Applicazioni pratiche della visione artificiale

Parte 1.3 – OpenCV

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

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

GetPixel

```
//lettura del valore assunto da un pixel
void CProvaDoc::OnGet()
{IplImage *src, *src_gray;
 char path[]="C\\BW.bmp";
 src=cvLoadImage(path);
 CvSize size_src=cvSize( src->width,src->height);
 src_gray=cvCreateImage(size_src,src->depth,1);
 iplColorToGray(src,src_gray);//oppure cvCvtColor
 int X=10;//coordinate del pixel da leggere
 int Y=10;
 …
```

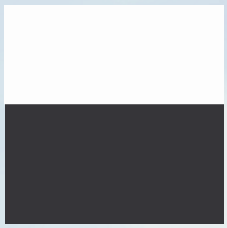
GetPixel

```
…
unsigned char pixel[1];
char pixelc[10];
//prendo il valore del pixel di coordinate (X,Y)
iplGetPixel(src_gray,Y,X,pixel);
//consultare il Reference Manual per accedere al pixel
//senza far uso delle IPL.
CString str="Il valore del pixel acquisito è: ";
int numero=(int)pixel[0];
//converto un intero in un array di caratteri
_itoa(numero,pixelc,10);
…
```


GetPixel

```
…
//concateno 2 stringhe
str=str+pixelc; //overloading dell'operatore "+"
//visualizzo in una message-box il contenuto di str
AfxMessageBox(str);
//rilascio la memoria allocata per le immagini
cvReleaseImage (&src);
cvReleaseImage (&src_gray);
}
```

GetPixel



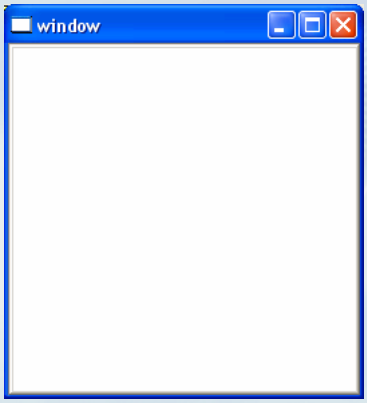
BW.bmp

Valore letto: 254

SetImage

```
//assegno un valore a tutti i pixel dell'immagine
void CProvaDoc::OnImageSet()
{       IplImage *src;  CvSize size_src=cvSize(256, 256);
        src=cvCreateImage(size_src,IPL_DEPTH_8U,1);
        //tutti i pixel di src sono posti a "255".
        CvScalar value=cvScalar(255); cvSet(src,value);
        //oppure iplSet(src,255);
        cvNamedWindow("window",CV_WINDOW_AUTOSIZE);
        cvShowImage("window",src);
        cvWaitKey(0);
        cvDestroyWindow("window");
        cvReleaseImage(&src);
}
```

SetImage



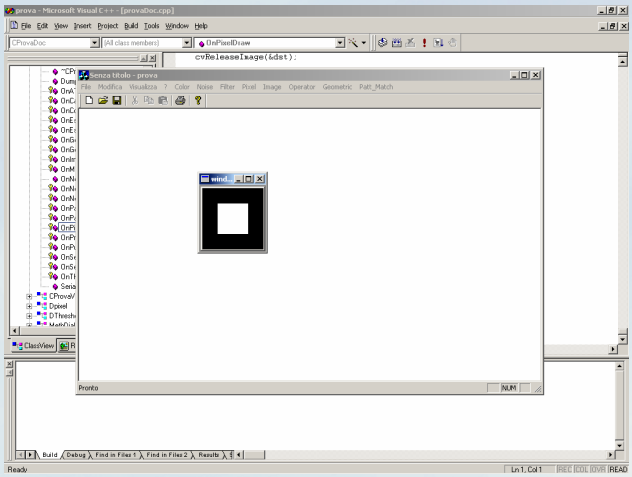
ImageDraw

```
// disegno un quadrato bianco su sfondo nero
void CProvaDoc::OnImageDraw()
{       IplImage *src
        CvSize size_src=cvSize(100, 100);
        src=cvCreateImage(size_src,IPL_DEPTH_8U,1);
        iplSet(src,0);
        CvRect roi_rect=cvRect( 25,25, 50, 50);
        cvSetImageROI (src,roi_rect);
        iplSet(src,255);
        …
```

