Un algorithme de suivi d'objets pour l'analyse des mouvements

Le suivi d'objets

- La vidéosurveillance

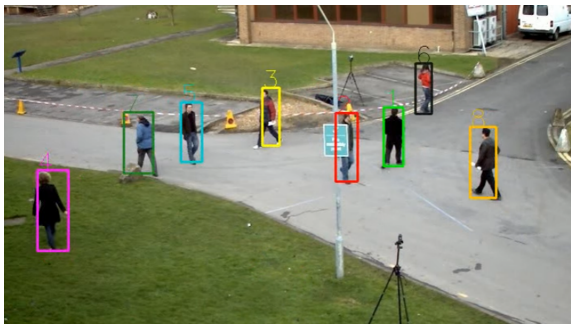


Figure 1 – motchallenge.net

- Les voitures autonomes

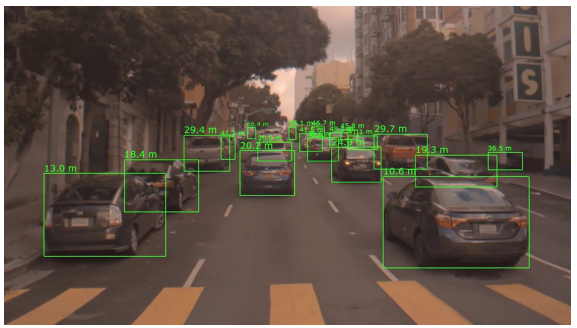


Figure 2 – nvidia.com/nvidia-drive-labs

- L'analyse des mouvements

Figure 3 – epfl.ch/labs/vita

Problématique

- Comment assurer un suivi efficace des joueurs dans un match de football ?

Position du problème

Contexte

- Caméra mobile.
- Objets multiples et déformables.
- Arrière plan relativement uniforme.

Figure 4 – Source : [MOT1].

Représentation des objets

Différentes méthodes de représentation :

(a) Un point, (b) Plusieurs points, (c) Rectangle englobant, (d) Ellipse, (e) Silhouette, (f) Contour, (g) Parties, (h) Squelette.

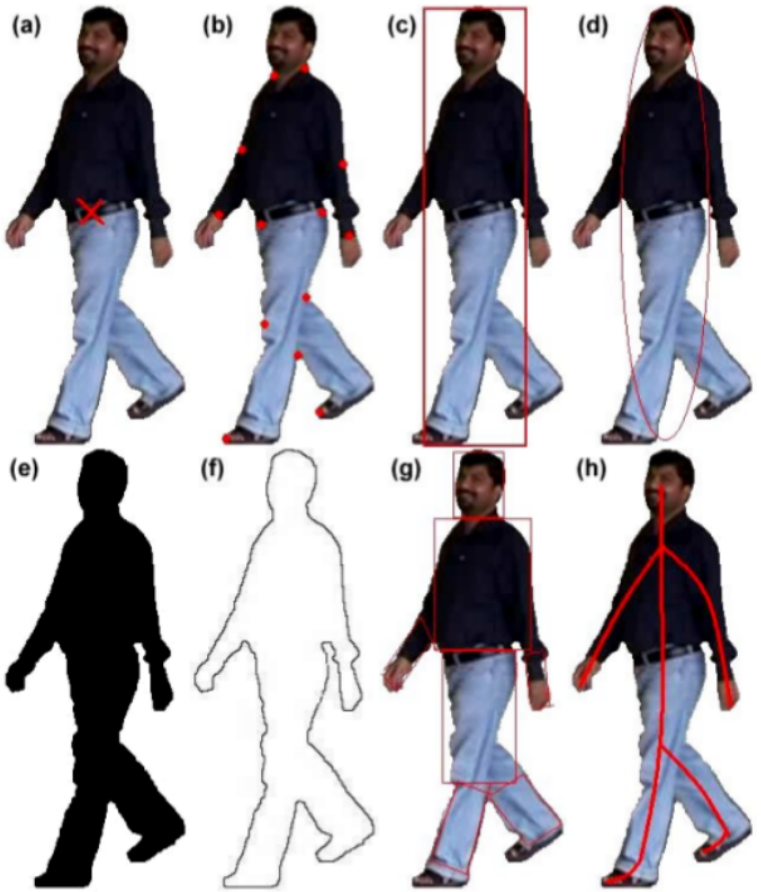


Figure 5 – Source : [SOA2].

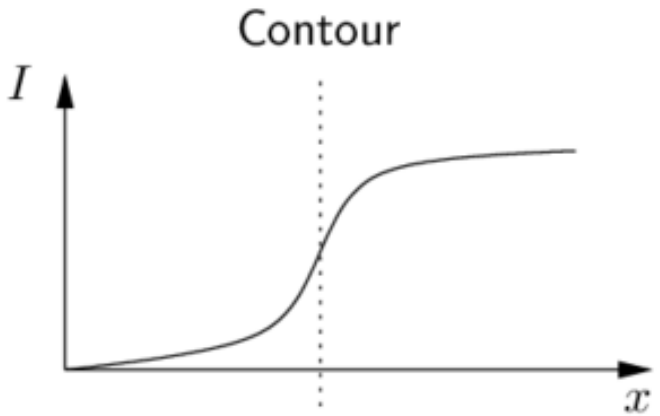
Gradient d'une image

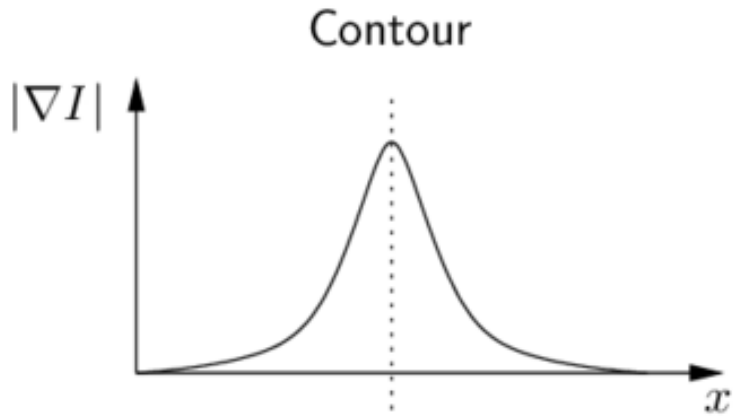
Définition

$$\nabla I(x,y) = \begin{pmatrix} \frac{\partial I}{\partial x}(x,y) \\ \frac{\partial I}{\partial y}(x,y) \end{pmatrix} \approx \begin{pmatrix} \frac{I(x+1,y) - I(x-1,y)}{2} \\ \frac{I(x,y+1) - I(x,y-1)}{2} \end{pmatrix}$$

$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

- I : intensité de l'image.





