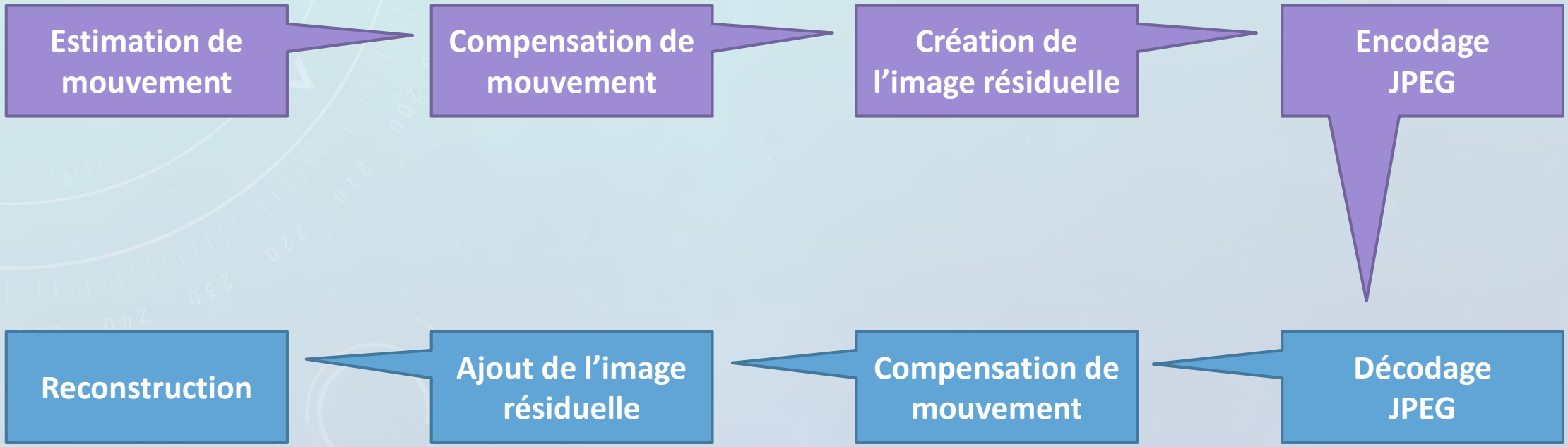
Compression, Streaming, Interaction Vidéo

Cours n°3 : Codage MPEG-2



Cours n°3 : Codage MPEG-2

Introduction

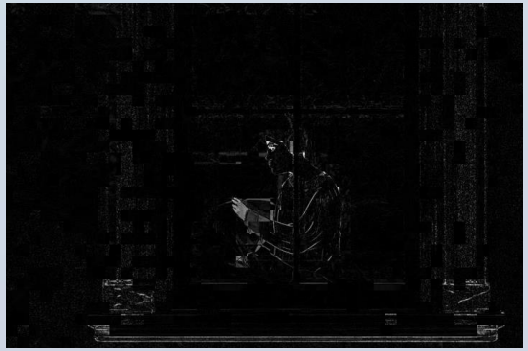
I. Compensation du mouvement

II. Estimation du mouvement
    1. Estimation basée pixels localement
    2. Estimation basée pixels dans un voisinage
    3. Estimation basée patchs
    4. Recherche du patch optimal
    5. Estimation par réseau de neurones

III. Codage MPEG-2

Introduction

Comment coder les images d’une vidéo ?

Niveau 1	Niveau 2	Niveau 3
		
		
Aucun action : MJPEG	Différence	Différence compensée : MPEG

I. Compensation du mouvement

I_{Ref}

$\vec{v}$

I_{Cou}

Image prédite I_{Pre}

Image résiduelle I_{Res}

I. Compensation du mouvement

Notations & définitions

$\boldsymbol{I_{Ref}}$: Image de référence de taille $\boldsymbol{M \times N}$

$\boldsymbol{I_{Cou}}$: Image courante (sur laquelle on va travailler) de taille $\boldsymbol{M \times N}$

$\vec{\boldsymbol{v}}$: Matrice des vecteurs de déplacements de l'image de référence $\boldsymbol{I_{Ref}}$ vers l'image courante $\boldsymbol{I_{Cou}}$ de taille $\boldsymbol{M \times N \times 2}$

→ Déplacements horizontaux et verticaux

$\boldsymbol{I_{Pre}}$: Image prédite de l'image courante $\boldsymbol{I_{Cou}}$ à partir des pixels de l'image de référence $\boldsymbol{I_{Ref}}$ et des déplacements $\vec{\boldsymbol{v}}$

→ $\boldsymbol{I_{Pre}(x,y) = I_{Ref} \circ \vec{v}(x,y) = I_{Ref}(x + v_x, y + v_y)}$

$\boldsymbol{I_{Res}}$: Image résiduelle qui est la différence entre l'image prédite $\boldsymbol{I_{Pre}}$ et l'image courante $\boldsymbol{I_{Cou}}$

→ $\boldsymbol{I_{Res}(x,y) = I_{Pre}(x,y) - I_{Cou}(x,y)}$

Le but de la compression vidéo est de coder $\boldsymbol{I_{Res}}$ et $\vec{\boldsymbol{v}}$ au lieu de $\boldsymbol{I_{Cou}}$ car :

→ $\boldsymbol{I_{Res}}$ contient beaucoup de valeurs nulles

→ $\vec{\boldsymbol{v}}$ contient beaucoup de déplacements similaires

II. Estimation du mouvement

L'estimation du mouvement entre deux images successives se base sur la corrélation entre ces dernières.

