Chapter 2: OpenCV

Andre L. C. Barczak

Computer Science
Massey University, Albany

Contents

Introduction to OpenCV

- Intel researchers.
- Freely available to different platforms (Unix, Windows, Android)
- Works with C/C++ (and Python, but we only use C/C++)
- The CV lab uses Linux.
- You can use OpenCV under Windows, but you have to sort out installation issues...
- Assignments: check your code in the lab to make sure they compile in the same version of GCC and OpenCV
- many changes to current version 3.4.5 (do not use version 4 for assignments)

Documentation

- http://opencv.org
- pdf file (available from the same places)
- there is also a book: “Learning OpenCV” (O’Reilly)
- ... just google for information...

Editing programs

- vi, ... etc etc
- Codeblocks (with a GUI and autocompletion)
- *.c or *.cpp
- Example of Include for OpenCV with C:
  #include cv.h (and cxcore.h)
  #include highgui.h (for GUI functions)
- Include for OpenCV with C++:
  #include "opencv2/opencv.hpp" (and other files)
  using namespace cv;

Compile and run

- gcc -o executable_name program_name.c
- g++ -o executable_name program_name.cpp
- Need to link shared libraries, e.g.:
    - g++ -o executable_name program_name.cpp -lopencv_core -lopencv_highui
- or use pkg-config:
  g++ -o executable_name program_name.cpp
  `pkg-config - -libs - -cflags opencv `
- To run type: ./executable_name

Shared libraries

- Where "-lopencv_core" comes from? This is to link a shared library
- There is a file called libopencv_core.so (or *.a) somewhere
- OpenCV >=3.0:
  -lopencv_core -lopencv_highgui -lopencv_imgproc
  -lopencv_legacy -lopencv_imgcodecs -lopencv_videoio
- Where to find these library files? Check the directory: /usr/local/lib

IplImage data stucture

- int nChannels;
- int depth;
- int width;
- int height;
- char*imageData;

Create IplImage

Listing 1: example

```
#include cv.h
//Declare IplImage pointer
cvCreateImage(CvSize size,int depth,int channels);
IplImage *image=0;
//for a single channel, 8 bits per pixel
image=cvCreateImage(cvSize(10,10),IPL_DEPTH_8U,1);
//or for a 3 channels, 8 bits/pix
image=cvCreateImage(cvSize(10,10),IPL_DEPTH_8U,3);
```

Accessing Pixels: method 1

```
...
uchar* pixel;
for (y=0;y<img->height;y++){
   for (x=0;x<img->width;x++){
      if(y == x){
         pixel=&((uchar*)(img->imageData+img->
            widthStep*y))[x];
         *pixel=255;
         //pixel[0]=255;// alternative
      }
   }
}
...
```

Accessing Pixels: method 2 (preferred)

```
#define pixel(image,x,y) ((uchar *)(image->imageData + (y)*image->widthStep))[(x)*image->nChannels]

...

if( (img = cvLoadImage( filename, 1)) == 0 )
  return -1;
  for (y=0;y<img->height;y++){
    for (x=0;x<img->width;x++){
      if(y == x) pixel(img,x,y)=255;
    }
  }
...
```

BGR Pixels: method 1

```
...
uchar* colourpixels;
...
for (y=0;y<img->height;y++){
   for (x=0;x<img->width;x++){
      colourpixels = &((uchar*)(img->imageData+img->
          widthStep*y))[x*3];
      if(y == x){
         colourpixels[0]=255;//blue
         colourpixels[1]=0;//green
         colourpixels[2]=0;//red
      }
   }
}
...
```

BGR Pixels: method 2

```
#define pixelB(image,x,y) ((uchar *)(image->imageData + (y)*image->widthStep))[(x)*image->nChannels]

#define pixelG(image,x,y) ((uchar *)(image->imageData + (y)*image->widthStep))[(x)*image->nChannels+1]

#define pixelR(image,x,y) ((uchar *)(image->imageData + (y)*image->widthStep))[(x)*image->nChannels+2]
```

BGR Pixels: method 2

```
uchar* colorpixels;
...
    for (y=0;y<img->height;y++){
        for (x=0;x<img->width;x++){
           if(y == x){
              pixelB(img,x,y)=255;
              pixelG(img,x,y)=255;
              pixelR(img,x,y)=255;
           }
...
```

