ДОДАТОК А

Текст програми

// App.h

```
#pragma once

#pragma once
#pragma comment(lib,"cv210.lib")
#pragma comment(lib,"cvaux210.lib")
#pragma comment(lib,"cxcore210.lib")
#pragma comment(lib,"highgui210.lib")

#include "afxwin.h"
#include <afxcmn.h>
#include <afxmt.h>
#include <math.h>

#include <highgui.h>
#include <cv.h>
#include "LegoNXT.h"

#define USE_ROBOT
#define DEFAULT_COMPORT 3

class CApp :
        public CWinApp
{  public:
        BOOL InitInstance(void);
};

class llist
{public:
 CPoint curpos;
 llist *next;
 llist(){curpos=CPoint(0,0);next=NULL;}
 llist(int cp,llist *p){curpos=cp;next=p;}
};

class cell
{   public:
        CPoint curpos;
        cell *next;
        cell(){curpos=CPoint(0,0);next=NULL;}
        cell(CPoint cp,cell *n){curpos=cp;next=n;}
};

class CMainWin :
        public CFrameWnd
{   public:
        CMainWin(void);
        void OnCamera();
        void OnClose();
        DECLARE_MESSAGE_MAP()
};
```

```
struct motorCommand{
        VECT dir;
        int type;
        bool isNew;
        double objectLoc[2];
        double currentPose[4];
        CMutex wait;
};

void on_mouse( int event, int x, int y, int flags, void* param );
CPoint FindCellCv(CPoint loc);
cell *pathCv(cell *c_path);
CPoint GetNextStepCv(CPoint cp, CPoint tp);
double distCv(CPoint cp, CPoint tp);
cell *reversCv(cell *in);
void FillCellCv(int i, int j, int c);
void detect_and_draw(IplImage* img);

CPoint cellAdress(int x, int y);
UINT RobotLoop(LPVOID);
int ControlLoop(int i1);
void RobotStart();
void RobotCommand(int);
void rpl2waypoint();
cell *roadripper(cell *);
void setBusyCell(CvRect);
int min_el(double a[8]);

UINT RobotRouter(LPVOID WinObjPtr);
UINT GripperRouter(LPVOID WinObjPtr);
void pathfinder();
bool robotItself(CvPoint);
void freeInitialPos(CPoint);

typedef struct
{float x;
 float y;
 float fi;
 int txn_type;
} fPoint;

typedef struct areaObject
{ fPoint centerPoint;
 bool state;
 float area;
 char name[255];
 fPoint currentPosition;
 fPoint finalPosition;
}pAreaObject;

enum{STOP, ROTATE, ROTATE_MOVE, TARGET_VECTOR, WAYPOINT, RELOCALIZE,
DISCONNECT,CURPOS,SET_GRIPPER, BACKUP};
```

// App.cpp

```
#include <afxwin.h>
#include <math.h>

#include "App.h"
#include "LegoNxt.h"
#include "resource.h"

extern CvCapture *capture;
extern CvFont im_font;
extern IplImage *img;
extern LegoNXT robot;
bool fcv=false;
bool r_init=false;

int maxX,maxY;
int p1,q1,p2,q2;
CPoint cp[33][25],sp,ep,curPoint,prevPoint;
bool bp[33][25],d=true,not_subrout=true;
bool vp[33][25];
cell *pl=new cell;
cell *r_pl=new cell;
bool setpoint=false;
CvScalar selColor;
int harrist=0,harrist2=0;;
int redt=0,greent=0,bluet=0;

CMainWin::CMainWin(void)
{Create(NULL,"Adaptive decision-maker",WS_OVERLAPPEDWINDOW);
 int rc=MessageBox("Will you control NXT","NXT Control",MB_ICONQUESTION|MB_YESNO);
 if(rc==IDYES){ if(robot.init(DEFAULT_COMPORT))
{   AfxMessageBox("Connected",MB_ICONINFORMATION);
          AfxBeginThread(RobotLoop,this);
          r_init=true;
  }
else AfxMessageBox("No connection to NXT \n Emulation mode",MB_ICONERROR);
  }
 else AfxMessageBox("No connection to NXT \n Emulation mode",MB_ICONERROR);
 int resp=MessageBox("Do you want camera to use",NULL,MB_YESNO|MB_ICONQUESTION);
 if(resp==IDYES)
