Applicazioni pratiche della visione artificiale

Parte 1.5 – OpenCV

Curato da: Ing. Francesco La Rosa
Università di Messina – Facoltà di Ingegneria

Corso di Calcolatori II – A.A. 2003/2004
Ing. Giancarlo Iannizzotto

Istogrammi

```
//Calcolo e tracciamento dell'istogramma di un'immagine gray-level
HINSTANCE   DLLhinst;
void CProvaDoc::OnHistCalc()
{     char path[256];
      //Inizializzazione di path
      memset(path,0,256);
      //scelgo il file da aprire attraverso la finestra open-file
      OPENFILENAME fn;
       fn.lStructSize = sizeof (OPENFILENAME);
       fn.hwndOwner = NULL;
       fn.lpstrFilter = NULL;
       fn.lpstrFile = path;
       fn.nMaxFile = 256;
       fn.lpstrFileTitle = NULL;
       fn.lpstrInitialDir = NULL;
       fn.lpstrTitle = NULL;
```

Istogrammi

```
fn.Flags = NULL;
fn.lpstrDefExt = "bmp";
fn.hInstance = DLLhinst;
fn.lpfnHook = NULL;
fn.lpstrCustomFilter = NULL;
fn.lCustData = NULL;

if(!GetOpenFileName(&fn))
                    return ;
char wndname01[] = "Source image";
char wndname02[] = "Histogram";
//#define DIMENSION_SIZE   50
int dims[1] = {DIMENSION_SIZE}; //numero di barre  per dimensione
float thresh[1][2] = { {0, 255} }; // range di valori da analizzare
float * pthresh[1] = { thresh[0] };
```


Istogrammi

```
IPLIMAGE   image01 = cvLoadImage( path );
// alias per IplImage* definito in highgui.h
// typedef IplImage* IPLIMAGE;

IplImage * image02 = cvCreateImage( cvSize(image01-> width,
                        image01->height), IPL_DEPTH_8U, 1);
IplImage * image03 = cvCreateImage( cvSize(image01-> width,
                        image01->height), IPL_DEPTH_8U, 3);

// converto l'immagine da colore a toni di grigio
iplColorToGray(image01,image02);
```

Istogrammi

```
// creo l' header dell'istogramma.
CvHistogram* hist01 = cvCreateHist(1, dims, CV_HIST_ARRAY,
                                   0, 1);
// 1: numero di dimensioni dell'istogramma.
// dims: size dell'array.
// CV_HIST_ARRAY: tipo dell'istogramma.
// 1: i bins (barre) sono distribuiti uniformemente.
CvHistogram* hist02 = cvCreateHist(1, dims, CV_HIST_ARRAY,
                                   0, 1);

cvSetHistBinRanges( hist01, pthresh, 1);
// hist01: istogramma destinazione.
// pthresh: puntatore ai valori "limite" dell'array.
// 1: i bins sono distribuiti uniformemente.
```


Istogrammi

```
cvNamedWindow( wndname01, CV_WINDOW_AUTOSIZE );
cvNamedWindow( wndname02, CV_WINDOW_AUTOSIZE );

cvShowImage( wndname01, image02 );

// Calculated the histogram.
cvCalcHist( &image02, hist01, 0, 0 );
// immagine sorgente.
// puntatore all'istogramma.
// Clear flag = 0 : l'istogramma viene azzerato prima del calcolo.
// IplImage* mask=0
```

Istogrammi

```
// preludio al tracciamento dell'istogramma.
int i;
CvPoint pt1, pt2;
pt2.x=0;
pt2.y= (image01->height – 1) ;
pt1.x=0;
pt1.y=0;
cvSetZero(image03); // sfondo nero per l'immagine dell'istogramma
//copio l'istogramma 1 nell'istogramma 2.
cvCopyHist(hist01, &hist02);
// hist01: istogramma sorgente.
// &hist02: puntatore all'istogramma destinazione.
```


Istogrammi

```
float minVal[1];
float maxVal[1];
int minIdx[1];
int maxIdx[1];

// calcolo di minimo e massimo dell'istogramma
// per adattare l'istogramma alle dimensioni dell'immagine
cvGetMinMaxHistValue (hist02, minVal,
                      maxVal, minIdx, maxIdx);

float bins;
```

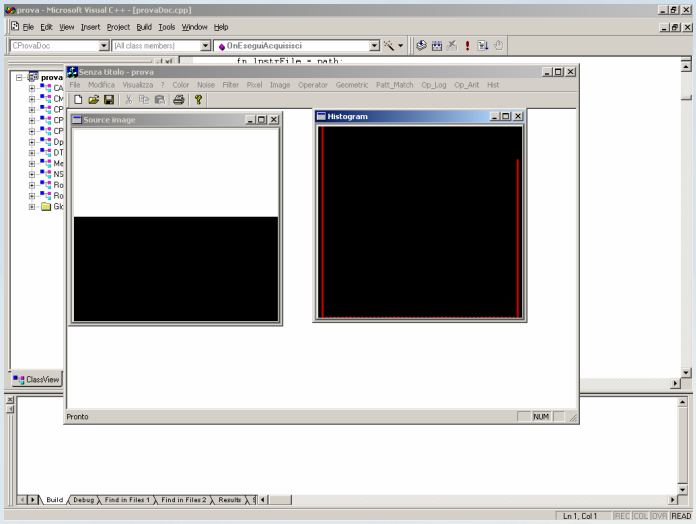
Istogrammi

```
//tracciamento dell'istogramma
for(i=0; i<DIMENSION_SIZE; i++)
{  pt1.x=pt1.x+5;
   // La funzione cvQueryHistValue_1D( )
   //  restituisce il valore del bin specificato.
   bins=cvQueryHistValue_1D(hist02,i);
   // normalizzazione dei "bin" alle dimensioni dell'immagine
   // (image03).
   int valore=int((image01->height)*(bins/maxVal[0]));
   pt1.y=image01->height - valore;
   pt2.x=pt2.x+5;
   cvLine(image03, pt1, pt2, CV_RGB(200,0,0), 2, 8);
   // CV_RGB(200,0,0): definisco un colore avente le
   // componenti R=200, G=0, B=0.
   // 2: spessore della linea, 8: connettività adottata.
}
```


