Praktikum 13
Motions and Tracking

1. Motion Template

Program berikut ini menerapkan motion detection dengan motion template.

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

#ifndef _EiC
// motion templates sample code
#include "cv.h"
#include "highgui.h"
#include <time.h>
#include <math.h>
#include <ctype.h>
#include <stdio.h>
#endif

// various tracking parameters (in seconds)
const double MHI_DURATION = 1;
const double MAX_TIME_DELTA = 0.5;
const double MIN_TIME_DELTA = 0.05;
// number of cyclic frame buffer used for motion detection
// (should, probably, depend on FPS)
const int N = 4;

// ring image buffer
IplImage **buf = 0;
int last = 0;

// temporary images
IplImage *mhi = 0; // MHI
IplImage *orient = 0; // orientation
IplImage *mask = 0; // valid orientation mask
IplImage *segmask = 0; // motion segmentation map
CvMemStorage* storage = 0; // temporary storage

// parameters:
//  img - input video frame
//  dst - resultant motion picture
//  args - optional parameters
void  update_mhi( IplImage* img, IplImage* dst, int diff_threshold )
{
    double timestamp = (double)clock()/CLOCKS_PER_SEC; // get current time in seconds
    CvSize size = cvSize(img->width,img->height); // get current frame size
    int i, idx1 = last, idx2;
    IplImage* silh;
    CvSeq* seq;
    CvRect comp_rect;
    double count;
    double angle;
    CvPoint center;
    double magnitude;
    CvScalar color;

    // allocate images at the beginning or
    // reallocate them if the frame size is changed
    if( !mhi || mhi->width != size.width || mhi->height != size.height ) {
        if( buf == 0 ) {
            buf = (IplImage**)malloc(N*sizeof(buf[0]));
            memset( buf, 0, N*sizeof(buf[0]));
        }

        for( i = 0; i < N; i++ ) {
            cvReleaseImage( &buf[i] );
            buf[i] = cvCreateImage( size, IPL_DEPTH_8U, 1 );
            cvZero( buf[i] );
        }
        cvReleaseImage( &mhi );
        cvReleaseImage( &orient );
        cvReleaseImage( &segmask );
        cvReleaseImage( &mask );

        mhi = cvCreateImage( size, IPL_DEPTH_32F, 1 );
        cvZero( mhi ); // clear MHI at the beginning
        orient = cvCreateImage( size, IPL_DEPTH_32F, 1 );
        segmask = cvCreateImage( size, IPL_DEPTH_32F, 1 );
        mask = cvCreateImage( size, IPL_DEPTH_8U, 1 );
    }

    cvCvtColor( img, buf[last], CV_BGR2GRAY ); // convert frame to grayscale

    idx2 = (last + 1) % N; // index of (last - (N-1))th frame
    last = idx2;

    silh = buf[idx2];
    cvAbsDiff( buf[idx1], buf[idx2], silh ); // get difference between frames

    cvThreshold( silh, silh, diff_threshold, 1, CV_THRESH_BINARY ); // and threshold
it
    cvUpdateMotionHistory( silh, mhi, timestamp, MHI_DURATION ); // update MHI

    // convert MHI to blue 8u image
    cvCvtScale( mhi, mask, 255./MHI_DURATION,
                (MHI_DURATION - timestamp)*255./MHI_DURATION );
    cvZero( dst );
    cvCvtPlaneToPix( mask, 0, 0, 0, dst );

    // calculate motion gradient orientation and valid orientation mask
    cvCalcMotionGradient( mhi, mask, orient, MAX_TIME_DELTA, MIN_TIME_DELTA, 3 );

    if( !storage )
        storage = cvCreateMemStorage(0);
    else
        cvClearMemStorage(storage);

    // segment motion: get sequence of motion components
    // segmask is marked motion components map. It is not used further
    seq = cvSegmentMotion( mhi, segmask, storage, timestamp, MAX_TIME_DELTA );

    // iterate through the motion components,
    // One more iteration (i == -1) corresponds to the whole image (global motion)
    for( i = -1; i < seq->total; i++ ) {

        if( i < 0 ) { // case of the whole image
            comp_rect = cvRect( 0, 0, size.width, size.height );
            color = CV_RGB(255,255,255);
            magnitude = 100;
        }
        else { // i-th motion component
            comp_rect = ((CvConnectedComp*)cvGetSeqElem( seq, i ))->rect;
            if( comp_rect.width + comp_rect.height < 100 ) // reject very small
components
                continue;
            color = CV_RGB(255,0,0);
            magnitude = 30;
        }