ImageDraw

```
…
 roi_rect=cvRect( 0,0, 100, 100);
 cvSetImageROI (src, roi_rect);
 cvNamedWindow("window", CV_WINDOW_AUTOSIZE);
 cvShowImage("window",src);
 cvWaitKey(0);
 cvDestroyWindow("window");
 cvReleaseImage(&src);
}
```


ImageDraw



PutPixel

```
//traccio due righe “nere” nell’immagine
void CProvaDoc::OnPixelPutrow()
{   IplImage *  src;
    CvSize size_src=cvSize(300, 200);
    src=cvCreateImage(size_src,IPL_DEPTH_8U,1);
    //tutti i pixel dell’immagine sono posti a 255
    iplSet(src,255);
    cvNamedWindow("window",CV_WINDOW_AUTOSIZE);
    cvShowImage("window",src);
    cvWaitKey(0);
…
```


PutPixel

```
//fisso le righe da tracciare nell’immagine
int Y1=50;
int Y2=150;
unsigned char pixel[1];
pixel[0]=(unsigned char)0;
//traccio le righe
for(int i=0;i<src->width;i++)
iplPutPixel(src,i,Y1,pixel);// assegno il valore “pixel”
                            // al punto di coordinate (Y1,i)
for(i=0;i<src->width;i++)
iplPutPixel(src,i,Y2,pixel);
…
```

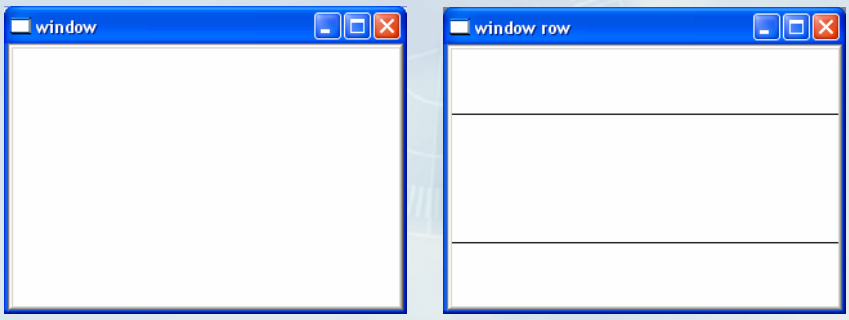
PutPixel

```
…
    cvNamedWindow("window row", CV_WINDOW_AUTOSIZE);
    cvShowImage("window row",src);

    cvWaitKey(0);

    cvDestroyWindow("window");
    cvDestroyWindow("window row");
    cvReleaseImage (&src);
}
```


PutPixel



And

```
//and logico tra 2 immagini
void CProvaDoc::OnOplogAnd()
{   IplImage *src1,*src2,*src_gray1, *src_gray2,*dst;
    char path1[]="C:\\and1.bmp";
    char path2[]="C:\\and2.bmp";

    src1=cvLoadImage(path1);
    src2=cvLoadImage(path2);
    //src1 & src2 sono immagini aventi le stesse dimensioni
    CvSize size_src=cvSize( src1->width,src1->height);
    src_gray1=cvCreateImage(size_src,IPL_DEPTH_8U,1);
    src_gray2=cvCreateImage(size_src,IPL_DEPTH_8U,1);
    …
```


And

```
…
//converto le immagini contenute in "src1" e "src2"
//a toni di grigio
iplColorToGray(src1,src_gray1);
//cvCvtColor(src1,src_gray1,CV_RGB2GRAY);
iplColorToGray(src2,src_gray2);
dst=cvCreateImage(size_src,IPL_DEPTH_8U,1);
//effettuo l'operazione di "and" logico tra le due
//immagini (a toni di grigio)
iplAnd(src_gray1,src_gray2,dst);
//cvAnd(src_gray1,src_gray2,dst); …
```