Contours actifs paramétriques

Principe de la technique

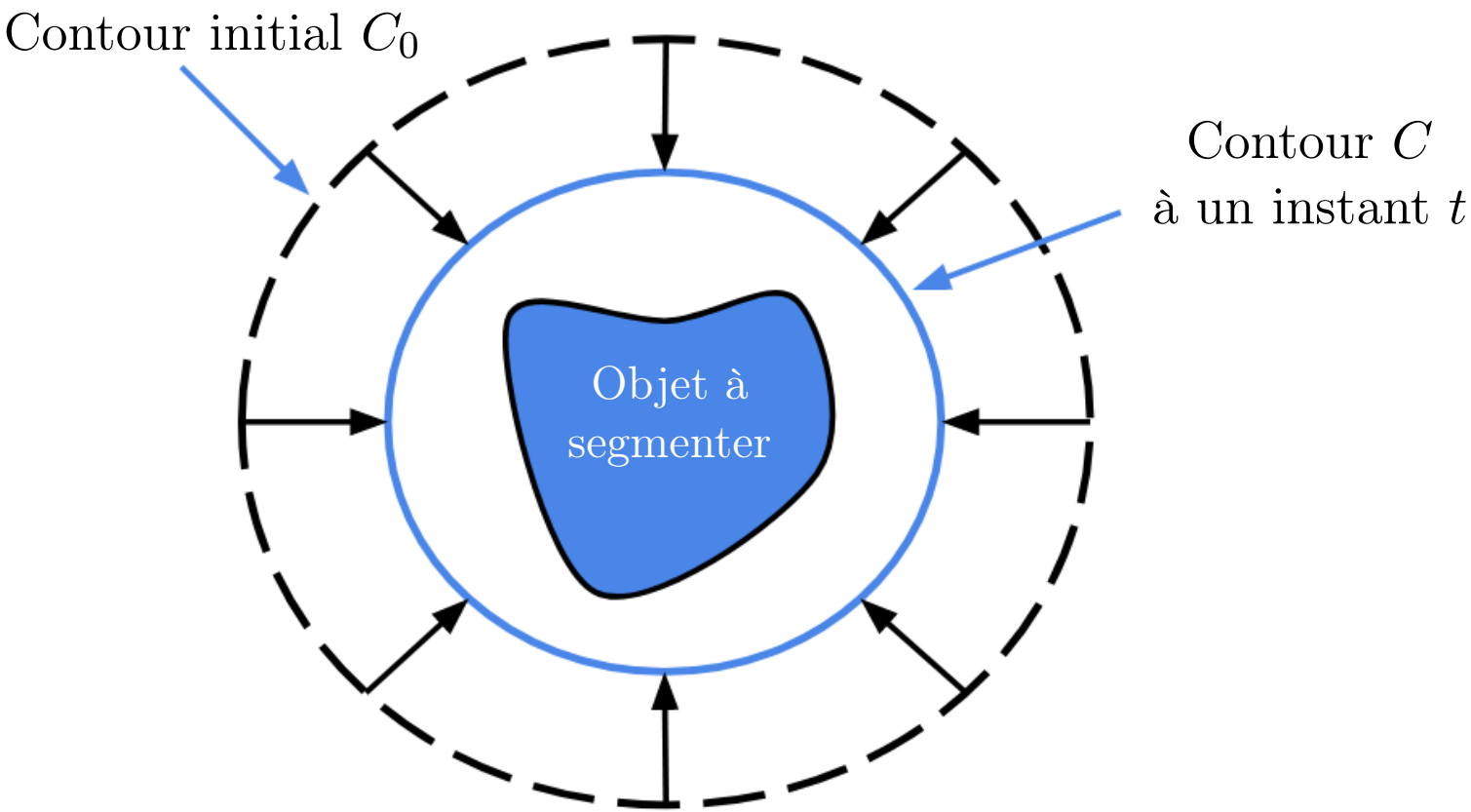


Figure 6 – Processus d'évolution du contour actif.

Contours actifs paramétriques

Formalisme continu

Représentation paramétrique

- $v : \left| \begin{array}{ccc} [0,1] & \longrightarrow & \mathbb{R}^2 \\ s & \longmapsto & (x(s), y(s)) \end{array} \right. \in \mathcal{C}^\infty([0,1], \mathbb{R}^2)$

- On note $C = \{v(s),\ s \in [0,1]\}$.

Contours actifs paramétriques

Énergie (coût) d'un contour

$$E(v) = E_{interne}(v) + E_{externe}(v)$$

- $E_{interne}$: propriétés intrinsèques au contour.
- $E_{externe}$: propriétés de l'image.

Modèle de M.Kass et al. [KASS3]

$$E(v) \stackrel{\text{def}}{=} \underbrace{\int_0^1 \left(\frac{\alpha}{2}\left|\dot{v}(s)\right|^2 + \frac{\beta}{2}\left|\ddot{v}(s)\right|^2\right) \mathrm{d}s}_{E_{interne}(v)} \underbrace{-\lambda \int_0^1 \left|\nabla I(v(s))\right|^2 \mathrm{d}s}_{E_{externe}(v)}$$

- $\alpha, \beta \in \mathbb{R}_+$, poids qui permettent de contrôler la régularité du contour.
- $\lambda \in \mathbb{R}_+$ poids de l'énergie externe.

Contours actifs paramétriques

Énergie (coût) d'un contour

$$E(v) = E_{interne}(v) + E_{externe}(v)$$

- $E_{interne}$: propriétés intrinsèques au contour.
- $E_{externe}$: propriétés de l'image.

Modèle de M.Kass et al. [KASS3]

$$E(v) \stackrel{\text{def}}{=} \underbrace{\int_0^1 \left(\frac{\alpha}{2} |\dot{v}(s)|^2 + \frac{\beta}{2} |\ddot{v}(s)|^2 \right) \mathrm{d}s}_{E_{interne}(v)} \underbrace{-\lambda \int_0^1 |\nabla I(v(s))|^2 \, \mathrm{d}s}_{E_{externe}(v)}$$

- $\alpha, \beta \in \mathbb{R}_+$, poids qui permettent de contrôler la régularité du contour.
- $\lambda \in \mathbb{R}_+$ poids de l'énergie externe.

Contours actifs paramétriques

Problème équivalent

$$\min_{v}\left(\int_0^1 (\frac{\alpha}{2}|\dot{v}(s)|^2+\frac{\beta}{2}|\ddot{v}(s)|^2-\lambda|\nabla I(v(s))|^2)\,\mathrm{d}s\right)$$

- Minimiser $E_{interne}$ permet de "lisser" le contour.
- Minimiser $E_{externe}$ permet d'atteindre les bords de l'objet.

Contours actifs paramétriques

Approche variationnelle

Théorème (Euler-Lagrange) :

Soit J la fonctionnelle définie sur l'espace des fonctions $\mathcal{C}^2(\mathbb{R},\mathcal{U})$ où $\mathcal{U}$ est un ouvert de $\mathbb{R}^n$ par :

$$J(y) = \int_0^1 \mathcal{L}(t, y(t), \dot{y}(t))\,\mathrm{d}t$$

où $\mathcal{L}$ est une fonction $\mathcal{C}^2$. Une fonction $y \in \mathcal{C}^2(\mathbb{R},\mathcal{U})$ est un point critique de J si et seulement si y est solution de l'équation d'Euler-Lagrange :

$$\partial_2\mathcal{L}(t, y(t), \dot{y}(t)) - \frac{d}{dt}\partial_3\mathcal{L}(t, y(t), \dot{y}(t)) = 0$$

Généralisation :

Pour

$$J(y) = \int_0^1 \mathcal{L}(t, y(t), \dot{y}(t), \ddot{y}(t), ..., y^{(n)}(t))\,\mathrm{d}t$$

si $\mathcal{L} \in \mathcal{C}^\infty$, l'équation d'Euler-Lagrange s'écrit :

$$\partial_2\mathcal{L} - \frac{d}{dt}\partial_3\mathcal{L} + \frac{d^2}{dt^2}\partial_4\mathcal{L} + ... + (-1)^n\frac{d^n}{dt^n}\partial_{n+2}\mathcal{L} = 0$$

Contours actifs paramétriques

Approche variationnelle

Théorème (Euler-Lagrange) :

Soit J la fonctionnelle définie sur l'espace des fonctions $\mathcal{C}^2(\mathbb{R},\mathcal{U})$ où $\mathcal{U}$ est un ouvert de $\mathbb{R}^n$ par :

$$J(y) = \int_0^1 \mathcal{L}(t, y(t), \dot{y}(t))\,\mathrm{d}t$$

où $\mathcal{L}$ est une fonction $\mathcal{C}^2$. Une fonction $y \in \mathcal{C}^2(\mathbb{R},\mathcal{U})$ est un point critique de J si et seulement si y est solution de l'équation d'Euler-Lagrange :

$$\partial_2\mathcal{L}(t, y(t), \dot{y}(t)) - \frac{d}{dt}\partial_3\mathcal{L}(t, y(t), \dot{y}(t)) = 0$$

Généralisation :

Pour

$$J(y) = \int_0^1 \mathcal{L}(t, y(t), \dot{y}(t), \ddot{y}(t), ..., y^{(n)}(t))\,\mathrm{d}t$$

si $\mathcal{L} \in \mathcal{C}^\infty$, l'équation d'Euler-Lagrange s'écrit :

$$\partial_2\mathcal{L} - \frac{d}{dt}\partial_3\mathcal{L} + \frac{d^2}{dt^2}\partial_4\mathcal{L} + ... + (-1)^n\frac{d^n}{dt^n}\partial_{n+2}\mathcal{L} = 0$$

Contours actifs paramétriques

Équation d'évolution

- D'après le théorème d'Euler-Lagrange, le contour optimal est solution de :

$$\boxed{\alpha v^{(2)}(s) - \beta v^{(4)}(s) + \lambda \nabla P(v(s)) = 0}$$

où $P(v(s)) = |\nabla I(v(s))|^2$ (potentiel de l'image).

- Pour un contour à n points, on note $V_t = (x_{t,1}, ..., x_{t,n}, y_{t,1}, ...y_{t,n})^T$ et $F(V_t) = -\lambda(\frac{\partial P}{\partial x}(x_{t,1}, y_{t,1}), ..., \frac{\partial P}{\partial y}(x_{t,n}, y_{t,n}))^T$.

  En approximant les dérivées par la méthode des différences finies, on obtient le schéma itératif :

$$\boxed{V_{t+1} = (I_{2n} - \gamma A)^{-1}(V_t - \gamma F(V_t))}$$

où $A \in \mathcal{M}_{2n}(\mathbb{R})$ (en annexe) et $\gamma \in \mathbb{R}_+$ est le pas d'évolution du contour.

Contours actifs paramétriques

Équation d'évolution

- D'après le théorème d'Euler-Lagrange, le contour optimal est solution de :

$$\boxed{\alpha v^{(2)}(s) - \beta v^{(4)}(s) + \lambda \nabla P(v(s)) = 0}$$

où $P(v(s)) = |\nabla I(v(s))|^2$ (potentiel de l'image).

- Pour un contour à n points, on note $V_t = (x_{t,1}, ..., x_{t,n}, y_{t,1}, ...y_{t,n})^T$ et $F(V_t) = -\lambda(\frac{\partial P}{\partial x}(x_{t,1}, y_{t,1}), ..., \frac{\partial P}{\partial y}(x_{t,n}, y_{t,n}))^T$.

