Lampiran 1: Listing Program Gabor.m

```
% Gabor Kernels generation
clear all;
domX = -16:16;
domY = -16:16;
sigma = 2*pi;
threshold = 3e-4;
kMax = pi/2;
kfac = 1 / sqrt(2);
kArray = kMax * [kfac^4 kfac^3 kfac^2 kfac 1];
thetaInit = 0;
dTheta = pi / 16;
thetaArray = thetaInit:dTheta:(pi-dTheta);

[xx, yy] = meshgrid(domX, domY);
xVec = xx + yy * i;
x = abs(xVec);

p = cell(length(thetaArray), length(kArray));
for iter1 = 1:length(thetaArray)
      for iter2 = 1:length(kArray)
            theta = thetaArray(iter1); k = kArray(iter2);
            kVec = k * exp(i * theta);
            pTemp = k^2 / sigma^2 * exp(-k^2 / (2 * sigma^2) * x.^2) .* (exp ...
            (i* (real(kVec) * real(xVec) + imag(kVec) * imag(xVec))) - ...
            exp(-sigma^2 / 2));
            [ki, kj] = find(abs(pTemp) > threshold);
            p{iter1, iter2} = pTemp(min(ki):max(ki), min(kj):max(kj));
      end
end

save gabor p  % save kernels

index=0;
figure; colormap(gray);
for theta = 0:15,
      for k = 0:4,
            index = index+1;
            subplot(16, 5, index);
            kern = p{theta+1, k+1};
            real_kern=real(kern);
            imag_kern=imag(kern);

            % REAL PART
            file1=['kernel\rea_' num2str(theta) '_' num2str(k) '.txt'];
            save files.txt real_kern -ASCII
            fid = fopen('files.txt','r');
            F   = fread(fid);
            s   = setstr(F');
            fclose(fid);
            fid = fopen(file1,'wb');
            fwrite(fid,s);
            fclose(fid);

            % IMAGINER PART
            file2=['kernel\ima_' num2str(theta) '_' num2str(k) '.txt'];
            save files.txt imag_kern -ASCII
```

```
            fid = fopen('files.txt','r');
            F   = fread(fid);
            s   = setstr(F');
            fclose(fid);
            fid = fopen(file2,'wb');
            fwrite(fid,s);
            fclose(fid);
            axis off;
        end
end
truesize;
```

Lampiran 2: Listing Program EnrollDlg.cpp

```
#include "stdafx.h"
#include "Hand Geometry Verification.h"
#include "EnrollDlg.h"
#include "mediatypes.h"
#include "kernel.h"

//Create database (TXT)
#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <math.h>
#include <fstream.h>
#include <stdio.h>
#include <string.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CEnrollDlg dialog
CEnrollDlg::CEnrollDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CEnrollDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CEnrollDlg)
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CEnrollDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CEnrollDlg)
	DDX_Control(pDX, IDC_STATUS, m_sbar);
	DDX_Control(pDX, IDC_IMAGE, m_image);
	DDX_Control(pDX, IDC_IMAGE_LIST, m_files);
	DDX_Control(pDX, IDC_DATABASE_LIST, m_db);
	DDX_Control(pDX, IDC_USER_NAME, m_user);
	DDX_Control(pDX, IDC_DIR, m_dir);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CEnrollDlg, CDialog)
	//{{AFX_MSG_MAP(CEnrollDlg)
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_LOAD, OnLoad)
	ON_LBN_SELCHANGE(IDC_IMAGE_LIST, OnShowImage)
	ON_EN_KILLFOCUS(IDC_DIR, OnLoad2)
	ON_BN_CLICKED(IDC_ENROLL, OnEnroll)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
////////////////////////////////////////////////////////////////////////////
// CEnrollDlg message handlers
BOOL CEnrollDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	// Set the icon for this dialog.  The framework does this automatically
	// when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon
	// TODO: Add extra initialization here
	GetCurrentDirectory(MAX_PATH, m_curdir);
	GetCurrentDirectory(MAX_PATH, m_defdir);
	m_dir.SetWindowText(m_curdir);
	FillFileList(m_curdir);
	m_db.InsertColumn(0, "User name",LVCFMT_LEFT, 90);
	m_db.InsertColumn(1, "File name",LVCFMT_LEFT, 110);
	FillDataBase();
	return TRUE;  // return TRUE unless you set the focus to a control
			// EXCEPTION: OCX Property Pages should return FALSE
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CEnrollDlg::OnPaint()
{
	// Get a pointer to the parent window
	CEnrollDlg *pWnd = (CEnrollDlg*)GetParent();
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting
		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);
		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;
		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CEnrollDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}
```

```
/************************************
*      PARAMETER MAIN PROGRAM      *
************************************/
// KERNEL (KERNEL.H)
int jkernel=5*8;
float *loop[ ] = { real_0_0, real_0_1, real_0_2, real_0_3, real_0_4,
                   real_1_0, real_1_1, real_1_2, real_1_3, real_1_4,
                   real_2_0, real_2_1, real_2_2, real_2_3, real_2_4,
                   real_3_0, real_3_1, real_3_2, real_3_3, real_3_4,
                   real_4_0, real_4_1, real_4_2, real_4_3, real_4_4,
                   real_5_0, real_5_1, real_5_2, real_5_3, real_5_4,
                   real_6_0, real_6_1, real_6_2, real_6_3, real_6_4,
                   real_7_0, real_7_1, real_7_2, real_7_3, real_7_4,
                   imag_0_0, imag_0_1, imag_0_2, imag_0_3, imag_0_4,
                   imag_1_0, imag_1_1, imag_1_2, imag_1_3, imag_1_4,
                   imag_2_0, imag_2_1, imag_2_2, imag_2_3, imag_2_4,
                   imag_3_0, imag_3_1, imag_3_2, imag_3_3, imag_3_4,
                   imag_4_0, imag_4_1, imag_4_2, imag_4_3, imag_4_4,
                   imag_5_0, imag_5_1, imag_5_2, imag_5_3, imag_5_4,
                   imag_6_0, imag_6_1, imag_6_2, imag_6_3, imag_6_4,
                   imag_7_0, imag_7_1, imag_7_2, imag_7_3, imag_7_4};

void CEnrollDlg::OnLoad()
{
    BROWSEINFO bi;
    bi.hwndOwner = NULL;
    bi.pidlRoot =NULL;
    bi.pszDisplayName = m_curdir;          //buffer File name
    bi.lpszTitle = "Choose test images directory" ;
    bi.ulFlags = 0;
    bi.lpfn = NULL;
    bi.lParam = 0;
    LPITEMIDLIST il = SHBrowseForFolder( &bi );
    char path[1024];
    if ( SHGetPathFromIDList( il, path ) )
    m_dir.SetWindowText(path);
    // Read the string in the media directory edit box.
