Institut
Mines-Telecom

Estimation du mouvement

B. Pesquet-Popescu, M. Cagnazzo

Sigma 207

Plan

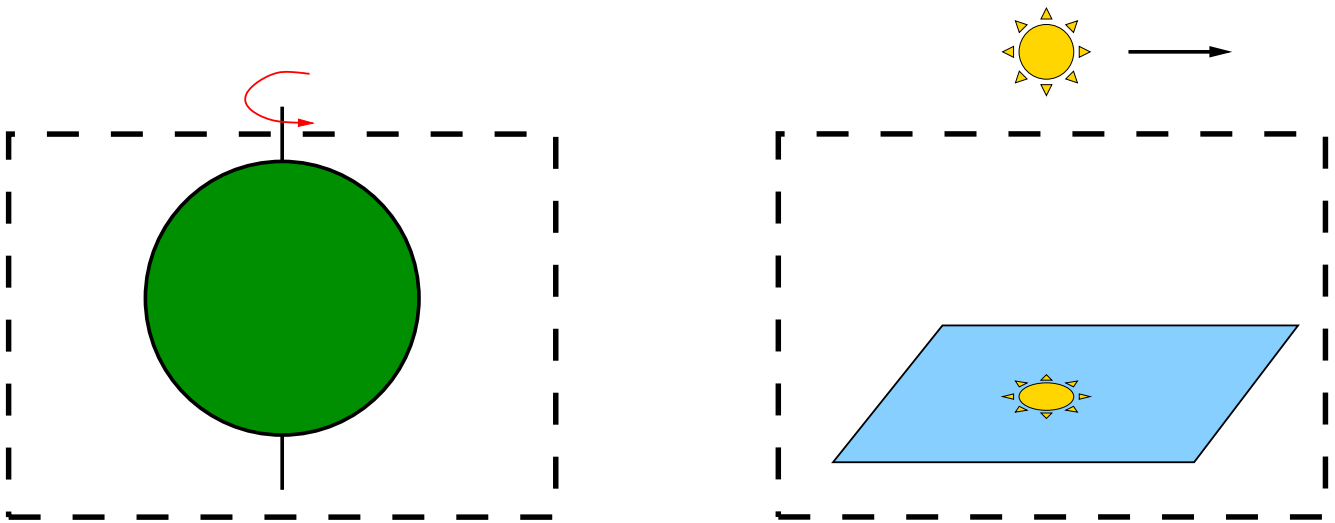
Introduction : Le mouvement

- Une séquence vidéo est une source très riche d'information
- Le mouvement fournit une grande partie de cette information
  - Le mouvement permet de identifier les objets
  - Les propriétés de l'images ont une corrélation élevée le long du mouvement
- *Motion detection* : décider si il y a eu de mouvement
- *Motion estimation* : mesurer le mouvement

Mouvement et mouvement apparent

- Champ de mouvement 2D :
  - Projection du mouvement physique sur le plan d'image
- Flux optique :
  - Mouvement apparent d'un "pattern" (schéma) de valeur de luminance
- Le champ de mouvement et le flux optique souvent coïncident, mais cela n'est pas toujours nécessairement vrai

Mouvement et mouvement apparent



Boule en rotation pure : flux optique nul, champ de vitesse réel non nul

Surface réfléchissante éclairée par une source en mouvement : flux optique non nul, champ de vitesse réel nul

Applications de l'estimation de mouvement

- Traitement
  - Segmentation
  - Débruitage
  - Restauration
  - Augmentation de la cadence d'image (frame-rate up-conversion)
  - Surveillance (protection)
- Compression
  - Prédiction de l'image courante à partir des images déjà codées
  - Compromis entre précision et cout de codage
  - Relation entre le mouvement estimé et le "vrai" mouvement
- Complexité

Classification des techniques

- Modèles
- Critères d'estimation
- Stratégies de recherche

Plan

Plan

Modèles spatiales de mouvement

- Objectif : estimer le mouvement apparent (mouvement 2D)
- Projection du mouvement 3D des objets
  - Modèle de formation de l'image : perspective, projection orthogonale
  - Modèle du mouvement 3D de l'objet (translation rigide, rotation, déformation)
- Mouvement 3D de la caméra
  - Peut être estimé ou mesuré

Modèles spatiales de mouvement des objets

Modèle translationale

- Projection orthogonale d'un objet rigide en translation
- Soit $\mathbf{v}(\mathbf{p})$ la vitesse instantané à la position $\mathbf{p} = [x, y]^T$ sur le plan d'image
- Les composantes $\mathbf{v}_x$ et $\mathbf{v}_y$ de $\mathbf{v}$ ne varient pas avec la position $\mathbf{p}$ et dépendent de la géométrie de la caméra et du mouvement de l'objet
- On peut écrire :

$$\mathbf{v}(\mathbf{p}) = \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

- Ce modèle est caractérisé par 2 paramètres, b_1 et b_2

Modèles spatiales de mouvement des objets

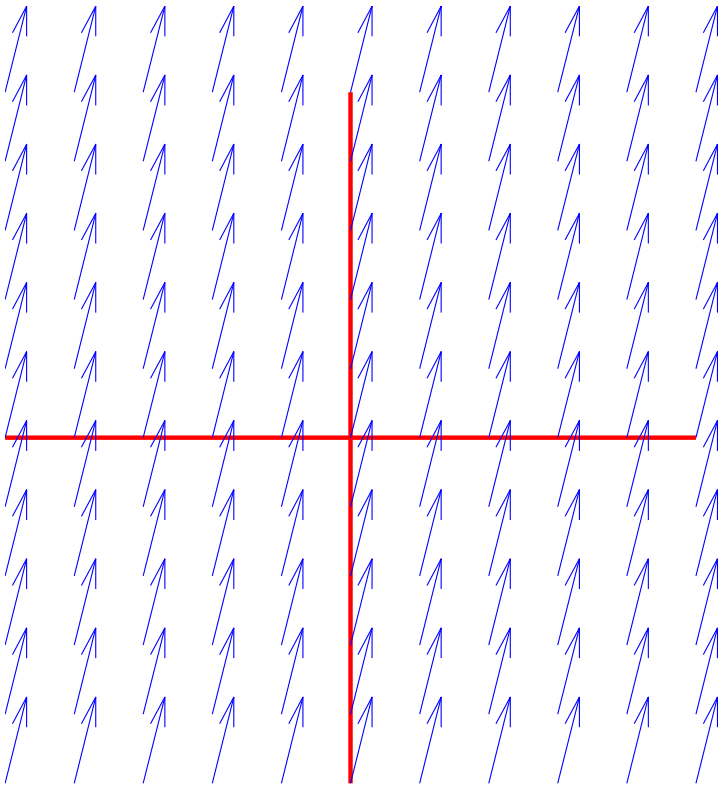
Modèle affine

- Un modèle légèrement plus complexe est le modèle affine :

$$\mathbf{v}(\mathbf{p}) = \mathbf{b} + \mathbf{B}\mathbf{p} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} b_3 & b_4 \\ b_5 & b_6 \end{bmatrix} \mathbf{p}$$

- La translation est un cas particulier du modèle affine
- Ce modèle est caractérisé par 6 paramètres, ou équivalemment, par **b** et **B**.

Modèle affine : exemples

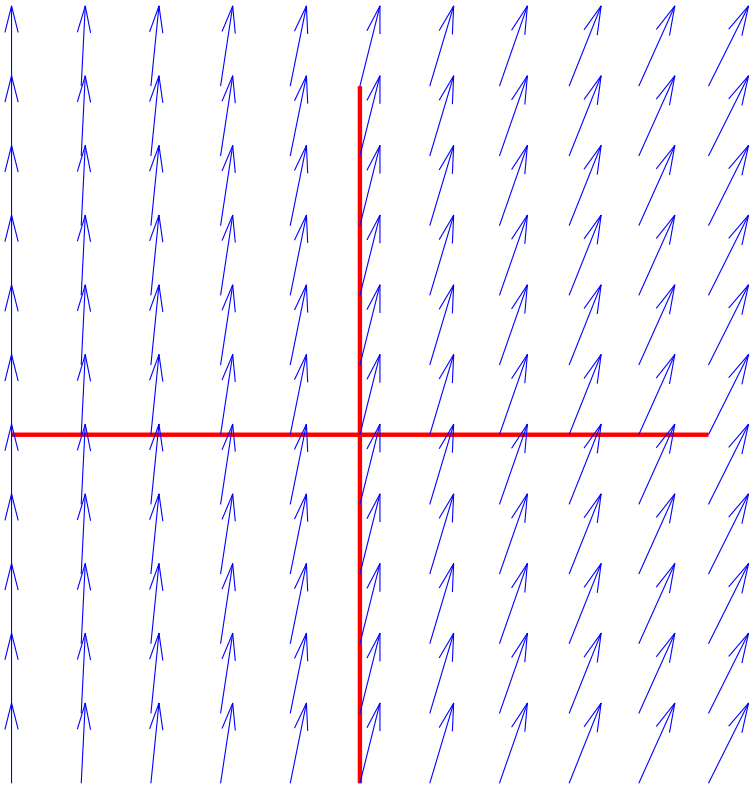


Translation :

$$\mathbf{b} = \begin{bmatrix} 0.5 \\ 2 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{v}_x = 0.5$$
$$\mathbf{v}_y = 2$$

Modèle affine : exemples

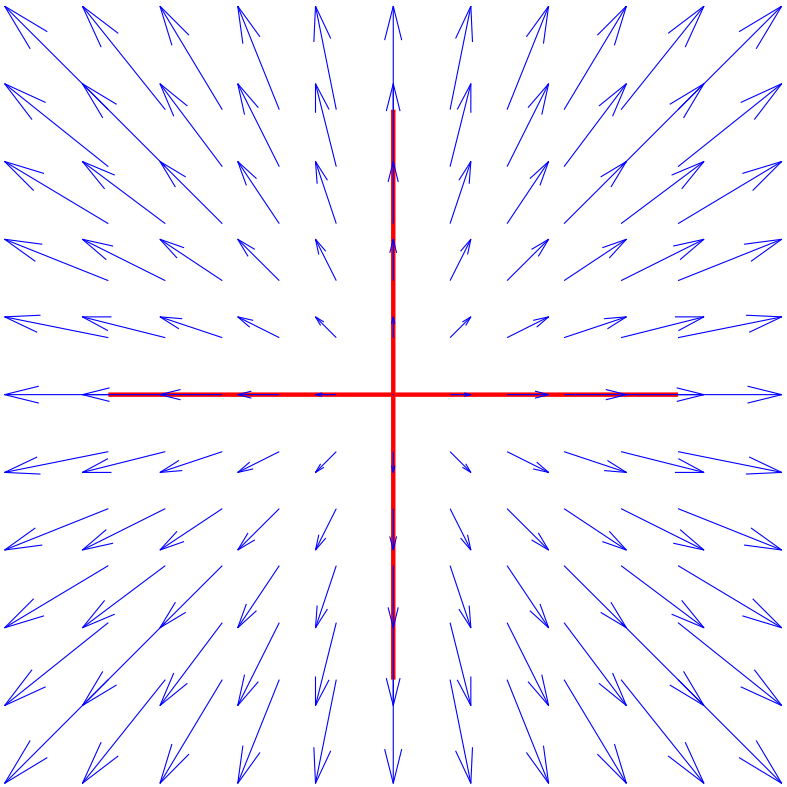


Mouvement affine:

$$\mathbf{b} = \begin{bmatrix} 0.5 \\ 2 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{v}_x = 0.5 + 0.1x$$
$$\mathbf{v}_y = 2$$

Modèle affine : exemples

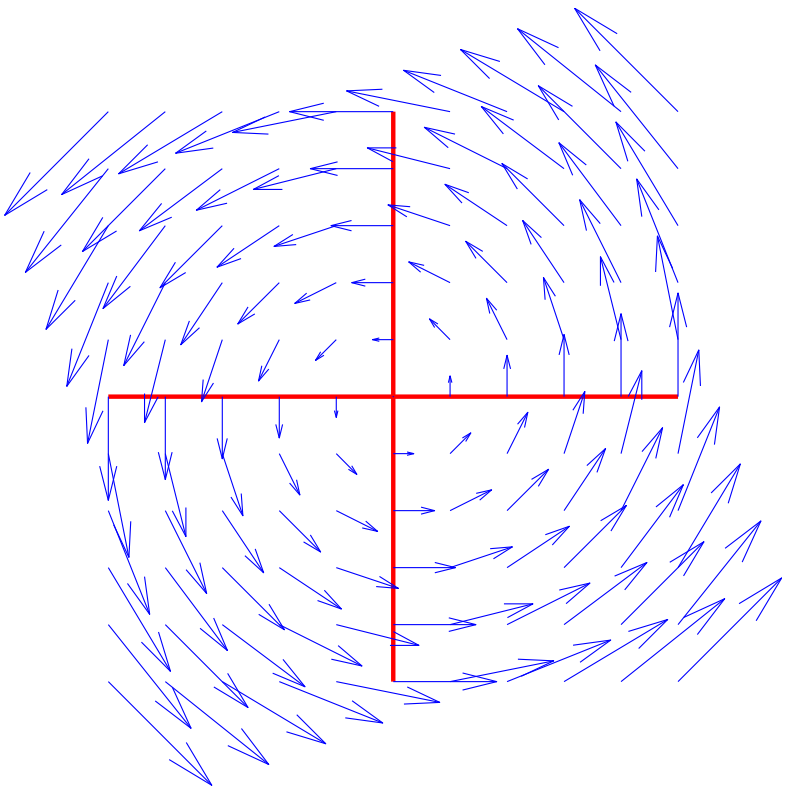


Divergence :

$$\mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$

$$\mathbf{v}_x = 0.5x$$
$$\mathbf{v}_y = 0.5y$$

Modèle affine : exemples

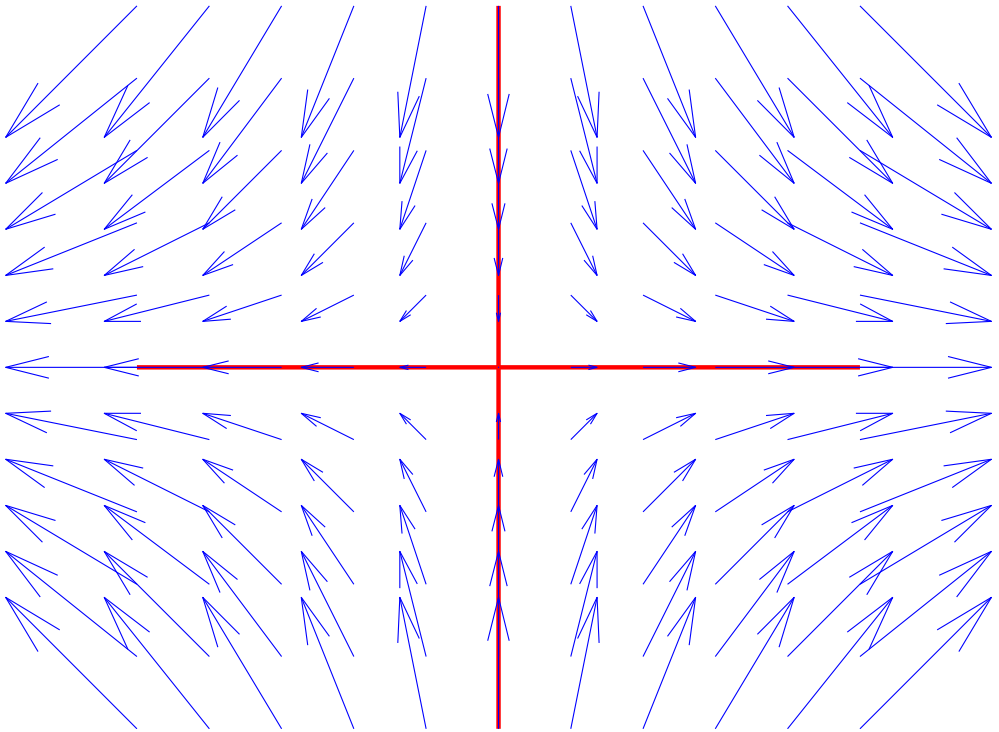


Rotation :

