Praktikum 6
Image Transformation 2

1. Laplacian Transform

Program berikut ini menggunakan fungsi Laplacian Transform pada sebuah video.

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

int main( int argc, char** argv )
{
    IplImage* laplace = 0;
    IplImage* colorlaplace = 0;
    IplImage* planes[3] = { 0, 0, 0 };
    CvCapture* capture = 0;

    capture = cvCaptureFromAVI( "video.avi" );

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

    cvNamedWindow( "Laplacian", 0 );

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

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

        if( !laplace )
        {
            for( i = 0; i < 3; i++ )
                planes[i] = cvCreateImage( cvSize(frame->width,frame->height), 8, 1 );
            laplace = cvCreateImage( cvSize(frame->width,frame->height),
IPL_DEPTH_16S, 1 );
            colorlaplace = cvCreateImage( cvSize(frame->width,frame->height), 8, 3 );
        }

        cvCvtPixToPlane( frame, planes[0], planes[1], planes[2], 0 );
        for( i = 0; i < 3; i++ )
        {
            cvLaplace( planes[i], laplace, 3 );
            cvConvertScaleAbs( laplace, planes[i], 1, 0 );
        }
        cvCvtPlaneToPix( planes[0], planes[1], planes[2], 0, colorlaplace );
        colorlaplace->origin = frame->origin;

        cvShowImage("Laplacian", colorlaplace );