En approximant les dérivées par la méthode des différences finies, on obtient le schéma itératif :

$$\boxed{V_{t+1} = (I_{2n} - \gamma A)^{-1}(V_t - \gamma F(V_t))}$$

où $A \in \mathcal{M}_{2n}(\mathbb{R})$ (en annexe) et $\gamma \in \mathbb{R}_+$ est le pas d'évolution du contour.

Contours actifs paramétriques

Premiers résultats

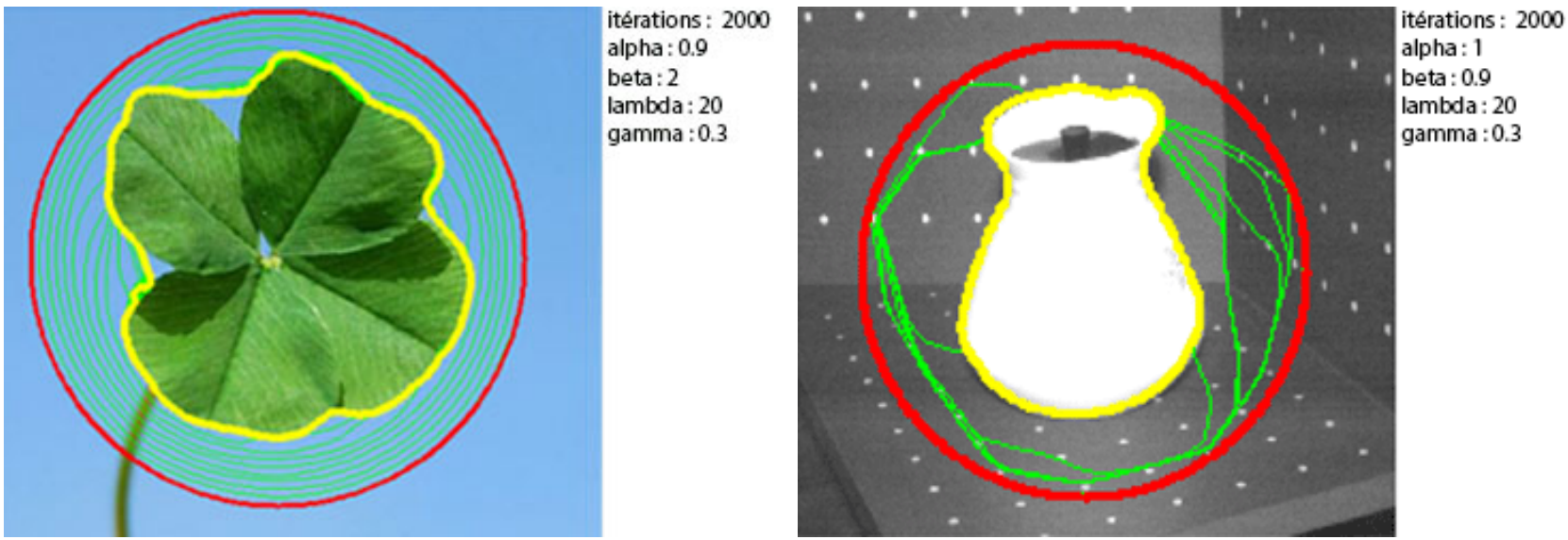


Figure 7 – Deux exemples d'application : contour initial en rouge et contour final en jaune.

Contours actifs paramétriques

Force normale

Force normale

$$\forall s \in [0,1],\ \overrightarrow{F_{normale}}(v(s)) = k\,\vec{n}(s)$$

- $k \in \mathbb{R}$, paramètre réglable.
- $\vec{n}(s)$ vecteur unitaire normal au contour.

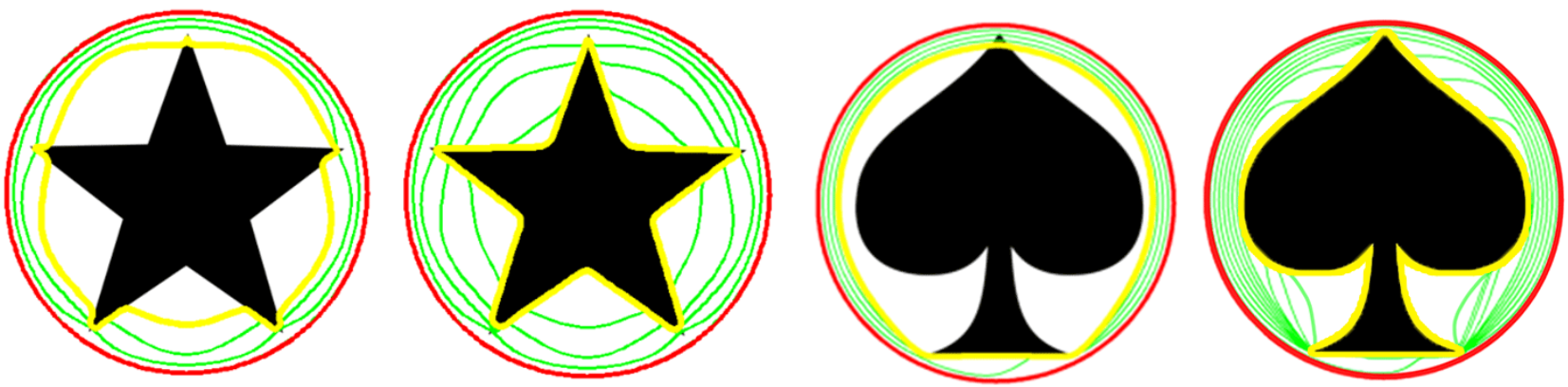


Figure 8 – Deux exemples : sans la force normale (à gauche) et avec la force normale (à droite) pour le même nombre d'itérations.

Contours actifs paramétriques

Influence des paramètres

Figure 9 – Influence de α : élasticité du contour.

Contours actifs paramétriques

Influence des paramètres

Figure 9 – Influence de α : élasticité du contour.

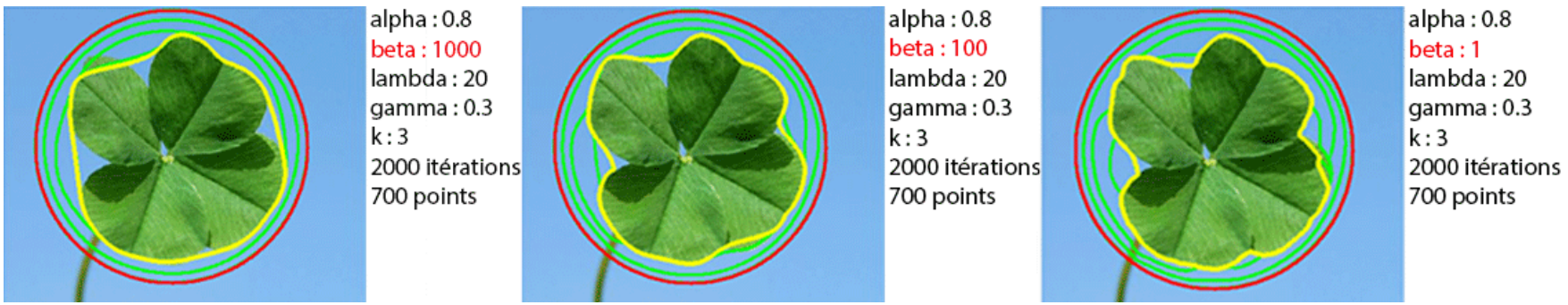


Figure 10 – Influence de β : rigidité du contour.

- Les concavités ne sont souvent pas atteintes.
- Le réglage des paramètres doit être adapté à chaque image.

Contours actifs paramétriques

Limites du modèle

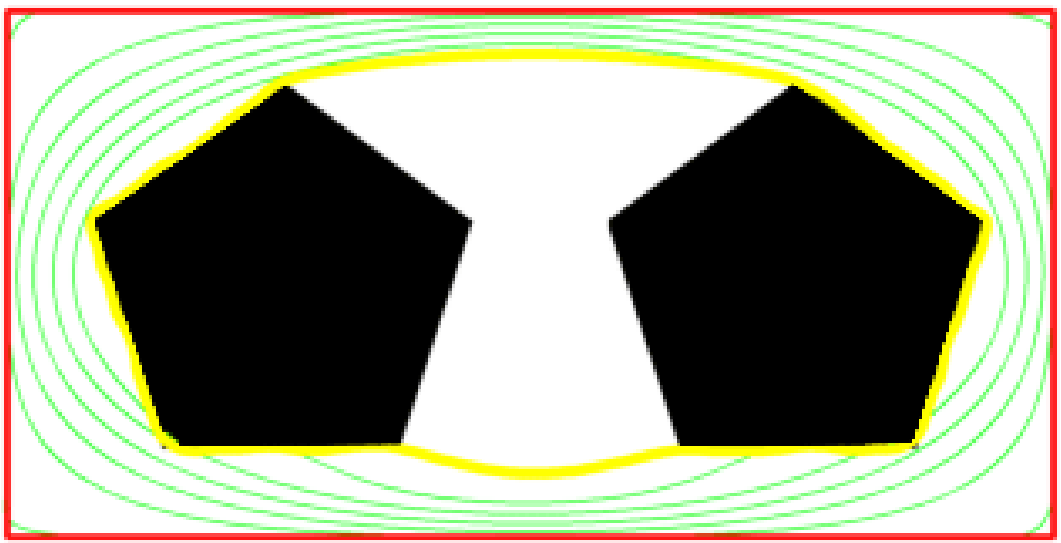


Figure 11 – Problème de changement de topologie.