    m_dir.GetWindowText(m_curdir, MAX_PATH);
     // Is this a valid directory name?
    DWORD dwAttr = GetFileAttributes(m_curdir);
    if ((dwAttr == (DWORD) -1) || (! (dwAttr & FILE_ATTRIBUTE_DIRECTORY)))
        {
        MessageBox("Please enter a valid directory name.","Confirmation",MB_OK);
        return;
     }
    int nLength = _tcslen(m_curdir);
    if (m_curdir[nLength - 1] != TEXT('\\'))
    wsprintf(m_curdir, TEXT("%s\\\0"), m_curdir);
    FillFileList(m_curdir);
}

void CEnrollDlg::OnLoad2()
{
     // Read the string in the media directory edit box.
    m_dir.GetWindowText(m_curdir, MAX_PATH);
    // Is this a valid directory name?
    DWORD dwAttr = GetFileAttributes(m_curdir);
```

```
    if ((dwAttr == (DWORD) -1) || (! (dwAttr & FILE_ATTRIBUTE_DIRECTORY)))
    {
        MessageBox ("Please enter a valid directory name.","Confirmation",MB_OK);
        return;
    }
    int nLength = _tcslen(m_curdir);
    if (m_curdir[nLength - 1] != TEXT('\\'))
    wsprintf(m_curdir, TEXT("%s\\\0"), m_curdir);
    FillFileList(m_curdir);
}

void CEnrollDlg::FillFileList(LPTSTR pszRootDir)
{
    UINT attr = 0;
    // Clear current file list
    m_files.ResetContent();
    ::SetCurrentDirectory(pszRootDir);
    // Add all of our known supported media types to the file list.
    // Add files of each type in order.
    for (int i=0; i < NUM_MEDIA_TYPES; i++)
    {
    m_files.Dir(attr, TypeInfo[i].pszType);
    }
}

/*********************
*   VIEW IMAGE          *
*********************/
void CEnrollDlg::OnShowImage()
{
    TCHAR nama[MAX_PATH];
    // If this is the currently selected file, do nothing
    int nItem = m_files.GetCurSel();
    // Remember the current selection to speed double-click processing
    m_nSel = nItem;
    // Read file name from list box
    m_files.GetText(nItem, nama);
    RECT rc;
    int width, height;
    m_image.GetClientRect(&rc);
    width =  rc.right - rc.left;
    height = rc.bottom  - rc.top;
    IplImage *temp=cvvLoadImage(nama);
    cvvSaveImage("c:\\temp.bmp", temp);
    HBITMAP hbitmap = (HBITMAP) ::LoadImage (AfxGetInstanceHandle(), "c:\\temp.bmp",
    IMAGE_BITMAP,
    width,height, LR_LOADFROMFILE | LR_DEFAULTSIZE);
    if (hbitmap)
    m_image.SetBitmap (hbitmap);
}

/*****************************
*       FILL DATABASE         *
*****************************/
void CEnrollDlg::FillDataBase()
{
    m_db.DeleteAllItems();
    TCHAR db[100];
```

```
    sprintf(db, "%s%s",m_defdir,"\\db.txt");
    ifstream inp(db);
    TCHAR szSize[30];
    int count=0;
    while (!inp.eof())
    {
        TCHAR buff_file[22];
        inp.getline(buff_file, 21);
        TCHAR buff_user[27];
        inp.getline(buff_user, 26);
        TCHAR buff_titik[3];
        for (int i=0; i<12; i++)
        {
            inp.getline(buff_titik, 2);
        }
        TCHAR buff_value[17];
        inp.getline(buff_value, 16);            //Change line
        for (i=0; i<12*jkernel; i++)
        {
            inp.getline(buff_value, 16);
        }
        inp.getline(buff_value, 16);             //Change line
        //buff_user empty ???
