```
// Structure LightDlg.cpp : implementation file
//
#include "stdafx.h"
#include "Structure Light.h"
#include "Structure LightDlg.h"

#include "windows.h"
#define portprint 0x378

#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <iostream.h>
#include <cv.h>
#include <cvaux.h>
#include <highgui.h>
#include <cvcam.h>

int rray1[480][640];
int rray2[480][640];

int tarray1[480][640];
int tarray2[480][640];
int i,j,c,A,B,C,D,E,F,G;
int width,height;
int moves;
int detailed1;

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

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
```

```
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CStructureLightDlg dialog

CStructureLightDlg::CStructureLightDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CStructureLightDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CStructureLightDlg)
	m_move1 = 0;
	m_detailed = 0;
	m_height = 0.0f;
	m_long = 0.0f;
	m_width = 0.0f;
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}
```

```
void CStructureLightDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CStructureLightDlg)
	DDX_Text(pDX, IDC_EDIT1, m_move1);
	DDX_Text(pDX, IDC_EDIT2, m_detailed);
	DDX_Text(pDX, IDC_HEIGHT, m_height);
	DDX_Text(pDX, IDC_LONG, m_long);
	DDX_Text(pDX, IDC_WIDTH, m_width);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CStructureLightDlg, CDialog)
	//{{AFX_MSG_MAP(CStructureLightDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_AUTO, OnAuto)
	ON_BN_CLICKED(IDC_MOVE, OnMove)
	ON_BN_CLICKED(IDC_DETAILED, OnDetailed)
	ON_BN_CLICKED(IDC_SIZE, OnSize)
	ON_BN_CLICKED(IDC_RIGHT, OnRight)
	ON_BN_CLICKED(IDC_SAVE, OnSave)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// CStructureLightDlg message handlers

BOOL CStructureLightDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
```

```
                        pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
                }
        }

        // 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

        return TRUE;  // return TRUE  unless you set the focus to a control
}

void CStructureLightDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
        if ((nID & 0xFFF0) == IDM_ABOUTBOX)
        {
                CAboutDlg dlgAbout;
                dlgAbout.DoModal();
        }
        else
        {
                CDialog::OnSysCommand(nID, lParam);
        }
}

// 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 CStructureLightDlg::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();
	}
}

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

void CStructureLightDlg::OnOK()
{

for (int a = 0;a < 150; a++)
{
	UpdateData (TRUE);

		_outp (0x378,0x01);
			Sleep(5);
		_outp (0x378,0x03);
			Sleep(5);
		_outp (0x378,0x02);
			Sleep(5);
		_outp (0x378,0x06);
			Sleep(5);
		_outp (0x378,0x04);
			Sleep(5);
		_outp (0x378,0x0C);
			Sleep(5);
		_outp (0x378,0x08);
			Sleep(5);
		_outp (0x378,0x09);
			Sleep(5);

	UpdateData (FALSE);

}
}
void CStructureLightDlg::OnAuto()
{
    int A1,B1,C1 ;
    CvCapture* capture = cvCaptureFromCAM(0);
```

```
    IplImage *img10,*gray11;
    cvGrabFrame(capture);
    img10  = cvRetrieveFrame(capture);
    gray11 = cvCloneImage( img10 );
    gray11 = cvCreateImage(cvSize(img10->width,img10->height),IPL_DEPTH_8U,1);
    gray11->origin = img10->origin;
    cvCvtColor (img10,gray11,CV_BGR2GRAY);

for(i=0;i<img10->height;i++)
{
for(j=0;j<img10->width;j++)
{
    CvScalar s;
    s = cvGet2D(gray11,i,j);
    s.val[0]=0;
    A1 = s.val[0];
    rray1[i][j] = A1;
    cvSet2D(gray11,i,j,s);
}}

cvReleaseCapture(&capture);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

for( int m=0;m<moves;m++)
{
CvCapture* capture1 = cvCaptureFromCAM(0);
IplImage *img01,*gray01=0;

cvNamedWindow("3D",CV_WINDOW_AUTOSIZE);

    cvGrabFrame(capture1);
    img01  = cvRetrieveFrame(capture1);
    gray01 = cvCloneImage( img01 );
    gray01 = cvCreateImage(cvSize(img01->width,img01->height),IPL_DEPTH_8U,1);
    gray01->origin = img01->origin;
    cvCvtColor (img01,gray01,CV_BGR2GRAY);

//////////////////////////////////////////////////////thresholding////////////////////////////////////////////////////////////

for(i=0;i<img01->height;i++)
{
for(j=0;j<img01->width;j++)
{
    CvScalar s1;
    s1 = cvGet2D(gray01,i,j);
```

```
 if (s1.val[0]<105)
 {
   s1.val[0]=0;
   cvSet2D(gray01,i,j,s1);
 }
 else
 {
   s1.val[0]=255;
   cvSet2D(gray01,i,j,s1);
 }
}}

////////////////////////////////////////////////////// thinning ////////////////////////////////////////////////////

for(i=2;i<(480-2);i++)