- Tout changement de topologie est impossible : on ne peut détecter qu'un seul objet à la fois.

Contours actifs géométriques

Méthode des courbes de niveau

- On définit le contour actif comme une courbe implicite d'une fonction Φ qui évolue au cours du temps :

$$C(t) = \{(x, y) \in \mathbb{R}^2 \mid \Phi(x, y, t) = 0\}$$

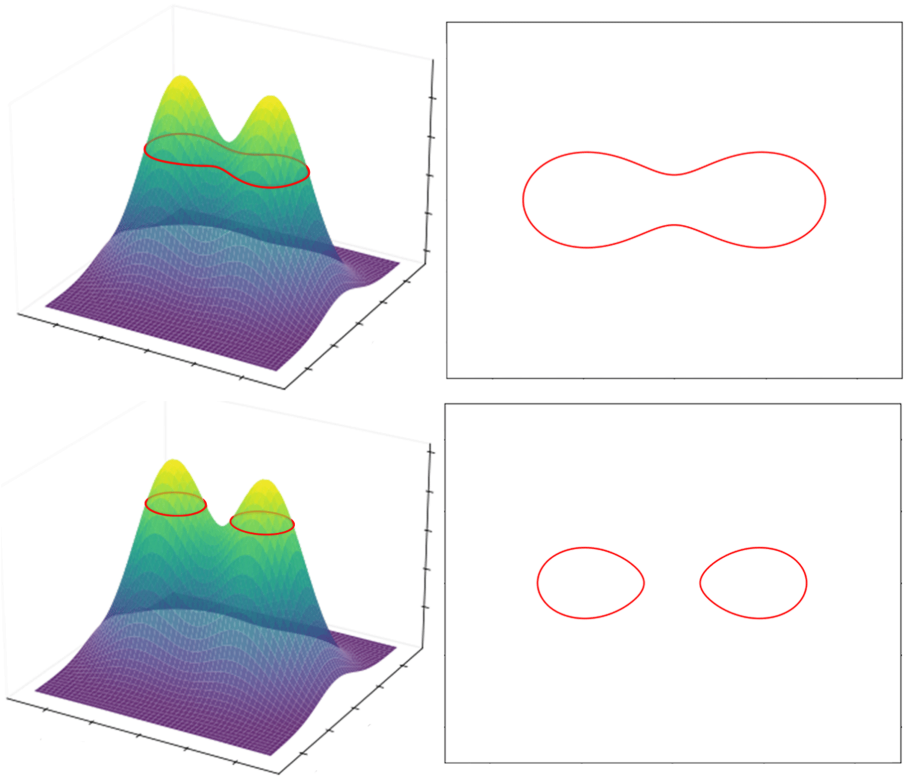


Figure 12 – Graphe de Φ (à gauche) et contour (à droite).

Contours actifs géométriques

Un nouveau modèle

Équation d'évolution, modèle de Caselles et al. [CAS4]

La fonction implicite Φ évolue suivant l'équation :

$$\frac{\partial \Phi}{\partial t} = g(I)\left(k + \operatorname{div}\left(\frac{\nabla \Phi}{|\nabla \Phi|}\right)\right)|\nabla \Phi|$$

où :

- g est une fonction d'arrêt aux contours définie par $g(I) = \dfrac{1}{1+|\nabla I|^2}$.
- $k \in \mathbb{R}$ paramètre réglable.

Contours actifs géométriques

Interprétation du modèle

- $\operatorname{div}\left(\frac{\nabla\Phi}{|\nabla\Phi|}\right)$: expression de la courbure algébrique du contour.

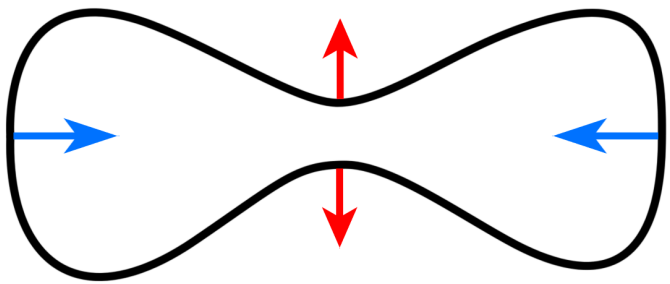


Figure 13 – Déformation selon la courbure (positive ou négative).

- k est réglé de sorte que la quantité $k + \operatorname{div}\left(\frac{\nabla\Phi}{|\nabla\Phi|}\right)$ reste positive : même effet que la force normale.

Contours actifs géométriques

Résultats

- Φ évolue suivant le schéma obtenu par la méthode des différences finies :

$$\Phi_t = \Phi_{t-1} + \gamma f(\Phi_{t-1})$$

où γ est le pas d'évolution.

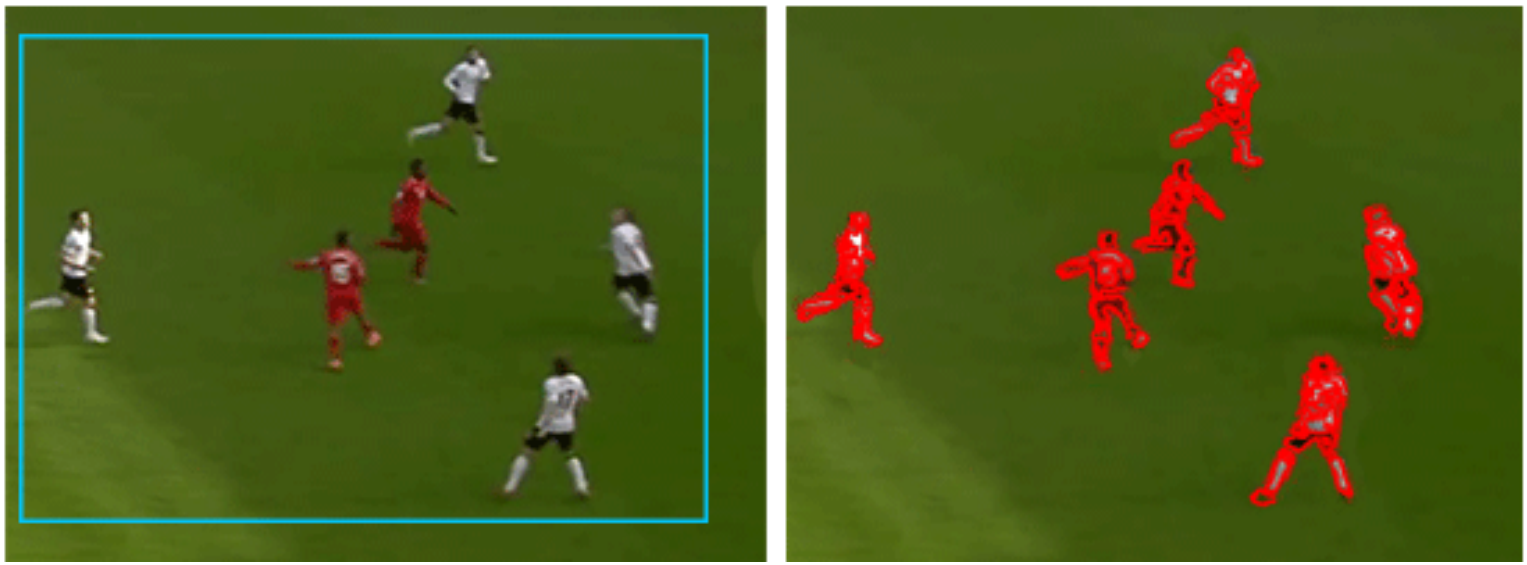


Figure 14 – Contour initial en bleu et contour final en rouge (7000 itérations, k=100, γ=3).

