Praktikum 9
Contours and Segmentation

1. Contours examples

Program berikut ini menerapkan contour detection untuk pada gambar.

```
#include <cv.h>
#include <highgui.h>

IplImage*      g_image = NULL;
IplImage*      g_gray = NULL;
int            g_thresh = 100;
CvMemStorage*  g_storage = NULL;

void on_trackbar(int){
        if( g_storage == NULL ){
                g_gray = cvCreateImage( cvGetSize( g_image ), 8, 1 );
                g_storage = cvCreateMemStorage(0);
        } else {
                cvClearMemStorage( g_storage );
        }

        CvSeq* contours = 0;
        cvCvtColor( g_image, g_gray, CV_BGR2GRAY );
        cvThreshold( g_gray, g_gray, g_thresh, 255, CV_THRESH_BINARY );
        cvFindContours( g_gray, g_storage, &contours );
        cvZero( g_gray );
        if( contours ){
                cvDrawContours(
                        g_gray,
                        contours,
                        cvScalarAll(255),
                        cvScalarAll(255),
                        100 );
        }
        cvShowImage( "Contours", g_gray );
}


int main()
{

        g_image = cvLoadImage( "image.jpg" );
        cvNamedWindow( "Contours", 1 );
        cvCreateTrackbar( "Threshold", "Contours", &g_thresh, 255, on_trackbar );
        on_trackbar(0);
        cvWaitKey();

        return 0;
}
```

Petunjuk praktikum:

- Jelaskan algoritma contour pada program di atas.
- Jelaskan fungsi berikut ini beserta dengan parameter yang ada di dalamnya.
    - `cvFindContours()`
    - `cvDrawContours()`

2. External and Internal Contours

Program berikut ini menerapkan external dan internal contour pada gambar.

```
#include <cv.h>
#include <highgui.h>
#include <stdio.h>

int main()
{
        cvNamedWindow( "Contours 2", 1 );
        IplImage* img_8uc1 = cvLoadImage( "image.jpg", 0 );
        IplImage* img_edge = cvCreateImage( cvGetSize(img_8uc1), 8, 1 );
        IplImage* img_8uc3 = cvCreateImage( cvGetSize(img_8uc1), 8, 3 );

        cvThreshold( img_8uc1, img_edge, 128, 255, CV_THRESH_BINARY );

        CvMemStorage* storage = cvCreateMemStorage();
        CvSeq* first_contour = NULL;

        int Nc = cvFindContours(
                img_edge,
                storage,
                &first_contour,
                sizeof(CvContour),
                CV_RETR_LIST );

        int n=0;
        printf( "Total Contours Detected: %d\n", Nc );
        CvScalar red = CV_RGB(250,0,0);
        CvScalar blue = CV_RGB(0,0,250);

        for( CvSeq* c=first_contour; c!=NULL; c=c->h_next ){
                cvCvtColor( img_8uc1, img_8uc3, CV_GRAY2BGR );
                cvDrawContours(
                        img_8uc3,
                        c,
                        red,            // Red
                        blue,           // Blue
                        1,                      // Vary max_level and compare results
                        2,
                        8 );
                printf( "Contour #%dn", n );
                cvShowImage( "Contours 2", img_8uc3 );
                printf( " %d elements:\n", c->total );

                for( int i=0; i<c->total; ++i ){
                        CvPoint* p = CV_GET_SEQ_ELEM( CvPoint, c, i );
                        printf(" (%d,%d)\n", p->x, p->y );
                }
                cvWaitKey();
                n++;
        }

        printf( "Finished all contours.\n");
        cvCvtColor( img_8uc1, img_8uc3, CV_GRAY2BGR );
        cvShowImage( "Contours 2", img_8uc3 );
        cvWaitKey();

        cvDestroyWindow( "Contours 2" );

        cvReleaseImage( &img_8uc1 );
        cvReleaseImage( &img_8uc3 );
        cvReleaseImage( &img_edge );

        return 0;
}
```

Petunjuk praktikum:

- Jelaskan algoritma external dan internal contour pada program di atas.

