TP7: Trackeur de Lucas Kanade

regis.perrier@imag.fr, elise.arnaud@imag.fr

Ce TP à pour objectif de poursuivre dans une séquence d'images un objet en utilisant la technique de Lucas Kanade. Cet algorithme a été proposé dans les années 80; il fait encore parti aujourd'hui des algorithmes les plus utilisés en vision par ordinateur.

Définitions

Par la suite, nous appellerons:

- i et j la position en ligne et colonne d'un pixel dans une image,
- $T(i, j)$ l'image modèle de l'objet à tracker de taille $n \times m$,
- $I^t(i, j)$ l'image à l'instant t de la séquence de taille $N \times M$,
- $I^t_o(i, j)$ l'image extraite de $I^t(i, j)$ de taille $n \times m$ et contenant l'objet à tracker.

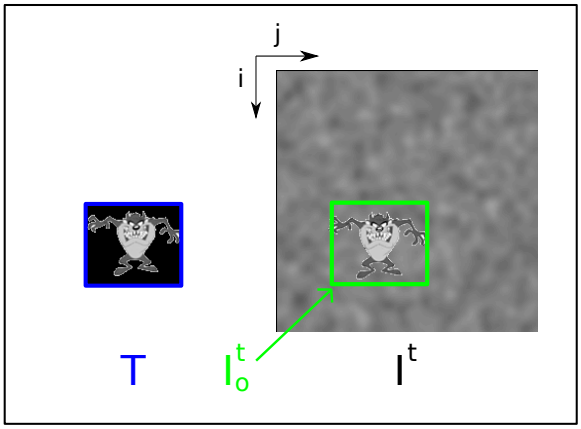

figure - définitions

Principe

Le Lucas Kanade cherche les paramètres Δi et Δj minimisant l'expression (non linéaire) suivante:

$$\sum_{i,j} \big(T(i,j) - I_o^t(i+\Delta i, j+\Delta j)\big)^2$$

Sous l'hypothèse que les déplacements sont suffisamment petits (i.e. Δi et Δj inférieurs à 4 pixels), l'expression précédente peut être linéarisée au premier ordre, et on obtient:

$$\sum_{i,j} \Big(T(i,j) - I_o^t(i,j) - \frac{\partial I_o^t(i,j)}{\partial i}\Delta i - \frac{\partial I_o^t(i,j)}{\partial j}\Delta j\Big)^2$$

Cette équation peut maintenant être résolue par un moindre carré classique (après quelques développements):

$$\begin{bmatrix} \Delta i \\ \Delta j \end{bmatrix} = \begin{bmatrix} \sum_{i,j} \frac{\partial I_o^t(i,j)^2}{\partial i} & \sum_{i,j} \frac{\partial I_o^t(i,j)}{\partial i}\frac{\partial I_o^t(i,j)}{\partial j} \\ \sum_{i,j} \frac{\partial I_o^t(i,j)}{\partial i}\frac{\partial I_o^t(i,j)}{\partial j} & \sum_{i,j} \frac{\partial I_o^t(i,j)^2}{\partial j} \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i,j} \frac{\partial I_o^t(i,j)}{\partial i}(T(i,j) - I_o^t(i,j)) \\ \sum_{i,j} \frac{\partial I_o^t(i,j)}{\partial j}(T(i,j) - I_o^t(i,j)) \end{bmatrix}$$

L'étape de linéarisation implique que cette équation doit être minimisée de façon itérative. En pratique, l'image $T(i,j)$ doit être interpolée successivement avec une estimation courante du déplacement Δ_i et Δ_j, et l'équation doit être à nouveau résolue jusqu'à ce que les déplacements estimés soient minimes. Nous en tiendrons compte lors de l'implémentation.

