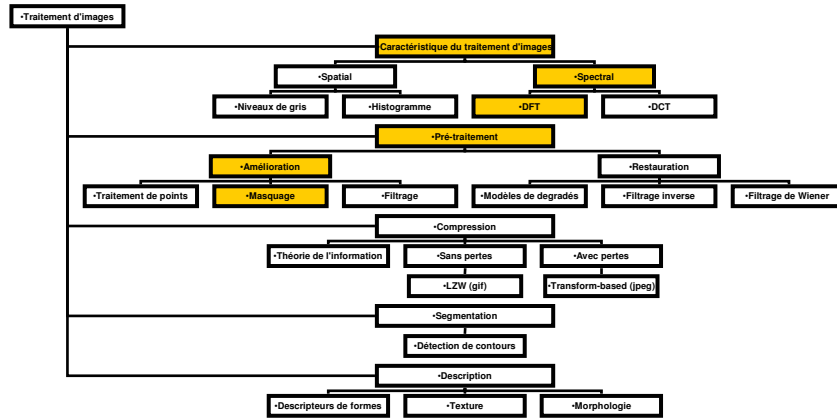

Caractérisation des applications graphiques et multimédias

Daniel Etiemble
de@lri.fr

Applications multimédia

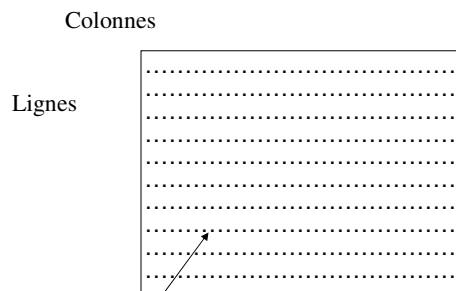
- Compression image/vidéo/audio (JPEG/MPEG/GIF/png)
- Frontal du pipeline graphique 3D (géométrie, éclairage)
- Synthèse audio haute qualité
- Traitement d'images
 - Adobe Photoshop
- Reconnaissance de la parole
 - Frontal: filtres/FFTs
 - Probabilité de phonèmes : réseaux neuronaux
 - Traitement principal : Viterbi/Recherche en faisceau

Traitement d'images



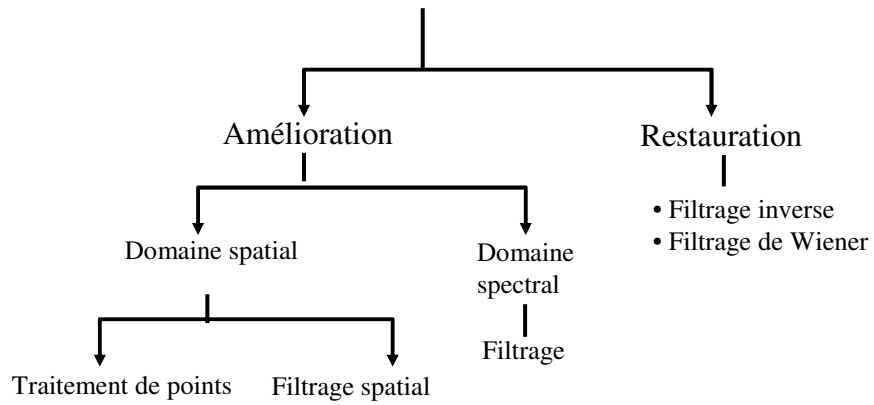
Image

- Matrice de pixels
 - Image
 - 640 x 480
 - 800 x 600
 - etc
 - Image Vidéo
 - 352 x 288 (vidéo VHS)
 - 704 x 576 (télévision)
 - 1440 x 1152 (télévision HD)
- Pixel
 - Gris
 - Couleurs
 - RGB
 - YCrCb



Pixel : octet (niveau de gris)
RGB (couleurs)