Istogrammi

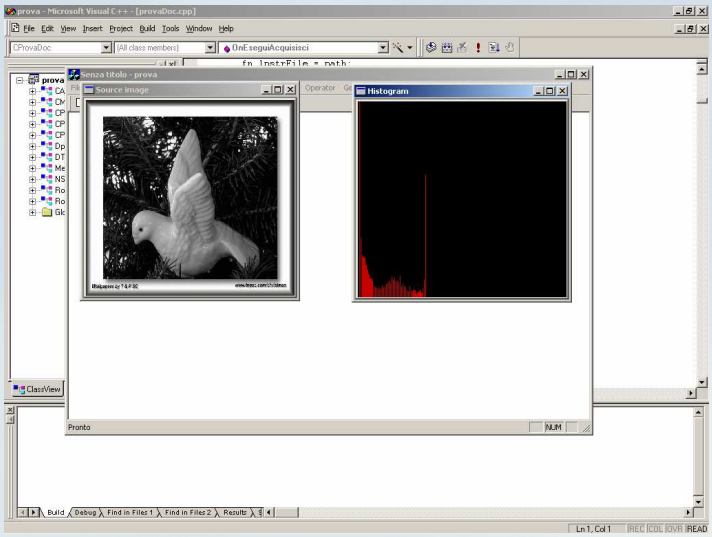
```
…
// show delle immagini e release della memoria
cvShowImage( wndname02, image03 );
cvWaitKey(0);
cvDestroyWindow(wndname01);
cvDestroyWindow(wndname02);
cvReleaseImage(&image01);
cvReleaseImage(&image02);
cvReleaseImage(&image03);
cvReleaseHist( &hist01 );
cvReleaseHist( &hist02 );
}
```

Istogrammi



parte 1.5 – OpenCV

Istogrammi



VisiLAB Computer Vision and Image Processing Lab - University of Messina

Istogrammi

```
// equalizzazione di un'immagine gray-level
void CProvaDoc::OnHistEqualize()
{  const int range = 256;
   IplLUT lut = { range+1, NULL,NULL,NULL,
   IPL_LUT_LOOKUP };
   IplLUT* plut = &lut;
   lut.key =(int*) malloc( sizeof(int)*(range+1) );
   lut.value =(int*) malloc( sizeof(int)*range );
   char wndname01[] = "Source image";
   char wndname02[] = "Equalize";
   char path[256];
   memset(path,0,256);
```


Istogrammi

```
…
//stabilisco il file da aprire
   OPENFILENAME fn;
   fn.lStructSize = sizeof(OPENFILENAME);
   fn.hwndOwner = NULL; fn.lpstrFilter = NULL;
   fn.lpstrFile = path;  fn.nMaxFile = 256;
   fn.lpstrFileTitle = NULL;  fn.lpstrInitialDir = NULL;
   fn.lpstrTitle = NULL;  fn.Flags = NULL;
   fn.lpstrDefExt = "bmp";  fn.hInstance = DLLhinst;
   fn.lpfnHook = NULL;  fn.lpstrCustomFilter = NULL;
   fn.lCustData = NULL;
   if(!GetOpenFileName(&fn))
         return ;
…
```

Istogrammi

```
…
IPLIMAGE image01 = cvLoadImage( path );
IplImage * image02 = cvCreateImage(
            cvSize(image01->width,image01->height),
                  IPL_DEPTH_8U, 1);
cvNamedWindow( wndname01, CV_WINDOW_AUTOSIZE );
cvNamedWindow( wndname02, CV_WINDOW_AUTOSIZE );

// Initialize the histogram levels
for(int i=0; i<=range; i++) lut.key[i] = i;
…
```