3. Background Averaging

Program berikut ini menerapkan *background averaging* pada video.

```
#include <cv.h>
#include <highgui.h>
#include <stdio.h>

// 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, Imaskt );
	cvOr( Imask, Imaskt, Imask );
	// channel 3
	cvInRange( Igray3, Ilow3, Ihi3, Imaskt );
	cvOr( Imask, Imaskt, 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 main()
{
	cvNamedWindow( "Background Averaging", CV_WINDOW_AUTOSIZE );
	CvCapture* capture = cvCreateFileCapture( "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();
}
```

Petunjuk praktikum:

- Jelaskan algoritma Background Averaging pada program di atas.

4. Floodfill Algorithm

Program berikut ini menerapkan algoritma *floodfill* pada sebuah gambar.

```
#ifdef _CH_
#pragma package <opencv>
#endif

#ifndef _EiC
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <stdlib.h>
#endif

IplImage* color_img0;
IplImage* mask;
IplImage* color_img;
IplImage* gray_img0 = NULL;
IplImage* gray_img = NULL;
int ffill_case = 1;
int lo_diff = 20, up_diff = 20;
int connectivity = 4;
int is_color = 1;
int is_mask = 0;
int new_mask_val = 255;

void on_mouse( int event, int x, int y, int flags, void* param )
{
    if( !color_img )
        return;

    switch( event )
    {
    case CV_EVENT_LBUTTONDOWN:
        {
            CvPoint seed = cvPoint(x,y);
            int lo = ffill_case == 0 ? 0 : lo_diff;
            int up = ffill_case == 0 ? 0 : up_diff;
            int flags = connectivity + (new_mask_val << 8) +
                        (ffill_case == 1 ? CV_FLOODFILL_FIXED_RANGE : 0);
            int b = rand() & 255, g = rand() & 255, r = rand() & 255;
            CvConnectedComp comp;

            if( is_mask )
                cvThreshold( mask, mask, 1, 128, CV_THRESH_BINARY );

            if( is_color )
            {
                CvScalar color = CV_RGB( r, g, b );
                cvFloodFill( color_img, seed, color, CV_RGB( lo, lo, lo ),
                             CV_RGB( up, up, up ), &comp, flags, is_mask ? mask : NULL
);
```

```
                cvShowImage( "image", color_img );
            }
            else
            {
                CvScalar brightness = cvRealScalar((r*2 + g*7 + b + 5)/10);
                cvFloodFill( gray_img, seed, brightness, cvRealScalar(lo),
                             cvRealScalar(up), &comp, flags, is_mask ? mask : NULL );
                cvShowImage( "image", gray_img );
            }

            printf("%g pixels were repainted\n", comp.area );

            if( is_mask )
                cvShowImage( "mask", mask );
        }
        break;
    }
}

int main( int argc, char** argv )
{
    char* filename = argc >= 2 ? argv[1] : (char*)"image.jpg";

    if( (color_img0 = cvLoadImage(filename,1)) == 0 )
        return 0;

    printf( "Hot keys: \n"
            "\tESC - quit the program\n"
            "\tc - switch color/grayscale mode\n"
            "\tm - switch mask mode\n"
            "\tr - restore the original image\n"
            "\ts - use null-range floodfill\n"