Prétraitement des images

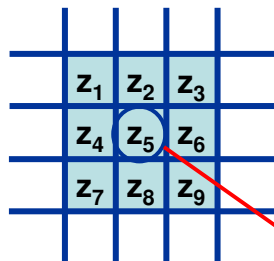


Programmes graphiques et multimédia sur PC
D. Etiemble

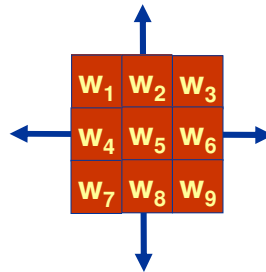
5

Filtrage spatial (Masquage)

Partie d'une image



Masque



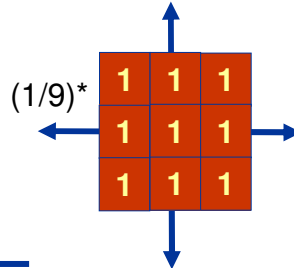
Remplacé **R** par $= w_1z_1 + w_2z_2 + \dots + w_9z_9$

Programmes graphiques et multimédia sur PC
D. Etiemble

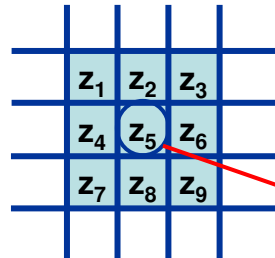
6

Filtres passe-bas

Filtre moyenne



Filtre médian

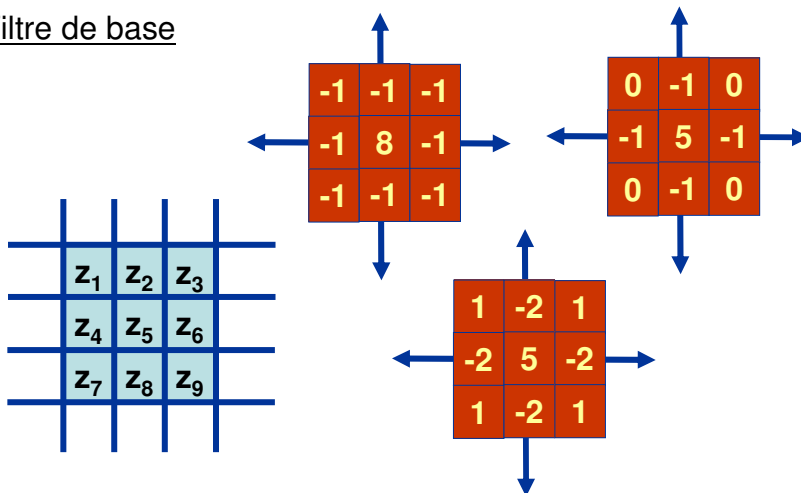


Remplacé
par

$$\mathbf{R} = \text{médiane}(z_1, z_2, \dots, z_9)$$

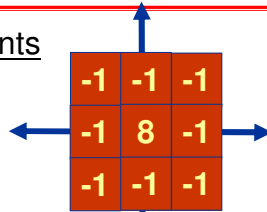
Filtres passe-haut

Filtre de base

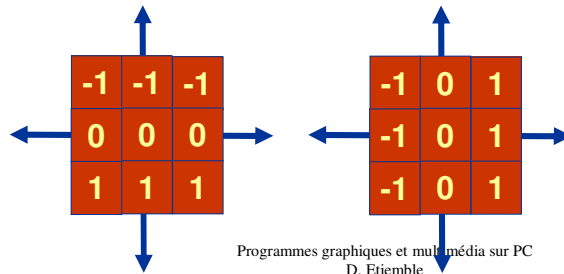


Détection de discontinuités

Détection de points



Détection de ligne (Gradient de Prewitt)

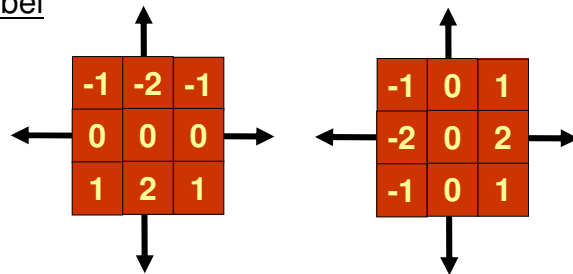


Programmes graphiques et multimédia sur PC
D. Etienne

9

Détection de contours

Masques de Sobel



Image[1] = Image*Sobel [horizontal]

