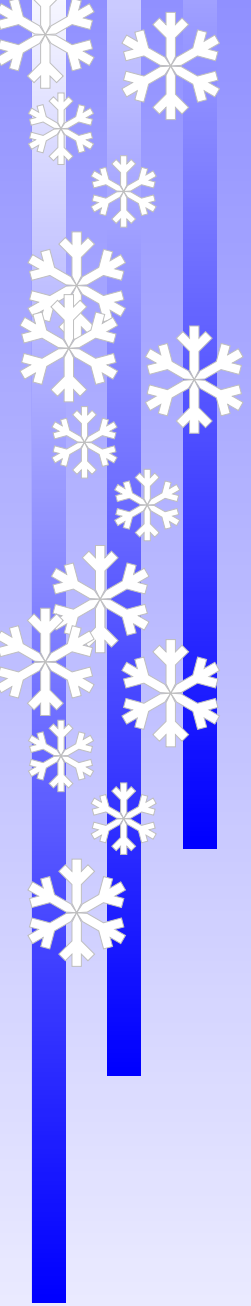


OpenCV

Open Source Computer Vision Library

http://www.intel.com/research/mrl/research/opencv/overview.htm

Introduction (1/2)

Bibliothèque de traitement d'images et de vision par ordinateur en langage C/C++, optimisée proposée par Intel pour Windows et Linux. Elle est installée à l'IRIT sur Linux, et comprend un très grand nombre d'opérateurs "classiques". Parmi lesquels :

- √ Création/libération d'images, macros d'accès rapide aux pixels
- √ Opérateurs standards :
  - ⋆ morphologie
  - ⋆ filtres dérivatifs, filtres de contours
  - ⋆ suppression de fond
  - ⋆ recherche de coins
- √ Recherche, manipulation, traitement de contours

Introduction (2/2)

- ✓ Pyramides d'images
- ✓ Dessins de primitives géométriques (lignes, rectangles, ellipses, polygones . . . et même du texte)
- ✓ Création et utilisation d'histogrammes
- ✓ Changements d'espaces de couleurs (RGB, HSV, L*a*b* et YCrCb)
- ✓ Interface Utilisateur
  - ⋆ Lecture/écriture d'images (JPEG, PPM, ...)
  - ⋆ Affichage à l'écran
  - ⋆ Gestion des signaux sur clic de fenêtre, fermeture, . . .

Opérateurs plus complexes

- √ Mean Shift
- √ Contours actifs
- √ Prédiction: Kalman et CONDENSATION
- √ Analyse du mouvement : Flot optique, MHI
- √ Détection de visages
- √ Calibrage de caméras (possible à partir d'un échiquier)
- √ Suivi d'objets 3D avec plusieurs caméras
- √ Mise en correspondance entre deux images
- √ lecture d'images à la volée directement depuis une vidéo AVI ou une caméra (☹Windows only)

Autres structures de données

- ✓ Tableaux, matrices et matrices creuses + opérateurs associés
  - ⋆ Opérateurs “classiques” de matrices (inversion, déterminant, transposée, valeurs/vecteurs propres, distance de Mahalanobis, . . .)
- ✓ Piles, files . . . génériques. Listes denses et listes creuses. Ensembles.
- ✓ Graphes
- ✓ Arbres
- ✓ Modèles de Markov Cachés (HMM) 1D et 2D, algorithme de Viterbi, ...

Création d'image

Déclaration et création :

```
#include <cv.h>
IplImage *im;
im = cvCreateImage(cvSize(512,256), IPL_DEPTH_8U, 3);
/* Image 512 x 256 de unsigned char avec 3 canaux */
```

Remplissage : (codage par défaut = BGR)

```
pteur = (unsigned char*)im->imageData;
fin = im->width*im->height;
for(i=0; i<fin; i++)
  {
    *(pteur++) = 255; /* Bleu */
    *(pteur++) = 0; /* Vert */
    *(pteur++) = 0; /* Rouge */
  }
```

Libération de l'espace mémoire

```
cvReleaseImage(&im);
```

Compiler un programme (sur Linux)

La commande *opencv-config* est bien pratique.