            "\tf - use gradient floodfill with fixed(absolute) range\n"
            "\tg - use gradient floodfill with floating(relative) range\n"
            "\t4 - use 4-connectivity mode\n"
            "\t8 - use 8-connectivity mode\n" );

    color_img = cvCloneImage( color_img0 );
    gray_img0 = cvCreateImage( cvSize(color_img->width, color_img->height), 8, 1 );
    cvCvtColor( color_img, gray_img0, CV_BGR2GRAY );
    gray_img = cvCloneImage( gray_img0 );
    mask = cvCreateImage( cvSize(color_img->width + 2, color_img->height + 2), 8, 1 );

    cvNamedWindow( "image", 0 );
    cvCreateTrackbar( "lo_diff", "image", &lo_diff, 255, NULL );
    cvCreateTrackbar( "up_diff", "image", &up_diff, 255, NULL );

    cvSetMouseCallback( "image", on_mouse, 0 );

    for(;;)
    {
        int c;

        if( is_color )
            cvShowImage( "image", color_img );
        else
            cvShowImage( "image", gray_img );

        c = cvWaitKey(0);
        switch( (char) c )
        {
        case '\x1b':
            printf("Exiting ...\n");
            goto exit_main;
        case 'c':
            if( is_color )
            {
                printf("Grayscale mode is set\n");
                cvCvtColor( color_img, gray_img, CV_BGR2GRAY );
                is_color = 0;
            }
            else
            {
                printf("Color mode is set\n");
                cvCopy( color_img0, color_img, NULL );
                cvZero( mask );
```

```
                is_color = 1;
            }
            break;
        case 'm':
            if( is_mask )
            {
                cvDestroyWindow( "mask" );
                is_mask = 0;
            }
            else
            {
                cvNamedWindow( "mask", 0 );
                cvZero( mask );
                cvShowImage( "mask", mask );
                is_mask = 1;
            }
            break;
        case 'r':
            printf("Original image is restored\n");
            cvCopy( color_img0, color_img, NULL );
            cvCopy( gray_img0, gray_img, NULL );
            cvZero( mask );
            break;
        case 's':
            printf("Simple floodfill mode is set\n");
            ffill_case = 0;
            break;
        case 'f':
            printf("Fixed Range floodfill mode is set\n");
            ffill_case = 1;
            break;
        case 'g':
            printf("Gradient (floating range) floodfill mode is set\n");
            ffill_case = 2;
            break;
        case '4':
            printf("4-connectivity mode is set\n");
            connectivity = 4;
            break;
        case '8':
            printf("8-connectivity mode is set\n");
            connectivity = 8;
            break;
        }
    }

exit_main:

    cvDestroyWindow( "test" );
    cvReleaseImage( &gray_img );
    cvReleaseImage( &gray_img0 );
    cvReleaseImage( &color_img );
    cvReleaseImage( &color_img0 );
    cvReleaseImage( &mask );

    return 1;
}

#ifdef _EiC
main(1,"ffilldemo.c");
#endif
```

