Vision par ordinateur: algorithmes et applications
Projets 2004/2005

Renaud.Keriven@ens.fr
http://cermics.enpc.fr/~keriven/

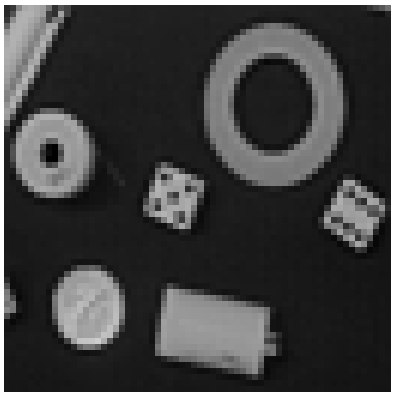

FIG. 1 – Contours actifs

1 Contours actifs

On extrait d'une image les contours des objets à l'aide d'une courbe qui évolue suivant une Équation aux Dérivées Partielles et qui se stabilise sur les objets.

1.1 Évolution d'une courbe fermée suivant une EDP : Level Sets

La courbe $\mathcal{C}(p,t)$ évolue suivant une vitesse normale β :

$$\frac{\partial \mathcal{C}}{\partial t} = \beta \mathbf{n}$$

On n'implémente pas l'évolution telle quelle mais en considérant une fonction $\phi(x,y,t)$ dont $\mathcal{C}(p,t)$ est le niveau zéro :

$$\forall t,\ \mathcal{C}(.,t) = \{(x,y) | \phi(x,y,t) = 0\}$$
$$\frac{\partial \phi}{\partial t} = \phi_t = \beta |\nabla \phi|$$

Comme fonction ϕ initiale, on prend la fonction distance à la courbe initiale, positive à l'extérieur, négative à l'intérieur.

1. Choisir un domaine discret pour représenter ϕ. On pourra utiliser une image à valeurs réelles, même si ϕ n'est pas une image. Calculer la fonction distance à un cercle et celle à un rectangle.
2. Savoir afficher les pixels correspondant au niveau 0 de ϕ (sans détermination précise du niveau zéro par interpolation). Prévoir de "grossir" l'affichage, de façon à afficher dans une fenêtre de taille suffisante même si la discrétisation de ϕ est de petites dimensions.
3. Faire évoluer ϕ (et donc la courbe) avec différents β. Les schémas numériques associés sont :

– $\beta = 1$
Prendre pour la dérivée de ϕ

$$\phi_t = \sqrt{d_x^2 + d_y^2}$$

avec les dérivées spatiales (pour $\Delta x = \Delta y = 1$)

$$D^{+x}(i,j) = \phi(i+1,j) - \phi(i,j)$$
$$D^{-x}(i,j) = \phi(i,j) - \phi(i-1,j)$$
$$idem\ en\ y$$
$$d_x = \max(0, D^{+x}, -D^{-x})$$
$$d_y = \max(0, D^{+y}, -D^{-y})$$

les conditions aux limites "miroir"

$$\phi(-1,j) = \phi(1,j)$$
$$\phi(W,j) = \phi(W-2,j)\ (si\ i \in [0, W-1])$$
$$idem\ en\ y$$

et le schéma explicite en temps

$$\phi(i,j,t+\Delta t) = \phi(i,j,t) + \phi_t(i,j)\Delta t$$

Attention au choix de Δt

– $\beta = \kappa$ (courbure, lissage euclidien)

$$\phi_t = 0\ si\ (D^x)^2 + (D^y)^2 = 0$$
$$\phi_t = \frac{K}{(D^x)^2 + (D^y)^2}\ sinon$$
$$avec$$
$$K = (D^x)^2 D^{yy} - 2D^x D^y D^{xy} + (D^y)^2 D^{xx}$$
$$D^x = (\phi(i+1,j) - \phi(i-1,j))/2$$
$$D^{xx} = \phi(i+1,j) - 2\phi(i,j) + \phi(i-1,j)$$
$$D^{xy} = (\phi(i+1,j+1) + \phi(i-1,j-1) - \phi(i-1,j+1) - \phi(i+1,j-1))/4$$

– $\beta = \kappa^{\frac{1}{3}}$ (lissage affine)

$$\phi_t = K^{\frac{1}{3}}$$

4. Vérifier qu'un rectangle reste un rectangle dans le premier cas, devient un cercle dans le deuxième cas et une ellipse dans le troisième cas.