Mat

Mat is a proper class for matrices (images are matrices...)
To create a 100 x 100 image greyscale:

```
Mat myimage;
myimage.create(100,100, CV_8UC1);
myimage.create(100,100, CV_8UC1,0);
```

Many methods available for copying, initializing, matrix operations etc
One common method to access matrices:

```
myimage.at<unsigned char>(i,j)
```

Macros for Mat structure with OpenCV3.0

```
//macros for Mat structures
#define MpixelB(image,x,y) ( (uchar *) ( ((image).data) + (y)*((image).step) ) ) [(x)*((image).channels())]

#define MpixelG(image,x,y) ( (uchar *) ( ((image).data) + (y)*((image).step) ) ) [(x)*((image).channels())+1]

#define MpixelR(image,x,y) ( (uchar *) ( ((image).data) + (y)*((image).step) ) ) [(x)*((image).channels())+2]
```

Using Macro for Mat

```
image = imread( argv[2], 1);
// Create the output image
image2.create(image.size(),CV_8UC3);
for (int x=0;x<image.cols;x++){
  for (int y=0;y<image.rows;y++){
    if(x == y){
      MpixelB(image2,x,y)=255;
      MpixelG(image2,x,y)=255;
      MpixelR(image2,x,y)=255;
    }
    else{
      MpixelB(image2,x,y)=MpixelB(image,x,y);
      MpixelG(image2,x,y)=MpixelG(image,x,y);
      MpixelR(image2,x,y)=MpixelR(image,x,y);
    }
  }
}
```

highgui

- simple GUI (no buttons, only mouse and trackbars)
- I/O: imread, imwrite, imshow
- stop and/or wait: cvWaitKey() and waitKey() (0 for infinite)
- slidebars (trackbars) are part of a window
- some examples...

imread() and imwrite()

```
Mat imagecolor, imagegrey;
imagecolor=imread(argv[1],1);//load as 3 channels
imagegrey=imread(argv[1],0);//load as 1 channel
imwrite("grey.jpg", imagegrey);
```

nameWindow(), imshow(), waitKey()

```
Mat image = imread(filename,1);
if(image.data==NULL) {exit(0);} //failed loading image
//create window
namedWindow("Window_number_1",1);//1 is for fixed size, 0 for resizable
//show image in window
imshow("Window_number_1",image);
//stop until press a key
waitKey(0);//zero is infinite time, otherwise msec
```

trackbars()

```
main(){
//always create this first
namedWindow("Show results",0);
//create trackbar named "switch"
//at the window above
//markerclick receives the values
//15 means how many values the trackbar have
//a call back function (more next...)
createTrackbar( "switch", "Show results",
      &markerclick, 15,  callback_trackbar_click );
//set initial position to zero
setTrackbarPos("switch","Show results",0);
...
}
```

callback functions

```
...
void callback_trackbar_click(int click, void *
    object){
  markerclick=click;
  if (markerclick==0){
    imshow("Show results",image1);
  }
  else {
    imshow("Show results",image2);
  }
}
...
```

using mouse callbacks

```
int main( int argc, char** argv )
{
  image1 = imread( argv[1]);
  namedWindow("Show results",0);
  //mouse will act on "Show restults" window
  setMouseCallback("Show results", on_mouse_example
      , NULL);
  //when showing and waiting, the mouse can be used
  imshow("Show results",image1);
  waitKey(0);
  return 0;
}
```

the mouse callback

```
void on_mouse_example(int event, int x, int y, int
    flag, void* obj)
{
  cout << "position " << x << " " << y << endl;
  if(event==EVENT_LBUTTONDOWN)//left click
  {
    cout << "clicked left button at " << x << " "
       << y << endl;
    pixel(image1,x,y)=255;
    imshow("Show results",image1);
  }
}
```

Creating Matrices and Vectors

Declaring and instantiating Mat is very easy.

```
Mat A(100,100,CV_8C1,0);
```

One can also create matrices preloaded with other values, and even with every element pre-specified.

```
//a 10x10 matrix initialised to 5.0 (floating point elements)
Mat A(10,10,CV_32F, Scalar(5));

Mat B(10,10,CV_32F);//same as above, with the Scalar
B = Scalar(5); //after the instantiation

//a 10x10 matrix initialised to 1 (uchar)
Mat C = Mat::ones(10,10,CV_8U);

//a 10x10 matrix initialised to 0 (uchar)
Mat D = Mat::zeros(10,10,CV_8U);
```