 {   if(capture=cvCaptureFromCAM(0))
           { cvNamedWindow("mainWin",CV_WINDOW_AUTOSIZE);
             cvNamedWindow("Contour",CV_WINDOW_AUTOSIZE);
             cvCreateTrackbar("HarrisTrack","Contour",&harrist,100,NULL);
     cvSetMouseCallback("mainWin", on_mouse,this);//
             cvInitFont(&im_font, CV_FONT_HERSHEY_PLAIN, 0.7f,0.7f,0,1,CV_AA);
             OnCamera();
           }
 }
 else MessageBox("Camera reading Error");
}

BOOL CApp::InitInstance(void)
{         m_pMainWnd = new CMainWin;
          m_pMainWnd->ShowWindow(m_nCmdShow);
          m_pMainWnd->UpdateWindow();
          return TRUE;
}
```

```
CApp App;

BEGIN_MESSAGE_MAP(CMainWin,CFrameWnd)
END_MESSAGE_MAP()

int t_counter=0;

bool f=false;

void FillCellCv(int i, int j, int c)
{if(!c)selColor=CV_RGB(192,192,192);
 else {if(c==2)selColor=CV_RGB(0,255,0);
            else if(vp[i][j])selColor=CV_RGB(255,255,0);
                else selColor=CV_RGB(255,0,0);
           }
 bp[i][j]=true;
 fcv=true;p1=cp[i][j].x-12;q1=cp[i][j].y-12;p2=cp[i][j].x+13;q2=cp[i][j].y+13;
 cvRectangle(img,cvPoint(p1,q1),cvPoint(p2,q2),CV_RGB(255,255,0),CV_FILLED);
}

cell *pathCv(cell *c_path)
{int i=curPoint.x; int j=curPoint.y;
 int i0=ep.x; int j0=ep.y;
if(curPoint==ep) return c_path;
 else if(CPoint(i,j-1)==ep)
            {       curPoint=CPoint(i,j-1);
                        cell *ac2=new cell(curPoint,c_path);
                        return ac2;   }
 else if(CPoint(i,j+1)==ep)
            {       curPoint=CPoint(i,j+1);
                        cell *ac2=new cell(curPoint,c_path);
                        return ac2;  }
 else if(CPoint(i-1,j)==ep)
            {       curPoint=CPoint(i-1,j);
                        cell *ac2=new cell(curPoint,c_path);
                        return ac2;  }
 else if(CPoint(i+1,j)==ep)
            {       curPoint=CPoint(i+1,j);
                        cell *ac2=new cell(curPoint,c_path);
                        return ac2;  }
else if(CPoint(i-1,j-1)==ep && !bp[i-1][j] && !bp[i][j-1])
            {       curPoint=CPoint(i-1,j-1);
                        cell *ac2=new cell(curPoint,c_path);
                        return ac2;  }
 else if(CPoint(i-1,j+1)==ep && !bp[i-1][j] && !bp[i][j+1])
            {       curPoint=CPoint(i-1,j+1);
                        cell *ac2=new cell(curPoint,c_path);
                        return ac2;  }
 else if(CPoint(i+1,j-1)==ep && !bp[i][j-1] && !bp[i+1][j])
            {       curPoint=CPoint(i+1,j-1);
                        cell *ac2=new cell(curPoint,c_path);
                        return ac2;  }
 else if(CPoint(i+1,j+1)==ep && !bp[i][j+1] && !bp[i+1][j])
            {       curPoint=CPoint(i+1,j+1);
                        cell *ac2=new cell(curPoint,c_path);
                        return ac2;  }
```

```
else	{CPoint cp2=curPoint;
	    curPoint=GetNextStepCv(curPoint,ep);
	prevPoint=cp2;
		cell *ac2=new cell(curPoint,c_path);
		return pathCv(ac2);
		}
}

cell *reversCv(cell *in)
{ cell *out=new cell(in->curpos,NULL);
 while(in->next)
 { in=in->next;
  out=new cell(in->curpos,out);
 }
  return out;
}

double distCv(CPoint cp, CPoint tp)
{	return sqrt((double)((cp.x-tp.x)*(cp.x-tp.x)+(cp.y-tp.y)*(cp.y-tp.y)));
}

CPoint FindCellCv(CPoint loc)