        int nItem = m_db.InsertItem(0, buff_user);
        m_db.SetItemText( nItem, 1, buff_file);
        count++;
    }
    inp.close();
    m_db.DeleteItem( 0 );
    wsprintf(szSize, " Database %d Record ", count-1);
    m_sbar.SetWindowText(szSize);
}

/*****************************
*            PROSES          *
*****************************/
int exist = 0;
void CEnrollDlg::OnEnroll()
{
    TCHAR nama[MAX_PATH];
    int nItem = m_files.GetCurSel();
    m_nSel = nItem;
    m_files.GetText(nItem, nama);
    ifstream file( nama, ios::nocreate);
    if ( file.good() )
    {
        TCHAR save[100];
        sprintf(save, "%s\\image\\%s",m_defdir,nama);
        //check source file there on directory image ???
        ifstream test( save, ios::nocreate );
        if ( test.fail() )
        {
            TCHAR user[25];
            // Read the string in the media directory edit box.
            m_user.GetWindowText(user,25);
            IplImage *source=cvvLoadImage( nama );//files[z] ); //
            // check Empty user
            int p = strlen(user);
```

```
                int abc = strncmp(user,"",p);
                if (abc != 0)
                {
                    proses1( nama );
                    cvvSaveImage(save, source);
                    cvReleaseImage(&source);
                    //Save to db.txt
                    TCHAR db[100];
                    sprintf(db, "%s%s",m_defdir,"\\db.txt");
                    char teks[4];
                    ofstream out;
                    out.open(db, ios::app);
                    out << setiosflags(ios::left) << setw(20) << nama;
                    out << setiosflags(ios::left) << setw(25) << user;
                    for( int i=0; i<12; i++)
                    {
                        if (i<11)
                            out << setiosflags(ios::left) << setw(1) << titik_live[i];
                        else
                            out << setiosflags(ios::left) << setw(1) << titik_live[i] << endl;
                    }
                    for( int j=0; j<12*jkernel; j++)
                    {
                        _gcvt( hasil_live[j][4], 9, teks );
                        if ( j < 12*jkernel-1 )
                            out << setiosflags(ios::left) << setw(15) << teks;
                        else
                            out << setiosflags(ios::left) << setw(15) << teks <<endl;
                    }
                    out.close();
                    FillDataBase();
                }
                else
                    MessageBox("User name empty","User name empty",
                    MB_OK|MB_ICONERROR );
            }
            else
            {
                char display[128];
                sprintf(display," File %s found on DataBase", nama);
                MessageBox(display,"File found on DataBase", MB_OK|MB_ICONERROR );
            }
            test.close();
}
    file.close();
}

void CEnrollDlg::proses1(char nama_file[20])
{
    //declare variable
    CvSize size_templ, size_source, size_result;
    int w =39, h =39;                   //Kernel declaration
    size_templ = cvSize( w, h );
    // declare image
    IplImage *source_8U_1,          *source_8U_1_copy;
    IplImage *source_roi_8U_1,      *source_roi_32F_1;
    IplImage *real_roi_32F_1,       *imag_roi_32F_1;
    IplImage *real_sqr_32F_1,       *imag_sqr_32F_1;
```

```
IplImage *add_sqr_32F_1,       *add_sqr_8U_1;
IplImage *source_32F_1,        *source_canny;
source_8U_3 = cvvLoadImage( nama_file );
size_source.width = source_8U_3->width;
size_source.height = source_8U_3->height;
size_result = cvSize( size_source.width-w+1, size_source.height-h+1);
source_8U_1 = cvCreateImage (size_source, IPL_DEPTH_8U, 1);
cvvConvertImage (source_8U_3, source_8U_1, 0);
source_canny = cvCreateImage (size_source, IPL_DEPTH_8U, 1);
cvCanny(source_8U_1, source_canny, 0, 255, 5 );
//Histogram
const int range=256;
int arr_1[range+1], arr_2[range], arr_3[range] ;
IplLUT lut = { range+1, NULL, NULL, NULL, IPL_LUT_LOOKUP };
IplLUT *plut = &lut;
for( int p=0; p<range+1; p++)
      arr_1[p]=p;
int *parray_1 = &arr_1[0];
lut.key = parray_1;
for( p=0; p<range; p++)
      arr_2[p]=0;
int *parray_2 = &arr_2[0];
lut.value = parray_2;
for( p=0; p<range; p++)
      arr_3[p]=0;
int *parray_3 = &arr_3[0];
lut.factor= parray_3;
iplComputeHisto( source_8U_1, &plut );
iplHistoEqualize( source_8U_1, source_8U_1, &plut );
source_32F_1 = cvCreateImage (size_source, IPL_DEPTH_32F, 1);
cvConvertScale( source_8U_1, source_32F_1, 1, 0 );
source_8U_1_copy               = cvCloneImage( source_8U_1 );
source_roi_8U_1                = cvCloneImage( source_8U_1 );
source_roi_32F_1               = cvCloneImage( source_32F_1 );
real_roi_32F_1                 = cvCloneImage( source_32F_1 );
imag_roi_32F_1                 = cvCloneImage( source_32F_1 );
real_sqr_32F_1                 = cvCloneImage( source_32F_1 );
