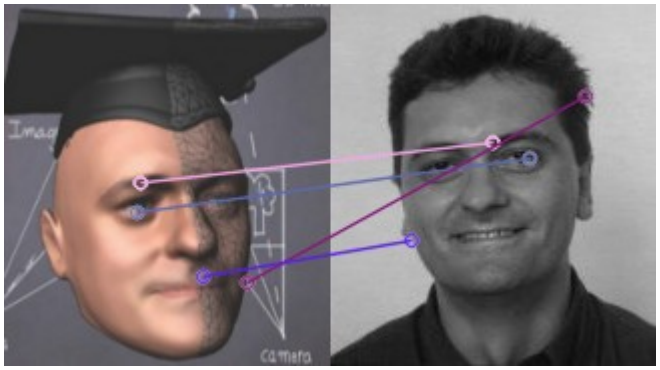


Rapport de Stage

Mise en correspondance de points d'intérêt dans des séries temporelles d'images issues d'observatoires photographiques du paysage

Par

Veysseire Daniel

Résumé du stage

La recherche et la mise en correspondance de point d'intérêts sont des méthodes utilisées dans le domaine de la vision par ordinateur et qui ont différents champs d'application, comme la « rephotography » (terme anglais), qui consiste a prendre deux photographies du même endroit avec une longue période entre les deux clichés.

Cette méthode est aussi utilisé pour la reconstruction 3D dans le cas de la mise en correspondance stéréoscopique. Elle peut aussi être utilisé pour le suivi de déplacement d'objets en mouvement ou détection de mouvement, ou pour la reconnaissance faciale.

Illustration 1: un exemple de rephotography

Illustration 2: un exemple de reconnaissance faciale

Le domaine d'application privilégié ici s' apparente à de la rephotography. Il s'agit de prendre deux photos du même paysage à plusieurs mois voir plusieurs années d'intervalle. La difficulté viens du décalage entre les deux images. Nous cherchons ici à représenter et quantifier ce décalage afin de recaler les photos par la suite.

Le travail présenté dans ce rapport a consisté à faire une recherche sur les différents détecteurs, descripteurs/extracteur et appariеur de points d'intérêts présent dans la bibliothèque C++ OpenCV et dans le logiciel Matlab, leurs fonctionnements et leur caractéristiques, puis d'évaluer leurs performances par des tests sur une série d'images.

Il a aussi consisté à la réalisation d'une petite interface graphique pour mieux visualiser les résultats.

Remerciements

Je remercie chaleureusement monsieur Alain Crouzil pour m'avoir pris en stage et accordé sa confiance, pour le café en capsule offert à ses frais, pour les gâteaux libanais qu'il a rapporté de Beyrouth, pour le temps qu'il m'a consacré autant sur le stage en lui même que sur le fonctionnement de l'IRIT et de la recherche, pour la paperasse à remplir et pour son humour et sa bonne humeur.

Je remercie aussi Monsieur Christophe Collet pour son aide à la résolution de problèmes d'ordre logiciel, notamment pour l'utilisation d' OpenCV en compilation avec g++ en ligne de commande, son aide pour les problèmes graphiques (problème avec Xorg ou du même genre), problèmes réseaux, problèmes de licence Matlab, et mise à jour des différents logiciels.

Je remercie Monsieur Denis Kouamé pour les connaissances de base qu'il m'a prodigué en analyse et en traitement d'image dans le cadre de l'option IATI que j'ai suivi. Ces connaissances m'ont été utile pour la compréhension de notions de base nécessaire à ce stage.

Je remercie aussi l'équipe TCI de m'avoir accepté parmi eux et de leurs échanges intéressant et instructif.

Leur aide et leur investissement ont été très bénéfiques non seulement pour la réalisation de ce stage, mais aussi pour la découverte du fonctionnement de l'IRIT et l'approfondissement de mes connaissances dans le domaine de l'image et de la vision.

Table des matières

Cadre du stage

Présentation de l'IRIT

L' IRIT (Institut de Recherche en Informatique de Toulouse) est une unité de recherche fondé en 1990 en partenariat entre l'Université Paul Sabatier de Toulouse (UT3), le CNRS, l'ENSEEIHT, l'institut national polytechnique de Toulouse et l'Université des Sciences Sociales de Toulouse (UT1) devenue Université Toulouse 1 Capitole en 2009. Il est issu de la fusion de 2 URA du CNRS et de l'Université Paul Sabatier, et du CICT de l'université Paul Sabatier.

Il a accueilli des équipes provenant de divers laboratoires des universités toulousaines, et l'Université Toulouse le Mirail a rejoint les tutelles initiales.

Il abrite environ 695 personnes dont :

- 255 enseignants chercheurs
- 28 chercheurs
- 264 doctorants
- 13 post-doctorants
- 42 personnels administratifs et techniques
- quelques stagiaires (moi!)

L'IRIT est la plus importante UMR CNRS en terme de nombre de chercheurs,

ainsi que le plus grand institut de recherche français en informatique.

Les recherches sont actuellement axés autour de sept thèmes :

- Analyse et synthèse de l'information (équipes SAMoVA, SC, TCI et VORTEX)
- Indexation et recherche d'informations (PYRAMIDE et SIG)
- Interaction, autonomie, dialogue et coopération (IC3, ELIPSE et SMAC)
- Raisonnement et décision (LILaC et ADRIA)
- Modélisation, algorithmes et calcul haute performance(APO)
- Architecture, systèmes et réseaux (IRT, SEPIA, SIERA, T2RS et TRACES)
- Sûreté de developpement du logiciel (ACADIE, ICS et MACAO)

Les recherches s'orientent selon 4 axes stratégiques :

- Informations pour la santé
- Masses de données et calcul
- Systèmes socio-techniques ambiant
- Systèmes embarqués critiques

Ce stage s'est déroulé au sein de l'équipe TCI.

Présentation de l'équipe TCI

Le thème abordé par l'équipe TCI est donc l' Analyse et synthèse de l'information.

TCI signifie Traitement et Compréhension d'Images. Le domaine de recherche de l'équipe est le traitement et l'interprétation de données provenant d'un ou plusieurs systèmes d'acquisition d'images. Les travaux menés dans l'équipe couvrent trois domaines: la vision par ordinateur, l'imagerie médicale et la langue des signes.

La vision par ordinateur est la discipline qui développe les bases théoriques et algorithmiques grâce auxquelles des informations sur le monde 3D peuvent être automatiquement extraites et analysées à partir d'une ou de plusieurs images numériques et d'un ordinateur. L'équipe TCI s'intéresse plus particulièrement au problème de la reconstruction d'une scène volumique fixe observée à partir d'un ou de plusieurs points de vue. D'une part, cela consiste à étudier les relations géométriques et photométriques spécifiques au processus de formation des images et aux connaissances préalables sur la scène observée. D'autre part, il s'agit de réaliser un appariement dense entre couples d'images stéréoscopiques, à partir de ces relations.

L'imagerie médicale occupe une place centrale dans les techniques d'investigation en santé et en médecine et nécessite de prendre en compte toute la chaîne de production et d'interprétation des images. La recherche de l'équipe TCI concerne d'une part les développements méthodologiques en traitement du signal et de l'image applicables ou extensibles à l'ensemble des principales modalités: elle porte sur l'amélioration de la lisibilité et la caractérisation quantitative des images. TCI étudie plus spécifiquement les problématiques et les applications inhérentes aux modalités d'imagerie médicale, l'imagerie ultrasonore, l'imagerie de Tomographie Optique Cohérente (OCT) et l'imagerie PET-Scan.

En langue des signes et plus généralement en communication visuo-gestuelle, les recherches effectuées par TCI concernent l'analyse et l'interprétation des attitudes et des gestes de communication émis par des personnes observées par un système de vision. Il s'agit donc de détecter et de suivre les différents composants corporels (mains, buste,

visage), de reconstruire la posture 3D des personnages, d'analyser leurs expressions et d'interpréter ces éléments en termes langagiers afin de reconstruire le sens de l'énoncé. Pour prendre en compte la variabilité du corps humain et la complexité de la langue des signes, on doit représenter et exploiter des connaissances a priori à tous les niveaux de la chaîne de traitements et introduire des contraintes. Les travaux de TCI portent donc d'une part sur l'étude de modèles dynamiques de gestes et d'expressions faciales et sur la modélisation de la langue des signes et d'autre part sur l'exploitation de ces modèles par un système d'interprétation.

La liste des membre de l'équipe TCI est mise à jour chaque nuit sur cette page :

http://www.irit.fr/Personnel,125?lang=fr

Et plus généralement, plus d'informations sur les projets ANR et européens, les publications de l'équipe, les sujets de stage et les rapports d'activité de l'équipe à cette adresse:

http://www.irit.fr/-Equipe-TCI-

Activités extra-Bureau

Le stage s'est déroulé principalement dans le bureau 204 du premier étage de l' IRIT. Cependant, je n'ai pas passé tout mon temps dans ce Bureau, et Monsieur Crouzil m'a proposé diverses activités que je considère comme faisant partie intégrante du stage.