{if(loc.x<5 || loc.y<5 || loc.x>(32*25+25) || loc.y>(24*25+25))return CPoint(0,0);
 else return CPoint(loc.x/25+1,loc.y/25+1);
}

int min_el(double a[8])
{	int j=0;
	while(!a[j])j++;
	double m=a[j];
	int index=j;
	for(int i=j+1;i<8;i++)
		if(a[i]<m && a[i]){m=a[i];index=i;}
	return index;
}

CPoint GetNextStepCv(CPoint cp, CPoint tp)
{	int i=cp.x,j=cp.y;
	pl;
	int w=(int)(bp[i-1][j]+vp[i-1][j]*not_subrout);//+tmp[i-1][j]);
	int e=(int)(bp[i+1][j]+vp[i+1][j]*not_subrout);//+tmp[i+1][j]);
	int n=(int)(bp[i][j-1]+vp[i][j-1]*not_subrout);//+tmp[i][j-1]);
	int s=(int)(bp[i][j+1]+vp[i][j+1]*not_subrout);//+tmp[i][j+1]);
	int nw=(int)(bp[i-1][j-1]+vp[i-1][j-1]*not_subrout);//+tmp[i-1][j]);
	int sw=(int)(bp[i-1][j+1]+vp[i-1][j+1]*not_subrout);//+tmp[i-1][j]);
	int ne=(int)(bp[i+1][j-1]+vp[i+1][j-1]*not_subrout);//+tmp[i-1][j]);
	int se=(int)(bp[i+1][j+1]+vp[i+1][j+1]*not_subrout);//+tmp[i-1][j]);

	if(curPoint!=prevPoint)
	{	if(prevPoint==CPoint(i-1,j))w++;
		if(prevPoint==CPoint(i+1,j))e++;
		if(prevPoint==CPoint(i,j-1))n++;
		if(prevPoint==CPoint(i,j+1))s++;
	}
	if(w && e && n && s)return cp;
	if(w && e && n)return CPoint(i,j+1); // s
	if(w && e && s)return CPoint(i,j-1); // n
```

```
        if(s && n && w)return CPoint(i+1,j); // e
        if(s && n && e)return CPoint(i-1,j); // w

        double ds=distCv(CPoint(i,j+1),ep); double dn=distCv(CPoint(i,j-1),ep);
        double de=distCv(CPoint(i+1,j),ep); double dw=distCv(CPoint(i-1,j),ep);
        double dnw=distCv(CPoint(i-1,j-1),ep); double dsw=distCv(CPoint(i-1,j+1),ep);
        double dne=distCv(CPoint(i+1,j-1),ep); double dse=distCv(CPoint(i+1,j+1),ep);

        if(s && n || w && e)
        {dne=0.0;dse=0.0;dsw=0.0;dnw=0.0;}

        if(n && e)dne=0.0;
        if(s && e)dse=0.0;
        if(w && s)dsw=0.0;
        if(w && n)dnw=0.0;

        if(n)dne=dnw=dn=0.0;
        if(s)dse=dsw=ds=0.0;
        if(w)dnw=dsw=dw=0.0;
        if(e)dne=dse=de=0.0;
        if(nw)dnw=0;
        if(sw)dsw=0;
        if(ne)dne=0;
        if(se)dse=0;

        double dlist[8]={de,dne,dn,dnw,dw,dsw,ds,dse};
        int ra=min_el(dlist);
        switch(ra)
        {case 0: return CPoint(i+1,j);
         case 1: return CPoint(i+1,j-1);
         case 2: return CPoint(i,j-1);
         case 3: return CPoint(i-1,j-1);
         case 4: return CPoint(i-1,j);
         case 5: return CPoint(i-1,j+1);
         case 6: return CPoint(i,j+1);
         case 7: return CPoint(i+1,j+1);
        }
        return cp;
}

void pathfinder()
{int i=0,j=0;
 while(r_pl)
  {i=r_pl->curpos.x;j=r_pl->curpos.y;
   curPoint=CPoint(i,j);
   FillCellCv(r_pl->curpos.x,r_pl->curpos.y,1);
   vp[i][j]=true;
   not_subrout=true;
   if(r_pl->next)
            if(!bp[r_pl->next->curpos.x][r_pl->next->curpos.y])
             {r_pl=r_pl->next;not_subrout=true;
      prevPoint=curPoint;    }
                  else {cell *ac2=new cell(CPoint(i,j),NULL);
                       not_subrout=false;
                             pl=pathCv(ac2);
                       r_pl=reversCv(pl);     }
   else AfxMessageBox("Rout is gone");
```

```
  }
 }
```

//OpenCV.cpp

```
#include "App.h"

CvCapture *capture;
IplImage *img,*img2,*dst=0,*dst1=0,*dst2=0,*dst3=0,*dst4=0,*dst5=0,*dst6=0;
int x;
char str[50];CString ss;