$$\mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & -0.5 \\ 0.5 & 0 \end{bmatrix}$$

$$\mathbf{v}_x = -0.5y$$
$$\mathbf{v}_y = 0.5x$$

Modèle affine : exemples

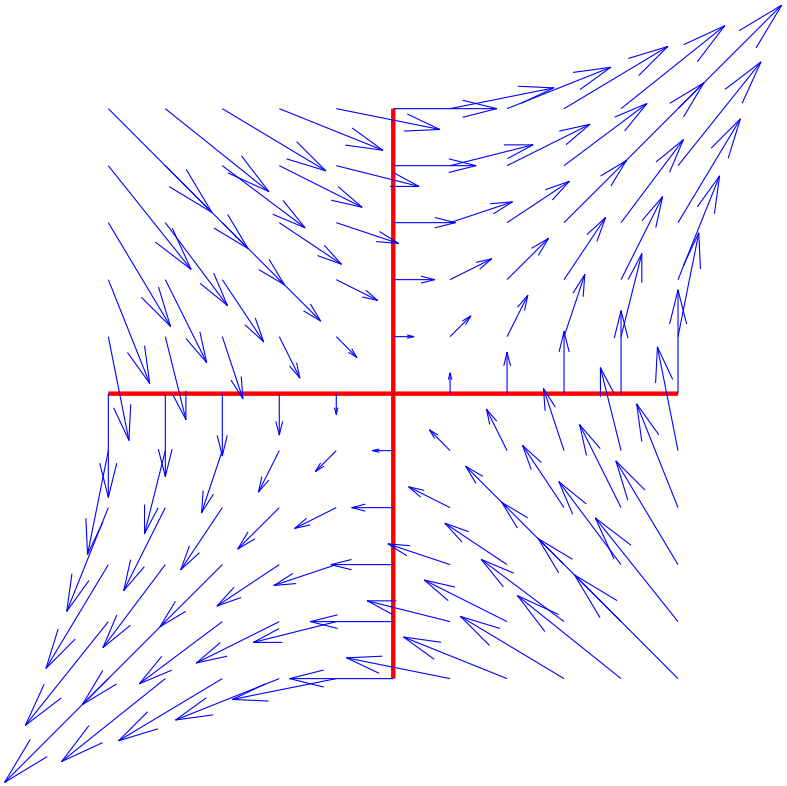


Mouvement hyperbolique (1) :

$$\mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0.5 & 0 \\ 0 & -0.5 \end{bmatrix}$$

$$\mathbf{v}_x = 0.5x$$
$$\mathbf{v}_y = -0.5y$$

Modèle affine : exemples



Mouvement hyperbolique (2) :

$$\mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0.5 \\ 0.5 & 0 \end{bmatrix}$$

$$\mathbf{v}_x = 0.5y$$
$$\mathbf{v}_y = 0.5x$$

Modèle affine : bilan

- Le modèle affine est simple et dépende de peu de paramètres
- Il peut être utilisé pour décrire le mouvement de petites régions
- L'estimation des paramètres de ce modèle est relativement robuste

Modèles temporels : modèle linéaire

- Les trajectoires des points d'une image dans l'espace $\mathbb{R}^3$ peuvent être très complexes
- Le modèle le plus simple est celui des trajectoires linéaires
- On considère l'intervalle temporel $[t_0, t_1]$
- Une trajectoire linéaire est :

$$\mathbf{x}(t) = \mathbf{p} + \mathbf{v}(\mathbf{p})(t - t_0)$$

- On définit $\mathbf{d}(\mathbf{p}, T) = \mathbf{v}(\mathbf{p}) \cdot T$ le vecteur de déplacement en $\mathbf{p}$ après un temps T
- Avec le modèle temporel linéaire, il faut trouver 2 paramètres (composantes de $\mathbf{v}$) par pixel

Modèles temporels : modèle linéaire

- Le modèle linéaire est simple
- Il correspond à un mouvement sans accélération sur l'intervalle $[t_0, t_1]$
- On peut considérer un modèle linéaire par morceaux : c'est la suite de trajectoires linéaires dont chacune doit être estimée
- Longueur de l'intervalle $[t_0, t_1]$: compromis entre précision et complexité

Modèles temporels : modèle quadratique

- Une extension naturelle du modèle linéaire est le modèle quadratique
- Ce modèle correspond à un mouvement uniformément accéléré
- On considère l'intervalle temporel $[t_0, t_1]$
- Une trajectoire quadratique est :

$$\mathbf{x}(t) = \mathbf{p} + \mathbf{v}(\mathbf{p})(t - t_0) + \frac{1}{2}\mathbf{a}(\mathbf{p})(t - t_0)^2$$

- Avec ce modèle temporel linéaire, il faut trouver 4 paramètres (composantes de **v** et de **a**) par pixel

Modèles spatio-temporels

- Au lieu de calculer les paramètres du modèle temporel pour tout pixel, on peut utiliser le modèle spatial
- Par exemple, on peut utiliser le modèle affine pour remplacer **v** dans le modèle linéaire ou quadratique, et cela pour tous les pixels d'un même objet
- Il reste le problème de comment trouver les pixels qui font partie d'un même objet
- Solutions possibles : segmentation, choix d'une région de support

Région de support

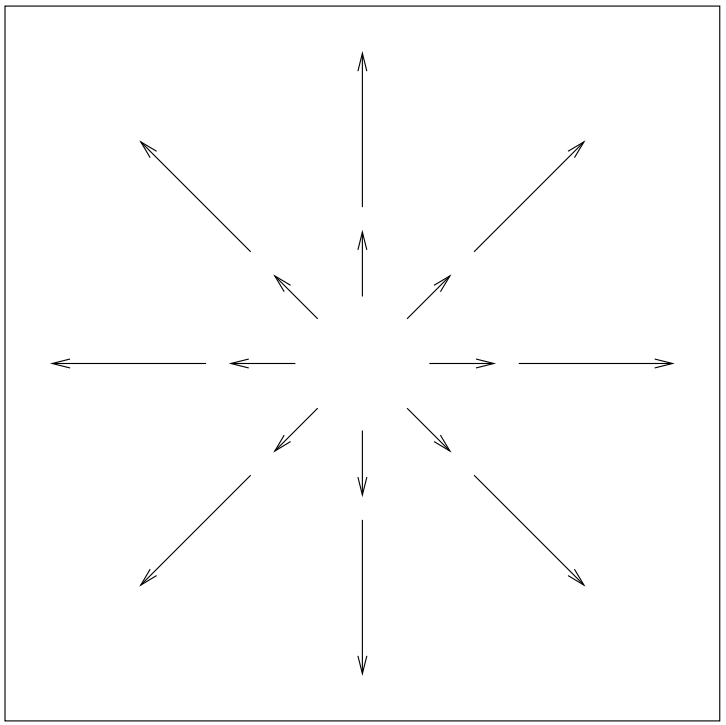
- Les points sur lesquels on applique un modèle (spatial ou temporel) de mouvement forment la région de support, $\mathcal{R}$
- Les choix du modèle et de la région sont critique
- Pour un modèle donné, choisir une région petite augmente la précision, mais aussi la complexité
  - Sur une région petite un modèle simple est raisonnablement correct
  - Le nombre de paramètres par région dépend uniquement du modèle, mais le nombre de régions est inversement proportionnelle à la taille des régions

Région de support

Il y a typiquement quatre types de régions de support

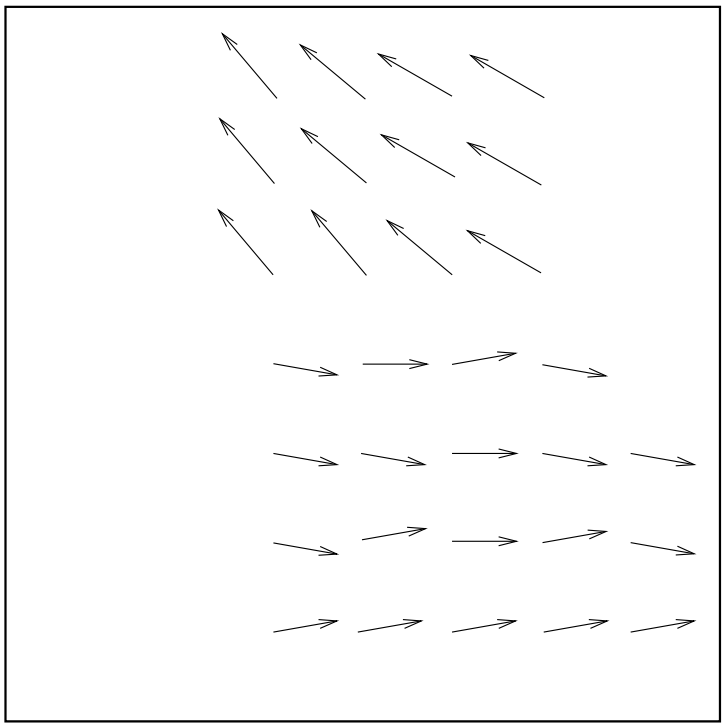
- $\mathcal{R}=$ Toute l'image
- $\mathcal{R}=$ Un pixel
- $\mathcal{R}=$ Un bloc rectangulaire
- $\mathcal{R}=$ Région de forme arbitraire

Région de support = l'image



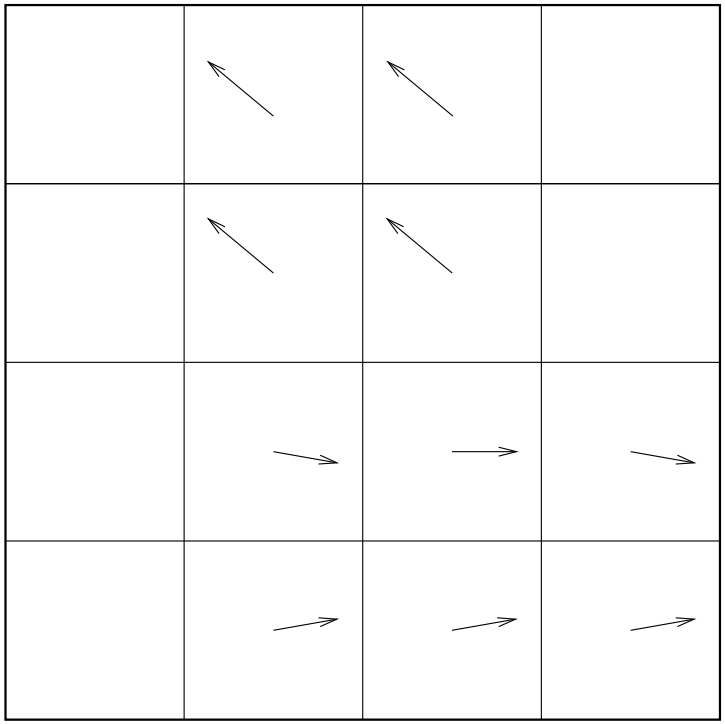
- Un seul modèle est appliqué à tous les points de l'image
- Ce cas s'adapte bien à l'estimation du mouvement provoqué par la caméra
- Ce le modèle les plus contraint (peu de champs de mouvement peuvent être représentés)...
- ... mais aussi le plus simple à estimer

Région de support = un pixel



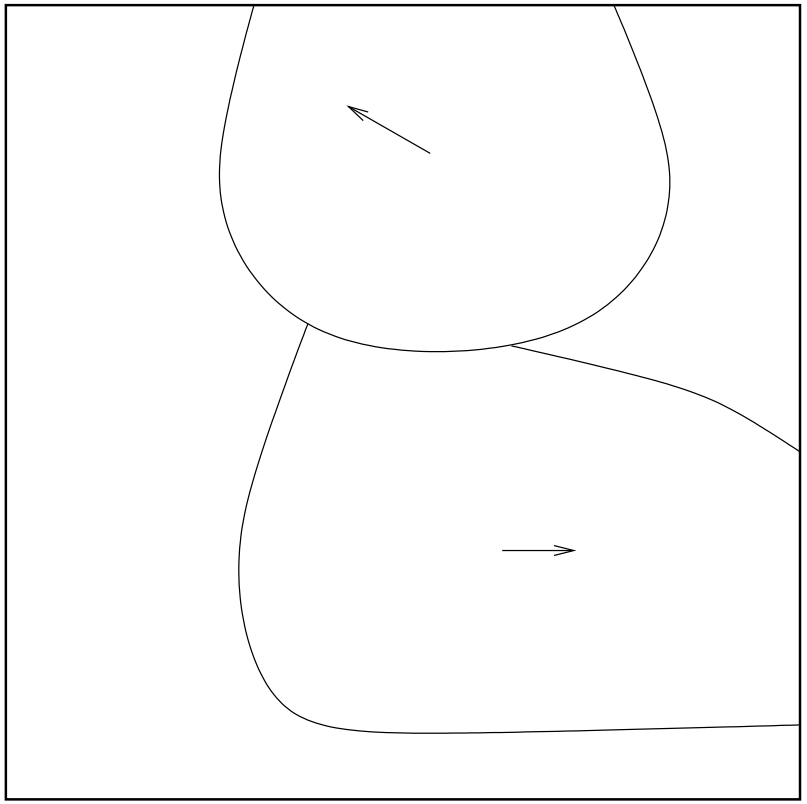
- Représentation *dense* du mouvement
- Typiquement on utilise un modèle spatial de translation avec un modèle temporel linéaire
- Le champ de mouvement est donc représenté par 2 paramètres par pixel et par intervalle temporel
- Complexité élevée

Région de support = un bloc



- Pour chaque bloc on applique le modèle de mouvement avec des paramètres spécifiques au bloc
- Taille du bloc : compromis entre complexité et précision
- Typiquement on utilise un modèle de translation (un vecteur par bloc)
- Simple, mais incapable de décrire correctement certains cas : zoom, rotation
- Variable block-size

Région de support = un objet



- Modèle qui essaie de reproduire le mouvement 3D d'un objet
- Le modèle affine décrit normalement très bien le mouvement d'un objet
- Comment effectuer la segmentation ?