And

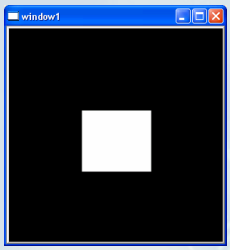
```
…
  //mostro a schermo le immagini sorgenti ed il risultato
 cvNamedWindow("window1",CV_WINDOW_AUTOSIZE);
  cvShowImage("window1",src1);
  cvNamedWindow("window2", CV_WINDOW_AUTOSIZE);
  cvShowImage("window2",src2);
  cvNamedWindow("window results", CV_WINDOW_AUTOSIZE);
  cvShowImage("window results",dst);
  cvWaitKey(0);
…
```

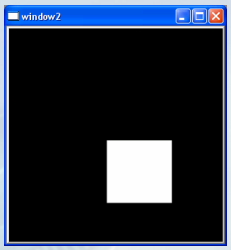

And

```
        …
         //distruggo le window aperte e libero la memoria
         // allocata in precedenza
        cvDestroyWindow("window1");
        cvDestroyWindow("window2");
        cvDestroyWindow("window results");
        cvReleaseImage(&src1);
        cvReleaseImage(&src2);
        cvReleaseImage(&src_gray1);
        cvReleaseImage(&src_gray2);
        cvReleaseImage(&dst);
}
```

And

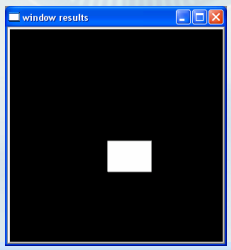
AND1





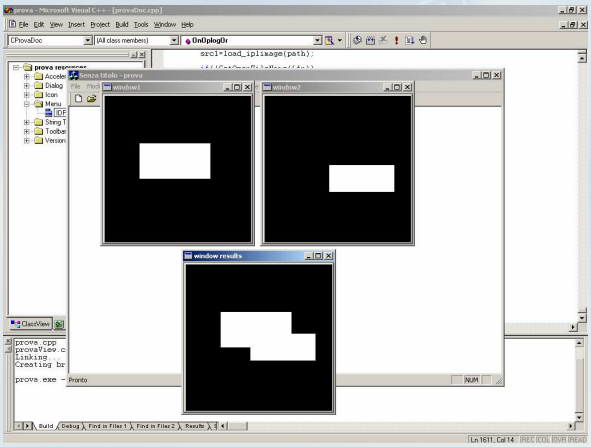
AND2

RESULTS



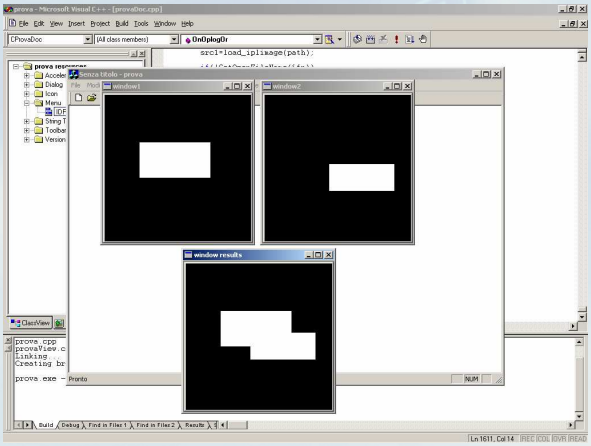
Ulteriori operazioni logiche

ø void iplOr(IplImage* srcImageA, IplImage* srcImageB, IplImage* dstImage);



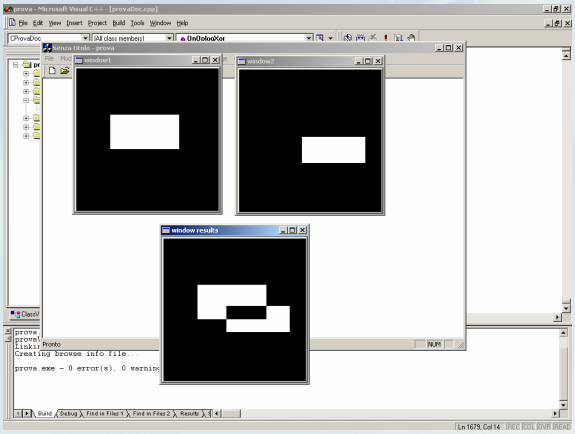
Ulteriori operazioni logiche

ø void cvOr(const CvArr* A, const CvArr* B, CvArr* C, const CvArr* mask=0);