double cont_area, cont_prev, orientation, tangens, spatial, spatial_prev;
bool bStart=true;
CvFont im_font;

CvPoint origin;
fPoint InitPoint;
CvRect selection;
CvRect track_window;

CvMoments moments;
CvSeq *contour=0;
CvSeq *sel_contour=0;
CvPoint centre_of_mass, point1, point2, text_out;
CvPoint2D32f cvCurPoint;
extern CPoint sp,ep,cp;
extern bool setpoint;
extern CPoint curPoint;
extern bool bp[33][25],vp[33][25],fcv,r_init;
extern cell *pl,*r_pl;
extern int p1,q1,p2,q2;
extern CvScalar selColor;
extern LegoNXT robot;
extern motorCommand cmd;
extern int harrist,harrist2,redt,greent,bluet;
int select_object = 0;
int track_object = 0;
char msg[100];
const char *cascade_name="nrobot.xml";

void CMainWin::OnCamera()
{harrist=0;
 int lk=0;
 bool firstFrame=true;
 while(1)
 {CvMemStorage *storage=cvCreateMemStorage(0);
  img=cvQueryFrame(capture);
  cvFlip(img,NULL,-1);
  int contour_counter=0;
  {dst=cvCreateImage(cvGetSize(img),IPL_DEPTH_32F,3);
   dst1=cvCreateImage(cvGetSize(img),IPL_DEPTH_8U,3);
   dst2=cvCreateImage(cvGetSize(img),IPL_DEPTH_8U,1);
   dst3=cvCreateImage(cvGetSize(img),IPL_DEPTH_8U,1);
   dst4=cvCreateImage(cvGetSize(img),IPL_DEPTH_8U,3);
   dst5=cvCreateImage(cvGetSize(img),IPL_DEPTH_8U,3);
   dst6=cvCreateImage(cvGetSize(img),IPL_DEPTH_8U,1);
```

```
   firstFrame=false;
  }
  CvSeq *sel_contour = cvCreateSeq
(CV_SEQ_ELTYPE_POINT,sizeof(CvSeq),sizeof(CvPoint),storage);
  fcv=false;

  cvCvtColor(img,dst2,CV_RGB2GRAY);
  cvThreshold(dst2,dst3,1,255,CV_THRESH_BINARY_INV|CV_THRESH_OTSU);
  cvCanny(dst3,dst3,128,255);
  cvShowImage("Canny",dst3);

  detect_and_draw(img);

  CvScalar color = CV_RGB( rand()&255, rand()&255, rand()&255 );
  int contrNum=cvFindContours(dst3, storage, &contour, sizeof(CvContour), CV_RETR_LIST,
CV_LINK_RUNS);
  if(!lk || lk==10 || lk==20)
  {
    for( ; contour != 0; contour = contour->h_next )
       {              cvMoments(contour, &moments, 0 );
                              centre_of_mass.x=(int) (moments.m10/moments.m00);
                              centre_of_mass.y=(int) (moments.m01/moments.m00);
                              if(!centre_of_mass.x && !centre_of_mass.y &&
!cvCheckContourConvexity(contour))continue;
                              double l=cvArcLength(contour,CV_WHOLE_SEQ,-1);
                              double s=cvContourArea(contour);
                              double itRobot=cvPointPolygonTest(contour,cvCurPoint,1);
                              if(fabs(s)<400 && fabs(s)>50 && l>150 && l<1000 && itRobot<-50)
                              {CvRect cont=cvBoundingRect(contour,0);

cvRectangle(dst4,cvPoint(cont.x,cont.y),cvPoint(cont.x+cont.width,cont.y+cont.height),CV_RGB(0,0,25
5),CV_FILLED);
                               setBusyCell(cont);
          cvSeqPush(sel_contour,&centre_of_mass);
                              orientation = atan(2*moments.m11/(moments.m20 - moments.m02));
                              point1.x = centre_of_mass.x+15*cos(orientation);
                              point1.y = centre_of_mass.y+15*sin(orientation);
                              point2.x = centre_of_mass.x-15*cos(orientation);
                              point2.y = centre_of_mass.y-15*sin(orientation);