Notamment la journée industrielle des technologies et de l'image qui a eu lieu le 21 Novembre 2013 au CIMI (Centre International de Mathématiques et d'Informatiques) qui a eu lieu au MRV (Maison de la Recherche et de la Valorisation). Cette journée a présenté diverses applications industrielles du traitement d'images, avec plusieurs intervenants de l'industrie et de la recherche. Les domaines présentés étaient la vision artificielle, l'imagerie médicale et l'imagerie spatiale.

De nature curieuse et scientifique, j'ai beaucoup profité de cette journée qui m'a permis d'avoir une vision plus large de la recherche et son lien avec l'industrie et des technologies en vogue actuellement ainsi que des problématiques actuelles qui restent toujours à résoudre.

Les présentations de la reconstruction 3D grand public très populaire actuellement et l'analyse géométrique et sémantique de la scène à partir d'images 2D ont retenus mon attention car ce sont des domaines plus ou moins en lien avec mon stage et que les résultats présentés m'ont impressionnés.

En plus de la journée CIMI, j'ai assisté aux réunions de l'équipe, ce qui, même si je n'ai pas participé oralement, m'a permis d'être en contact avec les différents membres et de comprendre les problématiques actuels de l'équipe et plus généralement de l'IRIT, divers fonctionnements administratifs comme la prise en charge d'un doctorant et les étapes successives amenant au déroulement d'une thèse.

L'échange avec des membres de l'équipe m'a aussi permis de m'intégrer et de partager des points de vue différents.

J'ai aussi assisté à la thèse de Renaud MORIN portant sur l'amélioration de la résolution en imagerie ultra-sonore à laquelle il m'avait convié. J'ai donc pu voir le déroulement d'une thèse en condition réel. Je me suis rendue compte que la présentation d' une thèse ne durait au final pas longtemps (20-30 minutes) et que ce sont surtout les questions des examinateurs, jurés et autres acteurs majeurs qui permettent de rentrer

dans les détails et d'éclaircir des points précis.

J'ai par la suite pu échanger sur la thèse avec certains membres de l'équipe et ainsi instaurer des relations instructives et plaisantes.

Expérience acquise

Ce stage m'a au final beaucoup profité. Il m'a permis de découvrir un domaine de la vision par ordinateur que je ne connaissais pas ou peu. Il m'a aussi permis de mieux évaluer mes capacités et d'avoir une plus grande confiance en moi. Du fait que j'ai du expliquer mes travaux à Mr Crouzil, j' ai appris à avoir une certaine rigueur quant à l'utilisation de certains termes techniques et à améliorer mon aisance et mon expression orale.

Les horaires souples et l' autonomie m'ont permis une approche différente du travail, cela m' a amené à chercher de par moi même plusieurs pistes pour résoudre les problématiques. J'ai toujours pensé qu'il était bon de d'abord chercher dans toutes les directions pour faire un petit état de l'art et trouver de nouvelles solutions, puis se concentrer sur une piste et organiser son travail que d'arriver déjà avec un planning rigoureux et une liste de tâche prémachées qui pourraient se résumer par un bête exercice de TP. Mr Crouzil m'a toujours laissé la liberté de faire ce qui me semblait le plus efficace et intéressant dans le cadre du stage sans m'imposer une méthodologie stricte, tout en m'écartant des pistes qu'il savait infructueuse ou mal adapté pour le stage.

Il m'a cependant proposé quelques fonctions à coder, notamment l'interface graphique et les diverses adaptations et complétude progressives des fonctions d'appariement de points d'intérêts.

Introduction sur la Mise en correspondance de points d'intérêt

La mise en correspondance de points d'intérêts se fait principalement grâce à trois étapes/traitements.

La détection de zones d'intérêts, l'extraction de caractéristiques et l'appariement des points.

La détection de zones d'intérêts

La détection de zones d'intérêts (le terme *feature* est très présent dans la littérature anglo-saxone): ce traitement consiste à repérer des zones jugées intéressantes pour l'analyse présentant des propriétés locales remarquables. Ces zones peuvent être des points, des courbes continues ou des régions connexes rectangulaires ou non. Au cours de ce stage, nous nous sommes plus particulièrement intéressés à des zones se limitant à des points.

La répétabilité, c'est à dire le fait que les mêmes zones d'intérêts puissent être détectées sur deux images différentes mais représentant la même scène est une propriété importante et généralement exigée pour tous les algorithmes de détection de zones d'intérêts.

Pour la détection de contours, les premières méthodes se fondent sur l'analyse des contours et des arêtes, c'est à dire des zones ou la couleur (ou luminance) de l'image change brusquement. Les opérateurs de Canny et Sobel peuvent être utilisés à l'aide de filtres (voir la fonction CannyDetector_Demo.cpp dans le répertoire testCanny).

Ces détecteurs sont basés sur une dérivation (gradient) de l'image. Les points de discontinuités peuvent être reliés entre eux pour former des arrêtes (edges).

Pour la détection de points d'intérêts : les algorithmes de détection de points d’intérêts se focalisent sur des points particuliers des contours sélectionnés selon un critère précis. Les coins (corners) sont les points de l'image où le contour change brutalement de direction, comme aux quatre sommets d'un rectangle. Il s'agit de points stables, et donc intéressant pour la répétabilité. Le détecteur de Harris est un des plus

connus.

La plupart des techniques de détection de point d'intérêt sont basées sur une analyse locale de l'image. Ce qui les différencie entre elles est l'opérateur de dérivation utilisé. Nous pouvons par exemple citer les méthodes basées sur les DoG (Difference of Gaussians), les LoG (Laplacian of Gaussian) ou les DoH (difference of Hessians).

La détection de régions d'intérêts se focalise sur des zones d'intérêts plus générales que des points, utiles lorsque les structures à rechercher ne correspondent pas à des points saillants. Ces techniques commencent souvent par identifier des points d'intérêt qui vont servir de barycentres des régions recherchés (appelées blobs en anglais), telles que les méthodes multi-échelles basées sur l'étude des détecteurs de points d'intérêts (Harris, DoG) à différentes échelles de l'image. Ceci permet d'obtenir des régions circulaires ou elliptiques. Ces méthodes sont souvent intégrés à des algorithmes plus généraux comme SIFT ou SURF.

Parmi les détecteurs de régions d'intérêts plus généraux existe aussi MSER. MSER est utilisé comme une technique de détection de tâches (blob) pour trouver des correspondances d'éléments entre deux images avec des points de vues différents.

Plus spécifiquement enfin, la recherche d'objets particulièrement allongés fait souvent appel à des techniques dites de détections de crêtes (ridge detection) qui se prêtent bien à l'analyse de vues aériennes de réseaux routier, ou à la détection de vaisseaux sanguins dans l'imagerie médicale par exemple.

L'extraction de caractéristiques

Après la détection, on applique souvent un algorithme de description qui va se concentrer sur chaque zone d'intérêts calculée précédemment pour en calculer des caractéristiques. C'est ce qu'on appelle l'extraction de caractéristiques.

L'extraction de caractéristiques visuelles consiste à utiliser des transformations mathématiques calculées sur les pixels d'une image numérique. Ces caractéristiques permettent de mieux rendre compte de certaines propriétés visuelles de l'image, utilisées pour des traitements ultérieurs. (comparaisons, appariement dans le cas qui nous intéresse).

On distingue les caractéristiques globales calculées sur toute l'image et les

caractéristique locales utilisées autour de points d'intérêts. Les caractéristiques semi locales quant à elles sont extraites dans des zones restreintes de l'image, résidant d'une segmentation ou d'une grille arbitraire.

La prise en compte de la couleur des images a été l'une des premières caractéristiques employée pour la recherche d'images par contenu. (1991 Swain et Ballard proposent d'utiliser un histogramme couleur).

Des 1995 ont commencés à apparaître des méthodes utilisant des moments calculés dans un espace de couleur donné pour caractériser les images. On se restreint en général aux deux ou trois premiers moments. (moyenne, variance, asymétrie) mais ils sont calculés dans chacun des trois plans de l'espace couleur choisi.

On peut aussi décrire les formes. Pour cela il faut faire au préalable une identification des régions. Cela peut résulter d'une segmentation de l'image ou de la détection de leurs contours. On peut alors caractériser les régions au moyen de divers indices telle leur orientation principale. Une méthode de description particulièrement utilisée est celle de Mokhtarian, appelée Curvature Scale Space. Elle consiste à décrire une région en fonction des variations de courbure de son contour.

Cette description est invariante par rotation et par translation. Pour le zoom, il faut normaliser par la largeur du contour. Le descripteur est généralement calculé à différentes échelles, chacune correspondant à la convolution par un noyau gaussien.

La texture est aussi utilisée comme caractéristique. La notion de texture n'est pas évidente et dépend de l'échelle. On distingue trois approches pour définir la notion de texture. Une première approche cherche à décrire une texture en terme de propriétés statistiques des valeurs et positions relatives des pixels (Haralick). C'est l'approche stochastique.

Une seconde suppose l'existence de primitives fondamentales, une texture étant décrite comme une combinaison complexe de ces primitives, exprimés avec des graphes par exemple. C'est l'approche structurelle.