Plan

Modèles de mouvement

Les techniques d'estimation
- Méthodes variationnelles
  - Flux optique
  - Méthodes "pel-based"
- Méthodes par blocs
  - Critères
  - Stratégie
  - Techniques avancées
- Méthodes paramétriques

Applications

Méthodes variationnelles

- État de l'art pour l'estimation du mouvement physique sous-jacent à la vidéo
- Résultat : un champ dense (un vecteur par pixel)
- Principes : terme de régularisation et terme d'attache au données
- Formulation typique :

$$\min_{u,v} \{R(u,v) + D[f_0, f_1, (u,v)]\}$$

Flux optique

- Le mouvement est observé sur la base des variation de luminosité
- Hypothèse : l'intensité lumineuse ne varie pas le long de la trajectoire $\mathbf{x}(t)$ du mouvement :

$$f(x,y,t+T) = f(x - c(x,y), y - d(x,y), t) \tag{1}$$

$$\mathbf{x}(t+T) = [x,y]^T = \mathbf{x}(t) + \mathbf{D} \qquad \mathbf{D}(x,y) = \begin{bmatrix} c(x,y) \\ d(x,y) \end{bmatrix}$$

- En pratique, on devrait considérer que :
  - Le signal vidéo est échantillonné en temps et espace
  - L'équation (1) n'est pas à la rigueur satisfaite pour le bruit, l'aliasing, les défauts d'acquisition, etc.

Flux optique

Considérons le cas continu. Le champ de vitesse est défini comme :

$$\mathbf{V}(x,y) = \lim_{T \to 0} \frac{\mathbf{D}(x,y)}{T} = \begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix}$$

Reprenons l'équation (1), avec développement de Taylor :

$$f(\mathbf{p}, t+T) = f(\mathbf{p}, t) - [c(\mathbf{p}) f_x(\mathbf{p}, t) + d(\mathbf{p}) f_y(\mathbf{p}, t)] + o[\|\mathbf{D}(\mathbf{p})\|]$$

$$\frac{f(\mathbf{p}, t+T) - f(\mathbf{p}, t)}{T} = -\frac{\mathbf{D}}{T} \nabla f + \frac{o[\|\mathbf{D}(\mathbf{p})\|]}{T}$$

$$f_t = -\mathbf{V} \nabla f$$

$$\boxed{u f_x + v f_y + f_t = 0}$$

Flux optique

- L'équation $uf_x + vf_y + f_t = 0$ (en forme continue ou discrétisé) est appelée équation du *flux optique*
- Elle est à la base de plusieurs techniques d'estimation de mouvement
- Néanmoins, c'est une équation avec deux inconnus u et v
- Donc elle ne permet que de connaitre la composante de **V** parallèle au gradient spatiale de l'image

Flux optique

- Des contraintes supplémentaires sont nécessaires pour résoudre cette équation
- En plus, la présence du bruit et autres dégradations font en sorte que cette équation ne soit pas parfaitement satisfaite
- Donc on minimise l'énergie de $uf_x + vf_y + f_t$ sous contrainte

Flux optique

Algorithme de Horn-Schunck

Horn et Schunck ont introduit une contrainte très raisonnable sur la variation totale, sur une région $\mathcal{R}$

$$\iint_{\mathcal{R}} (uf_x + vf_y + f_t)^2 \, dx \, dy = \min$$

$$\iint_{\mathcal{R}} \|\nabla u\|^2 + \|\nabla v\|^2 \, dx \, dy \leq \tau$$

Solution : méthode de Lagrange

$$J = \iint_{\mathcal{R}} (uf_x + vf_y + f_t)^2 \, dx \, dy +$$

$$\lambda \left[\iint_{\mathcal{R}} (\|\nabla u\|^2 + \|\nabla v\|^2) \, dx \, dy - \tau \right]$$

Flux optique

Algorithme de Horn-Schunck

Problème équivalent : minimiser (par rapport à u et v)

$$J = \iint_{\mathcal{R}} (uf_x + vf_y + f_t)^2 + \lambda(\|\nabla u\|^2 + \|\nabla v\|^2)\, dx\, dy$$

Or on sait que pour minimiser : $\iint_{\mathcal{R}} \Theta(x, y, \omega, \omega_x, \omega_y)\, dx\, dy$
il faut imposer :

$$\frac{\partial \Theta}{\partial \omega} - \frac{\partial^2 \Theta}{\partial x \partial \omega_x} - \frac{\partial^2 \Theta}{\partial y \partial \omega_y} = 0$$

Flux optique

Algorithme de Horn-Schunck

On impose :

$$\frac{\partial \Theta}{\partial \omega} - \frac{\partial^2 \Theta}{\partial x \partial \omega_x} - \frac{\partial^2 \Theta}{\partial y \partial \omega_y} = 0$$

avec $\omega = u$ et $\omega = v$ on obtient respectivement :

$$\lambda \nabla_2 u = (u f_x + v f_y + f_t) f_x$$
$$\lambda \nabla_2 v = (u f_x + v f_y + f_t) f_y$$

Flux optique

Algorithme de Horn-Schunck

- Ce type d'optimisation globale est complexe
- Il faut résoudre un système d'équations différentielles aux dérivées partielles
- Le choix de λ est critique : cela influence la régularité du résultat ainsi que la stabilité de l'algorithme

Flux optique

Algorithme de Lucas-Kanade

- On assume un champs de mouvement très peu variable dans une petite région $\mathcal{R}(\mathbf{p}_0)$ aux alentour du pixel courant $\mathbf{p}_0$
- Donc, l'équation du flux optique sera presque satisfaite si on utilise le vecteur $(u(\mathbf{p}_0), v(\mathbf{p}_0))$ dans les points de $\mathcal{R}$
- Plus précisément, on peut dire que l'énergie de $u(\mathbf{p}_0)f_x(\mathbf{p}) + v(\mathbf{p}_0)f_y(\mathbf{p}) + f_t(\mathbf{p})$ aux alentour de $\mathbf{p}_0$ est faible

Flux optique

Algorithme de Lucas-Kanade

On appelle $\mathbf{p}_1, \ldots, \mathbf{p}_N$ les points de $\mathcal{R}(\mathbf{p})$. On peut écrire alors :

$$(u(\mathbf{p}), v(\mathbf{p})) = \arg \min_{(w,z)\in\mathcal{W}} J(w,z)$$

$$J(w,z) = \sum_{i=0}^{N} \alpha_i \left[w f_x(\mathbf{p}_i) + z f_y(\mathbf{p}_i) + f_t(\mathbf{p}_i)\right]^2$$

Où $\mathcal{W} \subset \mathbb{R}^2$ est l'ensemble des vecteurs candidats et α_i est un poids qui permet de tolérer des écarts plus grands pour les points de $\mathcal{R}$ plus lointains de $\mathbf{p}$

Flux optique

Bilan

- Les algorithmes de flux optique donnent des champs denses (un vecteur par pixel) : utile pour débruitage, restauration, surveillance, analyse
- Ils sont basés sur une expansion de Taylor au premier ordre : valable seulement si le mouvement est petit
- Les équations doivent être discrétisée avant de pouvoir les appliquer : cela peut comporter des erreurs en présence de bruit
- Les contraintes de variation totale ou de mouvement régulier donnent des résultats peu satisfaisants sur les contours des objets
- Problème au départ : comment trouver la région $\mathcal{R}$?

Approches "pel-recursive"

- Ces techniques estiment récursivement le mouvement qui minimise la "displaced frame difference" (DFD)
- On considère l'image déjà discrétisée :

$$f(x,y,t)|_{x=nL_1,y=mL_2,t=kT}$$

où L_1 et L_2 sont les périodes d'échantillonnage spatiale et T est la période d'échantillonnage verticale

- Pour simplicité, on considère $T = 1$
- La DFD est alors :

$$\begin{aligned} DFD = & f(nL_1, mL_2, k+1) \\ & - f(nL_1 - c(n,m), mL_2 - d(n,m), k) \end{aligned}$$

Approches "pel-recursive"

Méthode de Cafforio-Rocca

- La méthode de Cafforio-Rocca calcule un champ dense
- On parcourt l'image en "raster-scan order" (balayage par lignes horizontales)
- Pour chaque pixel (n, m), on fait les opérations suivantes :
  1. Initialisation du vecteur : $D^0_{n,m}$
  2. Validation : $D^1_{n,m}$
  3. Raffinement : $D^2_{n,m}$
- Ce dernier est retenu comme vecteur pour le pixel (n, m)

Méthode de Cafforio-Rocca

Initialisation

- La valeur d'initialisation $D^0_{n,m}$ est une fonction des vecteurs voisins déjà estimés
- Typiquement, on choisi simplement le vecteur associé au pixel précédent :

$$D^0_{n,m} = D_{n-1,m}$$

- Plus en général, $D^0_{n,m}$ est une fonction des vecteurs $D_{n-n_0,m-m_0}$ avec $n_0 \geq 0, m_0 \geq 0$
- Exemple : vecteur moyen dans un voisinage anti-causale

Méthode de Cafforio-Rocca

Initialisation

- Ce choix d'initialisation se base sur l'hypothèse de champ régulier
- Les vecteurs de mouvement de deux pixels voisins sont proches
- Cela permet de améliorer la qualité de l'estimation au fur et à mesure qu'on parcourt l'image
- Problème : les champs de mouvement ont de temps en temps de discontinuités brusques (contours d'objet)
- Solution : validation de l'initialisation

Méthode de Cafforio-Rocca

Validation

- Si on passe d'un objet à un autre, l'initialisation peut produire une DFD très élevée, plus élevée de celle qu'on aurait avec le vecteur nul
- Ceci est donc le critère de validation : l'initialisation est retenue seulement si elle est meilleure de $(0,0)$

$$A = \left| f_{nL_1, mL_2, k+1} - f_{nL_1 - c^0_{n,m}, mL_2 - d^0_{n,m}, k} \right|$$

$$B = \left| f_{nL_1, mL_2, k+1} - f_{nL_1, mL_2, k} \right|$$

$$\mathbf{D}^1_{n,m} = \begin{cases} \mathbf{D}^0_{n,m} & \text{si } A < B + \gamma \\ \mathbf{0} & \text{si } A \geq B + \gamma \end{cases}$$

Méthode de Cafforio-Rocca

Raffinement

- Le vecteur validé est raffiné
- On veut minimiser l'erreur de prédiction sous contrainte que la modification du vecteur soit petite
- Le vecteur final est donc :

$$\mathbf{D}^2_{n,m} = \mathbf{D}^1_{n,m} + \delta\mathbf{D}$$

- La correction est trouvée en minimisant le critère :

$$\begin{aligned} J(\delta\mathbf{D}) = [&f(nL_1, mL_2, k+1)+ \\ &-f(nL_1 - c^1_{n,m} - \delta c_{n,m}, mL_2 - d^1_{n,m} - \delta d_{n,m}, k)]^2 \\ &+\lambda\|\delta\mathbf{D}\|^2 \end{aligned}$$

Méthode de Cafforio-Rocca

Raffinement

Expansion au premier ordre du critère :

$$\begin{aligned} J(\delta\mathbf{D}) \approx & [f(nL_1, mL_2, k+1) - f(nL_1 - c^1_{n,m}, mL_2 - d^1_{n,m}, k) \\ & + \delta c_{n,m} f_x(nL_1 - c^1_{n,m}, mL_2 - d^1_{n,m}, k) \\ & + \delta d_{n,m} f_y(nL_1 - c^1_{n,m}, mL_2 - d^1_{n,m}, k)]^2 + \lambda \|\delta\mathbf{D}\|^2 \\ = & [e + \delta\mathbf{D}^T \varphi]^2 + \lambda \|\delta\mathbf{D}\|^2 \end{aligned}$$

avec :

$$\begin{aligned} e &= f(nL_1, mL_2, k+1) - f(nL_1 - c^1_{n,m}, mL_2 - d^1_{n,m}, k) \\ \varphi &= \nabla f(nL_1 - c^1_{n,m}, mL_2 - d^1_{n,m}, k) \end{aligned}$$

Pour simplicité on a supprimé la dépendance de e et du gradient compensé φ de la position n, m

Méthode de Cafforio-Rocca

Raffinement

Minimisation de $J(\delta\mathbf{D})$

$$\begin{aligned}\frac{\partial J}{\partial \delta \mathbf{D}} &= 2[e + \delta\mathbf{D}^T\varphi]\varphi + 2\lambda\delta\mathbf{D} \\ &= 2e\varphi + 2(\lambda\mathbf{I} + \varphi\varphi^T)\delta\mathbf{D} \\ \delta\mathbf{D}^* &= -e\left(\lambda\mathbf{I} + \varphi\varphi^T\right)^{-1}\varphi\end{aligned}$$

Pour calculer le raffinement optimal $\delta\mathbf{D}^*$ on utilise le lemme d'inversion.