Problème mal posé car :

déplacements sont effectués dans un espace 3D / image n'en est qu'une projection 2D

Quelques applications :

traitement d'image, suivi d'objet, robotique …

Différentes méthodes utilisées pour estimer le mouvement :

Méthodes basées pixels	Méthodes basées patchs	Méthodes basées apprentissage
→ Méthodes variationnelles → Minimisation du flux optique	→ Recherche de patchs similaires entre 2 images → Différents types d'erreurs à minimiser	→ Réseau type encodeur-décodeur → Couches de corrélation dans l'encodeur

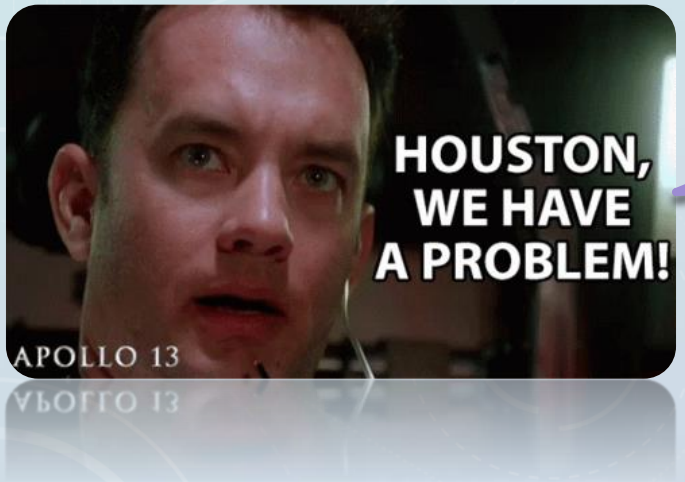
II.1. Estimation basée pixels localement

Hypothèse de conservation du flux optique : luminance constante le long des trajectoires

→ $I(x, y, t) = I(x + v_x, y + v_y, t + 1)$

But : trouver $\vec{v}$ en minimisant la différence des deux en norme

→ $\min_{\vec{v}} \left\| I\left(x + v_x, y + v_y, t + 1\right) - I(x, y, t) \right\|$



On a une équation et deux inconnues !

→ Solution : ajouter un a priori

$\|\nabla v_x\| + \|\nabla v_y\|$

C'est quelle norme ?

Minimiser en norme 2 (Horn & Schunck, 1981)

→ $\min_{\vec{v}} \frac{1}{2} \left\| I\left(x + v_x, y + v_y, t + 1\right) - I(x, y, t) \right\|_2^2 + \frac{\lambda}{2} \|\nabla v_x\|_2^2 + \frac{\lambda}{2} \left\|\nabla v_y\right\|_2^2$

Minimiser en norme 1 (Zack *et al.*, 2007)

→ $\min_{\vec{v}} \left\| I\left(x + v_x, y + v_y, t + 1\right) - I(x, y, t) \right\|_1 + \lambda \|\nabla v_x\|_1 + \lambda \left\|\nabla v_y\right\|_1$

II.1. Estimation basée pixels localement

Hypothèse supplémentaire pour pouvoir effectuer la minimisation :

→ les déplacements sont petits, ce qui permet de linéariser autour de $I(x, y, t+1)$

→ $I\big(x + v_x, y + v_y, t+1\big) = I(x, y, t+1) + \frac{\partial}{\partial x} I(x, y, t+1)\, v_x + \frac{\partial}{\partial y} I(x, y, t+1)\, v_y$

Le problème devient alors :

→ $\min_{\vec{v}} \left\| \frac{\partial}{\partial x} I(x, y, t+1)\, v_x + \frac{\partial}{\partial y} I(x, y, t+1)\, v_y + I(x, y, t+1) - I(x, y, t) \right\| + \mathrm{AP}$

→ $\min_{\vec{v}} \left\| I_x(x, y, t+1)\, v_x + I_y(x, y, t+1)\, v_y + I_t(x, y, t+1) \right\| + \mathrm{AP}$

→ $\min_{\vec{v}} \left\| \nabla I(x, y, t+1) \cdot \vec{v} + I_t(x, y, t+1) \right\| + \mathrm{AP} = F(\vec{v})$

La minimisation s'obtient en annulant le gradient ∇F de cette fonctionnelle. Pour l'obtenir, on utilise la dérivée directionnelle est définie comme suit :

$$F(\vec{v} + \vec{w}) = F(\vec{v}) + \langle \nabla F(\vec{v}) | \vec{w} \rangle + \mathrm{o}(\|\vec{w}\|)$$