imag_sqr_32F_1                 = cvCloneImage( source_32F_1 );
add_sqr_32F_1                  = cvCloneImage( source_32F_1 );
add_sqr_8U_1                   = cvCloneImage( source_8U_1 );
//Create point grid
int data_point[12][2]=
{
      70,190, 115,70, 123,96, 165,50, 160,95, 210,70,
      200,100, 250,115, 160,160, 200,160, 180,190, 180,220,
};
int count=0;
//Repeat 12 sampling point
for (int j=0; j<12; j++)
{
      point.x=data_point[j][0];
      point.y=data_point[j][1];
      if (point.x+20<=size_source.width &&
            point.y+20<=size_source.height &&
            point.x-20>=0 && point.y-20>=0 )
      {
            titik_live[j]=1;
```

```
            //GABOR FILTERING
            int a,b;
            float pixel[4];
            for (int i=0; i<jkernel; i++)
            {
                if ((i+1)%7)
                { a=33; b=16;}
                else
                { a=27; b=13;}
                IplConvKernelFP* r_kernel = iplCreateConvKernelFP( a, a, b, b,loop[i]);
                IplConvKernelFP*i_kernel= iplCreateConvKernelFP( a, a, b, b,loop[i+jkernel]);
                CvRect size_roi;
                size_roi = cvRect( point.x-b, point.y-b, a, a );
                CvSize size_image;
                size_image = cvSize( a, a);
                IplImage*source_roi_8U_1 = cvCreateImage( size_image, IPL_DEPTH_8U, 1);
                cvSetImageROI( source_8U_1_copy, size_roi);
                cvCopyImage( source_8U_1_copy, source_roi_8U_1);
                source_roi_32F_1 = cvCreateImage (size_image, IPL_DEPTH_32F, 1);
                cvConvertScale( source_roi_8U_1, source_roi_32F_1, 1 , 0 );
                cvReleaseImage(&source_roi_8U_1);
                real_roi_32F_1 = cvCreateImage (size_image, IPL_DEPTH_32F, 1);
                imag_roi_32F_1 = cvCreateImage (size_image, IPL_DEPTH_32F, 1);
                add_sqr_32F_1 = cvCreateImage (size_image, IPL_DEPTH_32F, 1);
                iplMultiplySFP(source_roi_32F_1, source_roi_32F_1, (float)1.0/255.0 );
                iplConvolve2DFP(source_roi_32F_1, real_roi_32F_1,&r_kernel, 1, IPL_SUM);
                iplConvolve2DFP(source_roi_32F_1,imag_roi_32F_1,&i_kernel, 1, IPL_SUM);
                iplSquare( real_roi_32F_1, real_sqr_32F_1);
                iplSquare( imag_roi_32F_1, imag_sqr_32F_1);
                cvReleaseImage(&real_roi_32F_1);
                cvReleaseImage(&imag_roi_32F_1);
                iplAdd( real_sqr_32F_1, imag_sqr_32F_1, add_sqr_32F_1);
                int cnt=0;
                for (int t=-1; t<2; t++)
                {
                    for (int u=-1; u<2; u++)
                    {
                        iplGetPixel( add_sqr_32F_1, b+t, b+u, pixel);
                        hasil_live[count][cnt]=pixel[0];
                        cnt++;
                    }
                }
                cvReleaseImage(&add_sqr_32F_1);
                count++;            //Count up, data ke.. kernel ke..
            }
        }
        else
        {
            titik_live[j]=0;
            for (int i=0; i<jkernel; i++)
            {
                for (int k=0; k<9; k++)
                hasil_live[count][k]=0;
                count++;
            }
        }
    }
    RECT rc;
```

```
    int width, height;
    m_image.GetClientRect(&rc);
    width =  rc.right - rc.left;
    height = rc.bottom  - rc.top;
    cvvSaveImage("c:\\temp.bmp", source_8U_3);
    HBITMAP hbitmap = (HBITMAP) ::LoadImage (AfxGetInstanceHandle(), "c:\\temp.bmp",
    IMAGE_BITMAP,
    width,height, LR_LOADFROMFILE | LR_DEFAULTSIZE);
    if (hbitmap)
    m_image.SetBitmap (hbitmap);
}
```

Lampiran 3: Listing Program VerifylDlg.cpp

```
#include "stdafx.h"
#include "Hand Geometry Verification.h"
#include "VerifyDlg.h"
#include "krnl.h"

//Create database (TXT)
#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <math.h>
#include <fstream.h>
#include <stdio.h>
#include <string.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CVerifyDlg dialog
CVerifyDlg::CVerifyDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CVerifyDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CVerifyDlg)
	//}}AFX_DATA_INIT
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CVerifyDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CVerifyDlg)
	DDX_Control(pDX, IDC_SIM1, m_sim1);
	DDX_Control(pDX, IDC_USER3, m_user3);
	DDX_Control(pDX, IDC_USER2, m_user2);
	DDX_Control(pDX, IDC_SIM3, m_sim3);
	DDX_Control(pDX, IDC_SIM2, m_sim2);
	DDX_Control(pDX, IDC_FILE3, m_file3);
	DDX_Control(pDX, IDC_FILE2, m_file2);
	DDX_Control(pDX, IDC_FILE1, m_file1);
	DDX_Control(pDX, IDC_USER1, m_user1);
	DDX_Control(pDX, IDC_SHOW3, m_image3);
	DDX_Control(pDX, IDC_SHOW2, m_image2);
	DDX_Control(pDX, IDC_SHOW1, m_image1);
	DDX_Control(pDX, IDC_STATUS, m_sbar);
	DDX_Control(pDX, IDC_IMAGE, m_image);
	DDX_Control(pDX, IDC_FILE, m_file);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CVerifyDlg, CDialog)
