TP contour deformable : les snakes

4 mars 2015

1 Introduction

1.1 Objectif

Ce TP a pour objectif de vous faire apprehender une technique de detection de contour actif : snake. Les conposantes de ce TP sont les suivantes :

1 Implementation C/C++ d'un algorithme de snakes
2 Caracterisation de l'algorithme en fonction des parametres
3 modification du modele

1.2 A rendre

Consigne à observer Le TP se fait en C/C++ pour la partie algorithmique et sous *octave* pour la visualisation de l'evolution des snakes. Aucun IDE n'est impose, cependant, si vous en utiliser un, nous conseillons fortement *Qtcreator* (le fichier cmake est fourni).

Consignes pour le rendu a rendre avant le 15 mars 23h55

À rendre sur le dépôt sur e-campus une archive nommée : TP_SNAKE_NOM1_NOM2.tar. Pour générer cette archive, utilisez la commande :

tar -cvf TP_SNAKE_NOM1_NOM2.tar fichier1 fichier 2... fichierN

qui génère l'archive TP_SNAKE_NOM1_NOM2.tar contenant les fichiers fichier1, fichier2 jusqu'à fichierN. Le contenu de l'archive est le suivant :

- un compte rendu qui explique ce que fait votre code en 5 ligne, et une page pour expliquer vos experiences
- votre code pour calculer l'evolution du snake (il faut que le code compile) qui comporte les fichier sources, un fichier pour compiler (makefile ou cmake) et les resources (images).
- un script pour afficher vos resultats.

Dans le cas ou votre script affiche des figures, n'oubliez pas de mettre des titres, des légendes, les labels, les conditions d'exécution, les paramètres de l'algorithme... tout ce qui permet de comprendre la figure car elle ne parle pas d'elle même (on voit clairement sur la figure... à bannir car le lecteur ne voit pas la même chose que vous).

Si votre script affiche des résultats dans la console, vous pouvez utiliser la commande :

printf('mon texte avec des variables %i %f', var1, var2)

qui vous permet d'afficher proprement du texte dans la console avec des valeurs (comme la fonction *printf* en C). Encore une fois, les valeurs ne parlent pas d'elle mêmes. Elles ont besoin d'avoir une interprétation. Est-ce qu'elles appartiennent à un domaine spécifique, est-ce des valeurs grandes/petites par rapport à quelque-chose, ont elles des unités, est-ce cohérent avec

ce à quoi vous vous attendiez et pourquoi, est-ce une valeur aléatoire ou déterministe, est-ce un paramètres, est-ce une valeur moyenne, y a-t-il besoin d'autant de chiffres après la virgule (significatif)... bref, expliciter les valeurs.

Remarque : touts les codes et scripts doivent etre commentes.

2 Rappel de cours

2.1 Formulation du probleme

Cahier des charges On souhaite segmenter des objets dans une image en les entourant avec une courbe parametree fermee. On va donc utiliser les outils mathematiques lies aux courbes parametriques. On rappel qu'une courbe (Γ) peut se parametree par l'application γ :

$$\begin{aligned} \gamma : [0,1] &\to \mathbb{R}^2 \\ s &\mapsto \begin{pmatrix} \gamma_x(s) \\ \gamma_y(s) \end{pmatrix} \end{aligned}$$

ou $\gamma_x(s)$ et $\gamma_y(s)$ sont les coordonnees de la courbe.

Une courbe avec de bonnes proprietes Deux proprietes sont desirees pour un courbe : la continuite et la regularite (lisse). La continuite se traduit par une norme de la derivee qui doit etre fini et la regularitee impose des derivees secondes fini en norme. Pour definir une energie penalisant les mauvaise configuration Γ, on ajoute toute les petites contributions de la norme de la derivee et de la derivee seconde sur le long de la courbe en ponderant les deux termes :

$$\mathcal{E}_{\text{int}}(\gamma) = \int_0^1 \frac{1}{2} \left(\lambda_1 \|\gamma'(s)\|_2^2 + \lambda_2 \|\gamma''(s)\|_2^2 \right) \mathrm{d}s$$

avec λ_1 et λ_2 des reels ponderant les effets de chaque terme.

Collage au contour... Une autre bonne propriete de la courbe est quelle soit collee aux contour des objets que l'on souhaite segmenter. Pour cela, on cherche un critere qui se minimise lorsaue la courbe vient se position nne sur le contour de l'objet segmente. Plusieurs solutions sont possibles, parmis lequelles on tourve :

1 la norme du gradient de l'image (son oppose) qui passe par un extremum sur les changements brute d'intensite qui ont tendence a representer des contours.
2 l'image elle meme dans le cas de dessin : contours lineaires fins fonce sur fond clairs
3 la transformee de distance dans le cas des images binaires : le gradient etant nul loin des contours (2 pixels) la courbe ne sera pas attiree par les zones de transitions, cependant, la transformee de distance est elevee lorsqu'on est loin des contours d'un objet et nul dans l'objet (point a prendre en compte pour ne pas laisse le snake se contracter en une singularite).