Petunjuk praktikum:

- Jelaskan algoritma *floodfill* pada program di atas.

5. Watershed Algorithm

Program berikut ini menerapkan algoritma *watershed* pada sebuah gambar.

```
#ifdef _CH_
#pragma package <opencv>
#endif

#ifndef _EiC
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <stdlib.h>
#endif

IplImage* marker_mask = 0;
IplImage* markers = 0;
IplImage* img0 = 0, *img = 0, *img_gray = 0, *wshed = 0;
CvPoint prev_pt = {-1,-1};

void on_mouse( int event, int x, int y, int flags, void* param )
{
    if( !img )
        return;

    if( event == CV_EVENT_LBUTTONUP || !(flags & CV_EVENT_FLAG_LBUTTON) )
        prev_pt = cvPoint(-1,-1);
    else if( event == CV_EVENT_LBUTTONDOWN )
        prev_pt = cvPoint(x,y);
    else if( event == CV_EVENT_MOUSEMOVE && (flags & CV_EVENT_FLAG_LBUTTON) )
    {
        CvPoint pt = cvPoint(x,y);
        if( prev_pt.x < 0 )
            prev_pt = pt;
        cvLine( marker_mask, prev_pt, pt, cvScalarAll(255), 5, 8, 0 );
        cvLine( img, prev_pt, pt, cvScalarAll(255), 5, 8, 0 );
        prev_pt = pt;
        cvShowImage( "image", img );
    }
}


int main( int argc, char** argv )
{
    char* filename = argc >= 2 ? argv[1] : (char*)"image.jpg";
    CvRNG rng = cvRNG(-1);

    if( (img0 = cvLoadImage(filename,1)) == 0 )
        return 0;

    printf( "Hot keys: \n"
            "\tESC - quit the program\n"
            "\tr - restore the original image\n"
            "\tw or SPACE - run watershed algorithm\n"
            "\t\t(before running it, roughly mark the areas on the image)\n"
            "\t  (before that, roughly outline several markers on the image)\n" );

    cvNamedWindow( "image", 1 );
    cvNamedWindow( "watershed transform", 1 );

    img = cvCloneImage( img0 );
    img_gray = cvCloneImage( img0 );
    wshed = cvCloneImage( img0 );
    marker_mask = cvCreateImage( cvGetSize(img), 8, 1 );
    markers = cvCreateImage( cvGetSize(img), IPL_DEPTH_32S, 1 );
    cvCvtColor( img, marker_mask, CV_BGR2GRAY );
    cvCvtColor( marker_mask, img_gray, CV_GRAY2BGR );

    cvZero( marker_mask );
    cvZero( wshed );
    cvShowImage( "image", img );
    cvShowImage( "watershed transform", wshed );
    cvSetMouseCallback( "image", on_mouse, 0 );
```

```
    for(;;)
    {
        int c = cvWaitKey(0);

        if( (char)c == 27 )
            break;

        if( (char)c == 'r' )
        {
            cvZero( marker_mask );
            cvCopy( img0, img );
            cvShowImage( "image", img );
        }

        if( (char)c == 'w' || (char)c == ' ' )
        {
            CvMemStorage* storage = cvCreateMemStorage(0);
            CvSeq* contours = 0;
            CvMat* color_tab;
            int i, j, comp_count = 0;
            //cvSaveImage( "wshed_mask.png", marker_mask );
            //marker_mask = cvLoadImage( "wshed_mask.png", 0 );
            cvFindContours( marker_mask, storage, &contours, sizeof(CvContour),
                            CV_RETR_CCOMP, CV_CHAIN_APPROX_SIMPLE );
            cvZero( markers );
            for( ; contours != 0; contours = contours->h_next, comp_count++ )
            {
                cvDrawContours( markers, contours, cvScalarAll(comp_count+1),
                                cvScalarAll(comp_count+1), -1, -1, 8, cvPoint(0,0) );
            }

            color_tab = cvCreateMat( 1, comp_count, CV_8UC3 );
            for( i = 0; i < comp_count; i++ )
            {
                uchar* ptr = color_tab->data.ptr + i*3;
                ptr[0] = (uchar)(cvRandInt(&rng)%180 + 50);
                ptr[1] = (uchar)(cvRandInt(&rng)%180 + 50);
                ptr[2] = (uchar)(cvRandInt(&rng)%180 + 50);
            }

            {
            double t = (double)cvGetTickCount();
            cvWatershed( img0, markers );
            t = (double)cvGetTickCount() - t;
            printf( "exec time = %gms\n", t/(cvGetTickFrequency()*1000.) );
            }

            // paint the watershed image
            for( i = 0; i < markers->height; i++ )
                for( j = 0; j < markers->width; j++ )
                {
                    int idx = CV_IMAGE_ELEM( markers, int, i, j );
                    uchar* dst = &CV_IMAGE_ELEM( wshed, uchar, i, j*3 );
                    if( idx == -1 )
                        dst[0] = dst[1] = dst[2] = (uchar)255;
                    else if( idx <= 0 || idx > comp_count )
                        dst[0] = dst[1] = dst[2] = (uchar)0; // should not get here
                    else
                    {
                        uchar* ptr = color_tab->data.ptr + (idx-1)*3;
                        dst[0] = ptr[0]; dst[1] = ptr[1]; dst[2] = ptr[2];
                    }
                }
            cvAddWeighted( wshed, 0.5, img_gray, 0.5, 0, wshed );
            cvShowImage( "watershed transform", wshed );
            cvReleaseMemStorage( &storage );
            cvReleaseMat( &color_tab );
        }
    }

    return 1;
}

#ifdef _EiC
main(1,"watershed.cpp");
#endif
```

Petunjuk praktikum:

- Jelaskan algoritma *watershed* pada program di atas.

Tugas:

Buatlah program untuk melakukan segmentasi dan filtering pada sebuah gambar dengan menggunakan fungsi di bawah ini:

- `cvPyrSegmentation()`
- `cvPyrMeanShiftFiltering()`