Ulteriori operazioni logiche

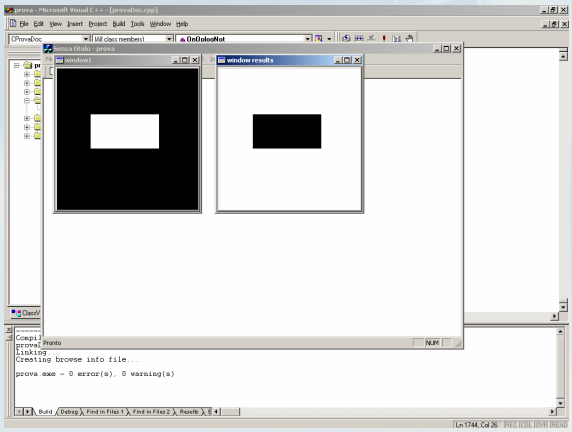
ø void iplXor(IplImage* srcImageA, IplImage* srcImageB, IplImage* dstImage);//oppure ***cvXor***



Ulteriori operazioni logiche

ø void iplNot(IplImage* srcImage, IplImage* dstImage);

//oppure ***cvNot***



SubImage

```
//differenza tra due immagini
void CProvaDoc::OnOparitSub()
{       IplImage *src1,*src2, *src_gray1, *src_gray2, *diff;
        char path1[]="C:\\rect2.bmp";
        char path2[]="C:\\rect1.bmp";
        src1=cvLoadImage(path1);
        src2=cvLoadImage(path2);
        //si fa uso di immagini aventi le stesse dimensioni
        CvSize size_src=cvSize( src1->width,src1->height);
        src_gray1=cvCreateImage(size_src,IPL_DEPTH_8U,1);
        src_gray2=cvCreateImage(size_src,IPL_DEPTH_8U,1);
…
```

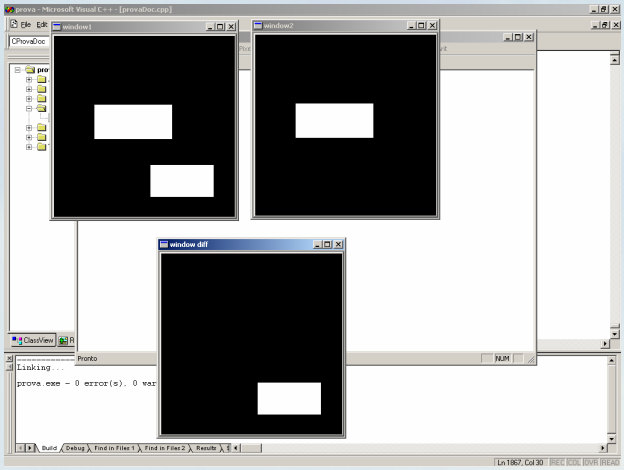
SubImage

```
…
iplColorToGray(src1,src_gray1);
iplColorToGray(src2,src_gray2);
diff=cvCreateImage(size_src,IPL_DEPTH_8U,1);
//diff=src_gray1- src_gray2;
iplSubtract(src_gray1,src_gray2, diff);
//cvSub(src_gray1, src_gray2, diff);
cvNamedWindow("window1",CV_WINDOW_AUTOSIZE);
cvShowImage("window1",src1);
cvNamedWindow("window2", CV_WINDOW_AUTOSIZE);
cvShowImage("window2",src2);
cvNamedWindow("window diff", CV_WINDOW_AUTOSIZE);
cvShowImage("window diff",diff); …
```