Pour le premier cas, on defini l'energie externe comme la contribution de tout les lieux de la courbe pondere par la valeur de la norme du gradient (la circulation du gradient le long de la courbe), et on l'ecrit :

$$\mathcal{E}_{\text{ext}}(\gamma) = \int_0^1 -\lambda_3 \left(\frac{\partial \mathrm{I}}{\partial x}^2 + \frac{\partial \mathrm{I}}{\partial y}^2 \right) (\gamma) \mathrm{d}s$$

avec λ_3 un reel positif ponderant la contribution de l'energie liee a l'image.

Au total, nous avons la formulation suivante de l'enegie :

$$\begin{aligned}\mathcal{E}_{\text{tot}}(\gamma) &= \int_0^1 \frac{1}{2}\left(\lambda_1\|\gamma'(s)\|_2^2 + \lambda_2\|\gamma''(s)\|_2^2\right) - \lambda_3\left(\frac{\partial \mathrm{I}}{\partial x}^2 + \frac{\partial \mathrm{I}}{\partial y}^2\right)(\gamma)\mathrm{d}s \\ &= \int_0^1 \mathrm{E}_{\text{int}}(\gamma', \gamma'') + \mathrm{E}_{\text{int}}(\gamma)\mathrm{d}s\end{aligned}$$

Pour minimiser $\mathcal{E}_{\text{tot}}$, nous allons avoir recours aux equations d'Euler-Lagrange.

2.2 Minimisation de fonctionnelle par equation d'Euler-Lagrange

On entend par fonctionnelle la generalisation des fonctions aux domaine des fonction. Par exemple, si f est une fonction de $\mathcal{C}^\infty(\mathbb{R}^n, \mathbb{R}^m)$ l'espace des fonction continue de $\mathbb{R}^n$ dans $\mathbb{R}^m$, alors $G(f, f', f'') = f + f' + f''$ est une fonctionnelle. Dans notre cas, la fonctionnelle est $\mathcal{E}$. Pour cette definition integrale, les eauqtions d'Euler-Lagrange donne un critere pour minimiser la fonctionnelle $\mathcal{E}$ qui est :

$$\gamma \in \underset{\gamma\in\mathcal{C}^\infty([0,1],\mathbb{R}^2)}{\arg\min} \int_0^1 \mathrm{E}_{\text{tot}}(\gamma, \gamma', \gamma'')\mathrm{d}s \quad \text{tel que} \quad \frac{\partial E_{\text{tot}}}{\partial \gamma} - \frac{\mathrm{d}}{\mathrm{d}s}\left(\frac{\partial E_{\text{tot}}}{\partial \gamma'}\right) + \frac{\mathrm{d}^2}{\mathrm{d}s^2}\left(\frac{\partial E_{\text{tot}}}{\partial \gamma''}\right) = 0$$

ce qui se traduit par l'equation d'equilibre suivante :

$$\forall s \in [0,1], \quad 0 = \lambda_1\gamma''(s) - \lambda_2\gamma^{(4)}(s) + \lambda_3\nabla(\|\nabla I\|^2)$$

Avec cette formulation, on a une condition sur la courbe lorsqu'elle est deja sur le coutour de l'objet lorsqu'elle est stationnaire, mais on ne peut pas la faire bouger. On introduit alors un autre parametre, le temps t, pour ecrire une equation d'evolution :

$$\forall s \in [0,1], t \in \mathbb{R}^+, \quad \frac{\partial \gamma}{\partial t}(s,t) = \lambda_1\frac{\partial^2\gamma}{\partial s^2}(s,t) - \lambda_2\frac{\partial^4\gamma}{\partial s^4}(s,t) + \lambda_3\nabla(\|\nabla I\|^2)(\gamma(s,t)) \qquad (1)$$

On dispose donc de deux equation scalaire pour decrire l'evolution de la courbe, une pour chaque composante de la courbe.