Méthode de Cafforio-Rocca

Raffinement

Pour toute matrice inversible $\mathbf{M}$ et pour tout vecteur non nul $\mathbf{x}$,

$$\left(\mathbf{M}+\mathbf{x}\mathbf{x}^T\right)^{-1}=\mathbf{M}^{-1}-\frac{\mathbf{M}^{-1}\mathbf{x}\mathbf{x}^T\mathbf{M}^{-1}}{1+\mathbf{x}^T\mathbf{M}^{-1}\mathbf{x}}$$

Donc pour $\mathbf{M}=\lambda\mathbf{I}$ et $\mathbf{x}=\varphi$,

$$\begin{aligned}\left(\lambda\mathbf{I}+\varphi\varphi^T\right)^{-1}&=\frac{1}{\lambda}\mathbf{I}-\frac{\frac{1}{\lambda^2}\varphi\varphi^T}{1+\frac{1}{\lambda}\|\varphi\|^2}\\&=\frac{1}{\lambda}\left[\mathbf{I}-\frac{\varphi\varphi^T}{\lambda+\|\varphi\|^2}\right]\end{aligned}$$

Méthode de Cafforio-Rocca

Raffinement

On obtient en fin :

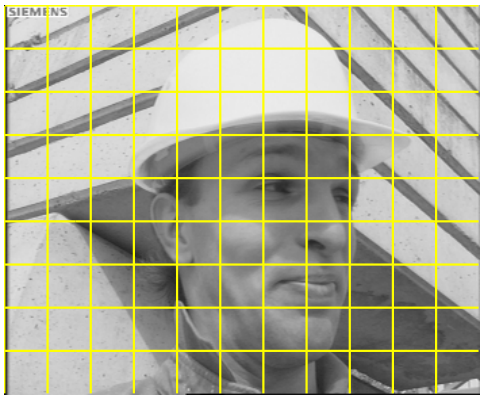
$$\delta\mathbf{D}^* = -e\frac{1}{\lambda}\left[\mathbf{I} - \frac{\varphi\varphi^T}{\lambda + \|\varphi\|^2}\right]\varphi$$

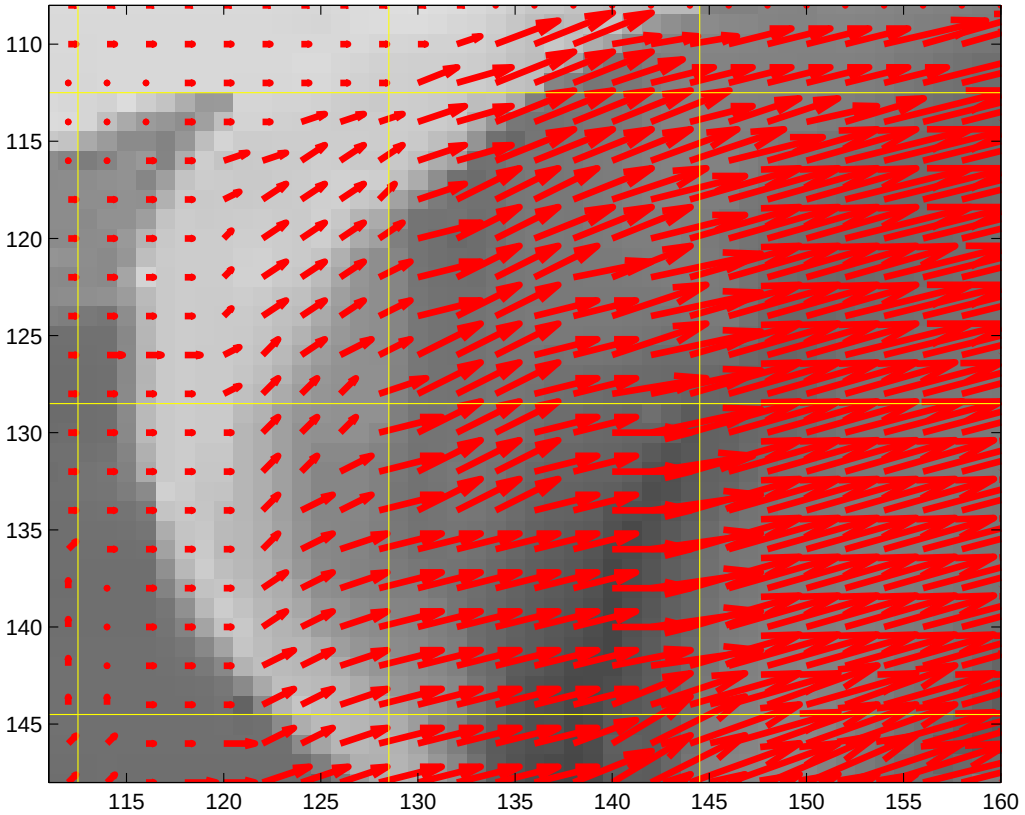
$$= -\frac{e}{\lambda}\left[\varphi - \frac{\|\varphi\|^2}{\lambda + \|\varphi\|^2}\varphi\right]$$

$$\boxed{\delta\mathbf{D}^* = \frac{-e\varphi}{\lambda + \|\varphi\|^2}}$$

Méthode de Cafforio-Rocca

Exemples





Méthode de Cafforio-Rocca

Bilan

- Algorithme récursif : amélioration progressive du résultat
- Description précise du mouvement (champ dense)
- La validation permet de mieux gérer les chargements soudains de mouvement (contours des objets)
- Reste une méthode complexe
- Convergence ?

Block matching

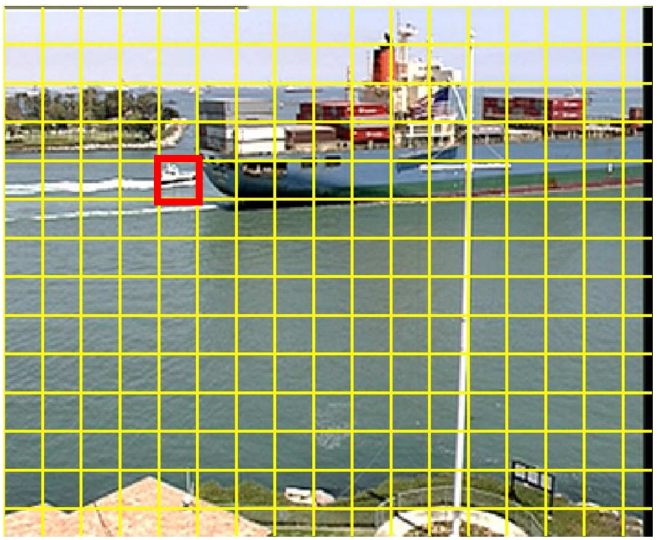
- La région de support est un bloc de pixel
- Le modèle de mouvement est de translation
  - De conséquence, on a un vecteur de mouvement par bloc
  - Des modèles affines, qui permettent de décrire rotations, zoom, . . . ont été parfois proposés avec moins de succès
- Méthodes très populaires en compression vidéo
- Compromis entre capacité de décrire le mouvement et coût de codage

Block matching

Notation

- On considère une image de taille $N \times M$
- Un bloc $B_{p,q}$ est défini par un ensemble d'indices qui démarrent en (p,q), de taille $P \times Q$

$$B_{p,q} = \{p, p+1, \ldots, p+P-1\} \times \{q, q+1, \ldots, q+Q-1\}$$



Block matching

Notation

- On introduit le symbole $\mathbf{f}_k(B_{p,q})$ pour indiquer le vecteur des PQ valeur de luminance du bloc $B_{p,q}$:

$$\begin{aligned}\mathbf{f}_k(B_{p,q}) = [&f(p,q,k), f(p+1,q,k), \ldots \\ &f(p+P-1,q), f(p,q+1,k), \ldots, \\ &f(p+P-1,q+Q-1,k)]^T\end{aligned}$$

- Le bloc matching est effectué en calculant une mesure de dissimilarité entre $\mathbf{f}_k(B_{p,q})$ et $\mathbf{f}_h(B_{p-i,q-j})$
- Le vecteur de mouvement qui minimise cette dissimilarité est retenu

Block matching

Notation

L'estimation de mouvement par block matching (BM) revient donc à résoudre, pour tout block $B_{p,q}$ de l'image courante k le problème :

$$\begin{aligned}\left(\hat{i},\hat{j}\right) &= \arg \min_{(i,j)\in\mathcal{W}} d\left[\mathbf{f}_k(B_{p,q}), \mathbf{f}_h(B_{p-i,q-j})\right] \\ &= \arg \min_{(i,j)\in\mathcal{W}} J(i,j)\end{aligned}$$

Les techniques de BM diffèrent en :

- Le critère à minimiser, c'est-à-dire la fonction d (ou J)
- L'ensemble $\mathcal{W}$ des vecteurs candidats et la stratégie pour parcourir cet ensemble
- La taille des blocs

Critères de BM

Critères basés sur la norme

- On doit choisir une mesure de dissimilarité entre deux vecteurs $\mathbf{f}_k(B_{p,q})$ et $\mathbf{f}_h(B_{p-i,q-j})$
- L'approche plus naturelle est la distance, c'est-à-dire la norme de la différence :

$$J(i,j) = \left\| \mathbf{f}_k(B_{p,q}) - \mathbf{f}_h(B_{p-i,q-j}) \right\|_p^p \qquad (2)$$

- Cas d'intérêt : $p = 1$, $p = 2$

Critères de BM

Critères basés sur la norme : SSD

- Si dans le critère en Eq. (2) on choisit $p = 2$ on a le critère dit *Sum of Squared Differences* ou SSD :

$$J_{\mathrm{SSD}}(i,j) = \sum_{(n,m \in B_{p,q})} [f(n,m,k) - f(n-i,m-j,h)]^2$$

- On définit l'image compensée comme celle obtenue en remplaçant $\mathbf{f}_k(B_{p,q})$ avec $\mathbf{f}_h(B_{p-i,q-j})$
- Pour une taille de bloc donnée la SSD minimise l'énergie de l'erreur de compensation de mouvement (ou résidu)
- Cela réduit le coût de codage du résidu

Critères de BM

Critères basés sur la norme : SSD

Problèmes de la SSD :

- Elle est relativement complexe : PQ multiplication
- Le carré exacerbe les erreurs (bruit)
- Ne prend pas en compte les changement d'illumination globale
- La norme $p = 1$ mitige l'impact des 2 premiers problèmes

Critères de BM

Critères basés sur la norme : SAD

- Si dans le critère en Eq. (2) on choisit $p = 1$ on a le critère dit *Sum of Absolute Differences* ou SAD :

$$J_{\mathrm{SAD}}(i,j) = \sum_{(n,m \in B_{p,q})} |f(n,m,k) - f(n-i,m-j,h)|$$

- L'erreur de prédiction a nécessairement une énergie non inférieure à celle du cas SSD
- Le champs de vecteur est généralement plus régulier