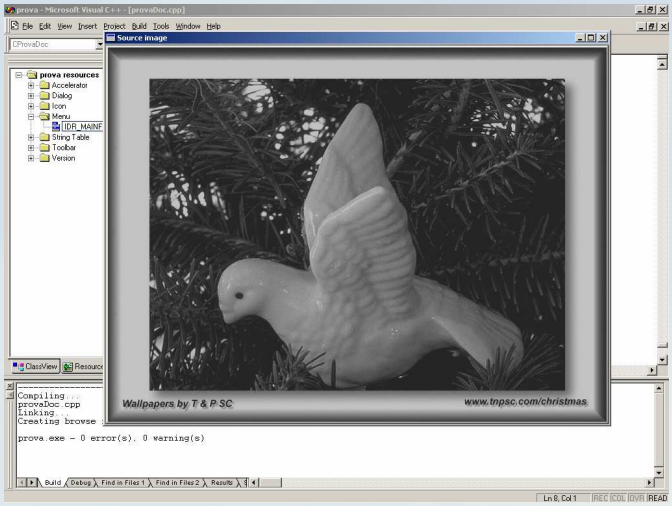

Istogrammi

```
…
iplColorToGray(image01,image02);

cvShowImage( wndname01, image02);

// Compute histogram
iplComputeHisto( image02, &plut );
// Equalize histogram = rescale range of image data
iplHistoEqualize( image02, image02, &plut );
cvShowImage( wndname02, image02);
cvWaitKey(0);
…
```

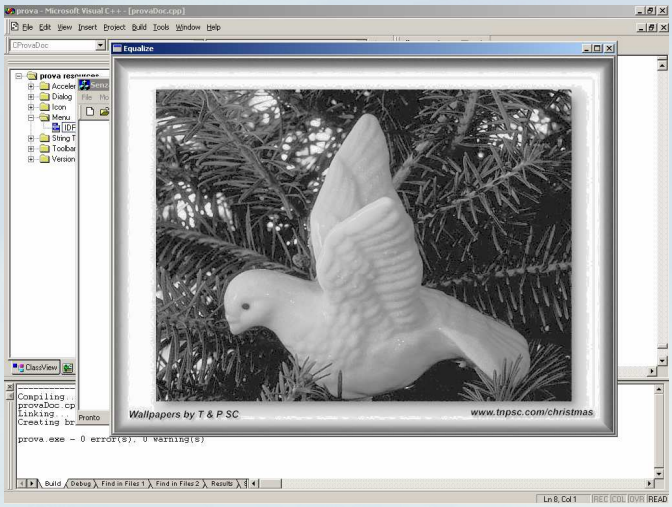
Istogrammi

```
…
cvDestroyWindow(wndname01);
cvDestroyWindow(wndname02);

cvReleaseImage(&image01);
cvReleaseImage(&image02);

if( lut.key ) free( lut.key );
if( lut.value ) free( lut.value );
}
```


Istogrammi



Istogrammi



NoiseGaussian

```
//introduco rumore gaussiano in un'immagine
void CProvaDoc::OnNoiseGaussian( )
{  IplImage  * src, * src_gray;
   char path[256];
   memset(path,0,256);//setto a 0 i primi 256 elementi di path
   //stabilisco il file da aprire
    OPENFILENAME fn;
    fn.lStructSize = sizeof(OPENFILENAME);
    fn.hwndOwner = NULL; fn.lpstrFilter = NULL;
    fn.lpstrFile = path;  fn.nMaxFile = 256;
    fn.lpstrFileTitle = NULL; fn.lpstrInitialDir = NULL;
    fn.lpstrTitle = NULL; fn.Flags = NULL;
    fn.lpstrDefExt = "bmp"; fn.hInstance = DLLhinst;
    fn.lpfnHook = NULL;  fn.lpstrCustomFilter = NULL;
    fn.lCustData = NULL;
```

NoiseGaussian

```
//carico l'immagine dal path scelto
src=cvLoadImage(path);
CvSize size_src=cvSize( src->width,src->height);
src_gray=cvCreateImage(size_src,src->depth,1);
iplColorToGray(src,src_gray);
cvNamedWindow("window ", CV_WINDOW_AUTOSIZE );
//apro una finestra denominata window
cvShowImage("window ",src_gray);
cvWaitKey(0);
```

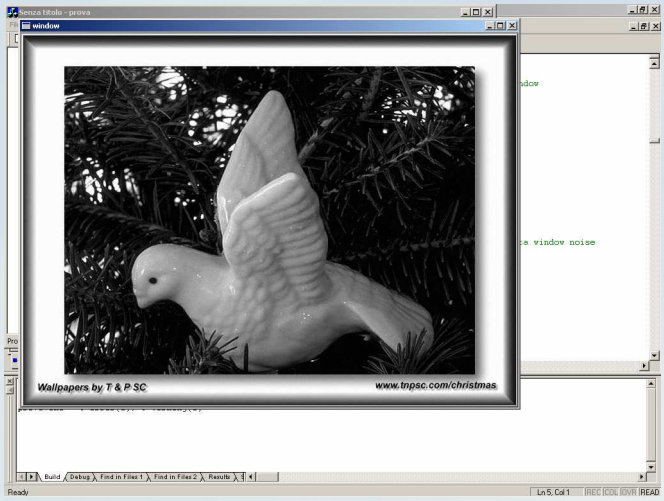

NoiseGaussian

```
…
cvDestroyWindow("window ");
//definisco noiseParam
IplNoiseParam noiseParam;
//uso  iplGaussianInit per inizializzare noiseParam
iplNoiseGaussianInit(&noiseParam,10,120,30);
//seme=10, media=120, deviazione-standard=30.
//introduco rumore gaussiano
iplNoiseImage(src_gray,&noiseParam);
…
```

