DUO Qt Samples

Overview

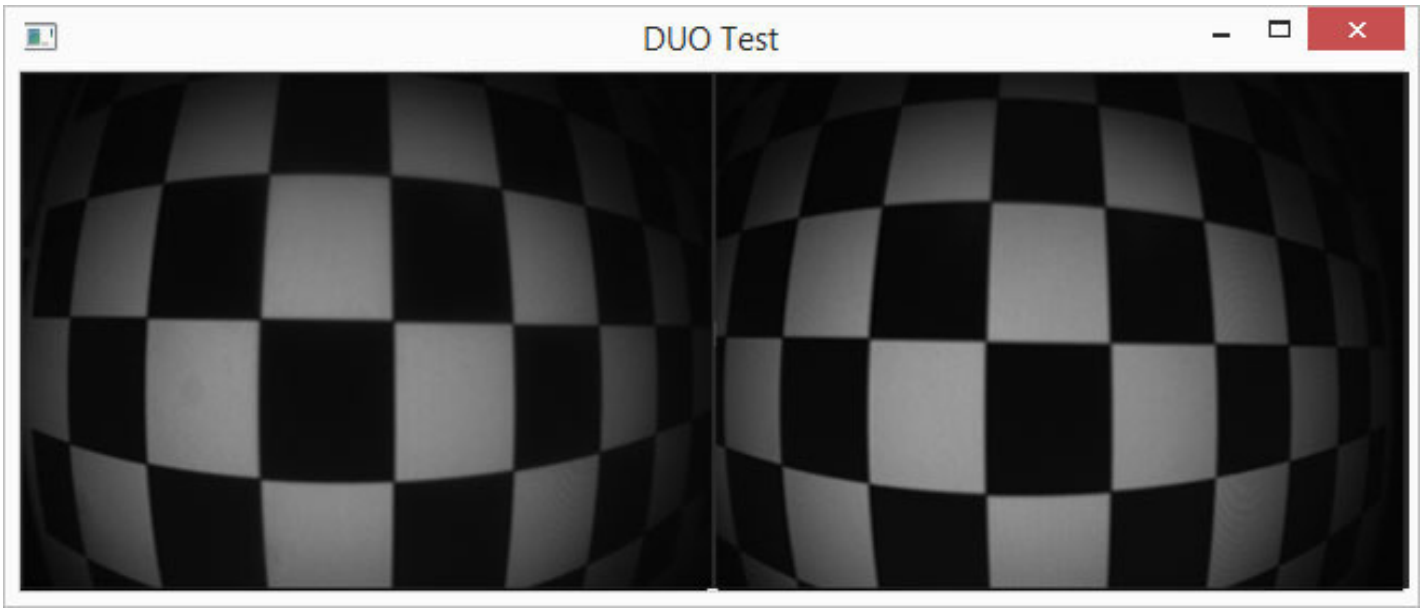
In this article we outline how to use the DUOLib with Qt 5. We initialize the DUO device and display the resulting captured frames in a standard Qt `QMainWindow` and `QWidget` classes. We use the OpenCV `Mat` class as a helper to convert the color space and display the images frames using `QImage` by implementing the `paintEvent` event.

Prerequisites

- DUOLib.lib & DUOLib dynamic library that match your compiler and architecture (x86/x64)
- Qt v5.1.1+
- OpenCV v2.4.7+
- QTCreator

Sample 01

With the DUO SDK we provide an example project to compile an application for the usage of DUO with OpenCV and Qt. This project can be found in your user directory `/user/DUO3D/Developers/Samples/Qt` or `DUO_SDK/Developers/Samples/Qt` .



Compiling

- Build the QtDUO project using Qmake or QtCreator.
- To run outside of QtCreator make sure the DUOLib.dll in the target release/debug folder.

Files

- `DUOLib.h` - DUOLib API include file.
- `globals.h` - Helper functions for console output and debugging.
- `main.cpp` - Boilerplate code for initializing the MainWindow class.
- `mainwindow.cpp` - Application main window class implementation.
- `mainwindow.h` - Application main window class declaration.
- `QtDUO.pro` - QtDUO QMake project file.

Implementation

Lets begin by reviewing the header class which contains the logic to display the images; `ImageObject` class which extends `QWidget` . Also note the `MainWindow` class which extends `QMainWindow` and is used for the base of the application.

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include
#include "DUOLib.h"

#include
using namespace cv;

class ImageOutput : public QWidget
{
    Q_OBJECT
public:
    ImageOutput()
    {
        setMinimumSize(1, 1);
        _image = QImage(QSize(1,1), QImage::Format_RGB888);
        _image.fill(Qt::black);
    }
public Q_SLOTS:
    // Mat imag must be BGR image
    void setImage(const Mat3b &image)
    {
        QMutexLocker lock(&_mutex); // Lock to assure access
        _image = QImage(image.data, image.cols, image.rows, QImage::Format_RGB888);
        update();
    }
private:
    void paintEvent(QPaintEvent *event)
    {
        QMutexLocker lock(&_mutex);
        QPainter painter(this);
        painter.setRenderHint(QPainter::SmoothPixmapTransform, true);
        painter.drawPixmap(event->rect(), QPixmap::fromImage(_image));
    }
private:
    QImage _image;
    QMutex _mutex;
```