Critères de BM

Critères basés sur la norme : exemples

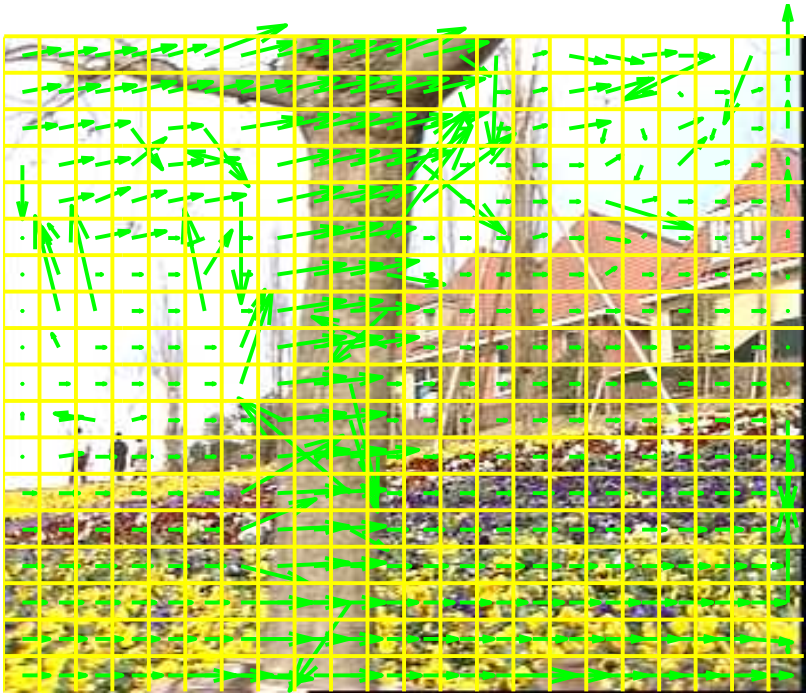
Reference image

Current image

Critères de BM

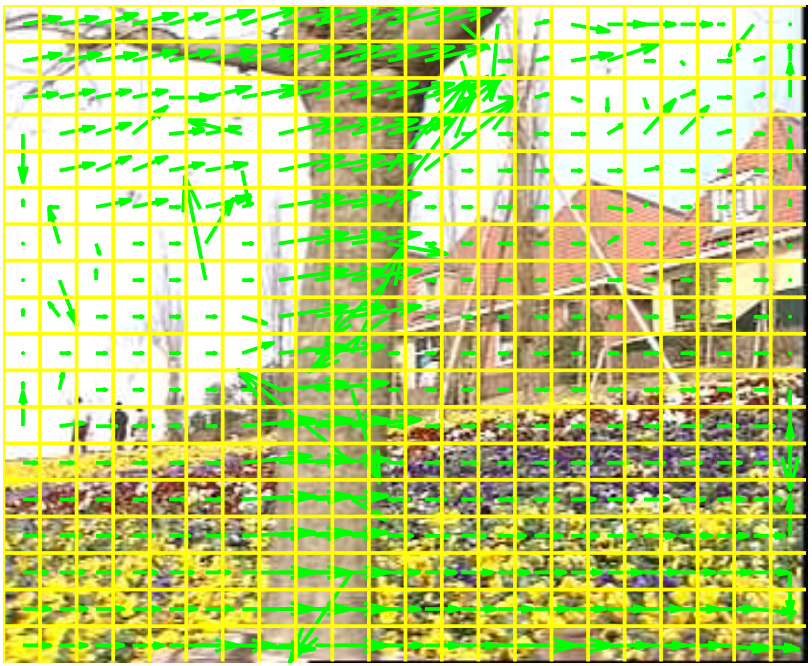
Critères basés sur la norme : exemples

SSD Rate(MV): 2143 bits − PSNR(Pred): 22.46 dB



SSD

SAD Rate(MV): 2103 bits − PSNR(Pred): 22.30 dB



SAD

Critères de BM

Critères basés sur la norme : exemples

Motion−compesated image

SSD

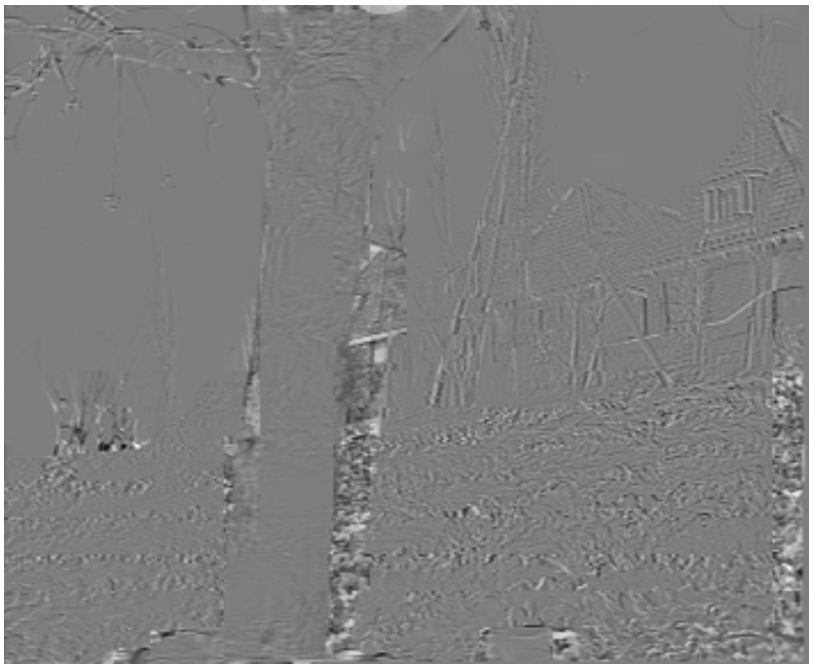
Motion−compesated image

SAD

Critères de BM

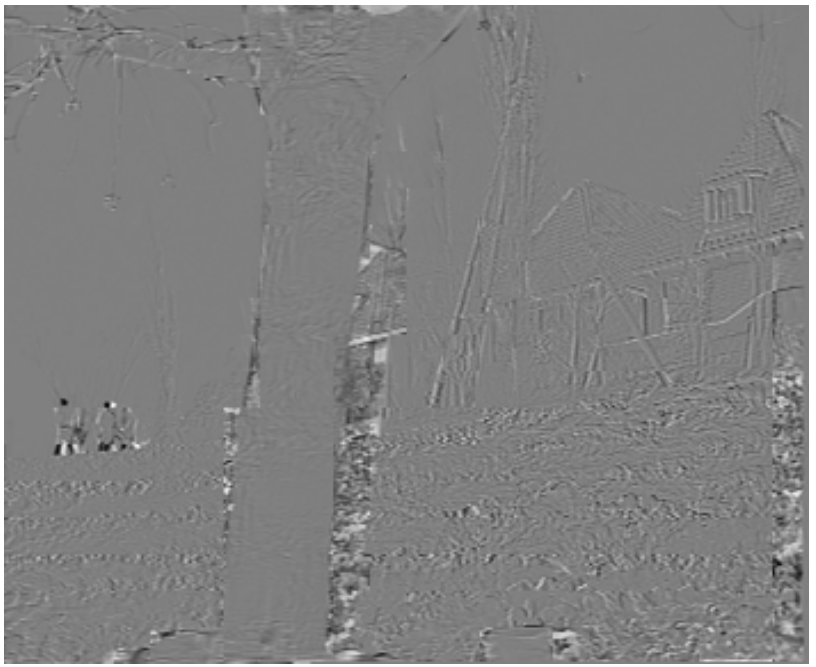
Critères basés sur la norme : exemples

MC error



SSD

MC error

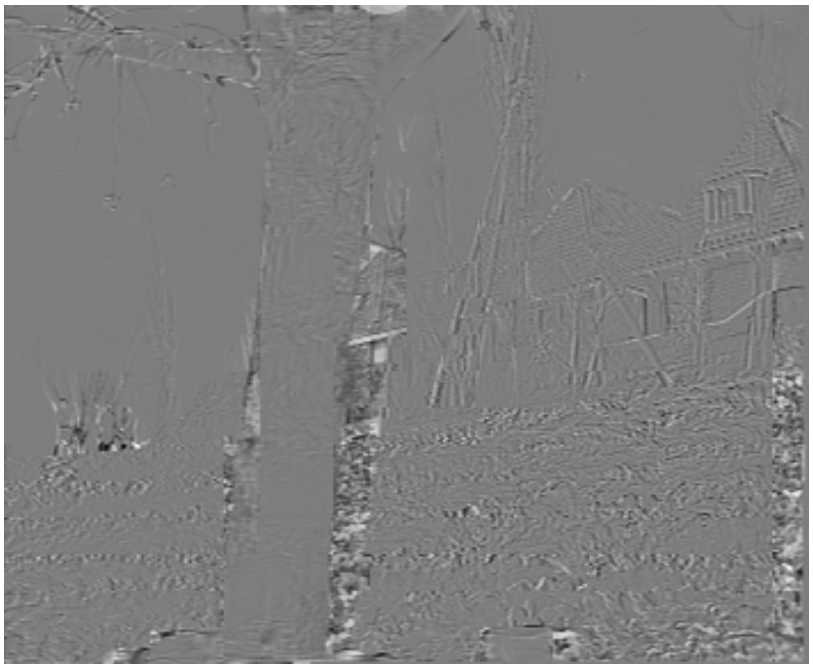


SAD

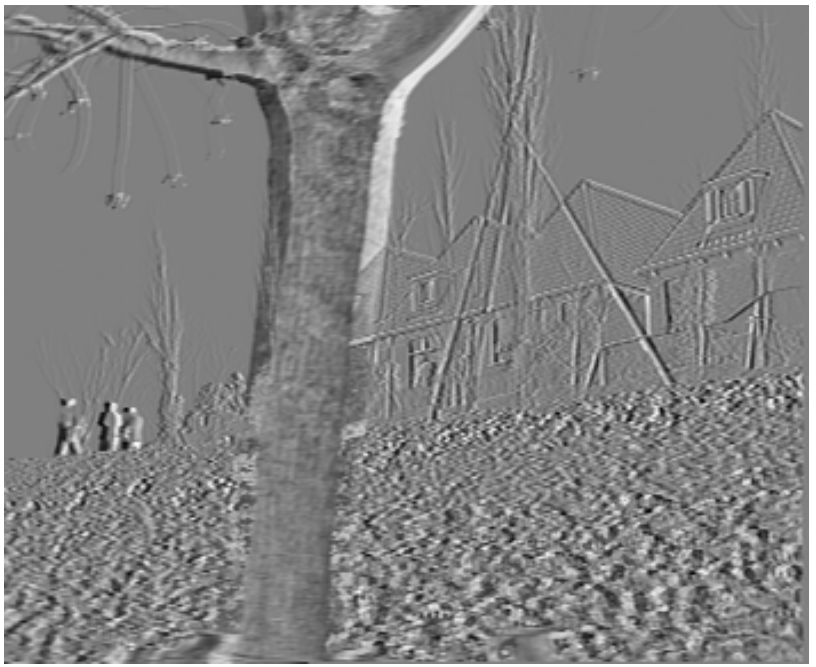
Critères de BM

Critères basés sur la norme : exemples

MC error



Difference image



Critères de BM

Critères basés sur la norme

- Les champs de vecteurs montrés ne sont pas toujours réguliers
- On a imposé aucune forme de régularité
- Avantage : minimisation de l'énergie de l'erreur
- Inconvénients : Pas le "vrai" mouvement sur régions homogènes ; coût de codage élevé
- Solution : régularisation

Critères de BM

Critères basés sur la norme régularisée

- On modifie le critère (2) en pénalisant les vecteurs trop différents du voisinage
- Cela est pris en compte par une fonction $R(i,j)$
- On obtient donc :

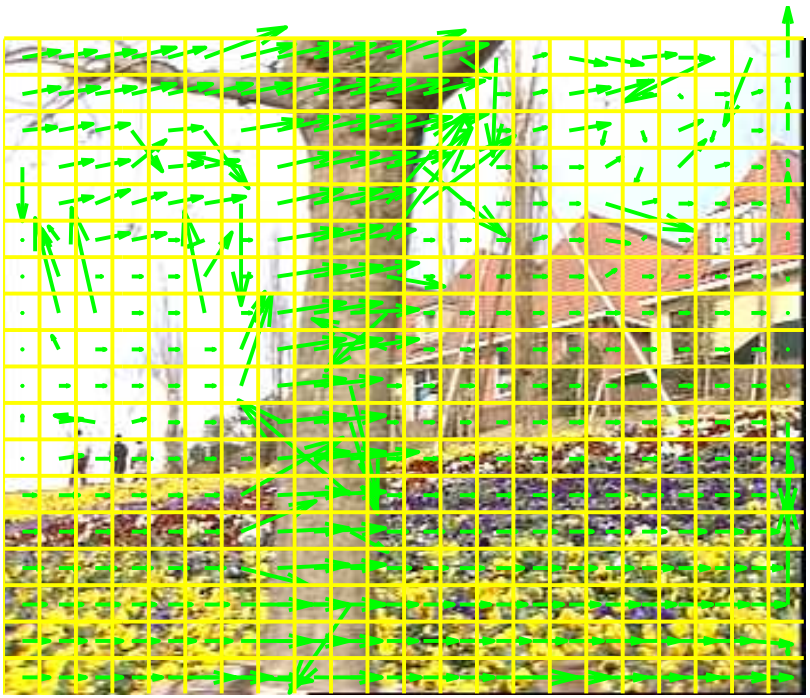
$$J_{\mathrm{REG}}(i,j) = \left\| \mathbf{f}_k(B_{p,q}) - \mathbf{f}_h(B_{p-i,q-j}) \right\|_p^p + \lambda R(i,j)$$

- Par exemple, R peut être la distance entre (i,j) et un représentant du voisinage, comme la moyenne des voisins

Critères de BM

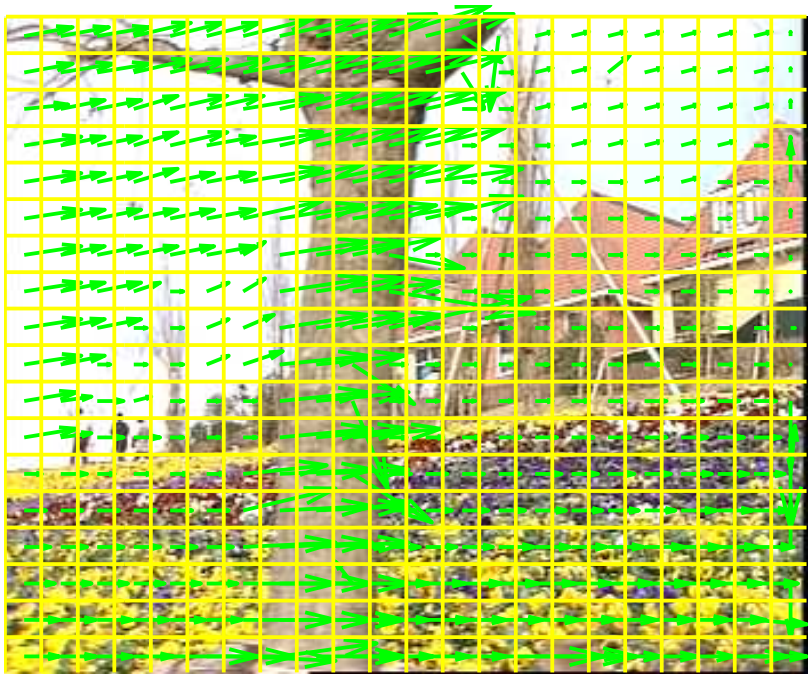
Critères basés sur la norme : exemples

SSD Rate(MV): 2143 bits − PSNR(Pred): 22.46 dB



SSD

SSD Rate(MV): 2008 bits − PSNR(Pred): 22.35 dB



SSD régularisé

Critères de BM

Critères basés sur la corrélation

- La corrélation est une mesure de similarité entre deux vecteurs
- Elle peut être calculée avec des techniques très rapides basées sur la FFT
- On peut donc penser de l'utiliser pour le BM

Critères de BM

Critères basés sur la corrélation

Le critère SSD peut s'écrire comme il suit :

$$\begin{aligned} J_{\mathrm{SSD}}(i,j) &= \left\|\mathbf{f}_k(B_{p,q}) - \mathbf{f}_h(B_{p-i,q-j})\right\|_2^2 \\ &= \left\langle \mathbf{f}_k(B_{p,q}) - \mathbf{f}_h(B_{p-i,q-j}), \mathbf{f}_k(B_{p,q}) - \mathbf{f}_h(B_{p-i,q-j}) \right\rangle \\ &= \left\|\mathbf{f}_k(B_{p,q})\right\|_2^2 - 2\left\langle \mathbf{f}_k(B_{p,q}), \mathbf{f}_h(B_{p-i,q-j}) \right\rangle + \left\|\mathbf{f}_h(B_{p-i,q-j})\right\|_2^2 \end{aligned}$$

Dans cette expression, $\left\|\mathbf{f}_k(B_{p,q})\right\|_2^2$ ne dépend pas de (i,j). Si $\left\|\mathbf{f}_h(B_{p-i,q-j})\right\|_2^2$ ne varie pas avec (i,j), minimiser la SSD est équivalent à maximiser la corrélation :

$$J_{\mathrm{CORR}}(i,j) = \left\langle \mathbf{f}_k(B_{p,q}), \mathbf{f}_h(B_{p-i,q-j}) \right\rangle$$

Critères de BM

Critères basés sur la corrélation

- Avantage : implémentation par FFT
- Inconvénients :
  - Dans les images naturelles, l'énergie varie beaucoup : $\|\mathbf{f}_h(B_{p-i,q-j})\|_2^2$ ne peut pas être constant
  - Par conséquent, la corrélation entre le bloc original et une région très lumineuse de l'image peut être supérieure à la corrélation avec le bloc qui correspond au mouvement
  - La corrélation est sensible à l'illumination globale
- Solution : corrélation normalisée

Critères de BM

Critères basés sur la corrélation

Avec la corrélation normalisée on réduit la dépendance de la luminosité globale et locale.
On introduit la version normalisé du bloc :

$$\tilde{\mathbf{f}}_k(B_{p,q})[\ell] = \mathbf{f}_k(B_{p,q})[\ell] - \frac{1}{PQ}\sum_{h=1}^{PQ}\mathbf{f}_k(B_{p,q})[h]$$

La corrélation normalisée est donc :

$$J_{\mathrm{N-CORR}} = \frac{\sum_{\ell=1}^{PQ}\tilde{\mathbf{f}}_k(B_{p,q})[\ell]\tilde{\mathbf{f}}_h(B_{p-i,q-j})[\ell]}{\left(\sum_{\ell=1}^{PQ}\tilde{\mathbf{f}}_k^2(B_{p,q})[\ell]\cdot\sum_{\ell=1}^{PQ}\tilde{\mathbf{f}}_h^2(B_{p-i,q-j})[\ell]\right)^{1/2}}$$

Ce critère est souvent utilisé pour des problèmes de *feature tracking*. On peut l'implémenter aussi avec des algorithmes de FFT.