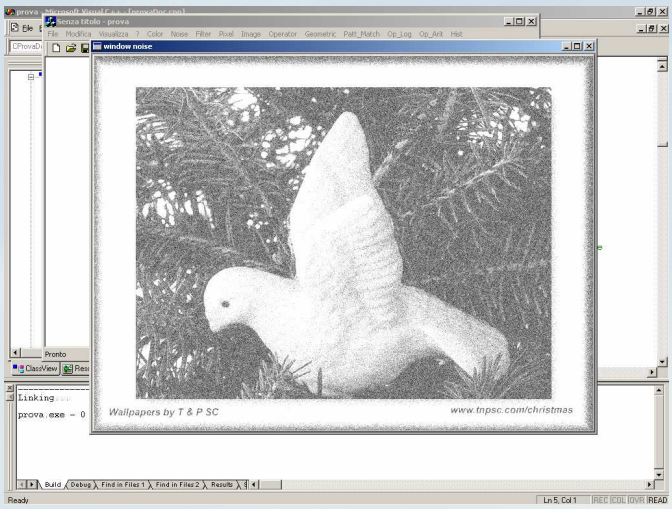
NoiseGaussian

```
…
//apro una finestra denominata window noise
cvNamedWindow("window noise",CV_WINDOW_AUTOSIZE);
cvShowImage("window noise",src_gray);
cvWaitKey(0);
cvDestroyWindow("window noise");
//rilascio la memoria allocata per le immagini
cvReleaseImage (&src);
cvReleaseImage (&src_gray);
}
```


NoiseGaussian



NoiseGaussian



NoiseUniform

```
//introduco rumore uniforme in un'immagine
void CProvaDoc::OnNoiseUniforme()
{  IplImage *src, * src_gray;
    char path[256];    memset(path,0,256);
   //stabilisco il file da aprire
   OPENFILENAME fn;
   fn.lStructSize = sizeof(OPENFILENAME);
   fn.hwndOwner = NULL; fn.lpstrFilter = NULL;
   fn.lpstrFile = path; fn.nMaxFile = 256;
   fn.lpstrFileTitle = NULL; fn.lpstrInitialDir = NULL;
   fn.lpstrTitle = NULL; fn.Flags = NULL;
   fn.lpstrDefExt = "bmp"; fn.hInstance = DLLhinst;
   fn.lpfnHook = NULL; fn.lpstrCustomFilter = NULL;
   fn.lCustData = NULL;
      if(!GetOpenFileName(&fn))  return ;
```

NoiseUniform

```
//carico l'immagine dal path scelto
src=cvLoadImage(path);
CvSize size_src=cvSize( src->width,src->height);
src_gray=cvCreateImage(size_src,src->depth,1);
iplColorToGray(src,src_gray);
cvNamedWindow("window ",1);
//apro una finestra denominata window
cvShowImage("window ",src_gray);
cvWaitKey(0);
cvDestroyWindow("window ");
```

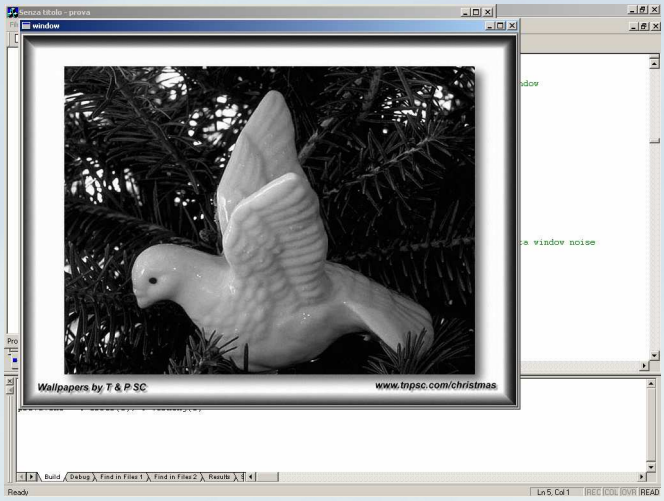

NoiseUniform

```
…
//definisco noiseParam
IplNoiseParam noiseParam;
//uso  iplNoiseUniformInit per inizializzare noiseParam
iplNoiseUniformInit(&noiseParam,10,50,100);
//seme=10 , rumore: low= 50 high=100.

//introduco rumore uniforme
iplNoiseImage(src_gray,&noiseParam);

//apro una finestra denominata window noise
cvNamedWindow("window noise", CV_WINDOW_AUTOSIZE);
cvShowImage("window noise", src_gray);
…
```

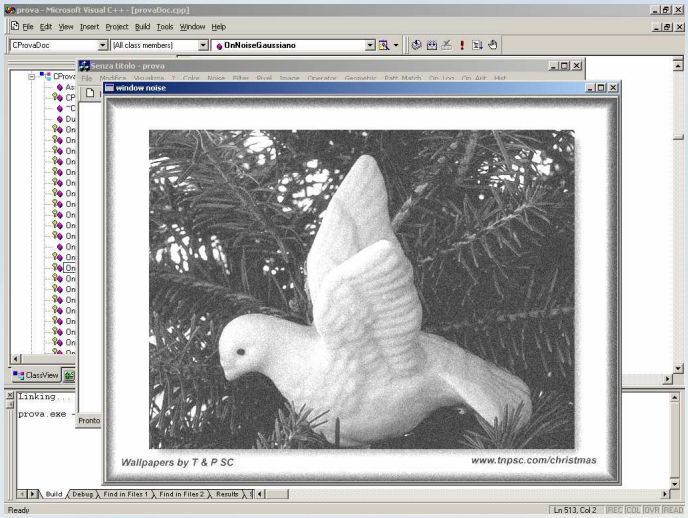
NoiseUniform

```
…
cvWaitKey(0);
cvDestroyWindow("window noise");

//rilascio la memoria allocata per le immagini
cvReleaseImage (&src);
cvReleaseImage (&src_gray);
}
```