	//{{AFX_MSG_MAP(CVerifyDlg)
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
```

```
    ON_BN_CLICKED(IDC_LOAD, OnLoad)
    ON_BN_CLICKED(IDC_VERIFY, OnVerify)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CVerifyDlg message handlers
BOOL CVerifyDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    // TODO: Add extra initialization here
    // Set the icon for this dialog.  The framework does this automatically
    //  when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE);            // Set big icon
    SetIcon(m_hIcon, FALSE);           // Set small icon
    // TODO: Add extra initialization here
    GetCurrentDirectory(MAX_PATH, m_defdir);
    CountDatabase();
    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}

void CVerifyDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting
        SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);
        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;
        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
    // Do not call CDialog::OnPaint() for painting messages
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CVerifyDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

/************************************
*      PARAMETER MAIN PROGRAM      *
************************************/
// KERNEL (KRNL.H)
int _jkernel=5*8;
float *_loop[ ] = {_real_0_0, _real_0_1, _real_0_2, _real_0_3, _real_0_4,
```

```
                    _real_1_0, _real_1_1, _real_1_2, _real_1_3, _real_1_4,
                    _real_2_0, _real_2_1, _real_2_2, _real_2_3, _real_2_4,
                    _real_3_0, _real_3_1, _real_3_2, _real_3_3, _real_3_4,
                    _real_4_0, _real_4_1, _real_4_2, _real_4_3, _real_4_4,
                    _real_5_0, _real_5_1, _real_5_2, _real_5_3, _real_5_4,
                    _real_6_0, _real_6_1, _real_6_2, _real_6_3, _real_6_4,
                    _real_7_0, _real_7_1, _real_7_2, _real_7_3, _real_7_4,
                    _imag_0_0, _imag_0_1, _imag_0_2, _imag_0_3, _imag_0_4,
                    _imag_1_0, _imag_1_1, _imag_1_2, _imag_1_3, _imag_1_4,
                    _imag_2_0, _imag_2_1, _imag_2_2, _imag_2_3, _imag_2_4,
                    _imag_3_0, _imag_3_1, _imag_3_2, _imag_3_3, _imag_3_4,
                    _imag_4_0, _imag_4_1, _imag_4_2, _imag_4_3, _imag_4_4,
                    _imag_5_0, _imag_5_1, _imag_5_2, _imag_5_3, _imag_5_4,
                    _imag_6_0, _imag_6_1, _imag_6_2, _imag_6_3, _imag_6_4,
                    _imag_7_0, _imag_7_1, _imag_7_2, _imag_7_3, _imag_7_4};

/*****************************
*           LOAD IMAGE              *
*****************************/
void CVerifyDlg::OnLoad()
{
    CFileDialog dlg( TRUE, "bmp", 0, OFN_HIDEREADONLY |OFN_OVERWRITEPROMPT,
    "Image File (*.BMP,*.JPG,*.JPE,*.TIF)|*.bmp;*.jpg;*.jpe;*.tif|All Files (*.*)|*.*||", this);
    if( dlg.DoModal())
    {
        GetDlgItem(IDC_FILE)->SetWindowText( dlg.GetPathName());
    }
    m_file.GetWindowText(m_curfile, MAX_PATH);
    ifstream file( m_curfile, ios::nocreate );
    if ( file.good() )
    {
        IplImage *temp=cvvLoadImage(m_curfile);
        cvvSaveImage("c:\\temp.bmp", temp);
        RECT rc;
        int width, height;
        m_image.GetClientRect(&rc);
        width =  rc.right - rc.left;
        height = rc.bottom  - rc.top;
        HBITMAP hbitmap = (HBITMAP)::LoadImage (AfxGetInstanceHandle(),
        "c:\\temp.bmp",  IMAGE_BITMAP,
        width,height, LR_LOADFROMFILE | LR_DEFAULTSIZE);
        if (hbitmap)
        m_image.SetBitmap (hbitmap);
    }
    file.close();
}

/*************************************
*           COUNT DATABASE              *
*************************************/
TCHAR szSize[30];
void CVerifyDlg::CountDatabase()
{
    int count=0;
    TCHAR db[100];
    sprintf(db, "%s%s",m_defdir,"\\db.txt");
    ifstream inp(db, ios::nocreate);
    while (!inp.eof())
```

```
    {
        TCHAR buff_file[22];
        inp.getline(buff_file, 21);
        TCHAR buff_user[27];
        inp.getline(buff_user, 26);
        TCHAR buff_titik[3];
        for (int i=0; i<12; i++)
        {
            inp.getline(buff_titik, 2);
        }
        TCHAR buff_value[17];
        inp.getline(buff_value, 16);          //Pindah baris
        for (i=0; i<12*_jkernel; i++)
        {
            inp.getline(buff_value, 16);
        }
        inp.getline(buff_value, 16);          //Pindah baris
        count++;
    }
    inp.close();
    m_nDB=count-1;
    wsprintf(szSize, " Check 0 / %d DataBase ", m_nDB);
    m_sbar.SetWindowText(szSize);
}

/*****************************
*          VERIFY IMAGE          *
*****************************/
char nama0[22]=("                ");
char nama1[22]=("                ");
char nama2[22]=("                ");
char nama3[22]=("                ");
char nama4[22]=("                ");
char nama5[22]=("                ");