Initialising matrices

```
float a[2][3] = {{1,2,3}, {4,5,6}};
Mat A = Mat(2, 3, CV_32FC1, a);
cout << "A = " << A << endl;
```

```
A = [1, 2, 3;
 4, 5, 6]
```

```
Mat IDENTITY = (Mat_<float>(2,2) << 1,0,0,1);
cout << "IDENTITY = " << IDENTITY << endl;
```

```
IDENTITY = [1, 0;
 0, 1]
```

Matrix Operations

```
float a[2][3] = {{1,2,3}, {4,5,6}};
float b[2][3] = {{7,8,9}, {0,1,2}};
float c[3][2] = {{7,8},{9,0},{1,2}};
Mat A = Mat(2, 3, CV_32FC1, a);
Mat B = Mat(2, 3, CV_32FC1, b);
Mat C = Mat(3, 2, CV_32FC1, c);
Mat RES;
RES = A + B;
cout << "A + B = " << RES << endl << endl;
RES = A * C;
cout << "A * C = " << RES << endl << endl;
```

```
A + B = [8, 10, 12;
 4, 6, 8]

A * C = [28, 14;
 79, 44]
```

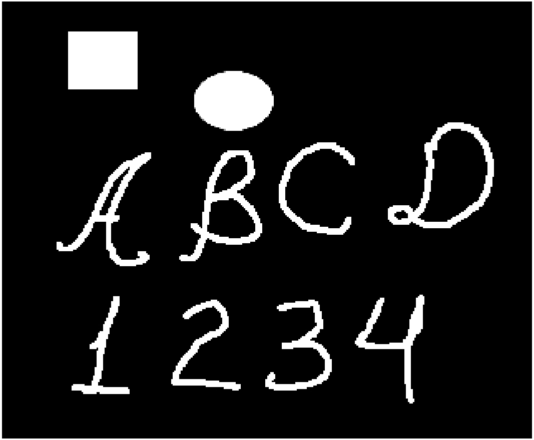
Other matrix operations: transpose, det, invert

```
transpose(A,RES);
cout << "Transpose = " << RES <<
    endl << endl;

float d[2][2] = {{1,2}, {2,5}};
Mat D = Mat(2, 2, CV_32FC1, d);
float det=determinant(D);
cout << "Determinant = " << det <<
    endl<<endl;

Mat INV;
invert(D,INV);
cout << "INV = " << INV << endl <<
    endl;
//Check inversion:
RES = D * INV;
cout << "RES = " << RES << endl <<
    endl;
```

```
Transpose =
    [1, 4;
 2, 5;
 3, 6]

Determinant
    = 1

INV = [5,
    -2;
 -2, 1]

RES = [1, 0;
 0, 1]
```

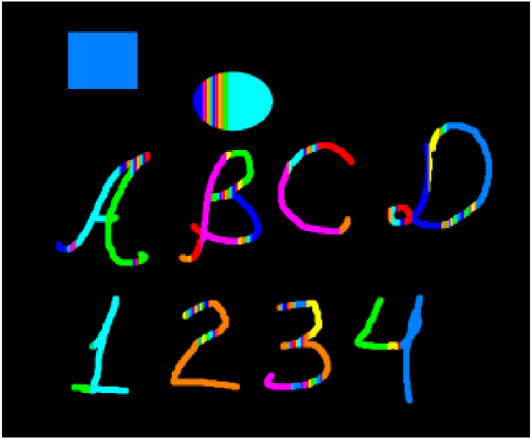
An arbitrary image

Consider this binary image:



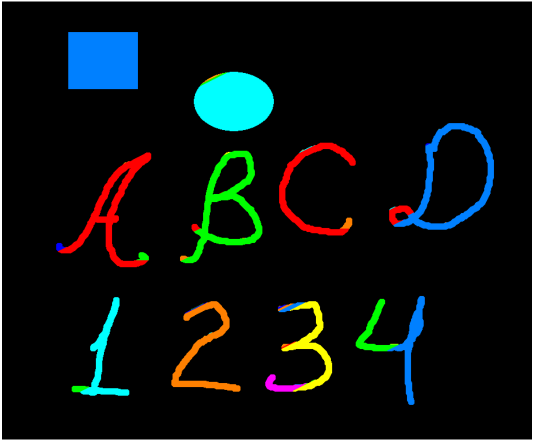
Step 1

Step 1 of a simple connectivity algorithm would show:



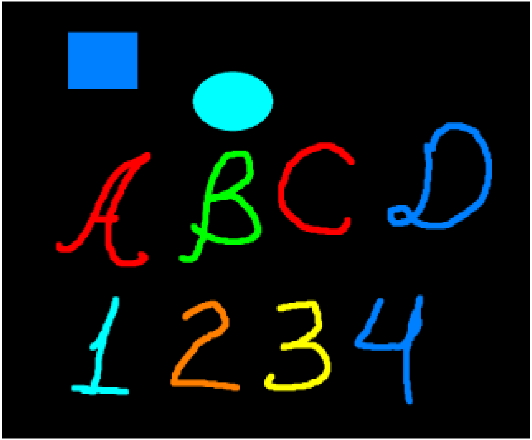
An arbitrary image

Coalescing part of the sets:



An arbitrary image

What the final result should be:



Algorithm 1

Require: a **binary** image $i(x, y)$

```
declare a vector of sets SET[ ]
declare integers counter = −1, s1, s2
declare A[i.width][i.height] {a 2D vector or mat initialised to -1}
for y = 1 to i.height do
    for x = 1 to i.width do
        if ( i(x, y) ≠ 0 ) then
            if (i(x − 1, y) ≠ 0 OR i(x, y − 1) ≠ 0) then
                s1 = A[x − 1][y]
                s2 = A[x][y − 1]
                if (s1 ≠ −1) then
                    i(x, y) → SET[s1] {insert point i(x, y) into SET[s1]}
                    A[x][y] = s1
                end if
                if (s2 ≠ −1) then
                    i(x, y) → SET[s2]
                    A[x][y] = s2
                end if
                if ((s1 ≠ s2) AND (s1 ≠ −1) AND (s2 ≠ −1)) then
                    SET[s1] ∪ SET[s2] {Union}
                    Reset all points of A(x, y) belonging to SET[s2] to s1
                    empty SET[s2]
                end if
            else
                counter = counter + 1
                Create new set SET[counter]
                i(x, y) → SET[counter]
                A[x][y] = counter
            end if
        end if
    end for
end for
```

Connected pixels: algorithm 1 example

Initializing ...

i(x,y)

0	0	0	0	0
0	1	1	1	0
1	1	1	0	0
0	0	0	0	1
0	1	0	1	1

A(x,y)

-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

SET[ ]

Connected pixels: algorithm 1 example

Step 1:

i(x,y)

0	0	0	0	0
0	1	1	1	0
1	1	1	0	0
0	0	0	0	1
0	1	0	1	1

A(x,y)

-1	-1	-1	-1	-1
-1	0	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

SET[0]={(1,1)}

Connected pixels: algorithm 1 example

Step 2:

i(x,y)

0	0	0	0	0
0	1	1	1	0
1	1	1	0	0
0	0	0	0	1
0	1	0	1	1

A(x,y)

-1	-1	-1	-1	-1
-1	0	0	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

SET[0]={(1,1), (2,1)}

Connected pixels: algorithm 1 example

Step 3:

i(x,y)

0	0	0	0	0
0	1	1	1	0
1	1	1	0	0
0	0	0	0	1
0	1	0	1	1

A(x,y)

-1	-1	-1	-1	-1
-1	0	0	0	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

SET[0]={(1,1),
(2,1),(3,1)}

Connected pixels: algorithm 1 example

Step 4:

i(x,y)

0	0	0	0	0
0	1	1	1	0
1	1	1	0	0
0	0	0	0	1
0	1	0	1	1

A(x,y)

-1	-1	-1	-1	-1
-1	0	0	0	-1
1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

SET[0]={(1,1), (2,1),(3,1)}

SET[1]={(0,2)}

Connected pixels: algorithm 1 example

Step 5:

i(x,y)

0	0	0	0	0
0	1	1	1	0
1	1	1	0	0
0	0	0	0	1
0	1	0	1	1

A(x,y)

-1	-1	-1	-1	-1
-1	0	0	0	-1
1	0	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

SET[0]={(1,1),
(2,1),(3,1),(1,2)}

SET[1]={(0,2),(1,2)}

UNION: sets [0] and [1]

Connected pixels: algorithm 1 example

Step 6:

i(x,y)

0	0	0	0	0
0	1	1	1	0
1	1	1	0	0
0	0	0	0	1
0	1	0	1	1

A(x,y)