II.1. Estimation basée pixels localement

Démonstration pour la norme 2 :

$$F(\vec{v}+\vec{w}) = \frac{1}{2}\|\nabla I \cdot (\vec{v}+\vec{w}) + I_t\|_2^2 + \frac{\lambda}{2}\|\nabla(v_x+w_x)\|_2^2 + \frac{\lambda}{2}\left\|\nabla\left(v_y+w_y\right)\right\|_2^2$$

$$F(\vec{v}+\vec{w}) = \frac{1}{2}\|\nabla I \cdot \vec{v} + \nabla I \cdot \vec{w} + I_t\|_2^2 + \frac{\lambda}{2}\|\nabla v_x + \nabla w_x\|_2^2 + \frac{\lambda}{2}\left\|\nabla v_y + \nabla w_y\right\|_2^2$$

$$F(\vec{v}+\vec{w}) = \frac{1}{2}\|\nabla I \cdot \vec{v} + I_t\|_2^2 + \langle \nabla I \cdot \vec{v} + I_t | \nabla I \cdot \vec{w}\rangle + \frac{\lambda}{2}\|\nabla v_x\|_2^2 + \lambda\langle \nabla v_x | \nabla w_x\rangle + \frac{\lambda}{2}\left\|\nabla v_y\right\|_2^2 + \lambda\left\langle \nabla v_y \middle| \nabla w_y\right\rangle + o(\|\vec{w}\|)$$

$$F(\vec{v}+\vec{w}) = F(\vec{v}) + \langle \nabla I \cdot \vec{v} + I_t | \nabla I \cdot \vec{w}\rangle + \lambda\langle \nabla v_x | \nabla w_x\rangle + \lambda\left\langle \nabla v_y \middle| \nabla w_y\right\rangle + o(\|\vec{w}\|)$$

$$F(\vec{v}+\vec{w}) = F(\vec{v}) + \langle \nabla I(\nabla I \cdot \vec{v} + I_t) | \vec{w}\rangle + \lambda\langle -\Delta v_x | w_x\rangle + \lambda\left\langle -\Delta v_y \middle| w_y\right\rangle + o(\|\vec{w}\|)$$

$$F(\vec{v}+\vec{w}) = F(\vec{v}) + \langle \nabla I(\nabla I \cdot \vec{v} + I_t) | \vec{w}\rangle + \left\langle -\lambda \begin{pmatrix}\Delta v_x \\ \Delta v_y\end{pmatrix} \middle| \vec{w}\right\rangle + o(\|\vec{w}\|)$$

$$F(\vec{v}+\vec{w}) = F(\vec{v}) + \left\langle \nabla I(\nabla I \cdot \vec{v} + I_t) - \lambda \begin{pmatrix}\Delta v_x \\ \Delta v_y\end{pmatrix} \middle| \vec{w}\right\rangle + o(\|\vec{w}\|)$$

II.2. Estimation basée pixels dans un voisinage

Estimation sur un voisinage de taille $(2n+1) \times (2n+1)$

p_{i-k}				
		p_i		
				p_{i+k}

(Lucas & Kanade, 1981)

$$\begin{cases} I_x(p_{i-k})v_x(p_i) + I_y(p_{i-k})v_y(p_i) = -I_t(p_{i-k}) \\ \vdots \\ I_x(p_{i+k})v_x(p_i) + I_y(p_{i+k})v_y(p_i) = -I_t(p_{i+k}) \end{cases}$$

Résolution des $(2n+1)^2$ équations par moindres carrés :

$$A = \begin{bmatrix} I_x(p_{i-k}) & I_y(p_{i-k}) \\ \vdots & \vdots \\ I_x(p_{i+k}) & I_y(p_{i+k}) \end{bmatrix} \quad \vec{v} = \begin{bmatrix} v_x(p_i) \\ v_y(p_i) \end{bmatrix} \quad b = \begin{bmatrix} -I_t(p_{i-k}) \\ \vdots \\ -I_t(p_{i+k}) \end{bmatrix}$$

II.2. Estimation basée pixels dans un voisinage

Calcul de $\vec{v}$ pour obtenir le déplacement :

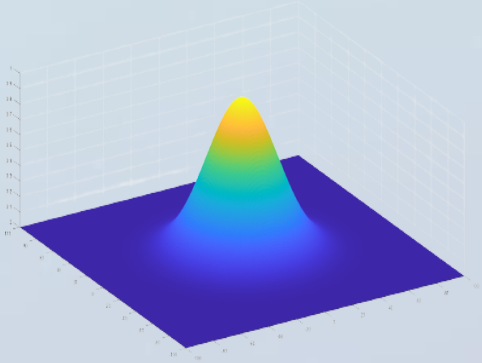
$$\vec{v} = \left(A^\top A\right)^{-1} A^\top b = \begin{bmatrix} \sum_k I_x(p_{i+k})^2 & \sum_k I_x(p_{i+k}) I_y(p_{i+k}) \\ \sum_k I_x(p_{i+k})\, I_y(p_{i+k}) & \sum_k I_y(p_{i+k})^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_k I_x(p_{i+k}) I_t(p_{i+k}) \\ -\sum_k I_y(p_{i+k}) I_t(p_{i+k}) \end{bmatrix}$$