Block matching : stratégies de recherche

Full search

- Solution naïve : tout vecteur (i,j) doit être testé
  - On doit calculer J un nombre de fois égal à $(N-P)(M-Q)$
  - Souvent il suffit de considérer un plus petite région centré en (p,q)
- On considère une *fenêtre de recherche* $\mathcal{W}$:

$$\mathcal{W} = \{-A,\ldots,-1,0,+1,\ldots,A\} \times \{-B,\ldots,-1,0,+1,\ldots,B\}$$

- Souvent $A = B$. Si $n = 2A+1$, il y a n^2 candidat dans $\mathcal{W}$
- Si on calcule J pour tout vecteur en $\mathcal{W}$, on parle de *full search*

Block matching : stratégies de recherche

Méthodes rapides

- Avec le full search on doit effectuer n^2 calculs de J pour trouver le meilleur vecteur qui correspond à un déplacement de $\pm A$ pixels (en horizontale et en verticale) au plus
- Les méthodes rapides permettent de estimer des mouvement de $\pm A$ pixels avec moins de n^2 calculs
- L'idée de base est de tester seulement un sous-ensemble des vecteurs
- En revanche, on n'est pas sûr de trouver le meilleur vecteur possible

Block matching : stratégies de recherche

Méthodes rapides

Algorithme générale

1. Initialisation : $k = 0$, on choisit $\mathcal{W}_1$ et (i_0, j_0)
2. Jusqu'à quand la condition d'arrêt n'est pas vérifiée, répéter :
   - 2.1 $k = k + 1$
   - 2.2 $(i_k, j_k) = \arg\min_{(i,j)\in\mathcal{W}_k} J(i,j)$
   - 2.3 $\mathcal{W}_{k+1} = \mathcal{F}(\Delta\mathcal{W}_k, (i_k, j_k))$
3. $(\hat{i}, \hat{j}) = (i_k, j_k)$

Block matching : stratégies de recherche

Méthodes rapides

- À chaque pas on fait une full search sur $\mathcal{W}_k$
- En suite, la fenêtre est mise à jour en fonction du vecteur courant et d'un schéma de recherche $\Delta\mathcal{W}_k$ qui peut varier à chaque pas
- La condition d'arrêt peut être sur le nombre d'itération, sur les caractéristique du vecteur trouvé où sur la valeur du critère
- Le critère est donc calculé un nombre de fois égal à $\sum_{k=1}^{K} \mathrm{card}(\mathcal{W}_k)$ pour K itérations

Block matching : stratégies de recherche

2D-log search

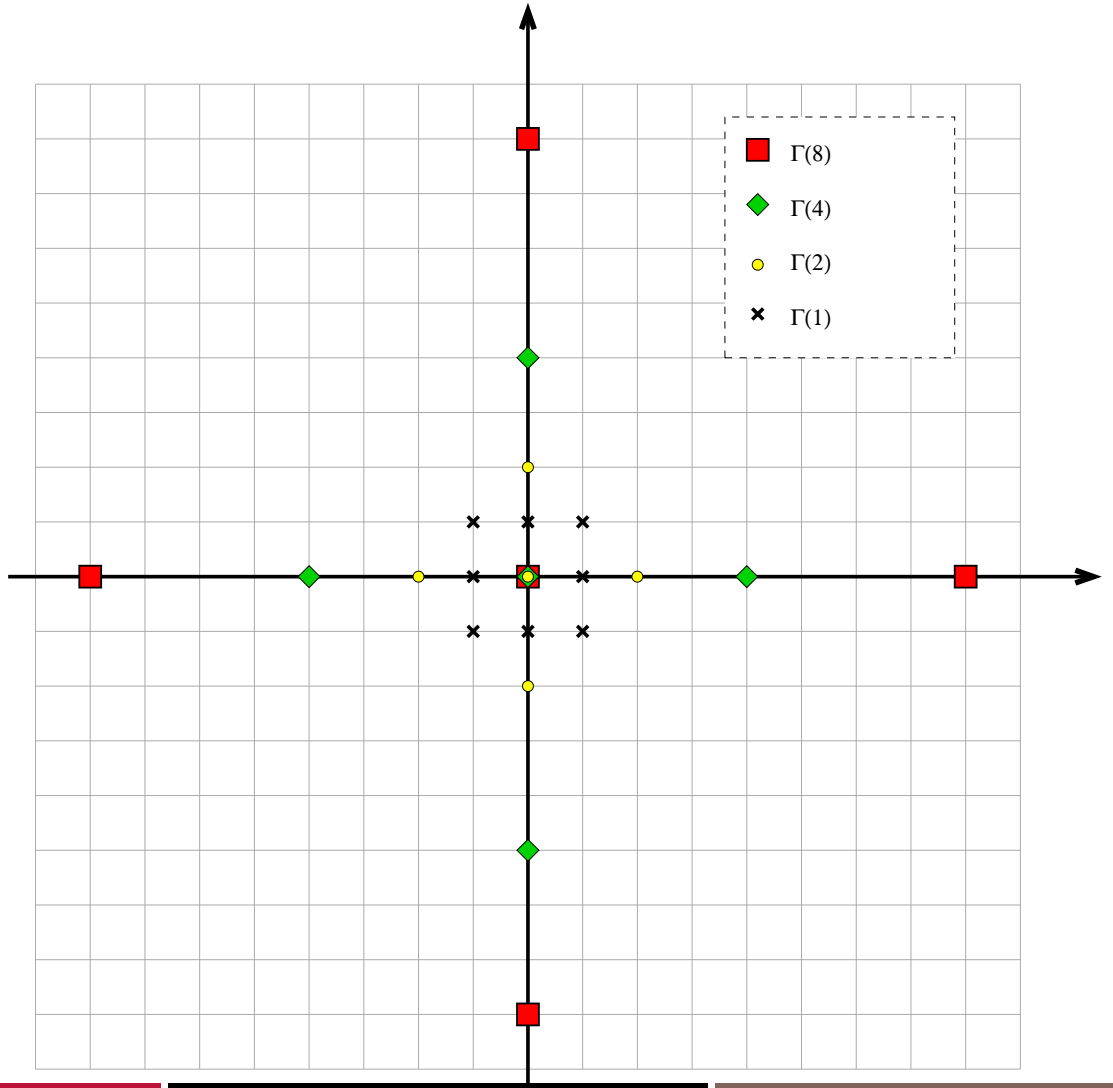
- On définit les ensembles suivants :

$$\Gamma(1) = \{-1, 0, 1\}^2$$
$$\Gamma(2^n) = \{(0,0), (\pm 2^n, 0), (0, \pm 2^n)\}$$

- On choisit $\mathcal{W}_1 = \Gamma(2^m)$ avec $m = 3$ ou 4 ; Pour $k = 1$, $r_k = 2^m$ est le rayon de recherche
- On trouve le meilleur candidat sur $\mathcal{W}_k$: si c'est $(0,0)$, on choisit $r_{k+1} = r_k/2$; sinon $r_{k+1} = r_k$
- $\mathcal{W}_{k+1} = \{(i_k, j_k) + (i,j) | (i,j) \in \Gamma(r_k)\}$
- Si $r_k = 1$ on arrête la recherche

Block matching : stratégies de recherche

2D-log search



Block matching : stratégies de recherche

Cross search

- La recherche se fait d'abord en horizontale et en suite en verticale
- On commence avec $(i_0, j_0) = (0, 0)$
- On a seulement 2 itérations :

$$\Delta\mathcal{W}_1 = \{-A, -A+1, \ldots, A\} \times \{0\}$$
$$\Delta\mathcal{W}_2 = \{0\} \times \{-A, -A+1, \ldots, A\}$$

- La fenêtre de recherche est modifiée comme il suit :

$$\mathcal{W}_{k+1} = \{(i_k, j_k) + (i, j) | (i, j) \in \Delta\mathcal{W}_k\}$$

- Nombre de calculs fixe : $2n - 1$ contre n^2 du full search pour le même A

Block matching : stratégies de recherche

Three steps search (3SS)

- La fenêtre de recherche est encore modifiée avec :

$$\mathcal{W}_{k+1} = \{(i_k, j_k) + (i,j) | (i,j) \in \Delta\mathcal{W}_k\}$$

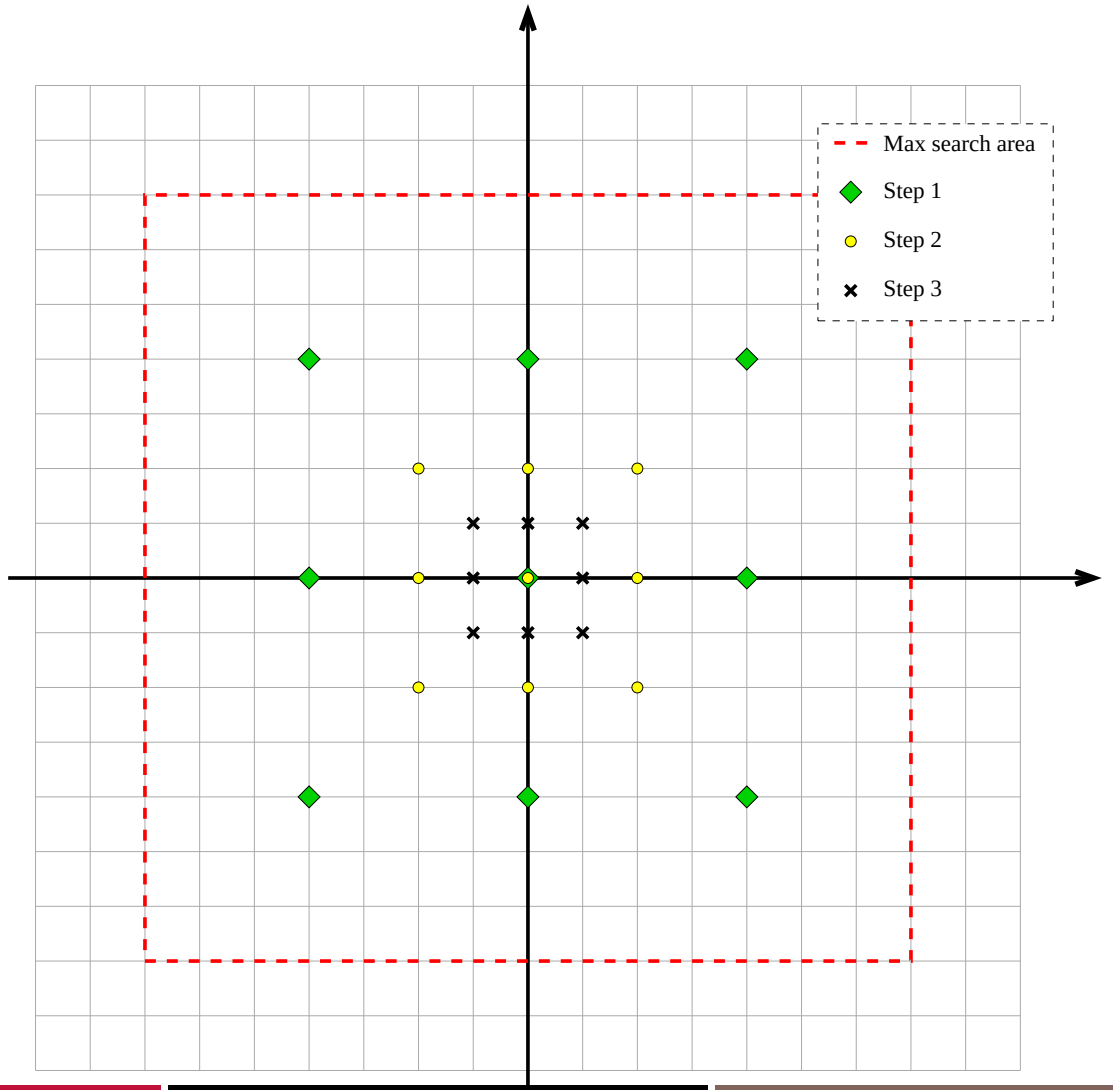
- Mais on a

$$\Delta\mathcal{W}_k = \{-2^k, 0, 2^k\}^2$$

- Le rayon de recherche est réduit de moitié à chaque itération
- On commence avec $\Delta\mathcal{W}_1 = \{4, 0, 4\}^2$ et on fait 3 itérations
- Complexité : 25 calculs (contre 225 du full-search, car $n = 15$)

Block matching : stratégies de recherche

Three steps search (3SS)



Block matching : stratégies de recherche

New 3SS

- 3SS est efficace mais on commence avec des vecteurs grands
- On est obligé à faire trois itérations même si souvent les vecteurs de mouvement sont petits
- N3SS : on ajoute des vecteurs petit à la première itération et une condition d'arrêt anticipé sur la valeur du critère

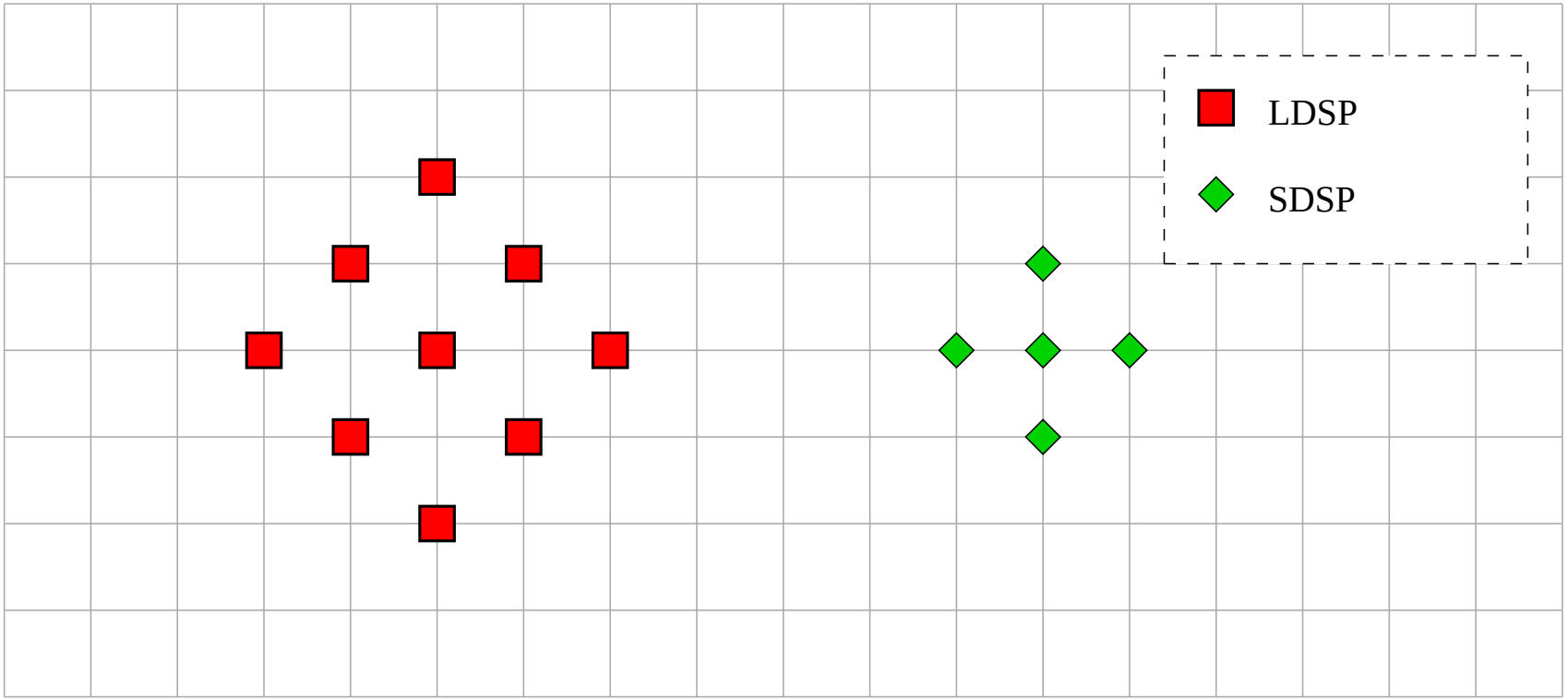
Block matching : stratégies de recherche

Diamond search

- On découple la taille du pattern de recherche du nombre d'itérations
- On permet d'estimer des vecteurs dont les composants vont au delà de ± 7
- On utilise un *Large Diamond Search Pattern* et un *Small Diamond Search Pattern*
- On commence avec le LDSP, et on passe au SDSP seulement si on choisit le vecteur au centre de $\mathcal{W}_k$
- Après une itération avec le SDPS l'algorithme s'arrête