Une troisième approche utilise la perception visuelle et s'inspire du processus de formation des textures du point de vue humain. C'est l' approche spectrale.

Dans l'approche stochastique une texture peut être décrite à l'aide d'une matrice de cooccurrence de niveau de gris. En gros cette matrice compte le nombre de fois qu'un pixel d'un niveau de gris se situe à coté d'un autre niveau de gris pour tous les niveaux de gris présent dans l'image.

Le modèle MSAR (Multiresolution Simultaneous Autoregressive Models) est une approche largement utilisée. Elle utilise un processus autoregressif, qui représente l' évolution d'une variable aléatoire.

On peut aussi essayer d'exprimer des périodicités et autres régularités dans le cadre de l'analyse spectrale. Une transformée de Fourier discrète ou une transformée de cosinus discrète peut être une base pour des descripteurs. Il est cependant plus courant d'utiliser des filtres de Gabor.

On peut également calculer en plusieurs points d'intérêts des vecteurs caractéristiques. Ces vecteurs contiennent parfois des données provenant de la détection, telle que l'orientation de l'arrête ou la magnitude du gradient dans la zone d'intérêt. Généralement le vecteur caractéristique en un pixel est calculé sur un voisinage de ce pixel. Il peut être calculé à des échelles différentes pour s'affranchir du zoom. Parmi les caractéristiques locales on trouve les histogrammes de couleur ou des vecteurs rendant compte de l'orientation des gradients de niveaux de gris.

Les méthodes SIFT et SURF incluent une détection de zone d'intérêts et le calcul d'un vecteur caractéristique en chacune de ces zones. SIFT utilise en gros un histogramme des orientations de gradient, SURF utilise le calcul d'approximation d'ondelettes de Haar.

Appariement

Pour l'appariement des points, on utilise des algorithmes de recherche des plus proches voisins (plus proche au niveau des caractéristiques des descripteurs dans le cas présent).

On peut utiliser bêtement une recherche linéaire, mais cette méthode est lente.

Lorsque la dimension de l'espace de recherche est petite (inférieur à 15) on peut utiliser des algorithmes de recherches efficaces. La structure la plus connues est le kd-tree.

Le problème de ces méthodes est que si la dimension D est trop grande, elles ont des performances comparables ou inférieur à la recherche linéaire. (malédiction de la dimension).

Une méthode cherchant à résoudre ce problème est LSH (locality sensitive hashing), utilisant une fonction de hachage.

L' échantillonnage par bit est une méthode permettant de construire une famille

LSH. Cette approche peut être adaptée à la distance de Hamming.

On peut aussi utiliser des bibliothèques comme FLANN (Fast Library for Approximate Nearest Neighbors) qui contient une collection d'algorithmes, et un système qui choisit automatiquement les algorithmes les plus performant et les meilleurs paramètres.

Travail réalisé

Interface Graphique

J'ai créé une petite interface graphique à l'aide d'exemples trouvés sur le web. Le but était d'arriver à mieux visualiser les résultats d'une correspondance de point. En effet, la visualisation standard consiste à mettre les deux images à comparer côte à côte, d'entourer (ou pas) les points d'intérêts des deux images, et de relier les points d'intérêts mis en correspondance par différentes couleurs.

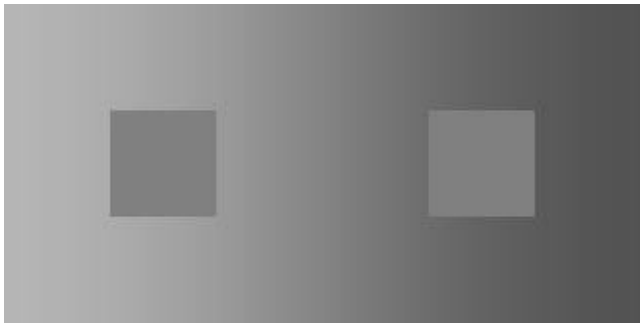
Cette visualisation devient très vite gênante, même quand le nombre de points d'intérêts à relier est faible, et est donc problématique pour évaluer la qualité des appariements.

Monsieur Crouzil m'a fait part de ce problème et a exprimé son besoin de «voir le point se déplacer, si possible avec une traîné derrière pour bien ressentir la direction et la force du mouvement, tel des vecteurs».

Partant de cet esprit là, j'ai mis en place une petite interface graphique à l'aide d'un trackbar, outil déjà présent dans OpenCV. Ce qui est étonnant car c'est quasiment le seul widget d'interface graphique présent dans OpenCV (on peut faire des choses basique comme ouvrir une fenêtre, gérer des événements de clic de souris, utiliser des boutons, dessiner des points, des lignes, des rectangles, du texte, à l'aide de la librairie Highgui, mais elle est très limitée. Il s'avère compliqué de construire des GUIs (Graphical Interfaces Users) avec OpenCV et il est préférable d'utiliser d'autres outils comme Qt).

Lorsque le curseur est à 0 (état initial), l'image de gauche est affichée en arrière plan avec les points d'intérêts appariés trouvés entourés de différentes couleurs en premier plan. Lorsque l'on déplace le curseur vers la droite, à l'arrière plan chaque pixel de l'image de gauche se transforme progressivement en un pixel de l'image de droite par interpolation linéaire. Cela permet de voir le nouvel emplacement de points remarquables (par exemple le haut d'un arbre ou le coin d'une maison) et de mieux évaluer les mises en correspondance. Les points d'intérêts se déplacent aussi vers leur point apparié de manière linéaire; un point apparié avec un point éloigné se déplacera donc plus rapidement et sera par conséquent très visible.

Pour le choix des couleurs des points, je suis parti du raisonnement que ce n'était pas une trop mauvaise idée de considérer que dans une composante (bleu, vert ou rouge), plus la différence de valeur entre deux pixels est importante, plus la différence visuelle est importante. La notion de « distance entre deux couleurs perçus par l'oeil humain » est en fait extrêmement subjective. Certaines illusions d'optique montrent que notre perception des couleurs dépend des couleurs alentours (sans même parler de couleur, un carré gris sur un fond blanc semble plus foncé qu'un carré gris sur un fond noir).



exemple de la non unicité de la perception humaine des couleurs

Je suis aussi parti du principe que pour ne pas confondre deux points d'intérêts, il valait mieux qu'aucun n'est la même valeur qu'un autre.

Il y a trois canaux de couleurs (je suis resté en RGB). Quitte à différencier les points d'intérêts autant mettre de la couleur.

Il y a donc 256*256*256 couleurs différentes.

Je fais donc $\frac{nbCouleurs}{nb\,de\ points\,à\,apparier}$ pour avoir le pas (la « distance » entre chaque couleur de point).

Puis j'effectue $numero\,du\ point \times pas\,couleur$ pour obtenir sa couleur actuelle, que nous appellerons Cactu.

Le reste de la division de Cactu par 256 donne l'intensité du bleu, on divise ensuite Cactu par 256, le reste de la division par 256 donne le vert, on redivise par 256 et on a ainsi la composante rouge.

Idées d'amélioration

Dans la majorité des cas cette méthode s'avère satisfaisante, on visualise bien mieux les résultats, et la couleur et le déplacement des points d'intérêts permet beaucoup mieux de suivre le mouvement et évaluer visuellement les résultats obtenus. Néanmoins parfois on a beaucoup de points blanc, gris et noir et on ne les distingue pas forcément très bien. Cela est du à la méthode utilisée, par exemple si on a 3 points:

noir pour le premier point (0,0,0)

$256^3/3$= ((85*256)+85)*256+85 (85,85,85) pour le deuxieme point

$2*(256^3/3)$=((170*256)+170)*256+170 (170,170,170) pour le troisième point

Ce qui correspond à trois nuances de gris, qui ne sont pas très visibles en générale. Le problème semble donc venir des puissance de 3.

On peut donc encore améliorer la lisibilité de cette interface graphique, en utilisant par exemple un autre espace que RGB. (les espaces TSV mais surtout TSL, qui décrient les couleurs distinguées par l' oeil humain, peuvent être envisagées).

On peut aussi ajouter un autre trackbar sur un seuil, pour n'afficher que les points d'intérêts qui sont suffisamment bien appariés par l'apparieur, mais cela peut s'avérer compliqué avec OpenCV qui n'a que peu d'outils permettant une interface graphique utilisateur (la trackbar est l'exception qui confirme la règle). De plus une trackbar ne me paraît pas idéal, je pensais utiliser un TextField pour indiquer le seuil désiré mais cet outil ne semble pas présent sur OpenCV.

On peut donc penser que l'utilisation d'une bibliothèque graphique comme Qt permettrait de faciliter la mise en place et d'améliorer l'interface graphique.

Recherche

J'ai effectué plusieurs recherches sur différents détecteurs, descripteurs et appariеurs, et étudié leurs façons de fonctionner et leurs particularités.