- ✓ Compilation :

```
g++ ‘opencv-config --cflags‘ -c mon_fichier.c
```

- ✓ Édition de liens :

```
g++ ‘opencv-config --libs‘ -o mon_executable mon_fichier.o
```

Compiler un programme (Windows)

- ✓ j’aimerais bien le savoir
- ✓ j’ai déjà réussi avec Visual Studio 6 grâce à la doc mais qu’une seule fois

Lire et afficher une image ⇒ highgui.h

- Lire une image

```
#include <highgui.h>
IplImage *im;
im = cvLoadImage("mon_image.jpg", 1); /* (1) */
/* 1 => 3 canaux (0 => 1 seul, -1 => automatique) */
im = cvLoadImage("mon_image.jpg", 0); /* (2) */
```

- Affi cher une image

```
cvNamedWindow("Ma fenetre", CV_WINDOW_AUTOSIZE);
cvShowImage("Ma fenetre", im);
cvWaitKey(0);     /* Attendre qu'une touche soit pressée */
```

(1)

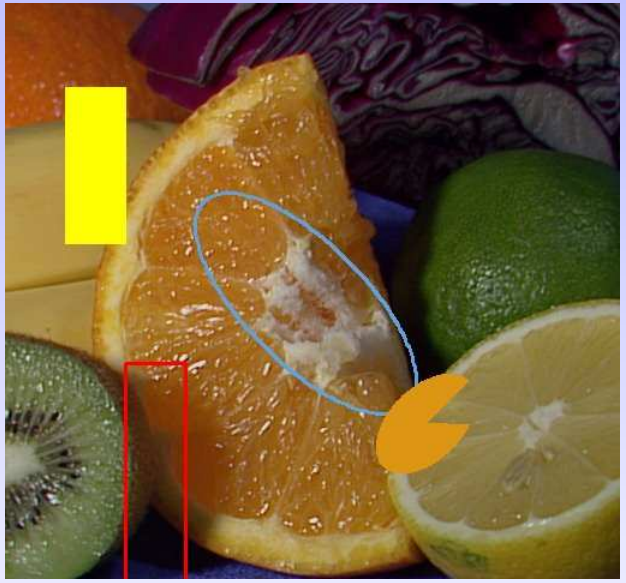
(2)

Dessins de primitives (1/2)

Attention au repère de l'image :

Par défaut repère possible

```
cvRectangle(im, cvPoint(100,300),
            cvPoint(150,700),
            CV_RGB(255,0,0), 3);
cvRectangle(im, cvPoint(50,70),
            cvPoint(100,200),
            CV_RGB(255,255,0),
            CV_FILLED);
cvEllipse(im, cvPoint(250, 250),
          cvSize(50, 120), 45.0,
          0, 360,
          CV_RGB(100,160,220),3);
cvEllipse(im, cvPoint(350, 350), cvSize(50, 30), 45.0,
          0, 300, CV_RGB(220,150,20), CV_FILLED);
```



Dessins de primitives (2/2)

Il existe de nombreuses autres fonctions de manipulations de primitives géométriques. Citons :

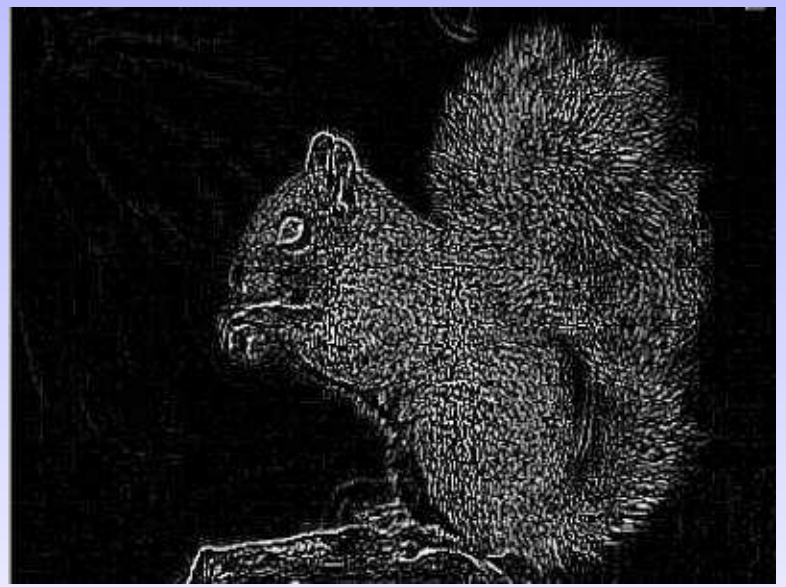
- ✓ renvoi de tous les points appartenant à un segment (délimité par ses deux extrémités)
- ✓ détection et traitement des contours

Détection de contours

On trouve les fi ltres de Canny, Sobel, Laplacien, détection de coins, . . .