2.3 Discretisation et evolution temporelle

L'equation (1) etant sous sa forme continue, il faut la discretiser pour la rendre utilisable. Il faut donc faire deux type de dicretisation, une temporelle et une spatiale. On notera avec un exposant k les iterations temporelles et un indice n les reperes spatiaux. On utilise des notation vectorielles pour les coordonnees de la courbe, γ_x^k et γ_y^k toutes les coordonnees discretes de la courbe a l'iteration k :

$$\gamma_x^k = (\gamma_{x,0}^k, \gamma_{x,1}^k \ldots, \gamma_{x,N-1}^k)^t, \quad \text{et} \quad \gamma_y^k = (\gamma_{y,0}^k, \gamma_{y,1}^k \ldots, \gamma_{y,N-1}^k)^t$$

avec N le nombre de point de discretisation de la courbe. A partir de ces vecteurs, on peut ecrire de facon matricielle les operations de derivations. On note la derivee seconde et quatrieme

respectivement D_2 et D_4.

$$D_2 = \begin{pmatrix} -2 & 1 & 0 & \dots & 0 & 1 \\ 1 & -2 & 1 & & 0 & 0 \\ 0 & 1 & -2 & \ddots & & \\ & & \ddots & \ddots & & \\ 0 & 0 & & & -2 & 1 \\ 1 & 0 & \dots & & 1 & -2 \end{pmatrix} \quad \text{et} \quad D_4 = \begin{pmatrix} 6 & -4 & 1 & 0 & \dots & 1 & -4 \\ -4 & 6 & -4 & \ddots & & 0 & 1 \\ 1 & -4 & 6 & \ddots & \ddots & & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & & 0 \\ 0 & & \ddots & \ddots & & & 1 \\ 1 & 0 & & & & -6 & -4 \\ -4 & 1 & 0 & \dots & 1 & -4 & 6 \end{pmatrix}$$

La discretisation temporelle se fait simplement en prenant la difference entre deux instant :

$$\frac{\partial \gamma}{\partial t} \simeq \frac{\gamma^k - \gamma^{k-1}}{\Delta t}$$

Au total, on trouve pour la composante x :

$$\frac{\gamma_x^k - \gamma_x^{k-1}}{\Delta t} = (\lambda_1 D_2 - \lambda_2 D_4)\gamma_x^k + \lambda_3 \nabla(\|\nabla I\|^2)_x \quad (2)$$

$$\frac{\gamma_y^k - \gamma_y^{k-1}}{\Delta t} = (\lambda_1 D_2 - \lambda_2 D_4)\gamma_y^k + \lambda_3 \nabla(\|\nabla I\|^2)_y \quad (3)$$

et donc, la formule de recurtion est

$$(I + \Delta t(\lambda_2 D_4 - \lambda_1 D_2))\gamma_x^k = (\gamma_x^{k-1} + \Delta t \lambda_3 \nabla(\|\nabla I\|^2)_x) \quad (4)$$

$$(I + \Delta t(\lambda_2 D_4 - \lambda_1 D_2))\gamma_y^k = (\gamma_y^{k-1} + \Delta t \lambda_3 \nabla(\|\nabla I\|^2)_y) \quad (5)$$

avec I la matrice identite. On obtient donc un systeme lineaire pour chaque composante de la courbe.

3 Pratique

3.1 Les outils

Pour ce TP, on vous demande de realise les calculs en C/C++ et la visualisation avec octave.

Nous mettons a votre disposition une classe *C_ matrix.h* pour gerer les matrices et faire des operations simples comme l'addition la soustraction, la multiplication, ainsi que des operations plus complexes comme l'inversion de matrice, la convolution, le calcul de gradient...

Exemple 1 Exemple de code pour la creation d'une matrice :

```
#include <C_matrix.h>//gestion des operations matricielles
#include <iostream>

int main(int argc, char *argv[])
{
    //creation d'un objet
    C_matrix<double> my_matrix(3,5);
    //initialisation random de la matrice
    my_matrix.randomf();
   //affichage sur la console
    my_matrix.show();
    //sauvegarde de l'objet dans un fichier ``myMatrixFile''
    my_matrix.save("myMatrixFile");

    //affichage de la somme de tout les elements de my_matrix
    std::cout << my_matrix.sum() << std::endl;

    return 1;
}
```

Exemple 2 Pour le chargement et l'affichage d'image, la classe *C_ imgMatrix* qui herite de la classe *C_ matrix* utilise la librairie *CImg* pour charger et afficher les images. La librairie *CImg* peut etre installer sur votre ordinateur avec la commande

apt-get install cimg-dev

ou si vous n'avez pas les droits necessaires, vous pouvez cloner le repository git de *CImg* en tapant la commande suivante :

git clone http ://git.code.sf.net/p/cimg/source CImg

Remarque : *git* est un gestionnaire de version. Les gestionnaire de version servent a gerer les version d'un projet et de pouvoir revenir a tout moment a une version precedente pour pouvoir retrouver un code/version qui fonctionne. Je vous recommande fortement d'utiliser des gestionnaire de version dans tout ce que vous faites. Par exemple, vous commencez un TP, vous creez un repository que vous actualiserez tout au long de la seance de sorte a ce que vous puissiez retrouver une version qui fonctionne. Pour plus d'information concernant *git*, je vous renvoie vers le lien ou un autre lien.