Le site d'OpenCV renvoie souvent à certaines publications expliquant le fonctionnement des détecteurs, descripteurs/extracteurs et appariеurs. Voici celles que j'ai relevé:

Pour le detecteur FAST:

E. Rosten. Machine Learning for High-speed Corner Detection, 2006.

Pour le detecteur STAR:

Agrawal, M., Konolige, K., & Blas, M. R. (2008). Censure: Center surround extremas for realtime feature detection and matching. In Computer Vision–ECCV 2008 (pp. 102-115). Springer Berlin Heidelberg.

Pour le decteur MSER: on a un lien vers wikipedia:

http://en.wikipedia.org/wiki/Maximally_stable_extremal_regions

Pour Harris:

La fonction exécute le détecteur de contours Harris sur l'image. Pour chaque pixel (x,y) elle calcule la matrice de covariance 2x2 $M^{(x,y)}$ sur un voisinage de blocksize x blocksize. Ensuite, elle calcule la caractéristique suivante:

Les coins de l'image peuvent être trouvés aux maxima locaux de cette réponse.

dst– Image ou stocker les réponses du détecteur de Harris. Ses elements sont de type CV_32FC1 et elle a la même taille que src.

BlockSize – taille du voisinage

k– paramètre libre du détecteur de Harris

Pour le descripteur SIFT:

Lowe, D. G., “Distinctive Image Features from Scale-Invariant Keypoints”, International Journal of Computer Vision, 60, 2, pp. 91-110, 2004.

Pour le descripteur SURF:

Class for extracting Speeded Up Robust Features from an image

Bay, H. and Tuytelaars, T. and Van Gool, L. “SURF: Speeded Up Robust Features”, 9th European Conference on Computer Vision, 2006.

Pour le descripteur BRIEF:

Calonder M., Lepetit V., Strecha C., Fua P. *BRIEF: Binary Robust Independent Elementary Features* , 11th European Conference on Computer Vision (ECCV), Heraklion, Crete. LNCS Springer, September 2010.

Pour le detecteur et descripteur ORB:

Class implementing the ORB (*oriented BRIEF*) keypoint detector and descriptor extractor

Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary R. Bradski: ORB: An efficient alternative to SIFT or SURF. ICCV 2011: 2564-2571.

Le detecteur/descripteur BRISK était absent de ma version d'openCV. On peut néanmoins trouver de la documentation sur l'algorithme utilisé ici:

Stefan Leutenegger, Margarita Chli and Roland Siegwart: BRISK: Binary Robust Invariant Scalable Keypoints. ICCV 2011: 2548-2555.

Le descripteur FREAK n'étant pas présent lui non plus, on peut se documenter sur cet algorithme ici :

A.Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast Retina Keypoint. In IEEE Conference on Computer Vision and Pattern Recognition, 2012. CVPR 2012 Open Source Award Winner.

Le detecteur DENSE est bien présent dans OpenCV, mais je ne suis pas parvenu à l'utiliser de manière satisfaisante. On peut trouver sa description ici, mais aucune publication conseillé par OpenCV:

http://docs.opencv.org/modules/features2d/doc/common_interfaces_of_feature_detectors.html#DenseFeatureDetector : public FeatureDetector

Le detecteur SIMPLEBLOB: Il est bien présent et il fonctionne

The class implements a simple algorithm for extracting blobs from an image:

1. Convert the source image to binary images by applying thresholding with several thresholds from minThreshold (inclusive) to maxThreshold (exclusive) with distance thresholdStep between neighboring thresholds.
2. Extract connected components from every binary image by findContours() and calculate their centers.
3. Group centers from several binary images by their coordinates. Close centers form one group that corresponds to one blob, which is controlled by the minDistBetweenBlobs parameter.
4. From the groups, estimate final centers of blobs and their radiuses and return as locations and sizes of keypoints.

This class performs several filtrations of returned blobs. You should set filterBy* to true/false to turn on/off corresponding filtration. Available filtrations:

- **By color**. This filter compares the intensity of a binary image at the center of a blob to blobColor. If they differ, the blob is filtered out. Use blobColor = 0 to extract dark blobs and blobColor = 255 to extract light blobs.
- **By area**. Extracted blobs have an area between minArea (inclusive) and maxArea (exclusive).
- **By circularity**. Extracted blobs have circularity ( ) between minCircularity (inclusive) and maxCircularity (exclusive).
- **By ratio of the minimum inertia to maximum inertia**. Extracted blobs have this ratio between minInertiaRatio (inclusive) and

maxInertiaRatio (exclusive).

- **By convexity**. Extracted blobs have convexity (area / area of blob convex hull) between minConvexity (inclusive) and maxConvexity (exclusive).

Default values of parameters are tuned to extract dark circular blobs.

L'apparieur FLANN : L'apparieur flann utilise plusieurs algorithmes pour augmenter le temps de calcul en faisant des approximation avec des kd-trees :

Marius Muja, David G. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration, 2009

J'ai aussi effectué des tests sur des fonctions hors OpenCV. J'ai utilisé le descripteur DAISY à l'aide d'un programme écrit en C téléchargé à cette adresse :

http://cvlab.epfl.ch/~tola

Attention, ce programme génère un fichier très très lourd contenant la valeur des descripteurs.

J'ai effectué un test avec Adjuster Adaptater, il peut être intéressant de l'étudier plus en profondeur et de l'utiliser dans les fonctions que j'ai implémenté.

Grid Adapted FeatureDetector a aussi été testé

L'application des filtres de sobel et Prewitt a aussi été testé avec des fonctions simple cpp. Elles peuvent être généralisé pour appliquer un filtre à une image.

Des images tests sont fournis dans le répertoire stageIrit/tests/images

Un test avec le détecteur Canny a été utilisé mais n'a pas servi pour ce stage.

Un test a été effectué avec une fonction SUSAN codée en C par le monsieur dont le site web est:

http://www.fmrib.ox.ac.uk/~steve

RANSAC a été testé sur Pig2PigBiDir et je trouve qu'il peut donner des résultats impressionnants en calculant la matrice fondamentale.

Nous allons ici nous concentrer sur le détecteur STAR, MSER et SIFT.

Présentations de plusieurs méthodes de détection

Le but de ce chapitre est de présenter rapidement les différentes approches utilisées par les différents détecteurs connus. Ils est très largement inspirés de l'ECCV de différentes années (European Conference on Computer Vision) que j'ai essayé de condenser.

Fast est utilisé car il réalise des calculs moins énergivore que ses homologues SIFT (DoG), Harris et SUSAN, ce qui permet de l'utiliser dans des applications en temps réel. Il a été conçu spécialement pour être rapide.

La majorité des algorithmes de détection de caractéristiques calcule une fonction de réponse à des points d'intérêts C dans l'image. Seuls les pixels au dessus d'un certain seuil sont retenus.

Moravec calcule la somme des différences des carrés (SSD) dans un voisinage d'un candidat retenu comme point d'intérêt. C est alors la plus petite valeur du SSD obtenue.

Harris construit une matrice en utilisant une approximation de la dérivée seconde de la SSD. Cette matrice est :

$$\mathbf{H} = \begin{bmatrix} \widehat{I_x^2} & \widehat{I_x I_y} \\ \widehat{I_x I_y} & \widehat{I_y^2} \end{bmatrix}$$

où ^ designe la moyenne effectuée dans le voisinage considéré. Une fenetre circulaire lisse peut être utilisée au lieu d'un rectangle pour améliorer le calcul de la moyenne et permettre d' avoir une réponse moins bruité et isotropique.

Harris definit la réponse C ainsi :

$$C = |\mathbf{H}| - k(\operatorname{trace} \mathbf{H})^2.$$

Ce résultat est important si les deux valeurs propres de H sont grandes, et il évite le calcul directe des valeurs propres.

En se basant sur l'hypothèse de déformation affine de l'image, une analyse mathématique de Shi et Tomasi conclue qu'il est préférable d'utiliser la valeure propre la plus petite de H en tant que fonction décrivant la force d'un poit d'intérêt:

$$C = \min(\lambda_1, \lambda_2).$$

De nombreuses suggestions ont été faites pour calculer C à partir de H, et elles sont toutes équivalente à de nombreuses normes de la matrice H.

Lowe, afin d'obtenir une invariance d'échelle, convolue une image avec un DoG à différentes échelles parcequ'il est une bonne approximation du LoG (Laplacian of Gaussian) et beaucoup plus rapide à calculer.

Une approximation du DoG a été proposé, qui pourvu que les échelles soient écarté de $\sqrt{2}$, accélérent le calcul par un facteur d'environ 2 par rapport à l'implémentation simple de la convolution Gaussienne.

Il est à noter que le LoG est un noyau espace-échelle particulièrement stable.

Des techniques d'espace-échelle ont été combiné avec l'approche de Harris qui calculent la réponse de Harris à différentes échelles et garde seulement ceux qui ont des optima de la réponse du LoG à plusieurs échelles.

L'invariance d'échelle a été étendue pour considérer les régions caractéristiques comme étant invariantes aux transformations affines.

Une arrête dans une image correspond à la bordure entre deux regions. Aux coins d'une région, cette bordure change de direction rapidedement. De nombreuses techniques ont donc été développées qui impliquent la detection et le « chainage des arrêtes » (chaining edges). Elles utilisent des maxima de coubure, des changements de direction ou des changements d'apparence. D'autres techniques évitent le chainage des arrêtes et à la place cherchent des maxima de courbure ou des changements de direction aux endroits où le gradient est important.