Exemple du Laplacien :

```
im = cvLoadImage(argv[1], 0);
im2 = cvCreateImage(cvGetSize(im), IPL_DEPTH_16S, 1);
cvLaplace(im, im2);
```

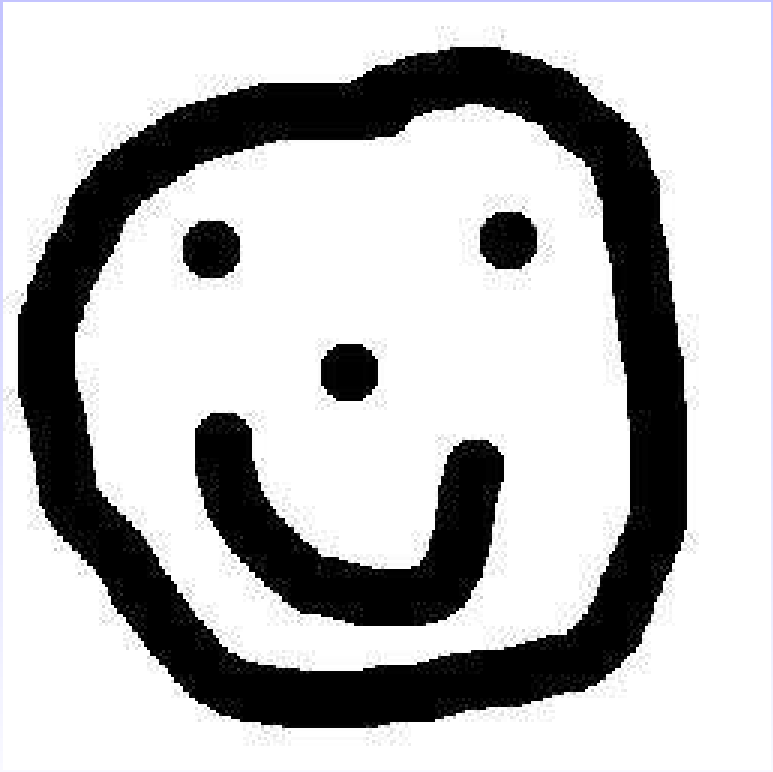


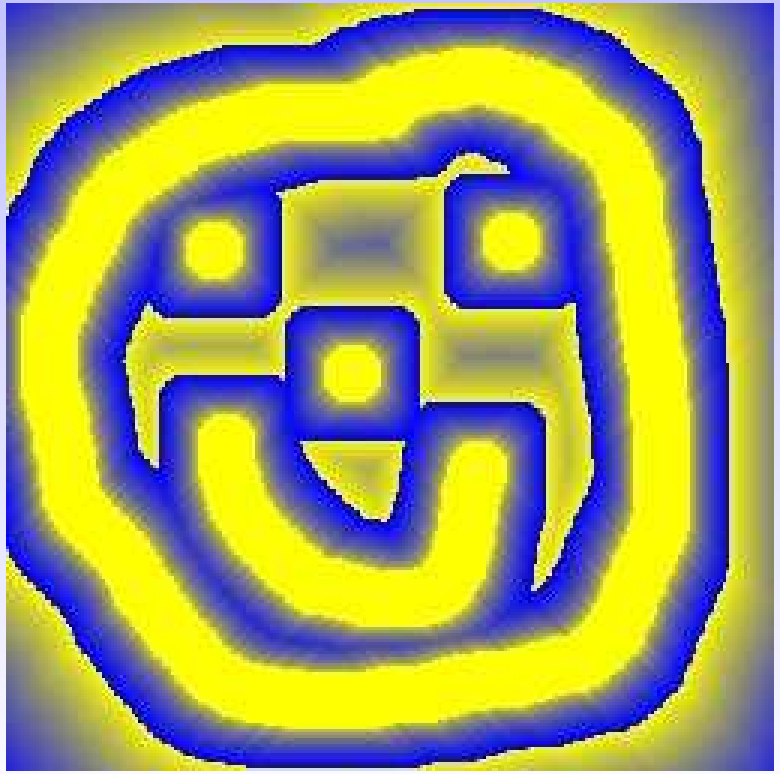
Transformée en distances

Il s'agit de la transformée de Borgefors, avec plusieurs filtres :

```
CV_DIST_C (3x3):  a=1, b=1
CV_DIST_L1 (3x3): a=1, b=2
CV_DIST_L2 (3x3): a=0.955, b=1.3693
CV_DIST_L2 (5x5): a=1, b=1.4, c=2.1969
```

On peut aussi définir ses propres filtres (en donnant a, b et c).