SubImage

```
…
        cvWaitKey(0);
        cvDestroyWindow("window1");
        cvDestroyWindow("window2");
        cvDestroyWindow("window diff");
        cvReleaseImage(&src1);
        cvReleaseImage(&src2);
        cvReleaseImage(&src_gray1);
        cvReleaseImage(&src_gray2);
        cvReleaseImage(&diff);
}
```

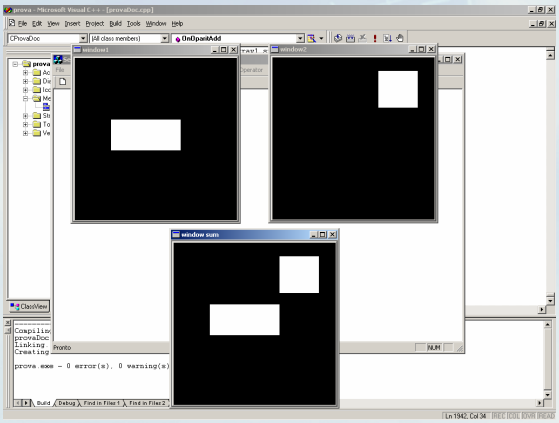
SubImage



AddImage

void iplAdd(IplImage* srcImageA, IplImage* srcImageB,
IplImage* dstImage);//***cvAdd***

dstImage = srcImageA + srcImageB// operazione compiuta



Ulteriori operazioni aritmetiche

Ø void iplMultiply(IplImage* srcImageA, IplImage* srcImageB, IplImage* dstImage);//***cvMul***

Per definizione:

dst_pixel= srcA_pixel* srcB_pixel;

Ø void iplMultiplyScale(IplImage* srcImageA, IplImage* srcImageB, IplImage* dstImage);

Per definizione:

dst_pixel= srcA_pixel* srcB_pixel/ max_val;

//max_val è il valore massimo rappresentabile nei formati //adottati.

Ulteriori operazioni aritmetiche

Ø void cvMul(const CvArr* A, const CvArr* B, CvArr* C, double scale=1);

Ø The function cvMul calculates per-element product of two arrays:

Ø C(I)=scale·A(I)·B(I)

N.B.: All the arrays must have the same type, and the same size (or ROI size).

Ulteriori operazioni aritmetiche

- void cvDiv(const CvArr* A, const CvArr* B, CvArr* C, double scale=1);
- The function cvDiv divides one array by another:
  - C(I)=scale·A(I)/B(I), if A!=NULL
  - C(I)=scale/B(I), if A=NULL

  N.B.: All the arrays must have the same type, and the same size (or ROI size).

Ulteriori operazioni matematiche

- void cvMax(const CvArr* A, const CvArr* B, CvArr* C);

  The function cvMax calculates per-element maximum of two arrays:
- C(I)=max(A(I), B(I))

  N.B.: All the arrays must have a single channel, the same data type and the same size (or ROI size).

  Esiste anche la funzione ***cvMin***.

Ulteriori operazioni matematiche

- void cvMaxS(const CvArr* A, const CvArr* B, CvArr* C);
- The function cvMaxS calculates per-element maximum of array and scalar:
  - C(I)=max(A(I), S)

  N.B: All the arrays must have a single channel, the same data type and the same size (or ROI size).

  Esiste anche la funzione ***cvMinS***.

Conversione di formato

- void cvConvertScale(const CvArr* A, CvArr* B, double scale=1, double shift=0);
- cvConvertScale copies one array to another with optional scaling, which is performed first, and/or optional type conversion, performed after:
- B(I)=A(I)*scale + (shift, shift, ...)

  N.B.: All the channels of multi-channel arrays are processed independently.

Altri esempi

- Ulteriori esempi (*open*) sulle funzioni introdotte in questa presentazione (e su molte altre) si trovano nel tutorial “Tutorial.IPL”:
  - *…\Ipl\examples\Tutorial.IPL\index.html*
- E’ possibile provare le potenzialità delle IPL attraverso la demo “Image Editor Demo” :
  - Esegui il file …\Ipl\examples\ipledit\ipledit.bat (consente all’applicazione di trovare le *dll* necessarie)
  - Esegui il file …\Ipl\examples\ipledit\ipledit.exe

IPLedit