NoiseUniform



NoiseUniform



Morphological Operations

Elementary morphological transformations

- Erosion
- Dilation

Advanced morphological transformations

- Open
- Close
- Gradient

Erosion

```
//applico un operatore di erosion su un'immagine binaria
void CProvaDoc::OnMorphErode()
{   IplImage *image01, *image02, *image03;
    IplConvKernel *st_elem;
    char path[256];
    memset(path,0,256);
    //stabilisco il file da aprire
    OPENFILENAME fn;
    fn.lStructSize = sizeof(OPENFILENAME);
    fn.hwndOwner = NULL; fn.lpstrFilter = NULL;
    fn.lpstrFile = path; fn.nMaxFile = 256;
    fn.lpstrFileTitle = NULL; fn.lpstrInitialDir = NULL;
    fn.lpstrTitle = NULL; fn.Flags = NULL;
```


Erosion

```
…
fn.lpstrDefExt = "bmp"; fn.hInstance = DLLhinst;
fn.lpfnHook = NULL; fn.lpstrCustomFilter = NULL;
fn.lCustData = NULL;

if(!GetOpenFileName(&fn))
        return ;

image01 = cvLoadImage( path );
image02 = cvCreateImage( cvSize(image01->width,
                        image01->height),IPL_DEPTH_8U,1);
image03 = cvCreateImage( cvSize(image01->width,
                        image01->height),IPL_DEPTH_8U,1);
…
```

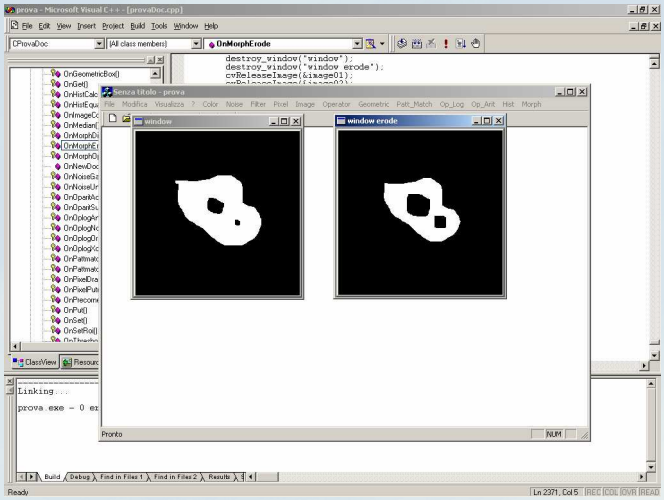
Erosion

```
…
iplColorToGray(image01,image02);
iplThreshold(image02, image02,127);

st_elem=cvCreateStructuringElementEx(3, 3, 1, 1,
                                     CV_SHAPE_RECT, NULL);
int iter=2;//numero di iterazioni per l'operazione di erosione
cvErode(image02, image03, st_elem, iter);
cvReleaseStructuringElement (&st_elem);
cvNamedWindow("window", CV_WINDOW_AUTOSIZE );
cvShowImage("window", image02);
cvNamedWindow("window erode",CV_WINDOW_AUTOSIZE );
cvShowImage("window erode", image03);
cvWaitKey(0);
…
```


Erosion

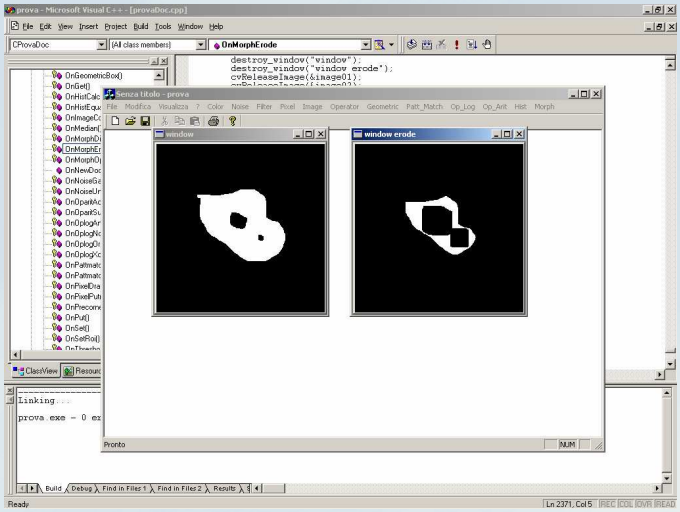
```
…
cvDestroyWindow("window");
cvDestroyWindow("window erode");
cvReleaseImage(&image01);
cvReleaseImage(&image02);
cvReleaseImage(&image03);
}
```