1.2 Extraction de contours

Partir de l'image `obj.pgm` que nous noterons I. On calcule le critère d'arrêt :

$$g = \frac{1}{1 + c_1|\nabla I|^2}$$

On utilise maintenant, pour minimiser le somme de g le long de la courbe, la vitesse

$$\beta = g(c_2 + \kappa) + \nabla g . \frac{\nabla \phi}{|\nabla \phi|}$$

ϕ est maintenant discrétisée sur la même grille que les pixels de I. En posant

$$minmod(a,b) = \begin{cases} \min(a,b) & \text{si } a > 0 \text{ et } b > 0 \\ \max(a,b) & \text{si } a < 0 \text{ et } b < 0 \\ 0 & \text{sinon} \end{cases}$$

Le schéma numérique associé est :

$$d'_x = minmod(D^{-x}, D^{+x})$$
$$d'_y = minmod(D^{-y}, D^{+y})$$
$$\phi_t = (\frac{\partial g}{\partial x} d'_x + \frac{\partial g}{\partial y} d'_y) + g(c_2\sqrt{d_x^2 + d_y^2} + \frac{K}{(D^x)^2 + (D^y)^2})$$

– Essayer d'ajuster convenablement (c_1, c_2), le pas de temps, et le lissage intervenant dans g et ses dérivées...
– Modifier l'affichage pour voir l'image en même temps que la courbe qui évolue.

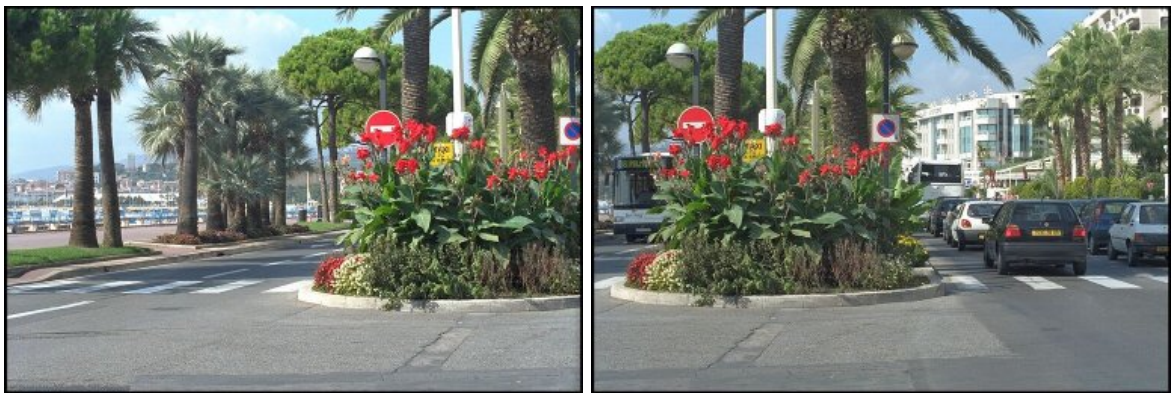

FIG. 2 – Stitching

2 Mosaïques ("Stitching")

Il s'agit de reconstituer une seule grande image à partir de plusieurs petites pour reconstituer une image panoramique plane, cylindrique , ou sphérique (plane dans notre cas). On utilisera les images dans `Data/Mosaic`, en commençant par `Cannes/miniimage0003.jpg` et `Cannes/miniimage0004.jpg`.

1. **Objectif** : Deux images $I_1(x_1, y1)$ et $I_2(x_2, y_2)$ prises d'un même point sont liées par une homographie :

$$(x_2, y_2) = H(x_1, y_1) = (\frac{h_0 x_1 + h_1 y_1 + h_2}{h_6 x_1 + h_7 y_1 + 1}, \frac{h_3 x_1 + h_4 y_1 + h_5}{h_6 x_1 + h_7 y_1 + 1})$$

   Il s'agit d'estimer cette homographie automatiquement (8 paramètres). On pourrait se contenter de quatre appariements mais on va plutôt l'estimer de manière robuste à partir du plus d'appariements possible. Une fois H connue, il ne restera plus qu'à reconstituer une grande image.