D'autres detecteurs de coins fonctionnent en examinant une petite region de l'image pour voir si elle « ressemble » à un coin. Etant donné que les dérivées secondes ne sont pas calculées, une étape de réduction de bruit (comme un lissage Gaussien) n'est

pas nécessaire. Par conséquent, ces detecteurs de coins sont calculatoirement efficaces puisque seulement un petit nombre de pixels sont examinés pour chaque coins détecté. Un corollaire de cette méthode est qu'ils ont tendance à être médiocre pour les images avec seulement des caractéristiques à grande échelle comme les images floues.

Certaines méthodes supposent qu'un coin ressemble à un « coin floue »(blurred wedge) ; et calculent les caractéristiques du coin (amplitude, angle et floue) en l'ajustant à l'image local. Cette idée est généralisée par une méthode calculant les autosimilitudes en regardant la proportion des pixels dans un disque dont l'intensité est proche d'un seuil correspondant à la valeur du centre (nucleus). Les pixels les plus proches en terme de valeur du centre ont un poids plus important. Cette mesure est connu sous le nom d' USAN (the Univalue Segment Assimilating Nucleus). Une faible valeur d'USAN indique un coin coin puisque le pixel du centre est tres différents des ses alentours. Un ensemble de règle est utilisé pour supprimer les caractéristiques de « mauvaise qualité ». Par la suite les minima locaux, les SUSANs (Smallest USAN), sont conservés comme candidats.

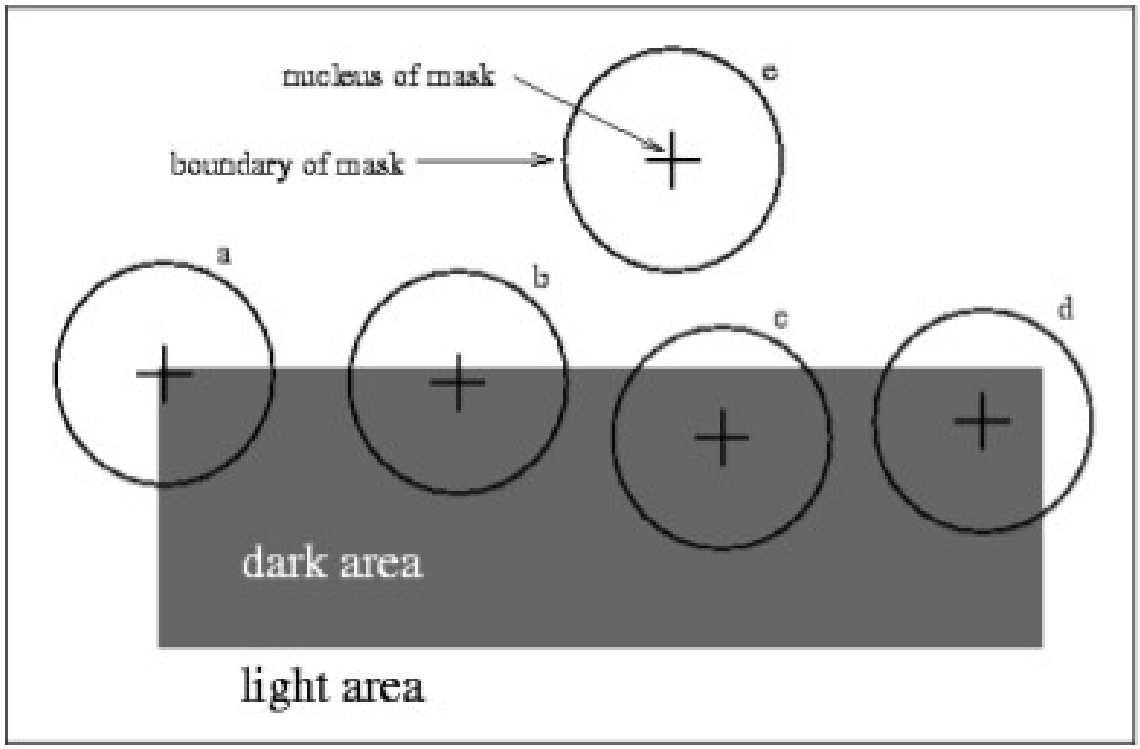


Figure 1. Four circular masks at different places on a simple image

Illustration de SUSAN issue de*: The Comparison and Application of Corner Detection Algorithms* Jie Chen, Li-hui Zou, Juan Zhang, Li-hua Dou

Trajkovic et Hedley utilisent une idée similaire : une région est non autosimilaire si les pixels semblent généralement différents du centre de la région. Cela est mesuré par un cercle à considerer. f_C est la valeur du pixel au centre du cercle, et les f_p sont les valeurs des pixels à chaque extrémité du diametre du cercle. La fonction réponse est définie comme étant :

$$C = \min_P (f_P - f_C)^2 + (f_P - f_C)^2.$$

Cette réponse ne peut être importante que dans le cas de points d'intérets. Le test est effectué dans un cercle de Bresenham. Puisque le cercle est discrétisé, une interpolation linéaire ou circulaire est utilisée dans des orientations discrètes afin de donner au détecteur une réponse plus isotropes. Pour finir, les auteurs presentent une méthode, grâce à laquelle le minimum de la fonction de réponse à toutes les positions interpolées entre deux pixels peuvent être calculés efficacement. Calculer la fonction de réponse nécessite d'effectuer une recherche dans toutes les orientations, mais chaque mesure fournit une borne supérieur à la réponse. Pour accélérer l'appariement, on regarde seulement dans les directions verticales et horizontales de la réponse. Si la borne suppérieur de la réponse est trop basse, alors le point d'intérêt potentiel est rejeté. Pour accélérer encore plus le procédé, cette vérification rapide est d'abord appliquée sur une échelle grossière.

Une rapide transformé de symétrie radiale est utilisée pour détecter les points. Les points ont un grand score quand le gradient est à la fois radialement symétrique, important et d'un signe constant le long du rayon. L'échelle peut varier en changeant la taille de la zone à examiner pour la symétrie radiale.

Une méthode alternative d'examination d'une petite région de l'image pour voir si il y a des coins est l'apprentissage automatique pour classer les régions de l'image comme des coins ou non coins. Les exemples utilisés dans l'ensemble d'apprentissage déterminent le type de caractéristiques détectées. Une méthode utilisée est de former un réseau neuronal à trois couches pour reconnaître les coins où les arrêtes se rencontrent à un multiple de 45°, proche du centre d'une fenetre de taille 8*8. Cela est appliqué à toute l'image après la détection de contours et « amincissement » (thinning). Il a été montré que le réseau neuronal a appris une représentation plus générale et a été en mesure de détecter des coins à une variété d'angles.

Pistes d'approfondissement de recherche

Il y a de nombreuses pistes à explorer.

On peut essayer d'utiliser des zones ne se limitant pas à un point mais des contours ou par exemple la détection de Blob. Est-ce que l'algorithme de détection de blob permet de définir le centre des tâches de manière rigoureuse, précise et unique ? Est-ce efficace ?

On peut voir plus en profondeur ce qui a été fait sur MATLAB et les fonctions présente, les détecteurs/descripteurs et extracteurs.

Essayer d'utiliser un apprentissage automatique pour repérer des éléments caractéristiques du paysage et ainsi mieux les apparier.(ça semble un peu difficile à mettre en œuvre tout de même).

On peut chercher à utiliser les trois approche différentes pour la texture avec MSAR pour l'approche stochastique, étudier les primitives fondamentales comme les Textons dans l'approche structurelle, et essayer de mettre en œuvre une approche spectrale.

On peut essayer de voir si des notions de teinte, de saturation, de luminosité et de valeur sont prises en compte dans les algorithmes utilisés, et s'il ne serait pas plus efficace d'utiliser les espaces TSV ou TSL, et grâce à leur caractéristiques spécifiques, aboutir à des algorithmes plus simple ou plus efficace. (cependant, rien que la complexité extrême de la façon de représenter ces espaces colorimétriques rend compte de la difficulté à mettre une notion de « distance » entre deux couleurs).

Réflexion personnel sur la notion de couleur et son implication dans le stage

Si on veut un truc parfait :

Un des plus gros problèmes vient de notre perception des couleurs, il y a des informations que nous ne percevons pas et que seul une machine peut avoir. Nous ne pouvons pas utiliser ces informations. On peut penser qu'un système parfait serait de prendre une « photo » ou chaque pixel serait une valeur d'intensité de longueurs d'onde. Que les rayons gamma, les rayons X, les ultraviolets, les infrarouges, les Térahertz, les micro-ondes, les ondes radio... soient toutes prisent en compte pour chaque pixel.

Ensuite il faut appliquer un poids à chacune de ces longueurs d'ondes possible pour « éliminer » ce qui ne dépend pas de l'espace ni du temps à une grande échelle (i.e par exemple il y a autant d'UV partout), ou ce qui varie tous le temps dans le temps.