Résultats et limites

Suivi final

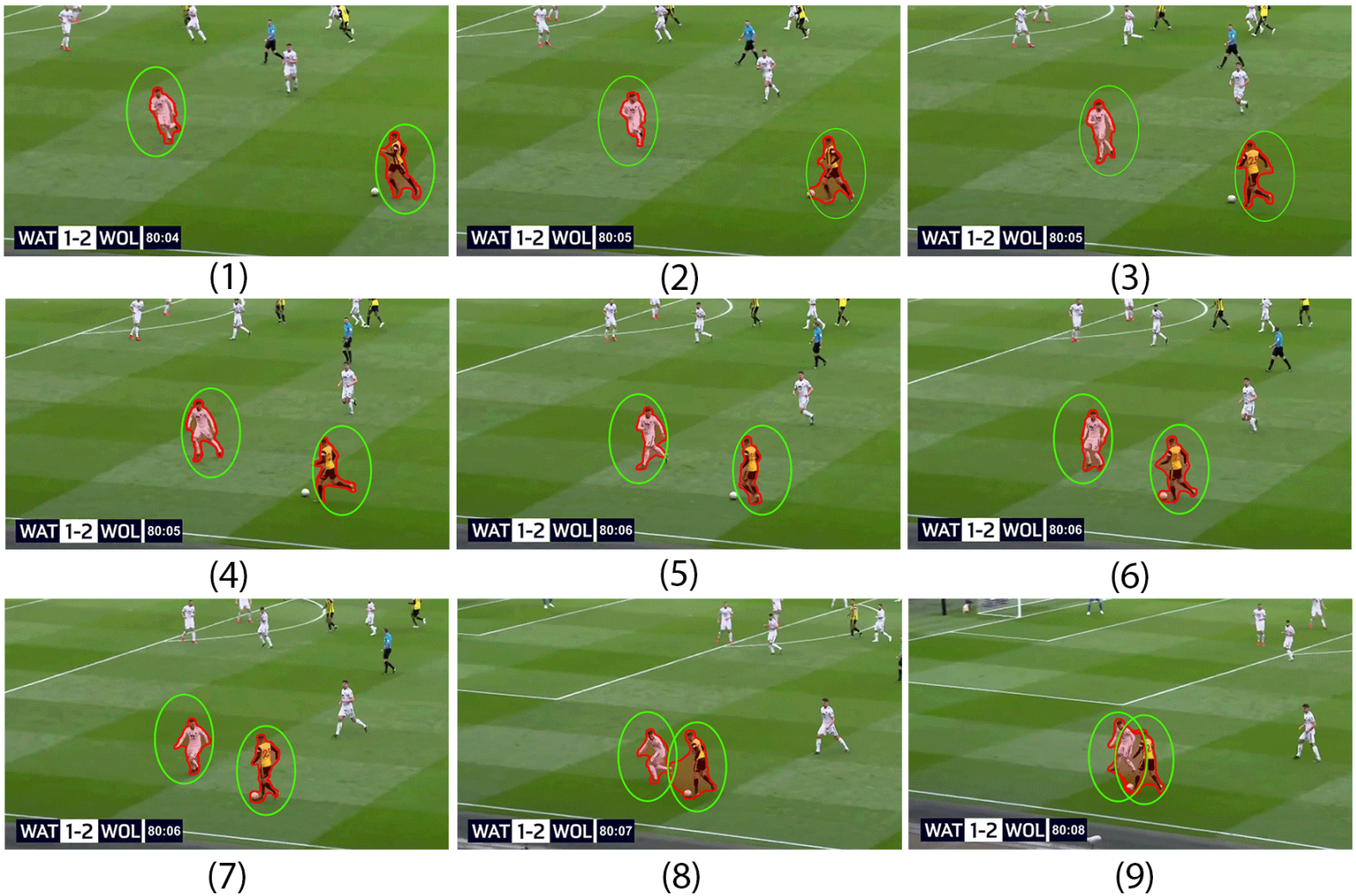


Figure 15 – Suivi de deux joueurs (contours actifs paramétriques).

Résultats et limites

Limites du modèle des contours actifs

Limites :

- Le modèle est très sensible à la qualité de l'image.
- La convergence dépend beaucoup du contour initial.
- Les obstructions ne sont pas gérées.

Solutions envisageables :

- Algorithmes de prétraitement (filtre gaussien, égalisation d'histogramme...).
- Algorithme correctif (filtre de Kalman).

Résultats et limites

Limites du modèle des contours actifs

Limites :

- Le modèle est très sensible à la qualité de l'image.
- La convergence dépend beaucoup du contour initial.
- Les obstructions ne sont pas gérées.

Solutions envisageables :

- Algorithmes de prétraitement (filtre gaussien, égalisation d'histogramme...).
- Algorithme correctif (filtre de Kalman).

Merci de votre attention.

Formalisme Lagrangien :

On note $\mathcal{U}$ un ouvert de $\mathbb{R}^n$ et $\mathcal{G}$ l'espace affine des fonctions $y \in \mathcal{C}^1([0,1], \mathcal{U})$ telles que $y(0) = a$ et $y(1) = b$ soient deux vecteurs fixés de $\mathcal{U}$ (condition aux bornes).
Soit $\mathcal{L} \in \mathcal{C}^2([0,1] \times \mathcal{U}^2, \mathbb{R})$ (appelé lagrangien).

On cherche à minimiser la fonctionnelle J définie par :

$$J : \left| \begin{array}{ccc} \mathcal{G} & \longrightarrow & \mathbb{R} \\ y & \longmapsto & \displaystyle\int_0^1 \mathcal{L}(t, y(t), \dot{y}(t))\,\mathrm{d}t \end{array} \right.$$

On notera dans tout ce qui suit $\dfrac{\partial \mathcal{L}}{\partial y}$ et $\dfrac{\partial \mathcal{L}}{\partial \dot{y}}$ les gradients de $\mathcal{L}$ vu comme fonction de son premier (resp. second) argument dans $\mathbb{R}^n$ (la notation peut porter à confusion mais est usuelle).

Définition 1 : Soit $y \in \mathcal{G}$. Une variation de y est une famille $(y_s)_{s\in]-\varepsilon;\varepsilon[}$ de $\mathcal{G}$ telle que :

$$y_0 = y \text{ et } \frac{dy}{ds}(s=0) = h$$

où $h \in \mathcal{C}^1([0,1],\mathbb{R}^n)$ est telle que $h(0) = h(1) = 0$.

Remarques :

- *On peut interpréter h comme le vecteur $\gamma'(0)$ tangent à la courbe $\gamma : s \longmapsto y_s$ au point $y = \gamma(0) \in \mathcal{G}$.*
  On note $T_y\mathcal{G}$ l'espace de toutes ces courbes h obtenues de cette manière (l'espace tangent à $\mathcal{G}$ en y).
- *Soit $h \in T_y\mathcal{G}$. Un exemple simple de variation $(y_s)_{s\in]-\varepsilon,\varepsilon[}$ associée à h est donnée par :*

$$y_s : t \longmapsto y(t) + sh(t)$$

Soit $(y_s)_{s\in]-\varepsilon,\varepsilon[}$ la variation précédente associée à $h \in T_y\mathcal{G}$. Pour tout s assez proche de 0, l'image de y_s est incluse dans $\mathcal{U}$.

Proposition : La fonction $s \longmapsto J(y_s)$ est dérivable au voisinage de 0.

Démonstration : On peut appliquer le théorème de dérivation sous l'intégrale à la fonction :

$$(s,t) \longmapsto \mathcal{L}(t, y_s(t), \dot{y}_s(t))$$

D'après la règle de la chaîne, on obtient pour $t \in [0,1]$:

$$\frac{d\mathcal{L}}{ds}(t, y_s(t), \dot{y}_s(t)) = \left\langle \frac{\partial \mathcal{L}}{\partial y_s}, \frac{dy_s}{ds}(t) \right\rangle + \left\langle \frac{\partial \mathcal{L}}{\partial \dot{y}_s}, \frac{d\dot{y}_s}{ds}(t) \right\rangle$$

En évaluant en $s = 0$, on a :

$$\frac{dJ}{ds}(y_s)|_{s=0} = \int_0^1 \left[\left\langle \frac{\partial \mathcal{L}}{\partial y}, h(t) \right\rangle + \left\langle \frac{\partial \mathcal{L}}{\partial \dot{y}}, \dot{h}(t) \right\rangle \right] \mathrm{d}t$$

On intègre par parties le deuxième terme de l'intégrale :

$$\int_0^1 \left[\left\langle \frac{\partial \mathcal{L}}{\partial \dot{y}}, \dot{h}(t)\right\rangle\right] \mathrm{d}t = \left[\left\langle \frac{\partial \mathcal{L}}{\partial \dot{y}}, h(t)\right\rangle\right]_0^1 - \int_0^1 \left\langle \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{y}}\right), h(t)\right\rangle \mathrm{d}t$$

Comme $h(0) = h(1) = 0$, le crochet est nul et :

$$\frac{dJ}{ds}(y_s)|_{s=0} = \int_0^1 \left\langle \left[\frac{\partial \mathcal{L}}{\partial y} - \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{y}}\right)\right], h(t)\right\rangle \mathrm{d}t$$

Remarque : Au sens de Gâteaux (moins fort que la différentiabilité au sens de Fréchet), J est différentiable en y et l'expression précédente donne $\mathrm{d}J_y(h)$, la différentielle de J en y évaluée en h.

Définition : On appelle extrémale (ou point critique) de J tout élément $y \in \mathcal{G}$ tel que $\mathrm{d}J_y$ est nulle au sens précédent.