Image[2] = Image*Sobel [vertical]

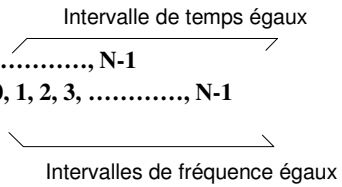
Image filtrée = $\sqrt{I[1]*I[1]+I[2]*I[2]}$

Programmes graphiques et multimédia sur PC
D. Etienne

10

DFT une dimension

- **Domaines discrets**
 - Temps discrétisé: $k = 0, 1, 2, 3, \dots, N-1$
 - Fréquence discrétisée: $n = 0, 1, 2, 3, \dots, N-1$



- **Transformée de Fourier discrétisée (DFT)**

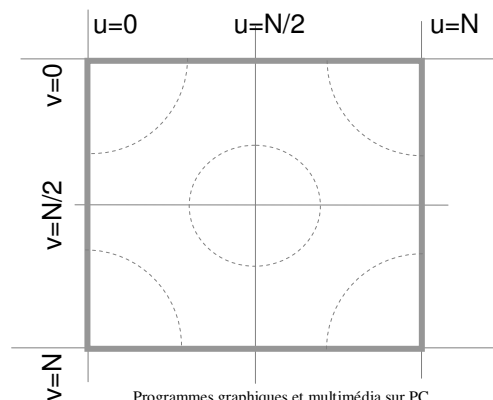
$$X[n] = \sum_{k=0}^{N-1} x[k] e^{-j\left(\frac{2\pi}{N}\right)nk}; \quad n = 0, 1, 2, \dots, N-1$$

- **DFT inverse**

$$x[k] = \frac{1}{N} \sum_{n=0}^{N-1} X[n] e^{j\left(\frac{2\pi}{N}\right)nk}; \quad k = 0, 1, 2, \dots, N-1$$

DFT deux dimensions

$$F(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \exp\left[-j \frac{2\pi(ux + vy)}{N}\right]$$



Algorithme de Cooley-Tukey (DFT)

- Algorithme DFT pour une puissance de 2 $N = 2^V$

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{nk} = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N}; W_N = e^{-j2\pi/N}$$

- Sommées séparées pour les valeurs paires et impaires de n :

$$X[k] = \sum_{n \text{ pair}} x[n]W_N^{nk} + \sum_{n \text{ impair}} x[n]W_N^{nk}$$

- Avec $n = 2r$ pour n pair et $n = 2r+1$ pour n impair

$$X[k] = \sum_{r=0}^{(N/2)-1} x[2r]W_N^{2rk} + \sum_{r=0}^{(N/2)-1} x[2r+1]W_N^{(2r+1)k}$$

Algorithme de Cooley-Tukey (DFT)

mais $W_N^2 = e^{-j2\pi 2/N} = e^{-j2\pi/(N/2)} = W_{N/2}$ et $W_N^{2rk} W_N^k = W_N^k W_{N/2}^{rk}$

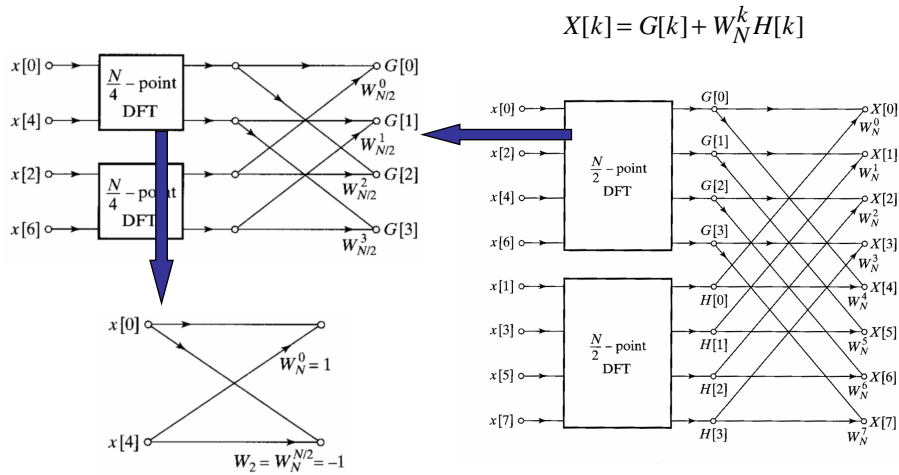
et ...