        // select component ROI
        cvSetImageROI( silh, comp_rect );
        cvSetImageROI( mhi, comp_rect );
        cvSetImageROI( orient, comp_rect );
        cvSetImageROI( mask, comp_rect );

        // calculate orientation
        angle = cvCalcGlobalOrientation( orient, mask, mhi, timestamp, MHI_DURATION);
        angle = 360.0 - angle;  // adjust for images with top-left origin

        count = cvNorm( silh, 0, CV_L1, 0 ); // calculate number of points within
silhouette ROI

        cvResetImageROI( mhi );
        cvResetImageROI( orient );
        cvResetImageROI( mask );
        cvResetImageROI( silh );

        // check for the case of little motion
        if( count < comp_rect.width*comp_rect.height * 0.05 )
            continue;

        // draw a clock with arrow indicating the direction
        center = cvPoint( (comp_rect.x + comp_rect.width/2),
                          (comp_rect.y + comp_rect.height/2) );

        cvCircle( dst, center, cvRound(magnitude*1.2), color, 3, CV_AA, 0 );
        cvLine( dst, center, cvPoint( cvRound( center.x +
magnitude*cos(angle*CV_PI/180)),
                cvRound( center.y - magnitude*sin(angle*CV_PI/180))), color, 3, CV_AA,
0 );
    }
}

int main(int argc, char** argv)
{
    IplImage* motion = 0;
    CvCapture* capture = 0;

    if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 && isdigit(argv[1][0])))
        capture = cvCaptureFromCAM( argc == 2 ? argv[1][0] - '0' : 0 );
    else if( argc == 2 )
        capture = cvCaptureFromFile( argv[1] );

    if( capture )
    {
        cvNamedWindow( "Motion", 1 );

        for(;;)
        {
            IplImage* image;
            if( !cvGrabFrame( capture ))
                break;
            image = cvRetrieveFrame( capture );

            if( image )
            {
                if( !motion )
                {
                    motion = cvCreateImage( cvSize(image->width,image->height), 8, 3
);
                    cvZero( motion );
                    motion->origin = image->origin;
                }
            }

            update_mhi( image, motion, 30 );
            cvShowImage( "Motion", motion );

            if( cvWaitKey(10) >= 0 )
                break;
        }
        cvReleaseCapture( &capture );
        cvDestroyWindow( "Motion" );
    }

    return 0;
}

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

Petunjuk praktikum:

- Jalankan program di atas, coba dengan 1 objek bergerak kemudian coba juga dengan 2 atau lebih objek bergerak kemudian amati hasilnya.
- Jelaskan fungsi dari indikator putih dan merah yang ada pada program.
- Jelaskan algoritma Motion Template pada program di atas.

2. CamShift

Program berikut ini menerapkan penerapan tracking dengan menggunakan algoritma CamShift.

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

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

IplImage *image = 0, *hsv = 0, *hue = 0, *mask = 0, *backproject = 0, *histimg = 0;
CvHistogram *hist = 0;

int backproject_mode = 0;
int select_object = 0;
int track_object = 0;
int show_hist = 1;
CvPoint origin;
CvRect selection;
CvRect track_window;
CvBox2D track_box;
CvConnectedComp track_comp;
int hdims = 16;
float hranges_arr[] = {0,180};
float* hranges = hranges_arr;
int vmin = 10, vmax = 256, smin = 30;

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

    if( image->origin )
        y = image->height - y;

    if( select_object )
    {
        selection.x = MIN(x,origin.x);
        selection.y = MIN(y,origin.y);
        selection.width = selection.x + CV_IABS(x - origin.x);
        selection.height = selection.y + CV_IABS(y - origin.y);

        selection.x = MAX( selection.x, 0 );
        selection.y = MAX( selection.y, 0 );
        selection.width = MIN( selection.width, image->width );
        selection.height = MIN( selection.height, image->height );
        selection.width -= selection.x;
        selection.height -= selection.y;
    }

    switch( event )
    {
    case CV_EVENT_LBUTTONDOWN:
        origin = cvPoint(x,y);
        selection = cvRect(x,y,0,0);
        select_object = 1;
        break;
    case CV_EVENT_LBUTTONUP:
        select_object = 0;
        if( selection.width > 0 && selection.height > 0 )
            track_object = -1;
        break;
    }
}