Théorème (équation de Euler-Lagrange) :
La courbe $y : t \longmapsto y(t)$ est une extrémale de J si et seulement si :

$$\boxed{\forall t \in [0,1], \quad \frac{\partial \mathcal{L}}{\partial y}(t, y(t), \dot{y}(t)) - \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{y}}\right)(t, y(t), \dot{y}(t)) = 0}$$

Démonstration : La condition est suffisante d'après le calcul précédent.
Réciproquement, pour tout $h \in T_y\mathcal{G}$ tel que $h(1) = h(0) = 0$:

$$\int_0^1 \Big\langle v(t), h(t) \Big\rangle \, \mathrm{d}t = 0, \text{ où } v(t) = \frac{\partial \mathcal{L}}{\partial y} - \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{y}}\right)$$

Par l'absurde, s'il existe t_0 tel que le vecteur $v(t_0)$ est non nul, on peut supposer sans perdre de généralité que $t_0 \in]0,1[$ par continuité de v. Posons alors $h : t \longmapsto \chi(t)v(t)$ où χ est une fonction positive et lisse telle que $\chi(t_0) > 0$, et telle que $\chi(0) = \chi(1) = 0$.
La fonction $t \longmapsto \langle v(t), h(t)\rangle = \chi(t)\|v(t)\|^2$ est alors continue, positive et non nulle, ce qui donne :

$$\int_0^1 \langle v(t), h(t)\rangle \, \mathrm{d}t = \int_0^1 \chi(t)\|v(t)\|^2 \, \mathrm{d}t > 0$$

en contradiction avec l'hypothèse selon laquelle y est une extrémale de J.

- D'après le théorème d'Euler-Lagrange, si le contour v minimise $\int_0^1 G\left(s, v, \dot{v}(s), \ddot{v}(s)\right) \mathrm{d}s$ alors :

$$\frac{\partial G}{\partial v} - \frac{\partial}{\partial s}\frac{\partial G}{\partial \dot{v}} + \frac{\partial^2}{\partial s^2}\frac{\partial G}{\partial \ddot{v}} = 0$$

Puisque $G\left(s, v(s), \dot{v}(s), \ddot{v}(s)\right) = \alpha|\dot{v}(s)|^2 + \beta|\ddot{v}(s)|^2 - \lambda P(v(s))$, on a :

$$\frac{\partial G}{\partial v} = -\lambda \nabla P(v(s)), \quad \frac{\partial}{\partial s}\frac{\partial G}{\partial \dot{v}} = \alpha v^{(2)}(s), \quad \frac{\partial^2}{\partial s^2}\frac{\partial G}{\partial \ddot{v}} = \beta v^{(4)}(s)$$

On obtient finalement l'équation :

$$\boxed{\alpha v^{(2)}(s) - \beta v^{(4)}(s) + \lambda \nabla P(v(s)) = 0}$$

Pour résoudre numériquement le système, on le transforme en système d'équations d'évolution temporelle. En notant $v(s,t) = (x(s,t), y(s,t))$, on a :

$$\begin{cases} \alpha \dfrac{\partial^2 x}{\partial s^2}(s,t) - \beta \dfrac{\partial^4 x}{\partial s^4}(s,t) + \lambda \dfrac{\partial P}{\partial x}(x(s,t), y(s,t)) = -\dfrac{\partial x}{\partial t}(s,t) & (1) \\ \alpha \dfrac{\partial^2 y}{\partial s^2}(s,t) - \beta \dfrac{\partial^4 y}{\partial s^4}(s,t) + \lambda \dfrac{\partial P}{\partial y}(x(s,t), y(s,t)) = -\dfrac{\partial y}{\partial t}(s,t) & (2) \end{cases}$$

Méthode des différences finies :

Euler explicite : $\dfrac{\partial f}{\partial x} \simeq \dfrac{f(x_{t+1}) - f(x_t)}{h}$

Euler implicite : $\dfrac{\partial f}{\partial x} \simeq \dfrac{f(x_t) - f(x_{t-1})}{h}$

Ordre 2 : $\dfrac{\partial^2 f}{\partial x^2} \simeq \dfrac{f(x_{t+1}) - 2f(x_t) + f(x_{t-1})}{h^2}$

Ordre 4 : $\dfrac{\partial^4 f}{\partial x^4} \simeq \dfrac{f(x_{t+2}) - 4f(x_{t+1}) + 6f(x_t) - 4f(x_{t-1}) + f(x_{t-2})}{h^4}$

Pour un contour à n points, on définit $M_2, M_4 \in \mathcal{M}_n(\mathbb{R})$, les matrices de passage du point de l'instant t à celui de l'instant $t+1$ (correspondant respectivement à la dérivée seconde et à la dérivée quatrième) avec la condition $v_0 = v_n$ (contour fermé).

$$M_2 = \begin{pmatrix} -2 & 1 & 0 & \dots & 0 & 1 \\ 1 & -2 & 1 & \dots & 0 & 0 \\ 0 & 1 & -2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -2 & 1 \\ 1 & 0 & 0 & \dots & 1 & -2 \end{pmatrix} \quad M_4 = \begin{pmatrix} 6 & -4 & 1 & \dots & 1 & -4 \\ -4 & 6 & -4 & \dots & 0 & 1 \\ 1 & -4 & 6 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & 0 & \dots & 6 & -4 \\ -4 & 1 & 0 & \dots & -4 & 6 \end{pmatrix}$$

Notons alors $M = \alpha M_2 - \beta M_4$. On note également le vecteur des positions des points du contour :

$$V_t = (x_{t,1}, ..., x_{t,n}, y_{t,1}, ...y_{t,n})^T = \begin{pmatrix} X_t \\ Y_t \end{pmatrix} \in \mathbb{R}^{2n}$$

et :

$$F(V_t) = \lambda(\frac{\partial P}{\partial x}(x_{t,1}, y_{t,1}), ..., \frac{\partial P}{\partial y}(x_{t,n}, y_{t,n}))^T = \begin{pmatrix} F_x(X_t) \\ F_y(Y_t) \end{pmatrix} \in \mathbb{R}^{2n}$$

En notant γ le pas temporel, l'équation devient :

$$\begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix} V_{t+1} + F(V_t) = -\frac{V_{t+1} - V_t}{\gamma}$$

D'où :

$$(I_{2n} + \gamma A)V_{t+1} + \gamma F(V_t) = V_t$$

où $A = \begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix} \in \mathcal{M}_{2n}(\mathbb{R})$. On a donc le schéma itératif :

$$\boxed{V_{t+1} = (I_{2n} + \gamma A)^{-1}(V_t - \gamma F(V_t))}$$

En ajoutant la force "normale", on a donc :

$$\boxed{V_{t+1} = (I_{2n} + \gamma A)^{-1}(V_t - \gamma(F(V_t) + kN_t))}$$

où $N_t \in \mathbb{R}^{2n}$ contient les coordonnées des vecteurs normaux aux contours aux points $(x_{t,1}, y_{t,1}), ..., (x_{t,n}, y_{t,n})$ au temps t.

Remarque :
La matrice $I - \gamma A$ pouvant être très grande (suivant le nombre de points du contour actif), le choix de l'algorithme d'inversion est important. L'inversion LU étant particulièrement efficace sur les matrices pentadiagonales ou tridiagonales, cette méthode est privilégiée.

Définition 1 :
Soit $\gamma : [a,b] \longrightarrow \mathbb{R}^2$ une courbe paramétrée de classe $\mathcal{C}^2$, soit $t \in [a,b]$ tel que $\gamma'(t) \neq 0$. Le repère de Serret-Frenet de γ au point $\gamma(t)$ est le repère orthonormé :

$$\boxed{\left(\gamma(t), \overrightarrow{T}(t), \overrightarrow{N}(t)\right)}$$

où $\overrightarrow{T}(t) = \dfrac{\gamma'(t)}{\|\gamma'(t)\|}$ et $\left(\overrightarrow{T}(t), \overrightarrow{N}(t)\right)$ est une base orthonormée direct du plan affine.

Définition 2 : (mêmes notations)
Soit $t \in [a,b]$ tel que $(\gamma'(t), \gamma''(t))$ est libre. On appelle cercle osculateur en $\gamma(t)$ le cercle passant par $\gamma(t)$, dont la tangente en $\gamma(t)$ est égale à $\gamma'(t)$ et qui approche le plus possible la courbe de γ au voisinage de ce point.
Son rayon $R(t) \neq 0$ est appelé rayon de courbure en t.
La courbure en t est définie par :

$$\boxed{\kappa(t) = \frac{1}{R(t)}}$$

Proposition 1 :
Soit $\gamma : [a,b] \longrightarrow \mathbb{R}^2$ une courbe paramétrée régulière de classe $\mathcal{C}^2$ et soit $t_0 \in [a,b]$ tel que $(\gamma'(t_0), \gamma''(t_0))$ est libre. Pour $t \in [a,b]$, notons $(x(t), y(t))$ les coordonnées de $\gamma(t)$ dans le repère de Serret-Frenet.

Dans le repère de Serret-Frenet, on a : $\begin{cases} \gamma'(t_0) = \|\gamma'(t_0)\| \overrightarrow{T}(t_0) \\ \gamma''(t_0) = \alpha \overrightarrow{T}(t_0) + \beta \overrightarrow{N}(t_0) \end{cases}$

Alors la courbure en t_0 s'écrit $\boxed{\kappa(t_0) = \dfrac{\beta}{\|\gamma'(t_0)\|^2}}$.