                              cvLine (dst4, point1, point2,CV_RGB(255,255,255), 4, CV_AA);

                              point1.x = centre_of_mass.x+15*cos(orientation);
                              point1.y = centre_of_mass.y+15*sin(-orientation);
                              point2.x = centre_of_mass.x-15*cos(orientation);
                              point2.y = centre_of_mass.y-15*sin(-orientation);
                              cvLine (dst4, point1, point2,CV_RGB(255,255,255), 4, CV_AA);

                              cvCircle(dst4,centre_of_mass, 5,CV_RGB(0,0,255),2);

                              cvDrawContours( dst4, sel_contour, color, color, -1,1, 8 );
                              text_out.x=point1.x;
                              text_out.y=point1.y;
                              contour_counter++;
                              ss.Format(_T("%f %f %d %d"),s,l,centre_of_mass.x,centre_of_mass.y);
                              if(ss)cvPutText(dst4,ss, text_out, &im_font,CV_RGB(255,255,255));
                             }
                             cvShowImage("Contour",dst4);
```

```
        }
            }
    cvShowImage("mainWin",img);
            cvReleaseMemStorage(&storage);
            cvReleaseImage(&dst);
cvReleaseImage(&dst1);cvReleaseImage(&dst2);cvReleaseImage(&dst3);
            cvReleaseImage(&dst4);cvReleaseImage(&dst5);cvReleaseImage( &dst6);
    x=cvWaitKey(10);
            if(x=='x'){RobotCommand(2);break;}
            lk++;
            if(lk==30){for(int i=0;i<26;i++)for(int j=0;j<20;j++)bp[i][j]=false;lk=0;}
 }
 cvReleaseCapture(&capture);
 cvDestroyWindow("mainWin");
 cvDestroyWindow("Laplace");
 cvDestroyWindow("Canny");
 cvDestroyWindow("Contour");
}

void CMainWin::OnClose()
{           cvReleaseCapture(&capture);
            cvDestroyWindow("mainWin");
            cvDestroyWindow("Laplace");
            cvDestroyWindow("Canny");
            DestroyWindow();
}

void on_mouse( int ev, int x, int y, int flags, void* param )
{
            CMainWin *ptr=(CMainWin*)param;
            if( !img ) return;
    if( img->origin ) y = img->height - y;
    switch(ev)
    {
    case CV_EVENT_LBUTTONDOWN:
                        if(bStart)
                        {sp=FindCellCv(CPoint(x,y));
                         freeInitialPos(sp);
                         InitPoint.x=1.47-sp.x/32.0*1.47;
                         InitPoint.y=sp.y/24.0*1.1;
                         cvCurPoint.x = (float)x; cvCurPoint.y = (float)y;
                         RobotCommand(1);
                         bStart=false;
                         if(sp==CPoint(0,0))AfxMessageBox("StartPoint isn't set");
                         else {curPoint=sp;
                                     vp[sp.x][sp.y]=true;
                                     setpoint=true;
                                    }
                        }
       break;
    case CV_EVENT_RBUTTONDOWN:
                        sp.x = (1.47 - cmd.currentPose[0])*32.0/1.47; sp.y = cmd.currentPose[1] * 24.0/1.1;
                        wsprintf(msg,"(%d %d)",x,y);
                        if(setpoint)
                        {
                        ep=FindCellCv(CPoint(x,y));
                        if(ep==CPoint(0,0))AfxMessageBox("EndPoint isn't set");