2. **Points de Harris :** on extrait des points particuliers caractéristiques, par exemple les "coins", en utilisant le détecteur de Harris.
   - Soit I(x,y) la luminosité du point (x,y)
   - $A(x, y) = \frac{\partial I}{\partial x}(x, y)^2$
   - $B(x, y) = \frac{\partial I}{\partial y}(x, y)^2$
   - $C(x, y) = \frac{\partial I}{\partial x}(x, y) \times \frac{\partial I}{\partial y}(x, y)$
   - $\tilde{A}$, $\tilde{B}$ et $\tilde{C}$ des lissages de A, B et C.
   - $M(x, y) = \begin{pmatrix} \tilde{A} & \tilde{C} \\ \tilde{C} & \tilde{B} \end{pmatrix}$
   - $H(x, y) = det(M) - k\ tr(M)$ où k est une constante ($k = 0.04$ marche bien !)
   - Les "coins" sont les maxima locaux de H de valeur supérieure à un certain seuil.

3. **Corrélation :** On essaie de trouver le correspondant dans la deuxième image par corrélation. Pour un pixel (x_1, y_1) de I_1, on estime la "ressemblance" du pixel (x_2, y_2) de I_2. Pour cela, on compare les valeurs des pixels autour de (x_1, y_1) et de (x_2, y_2). Plus précisément, on mesure la corrélation entre deux "fenêtres" autour de ces points. On définit par étapes la corrélation C_{12} :
   - Moyennes :

$$\bar{I}_i(x_i, y_i) = \frac{1}{(2N+1)^2} \sum_{u=-N}^{N} \sum_{v=-N}^{N} I_i(x_i + u, y_i + v)$$

   - Produit scalaire :

$$\langle I_i, I_j \rangle(x_i, y_i, x_j, y_j) = \frac{1}{(2N+1)^2} \sum_{u=-N}^{N} \sum_{v=-N}^{N} (I_i(x_i + u, y_i + v) - \bar{I}_i(x_i, y_i)) \\ (I_j(x_j + u, y_j + v) - \bar{I}_j(x_j, y_j))$$

   - Norme :

$$|I_i|(x_i, y_i) = \sqrt{\langle I_i, I_i \rangle(x_i, y_i, x_i, y_i)}$$

   - Corrélation-croisée normalisée :

$$C_{12}(x_1, y_1, x_2, y_2) = \frac{\langle I_1, I_2 \rangle(x_1, y_1, x_2, y_2)}{|I_1|(x_1, y_1)|I_2|(x_2, y_2)}$$

   qui vérifie $-1 \leq C_{12} \leq 1$ (ce qui est un bon test de correction du programme !)

Pour un point (x_1, y_1), on pourrait choisir alors comme point correspondant le point (x_2, y_2) maximisant la corrélation $C_{12}(x_1, y_1, x_2, y_2)$.

4. **Least Median Squares (LMS) :** On va en fait chercher entre les deux images toutes les paires de points de Harris qui corrèlent "bien" (un point de I_1 peut donc être associé à plusieurs points de I_2). On estime ensuite l'homographie avec l'algorithme LMS, capable de trouver la meilleure homographie tout en détectant les faux appariements. Pour cela, il faut fournir au programme (cf exemple) :
   – Pour quatre paires données, le calcul de H.
   – Pour un H donné, le calcul de l'erreur entre les (x_2, y_2) et les $H(x_1, y_1)$
5. On colle enfin les deux images.

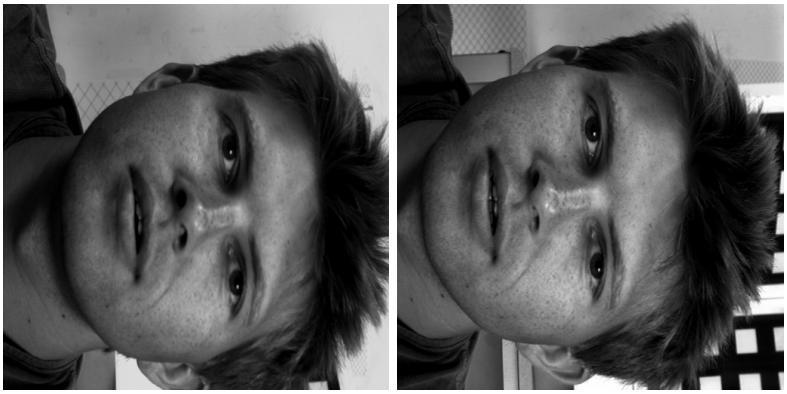

FIG. 3 – Stéreo

3 Stéreovision par corrélation

On considère deux images d'une même scène. Le but est de retrouver la forme des objets vus, modélisée par le graphe d'une fonction $z = f(x, y)$. Plus précisément, pour chaque pixel (x_1, y_1) de l'image 1, on veut retrouver la profondeur z associée. L'idée est de retrouver sur l'épipolaire de (x_1, y_1) dans l'image 2 le point (x_2, y_2) qui lui correspond "le mieux". Avant cela, on commence par rectifier les images... Les matrices des deux caméras sont connues.