$$X[k] = \underbrace{\sum_{n=0}^{(N/2)-1} x[2r]W_{N/2}^{rk}}_{\text{DFT } N/2 \text{ points de } x[2r]} + W_N^k \underbrace{\sum_{n=0}^{(N/2)-1} x[2r+1]W_{N/2}^{rk}}_{\text{DFT } N/2 \text{ points de } x[2r+1]}$$

DFT $N/2$ points de $x[2r]$ DFT $N/2$ points de $x[2r+1]$

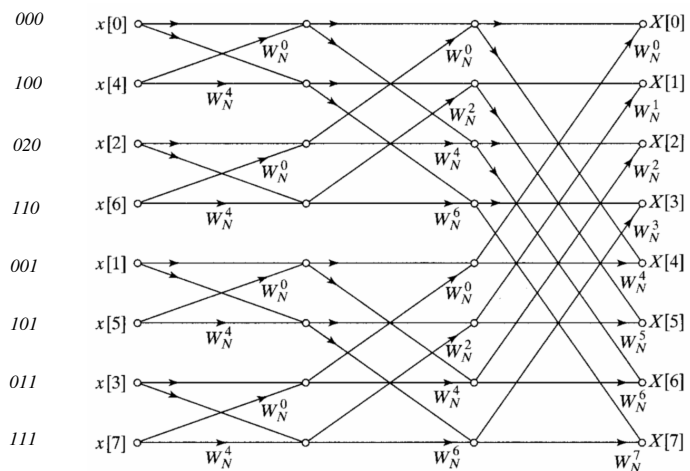
$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n]W_N^{nk} \\ &= \sum_{n=0}^{(N/2)-1} x[2r]W_{N/2}^{rk} + W_N^k \sum_{n=0}^{(N/2)-1} x[2r+1]W_{N/2}^{rk} \end{aligned}$$

Représentation d'une DFT 8 points



La FFT complète 8 points

*Ordre
opposé
des bits
en
entrée*



La DFT 2 points

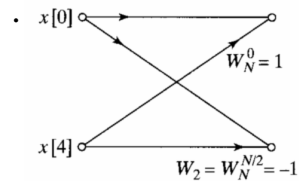
• Expression :

$$X[k] = \sum_{n=0}^1 x[n]W_2^{nk} = \sum_{n=0}^1 x[n]e^{-j2\pi nk/2}$$

- Avec $k = 0, 1$ on obtient

$$X[0] = x[0] + x[1]$$

$$X[1] = x[0] + e^{-j2\pi/2}x[1] = x[0] - x[1]$$



Papillon FFT

FFT : Version « Embree »

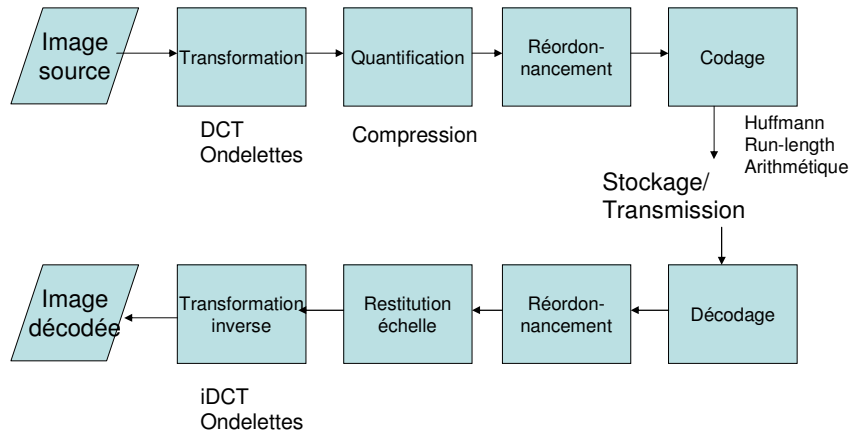
```
void fft_c(int n, COMPLEX *x, COMPLEX *w)
{ COMPLEX u, temp, tm;
  int i, j, le, windex;
  windex = 1;
  for(le=n/2 ; le > 0 ; le/=2) {
    wptr = w;
    for(j = 0 ; j < le ; j++) {
      u = *wptr;
      for(i = j ; i < n ; i = i + 2*le) {
        xi = x + i;
        xip = xi + le;
        temp.real = xi->real + xip->real;
        temp.imag = xi->imag + xip->imag;
        tm.real = xi->real - xip->real;
        tm.imag = xi->imag - xip->imag;
        xip->real = tm.real*u.real - tm.imag*u.imag;
        xip->imag = tm.real*u.imag + tm.imag*u.real;
        *xi = temp; }
      wptr = wptr + windex; }
    windex = 2*windex; }
```