Amélioration possible : donner plus d'importance au pixel du centre qu'à ceux les plus éloignés

→ assigner un poids pour chaque pixel, comme superposer une gaussienne 2D par exemple

→ choix de l'écart-type pour être plus ou moins sélectif

$$G = \begin{bmatrix} g(p_{i-k}) \\ \vdots \\ g(p_{i+k}) \end{bmatrix} \rightarrow \vec{v} = \left(A^\top G A\right)^{-1} A^\top G b$$



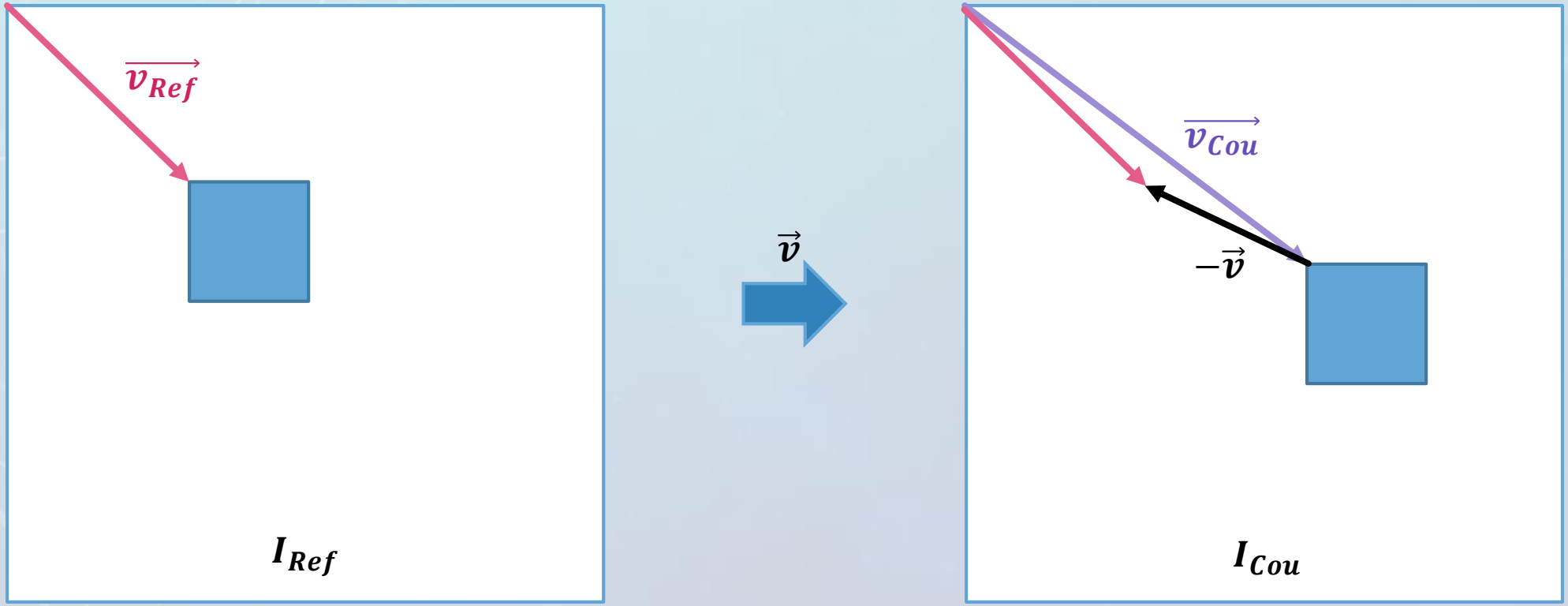
On est toujours sous l'hypothèse que les déplacements sont petits !
Dans le cas contraire, il faut passer par une approche multi-résolution et prendre le déplacement de la résolution précédente comme point de départ de la résolution suivante.

II.3 Estimation basée patchs

Définition des 2 patchs de taille t dans les 2 images :

$\text{patch}_{Ref}(\overrightarrow{v_{Ref}}) = I_{Ref}(\overrightarrow{v_{Ref}} \text{ à } \overrightarrow{v_{Ref}} + t - 1)$

$\text{patch}_{Cou}(\overrightarrow{v_{Cou}}) = I_{Cou}(\overrightarrow{v_{Cou}} \text{ à } \overrightarrow{v_{Cou}} + t - 1)$



II.4 Recherche du patch optimal

La recherche du patch optimal (appelée *Block Matching* en anglais) se fait en :

→ minimisant la distance / différence entre le patch courant patch_{Cou} et un des patch de I_{Ref}

→ maximisant une ressemblance entre le patch courant patch_{Cou} et un des patch de I_{Ref}

$$\min_{\text{patch}_{Ref}} \text{erreur}(\text{patch}_{Cou}, \text{patch}_{Ref}) \Leftrightarrow \max_{\text{patch}_{Ref}} \text{ressemblance}(\text{patch}_{Cou}, \text{patch}_{Ref})$$

Comment parcourir l'image de référence à la recherche du patch optimal ?