CvScalar hsv2rgb( float hue )
```

```
{
    int rgb[3], p, sector;
    static const int sector_data[][3]=
        {{0,2,1}, {1,2,0}, {1,0,2}, {2,0,1}, {2,1,0}, {0,1,2}};
    hue *= 0.033333333333333333333333333333333f;
    sector = cvFloor(hue);
    p = cvRound(255*(hue - sector));
    p ^= sector & 1 ? 255 : 0;

    rgb[sector_data[sector][0]] = 255;
    rgb[sector_data[sector][1]] = 0;
    rgb[sector_data[sector][2]] = p;

    return cvScalar(rgb[2], rgb[1], rgb[0],0);
}

int main( int argc, char** argv )
{
    CvCapture* capture = 0;

    if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 && isdigit(argv[1][0])))
        capture = cvCaptureFromCAM( argc == 2 ? argv[1][0] - '0' : 0 );
    else if( argc == 2 )
        capture = cvCaptureFromAVI( argv[1] );

    if( !capture )
    {
        fprintf(stderr,"Could not initialize capturing...\n");
        return -1;
    }

    printf( "Hot keys: \n"
        "\tESC - quit the program\n"
        "\tc - stop the tracking\n"
        "\tb - switch to/from backprojection view\n"
        "\th - show/hide object histogram\n"
        "To initialize tracking, select the object with mouse\n" );

    cvNamedWindow( "Histogram", 1 );
    cvNamedWindow( "CamShiftDemo", 1 );
    cvSetMouseCallback( "CamShiftDemo", on_mouse, 0 );
    cvCreateTrackbar( "Vmin", "CamShiftDemo", &vmin, 256, 0 );
    cvCreateTrackbar( "Vmax", "CamShiftDemo", &vmax, 256, 0 );
    cvCreateTrackbar( "Smin", "CamShiftDemo", &smin, 256, 0 );

    for(;;)
    {
        IplImage* frame = 0;
        int i, bin_w, c;

        frame = cvQueryFrame( capture );
        if( !frame )
            break;

        if( !image )
        {
            /* allocate all the buffers */
            image = cvCreateImage( cvGetSize(frame), 8, 3 );
            image->origin = frame->origin;
            hsv = cvCreateImage( cvGetSize(frame), 8, 3 );
            hue = cvCreateImage( cvGetSize(frame), 8, 1 );
            mask = cvCreateImage( cvGetSize(frame), 8, 1 );
            backproject = cvCreateImage( cvGetSize(frame), 8, 1 );
            hist = cvCreateHist( 1, &hdims, CV_HIST_ARRAY, &hranges, 1 );
            histimg = cvCreateImage( cvSize(320,200), 8, 3 );
            cvZero( histimg );
        }

        cvCopy( frame, image, 0 );
        cvCvtColor( image, hsv, CV_BGR2HSV );

        if( track_object )
        {
            int _vmin = vmin, _vmax = vmax;

            cvInRangeS( hsv, cvScalar(0,smin,MIN(_vmin,_vmax),0),
                        cvScalar(180,256,MAX(_vmin,_vmax),0), mask );
            cvSplit( hsv, hue, 0, 0, 0 );

            if( track_object < 0 )
            {
                float max_val = 0.f;
                cvSetImageROI( hue, selection );
                cvSetImageROI( mask, selection );
                cvCalcHist( &hue, hist, 0, mask );
                cvGetMinMaxHistValue( hist, 0, &max_val, 0, 0 );
                cvConvertScale( hist->bins, hist->bins, max_val ? 255. / max_val : 0.,
0 );
                cvResetImageROI( hue );
                cvResetImageROI( mask );
                track_window = selection;
                track_object = 1;

                cvZero( histimg );
                bin_w = histimg->width / hdims;
                for( i = 0; i < hdims; i++ )
                {
                    int val = cvRound( cvGetReal1D(hist->bins,i)*histimg->height/255
);
                    CvScalar color = hsv2rgb(i*180.f/hdims);
                    cvRectangle( histimg, cvPoint(i*bin_w,histimg->height),
                                 cvPoint((i+1)*bin_w,histimg->height - val),
                                 color, -1, 8, 0 );
                }
            }