    {
        for(j=2;j<(640-2);j++)
        {
            CvScalar s,s1,s2,s3,s4;
            s  = cvGet2D(gray01,i-1,j);
            s1 = cvGet2D(gray01,i-2,j);
            s2 = cvGet2D(gray01,i,j);
            s3 = cvGet2D(gray01,i+1,j);
            s4 = cvGet2D(gray01,i+2,j);
            F  = (s1.val[0]+s4.val[0])+(2*(s.val[0]+s3.val[0]))+(3*s2.val[0]);
            tarray1[i][j]=F;
        }
    }
for(i=2;i<(480);i++)
    {
        for(j=0;j<(640-2);j++)
        {
            A = tarray1[i-1][j];
            B = tarray1[i-2][j];
            C = tarray1[i][j];
            D = tarray1[i+1][j];
            E = tarray1[i+2][j];

            if(C>A && C>B && C<D && C<E)
            {
              CvScalar s;
              s.val[0]=0;
              cvSet2D(gray01,i,j,s);
            }

            if(C<A && C<B && C>D && C>E)
            {
              CvScalar s;
```

```
                s.val[0]=0;
                cvSet2D(gray01,i,j,s);
            }
        }
    }
for(i=2;i<(480-2);i++)
    {
        for(j=2;j<(640-2);j++)
        {
            CvScalar s,s1,s2,s3,s4;
            s  = cvGet2D(gray01,i,j-1);
            s1 = cvGet2D(gray01,i,j-2);
            s2 = cvGet2D(gray01,i,j);
            s3 = cvGet2D(gray01,i,j+1);
            s4 = cvGet2D(gray01,i,j+2);
            G  = (s1.val[0]+s4.val[0])+(2*(s.val[0]+s3.val[0]))+(3*s2.val[0]);
            tarray2[i][j] = G;
        }
    }
for(i=0;i<(480);i++)
    {
        for(j=2;j<(640-2);j++)
        {
            A = tarray2[i][j-1];
            B = tarray2[i][j-2];
            C = tarray2[i][j];
            D = tarray2[i][j+1];
            E = tarray2[i][j+2];

            if(C>A && C>B && C<D && C<E)
            {
              CvScalar s;
              s.val[0]=0;
              cvSet2D(gray01,i,j,s);
            }
            if(C<A && C<B && C>D && C>E)
            {
              CvScalar s;
              s.val[0]=0;
              cvSet2D(gray01,i,j,s);
            }
        }
    }
```

//

```
for(i=0;i<img10->height;i++)
    {
        for(j=0;j<img10->width;j++)
        {
            CvScalar s1;
            s1 = cvGet2D(gray01,i,j);

            if (s1.val[0]<150)
            {
              s1.val[0]=0;
              B1 = s1.val[0];
              rray2[i][j] = B1;
              cvSet2D(gray01,i,j,s1);
            }
            else
            {
              s1.val[0]=255;
              B1 = s1.val[0];
              rray2[i][j] = B1;
              cvSet2D(gray01,i,j,s1);
            }
        }
    }
```

//All//

```
for(i=0;i<480;i++)
    {
        for(j=0;j<640;j++)
        {

            CvScalar s;
            s = cvGet2D(gray01,i,j);

            A1 = rray1[i][j];
            B1 = rray2[i][j];
            C1 = A1 + B1;

            if(C1>=255)
            {
              s.val[0] = 255;
              A1 = s.val[0];
              rray1[i][j] = A1;
              cvSet2D(gray01,i,j,s);
            }
```

```
            if(C1<255)
            {
              s.val[0] = 0;
              A1 = s.val[0];
              rray1[i][j] = A1;
              cvSet2D(gray01,i,j,s);
            }

        }
    }
        cvShowImage("3D",gray01);
        cvSaveImage("Z.bmp",gray01);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

for (int  a = 0;a < detailed1; a++)
    {
        UpdateData (TRUE);

                _outp (0x378,0x09);
                        Sleep(5);
                _outp (0x378,0x08);
                        Sleep(5);
                _outp (0x378,0x0C);
                        Sleep(5);
                _outp (0x378,0x04);
                        Sleep(5);
                _outp (0x378,0x06);
                        Sleep(5);
                _outp (0x378,0x02);
                        Sleep(5);
                _outp (0x378,0x03);
                        Sleep(5);
                _outp (0x378,0x01);
                        Sleep(5);

        UpdateData (FALSE);
    }

cvDestroyWindow("3D");
cvReleaseCapture(&capture1);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
for (int  a01 = 0;a01 < detailed1*moves; a01++)
    {
        UpdateData (TRUE);

                _outp (0x378,0x01);
                    Sleep(5);
                _outp (0x378,0x03);
                    Sleep(5);
                _outp (0x378,0x02);
                    Sleep(5);
                _outp (0x378,0x06);
                    Sleep(5);
                _outp (0x378,0x04);
                    Sleep(5);
                _outp (0x378,0x0C);
                    Sleep(5);
                _outp (0x378,0x08);