```

```
                else {cell *ac2=new cell(curPoint,NULL);
                        pl=pathCv(ac2);
                        r_pl=reversCv(pl);
                        r_pl=roadripper(r_pl);
                        while(pl){FillCellCv(pl->curpos.x,pl->curpos.y,2);pl=pl->next;}
                        if(r_init)AfxBeginThread(RobotRouter,NULL);
                        for(int i=1;i<33;i++)
                        for(int j=1;i<25;i++)    vp[i][j]=false;
                        sp=ep;
          }         }
                else AfxMessageBox("StartPoint isn't set");
      break;
   }      }


void setBusyCell(CvRect BusyArea)
{          for(int i=BusyArea.x/25;i<=(BusyArea.x+BusyArea.width)/25+1;i++)
           for(int j=BusyArea.y/25;j<=(BusyArea.y+BusyArea.height)/25+1;j++)
           FillCellCv(i,j,3);
}

void freeInitialPos(CPoint cp)
{for(int i=cp.x-4;i<cp.x+4;i++)
 for(int j=cp.y-4;j<cp.y+4;j++)
            bp[i][j]=false;
}

bool robotItself(CvPoint centre_of_mass)
{bool res=false;
 for(int i=curPoint.x-1;i<curPoint.x+1;i++)
 for(int j=curPoint.y-1;j<curPoint.y+1;j++)
            if(CPoint(i,j)==FindCellCv(CPoint(centre_of_mass.x,centre_of_mass.y)))res=true;
 return res;
}

void detect_and_draw( IplImage* img )
{  static CvMemStorage* storage = 0;
   static CvHaarClassifierCascade* cascade = 0;
   int scale = 1;
   CvPoint pt1, pt2;
   int i;
   cascade = (CvHaarClassifierCascade*)cvLoad( cascade_name, 0, 0, 0 );
   if( !cascade )
   {  fprintf( stderr, "ERROR: Could not load classifier cascade\n" );
      return;
   }
   storage = cvCreateMemStorage(0);
   cvNamedWindow( "result", 1 );
   cvClearMemStorage( storage );
   if( cascade )
   {   CvSeq* hands = cvHaarDetectObjects( img, cascade, storage,
                                   1.1, 2, CV_HAAR_DO_CANNY_PRUNING,
                                   cvSize(40, 40) );
      for( i = 0; i < (hands ? hands->total : 0); i++ )
      {  CvRect* r = (CvRect*)cvGetSeqElem( hands, i );
         pt1.x = r->x*scale;
         pt2.x = (r->x+r->width)*scale;
```

```
            pt1.y = r->y*scale;
            pt2.y = (r->y+r->height)*scale;

            cvRectangle( img, pt1, pt2, CV_RGB(230,20,232), 3, 8, 0 );
                                    std::cout<<"I Find YOU >>>__<<< "<<i<<endl;
        }   }   }


IplImage *rotateImage(IplImage *s0, double angle)
{CvPoint2D32f src_center;
 src_center=cvPoint2D32f(s0->height/2.0f,s0->width/2.0f);
 CvMat *rot_map=NULL;
cv2DRotationMatrix(src_center,angle,1.0,rot_map);
 cvWarpAffine(&s0,&img,rot_map);
 return img;
 cvShowImage("result", img);
 }
```

ДОДАТОК Б

Демонстраційний матеріал

Формат	Позиція.	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кільк.	Примітки
			Текстові документи		
А4	1	ГЮІК 503320.006ПЗ	Пояснювальна записка	65	с.
			Інші документи		
А4	2		Текст програми	11	с.
А4	3		Демонстраційний графічний матеріал	12	с.

					ГЮІК 503320.006ВД
Змін.	Арк	Номер докум.	Підпис	Дата	
Розроб.		Кусков А.В.			Розробка системи автоматизації для управління мобільною платформою підвищеної прохідності Відомість кваліфікаційної роботи
Переві		Аллахверанов Р.Ю.			
Н.конт		Стародубцев М.Г			
Затв.		Невлюдов І.Ш.			

Літера	Арку	Аркуш
Н	1	1

Кафедра КІТАР ХНУРЕ