Il ne faut accorder de l'importance qu'au données utiles.

Ensuite, la seule solution est que la machine apprenne par elle même, qu'elle s'adapte au fur et à mesure. Comme nous ne pouvons pas percevoir certains « liens » entre deux photos, la machine est obligé d'apprendre par elle même, nous ne pouvons pas lui dire « oui ce paramètre là est important et très caractéristique » car il s'agit de choses que nous ne percevons pas. Et nous n'avons aucun moyen de savoir si la logique qui a amené à un appariement est la bonne.

Faut il laisser des machines décrire elles même des couleurs ou toujours utiliser des elements que l'on peut percevoir ? Comment manipuler des choses que l'on ne voit pas ?

CIE est une methode basé sur une moyenne de la perception visuelle humaine. Le système colorimétrique s'adapte à l'homme, à l'évolution.

Lors d'une prise de photo, un systeme linéaire est utilisé pour la capture il me semble. Fidélité du capteur ?

La luminosité L* (fonction non linéaire de la luminance L) a été définie pour adapter les notions de distance de couleurs à l' oeil humain.

Codage

Monsieur Crouzil m'a proposé plusieurs pistes de recherche et notamment m'a confié la tâche de coder une série de fonction:

Les fonctions sont généralistes : on leurs passe le détecteur, le descripteur et l'apparieur en parametre.

Les détecteurs, descripteurs et apparieurs peuvent être paramétésr à l'intérieur de chaque fonction. Voir le fichier readme.txt pour voir rapidement leur utilisation.

Le seuil threshMatches peut aussi être régler pour afficher les appariements dont la « distance des descripteurs » est la plus faible, c'est à dire les appariements de point qui ont des caractéristiques assez proche.

Lorsqu'un voisinage doit être pris en compte, il est généralement paramétrable par l'attribut eucDist.

PoiL2PoiR: (Point Of Interest Left To Point Of Interest Right)

Cette fonction prend en argument un détecteur, un descripteur/extracteur et un apparieur parmi ceux que j'ai réussi à implémenter.

Le but est de détecter les points d'intérêts dans l'image source 1 (image left), les points d'intérêts dans l'image source 2 (image right), et de les apparier à l'aide de l'apparieur.

La fonction string type2str(int type) n'est pas présente ici.

On peut paramétrer le détecteur FAST, SURF,ORB,HARRIS(mais il faut laisser l'argument à true pour continuer à utiliser Harris),SIMPLE BLOB facilement à l'interieur de la fonction.

PoiL2PoiRInNePoiL: (Point of interest Left to point of interest Right in the neighbourg of the Point of interest Left)

Cette fonction va détecter les points d'intérêts dans l'image source 1 (image left) et va choisir pour chacun le meilleur point d'intérêt correspondant dans l'image source 2

parmi ceux situés dans un voisinage spécifique.

Cela part du principe qu'un point d'intérêt et son correspondant sont proches du point de vue de la distance Euclidiene.

PoiL2PixIn9NPoiRInNePoiL : (Point of interest Left to pixels in the 9-neighborhood of Point of interest Right In neighborhood of Point of interest Left)

Cette fonction va détecter les points d'intérêts dans l'image source 1, et va, pour chacun de ces points, chercher dans l'image source 2 les points d'intérêts situés dans un voisinage spécifique, ainsi que leur 8-voisinage, puis choisir le meilleur appariement parmi ces points.

Le voisinage spécifique peut être paramétrée par EucDist. Cela correspond en distance en pixel au rayon du voisinage, qui a donc une forme circulaire.

Cela suppose qu'il peut y avoir des pixels autour des points d' intérêts de l'image source 2 qui sont plus proche du point d'intérêt de l'image source 1 en terme de distance calculé à partir des descripteurs.

PoiL2PixInNePoiL : (Point of interest Left to pixels in the neighborhood of Point of interest Left)

Cette fonction détecte les points d'intérêts dans l'image source 1, puis, pour chacun de ces pixels, va chercher dans l'image 2 le pixel situé dans un certain voisinage le plus proche du point de vue distance des descripteurs.

Ci dessus le voisinage choisi est un cercle mais peut très facilement être adapté en carré.

J'ai choisi d'utiliser le cercle comme voisinage afin de respecter au mieux la notion de distance Euclidienne.

J'ai laissé volontairement la fonction « type2str » dans le code pour quel'utilisateur ait les moyens de connaître le type d'une matrice.

PoiL2PixInNePoiLBidir :(Point of interest Left to pixels in the neighborhood of Point of interest Left Cidirectional)

Cette fonction détecte les points d'intérêts dans l'image source 1, puis, pour chacun de ces pixels, va chercher dans l'image 2 le pixel situé dans un certain voisinage, i.e à une distance Euclidienne EucDist, le plus proche du point de vue distance des descripteurs.

Ensuite, on cherche à re-apparier les pixels précédemment appariés dans un voisinage de même dimension. Si on retombe sur le point d'intérêt original (avec un écart de 1 pixel autorisé), c'est que l'appariement est bon. Sinon, on ne le garde pas.

Ci dessus le voisinage choisi est un cercle mais peut très facilement être adapté en carré.

J'ai choisi d'utiliser le cercle comme voisinage afin de respecter au mieux la notion de distance Euclidienne.

PoiL2PoiRBiDir: (Point Of Interest Left To Point Of Interest Right BiDirectional)

Cette fonction consiste à apparier chacun des points d'intérêts de l'image source 1 à un point d'intérêt détecté dans l'image source 2. A partir des points appariés dans l'image source 2, on va les ré apparier avec les points d'intérêts de l'image source 1. Si on retombe sur les points d'intérêts orginaux (avec un écart de 1 pixel autorisé), c'est que l'appariement est bon. On le garde. Sinon on l'enlève. Cela permet d'épurer grandement les mauvais appariements.

Cela suppose qu'un bon appariement se fait dans les deux sens.

Un seuil peut être réglé sur chacune de ces fonctions pour garder les meilleurs appariements en terme de distance de descripteurs.

Ces fonctions sont inspirés de la fonction surf (mpi.cpp) que Mr Crouzil m'avait fourni et du livre "OpenCV 2 Computer Vision Application Programming Cookbook" opensource de Robert Laganière, qui est une excellente introduction à l'univers d'openCV.

Idée d'amélioration de conception et d'ajout de méthodes

On peut recoder toute ces méthodes pour les adapter à la comparaison de deux images de tailles différentes.

Mettre un CmakeList dans tous les modules de fonction.

Mettre les deux seuil éventuellement en paramètre avec des option

Pouvoir paramétrer les descripteurs avec des options (-d par exemple)

Ici on a codé des fonctions relativement petites, il aurait pu être utile de rajouter des .h ou de séparer le main en petites fonctions. Etant encore novice en C++ et cette étape nécessitant une connaissance plus approfondie des effets de bords et des pointeurs, je l'ai laissé de coté.

Le nombre de Kp pet augmenter ou diminuer après le passage dans le descripteur. Il serait bon d'améliorer ce point.

Dans bidirectionel, pourquoi dois-je redéclarer p1x,p1y, p2x, p2y et distance ? Parcequ'ils sont dans une boucle ?

→ problème lorsqu'on fait une déclaration dans un if ou une boucle, il faut redéclarer derrière, d'ou l'usage de pointeur

ajouter EucDist dans PoiL2PoiRBiDir

Grande série de Tests

Images testées

Pour réaliser ce test nous avons utilisé plusieurs images présentant des caractéristiques différentes. Le stage portant sur des paysages avec un faible décalage de pixel, on aurait pu se contenter de photos de ce type. Mais j'ai voulu pousser les tests plus loin et voir si les résultats sont bons avec des images ayant un fort décalage temporelle ou spatiale.

Les tests portent en premier lieu sur les images im0.ppm et im2.ppm situé dans le répertoire images/Autoroutes. Ce sont des images de paysage avec un faible décalage spatiale et un décalage temporelle assez important,puisqu'un pont a été construit entre temps.

Zone obscure du programme et autres problématiques

Il s'avère que OpenCV réalise parfois des traitements que je n'explique pas, et que je n'ai pas compris.

L'un d'eux est que après avoir obtenu nos points d'intérêts lors d'un passage dans le détecteur, le descripteur/extracteur a enlevé des points d'intérêts à la liste trouvé. Je suppose donc qu'il doit faire une sélection des points d’intérêts qu'il juge le plus intéressant, ou alors lorsque plusieurs points d'intérêts sont proche, utilise des maximas de fonctions diverses pour n'en garder qu'un seul.

Évaluation de l'efficacité de la bidirectionnalité

Dans ce paragraphe nous allons tenter d'évaluer l'efficacité de la bidirectionnalité.

Nous allons faire un comparatif des fonctions PoiL2PoiR et PoiL2PoiRBiDir. Aucune de ces deux fonctions n'utilise de voisinage, on a donc juste à changer le paramètre threshMatches.

Les tests sont visuels donc subjectif.

On prend un valeur de threshMatches à 50, seuil que j'estime satisfaisant pour une bonne qualité.