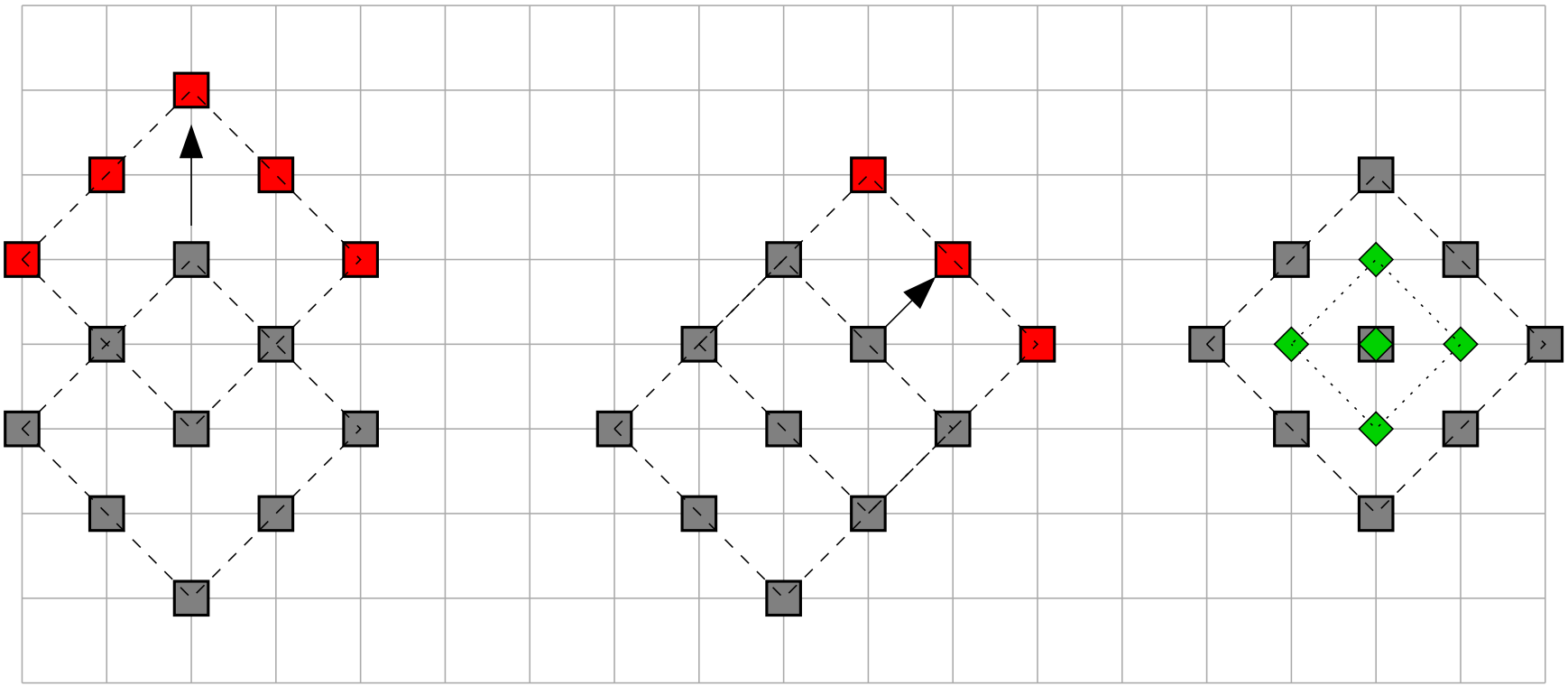
Block matching : stratégies de recherche

Diamond search



Block matching : stratégies de recherche

Diamond search



Block matching : stratégies de recherche

Diamond search

- Le pattern n'est pas trop petit (risque de minimum local) ni trop grand comme dans 3SS (risque de partir dans une mauvaise direction
- Très bonnes performances : beaucoup meilleur de 3SS
- Qualité similaire à N3SS avec 20% de réduction de complexité
- Intégré en MPEG-4 ASP

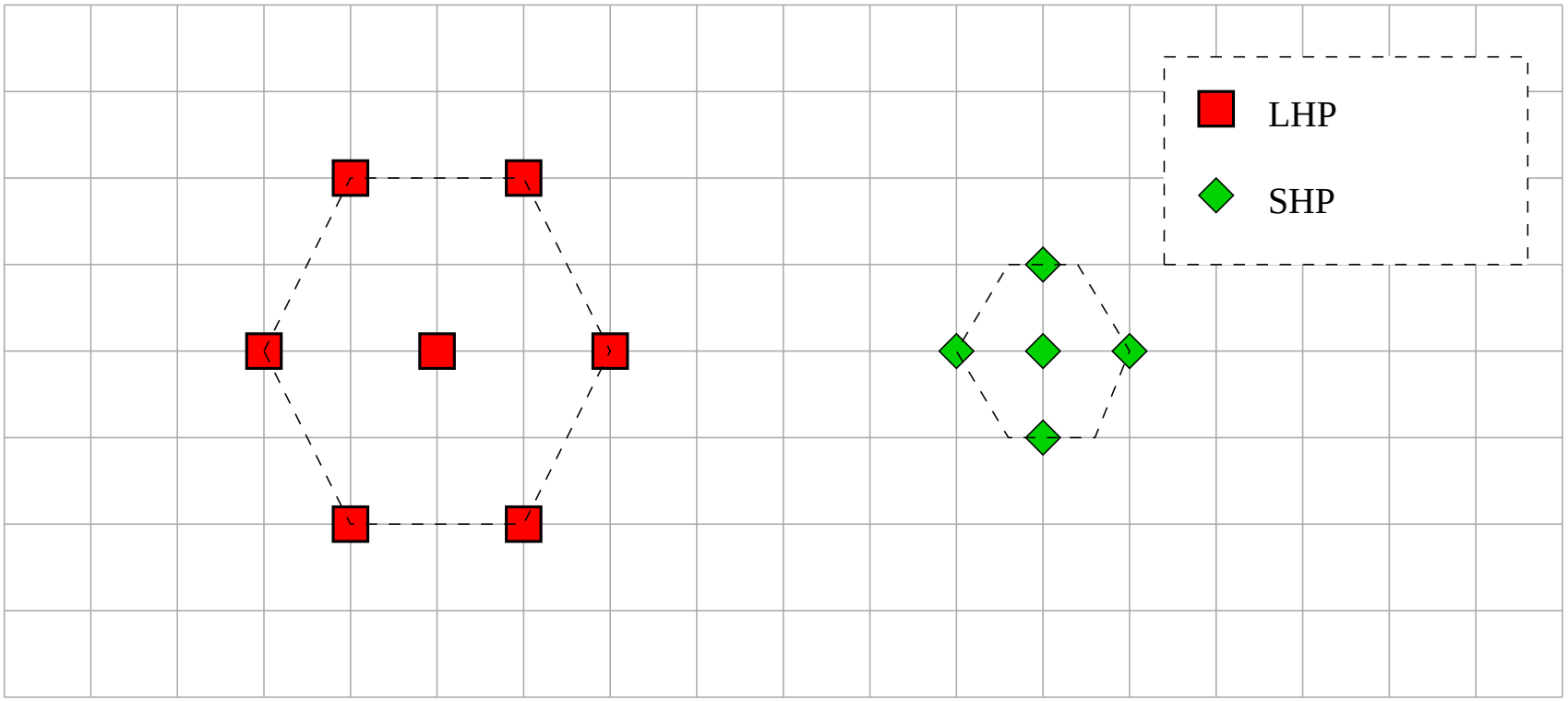
Block matching : stratégies de recherche

Hexagon search

- DS : selon la direction du déplacement on doit faire 5 ou 3 calculs par itération
- Vitesse de déplacement : 2 pixel en H/V, seulement $\sqrt{2}$ en diagonal
- Avec un pattern de forme plus régulière, on peut avoir une vitesse plus uniforme est moins de calculs

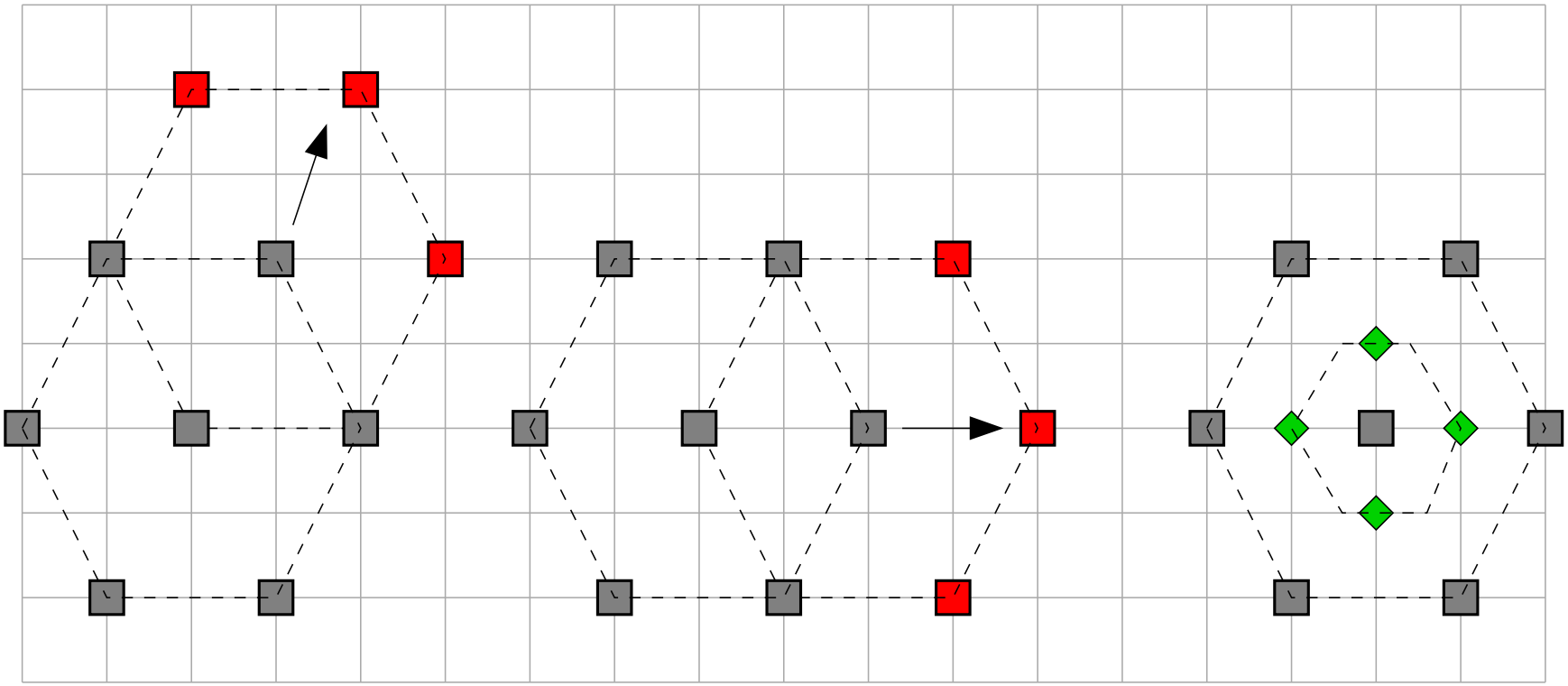
Block matching : stratégies de recherche

Hexagon search



Block matching : stratégies de recherche

Hexagon search

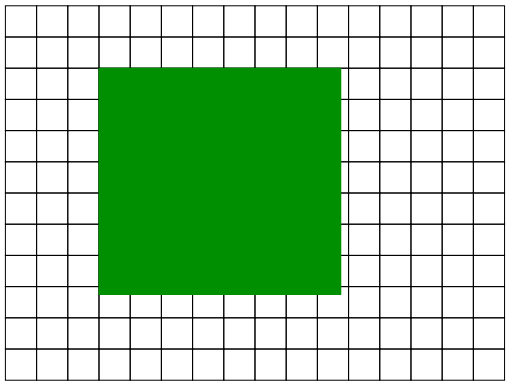


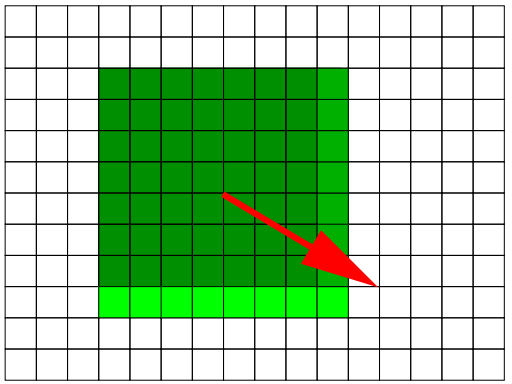
Block matching : stratégies de recherche

Hexagon search

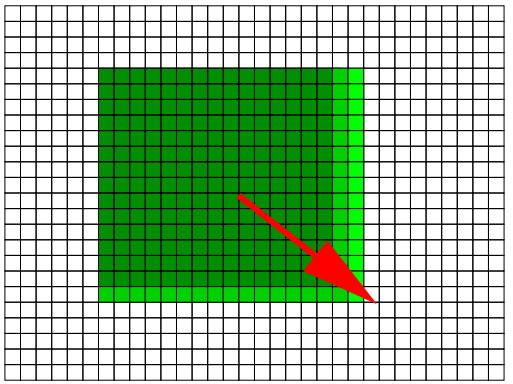
- Vitesse : 2 pixel en H, $\sqrt{5} \approx 2.2$ en diagonale
- Nombre de calculs indépendant de l'itération
- Réduction de la complexité du 40% par rapport à DS pour la même qualité d'estimation
- Intégré dans H.264

Block matching : précision subpixelique

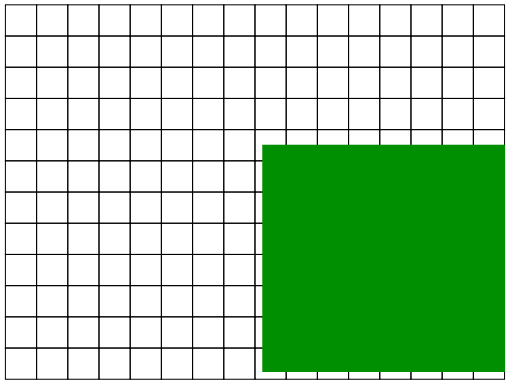


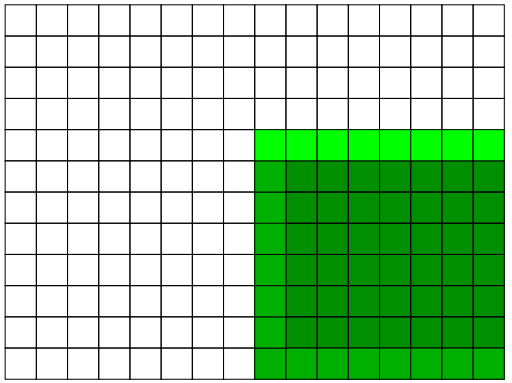


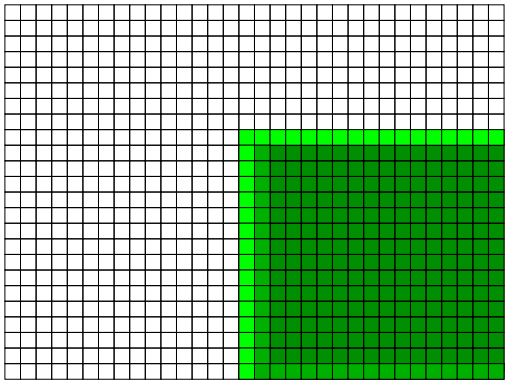
v=(5,3)



v=(5.5,2.5)



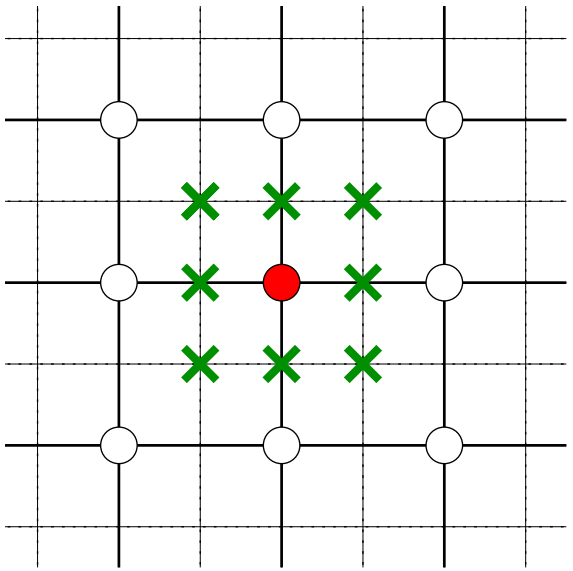




Block matching : précision subpixelique

Approche hiérarchique :

- d'abord on trouve $(\hat{i},\hat{j}) \in \mathbb{Z}^2$;
- en suite on teste $(\hat{i} \pm \frac{1}{2},\hat{j} \pm \frac{1}{2})$;
- en suite on vérifie les voisins à distance $1/4$ de pixel, etc.