        if( cvWaitKey(10) >= 0 )
            break;
    }

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

    return 0;
}
```

Petunjuk praktikum:

- Jelaskan konsep metode Laplacian Transform pada program di atas.
- Jelaskan fungsi berikut ini beserta dengan parameter yang ada di dalamnya.
    - cvLaplace()
    - cvConvertScaleAbs()

2. Discrete Fourier Transform (DFT)

Program berikut ini menerapkan fungsi Discrete Fourier Transform (DFT) pada sebuah gambar.

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

// Rearrange the quadrants of Fourier image so that the origin is at
// the image center
// src & dst arrays of equal size & type
void cvShiftDFT(CvArr * src_arr, CvArr * dst_arr )
{
    CvMat * tmp;
    CvMat q1stub, q2stub;
    CvMat q3stub, q4stub;
    CvMat d1stub, d2stub;
    CvMat d3stub, d4stub;
    CvMat * q1, * q2, * q3, * q4;
    CvMat * d1, * d2, * d3, * d4;

    CvSize size = cvGetSize(src_arr);
    CvSize dst_size = cvGetSize(dst_arr);
    int cx, cy;

    if(dst_size.width != size.width ||
       dst_size.height != size.height){
        cvError( CV_StsUnmatchedSizes, "cvShiftDFT", "Source and Destination arrays
must have equal sizes", __FILE__, __LINE__ );
    }

    if(src_arr==dst_arr){
        tmp = cvCreateMat(size.height/2, size.width/2, cvGetElemType(src_arr));
    }

    cx = size.width/2;
    cy = size.height/2; // image center

    q1 = cvGetSubRect( src_arr, &q1stub, cvRect(0,0,cx, cy) );
    q2 = cvGetSubRect( src_arr, &q2stub, cvRect(cx,0,cx,cy) );
    q3 = cvGetSubRect( src_arr, &q3stub, cvRect(cx,cy,cx,cy) );
    q4 = cvGetSubRect( src_arr, &q4stub, cvRect(0,cy,cx,cy) );
    d1 = cvGetSubRect( src_arr, &d1stub, cvRect(0,0,cx,cy) );
    d2 = cvGetSubRect( src_arr, &d2stub, cvRect(cx,0,cx,cy) );
    d3 = cvGetSubRect( src_arr, &d3stub, cvRect(cx,cy,cx,cy) );
    d4 = cvGetSubRect( src_arr, &d4stub, cvRect(0,cy,cx,cy) );

    if(src_arr!=dst_arr){
        if( !CV_ARE_TYPES_EQ( q1, d1 )){
            cvError( CV_StsUnmatchedFormats, "cvShiftDFT", "Source and Destination
arrays must have the same format", __FILE__, __LINE__ );
        }
        cvCopy(q3, d1, 0);
        cvCopy(q4, d2, 0);
        cvCopy(q1, d3, 0);
        cvCopy(q2, d4, 0);
    }
```

```
    else{
        cvCopy(q3, tmp, 0);
        cvCopy(q1, q3, 0);
        cvCopy(tmp, q1, 0);
        cvCopy(q4, tmp, 0);
        cvCopy(q2, q4, 0);
        cvCopy(tmp, q2, 0);
    }
}

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

    IplImage * realInput;
    IplImage * imaginaryInput;
    IplImage * complexInput;
    int dft_M, dft_N;
    CvMat* dft_A, tmp;
    IplImage * image_Re;
    IplImage * image_Im;
    double m, M;

    im = cvLoadImage( filename, CV_LOAD_IMAGE_GRAYSCALE );
    if( !im )
        return -1;

    realInput = cvCreateImage( cvGetSize(im), IPL_DEPTH_64F, 1);
    imaginaryInput = cvCreateImage( cvGetSize(im), IPL_DEPTH_64F, 1);
    complexInput = cvCreateImage( cvGetSize(im), IPL_DEPTH_64F, 2);

    cvScale(im, realInput, 1.0, 0.0);
    cvZero(imaginaryInput);
    cvMerge(realInput, imaginaryInput, NULL, NULL, complexInput);

    dft_M = cvGetOptimalDFTSize( im->height - 1 );
    dft_N = cvGetOptimalDFTSize( im->width - 1 );

    dft_A = cvCreateMat( dft_M, dft_N, CV_64FC2 );
    image_Re = cvCreateImage( cvSize(dft_N, dft_M), IPL_DEPTH_64F, 1);
    image_Im = cvCreateImage( cvSize(dft_N, dft_M), IPL_DEPTH_64F, 1);

    // copy A to dft_A and pad dft_A with zeros
    cvGetSubRect( dft_A, &tmp, cvRect(0,0, im->width, im->height));
    cvCopy( complexInput, &tmp, NULL );
    if( dft_A->cols > im->width )
    {
        cvGetSubRect( dft_A, &tmp, cvRect(im->width,0, dft_A->cols - im->width, im-
>height));
        cvZero( &tmp );
    }

    // no need to pad bottom part of dft_A with zeros because of
    // use nonzero_rows parameter in cvDFT() call below

    cvDFT( dft_A, dft_A, CV_DXT_FORWARD, complexInput->height );

    cvNamedWindow("win", 0);
    cvNamedWindow("magnitude", 0);
    cvShowImage("win", im);

    // Split Fourier in real and imaginary parts
    cvSplit( dft_A, image_Re, image_Im, 0, 0 );

    // Compute the magnitude of the spectrum Mag = sqrt(Re^2 + Im^2)
    cvPow( image_Re, image_Re, 2.0);
    cvPow( image_Im, image_Im, 2.0);
    cvAdd( image_Re, image_Im, image_Re, NULL);
    cvPow( image_Re, image_Re, 0.5 );

    // Compute log(1 + Mag)
    cvAddS( image_Re, cvScalarAll(1.0), image_Re, NULL ); // 1 + Mag
    cvLog( image_Re, image_Re ); // log(1 + Mag)
```

```
    // Rearrange the quadrants of Fourier image so that the origin is at
    // the image center
    cvShiftDFT( image_Re, image_Re );

    cvMinMaxLoc(image_Re, &m, &M, NULL, NULL, NULL);
    cvScale(image_Re, image_Re, 1.0/(M-m), 1.0*(-m)/(M-m));
    cvShowImage("magnitude", image_Re);

    cvWaitKey(-1);
    return 0;
}
```

Petunjuk praktikum:

- Jelaskan konsep Discrete Fourier Transform pada program di atas.
- Jelaskan fungsi berikut ini beserta dengan parameter yang ada di dalamnya.
    - `cvDFT()`
    - `cvShiftDFT()`

3. Distance Transform

Program berikut ini menggunakan fungsi Distance Transform yang digunakan untuk menghitung zero pixel terdekat dari semua non-zero pixel dari sebuah gambar.