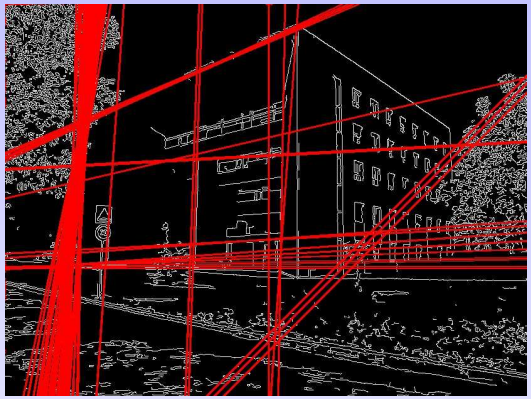




Transformée de Hough

Détection de lignes en utilisant le plan (ρ, θ).

- ✓ méthode “standard” (1) : renvoie un ensemble de droites
- ✓ méthode “probabiliste” (2) : renvoie un ensemble de segments

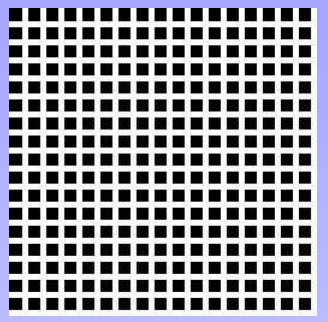


(1) (2)

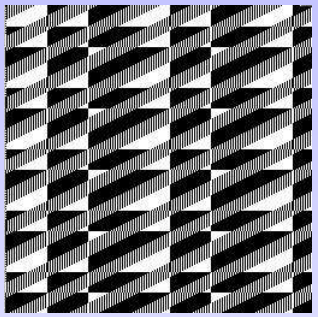
Un exemple de programme est disponible dans l’aide en ligne.

Transformée de Fourier et transformée cosinus

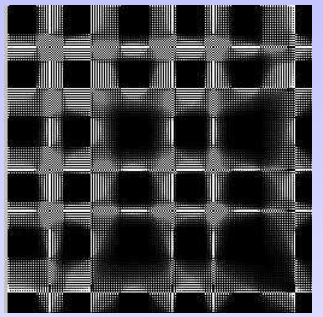
- transformée de Fourier (DFT) :
  - $\mathbb{R} \rightarrow \mathbb{R}$
  - $\mathbb{C} \rightarrow \mathbb{C}$
- transformée en cosinus (DCT) :
  - $\mathbb{R} \rightarrow \mathbb{R}$
- pour les deux :
  - transformées 1D et 2D
  - transformée inverse



image



DFT



DCT

```
imf = cvCreateImage(cvGetSize(im), IPL_DEPTH_32F, 1);
dst = cvCreateImage(cvGetSize(im), IPL_DEPTH_32F, 1);
cvConvert(im, imf);
cvDFT(imf, dst, CV_DXT_FORWARD);
```

Histogrammes

On peut défi nir des histogrammes “standards” mais aussi des histogrammes “creux”. La création n’est pas directe : il faut auparavant désentrelacer les canaux de l’image.
Il existe des fonctions de comparaison d’histogrammes, traitement, rétroprojection sur une image, . . .

```
/* Declaration (on suppose que im existe et est definie */
    IplImage* rgb[3];
    int taillehist[3] = {32,32,16};
/* Desentracelacement de l’image */
    rgb[0] = cvCreateImage(cvGetSize(im), IPL_DEPTH_8U, 1);
    rgb[1] = cvCreateImage(cvGetSize(im), IPL_DEPTH_8U, 1);
    rgb[2] = cvCreateImage(cvGetSize(im), IPL_DEPTH_8U, 1);
    cvCvtPixToPlane(im, rgb[2], rgb[1], rgb[0], NULL);
/* Creation de l’histogramme */
    hist = cvCreateHist(3, taillehist, CV_HIST_ARRAY, NULL, 1);
    cvCalcHist(rgb, hist, 0, NULL);
/* Liberation */
    cvReleaseHist(&hist);
```

Lecture d'une vidéo AVI

Il est possible de lire une séquence d'images à partir d'un fi chier vidéo. Pour chaque étape, une image de type *IplImage* est automatiquement allouée (puis libérée) en mémoire.

```
/* Variables */
  IplImage *im;
  CvCapture *avi;

/* Ouverture de la video */
  avi = cvCaptureFromAVI("ma_video.AVI");
  while(cvGrabFrame(avi))
    {
      im = cvRetrieveFrame(avi);

      /* Traitement de l'image */

    }
```

Conclusion

- ✓ il existe encore de nombreuses fonctions, structures de données, . . .
- ✓ toutes les fonctions matricielles fonctionnent aussi sur les images.
- ✓ la documentation (papier et en ligne) décrit
  - ⋆ pratiquement tous les algorithmes implémentés
  - ⋆ quelques références bibliographiques
  - ⋆ des exemples de code
  - ⋆ des fichiers d'exemples de code pour les problèmes assez complexes