Erosion



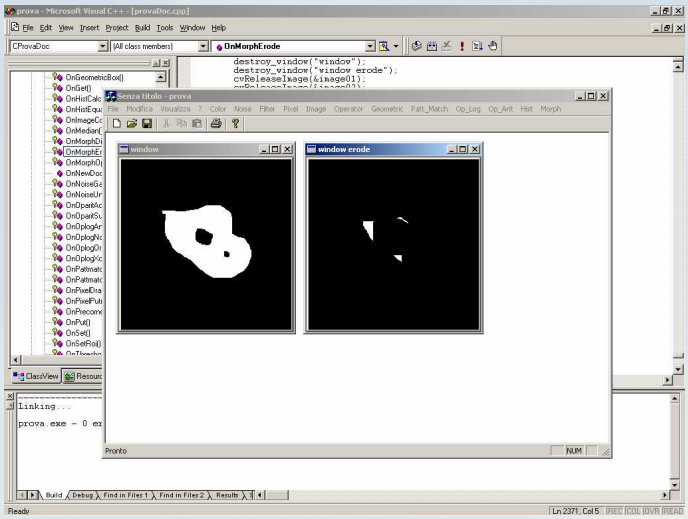
Numero iterazioni= 5.

Erosion



Numero iterazioni= 10.

Erosion



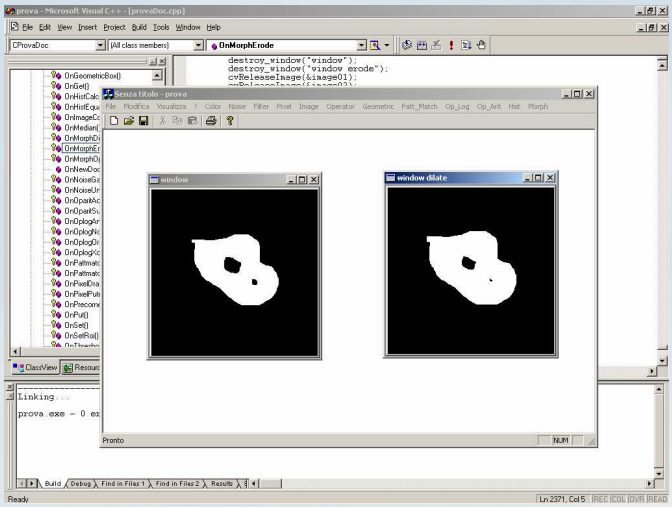
Numero iterazioni= 15.

Dilation

void cvDilate (IplImage* *pSrc*, IplImage* *pDst*,
IplConvKernel* *B*, int *iterations*);

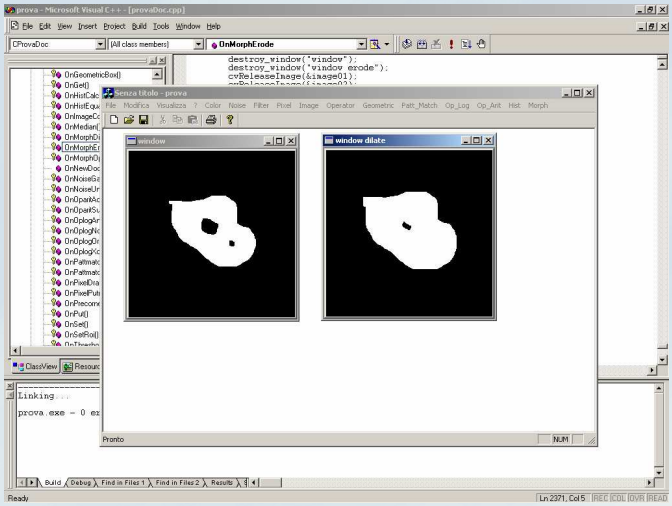
pSrc	Source image.
pDst	Destination image.
B	Structuring element used for dilation. If NULL, a 3x3 rectangular structuring element is used.
iterations	Number of times dilation is applied.

Dilation



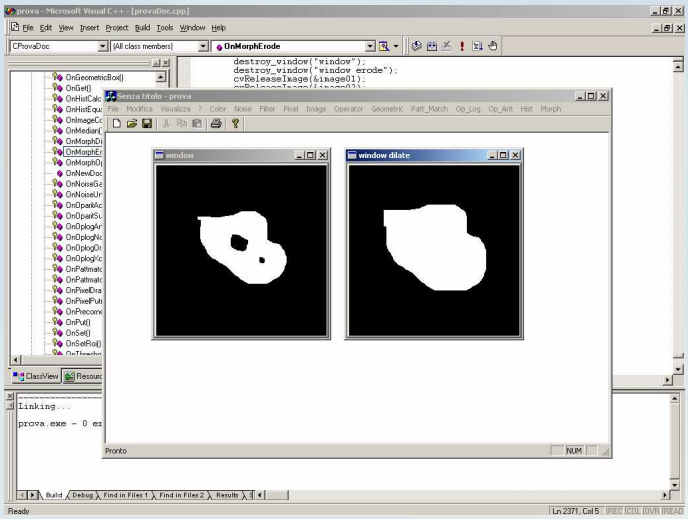
Numero iterazioni= 2.

Dilation



Numero iterazioni= 5.

Dilation



Numero iterazioni= 10.

MorphologyEx

void cvMorphologyEx (IplImage* *src*, IplImage* *dst*, IplImage* *temp*, IplConvKernel* *B*, CvMorphOp *op*, int *iterations*);

src Source image.

dst Destination image.

temp Temporary image, required in some cases.