→ recherche exhaustive : beaucoup trop coûteux de parcourir tous les patchs possibles dans I_{Ref}

→ recherche dans un voisinage : cela suppose que le déplacement est petit (pas toujours vérifié)

→ recherche suivant un « chemin » qui fait diminuer l'erreur entre les 2 patchs

II.4 Recherche du patch optimal

Les métriques souvent utilisées pour la recherche du patch optimal, et donc du déplacement associé :

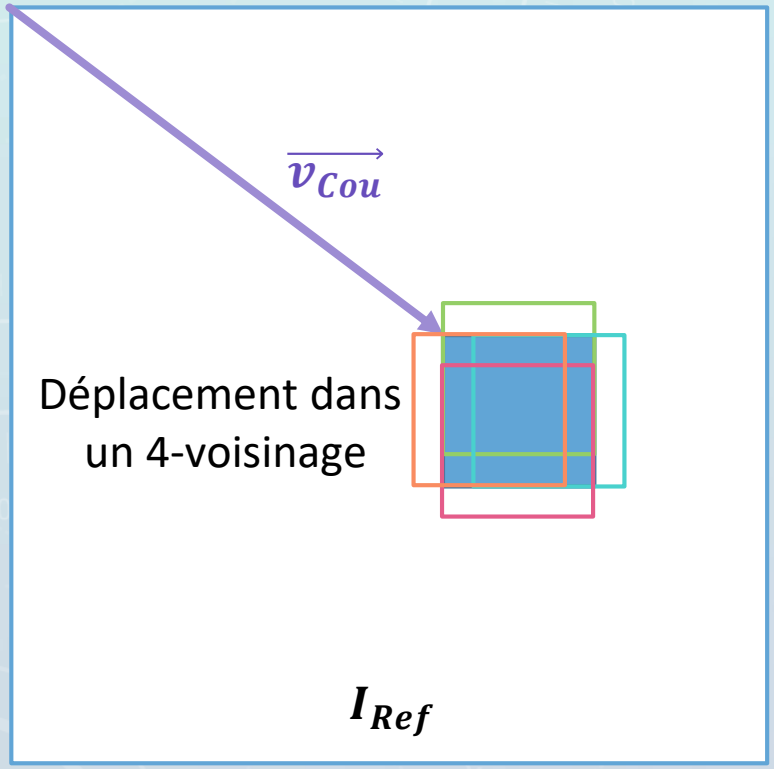
$$\text{EQM} : \min_{\vec{v}} \sum_{i,j} \left(I_{Cou}\left(\overrightarrow{v_{Cou}} + \begin{pmatrix} i \\ j \end{pmatrix}\right) - I_{Ref}\left(\overrightarrow{v_{Cou}} - \vec{v} + \begin{pmatrix} i \\ j \end{pmatrix}\right)\right)^2$$

$$\text{EAM} : \min_{\vec{v}} \sum_{i,j} \left| \left(I_{Cou}\left(\overrightarrow{v_{Cou}} + \begin{pmatrix} i \\ j \end{pmatrix}\right) - I_{Ref}\left(\overrightarrow{v_{Cou}} - \vec{v} + \begin{pmatrix} i \\ j \end{pmatrix}\right)\right)\right|$$

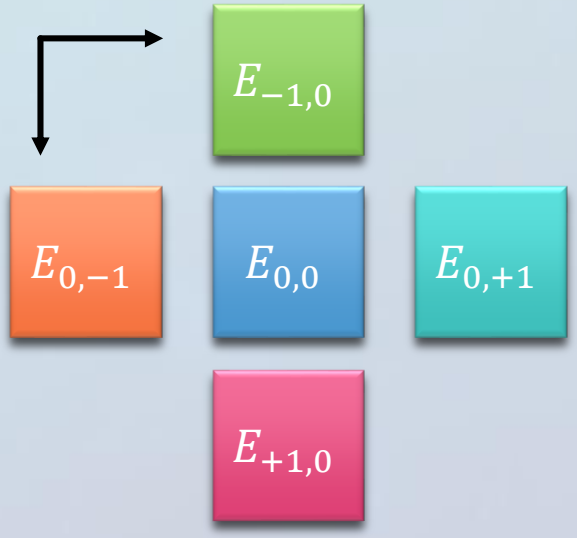
$$\text{CCMN} : \max_{\vec{v}} \frac{\sum_{i,j}\left(I_{Cou}\left(\overrightarrow{v_{Cou}}+\begin{pmatrix} i \\ j \end{pmatrix}\right)-\overline{I_{Cou}}\right)\left(I_{Ref}\left(\overrightarrow{v_{Cou}}-\vec{v}+\begin{pmatrix} i \\ j \end{pmatrix}\right)-\overline{I_{Ref}}\right)}{\sqrt{\sum_{i,j}\left(I_{Cou}\left(\overrightarrow{v_{Cou}}+\begin{pmatrix} i \\ j \end{pmatrix}\right)-\overline{I_{Cou}}\right)}\sqrt{\sum_{i,j}\left(I_{Ref}\left(\overrightarrow{v_{Cou}}-\vec{v}+\begin{pmatrix} i \\ j \end{pmatrix}\right)-\overline{I_{Ref}}\right)}}$$

