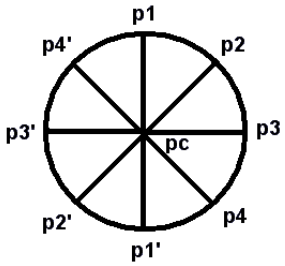
avec :

$$B1 = (p2 - p1)(p1 - pc) + (p2' - p1')(p1' - pc)$$
$$B2 = (p2 - p1')(p1' - pc) + (p2' - p1)(p1 - pc)$$
$$B = min(B1, B2)$$
$$C = r1$$
$$A = r2 - r1 - 2B$$

Nous supprimons ensuite tous les points de la matrice m inférieurs à un seuil T, puis une détection des maxima locaux est appliquée.

3.4.2 Méthode 8 voisins

Une amélioration de la première version peut être de prendre 8 voisins au lieu de 4. Cette augmentation du nombre de variations sur le cercle permet de supprimer les erreurs dues aux bruits.



On a alors $r1, r2, r3, r4$ au lieu de $r1, r2$ et on adapte les équations en conséquence.

3.5 Notre choix de détecteur

Trajkovic est un détecteur avec des calculs simples à implémenter. Cependant, il est plus imprécis que Harris, qui serait un peu plus complexe à faire fonctionner.

Cependant, la moitié du travail sur Harris est déjà fait, grâce à notre fonction de prétraitement sur la convolution. Il suffirait juste d'implémenter une dérivation de l'image en ligne et en colonne.

SUSAN a beaucoup de calculs, ce qui résulte en une complexité plus élevée. FAST serait aussi un détecteur viable à implémenter, même s'il souffre des mêmes problèmes d'imprécision que Trajkovic.

Nous hésitons actuellement donc entre Trajkovic et Harris.

3.6 RANSAC

RANSAC est une abréviation pour "RANdom SAmple Consensus".
Il s'agit d'une méthode itérative pour estimer les paramètres d'un modèle mathématique à partir d'un ensemble de données observées qui contient des valeurs aberrantes (« outliers »). Il s'agit d'un algorithme non-déterministe dans le sens où il produit un résultat correct avec une certaine probabilité seulement, celle-ci augmentant à mesure que le nombre d'itérations est grand.

RANSAC atteint son objectif en sélectionnant itérativement un sous-ensemble aléatoire des données d'origine. Ces données sont d'hypothétiques inliers et cette hypothèse est ensuite testée comme suit :

1. Un modèle est ajusté aux inliers hypothétiques, c'est-à-dire que tous les paramètres libres du modèle sont estimés à partir de cet
2. Toutes les autres données sont ensuite testées sur le modèle précédemment estimé. Si un point correspond bien au modèle estimé alors il est considéré comme un inlier candidat.
3. Le modèle estimé est considéré comme correct si suffisamment de points ont été classés comme inliers candidats.
4. Le modèle est ré-estimé à partir de cet ensemble des inliers candidats.
5. Finalement, le modèle est évalué par une estimation de l'erreur des inliers par rapport au modèle.

Cette procédure est répétée un nombre fixe de fois, chaque fois produisant soit un modèle qui est rejeté parce que trop peu de points sont classés comme inliers, soit un modèle réajusté et une mesure d'erreur correspondante. Dans ce dernier cas, on conserve le modèle réévalué si son erreur est plus faible que le modèle précédent.

Appliqué au calcul d'homographies, RANSAC prend à chaque itération 4 correspondances(donc 2 paires de points d'intérêt) au hasard et calcule une homographie H. Ensuite, chaque nouvelle paire de points est classée par rapport à sa concurrence à H. Quand toutes les itérations ont eu lieu, on choisit celle avec le plus de paires compatibles. On recalcule alors H en fonction de toutes les paires considérées comme compatibles dans cette itération.

4 Avancement actuel du programme

Les fonctions de prétraitement sont toutes terminées.
Le parseur n'est cependant pas tout à fait opérationnel.
Si la première option est -i, le parseur va effectuer la ligne de commande et indiquer une erreur dans la console, mais il va bien créer les fichiers demandés avec le nom spécifié après chaque option.
Par exemple :

./panorama -i gris2.ppm -o toto.ppm -g

Le parseur ne fonctionne pas encore avec plusieurs fichiers d'entrées et de sorties.

Suite à ces problèmes sur un élément important du programme, nous avons ajouté à ce livrable deux versions du programme : Une avec le menu/parseur et un Makefile(Dossier Final), et une deuxième permettant de tester l'ensemble des fonctions de prétraitement sans problèmes dus au parseur(Dossier TestImage).

5 Bibliographie

Automatic Panoramic Image Stitching using Invariant Features - Matthew Brown et David G. Lowe
http://www.cs.ubc.ca/~lowe/papers/07brown.pdf

Homography Estimation - Elan Dubrofsky
https://www.cs.ubc.ca/grads/resources/thesis/May09/Dubrofsky_Elan.pdf

CS 195-G : Automated Panorama Stitching - Evan Wallace
http://cs.brown.edu/courses/csci1950-g/results/proj6/edwallac/

Project 4 : Auto Stitching Photo Mosaics - William Wedler
http://www.cs.cmu.edu/afs/andrew/scs/cs/15-463/f07/proj4/www/wwedler/

Détection de points d'interêt
http://devernay.free.fr/cours/vision/pdf/c4.pdf

ProjectiveHomography Class Reference
http://www.clemson.edu/ces/crb/students/vilas/projects/cvutils/docs/Homography/htm/classProjectiveHomography.htm

Documentation GIMP : Matrices de Convolution
http://docs.gimp.org/fr/plug-in-convmatrix.html

Opérations morphologiques de base : dilatation, érosion, ouverture et fermeture binaires
http://dpt-info.u-strasbg.fr/~cronse/TIDOC/MM/deof.html