Démonstration :
Quitte à translater la courbe, on peut supposer que $\gamma(t_0) = (0,0)$.
Un développement limité donne :

$$\begin{aligned} x(t) &\underset{t\to t_0}{=} x(t_0) + (t-t_0)x'(t_0) + o\left((t-t_0)\right) \\ &\underset{t\to t_0}{=} (t-t_0)\|\gamma'(t_0)\| + o\left((t-t_0)\right) \\ y(t) &\underset{t\to t_0}{=} y(t_0) + (t-t_0)y'(t_0) + \frac{(t-t_0)^2}{2}y''(t_0) + o\left((t-t_0)^2\right) \\ &\underset{t\to t_0}{=} \beta\frac{(t-t_0)^2}{2} + o\left((t-t_0)^2\right) \end{aligned}$$

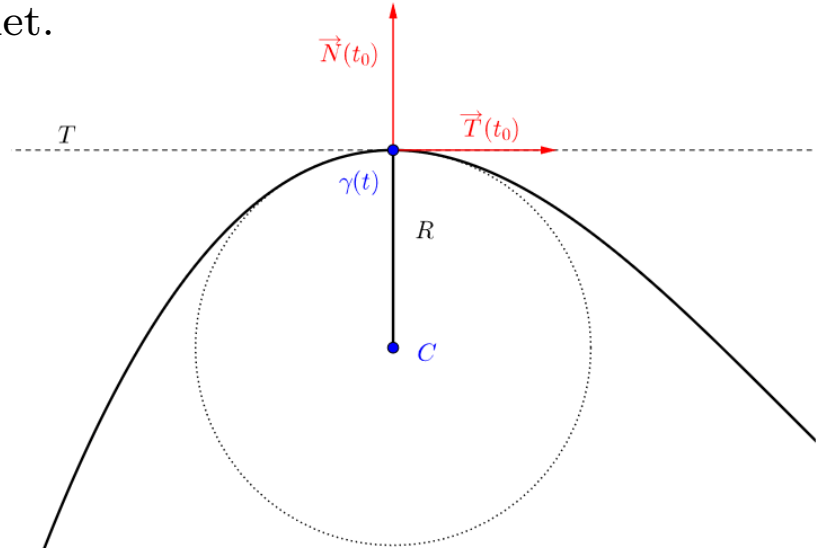
Le centre du cercle osculateur en t_0 a pour coordonnées $(0, R)$ dans le repère de Serret-Frenet puisque la tangente au cercle en t_0 est égale à la tangente à la courbe en ce même point.
Ainsi, dans ce repère, le cercle osculateur a pour équation :

$$x^2 + (y - R)^2 = R^2$$

Or, les développements limités donnent :

$$\begin{aligned} x(t)^2 + (y(t) - R)^2 &\underset{t \to t_0}{=} R^2 + \|\gamma'(t_0)\|^2 (t - t_0)^2 - 2R\beta \frac{(t - t_0)^2}{2} + o\left((t - t_0)^2\right) \\ &\underset{t \to t_0}{=} R^2 + \left(\|\gamma'(t_0)\|^2 - \beta R\right)(t - t_0)^2 + o\left((t - t_0)^2\right) \end{aligned}$$

Puisque $(\gamma'(t_0), \gamma''(t_0))$ est libre, $\beta \neq 0$ donc le cercle qui approche le mieux la courbe en $\gamma(t_0)$ est le cercle de rayon $R = \dfrac{\|\gamma'(t_0)\|^2}{\beta}$ et de centre $(0, R)$ dans le repère de Serret-Frenet.



Proposition 2 : (mêmes notations)
Soit $t_0 \in [a,b]$ tel que $\gamma'(t_0) \neq 0$.
Alors :

$$\boxed{\kappa(t_0) = \frac{det(\gamma'(t_0), \gamma''(t_0))}{\|\gamma'(t_0)\|^3}}$$

En particulier, en notant $\gamma(t_0) = (x(t_0), y(t_0))$, on a :

$$\boxed{\kappa(t_0) = \frac{x'(t_0)y''(t_0) - x''(t_0)y'(t_0)}{(x'(t_0)^2 + y'(t_0)^2)^{3/2}}}$$

Démonstration :
Puisque le déterminant est invariant par changement de base directe :

$$\frac{det\left(\gamma'(t_0), \gamma''(t_0)\right)}{\|\gamma'(t_0)\|^3} = \frac{\begin{vmatrix} \|\gamma'(t_0)\| & \alpha \\ 0 & \beta \end{vmatrix}}{\|\gamma'(t_0)\|^3} = \frac{\beta}{\|\gamma'(t_0)\|^2} = \kappa(t_0)$$

Proposition 3 :
Soit $\Phi \in \mathcal{C}^2(\mathbb{R}^2, \mathbb{R})$. Notons $C = \left\{(x,y) \in \mathbb{R}^2 \mid \Phi(x,y) = 0\right\}$.
Soit $\gamma : s \longmapsto (x(s), y(s)) \in \mathcal{C}^2([a,b], \mathbb{R}^2)$ une paramétrisation de C telle que pour tout $s \in [a,b]$, $\|\gamma'(s)\| = 1$ (résultat admis).
Supposons que pour tout $s \in [a,b]$, $\gamma'(s) \neq 0$ et pour tout $(x,y) \in C$, $\nabla\Phi(x,y) \neq 0$.
Alors :

$$\boxed{\forall s \in [a,b],\ \kappa(s) = \operatorname{div}\left(\frac{\nabla\Phi(x(s),y(s))}{\|\nabla\Phi(x(s),y(s))\|}\right)}$$

Démonstration :
Pour simplifier les notations, notons $\Phi_x = \dfrac{\partial\Phi}{\partial x}(x(s),y(s))$, $\Phi_y = \dfrac{\partial\Phi}{\partial y}(x(s),y(s))$, $x_s = x'(s)$, $y_s = y'(s)$ pour $s \in [a,b]$.

$$\forall s \in [a,b],\ \Phi(x(s),y(s)) = 0$$

donne, en dérivant par rapport à s :

$$\Phi_x x_s + \Phi_y y_s = 0$$

Puis :

$$\Phi_{xx}(x_s)^2 + 2\Phi_{xy}x_s y_s + \Phi_{yy}(y_s)^2 + \Phi_x x_{ss} + \Phi_y y_{ss} = 0$$

D'où, en remplaçant dans l'expression $\kappa(s) = \dfrac{x_s y_{ss} - x_{ss} y_s}{(x_s^2 + y_s^2)^{3/2}}$, on obtient :

$$\kappa(s) = \frac{\Phi_{xx}(\Phi_y)^2 - 2\Phi_{xy}\Phi_x\Phi_y + \Phi_{yy}(\Phi_x)^2}{((\Phi_x)^2 + (\Phi_y)^2)^{3/2}}$$

Finalement, on vérifie que :

$$\forall s \in [a, b],\ \text{div}\left(\frac{\nabla\Phi(x(s), y(s))}{\|\nabla\Phi(x(s), y(s))\|}\right) = \kappa(s)$$

```python
from scipy.ndimage import gaussian_filter, sobel
import numpy as np
import matplotlib.pyplot as plt
from skimage import color, io
import matplotlib.image as mimg
import os
import cv2

dossier='C:/sportsmot_publish/dataset/foot1'

def egalise_histo(image):
    """
    Egalise l'histogramme de l'image (prétraitement).
    """
    gris = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    egalise = cv2.equalizeHist(gris)
    return egalise

def rectangle(image, largeur, longueur):
    """
    Initialise le contour sous la forme d'un rectangle centré au milieu de l'image.
    """
    img_h, img_w = image.shape[:2]
    centre_x, centre_y = img_w // 2, img_h // 2
    haut_gauche = (centre_x - largeur//2, centre_y - longueur // 2)
    bas_droite = (centre_x + largeur // 2, centre_y + longueur // 2)
    cv2.rectangle(image, haut_gauche, bas_droite, color = (0, 255, 0), thickness = 2)
    return image
```

Code informatique


```python
    def ellipse(centre, rx, ry, n):
        """
        Initialise le contour sous la forme d'une ellipse dont la position du centre et
        les longueurs du petit et du grand axe sont données.
        """
        resx = np.zeros(n)
        resy = np.zeros(n)
        for i in range(n):
            resx[i] = centre[0] + rx * np.sin(2 * np.pi * i / n)
            resy[i] = centre[1] + ry * np.cos(2 * np.pi * i / n)
        return resx, resy

    def coord_objet(X, Y):
        """
        Entrée : Coordonnées des points du contour final de l'image précédente.
        Sortie : Coordonnées de l'objet.
        """
        n = len(X)
        return sum(X) / n, sum(Y) / n

    def dans_image(X, Y, img):
        """
        Vérifie si chaque point de coordonnées (x,y) est bien dans l'image.
        Sinon, renvoie le point sur le bord de l'image le plus proche.
        """
        for i in range(len(X)):
            if Y[i] < 0:
                Y[i] = 0
            if Y[i] > img.shape[0] - 1:
                Y[i] = img.shape[0] - 1
            if X[i] < 0:
                X[i] = 0
            if X[i] > img.shape[1] - 1:
                X[i] = img.shape[1] - 1
        return X, Y
```