            cvCalcBackProject( &hue, backproject, hist );
            cvAnd( backproject, mask, backproject, 0 );
            cvCamShift( backproject, track_window,
                        cvTermCriteria( CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 10, 1 ),
                        &track_comp, &track_box );
            track_window = track_comp.rect;

            if( backproject_mode )
                cvCvtColor( backproject, image, CV_GRAY2BGR );
            if( !image->origin )
                track_box.angle = -track_box.angle;
            cvEllipseBox( image, track_box, CV_RGB(255,0,0), 3, CV_AA, 0 );
        }

        if( select_object && selection.width > 0 && selection.height > 0 )
        {
            cvSetImageROI( image, selection );
            cvXorS( image, cvScalarAll(255), image, 0 );
            cvResetImageROI( image );
        }

        cvShowImage( "CamShiftDemo", image );
        cvShowImage( "Histogram", histimg );

        c = cvWaitKey(10);
        if( (char) c == 27 )
            break;
        switch( (char) c )
        {
        case 'b':
            backproject_mode ^= 1;
            break;
        case 'c':
            track_object = 0;
            cvZero( histimg );
            break;
        case 'h':
            show_hist ^= 1;
            if( !show_hist )
                cvDestroyWindow( "Histogram" );
            else
                cvNamedWindow( "Histogram", 1 );
            break;
        default:
            ;
        }
```

```
    }

    cvReleaseCapture( &capture );
    cvDestroyWindow("CamShiftDemo");

    return 0;
}

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

Petunjuk praktikum:

- Jalankan program diatas, kemudian ambil sebuah objek dan berikan *boundary* (batasan) pada objek tersebut kemudian gerak gerakkan. Amati apa yang terjadi.
- Rubah parameter Vmin, Vmax, Smin, kemudian amati perbuahannya. Jelaskan fungsi-fungsi parameter tersebut.
- Jelaskan algoritma CamShift pada program di atas.