- Embree, C algorithms for Real-Time DSP, Prentice Hall

- Caractéristiques

- POINTEURS
- STRUCTURE
- PAS NON UNITAIRES

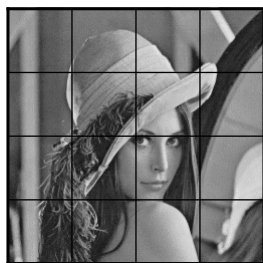
Codeurs et décodeurs vidéo



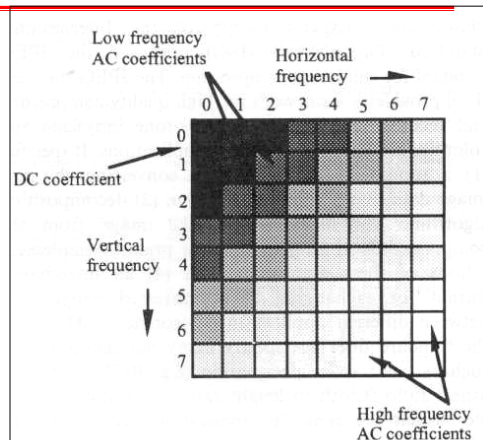
Programmes graphiques et multimédia sur PC
D. Etiemble

19

Codage vidéo – DCT



DCT

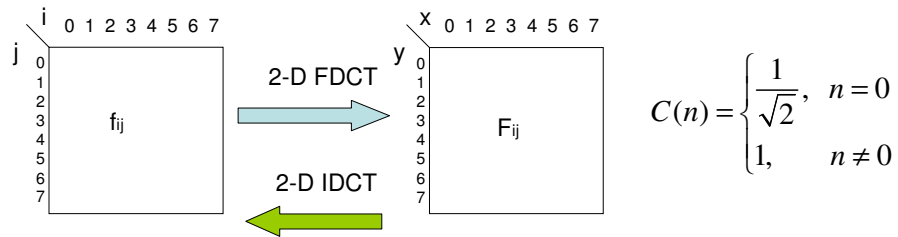


- ❑ La DCT opère sur des blocs 8 x 8.
- ❑ Donne l'information de fréquence de l'image

Programmes graphiques et multimédia sur PC
D. Etiemble

20

Transformée en cosinus (DCT, iDCT)



$$F_{x,y} = \frac{C(x)C(y)}{4} \sum_{i=0}^7 \sum_{j=0}^7 f_{i,j} \cos\left(\frac{(2i+1)x\pi}{16}\right) \cos\left(\frac{(2j+1)y\pi}{16}\right)$$

$$f_{i,j} = \frac{C(x)C(y)}{4} \sum_{i=0}^7 \sum_{j=0}^7 F_{x,y} \cos\left(\frac{(2i+1)x\pi}{16}\right) \cos\left(\frac{(2j+1)y\pi}{16}\right)$$