Exemple de code pour charger une image a l'aide de la classe *C_ imgMatrix* :

```
#include <C_imgMatrix.h> //inclu la library de chargement d'image CImg
#include <iostream>

int main(int argc, char *argv[])
{
    if(argc!=2)
    {
        std::cout << "Commande : " << argv[0] << " fileName<string>" << std::endl;
        return -1;
    }
    //charge l'image argv[1] dans la matrice myImg
    C_imgMatrix<double> myImg(argv[1]);
    //montre l'image
    myImg.display(0);
    //calcul le gradient dans la direction X
    C_imgMatrix<double> myGradX = myImg.gradX();
    //affiche l'image du gradient
    myGradX.display(1);
    return 1;
}
```

presentation des classes a utiliser documentation principale et hello world optionel git ? il faut qu'il m'envoie leur clef ssh

3.2 Question

3.2.1 Charger et afficher une image

Tester le chargement d'une image avec la classe *C_ imgMatrix* et afficher votre image.

3.2.2 Effectuer des operations simple

Filtrez une image par un noyau gaussien (Attention ! Le noyau doit etre de dimension impaire). Affichez le resultats. Affichez la valeur absolue de la difference entre l'image d'origine et l'image filtree. Sur une image binarisee, utilisez la methode *bwdistEuclidean* pour calculer la carte de distance, la visualiser.

3.3 Generation de deux courbes

Pour des soucis de simplicite conceptuelle, nous utiliserons des objets qui encoderont les points et les courbe, respectivement les classes *C_ point* et *C_ curve*. La classe *C_ point* contient des attributs de position x et y, de vitesse vx et vy ainsi que d'energie image Px et Py. La classe *C_ curve* contient un vecteur de point ainsi que deux methodes pour initialiser la positions des points de la courbe, une en forme de cercle, l'autre en forme de carre. Regardez rapidement de quoi sont faites ces classes avant de commencer. Dans la classe *C_ curve*, creez deux methodes pour extraire les coordonnees x et y de la courbe et les stocker dans deux matrices colone. Creez une matrice pour calculer la derivee de la courbe, calculez la derivee et sauvegardez les resultats dans des fichiers. Calculez de l'energie interieure de la courbe au sens des snakes. Sous octave, chargez les fichiers et les afficher a l'aide des fonction *load* et *plot*.

3.4 Energie interieure du Snake

Dans la classe *C_ snakes*, dont un squelette est propose, implementez deux methodes qui genere les matrices de derivation $D1$ et derivation seconde $D2$. Les deux matrice $D1$ et $D2$ sont des attributs de la classe. Creez ensuite une methode qui realise le calcul de l'energie interne de la courbe.

3.5 Carte d'energie exterieur

Dans la classe *C_ snakes*, creez une methode qui calcul l'energie image en chaque point de l'image, ainsi que les composantes du gradient de l'energie image. On stockera l'energie image dans *imgEnergy* et les composante du gradient dans les attributs PX et PY.

3.6 Evolution temporelle du snake

Creez une methode qui genere la matrice de derivee $4^{\text{ème}}$ $D4$. Implementez une methode qui genere le systeme defini a l'equation (4) et (5) et qui stocke l'inversion du systeme dans la matrice Ainv. Creez une methode qui met a jour les attributs Px et Py de chaque points de la courbe. Implementez deux methodes qui retourne les second menbre des equations (4) et (5). Pour faire evoluer la courbe depuis un point de depart fixe, definissez une methodes qui initialise tout les elements necessaire a l'evolution, puis creez une methode qui encode une pas temporelle de l'evolution du snake. Enfin, rassemblez tout les elements pour code auparavant de sorte a ce qu'une seule methode soit appellee dans le *main*.

Testez votre code sur les images qui vous sont fournies et determinez les parametres λ_1, λ_2, λ_3 et dt pour ces images. Visualisez les effets de chaque parametre pour les differentes images. Montrez l'evolution du snake et de l'energie totale en fonction du temps.

sur les differentes images, faire evoluer le snake et trouver des bons parametres pour trouver les contours desires

3.7 Pour aller plus loin

Si vous avez le temps pendant la seance de TP, vous pouvez explorer les pistes suivantes :

1 Changez la forme de l'energie interieure

2 Appliquez une force de pression sur le snake en ajoutant une force exterieure selon la normale a la courbe. Vous povez vous referez a l'article On active contour models and ballons

3 Pour arriver a detecter les contours dans une forme concave, vous pouvez utiliser une technique qui consiste a faire evoluer le snake dans un champ de vecteur. Vous pouvez vous referer a la publication Snakes, shapes, and gradient vector flow