-1	-1	-1	-1	-1
-1	0	0	0	-1
0	0	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

SET[0]={(1,1), (2,1), (3,1),(0,2),(1,2)}

SET[1]={}

Connected pixels: algorithm 1 example

Step 7:

i(x,y)

0	0	0	0	0
0	1	1	1	0
1	1	1	0	0
0	0	0	0	1
0	1	0	1	1

A(x,y)

-1	-1	-1	-1	-1
-1	0	0	0	-1
0	0	0	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

SET[0]={(1,1), (1,2), (1,3),(2,0),(2,1),(2,2)}

SET[1]={}

Connected pixels: algorithm 1 example

Step 8:

i(x,y)

0	0	0	0	0
0	1	1	1	0
1	1	1	0	0
0	0	0	0	1
0	1	0	1	1

A(x,y)

-1	-1	-1	-1	-1
-1	0	0	0	-1
0	0	0	-1	-1
-1	-1	-1	-1	2
-1	-1	-1	-1	-1

SET[0]={(1,1), (2,1), (3,1),(0,2),(1,2),(2,2)}

SET[1]={}

SET[2]={(3,4)}

Connected pixels: algorithm 1 example

Step 9:

i(x,y)

0	0	0	0	0
0	1	1	1	0
1	1	1	0	0
0	0	0	0	1
0	1	0	1	1

A(x,y)

-1	-1	-1	-1	-1
-1	0	0	0	-1
0	0	0	-1	-1
-1	-1	-1	-1	2
-1	3	-1	-1	-1

SET[0]={(1,1), (2,1),(3,1)
(0,2),(1,2),(2,2)}
SET[1]={}
SET[2]={(4,3)}
SET[3]={(1,4)}

Connected pixels: algorithm 1 example

Step 10:

i(x,y)

0	0	0	0	0
0	1	1	1	0
1	1	1	0	0
0	0	0	0	1
0	1	0	1	1

A(x,y)

-1	-1	-1	-1	-1
-1	0	0	0	-1
0	0	0	-1	-1
-1	-1	-1	-1	2
-1	3	-1	4	-1

SET[0]={(1,1), (2,1), (3,1),(0,2),(1,2),(2,2)}

SET[1]={}

SET[2]={(4,3)}

SET[3]={(1,4)}

SET[4]={(3,4)}

Connected pixels: algorithm 1 example

Step 11:

i(x,y)

0	0	0	0	0
0	1	1	1	0
1	1	1	0	0
0	0	0	0	1
0	1	0	1	1

A(x,y)

-1	-1	-1	-1	-1
-1	0	0	0	-1
0	0	0	-1	-1
-1	-1	-1	-1	2
-1	3	-1	4	2

UNION: sets [2] and [4]

SET[0]={(1,1), (2,1),(3,1)
(0,2),(1,2),(2,2)}

SET[1]={}

SET[2]={(4,3),(4,4)}

SET[3]={(1,4)}

SET[4]={(3,4)}

Connected pixels: algorithm 1 example

Step 12:

i(x,y)

0	0	0	0	0
0	1	1	1	0
1	1	1	0	0
0	0	0	0	1
0	1	0	1	1

A(x,y)

-1	-1	-1	-1	-1
-1	0	0	0	-1
0	0	0	-1	-1
-1	-1	-1	-1	2
-1	3	-1	2	2

SET[0]={(1,1), (2,1),(3,1)
(0,2),(1,2),(2,2)}

SET[1]={}

SET[2]={(4,3),(4,4),(3,4)

SET[3]={(1,4)}

SET[4]={}

Finding the centre

To find the centre of mass of blobs, you can use:

$$x_{centre} = \frac{\sum_{i=0}^{n} x_i}{n} \qquad y_{centre} = \frac{\sum_{i=0}^{n} y_i}{n} \tag{1}$$

Exercise 1

1. Write a program using OpenCV that opens two images and shows them in the screen in sequence (one at a time).
2. Write a program to copy an image and mirror it. Show both the original image and the mirrored image simultaneously (open two windows).
3. Modify the program to copy the image to a grey-scale image. Save the grey-scale image with a different format than the original one.
4. Write a program that loads a colour image. Using a trackbar, the user should be able to change the image to grey-scale and back to colour (the trackbar should allow for values 0 and 1).

Bibliography

(partial)

Gary Bradski and Adrian Kaehler, Learning OpenCV, O'Reilly, 2008.