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

char wndname[] = "Distance transform";
char tbarname[] = "Threshold";
int mask_size = CV_DIST_MASK_5;
int build_voronoi = 0;
int edge_thresh = 100;
int dist_type = CV_DIST_L1;

// The output and temporary images
IplImage* dist = 0;
IplImage* dist8u1 = 0;
IplImage* dist8u2 = 0;
IplImage* dist8u = 0;
IplImage* dist32s = 0;

IplImage* gray = 0;
IplImage* edge = 0;
IplImage* labels = 0;

// threshold trackbar callback
void on_trackbar( int dummy )
{
    static const uchar colors[][3] =
    {
        {0,0,0},
        {255,0,0},
        {255,128,0},
        {255,255,0},
        {0,255,0},
        {0,128,255},
        {0,255,255},
        {0,0,255},
        {255,0,255}
    };

    int msize = mask_size;
```

```
    int _dist_type = build_voronoi ? CV_DIST_L2 : dist_type;

    cvThreshold( gray, edge, (float)edge_thresh, (float)edge_thresh, CV_THRESH_BINARY
);

    if( build_voronoi )
        msize = CV_DIST_MASK_5;

    if( _dist_type == CV_DIST_L1 )
    {
        cvDistTransform( edge, edge, _dist_type, msize, NULL, NULL );
        cvConvert( edge, dist );
    }
    else
        cvDistTransform( edge, dist, _dist_type, msize, NULL, build_voronoi ? labels :
NULL );

    if( !build_voronoi )
    {
        // begin "painting" the distance transform result
        cvConvertScale( dist, dist, 5000.0, 0 );
        cvPow( dist, dist, 0.5 );

        cvConvertScale( dist, dist32s, 1.0, 0.5 );
        cvAndS( dist32s, cvScalarAll(255), dist32s, 0 );
        cvConvertScale( dist32s, dist8u1, 1, 0 );
        cvConvertScale( dist32s, dist32s, -1, 0 );
        cvAddS( dist32s, cvScalarAll(255), dist32s, 0 );
        cvConvertScale( dist32s, dist8u2, 1, 0 );
        cvMerge( dist8u1, dist8u2, dist8u2, 0, dist8u );
        // end "painting" the distance transform result
    }
    else
    {
        int i, j;
        for( i = 0; i < labels->height; i++ )
        {
            int* ll = (int*)(labels->imageData + i*labels->widthStep);
            float* dd = (float*)(dist->imageData + i*dist->widthStep);
            uchar* d = (uchar*)(dist8u->imageData + i*dist8u->widthStep);
            for( j = 0; j < labels->width; j++ )
            {
                int idx = ll[j] == 0 || dd[j] == 0 ? 0 : (ll[j]-1)%8 + 1;
                int b = cvRound(colors[idx][0]);
                int g = cvRound(colors[idx][1]);
                int r = cvRound(colors[idx][2]);
                d[j*3] = (uchar)b;
                d[j*3+1] = (uchar)g;
                d[j*3+2] = (uchar)r;
            }
        }
    }

    cvShowImage( wndname, dist8u );
}

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

    if( (gray = cvLoadImage( filename, 0 )) == 0 )
        return -1;

    printf( "Hot keys: \n"
        "\tESC - quit the program\n"
        "\tC - use C/Inf metric\n"
        "\tL1 - use L1 metric\n"
        "\tL2 - use L2 metric\n"
        "\t3 - use 3x3 mask\n"
        "\t5 - use 5x5 mask\n"
        "\t0 - use precise distance transform\n"
        "\tv - switch Voronoi diagram mode on/off\n"
        "\tSPACE - loop through all the modes\n" );

    dist = cvCreateImage( cvGetSize(gray), IPL_DEPTH_32F, 1 );
    dist8u1 = cvCloneImage( gray );
```

```
    dist8u2 = cvCloneImage( gray );
    dist8u = cvCreateImage( cvGetSize(gray), IPL_DEPTH_8U, 3 );
    dist32s = cvCreateImage( cvGetSize(gray), IPL_DEPTH_32S, 1 );
    edge = cvCloneImage( gray );
    labels = cvCreateImage( cvGetSize(gray), IPL_DEPTH_32S, 1 );

    cvNamedWindow( wndname, 1 );

    cvCreateTrackbar( tbarname, wndname, &edge_thresh, 255, on_trackbar );

    for(;;)
    {
        int c;

        // Call to update the view
        on_trackbar(0);

        c = cvWaitKey(0);