char user0[22]=("               ");
char user1[27]=("                  ");
char user2[27]=("                  ");
char user3[27]=("                  ");
char user4[27]=("                  ");
char user5[27]=("                  ");
double similar[6]={0.0,0.0,0.0,0.0,0.0,0};

void CVerifyDlg::OnVerify()
{
    similar[0]=-1.0;
    similar[1]=-1.0;
    similar[2]=-1.0;
    similar[3]=-1.0;
    similar[4]=-1.0;
    similar[5]=-1.0;
    double a=0.0;
    double c=0.0;
    double d=0.0;
    double S=0.0;
    m_file.GetWindowText(m_curfile, MAX_PATH);
    ifstream file( m_curfile, ios::nocreate );
    if ( file.good() )
    {
```

```
proses( m_curfile );
CountDatabase();
int inte;
double flot;
float hasil_db[12*56], hasil_max[12*56];
int titik_db[12];
TCHAR db[100];
sprintf(db, "%s%s",m_defdir,"\\db.txt");
ifstream inp(db, ios::nocreate);
int count=0;
while (!inp.eof())
{
	char buff_file[22];
	inp.getline(buff_file, 21);
	char buff_user[27];
	inp.getline(buff_user, 26);
	char buff_titik[3];
	for (int i=0; i<12; i++)
	{
		inp.getline(buff_titik, 2);
		inte =atoi( buff_titik );
		titik_db[i]=inte;
	}
	char buff_value[17];
	inp.getline(buff_value, 16);	//Pindah baris
	for (i=0; i<12*_jkernel; i++)
	{
		inp.getline(buff_value, 16);
		flot =atof( buff_value );
		hasil_db[i]=flot;
	}
	inp.getline(buff_value, 16);	//Pindah baris
	//Get Maximum value nearest live image
	float sel1;
	for( int s=0; s<12*_jkernel; s++)
	{
		sel1=hasil_db[s]-hasil_live[s][4];
		hasil_max[s]=hasil_live[s][4];
		if (sel1<0)
		sel1=sel1* -1;
	}
	//Similarity rank 1-5
	double a=0;
	double c=0;
	double d=0;
	long double S;
	int kern=_jkernel;
	for( i=0; i<12; i++)
	{
		if ( titik_live[i]==1 && titik_db[i]==1 )
		{
			for( int j=0; j<_jkernel; j++)
			{
				d+=(hasil_db[kern*i+j]*hasil_db[kern*i+j]);
				a+=(hasil_max[kern*i+j]*hasil_db[kern*i+j]);
				c+=(hasil_max[kern*i+j]*hasil_max[kern*i+j]);
			}
		}
```

```
}
S = a /  (sqrt(c) * sqrt(d)) ;
if (S<0.9999999999999)
{
      int o;
      if (S > similar[0])
      {
            similar[5]=similar[4];
            similar[4]=similar[3];
            similar[3]=similar[2];
            similar[2]=similar[1];
            similar[1]=similar[0];
            similar[0]=S;
            for( o=0; o<21; o++)
            {
                  nama5[o]=nama4[o];
                  nama4[o]=nama3[o];
                  nama3[o]=nama2[o];
                  nama2[o]=nama1[o];
                  nama1[o]=nama0[o];
                  nama0[o]=buff_file[o];
                  user5[o]=user4[o];
                  user4[o]=user3[o];
                  user3[o]=user2[o];
                  user2[o]=user1[o];
                  user1[o]=user0[o];
                  user0[o]=buff_user[o];
            }
      }
      else
      {
            if (S > similar[1])
            {
                  similar[5]=similar[4];
                  similar[4]=similar[3];
                  similar[3]=similar[2];
                  similar[2]=similar[1];
                  similar[1]=S;
                  for( o=0; o<21; o++)
                  {
                        nama5[o]=nama4[o];
                        nama4[o]=nama3[o];
                        nama3[o]=nama2[o];
                        nama2[o]=nama1[o];
                        nama1[o]=buff_file[o];
                        user5[o]=user4[o];
                        user4[o]=user3[o];
                        user3[o]=user2[o];
                        user2[o]=user1[o];
                        user1[o]=buff_user[o];
                  }
            }
            else
            {
                  if (S > similar[2])
                  {
                        similar[5]=similar[4];
                        similar[4]=similar[3];
```

```
            similar[3]=similar[2];
            similar[2]=S;
            for( o=0; o<21; o++)
            {
                nama5[o]=nama4[o];
                nama4[o]=nama3[o];
                nama3[o]=nama2[o];
                nama2[o]=buff_file[o];
                user5[o]=user4[o];
                user4[o]=user3[o];
                user3[o]=user2[o];
                user2[o]=buff_user[o];
            }
        }
        else
        {
            if (S > similar[3])
            {
                similar[5]=similar[4];
                similar[4]=similar[3];
                similar[3]=S;
                for( o=0; o<21; o++)
                {
                    nama5[o]=nama4[o];
                    nama4[o]=nama3[o];
                    nama3[o]=buff_file[o];
                    user5[o]=user4[o];
                    user4[o]=user3[o];
                    user3[o]=buff_user[o];
                }
            }
            else
            {
                if (S > similar[4])
                {
                    similar[5]=similar[4];
                    similar[4]=S;
                    for( o=0; o<21; o++)
                    {
                        nama5[o]=nama4[o];