B Structuring element.

MorphologyEx

op Type of morphological operation:

- CV_MOP_OPEN, opening;
- CV_MOP_CLOSE, closing;
- CV_MOP_GRADIENT, morphological gradient;
- CV_MOP_TOPHAT, top hat;
- CV_MOP_BLACKHAT, black hat.

iterations Number of times erosion and dilation are applied during the complex operation.

MorphologyEx

Opening

$$A \circ B = (A \ominus nB) \oplus nB;$$

Closing

$$A \bullet B = (A \oplus nB) \ominus nB;$$

MorphologyEx

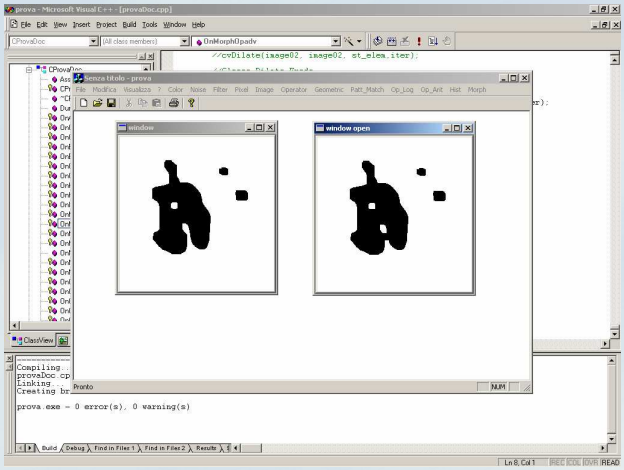
Gradient

$$Temp = A \Theta B;$$
$$Dest = A \oplus B;$$
$$Dest = Temp - Dest;$$

MorphologyEx

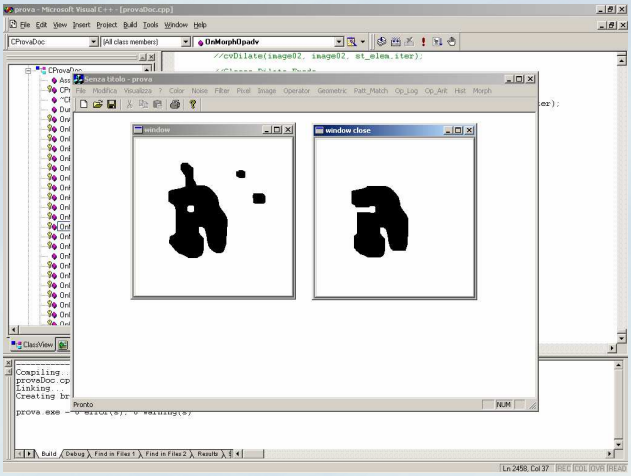
Test Image

Opening



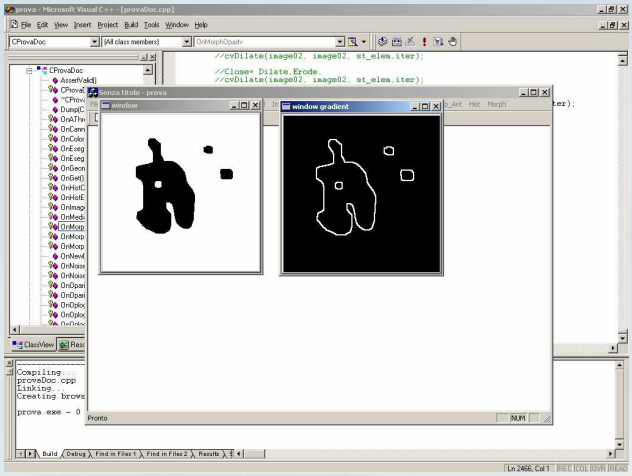
Numero Iterazioni= 3.

Closing



Numero Iterazioni= 10.

Gradient



Numero Iterazioni= 1.

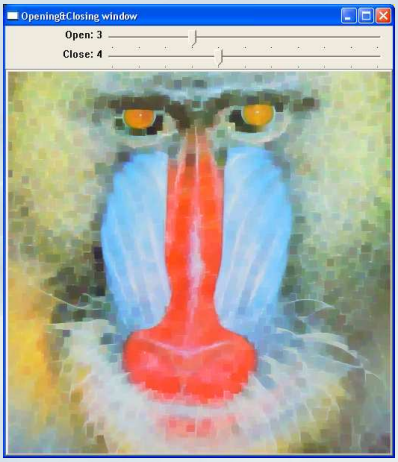
Esercizio

- Implementare le seguenti operazioni:
  - Opening;
  - Closing;
  - Boundary Extraction;
  - …

  Per un sommario di possibili morphological transformations vedi “R.C. Gonzalez and R.E. Woods. *Digital Image Processing* - pagg. 544-545”.

Programma

Nel programma morphology.c contenuto nella cartella sample di OpenCV sono usate le principali funzionalità di MM messe a disposizione da OpenCV.