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

        if( (char)c == 'c' || (char)c == 'C' )
            dist_type = CV_DIST_C;
        else if( (char)c == '1' )
            dist_type = CV_DIST_L1;
        else if( (char)c == '2' )
            dist_type = CV_DIST_L2;
        else if( (char)c == '3' )
            mask_size = CV_DIST_MASK_3;
        else if( (char)c == '5' )
            mask_size = CV_DIST_MASK_5;
        else if( (char)c == '0' )
            mask_size = CV_DIST_MASK_PRECISE;
        else if( (char)c == 'v' )
            build_voronoi ^= 1;
        else if( (char)c == ' ' )
        {
            if( build_voronoi )
            {
                build_voronoi = 0;
                mask_size = CV_DIST_MASK_3;
                dist_type = CV_DIST_C;
            }
            else if( dist_type == CV_DIST_C )
                dist_type = CV_DIST_L1;
            else if( dist_type == CV_DIST_L1 )
                dist_type = CV_DIST_L2;
            else if( mask_size == CV_DIST_MASK_3 )
                mask_size = CV_DIST_MASK_5;
            else if( mask_size == CV_DIST_MASK_5 )
                mask_size = CV_DIST_MASK_PRECISE;
            else if( mask_size == CV_DIST_MASK_PRECISE )
                build_voronoi = 1;
        }
    }

    cvReleaseImage( &gray );
    cvReleaseImage( &edge );
    cvReleaseImage( &dist );
    cvReleaseImage( &dist8u );
    cvReleaseImage( &dist8u1 );
    cvReleaseImage( &dist8u2 );
    cvReleaseImage( &dist32s );
    cvReleaseImage( &labels );

    cvDestroyWindow( wndname );

    return 0;
}
```

Petunjuk praktikum:

- Tekan tombol “SPACE” untuk berganti mode distance transform, kemudian ubah nilai thresholdnya. Amati perubahan pada gambar setelah dilakukan distance transform pada mode-mode yang berlainan.

- Jelaskan konsep metode Distance Transform pada program di atas.

- Jelaskan fungsi berikut ini beserta dengan parameter yang ada di dalamnya.

    - `cvDistTransform()`
    - `cvConvert()`

4. Inpaint

Program berikut ini menggunakan fungsi Inpaint yang digunakan untuk memperbaiki sebuah gambar yang rusak.

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

IplImage* inpaint_mask = 0;
IplImage* img0 = 0, *img = 0, *inpainted = 0;
CvPoint prev_pt = {-1,-1};

void on_mouse( int event, int x, int y, int flags, void* )
{
    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( inpaint_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";

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

    printf( "Hot keys: \n"
            "\tESC - quit the program\n"
            "\tr - restore the original image\n"
            "\ti or SPACE - run inpainting algorithm\n"
            "\t\t(before running it, paint something on the image)\n" );

    cvNamedWindow( "image", 1 );
```

```
    img = cvCloneImage( img0 );
    inpainted = cvCloneImage( img0 );
    inpaint_mask = cvCreateImage( cvGetSize(img), 8, 1 );

    cvZero( inpaint_mask );
    cvZero( inpainted );
    cvShowImage( "image", img );
    cvShowImage( "watershed transform", inpainted );
    cvSetMouseCallback( "image", on_mouse, 0 );

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

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

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

        if( (char)c == 'i' || (char)c == ' ' )
        {
            cvNamedWindow( "inpainted image", 1 );
            cvInpaint( img, inpaint_mask, inpainted, 3, CV_INPAINT_TELEA );
            cvShowImage( "inpainted image", inpainted );
        }
    }

    return 1;
}
```

Petunjuk praktikum:

- Buatlah tulisan pada media gambar yang sudah disediakan, kemudian tekan tombol “R” untuk melakukan proses *inpaint*. Amati apa yang terjadi pada gambar yang baru. Jika ingin mengulangi kembali tekan tombol “SPACE”.
- Jelaskan konsep metode Inpaint pada program di atas.
- Jelaskan fungsi berikut ini beserta dengan parameter yang ada di dalamnya.
    - `cvInpaint()`

Tugas: Advanced Morphological Transformation

Buatlah program untuk transformasi morfologi menggunakan fungsi berikut ini:

- `cvMorphologyEx()`