                        nama4[o]=buff_file[o];
                        user5[o]=user4[o];
                        user4[o]=buff_user[o];
                    }
                }
                else
                {
                    if (S >similar[5])
                    {
                        similar[5]=S;
                        for (o=0; o<21; o++)
                        {
                            nama5[o]=buff_file[o];
                            user5[o]=buff_user[o];
                        }
                    }
                }
            }
```

```
                                }
                            }
                        }
                    }
                count++;
                wsprintf(szSize, " Check %d / %d DataBase ", count-1, m_nDB);
                m_sbar.SetWindowText(szSize);
                }
        Rank();
        }
}

//Show rank
void CVerifyDlg::Rank()
{
        TCHAR szSize[300];
        RECT rc;
        int width, height;
        m_image1.GetClientRect(&rc);
        width =  rc.right - rc.left;
        height = rc.bottom  - rc.top;
        IplImage *temp;
        TCHAR db[100];
        sprintf(db, "%s%s%s",m_defdir,"\\image\\",nama0);
        ifstream file0( db, ios::nocreate );
        if ( file0.good() )
        {
                temp=cvvLoadImage(db);
                cvvSaveImage("c:\\temp.bmp", temp);
                HBITMAP hbitmap = (HBITMAP) ::LoadImage (AfxGetInstanceHandle(),
                "c:\\temp.bmp", IMAGE_BITMAP,
                width,height, LR_LOADFROMFILE | LR_DEFAULTSIZE);
                if (hbitmap)
                m_image1.SetBitmap (hbitmap);
                sprintf(szSize, "Nama File : %s", nama0);
                m_file1.SetWindowText(szSize);
                sprintf(szSize, "Nama User : %s", user0);
                m_user1.SetWindowText(szSize);
                sprintf(szSize, "Similiarity = %lf", similar[0]);
                m_sim1.SetWindowText(szSize);
        }
        file0.close();
        sprintf(db, "%s%s%s",m_defdir,"\\image\\",nama1);
        ifstream file1( db, ios::nocreate );
        if ( file1.good() )
        {
                temp=cvvLoadImage(db);
                cvvSaveImage("c:\\temp.bmp", temp);
                HBITMAP hbitmap = (HBITMAP) ::LoadImage (AfxGetInstanceHandle(),
                "c:\\temp.bmp", IMAGE_BITMAP,
                width,height, LR_LOADFROMFILE | LR_DEFAULTSIZE);
                if (hbitmap)
                m_image2.SetBitmap (hbitmap);
                sprintf(szSize, "Nama File : %s", nama1);
                m_file2.SetWindowText(szSize);
                sprintf(szSize, "Nama User : %s", user1);
                m_user2.SetWindowText(szSize);
                sprintf(szSize, "Similiarity = %lf", similar[1]);
```

```
            m_sim2.SetWindowText(szSize);
        }
        file1.close();
        sprintf(db, "%s%s%s",m_defdir,"\\image\\",nama2);
        ifstream file2( db, ios::nocreate );
        if ( file2.good() )
        {
            temp=cvvLoadImage(db);
            cvvSaveImage("c:\\temp.bmp", temp);
            HBITMAP hbitmap = (HBITMAP) ::LoadImage (AfxGetInstanceHandle(),
            "c:\\temp.bmp", IMAGE_BITMAP,
            width,height, LR_LOADFROMFILE | LR_DEFAULTSIZE);
            if (hbitmap)
            m_image3.SetBitmap (hbitmap);
            sprintf(szSize, "Nama File : %s", nama2);
            m_file3.SetWindowText(szSize);
            sprintf(szSize, "Nama User : %s", user2);
            m_user3.SetWindowText(szSize);
            sprintf(szSize, "Similiarity = %lf", similar[2]);
            m_sim3.SetWindowText(szSize);
        }
        file2.close();
}

/**********************
*            PROSES          *
**********************/
void CVerifyDlg::proses(char nama_file[20])
{
        //declare variable
        CvSize size_templ, size_source, size_result;
        int w =39, h =39;                    //Kernel declaration
        size_templ = cvSize( w, h );
        // declare image
        IplImage *source_8U_1,          *source_8U_1_copy;
        IplImage *source_roi_8U_1,     *source_roi_32F_1;
        IplImage *real_roi_32F_1,       *imag_roi_32F_1;
        IplImage *real_sqr_32F_1,       *imag_sqr_32F_1;
        IplImage *add_sqr_32F_1,        *add_sqr_8U_1;
        IplImage *source_32F_1,         *source_canny;
        source_8U_3 = cvvLoadImage( nama_file );
        size_source.width = source_8U_3->width;
        size_source.height = source_8U_3->height;
        size_result = cvSize( size_source.width-w+1, size_source.height-h+1);
        source_8U_1 = cvCreateImage (size_source, IPL_DEPTH_8U, 1);
        cvvConvertImage (source_8U_3, source_8U_1, 0);
        source_canny = cvCreateImage (size_source, IPL_DEPTH_8U, 1);
        cvCanny(source_8U_1, source_canny, 0, 255, 5 );
        //HISTOGRAM
        const int range=256;
        int arr_1[range+1], arr_2[range], arr_3[range] ;