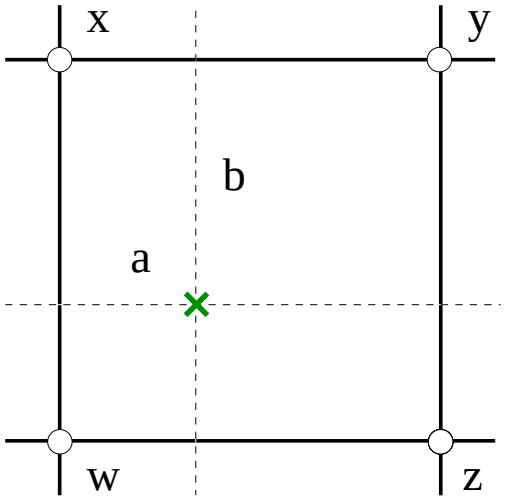
Positions pixel entier

Vecteur pixel entier sélectionné

Positions demi-pixel testées

Block matching : précision subpixelique

Pour évaluer $f(n \pm \alpha, m \pm \beta, k)$ on fait une interpolation



$$f(n,m) = x \quad f(n+1,m) = y$$

$$f(n,m+1) = z \quad f(n+1,m+1) = w$$

Interpolation bi-linéaire : moyenne en horizontale, puis en verticale

$$f(n+a,m+b) = (1-a)(1-b)x + a(1-b)y + (1-a)bz + abw$$

Autres techniques d'interpolation avec des filtres d'ordre supérieure

Block matching : bloc de taille variable

- Hypothèse sous-jacente au BM: mouvement homogène dans un bloc
- Ce n'est pas vrai si les contours d'un objet sont dans le bloc
- Solution 1 : blocs plus petit
  - Meilleure précision
  - Complexité augmentée
  - Coût de codage (augmentation du nombre de vecteur)
- Solution 2 : blocs de taille variable
  - Coût de codage et complexité additionnelle seulement si nécessaire
  - Approche hiérarchique : un bloc est coupé si la valeur du critère est trop grande
  - Solution adoptée dans les normes récentes

Estimation du mouvement paramétrique

- Au lieu de chercher le mouvement bloc par bloc, ou pixel par pixel, on peut fixer un modèle globale, et en estimer seulement les paramètres
- On parle d'estimation de mouvement globale
- Souvent il s'agit d'estimer les mouvements dus à la caméra
  - Mouvement le long des 3 axes
  - Rotations autour des 3 axes
- Modèles de translation ou affines

Estimation indirecte

- Une solution est d'estimer d'abord un champs dense, et ensuite en déduire les paramètres
- L'approche est celui des moindres carrés
- Soit $u(n,m)$ et $v(n,m)$ les composantes du champ dense estimé
- Soit $u_\pi(n,m)$ et $v_\pi(n,m)$ des champs denses qui dépendent de l'ensemble des paramètres π
- L'estimation indirecte reviens à :

$$\pi^* = \arg\min_{\pi} \sum_{n,m\in\mathcal{R}} [u(n,m) - u_\pi(n,m)]^2 + [v(n,m) - v_\pi(n,m)]^2$$

Estimation indirecte

- Le problème peut être résolu avec l'algorithme du gradient
- Inconvénients :
  - Sensibilité par rapport à l'estimation initiale
  - Nécessité de trouver une région de support $\mathcal{R}$ où le mouvement soit homogène

Estimation directe

- On introduit le modèle paramétrique au moment de l'estimation
- Par exemple, l'équation du flux optique devient :

$$\pi^* = \arg\min_{\pi} \sum_{n,m\in\mathcal{R}} [u_\pi(n,m)f_x(n,m) + v_\pi(n,m)f_y(n,m) + f_t(n,m)]^2$$

- Un autre approche est de minimiser la SSD ou la SAD correspondent au champ paramétrique :

$$\pi^* = \arg\min_{\pi} \sum_{n,m\in\mathcal{R}} [e(n,m)]^2 \qquad \text{avec}$$

$$e(n,m) = f(n - u_\pi(n,m), m - v_\pi(n,m), t-1) - f(n,m,t)$$

Estimation du mouvement globale

Estimation fréquentielle de la translation

- On suppose que $f_k(\mathbf{p}) = f_h(\mathbf{p} + \mathbf{d})$
- Soit $\hat{f}_k(\mathbf{w}) = \mathcal{F}[f_k(\mathbf{p}), \mathbf{p} \to \mathbf{w}]$ la TFD 2D de f_k
- On trouve :

$$\begin{aligned}\hat{f}_h(\mathbf{w}) = \mathcal{F}[f_h(\mathbf{p}), \mathbf{p} \to \mathbf{w}] &= \mathcal{F}[f_k(\mathbf{p} + \mathbf{d}), \mathbf{p} \to \mathbf{w}] \\ &= \hat{f}_k(\mathbf{w}) e^{-j2\pi \mathbf{d}^T \mathbf{w}} \\ \arg\{\mathcal{F}[f_k(\mathbf{p})]/\mathcal{F}[f_h(\mathbf{p})]\} &= -2\pi \mathbf{d}^T \mathbf{w}\end{aligned}$$

- Donc l'argument dépend linéairement du mouvement global $\mathbf{d}$, qui peut être estimé p.e. par moindre carrés

Plan

Estimation du mouvement : applications

Compression

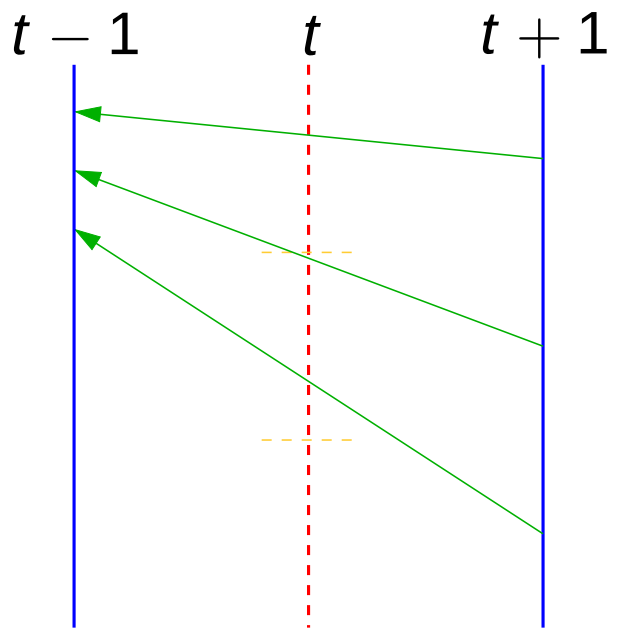
- L'estimation de mouvement permet de produire une prédiction de l'image courante
- Au lieu d'envoyer l'image courante, on code l'erreur de prédiction
- Si la prédiction est bonne (c'est-à-dire, erreur de faible énergie), coder l'erreur coûte moins cher que coder l'image (pour la même qualité)

Estimation du mouvement : applications

Frame-rate up conversion

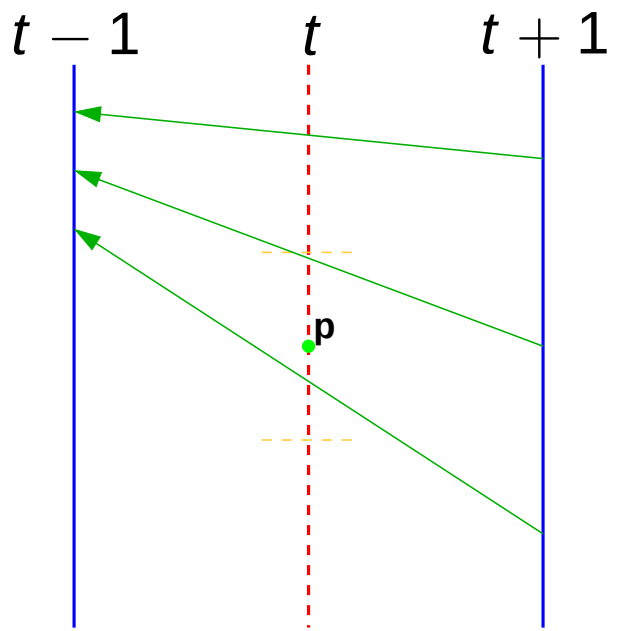
- Augmenter la cadence d'images par seconde
- On insère des images crées avec compensation du mouvement
- Typiquement on utilise une MC bi-directionnelle

Algorithme d'interpolation de mouvement DISCOVER



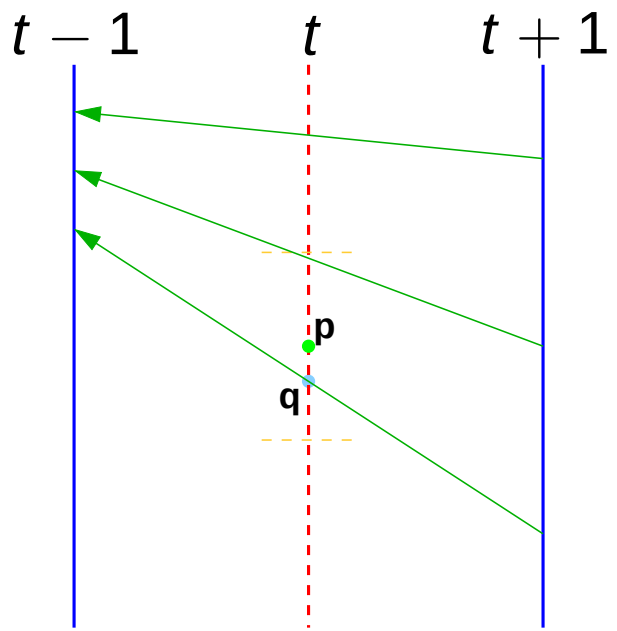
On effectue un block matching entre les images $t+1$ et $t-1$.

Algorithme d'interpolation de mouvement DISCOVER



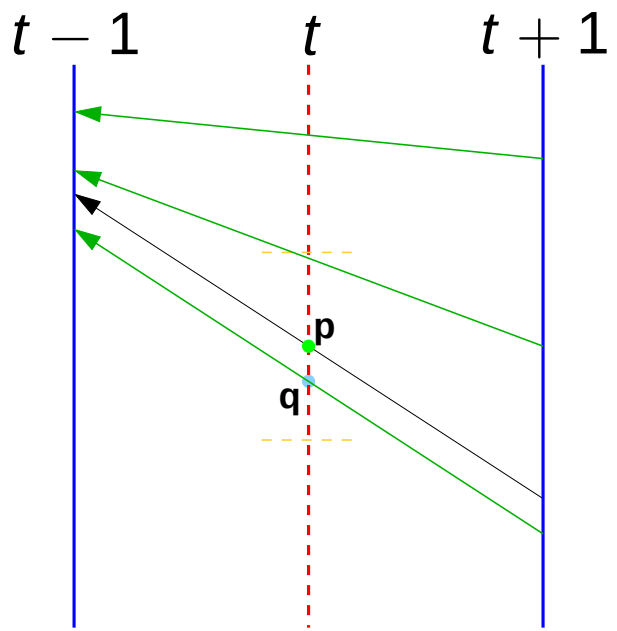
Le bloc centré en **p** doit être estimé.

Algorithme d'interpolation de mouvement DISCOVER



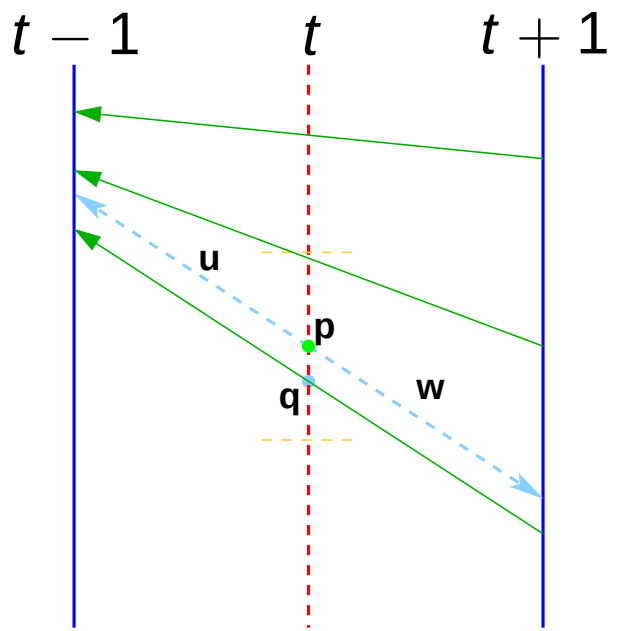
On cherche la trajectoire qui passe le plus proche du centre du bloc dans l'image à interpoler. Soit **q** ce point.

Algorithme d'interpolation de mouvement DISCOVER



On assume que le mouvement en **p** soit le même qu'un **q**.

Algorithme d'interpolation de mouvement DISCOVER



On coupe ce vecteur en **u** et **w**.

Estimation du mouvement : applications

- Dé-bruitage
  - Le débruitage se fait normalement avec un filtre passe-bas
  - En filtrant le long du mouvement on évite d'introduire trop de flou
- Segmentation
  - Le mouvement est cohérent à l'intérieur d'un objet
  - L'estimation du mouvement peut donc aider à trouver les objets dans une vidéo
- Surveillance
  - Détection de trajectoire