II.4 Recherche du patch optimal

Parcourir un « chemin » dans l'image avec le *Cross Search Algorithm*



Principe : se déplacer dans la « case » où l'erreur entre le patch courant patch_{Cou} et le patch de référence patch_{Ref} est plus faible que celle calculée à la position en cours $E_{0,0}$ → déplacement en haut, à gauche, à droite, en bas puis on réitère jusqu'à ce que $E_{0,0}$ soit minimale.



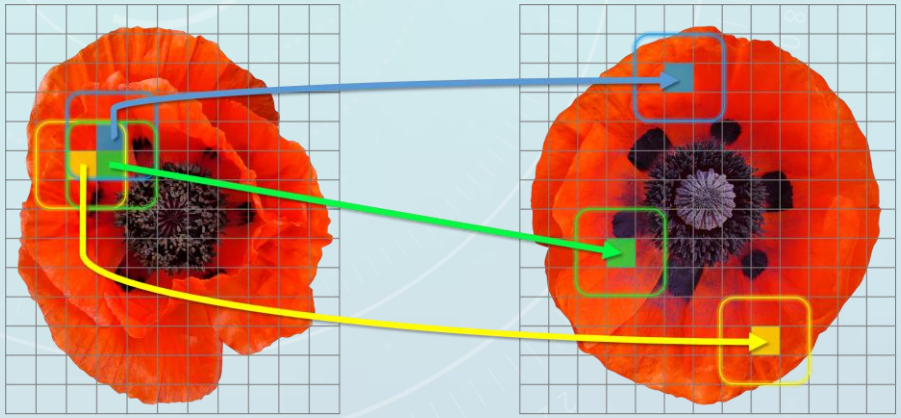
Seulement un minimum local cependant …

D'où l'intérêt d'avoir un bon point d'initialisation !

II.4 Recherche du patch optimal

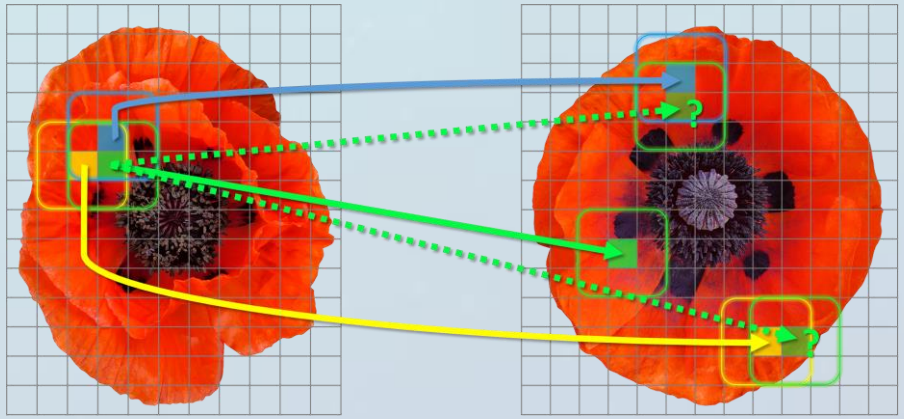
Essayer de trouver un minimum global sans être exhaustif

1) Initialisation des correspondances

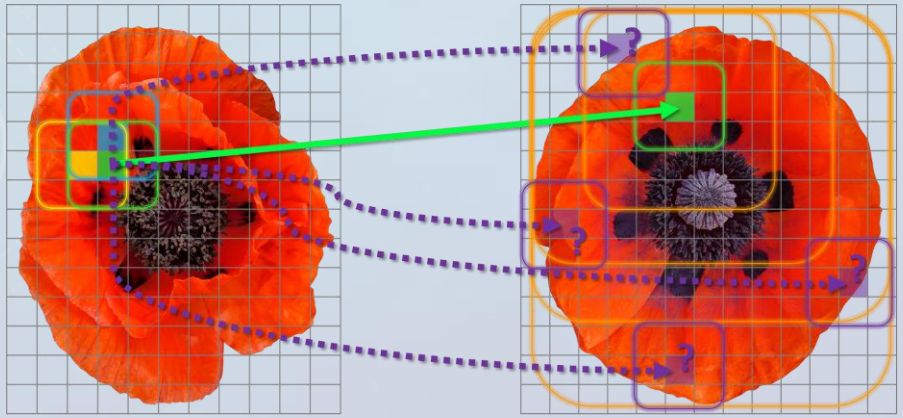


PatchMatch
(Barnes et al. ‘09)

