

Папка содержит распознаватель меток MarkRecog 3.0 для ОС Windows XP и старше:

== Сборка библиотеки MarkRecogLib.dll ==

1. Установить Visual Studio 2010 (можно Express: )

2. Установить OpenCV 2.4.2 (<http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.2/>)  
например, в папку D:\SDK\opencv242

3. Создать переменную среды SDK\_OPENCV\_242 на папку установки OpenCV 2.4.2 (например, D:\SDK\opencv242) - установить можно через Мой компьютер-Свойства-Расширенные-Переменные среды-создать новую переменную

4. Открыть .\src\MarkRecogLib.sln и собрать проект

== Пример работы с библиотекой (C++, на базе Qt SDK - QImage) ==

typedef std::list<cv::Point3f> tMarks; // x,y - center of circle, z - radix

QString process(QPainter& painter, const QPixmap& pix, int minR, int maxR, double thres, double thres2, float accuracy,

tMarks& marks, tMarks::const\_iterator& res, const QString& shotPath, bool bShow)

{

tMarkRecogParams params;  
params.size = sizeof(tMarkRecogParams);  
params.minR = minR;  
params.maxR = maxR;  
params.thres = thres;  
params.thres2 = thres2;  
params.accuracy = accuracy;  
params.shotPath = shotPath.toAscii().constData();  
params.bShow = int(bShow);

QImage src = pix.toImage();  
QImage img = src.convertToFormat(QImage::Format\_RGB32);  
int dwPitch = img.bytesPerLine()/4;  
int height = img.height(), width = img.width(), pitch = width;  
std::unique\_ptr<unsigned char[]> data(new unsigned char[pitch\*height]);  
QRgb \*psrc = reinterpret\_cast<QRgb\*>(img.bits());  
for(int y=0; y<img.height(); y++)  
{  
    for(int x=0; x<dwPitch; x++)  
    {  
        QRgb rgb = psrc[y\*dwPitch + x];  
        unsigned char i = static\_cast<unsigned char>(qGray(rgb));  
        data[y\*pitch + x] = i;  
    }  
}

const tMarkRecogResults\* mres = MarkRecog(data.get(), width, height, pitch, &params);  
if(mres->status == MRSTATUS\_ERROR)  
{  
    QString ret(mres->errorMessage);  
    MarkRecogResultsRelease(mres);  
    return ret;  
}

for(unsigned n = 0; n<mres->num; n++)  
    marks.push\_back(cv::Point3f(mres->marks[n].x, mres->marks[n].y, mres->marks[n].r));  
res = marks.begin();  
int workTime = mres->time;

MarkRecogResultsRelease(mres);

```

        painter.setPen(Qt::darkGreen);
for(tMarks::const_iterator i = marks.begin(); i!= marks.end(); i++)
{
    float x = i->x, y = i->y, r = i->z;
    if(i==res)
        painter.setPen(Qt::darkRed);
    painter.drawLine(QPointF(x,y-r),QPointF(x,y+r));
    painter.drawLine(QPointF(x-r,y),QPointF(x+r,y));
    painter.drawEllipse(QPointF(x,y),r,r);
    if(i==res)
        painter.setPen(Qt::darkGreen);
}

    return QString();
}

```