DCT rapides

$$F_{x,y} = \frac{C(x)C(y)}{4} \sum_{i=0}^7 \sum_{j=0}^7 f_{i,j} \cos\left(\frac{(2i+1)x\pi}{16}\right) \cos\left(\frac{(2j+1)y\pi}{16}\right)$$



$$F_{x,y} = \frac{C(y)}{2} \sum_{j=0}^7 F_x \cos\left(\frac{(2j+1)y\pi}{16}\right) \text{ avec } F_x = \frac{C(x)}{2} \sum_{i=0}^7 f_i \cos\left(\frac{(2i+1)x\pi}{16}\right)$$

$$F_{x,y} = \text{1D-DCT}_{\text{direction y}} (\text{1D DCT}_{\text{direction x}})$$

Etc... (les optimisations prennent en compte les propriétés des cosinus)

Code C pour la DCT 2D

```

Void 2D_DCT (image[width][height])
{
  for (i=0;i<height/8;i++)
    for (j=0;j<width/8;j++)
      {
        1D_row_dct (coeff[8][8], block[8][8]);
        1D_col_dct (coeff[8][8], block[8][8]);
      }
}

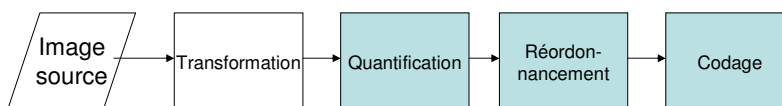
```

```

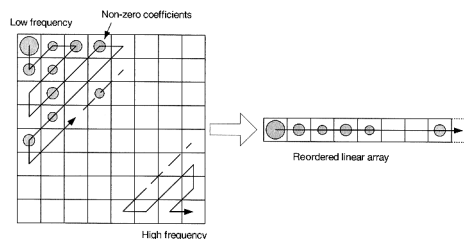
Void 1D_XXX_DCT (coeff[8][8], block[8][8])
{
  Tranpose (block[8][8])
  for (k=0;k<8;k++) {
    for (l=0;l<8;l++){
      temp=0;
      for (m=0;m<8;m++)
        temp+=coeff[k][m]*block[l][m]
      output[k][l]=temp;
    }
  }
}

```

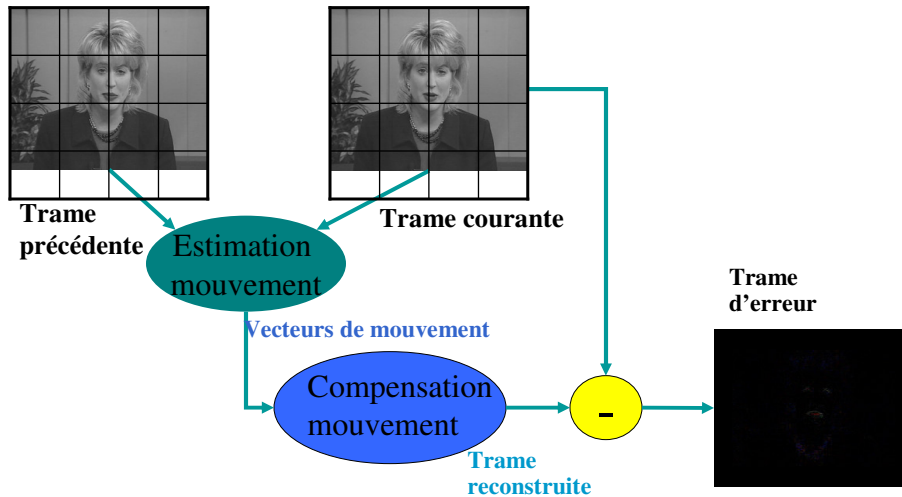
Codec (suite)



- **Quantification**
 - Granularité pour éliminer les « petits » coefficients
- **Réordonnement**
 - Parcours en zig-zag
- **Codage**
 - «run length »
 - Codage de Huffman