exercice 1

Construisez une fonction permettant d'extraire de votre image de séquence $I^t(i,j)$ une vignette de taille $n \times m$ en partant du point (i,j) (soit de i à $i+n-1$ et de j à $j+m-1$). Cette vignette constituera votre image $I_o^t(i,j)$.
Construisez une fonction estimant le gradient selon i et j sur une image.
Construisez une fonction calculant le résidu (erreur) entre deux images (soit $T(i,j) - I_o^t(i,j)$).
Construisez une fonction permettant d'inverser une matrice 2×2.

exercice 2

Construisez une fonction d'interpolation d'une image prenant en paramètres d'entrée: l'image, la translation selon i, la translation selon j. On obtient en sortie de fonction l'image interpolée aux nouveaux points. La valeur d'un nouveau pixel est une somme des valeurs de ses quatre pixels voisins pondérées par la distance aux centres de ses voisins.

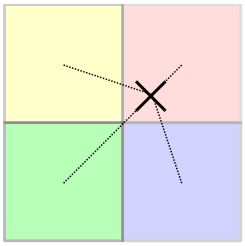

figure - interpolation bilinéaire, la valeur du nouveau pixel positionné sur la croix noire est une somme de ses quatre pixels voisins pondérée par la distance au centres de ses voisins.

exercice 3

Dans le jeu de données `synthetic`, chargez la première image `taz001.pgm` (image $I^1(i,j)$).
Découpez une vignette de taille 84×100 pixels à la position $(i,j) = (50, 130)$ (image $I_o^1(i,j)$).
Caculez le gradient selon i et j de cette image.
Chargez l'image `taz.pgm` (image $T(i,j)$) qui constitue votre modèle.
Exécutez sur plusieurs itérations (50 au maximum) les commandes suivantes:

Initialisation: $\Delta_i = 0$, $\Delta_j = 0$, $T_w(i,j) = T(i,j)$

- Calculez l'erreur entre l'image $T_w(i,j)$ et l'image $I^1(i,j)$.
- Appliquez la formule décrite précédemment pour estimer le déplacement en i (noté δ_i) et le déplacement en j (noté δ_j).
- Mettez à jour les déplacements: $\Delta_i = \delta_i + \Delta_i$ et $\Delta_j = \delta_j + \Delta_j$.
- Interpolez l'image $T(i,j)$ pour les déplacements Δ_i et Δ_j et enregistrez la dans l'image $T_w(i,j)$.

Jusqu'à ce que δ_i et δ_j soient inférieurs à un seuil ϵ (par exemple $\epsilon = 0.001$).
Conseil: vérifiez bien que l'image interpolée $T_w(i,j)$ se déplace dans le bon sens par rapport à votre estimation de Δ_i et Δ_j et votre image $I_o^1(i,j)$!

exercice 4

Appliquez la technique précédente sur toutes les images de la séquence; prenez soin d'initialiser votre vignette (image $I_o^t(i,j)$) à la position précédemment estimée. Pour chaque image trackée, enregistrez la avec le cadre détourant l'objet tracké après convergence de l'algorithme.

suggestions pour le rapport

Dans votre rapport vous pourrez:

- montrer des exemples d'interpolation d'image (n'importe quel image),
- appliquer une translation (Δ_i, Δ_j) et son inverse $(-\Delta_i, -\Delta_j)$ sur une même image $I_w(i,j)$, et la comparer avec l'image originale $I(i,j)$ (exemple de comparaison: $I_w(i,j) - I(i,j)$, ceci afin d'observer les limites de l'interpolation bilinéaire),
- afficher l'image d'erreur $T(i,j) - I_o^t(i,j)$ pour chaque itération du Lucas Kanade, et observer que l'erreur diminue au cours du recalage, (le recalage doit bien fonctionner pour des petits déplacements au début de la séquence d'images, et moins bien à la fin car les déplacements sont plus importants),
- mettre en évidence des cas non fonctionnels du trackeur,
- montrer des images où l'objet est tracké au cours de la séquence (placez un cadre autour de l'objet à partir de ce que l'algorithme a estimé),
- donner la série de translations estimées pour chaque image de la séquence.