Code informatique


```python
def creer_M(alpha, beta, n):
    """
    Crée la matrice M correspondant à la dérivation discrète de v obtenue par la méthode
    des différences finies pour les paramètres alpha et beta.
    """
    assert n > 5
    M2 = np.array([[0] * n for _ in range(n)])
    M4 = np.array([[0] * n for _ in range(n)])
    for i in range(n):
        M2[i][i] = -2
        M4[i][i] = 6
        if i > 0:
            M2[i][i-1] = 1
            M2[i-1][i] = 1
            M4[i][i-1] = -4
            M4[i-1][i] = -4
        if i > 1:
            M4[i][i-2] = 1
            M4[i-2][i] = 1
        if i < n-1:
            M4[i][i+1] = -4
            M4[i+1][i] = -4
        if i < n-2:
            M4[i][i+2] = 1
            M4[i+2][i] = 1
    M2[0][-1] = 1
    M2[n-1][0] = 1
    M4[0][n-2] = 1
    M4[0][n-1] = -4
    M4[1][n-1] = 1
    M4[n-2][0] = 1
    M4[n-1][0] = -4
    M4[n-1][1] = 1
    return alpha * M2 - beta * M4
```

```
 98    def Fx(X, Y, img, lambd, grad2x):
 99        """
100        Entrée : Coordonnées X,Y des points du contour, l'image à segmenter, lambda
101        le poids de l'énergie externe, le gradient du module du gradient de l'intensité
102        suivant x.
103        Sortie : terme de force Fx correspondant à l'énergie externe (liste de taille n).
104        """
105        X, Y = dans_image(X, Y, img)
106        return grad2x[ (Y.round().astype(int), X.round().astype(int)) ] * lambd
107
108    def Fy(X, Y, img, lambd, grad2y):
109        """
110        Entrée : Coordonnées X,Y des points du contour, l'image à segmenter, lambda
111        le poids de l'énergie externe, le gradient du module du gradient de l'intensité
112        suivant y.
113        Sortie : terme de force Fy correspondant à l'énergie externe (liste de taille n).
114        """
115        X, Y = dans_image(X, Y, img)
116        return grad2y[ (Y.round().astype(int), X.round().astype(int)) ] * lambd
```

Code informatique


```
117    def evolution_contour(img, X, Y, alpha, beta, gamma, n_iters, lambd, Fnorm, sigma):
118        """
119        Entrée : image, coordonnées initiales du contour (X,Y),
120        paramètre alpha (élasticité), beta (rigidité), gamma (pas temporel), n_iters
121        (nombre d'itérations), lambda (poids de l'énergie liée à l'image),
122        Fnorm (force normale), sigma (écart-type du flou gaussien).
123        Sortie : Liste finale des positions X,Y des points du contour actif.
124        """
125        n = len(X)
126        A = creer_M(alpha, beta, n)
127        M = np.eye(n) - gamma * A
128        InvM = np.inv(M)
129        #Définition des vecteurs normaux pour la force normale
130        X_normal = np.zeros(n)
131        Y_normal = np.zeros(n)
132        im_traitee = gaussian_filter(img, sigma)
133        #Calcul du gradient de l'image
134        grad_x = sobel(im_traitee, axis = 0)
135        grad_y = sobel(im_traitee, axis = 1)
136        norme_grad = np.sqrt(grad_x**2 + grad_y**2)
137        grad_x2 = sobel(norme_grad, axis = 0)
138        grad_y2 = sobel(norme_grad, axis = 1)
139        for i in range(n_iters):
140            #Calcul des vecteurs normaux au contour
141            for j in range(n-1):
142                dX = X[j+1] - X[j]
143                dY = Y[j+1] - Y[j]
144                norme = np.sqrt(dX**2 + dY**2)
145                X_normal[j] = - dX / norme
146                Y_normal[j] = - dY / norme
147            #Evolution suivant le schéma itératif
148            Xiter = np.dot(InvM, X - gamma * (Fx(X, Y, img, lambd, grad_x2) + Fnorm * X_normal))
149            Yiter = np.dot(InvM, Y - gamma * (Fy(X, Y, img, lambd, grad_y2) + Fnorm * Y_normal))
150            X, Y = Xiter.copy(), Yiter.copy()
151        return (X,Y)
```

```
152    #Contours actifs géométriques, méthode des courbes de niveau.
153
154    def grad(f):
155        """
156        Renvoie le gradient de f (avec numpy).
157        """
158        return np.array(np.gradient(f))
159
160    def norme(x, axis = 0):
161        """
162        Renvoie la norme de x (vecteur).
163        """
164        return np.sqrt(np.sum(np.square(x), axis = axis))
165
166    def fonction_darret(I):
167        """
168        Correspond à la fonction g définie dans la présentation.
169        """
170        return 1 / (1 + norme(grad(I))**2)
171
172    def div(fx, fy):
173        """
174        Renvoie la divergence du vecteur (fx, fy).
175        """
176        fyy, fyx = grad(fy)
177        fxy, fxx = grad(fx)
178        return fxx + fyy
```

```
179    def courbure(f):
180        """
181        Renvoie la courbure de la courbe f(x, y) = 0.
182        """
183        fy, fx = grad(f)
184        norme = np.sqrt(fx**2 + fy**2)
185        Nx = fx / norme
186        Ny = fy / norme
187        return div(Nx, Ny)
188
189    def phi_init(I):
190        """
191        Initialise la fonction Phi
192        de sorte que le contour initial
193        englobe la quasi-totalité de l'image.
194        """
195        phi = np.ones(I.shape[:2])
196        phi[1:-1, 1:-1] = -1
197        return phi
198
199    #Paramètres
200    n_iter = 7000
201    k = 100
202    gamma = 3
203    sigma = 1.5
204
205    img = io.imread('C:/Users/arthr/Documents/football1.png')
206    img = color.rgb2gray(img)
207    img = img - np.mean(img)
208    img = egalise_histo(img)
209    img_lisse = gaussian_filter(img, sigma)
```

```
210    g = fonction_darret(img_lisse)
211    dg = grad(g)
212    phi = phi_init(img_lisse)
213
214    #Processus d'évolution de Phi.
215    for i in range(n_iter):
216        dphi = grad(phi)
217        dphi_norme = norme(dphi)
218        kappa = courbure(phi)
219        lissage = g * kappa * dphi_norme
220        force_normale = g * dphi_norme * k
221        dphi_t = lissage + force_normale
222
223        phi = phi + gamma * dphi_t
224
225    plt.figure()
226    plt.imshow(img, cmap = 'gray')
227    plt.contour(phi, levels = [0], colors = 'r')
228    plt.contour(phi_init(img_lisse), levels = [0], colors = 'b')
229    plt.axis('off')
230    plt.show()
```

Code informatique


```
231     def suivi_final(Coord_init, rx, ry, n, alpha, beta, gamma, n_iters, lambd, Fnorm, sigma):
232         contour = []
233         # Création du contour initial pour chaque objet.
234         im_init = mimg.imread(os.path.join(dossier, os.listdir(dossier)[0]))
235         # Parcours de toutes les images de la vidéo.
236         for nom in os.listdir(dossier):
237             chemin = os.path.join(dossier, nom)
238             img = mimg.imread(chemin)
239             img = egalise_histo(img)
240             img = gaussian_filter(img, sigma)
241             for x0, y0 in Coord_init:
242                 X, Y = ellipse((x0, y0), rx, ry, n)
243                 contour.append((X, Y))
244             contour_init = contour.copy()
245             fig = plt.figure()
246             ax = fig.add_subplot(111)
247             ax.imshow(img)
248             ax.set_xticks([])
249             ax.set_yticks([])
250             ax.set_xlim(0, im_init.shape[1])
251             ax.set_ylim(im_init.shape[0], 0)
252             for i in range(len(contour_init)):
253                 contour[i] = evolution(img, contour[i][0], contour[i][1], alpha, beta,
254                 gamma, n_iters, lambd, Fnorm, sigma)
255                 # Trace le contour initial.
256                 ax.plot(contour_init[i][0], contour_init[i][1], c = 'green', lw = 2)
257                 # Trace le contour final.
258                 ax.plot(contour[i][0], contour[i][1], c = 'red', lw = 2)
259                 plt.fill(contour[i][0], contour[i][1], c = 'red', alpha = 0.5)
260                 #Actualise les coordonnées de l'objet en mouvement.
261                 x_obj, y_obj = coord_objet(contour[i][0], contour[i][1])
262                 Coord_init[i] = (x_obj, y_obj)
263             plt.show()
```

Bibliographie

- **[MOT1]** Cui et al. in SportsMOT :
  A Large Multi-Object Tracking Dataset in Multiple Sports Scenes (2023).

- **[SOA2]** Jalal, A., Singh, V. : :
  The State-of-the-Art in Visual Object Tracking Informatica 36 (2012).

- **[KASS3]** Kass, M., Witkin, A., Terzopoulos, D. Snakes :
  Active contour models. Int J Comput Vision 1, 321–331 (1988).

- **[CAS4]** Caselles, V., Catté, F., Coll, T. et al. :
  A geometric model for active contours in image processing. Numer. Math. 66, 1–31 (1993).