2) Propagation dans le sens direct



3) Recherche aléatoire

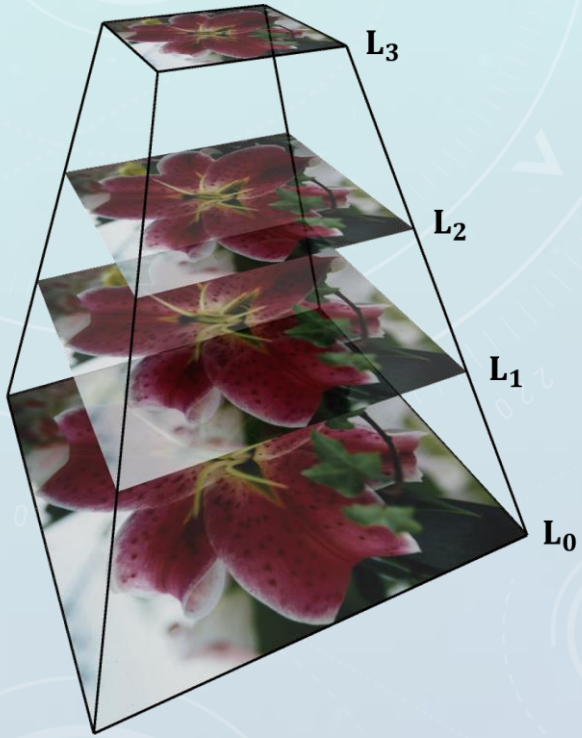


Est basé sur la loi des grands nombres pour déterminer qu’un minimum acceptable sera trouvé

Répété au moins 5 fois en alternant sens direct et sens indirect

II.5 Estimation par réseau de neurones

Vers la multi-résolution implicite



Estimer les déplacements à la plus grande échelle reste compliqué :

→ Problème de minimisation d'énergie trop complexe dans le cas non linéaire

→ Problème de convergence vers un minimum local car pas convexe

L'utilisation de l'approche par multi-résolution est la solution, mais :

→ Il faut faire attention au contenu fréquentiel que l'on veut conserver

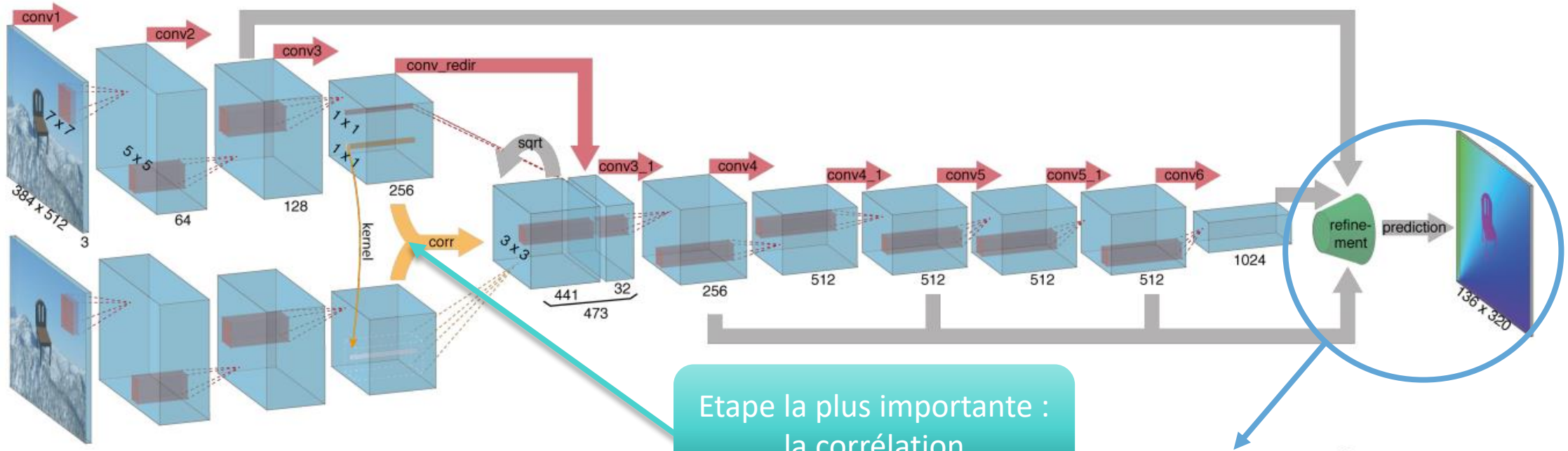
→ Une étape de filtrage pour éliminer les hautes fréquences est nécessaire

Les réseaux de neurones ont les outils pour cela :