        IplLUT lut = { range+1, NULL, NULL, NULL, IPL_LUT_LOOKUP };
        IplLUT *plut = &lut;
        for( int p=0; p<range+1; p++)
            arr_1[p]=p;
        int *parray_1 = &arr_1[0];
        lut.key = parray_1;
        for( p=0; p<range; p++)
```

```
        arr_2[p]=0;
int *parray_2 = &arr_2[0];
lut.value = parray_2;
for( p=0; p<range; p++)
        arr_3[p]=0;
int *parray_3 = &arr_3[0];
lut.factor= parray_3;
iplComputeHisto( source_8U_1, &plut );
iplHistoEqualize( source_8U_1, source_8U_1, &plut );
source_32F_1 = cvCreateImage (size_source, IPL_DEPTH_32F, 1);
cvConvertScale( source_8U_1, source_32F_1, 1, 0 );
source_8U_1_copy    = cvCloneImage( source_8U_1 );
source_roi_8U_1     = cvCloneImage( source_8U_1 );
source_roi_32F_1    = cvCloneImage( source_32F_1 );
real_roi_32F_1      = cvCloneImage( source_32F_1 );
imag_roi_32F_1      = cvCloneImage( source_32F_1 );
real_sqr_32F_1      = cvCloneImage( source_32F_1 );
imag_sqr_32F_1      = cvCloneImage( source_32F_1 );
add_sqr_32F_1       = cvCloneImage( source_32F_1 );
add_sqr_8U_1        = cvCloneImage( source_8U_1 );
// Create Point Grid
int data_point[12][2]=
{
        70,190, 115,70, 123,96, 165,50, 160,95, 210,70,
        200,100, 250,115, 160,160, 200,160, 180,190, 180,220,
};
int count=0;
//Repeat 12 sampling point
for (int j=0; j<12; j++)
{
        point.x=data_point[j][0];
        point.y=data_point[j][1];
        if (point.x+20<=size_source.width &&
        point.y+20<=size_source.height &&
        point.x-20>=0 && point.y-20>=0 )
        {
                titik_live[j]=1;
                // GABOR FILTERING
                int a,b;
                float pixel[4];
                //REPEAT 40 KERNEL
                for (int i=0; i<_jkernel; i++)
                {
                        if ((i+1)%7)
                        { a=33; b=16;}
                        else
                        { a=27; b=13;}
                        IplConvKernelFP* r_kernel=iplCreateConvKernelFP( a, a, b, b,_loop[i]);
                        IplConvKernelFP*i_kernel=iplCreateConvKernelFP(a,a,b,b,_loop[i+_jkernel]);
                        CvRect size_roi;
                        size_roi = cvRect( point.x-b, point.y-b, a, a );
                        CvSize size_image;
                        size_image = cvSize( a, a);
                        IplImage*source_roi_8U_1 = cvCreateImage( size_image, IPL_DEPTH_8U, 1);
                        cvSetImageROI( source_8U_1_copy, size_roi);
                        cvCopyImage( source_8U_1_copy, source_roi_8U_1);
                        source_roi_32F_1 = cvCreateImage (size_image, IPL_DEPTH_32F, 1);
                        cvConvertScale( source_roi_8U_1, source_roi_32F_1, 1 , 0 );
```

```
                    cvReleaseImage(&source_roi_8U_1);
                    real_roi_32F_1 = cvCreateImage (size_image, IPL_DEPTH_32F, 1);
                    imag_roi_32F_1 = cvCreateImage (size_image, IPL_DEPTH_32F, 1);
                    add_sqr_32F_1 = cvCreateImage (size_image, IPL_DEPTH_32F, 1);
                    iplMultiplySFP( source_roi_32F_1, source_roi_32F_1, (float)1.0/255.0 );
                    iplConvolve2DFP(source_roi_32F_1, real_roi_32F_1, &r_kernel, 1, IPL_SUM);
                    iplConvolve2DFP(source_roi_32F_1,imag_roi_32F_1,&i_kernel, 1, IPL_SUM ;
                    iplSquare( real_roi_32F_1, real_sqr_32F_1);
                    iplSquare( imag_roi_32F_1, imag_sqr_32F_1);
                    cvReleaseImage(&real_roi_32F_1);
                    cvReleaseImage(&imag_roi_32F_1);
                    iplAdd( real_sqr_32F_1, imag_sqr_32F_1, add_sqr_32F_1);
                    int cnt=0;
                    for (int t=-1; t<2; t++)
                    {
                        for (int u=-1; u<2; u++)
                        {
                            iplGetPixel( add_sqr_32F_1, b+t, b+u, pixel);
                            hasil_live[count][cnt]=pixel[0];
                            cnt++;
                        }
                    }
                    cvReleaseImage(&add_sqr_32F_1);
                    count++;
                }
            }
            else
            {
                titik_live[j]=0;
                for (int i=0; i<_jkernel; i++)
                {
                    for (int k=0; k<9; k++)
                    hasil_live[count][k]=0;
                    count++;                    //Count up, data ke.. kernel ke..
                }
            }
        }
        RECT rc;
        int width, height;
        m_image.GetClientRect(&rc);
        width =  rc.right - rc.left;
        height = rc.bottom  - rc.top;
        cvvSaveImage("c:\\temp.bmp", source_8U_3);
        HBITMAP hbitmap = (HBITMAP) ::LoadImage (AfxGetInstanceHandle(), "c:\\temp.bmp",
        IMAGE_BITMAP,
        width,height, LR_LOADFROMFILE | LR_DEFAULTSIZE);
        if (hbitmap)
        m_image.SetBitmap (hbitmap);
    }
```