3.1 Données

On partira des images dans `Data/Stereo`. Les `.gif` sont les images et les `.mat` les matrices des caméras, au format ASCII.

3.2 Rectification

Le but de la rectification est de faire en sorte que les droites épipolaires correspondent aux horizontales des images : l'épipolaire dans l'image j du point (x_i, y_i) est la droite $y_j = y_i$. Il suffit pour cela de reprojeter les images dans un même plan rétinien $\mathcal{R}$ parallèle à la (C_1C_2) joignant les centres optiques. Afin de déformer le moins possible les deux images, on choisira pour $\mathcal{R}$ un plan également parallèle à $\mathcal{R}_1 \cap \mathcal{R}_2$. Il reste, pour définir complètement ce plan, à lui choisir un point. On prendra par exemple le point $O = (0, 0, 0)$. Les formules sont donc les suivantes :

- Les points 3D sont initialement dans un repère $(O, \mathbf{I}, \mathbf{J}, \mathbf{K})$. On cherche un nouveau repère $(O, \mathbf{i}, \mathbf{j}, \mathbf{k})$ tel que le plan $\mathcal{R}$ soit $(O, \mathbf{i}, \mathbf{j})$.
- Horizontale : $\mathbf{i} = (C_1C_2)/|(C_1C_2)|$
- Intersection de $\mathcal{R}_1$ et $\mathcal{R}_2$: $\mathbf{j}' = \mathbf{n}_1 \wedge \mathbf{n}_2$ où les $\mathbf{n}_i$ sont les normales aux rétines initiales (que l'on retrouve, non normalisées, au début de la dernière ligne des matrices des caméras).
- Normale : $\mathbf{k} = \mathbf{i} \wedge \mathbf{j}'$
- Verticale : $\mathbf{j} = \mathbf{k} \wedge \mathbf{i}$

Conseils :

- A un point $m = (u, v)$ d'une image initiale correspond un point (α, β) de la nouvelle. Soit M le point 3D se projetant en (u, v) (cf notes de cours : $M = P^{-1}(\lambda\tilde{m} - \tilde{p})$) et qui appartient à $\mathcal{R}$ (c'est-à-dire tel que $M.\mathbf{k} = 0$), on a : $\alpha = M.\mathbf{i}$ et $\beta = M.\mathbf{j}$.
- A un point (α, β) de la nouvelle image correspond dans une image initiale le point (u, v) obtenu par projection du point 3D $M = \alpha\mathbf{i} + \beta\mathbf{j}$.

– Transformer α et β pour stocker les images rectifiées dans des intervalles “convenables” à coordonnées entières (tout en conservant les épipolaires sur une même horizontale).
– Les images rectifiées sont à valeurs réelles, interpolées à partir de celles des images départ.

3.3 Algorithme

A un point (u_1, v_1) de la première image rectifiée, correspond un point (u_2, v_1) dans la deuxième sur la même horizontale. On cherche à estimer la disparité $d(u_1, v_1) = u_2 - u_1$.

1. Extraire les points de Harris dans la première image
2. Prendre comme "graines" les appariements sûrs suivants : point de Harris corrélant "très bien" (seuil haut) avec un point de l'image 2 sur la même horizontale.
3. De proche en proche : si un point est correctement apparié avec une disparité d, alors un de ses voisins non encore apparié est considéré lui aussi comme correctement apparié s'il corrèle "bien" (seuil bas) pour une disparité $d-1$, d ou $d+1$ (garder la meilleure des trois).
4. Une fois tous les appariements estimés, la carte de disparité $d(u_1, v_1)$ a des valeurs entières. On peut la rendre à valeurs réelles en la lissant, où en estimant par interpolation la position exacte du maximum de corrélation entre $d-1$ et $d+1$ (interpoler les corrélations avec une parabole).

3.4 3D

Revenir à la vraie profondeur z. Afficher le visage en 3D.

4 Stéreovision et segmentation par graph-cuts

Il s'agira d'implémenter un algorithme efficace (et à la mode !) de détermination d'une coupe minimale dans un graphe et de l'utiliser pour deux applications : stéreovision et contours actifs. Travail à partir d'articles fournis.