

OpenCV Tutorial 8 - Chapter 9

Author: Noah Kuntz (2009)

[Contact: nk752@drexel.edu](mailto:nk752@drexel.edu)

Keywords: OpenCV, computer vision, image processing, segmentation

My Vision Tutorials Index

This tutorial assumes the reader:

- (1) Has a basic knowledge of Visual C++
- (2) Has some familiarity with computer vision concepts
- (3) Has read the previous tutorials in this series

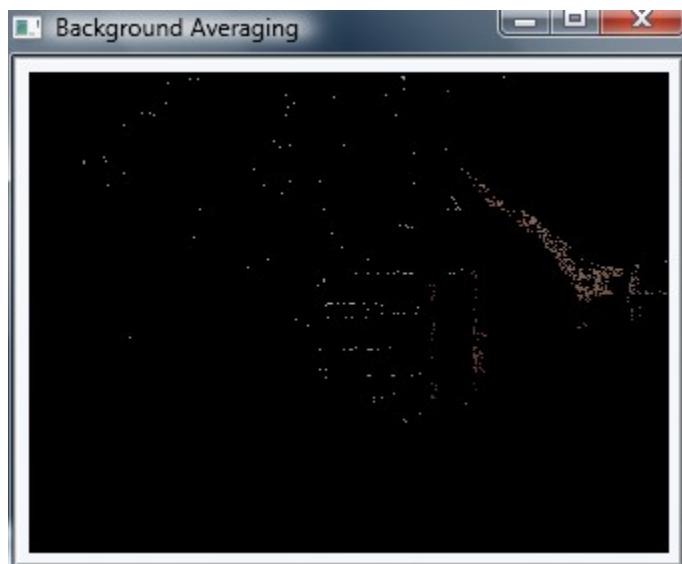
The rest of the tutorial is presented as follows:

- [Step 1: Background Averaging](#)
- [Step 2: Other Segmentation Functions](#)
- [Final Words](#)

Important Note!

More information on the topics of these tutorials can be found in this book: [Learning OpenCV: Computer Vision with the OpenCV Library](#)

Step 1: Background Averaging



Background averaging example

The background averaging method of image segmentation computes the average difference between

each image pixel and a model of the background. The background model is created by accumulating a set of frames and the difference between the current frame and the last frame with *cvAcc* and *cvAbsDiff*. After the model is learned, a mask is created using *cvInRange* and combining the channels with *cvOr*. This example uses *AllocateImages(IplImage* I)* to setup the images needed for scratch work, *accumulateBackground(IplImage* I)* to build up model information, *createModelFromStats()* to build the background model, *backgroundDiff(IplImage *I, IplImage *Imask)* to calculate the difference between the current image and the model, and finally *DeallocateImages()* to release the image objects used. Here is the code:

```
// Global storage
IplImage *IavgF, *IdiffF, *IprevF, *IhiF, *IlowF;
IplImage *Iscratch, *Iscratch2;
IplImage *Igray1, *Igray2, *Igray3;
IplImage *Ilow1, *Ilow2, *Ilow3;
IplImage *Ihi1, *Ihi2, *Ihi3;
IplImage *Imaskt;
float Icount;

// Function of allocating images
void AllocateImages( IplImage* I ){
    CvSize sz = cvGetSize( I );

    IavgF = cvCreateImage( sz, IPL_DEPTH_32F, 3 );
    IdiffF = cvCreateImage( sz, IPL_DEPTH_32F, 3 );
    IprevF = cvCreateImage( sz, IPL_DEPTH_32F, 3 );
    IhiF = cvCreateImage( sz, IPL_DEPTH_32F, 3 );
    IlowF = cvCreateImage( sz, IPL_DEPTH_32F, 3 );
    Ilow1 = cvCreateImage( sz, IPL_DEPTH_32F, 1 );
    Ilow2 = cvCreateImage( sz, IPL_DEPTH_32F, 1 );
    Ilow3 = cvCreateImage( sz, IPL_DEPTH_32F, 1 );
    Ihi1 = cvCreateImage( sz, IPL_DEPTH_32F, 1 );
    Ihi2 = cvCreateImage( sz, IPL_DEPTH_32F, 1 );
    Ihi3 = cvCreateImage( sz, IPL_DEPTH_32F, 1 );
    cvZero( IavgF );
    cvZero( IdiffF );
    cvZero( IprevF );
    cvZero( IhiF );
    cvZero( IlowF );
    Icount = 0.0001;           // protect against divid by 0

    Iscratch= cvCreateImage( sz, IPL_DEPTH_32F, 3 );
    Iscratch2 = cvCreateImage( sz, IPL_DEPTH_32F, 3 );
    Igray1 = cvCreateImage( sz, IPL_DEPTH_32F, 1 );
    Igray2 = cvCreateImage( sz, IPL_DEPTH_32F, 1 );
    Igray3 = cvCreateImage( sz, IPL_DEPTH_32F, 1 );
    Imaskt = cvCreateImage( sz, IPL_DEPTH_8U, 1 );
    cvZero( Iscratch );
    cvZero( Iscratch2 );
}

// Learn the background statistics for one more frame
void accumulateBackground( IplImage *I ){
    static int first = 1;
    cvCvtScale( I, Iscratch, 1, 0 );
    if( !first ){
        cvAcc( Iscratch, IavgF );
```

```

        cvAbsDiff( Iscratch, IprevF, Iscratch2 );
        cvAcc( Iscratch2, IdiffF );
        Icount += 1.0;
    }
    first = 0;
    cvCopy( Iscratch, IprevF );
}

void setHighThreshold( float scale ) {
    cvConvertScale( IdiffF, Iscratch, scale );
    cvAdd( Iscratch, IavgF, IhiF );
    cvSplit( IhiF, Ihi1, Ihi2, Ihi3, 0 );
}

void setLowThreshold( float scale ) {
    cvConvertScale( IdiffF, Iscratch, scale );
    cvAdd( Iscratch, IavgF, IlowF );
    cvSplit( IlowF, Ilow1, Ilow2, Ilow3, 0 );
}

void createModelsfromStats(){
    cvConvertScale( IavgF, IavgF, (double)(1.0/Icount) );
    cvConvertScale( IdiffF, IdiffF, (double)(1.0/Icount) );

    //Make sure diff is always something
    cvAddS( IdiffF, cvScalar( 1.0, 1.0, 1.0 ), IdiffF );
    setHighThreshold( 7.0 );
    setLowThreshold( 6.0 );
}

// Create a mask
void backgroundDiff( IplImage *I, IplImage *Imask ){
    cvCvtScale( I, Iscratch, 1, 0 );
    cvSplit( Iscratch, Igray1, Igray2, Igray3, 0 );

    // channel 1
    cvInRange( Igray1, Ilow1, Ihi1, Imask );
    // channel 2
    cvInRange( Igray2, Ilow2, Ihi2, Imask );
    cvOr( Imask, Imask, Imask );
    // channel 3
    cvInRange( Igray3, Ilow3, Ihi3, Imask );
    cvOr( Imask, Imask, Imask );
}

void DeallocateImages()
{
    cvReleaseImage( &IavgF );
    cvReleaseImage( &IdiffF );
    cvReleaseImage( &IprevF );
    cvReleaseImage( &IhiF );
    cvReleaseImage( &IlowF );
    cvReleaseImage( &Ilow1 );
    cvReleaseImage( &Ilow2 );
    cvReleaseImage( &Ilow3 );
    cvReleaseImage( &Ihi1 );
    cvReleaseImage( &Ihi2 );
    cvReleaseImage( &Ihi3 );
    cvReleaseImage( &Iscratch );
    cvReleaseImage( &Iscratch2 );
}