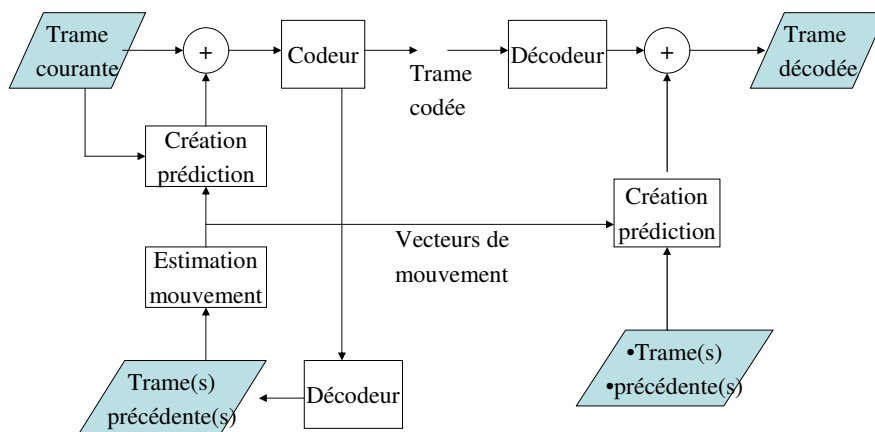
Codage vidéo – Compensation mouvement



Programmes graphiques et multimédia sur PC
D. Etiemble

25

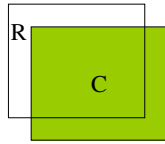
Codec avec estimation de mouvement et compensation



Programmes graphiques et multimédia sur PC
D. Etiemble

26

Mouvement (comparaison de blocs)



Blocs 8 x 8

$$MSE = \frac{1}{64} \sum_{i=0}^{i=7} \sum_{j=0}^{j=7} (C_{ij} - R_{ij})^2$$

$$MAE = \frac{1}{64} \sum_{i=0}^{i=7} \sum_{j=0}^{j=7} |C_{ij} - R_{ij}|$$

« Minimiser la différence d'énergie »

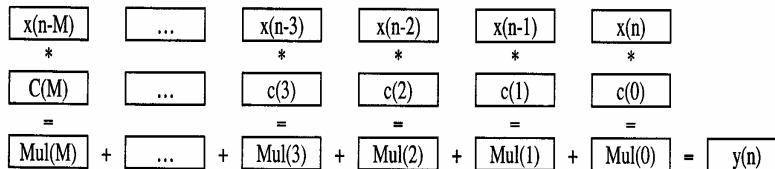
$$SAE = \sum_{i=0}^{i=7} \sum_{j=0}^{j=7} |C_{ij} - R_{ij}|$$

Instruction spéciale dans toutes les extensions SIMD existantes des jeux d'instructions des processeurs généralistes

Algorithme rapide pour la comparaison des deux images utilisant la comparaison de blocs

Les filtres à réponse impulsionnelle (FIR et IIR)

$$\text{FIR} \quad y(n) = \sum_{k=0}^M c_k x(n-k)$$



$$\text{IIR} \quad y(n) = \sum_{k=0}^M c_k x(n-k) - \sum_{p=1}^P a_p y(n-p)$$

Caractéristiques du traitement multimédia

- Traitement numérique du signal, rendu graphique 2D/3D, compression/décompression image/audio
- Contrainte temps réel, haute performance
- Grande quantité de parallélisme de données, tolérance à la latence
- Traitement continu des données, très peu de réutilisation des données
- Calcul intensif, avec 100-200 opérations arithmétiques sur chaque élément de données
- Flot de données prévisible (nids de boucle)
- Flot continu des accès mémoire
- Structures de données 8 bits, 16 bits, 24 bits
- Accès mémoire spéciaux

Les accès aux données typiques dans les noyaux DSP et multimédia

- Accès séquentiel
 - $A_0, A_1, A_2, \dots, A_{N-1}$
- Accès séquentiel avec déplacement
 - $A_{0+k}, A_{1+k}, A_{2+k}, \dots, A_{N-1+k}$
- Adressage avec permutation ($N/r=P$)
 - $A_0, A_p, A_{2p}, \dots, A_1, A_{p+1}, A_{2p+1}, \dots, A_2, A_{p+2}, A_{2p+2}, \dots$
- Adressage avec inversion de bit ($N=8$)
 - $A_0, A_4, A_2, A_6, A_1, A_5, A_3, A_7$
- Adressage avec réflexion
 - $A_0, A_{N-1}, A_1, A_{N-2}, \dots, A_m, A_{N-m}, \dots, A_{N/2-1}, A_{N/2}$

Les approches du traitement multimédia