Comparaison entre im0.ppm et im1.ppm avec BruteForce-Hamming

	PoiL2PoiRBiDir				PoiL2PoiR			
	BRIEF	efficacité	ORB	efficacité	BRIEF	efficacité	ORB	efficacité
FAST	536 487 722 639 41 2	39/41 =95,1 %	536 467 722 627 40 10	30/40 =75 %	536 487 722 639 47 5	42/47= 89,4 %	536 467 722 627 48 15	33/48= 69 %
STAR	181 181 301 301 20 0	100 %	181 181 301 301 28 8	20/28= 71,4 %	181 181 301 301 20 0	100 %	181 181 301 301 31 10	21/31= 67,7 %
SIFT	1617 1484 1816	101/111 =91 %	Seg fault		1617 1848 1816	122/152 =80 %	Seg fault	

	1692 111 10				1692 152 30			
SURF	(peut changer) 1766 1698 2142 2064 169 15	154/169 91,1 %	(peut changer ! bizarre) 1766 1671 2142 2032 ~~11~~ ~~3~~ ~~12~~ ~~2~~ ~~9~~ 2	9/11= 81,8 %	1766 1698 2142 2064 253 a lot	0 %	1766 1671 2142 2032 345 a lot	0 %
ORB	702 702 702 702 ~~33~~ ~~56~~ ~~106~~ 0	100 %	702 702 702 702 ~~3~~ ~~1~~ 1	2/3= 66,7 %	200 200 200 200 12 0	100 %	200 200 200 200 17 2	15/17= 88,2 %
MSER	409 347 459 396 4 0	100 %	409 326 459 371 7 3	4/7= 57,1 %	409 347 459 396 5 0	100 %	409 326 459 371 8 5	3/8= 37,5 %

GFTT	1000 846 1000 847 69 0	100 %	1000 815 1000 825 59 14	45/59= 76,3 %	1000 846 1000 847 87 4	83/87= 95,4 %	1000 815 1000 825 66 18	48/66= 72,7 %
HARRIS	1000 869 806 721 71 2	69/71= 97 %	1000 845 806 715 61 13	48/61= 78,7%	1000 869 806 721 91 10	81/91= 89 %	1000 845 806 715 79 15	64/79= 81 %
SIMPLE BLOB (thresh Matches = 75)	173 150 153 132 6 4	2/6= 33,3 %	173 143 153 127 27 All false	0%	173 150 153 132 6 4	2/6= 33,3 %	173 143 153 127 37 practically all false	0 %

Le nombre de points matchés dépend de chaque utilisation du programme !!
(surtout SURF + ORB).

On remarque que la bidirectionnalité semble plus efficace (sauf dans le cas HARRIS+ORB)

FAST+BRIEF: les points bien appariés correspondent aux contours des montagnes / collines avec le ciel, aux villages au loin, aux routes.

Les deux points mal appariés correspondent à des contours de montagnes/collines avec le ciel

STAR+BRIEF : les points appariés correspond à des bordures de routes,deux bords de

buisson et un à la bordure pleine/forêt.

SIFT+BRIEF : les points appariés correspondent à des arbres, des points situés dans une limite séparé par la plaine et la forêt, des villages au loin, des routes. Les points mal appariés correspondent à un point dans le ciel, 3 points situés sur le contour des montagnes avec le ciel et 4/5 points situés dans la limite séparé par la plaine et la forêt.

SURF+BRIEF:Un peu pareil qu'avec SIFT mais avec plus de points trouvés.
Mal apparié limite ciel/coline:5
Mal apparié limite plaine/forêt:10

ORB+BRIEF : on remarque que seul 702 points d'intérêts ont été gardés
un point d'intérêt est apparié au sommet d'un arbre au loin dans le contour de la colline. 3 points ont appariés un village. Un bon appariement est situé dans la limite plaine forêt. Un a bien apparié un arbre. Les autres sont regroupés dans 3 zones correspondant à des routes. Aucune mauvaise correspondance.

MSER+BRIEF : Les 4 points bien appariés correspondent à des zones pas vraiment remarquable : point au milieu de la montagne, de la forêt, de la plaine... aucune caractéristiques probante mais bons appariements.

GFTT+BRIEF : Les points sont regroupés dans 5/6 zones principales :
une situé à la frontière colline/ciel, trois sur des routes spécifiques, et quelques uns dans la limite plaine/forêt

HARRIS+BRIEF : Similarités avec GFTT mais deux mauvais appariements

SIMPLE BLOB+BRIEF : Peut être 3 bons appariement. Le reste sont des mauvaises mises en correspondances.

Pour calculer l'efficacité, nous ferons le résultat de (nb de bons appariements/nb d'appariements totaux)

Les tests fait avec le descripteur ORB sont généralement moins bons
PoiL2PoiR trouve donc plus d'appariements que PoiL2PoiRBiDir
Dans ce test, on remarque que STAR,ORB,MSER et GFTT ont 100 % d'efficacité avec BRIEF pour PoiL2PoiRBiDir. Cependant MSER n'a trouvé que 4 points (peut être du au seuil trop faible), STAR 20 concentré en seulement 3 zones et ORB 33 répartie en 4 zones. GFTT lui a trouvé 69 points réparties dans à peu près toute l'image.
Pour ce cas là, avec ce seuil choisi on peut considérer que c'est le meilleur choix. Simple Blob donne des résultats mauvais, il est peut être mal paramétré. Quant à MSER, il ne fait que peu d'appariements ce qui fait qu'il a une efficacité très changeante.
Pour mieux différencier STAR, ORB ,MSER et GFTT, il faudrait utiliser des seuils plus haut.

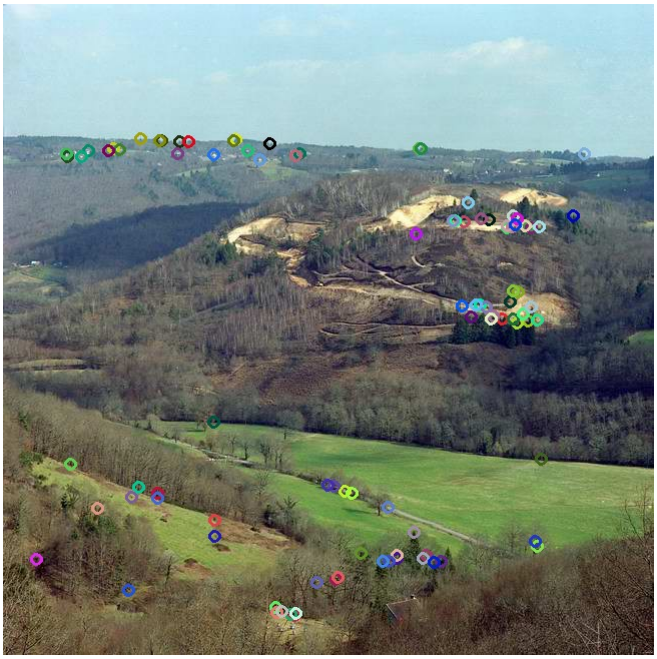
SURF+BRIEF et SURF+ORB donne bcp de points bien et mal apparié. On peut donc penser qu'un meilleur seuil permettrait de mieux évaluer l'efficacité.

	PoiL2PoiRBiDir descripteur BRIEF	
STAR	Nb matching=28 false=3	seuil=55
ORB	Nb matching=81 false=2	seuil=65
MSER	Nb Matching=19 false=3	seuil=65
GFTT	Nb matching=97 false=3	seuil=60
SIFT	Nb matching=33 false=4	seuil=35
SURF	Nb of matching=58 false=2	seuil=35

ORB GFTT et SURF comme détecteur avec BRIEF comme descripteur donnent des résultats intéréssant avec des bonnes valeurs de seuil.

ORB+BRIEF+BF-Hamming

GFTT + BRIEF + BF-Hamming



SURF+BRIEF+BF-Hamming

GFTT + BRIEF + BF-Hamming a l'air d'être la combinaison avec laquel on a trouvé le plus de « régions » remplies de points d'intérêts différents.

Nous n'avons pas encore testé les descripteurs SIFT et SURF. Après avoir effectué quelques tests avec les descripteurs SIFT et SURF, on s'aperçoit que les résultats sont très mauvais. Cela peut soit venir de la mauvaise méthode (PoiL2PoiRBiDir pas adapté) ou provenir d'un problème de codage (SIFT et SURF renvoient des 32FC1, et de plus, le nombre de point d'intérêt augmente après être passé dans le descripteur. Cependant quelques fois certains points sont bien appariés)

	SIFT(BF)	SIFT(FB)
FAST	Seuil=10000 nb match=8 false=5	Seuil=10000 nb match=8 false=3
STAR	mauvais	mauvais
SIFT	mauvais	Aucun appariement

SURF	(varie) 1766 2144 2142 2615 11	Seuil=500 9 false 4
ORB	mauvais	mauvais
MSER	Tres mauvais	
GFTT	mauvais	
HARRIS	mauvais	
SIMPLE BLOB (threshMatches = 75)	mauvais	

Il y a deux paramètres à choisir, EucDist et threshMatches.
Pour avoir une bonne idée du parametre EucDist, on peut essayer de calculer à la main le décalage et prendre le plus grand décalage observé. Généralement le décalage le plus grand est rencontré sur les objets situés au premier plan. Ici les photos sont assez proche on prend un décalage de 10.
Plus le décalage sera grand et plus le temps d'exécution sera important.
3min d' exécution

On fait donc une série de test avec PoiL2PixInNePoiL.