Trasformazioni Geometriche Esempi

- Zoom
- Resize
- Rotation

Zoom

void iplZoom(IplImage* srcImage, IplImage* dstImage, int
int xDst, int xSrc, int yDst, int ySrc, int interpolate);

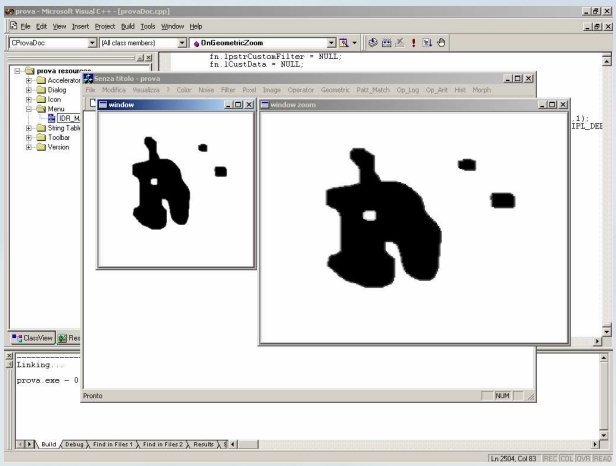
- srcImage The source image.
- dstImage The resultant image.
- xDst, xSrc, yDst, ySrc Positive integers specifying the fractions.

xDst/xSrc 1 and yDst/ySrc 1 - the factors
by which the *x* and *y* dimensions of the image's
ROI are changed. For example, setting
xDst= 2, xSrc = 1, yDst= 2, ySrc= 1
doubles the image size in each dimension to
increase the image area by a factor of four.

Zoom

- interpolate The type of interpolation to perform for resampling. Can be one of the following:
  - IPL_INTER_NN Nearest neighbor.
  - IPL_INTER_LINEAR Linear interpolation.
  - IPL_INTER_CUBIC Cubic interpolation.

Zoom



X-> 2:1, Y-> 3:2.

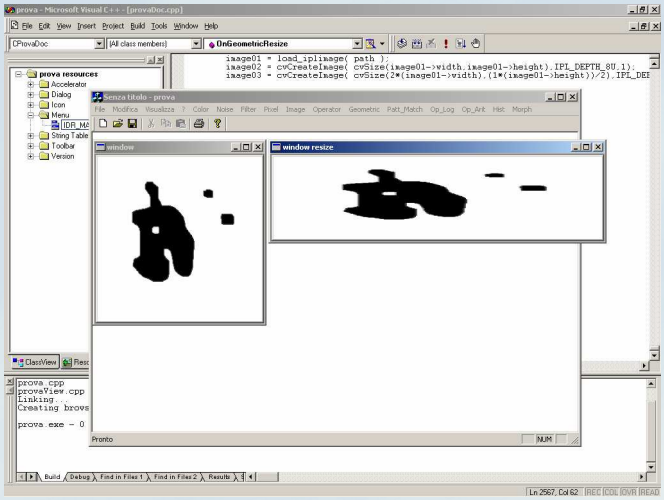
Resize

```
void iplResize(IplImage* srcImage, IplImage* dstImage,
        int xDst, int xSrc, int yDst, int ySrc, int interpolate);
```

Note:

The function iplResize differs from iplZoom in that it can increase one dimension of an image while decreasing the other dimension.

Resize



X-> 2:1, Y-> 1:2.

Rotation

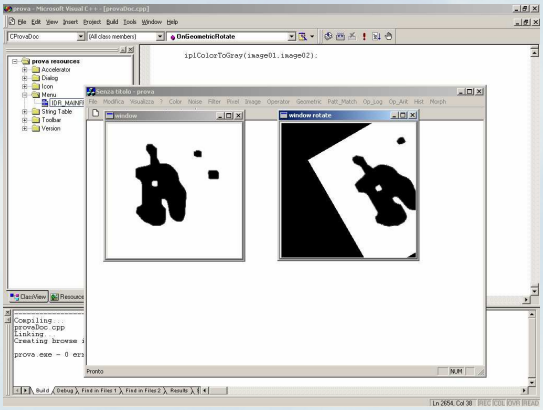
void iplRotate(IplImage* srcImage, IplImage* dstImage,
double angle, double xShift, double yShift, int interpolate);

Note:

The function iplRotate() rotates the source image srcImage by angle degrees around the origin (0,0) and shifts it by xShift and yShift along the *x*- and *y*-axis, respectively.

Rotation

iplRotate (image01 , image02, 30, 50, 70, IPL_INTER_NN);

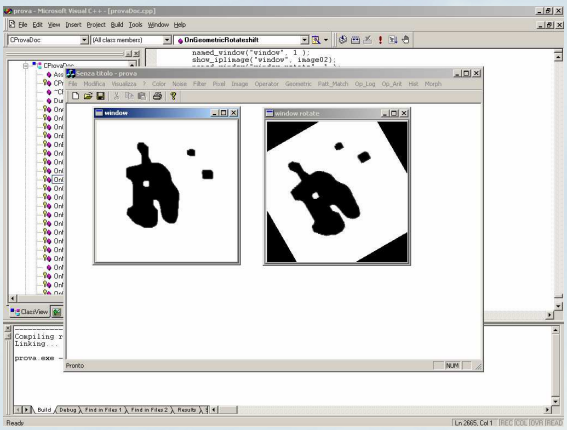


Angle =30, xShift=50, yShift=70.

Rotation - Center

iplRotateCenter (image01,image02,30, (image01->width)/2, (image01->height)/2, IPL_INTER_NN);

Angle=30.



xCenter= (image01->width)/2, yCenter= (image01->height)/2.