```
};

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    MainWindow(QWidget *parent = 0);
    ~MainWindow();
    void closeEvent(QCloseEvent *);

private:
    static void CALLBACK newFrameCb(const PDUOFrame pFrameData, void *pUserData)
    {
        ((MainWindow *)pUserData)->onNewFrame(pFrameData);
    }
    void onNewFrame(const PDUOFrame pFrameData);

private:
    DUOInstance _duo;
    ImageOutput *_img[2];
    Mat _leftRGB, _rightRGB;
};

#endif // MAINWINDOW_H
```

Moving into the implementation where we initialize the DUO device and in the `onNewFrame` callback we convert the grayscale image into color image suitable for the display and then emit an event which calls the `setImage` method declared in our header.

---

mainwindow.cpp

```
#include "globals.h"
#include "mainwindow.h"

MainWindow::MainWindow(QWidget *parent): QMainWindow(parent),
_duo(NULL)
{
    resize(320*2, 240);
    setWindowTitle("DUO Test");
    setGeometry(QStyle::alignedRect(Qt::LeftToRight, Qt::AlignCenter, size(),
                                    qApp->desktop()->availableGeometry()));

    QSplitter *hs = new QSplitter();
    hs->addWidget(_img[0] = new ImageOutput());
    hs->addWidget(_img[1] = new ImageOutput());
    setCentralWidget(hs);
```

```
    DUOResolutionInfo ri;
    if(EnumerateResolutions(&ri, 1, 320, 240, DUO_BIN_HORIZONTAL2 + DUO_BIN_VERTICAL2, 60))
    {
        Print("[%dx%d], [%f-%f], %f, [%d]", ri.width, ri.height, ri.minFps, ri.maxFps, ri.fps, ri.bi
nning);

        if(OpenDUO(&_duo))
        {
            char buf[256];
            GetDUOSerialNumber(_duo, buf);       Print("Serial Number: %s", buf);
            GetDUOFirmwareVersion(_duo, buf);    Print("Firmware Version: v%s", buf);
            GetDUOFirmwareBuild(_duo, buf);      Print("Firmware Build Time: %s", buf);
            Print("Library Version: v%s", GetLibVersion());
            Print("-----------------------------------------------");

            SetDUOResolutionInfo(_duo, ri);
            uint32_t w, h;
            GetDUOFrameDimension(_duo, &w, &h);  Print("Frame Dimension: [%d, %d]", w, h);

            StartDUO(_duo, newFrameCb, this);
            SetDUOLedPWM(_duo, 30);
            SetDUOGain(_duo, 0);
            SetDUOExposure(_duo, 50);
            SetDUOVFlip(_duo, true);
        }
    }
}

MainWindow::~MainWindow()
{
    if(_duo) CloseDUO(_duo);
}

void MainWindow::closeEvent(QCloseEvent *)
{
    if(_duo) StopDUO(_duo);
}

void MainWindow::onNewFrame(const PDUOFrame pDFrame)
{
    Mat left(Size(pDFrame->width, pDFrame->height), CV_8UC1, pDFrame->leftData);
    Mat right(Size(pDFrame->width, pDFrame->height), CV_8UC1, pDFrame->rightData);
    cvtColor(left, _leftRGB, COLOR_GRAY2BGR);
    cvtColor(right, _rightRGB, COLOR_GRAY2BGR);
    Q_EMIT _img[0]->setImage(_leftRGB);
    Q_EMIT _img[1]->setImage(_rightRGB);
}
```

Brining it all together, we use the following QMake project to properly link against the DUOLib and OpenCV. This project file is provided for Windows but with a few small changes it could be adjusted to target OSX or Linux.

QtDUOSample.pro

```
TARGET = QtDUO
TEMPLATE = app

QT += core gui widgets

CONFIG += console
#CONFIG -= app_bundle

DEFINES += SHOW_DEBUG_OUTPUT

HEADERS += DUOLib.h globals.h mainwindow.h
SOURCES += main.cpp mainwindow.cpp

LIBS += -L$$PWD
LIBS += -lDUOLib

# OpenCV dependency
INCLUDEPATH += C:\OpenCV\2.4.7.2\include
!contains(QMAKE_TARGET.arch, x86_64) {
    LIBS += -LC:\OpenCV\2.4.7.2\x86\vc10\lib
} else {
    LIBS += -LC:\OpenCV\2.4.7.2\x64\vc10\lib
}
LIBS += -lopencv_core247 \
        -lopencv_highgui247 \
        -lopencv_imgproc247 \
        -lopencv_features2d247 \
        -lopencv_calib3d247
```

Summary

With this example we aim to showcase a reliable method for capturing and displaying DUO image frame data using OpenCV as a helper. We provide a standard Qt ".pro" project which allows for easy compilation with the latest Qt development tools.

Resources

Related

- DUO API
- DUO SDK
- DUO Developers
- DUO Devices
- DUO Downloads