La méthode d'évaluation étant difficile à évaluer (les points sont proches on sait pas si bons appariements ou pas) on se base sur une impression de translation de la photo.

	SIFT(BF)	SURF(BF)	
FAST	Seuil = 200 536 635	Seuil=0,25 nb matches = 80 très bon résultat	

	nb of matches=80	malgrés quelques points mal appariés.	
STAR	Seuil=150 4 false=3	Seuil=0,3 nb matches=26 des points bien appariés et des points mal appariés. Moins efficace que FAST	
SIFT	Seuil=150 6 0 tres peu probant	Très très mauvais	
SURF	seuil=150 125 difficile à evaluer. Plutôt bon.	Très mauvais aussi	
ORB	Seuil=200 matches=42 false=1 bien plus rapide que les précédents et bons resultats apparents.	Seuil=0,3 matches=70 résultats sont bons	
MSER	Seuil=300 matches=22	Mauvais	

	Difficile à evaluer mais plutôt mauvais		
GFTT	Seuil=150 matches=100 beaucoup de points, bons pour une bonne part mais mauvais pour une autre	Seuil=0,3 matches=65 bons resultats	
HARRIS	Seuil=150 matches=167 pareil qu'au dessus	Seuil=0,3 matches=106 bons resultats	

Il est très difficile d'évaluer MSER, car MSER trouve des points qui n'ont pas des caractéristiques bien marquées pour l'œil humain. Le programme PoiL2PixInNePoiL utilisant un voisinage petit, il est difficile de juger si les points sont bien appariés.

Légende :

TD : type du descripteur

question :
Est-ce parce qu' on a plus de points d'intérêts que la méthode est forcément meilleurs? Une méthode qui fait moins d'appariement mais sur des zones diverses et dispersé est-elle meilleur qu'une méthode qui fait beaucoup d'appariements mais seulement dans quelques zones isolées ?

Nous : on juge les méthodes sur le pourcentage de bon appariement sur le nombre total de points détectés après le passage dans le descripteur.

Changer la fonction pour trier dans tous les progs.

J'ai confondu la fonction nth_element avec la fonction sort, ce qui a fait que j'ai du refaire de nombreux tests.

Documents utilisés

Les documents utilisés ont en grande partie étaient piochés à diverses adresses Web, en particulier la documentation d'openCV.

http://docs.opencv.org/

Surtout les modules features2D contenant les détecteurs, les descripteurs et les extracteurs.

http://docs.opencv.org/modules/features2d/doc/features2d.html

Néanmoins de nombreux problèmes se sont posés, notamment l'utilisation de fonctionnalité « non free » qui ont été breveté dans certain pays et qui sont donc limités dans leur utilisation.

(voir http://docs.opencv.org/modules/nonfree/doc/nonfree.html)

La documentation d'OpenCV n'est pas rigoureusement exact avec la version que j'avais sur leffe, j'ai donc utilisé une documentation trouvée sur un autre site pour comprendre de quoi quel objet dépendait-il, où le trouver, et donc comprendre l'héritage des différentes classe. Surtout les ressources venant du site ci dessous :

http://physics.nyu.edu/grierlab/manuals/opencv/classcv_1_1FeatureDetector.html

C'est donc une utilisation conjointe de ces deux documentations qui m'a permis de comprendre le fonctionnement et l'utilisation de différentes classes d'OpenCV.

Les notions d'héritage expliquées dans le livre « Exercices en langage C++ » de Claude Delanoy m'ont aussi permis d'utiliser ces fonctions.

Les incohérences d'openCV :

Mis à part la présence de module nonfree dont on se demande si ils sont présents ou non dans notre version d'OpenCV, il y a certaines choses étranges que j'ai remarqué.

Par exemple, pour le détecteur/descripteur Orb, il y a deux manières de l'invoquer. Soit en utilisant la fonction ORB présente dans features2D, soit en utilisant OrbFeatureDetector présente dans FeatureDetector et non repertorié sur ce site:

http://physics.nyu.edu/grierlab/manuals/opencv/classcv_1_1FeatureDetector.html

mais bien repertorié sur le site

http://www.opencv.org.cn/opencvdoc/2.3.1/html/modules/features2d/doc/common_interfaces_of_feature_detectors.html

! ATTENTION, cette page contient une erreur latex, selon votre navigateur et votre OS l'ouverture de cette page peut causer de graves problèmes comme l'ouverture de beaucoup beaucoup de fenêtre d'erreur, et vous obliger à killer le processus.

Les paramètres fournit aux deux fonctions ne sont pas les même. Pour la fonction ORB (ou plutôt la classe) on peut mettre en paramètre un seuil

```
ORB detector(200);
```

ORB(int **nfeatures**=500, float **scaleFactor**=1.2f, int **nlevels**=8, int **edgeThreshold**=31, int **firstLevel**=0, int **WTA_K**=2, int **scoreType**=ORB::HARRIS_SCORE, int**patchSize**=31)

```
detector(image1, Mat(), keypoints1);
```

Pour la fonction OrbFeatureDetector, le constructeur ne prends qu'un parametre
OrbFeatureDetector(size_t n_features);

et s'utilise ainsi

```
OrbFeatureDetector detector;
        detector.detect(image1, keypoints1);
```

SIFT

SiftFeatureDetector(double threshold, double edgeThreshold,

```
                        int nOctaves=SIFT::CommonParams::DEFAULT_NOCTAVES,
                        int
nOctaveLayers=SIFT::CommonParams::DEFAULT_NOCTAVE_LAYERS,
                        int firstOctave=SIFT::CommonParams::DEFAULT_FIRST_OCTAVE,
                        int angleMode=SIFT::CommonParams::FIRST_ANGLE );
```

SIFT(int **nfeatures**=0, int **nOctaveLayers**=3, double **contrastThreshold**=0.04, double **edgeThreshold**=10, double **sigma**=1.6)

les paramètres ne sont donc pas les mêmes. Il est donc important de

faire attention à la méthode utilisée et les paramètres transmis.
De plus, la fonction BRISK est censée se trouver dans Feature2D, elle n'y ait pas, et la class FREAK hériterait de DescriptorExtractor, ce qui semble ilogique.
(vu sur http://docs.opencv.org/modules/features2d/doc/feature_detection_and_description.html)

La classe descriptorMatcher est censé héritée de la classe Algorithme. Cette classe est introuvable.

Il faut faire attention aux types de donnés, parfois on a de CV_8UC1 et des fois 32FC1

Index :

logiciels utilisés:

MATLAB : Langage de programmation et environnement de développement utilisé pour des calculs numérique. Il permet de manipuler matrices, courbes, données, mettre en œuvre des algorithmes, créer des petites interfaces utilisateurs.

OpenCV : (pour Open Computer Vision) est une bibliothèque graphique libre spécialisée dans le traitement d'impages en temps réel. Elle est écrite en C++, Python et Java. Nous l'utilisons dans le cadre de notre stage avec le langage C++.

Evernote: Logiciel permettant de prendre des notes, mettre des images, vidéos, page web. Il est disponible dans un navigateur donc est multi-plateforme et permet d'être synchronisé.

http://indentcode.net/ : outil en ligne permettant d'indenter le C++. En effet sur Gedit ce n'est pas très pratique. Attention tout de même, cet outil transforme les sauts de ligne « \n » en « n ».

Dropbox : service de stockage et de partage de fichier en ligne.

Astuces utilisés :

Pour écrire le rapport de stage, entre libre office et open office c'était pas évident. Au début en copiant collant une image j'ai corrompu mon document, résultat je ne pouvais plus l'ouvrir, j'ai du le supprimer et perdre mon travail.

J'ai donc décidé d'utiliser conjointement la tour de l'IRIT et mon MacBook. Comme les documents que j'ai édité sur OO avec mon mac ne sont pas éditable avec LO sur la tour et vice-versa (droit d'écriture, vice-versa) J'ai deux version du rapport de stage. J'ai donc juste à copier l'ancienne version et la coller sur la nouvelle. Ainsi j'ai toujours une copie du rapport ce qui me permet non seulement de sauvegarder mon travail mais en plus d'éditer les documents sur les deux machines.

« Le contenu du signet ou de la sélection est en lecture seule. Aucune modification permise. »
J'ai eu une erreur de ce type. J'ai donc du aller dans format/sections et décocher « protégé »

g++ cornerHarris_Demo.cpp `pkg-config --cflags --libs opencv`

./a.out ../images/A89-01-0.jpg ../images/A89-01-1.jpg FAST BRIEF BruteForce-Hamming

il faut dire lorsqu'il y a un problème de type

faire les Cmake files pour toutes les fonctions

voir pour rows et cols si on peut pas faire mieux dans la fonction interface, i.e prendre le min.