                    Sleep(5);
                _outp (0x378,0x09);
                    Sleep(5);

        UpdateData (FALSE);

    }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

        c=cvWaitKey(0);
        switch(c){case char (27):goto exit_main;break;}
        exit_main:;
}

void CStructureLightDlg::OnMove()
{
        UpdateData(TRUE);

        moves = m_move1;

        UpdateData(FALSE);
}

void CStructureLightDlg::OnDetailed()
{
        UpdateData(TRUE);

        detailed1 = m_detailed;

        UpdateData(FALSE);
}
```

```
void CStructureLightDlg::OnSize()
{

//////////////////////////////////////////////////////////height/////////////////////////////////////////////////////////

        int d=0,z=0,e=0;
        int i,j,a,b,c;
        int width,height,channels;
        IplImage* img = 0;
        IplImage* gray = 0;
        IplImage* img1 = 0;
        IplImage* gray1 = 0;

        img = cvLoadImage("b1.bmp",1);
        gray = cvCloneImage(img);
        gray = cvCreateImage(cvSize(img->width,img->height),IPL_DEPTH_8U,1);

        gray->origin=img->origin;
        cvCvtColor(img,gray,CV_BGR2GRAY);
        width = img->width;
        height = img->height;
        channels = img->nChannels;

for(i=0;i<480;i++)
    {
        for(j=100;j<101;j++)
        {
            CvScalar s;
            s = cvGet2D(gray,i,j);

            if(s.val[0]==255)
            {
                a=i;
            }
    }
}
for(i=0;i<480;i++)
    {
        for(j=320;j<321;j++)
        {
            CvScalar s;
            s = cvGet2D(gray,i,j);

            if(s.val[0]==255)
            {
                b=i;
                e++;
            }
```

```
        }
    }
        m_height = ((a-b)*0.2);
        UpdateData(FALSE);

////////////////////////////////////////////////////////height//////////////////////////////////////////////////////////

////////////////////////////////////////////////////////width//////////////////////////////////////////////////////////

for(i=36;i<37;i++)
    {
        for(j=0;j<640;j++)
        {
            CvScalar s;
            s = cvGet2D(gray,i,j);
            if(s.val[0]==255)
            {
                d++;
            }
        }
    }
        m_width = d*0.139;
        UpdateData(FALSE);

////////////////////////////////////////////////////////width//////////////////////////////////////////////////////////

////////////////////////////////////////////////////////long//////////////////////////////////////////////////////////

img1 = cvLoadImage("Z.bmp",1);
gray1 = cvCloneImage(img1);
gray1 = cvCreateImage(cvSize(img1->width,img1->height),IPL_DEPTH_8U,1);

gray1->origin=img1->origin;
cvCvtColor(img1,gray1,CV_BGR2GRAY);
width = img1->width;
height = img1->height;
channels = img1->nChannels;

for(i=0;i<480;i++)
    {
        for(j=495;j<496;j++)
        {
            CvScalar s;
            s = cvGet2D(gray1,i,j);
            if(s.val[0]==255)
            {
                c=i;
            }
```

```
        }}
        m_long = (((c-b)+e)*0.2);
        UpdateData(FALSE);
}
void CStructureLightDlg::OnRight()
{

int a;

for (a = 0;a < 150; a++)
    {
        UpdateData (TRUE);



                _outp (0x378,0x09);
                        Sleep(5);
                _outp (0x378,0x08);
                        Sleep(5);
                _outp (0x378,0x0C);
                        Sleep(5);
                _outp (0x378,0x04);
                        Sleep(5);
                _outp (0x378,0x06);
                        Sleep(5);
                _outp (0x378,0x02);
                        Sleep(5);
                _outp (0x378,0x03);
                        Sleep(5);
                _outp (0x378,0x01);
                        Sleep(5);

        UpdateData (FALSE);
    }
}

void CStructureLightDlg::OnSave()
{

CvCapture* capture = cvCaptureFromCAM(0);
IplImage *img10,*gray11;
cvGrabFrame(capture);
img10  = cvRetrieveFrame(capture);
gray11 = cvCloneImage( img10 );
gray11 = cvCreateImage(cvSize(img10->width,img10->height),IPL_DEPTH_8U,1);
gray11->origin = img10->origin;
cvCvtColor (img10,gray11,CV_BGR2GRAY);
```

```
for(i=0;i<img10->height;i++)
    {
        for(j=0;j<img10->width;j++)
        {
            CvScalar s;
            s = cvGet2D(gray11,i,j);
            if (s.val[0]<80)
            {
                s.val[0]=0;
                cvSet2D(gray11,i,j,s);
            }
            else
            {
                s.val[0]=255;
                cvSet2D(gray11,i,j,s);
            }
    }}
            cvSaveImage("b1.bmp",gray11);

}
```