3. Lucas Kanade

Program berikut ini menerapkan tracking dengan menggunakan algoritma Lucas Kanade.

```
#/* Demo of modified Lucas-Kanade optical flow algorithm.
   See the printf below */

#ifdef _CH_
#pragma package <opencv>
#endif

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

IplImage *image = 0, *grey = 0, *prev_grey = 0, *pyramid = 0, *prev_pyramid = 0,
*swap_temp;

int win_size = 10;
const int MAX_COUNT = 500;
CvPoint2D32f* points[2] = {0,0}, *swap_points;
char* status = 0;
int count = 0;
int need_to_init = 0;
int night_mode = 0;
int flags = 0;
int add_remove_pt = 0;
CvPoint pt;


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

    if( image->origin )
        y = image->height - y;

    if( event == CV_EVENT_LBUTTONDOWN )
    {
        pt = cvPoint(x,y);
        add_remove_pt = 1;
    }
}


int main( int argc, char** argv )
{
    CvCapture* capture = 0;

    if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 && isdigit(argv[1][0])))
        capture = cvCaptureFromCAM( argc == 2 ? argv[1][0] - '0' : 0 );
    else if( argc == 2 )
        capture = cvCaptureFromAVI( argv[1] );

    if( !capture )
    {
        fprintf(stderr,"Could not initialize capturing...\n");
        return -1;
    }

    /* print a welcome message, and the OpenCV version */
    printf ("Welcome to lkdemo, using OpenCV version %s (%d.%d.%d)\n",
           CV_VERSION,
           CV_MAJOR_VERSION, CV_MINOR_VERSION, CV_SUBMINOR_VERSION);

    printf( "Hot keys: \n"
            "\tESC - quit the program\n"
            "\tr - auto-initialize tracking\n"
            "\tc - delete all the points\n"
            "\tn - switch the \"night\" mode on/off\n"
            "To add/remove a feature point click it\n" );

    cvNamedWindow( "LkDemo", 0 );
    cvSetMouseCallback( "LkDemo", on_mouse, 0 );

    for(;;)
    {
        IplImage* frame = 0;
        int i, k, c;

        frame = cvQueryFrame( capture );
        if( !frame )
            break;

        if( !image )
        {
            /* allocate all the buffers */
            image = cvCreateImage( cvGetSize(frame), 8, 3 );
            image->origin = frame->origin;
            grey = cvCreateImage( cvGetSize(frame), 8, 1 );
            prev_grey = cvCreateImage( cvGetSize(frame), 8, 1 );
            pyramid = cvCreateImage( cvGetSize(frame), 8, 1 );
            prev_pyramid = cvCreateImage( cvGetSize(frame), 8, 1 );
            points[0] = (CvPoint2D32f*)cvAlloc(MAX_COUNT*sizeof(points[0][0]));
            points[1] = (CvPoint2D32f*)cvAlloc(MAX_COUNT*sizeof(points[0][0]));
            status = (char*)cvAlloc(MAX_COUNT);
            flags = 0;
        }

        cvCopy( frame, image, 0 );
        cvCvtColor( image, grey, CV_BGR2GRAY );

        if( night_mode )
            cvZero( image );

        if( need_to_init )
        {
            /* automatic initialization */
            IplImage* eig = cvCreateImage( cvGetSize(grey), 32, 1 );
            IplImage* temp = cvCreateImage( cvGetSize(grey), 32, 1 );
            double quality = 0.01;
            double min_distance = 10;

            count = MAX_COUNT;
            cvGoodFeaturesToTrack( grey, eig, temp, points[1], &count,
                                   quality, min_distance, 0, 3, 0, 0.04 );
            cvFindCornerSubPix( grey, points[1], count,
                cvSize(win_size,win_size), cvSize(-1,-1),
                cvTermCriteria(CV_TERMCRIT_ITER|CV_TERMCRIT_EPS,20,0.03));
            cvReleaseImage( &eig );
            cvReleaseImage( &temp );

            add_remove_pt = 0;
        }
        else if( count > 0 )
        {
            cvCalcOpticalFlowPyrLK( prev_grey, grey, prev_pyramid, pyramid,
                points[0], points[1], count, cvSize(win_size,win_size), 3, status, 0,
                cvTermCriteria(CV_TERMCRIT_ITER|CV_TERMCRIT_EPS,20,0.03), flags );
            flags |= CV_LKFLOW_PYR_A_READY;
            for( i = k = 0; i < count; i++ )
            {
                if( add_remove_pt )
                {
                    double dx = pt.x - points[1][i].x;
                    double dy = pt.y - points[1][i].y;

                    if( dx*dx + dy*dy <= 25 )
                    {
                        add_remove_pt = 0;
                        continue;
                    }
                }

                if( !status[i] )
                    continue;

                points[1][k++] = points[1][i];
                cvCircle( image, cvPointFrom32f(points[1][i]), 3, CV_RGB(0,255,0), -1,
8,0);
            }
            count = k;
        }

        if( add_remove_pt && count < MAX_COUNT )
        {
            points[1][count++] = cvPointTo32f(pt);
            cvFindCornerSubPix( grey, points[1] + count - 1, 1,
                cvSize(win_size,win_size), cvSize(-1,-1),
                cvTermCriteria(CV_TERMCRIT_ITER|CV_TERMCRIT_EPS,20,0.03));
            add_remove_pt = 0;
        }

        CV_SWAP( prev_grey, grey, swap_temp );
        CV_SWAP( prev_pyramid, pyramid, swap_temp );
        CV_SWAP( points[0], points[1], swap_points );
        need_to_init = 0;
        cvShowImage( "LkDemo", image );

        c = cvWaitKey(10);
        if( (char)c == 27 )
            break;
        switch( (char) c )
        {
        case 'r':
            need_to_init = 1;
            break;
        case 'c':
            count = 0;
            break;
        case 'n':
            night_mode ^= 1;
            break;
        default:
            ;
        }
    }

    cvReleaseCapture( &capture );
    cvDestroyWindow("LkDemo");
    return 0;
}

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

Petunjuk praktikum:

- Jalankan program di atas, kemudian berikan titik-titik fitur pada objek, gerak-gerakkan objek. Amati apa yang terjadi jika objek;
    - Bergerak vertical dan horizontal
    - Berubah bentuk, dan
    - Keluar dari bidang gambar

  Jelaskan mengapa hal tersebut terjadi.

- Jelaskan algoritma Lucas Kanade pada program di atas.