→ Les couches de convolutions reviennent à effectuer un filtrage fréquentiel

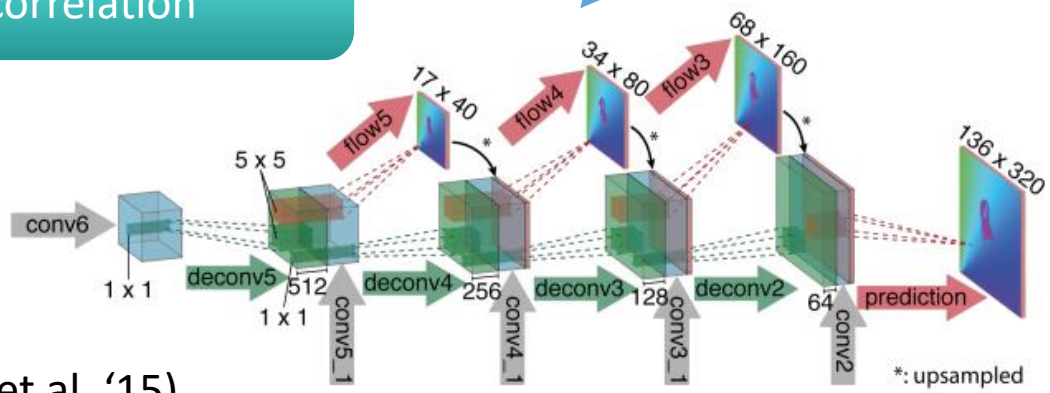
→ L'étape de *pooling* fait quant à elle office de réduction de la dimension spatiale

II.5 Estimation par réseau de neurones



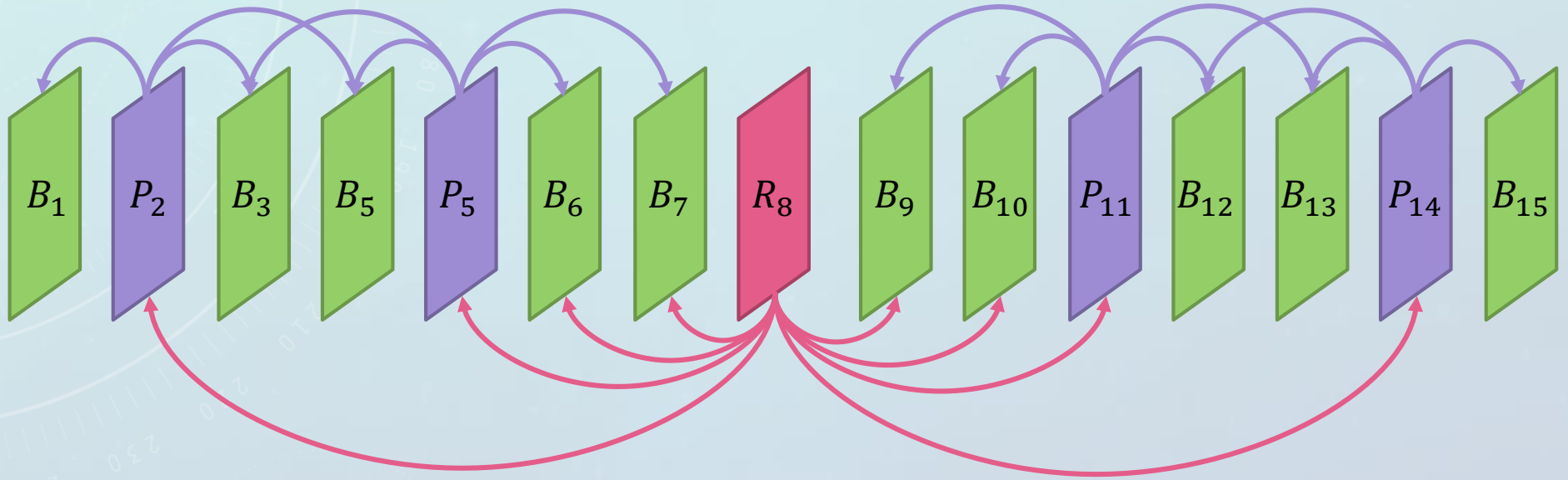
Etape la plus importante : la corrélation

Fonction de coût du réseau :
End Point Error (EPE) = $\|\overrightarrow{v_{GT}} - \overrightarrow{v_{estimé}}\|$



FlowNet (Fischer et al. ‘15)

III. Codage MPEG-2



R = Image de référence (codée en JPEG)
P = Image prédite à partir de R (codée en MPEG)
B = Image bidirectionnelle prédite à partir de R et P ou P et P (codée en MPEG)

III. Codage MPEG-2

Avantages et inconvénients des images bidirectionnelles

→ Elles permettent de gérer le problème des changements de scène vu qu'il y a 2 images différentes comme « références »

→ Comme des images prédites sont utilisées lors de la prédiction des images bidirectionnelles, les erreurs sont également propagées.

Les différentes étapes du codage MPEG-2

1) Estimation des déplacements $\vec{v}$ entre I_{Cou} et I_{Ref} (*Block Matching* avec algorithme *CSA*)
2) Calcul de l'image prédite I_{Pre} avec I_{Ref} et $\vec{v}$
3) Calcul de l'image résiduelle I_{Res} entre I_{Cou} et I_{Pre}
4) Compression JPEG de I_{Ref}
5) Compression JPEG de I_{Res} (cf. table pour les résidus)
6) Codage entropique des déplacements $\vec{v}$