```

```

        cvReleaseImage( &Igray1 );
        cvReleaseImage( &Igray2 );
        cvReleaseImage( &Igray3 );
        cvReleaseImage( &Imaskt );
    }

int _tmain(int argc, _TCHAR* argv[])
{
    cvNamedWindow( "Background Averaging", CV_WINDOW_AUTOSIZE );
    CvCapture* capture = cvCreateFileCapture( "Averaging_Video.avi" );
    IplImage *frame, *mask1, *mask3;

    int frameCount = 0;
    while(1) {
        frameCount++;
        frame = cvQueryFrame( capture );
        if( !frame ) break;
        CvSize sz = cvGetSize( frame );
        mask1 = cvCreateImage( sz, IPL_DEPTH_8U, 1 );
        mask3 = cvCreateImage( sz, IPL_DEPTH_8U, 3 );
        if(frameCount == 1)
            AllocateImages( frame );

        if( frameCount < 30 ){
            accumulateBackground( frame );
        }else if( frameCount == 30 ){
            createModelsfromStats();
        }else{
            backgroundDiff( frame, mask1 );

            cvCvtColor(mask1,mask3,CV_GRAY2BGR);
            cvNorm( mask3, mask3, CV_C, 0 );
            cvThreshold(mask3, mask3, 100, 1, CV_THRESH_BINARY);
            cvMul( frame, mask3, frame, 1.0 );
            cvShowImage( "Background Averaging", frame );
        }

        char c = cvWaitKey(33);
        if( c == 27 ) break;
    }
    cvReleaseCapture( &capture );
    cvDestroyWindow( "Background Averaging" );
    DeallocateImages();
}

```

Step 2: Other Segmentation Functions

I want to touch on more processing that can be done with segmentation, please see the text to understand how to implement these functions. Codebooks can be used to build a more complex model of the background, there is an example in the text. There are other methods such as the watershed algorithm, that divides the image into "mountains" of edges separating "plains" of more uniform areas. And there is also Delaunay Triangulation, a method for noting the relationship between feature points in order to track an object. These methods and more are further examined in the text.

Final Words

This tutorial's objective was to show how to use some methods of segmentation.

Click [here](#) to email me.

Click [here](#) to return to my Tutorials page.