Detecting motion of a coloured object from camera viewfinder

This article explains how to process images captured from QCamera to track a coloured object, and then to determine if its movement matches a horizontal swipe gesture.

Note: This is an entry in the PureView Imaging Competition 2012Q2

Introduction

This article demonstrate how to:

1. Process a raw image captured from `QCamera` to track a particular coloured object
2. Detect a gesture based on coloured object's movement
3. Process the detected gesture

Here is demo of my sample application from N9, but we can easily run it on Symbian device as well, as it does not use any hardware specific feature.

The media player is loading...

Getting raw image from QCamera

The article MeeGo Camera VideoSurface manipulation shows how to get a raw image from `QCamera`. This article uses similar code described in this blog post.

Tracking coloured object

This section describes a simple algorithm for detecting some predefined colour in an image. Note that if the image contains multiple object with the colour the algorithm returns a rectangle which covers all objects, not an individual rectangle of each object.

The image is first reduced to half the size in order to reduce the number of pixels that need to be processed. This is acceptable because we are not interested in the picture detail, only the presence of th colour.

To detect colour in image, we convert the image from RGB colour spec to HSV (it's easier to process HSV to detect colour). After conversion to HSV colour spec, we then convert the image to black and white - the black portion is the detected object and everything else is whilte. After getting this image we just need to scan image to find area of black portion of image.

So now we have coordinate of coloured object which we are detecting.

The following code implements the logic described above. The code combines the process of converting the image to black and white and detecting the black portion of image.

```
QRect ColorMotionDetector::detectColor( const QImage& origImage)
{
    //reduce size of image
    QImage image(origImage);
    image = image.scaled(QSize(320,240));

    emit originalImage(image);

    //rectangle of detected colored object
    int maxX = -1;
    int minX = 99999;
    int maxY = -1;
    int minY =  99999;

    int width = image.width();
    int height = image.height();
```


(C) Copyright Nokia 2014. All rights reserved.

```
    bool detected = false;

    //black and white image
    QImage converted(image.size(),image.format());

    for (int y = 0; y< height; ++y ) {
        for( int x = 0; x < width; ++x ) {
            //convert individual pixel to HSV from RGB
            QRgb pixel = image.pixel(x,y);
            QColor color(pixel);
            color = color.toHsv();


            //default whitel color for other object
            QRgb newPixel = qRgb(255, 255, 255);

            //detecting red color
            if( color.hue() >= 0 && color.hue() <= 22
                    && color.saturation() <= 255 && color.saturation() >= 240
                    && color.value() <= 255 && color.value() >= 100 ) {

                detected = true;

                if( x > maxX ) {
                    maxX = x;
                } else if( x < minX )  {
                    minX = x;
                }

                if( y > maxY ) {
                    maxY = y;
                } else if( x < minY )  {
                    minY = y;
                }

                //black color for detected object
                newPixel = qRgb(0, 0, 0);
            }
            converted.setPixel(x,y,newPixel);
        }
    }

    QRect rect;
    if( detected) {
        rect = QRect(minX,minY, maxX - minX, maxY-minY );

        //drawing red rectangle around detected object
        QPainter painter( &converted );
        painter.setPen(QPen(Qt::red));
        painter.drawRect(rect);
        painter.end();
    }
    emit processedImage(converted);

    return rect;
}
```


(C) Copyright Nokia 2014. All rights reserved.

Detecting swipe gesture

This section shows how we track the movement of the coloured part of the image to determine if it is some kind of gesture. This example detects a horizontal swipe gesture, but we can easily extend it to detect vertical or diagonal swipes.

We will use following logic to detect swipe gesture:

1. As colour detection code returns position of tracked object, We compare this new position with its old position.
2. If there is any progress in motion of object, we add difference of x coordinate to total progess made. In case of no progress, we discard whole gesture and reset variable that keep track of motion.
3. While doing so if we detect certain amount of movement in particular direction, we decide if gesture was left swipe or right swipe using difference in position of object and reset the variables.

Following code implement above logic.

```
Gesture ColorMotionDetector::detectGesture(QRect rect) {

    //not valid rectangle, mean no object detected
    if( !rect.isValid()) {
        mLastRect = QRect();
        mXDist = 0;
        return Invalid;
    }

    //there is no previous cordinate, store rect
    if( !mLastRect.isValid() ) {
        mLastRect = rect;
        mXDist= 0;
        return Invalid;
    }

    Gesture gesture = Invalid;
    int x = rect.x();
    int lastX = mLastRect.x();
    int diff = lastX - x;

    mLastRect = rect;
    //check if there is certain amount of movement
    if( qAbs( diff ) > 10 ) {
        //there is movement in x direction, store amout of movement in total movement
        mXDist += diff;
        qDebug() << "mXDist="<< mXDist <<":diff="<<diff <<":lastx="<< lastX << ":x=" <<
x;

        //x motion match to amount required for perticular gesture
        //check if motion of let to right or right to left
        if( mXDist >  150 ) {
            qDebug() << "Right horizontal swipe detected..." << mXDist;
            mXDist = 0;
            gesture = SwipeRight;
        } else if ( mXDist < -150 ) {
            qDebug() << "Left horizontal swipe detected..." << mXDist;
            mXDist = 0;
            gesture = SwipeLeft;
        }
    } else {
        //discard the gesture
```


(C) Copyright Nokia 2014. All rights reserved.

```
        mXDist = 0;
        mLastRect = QRect();
    }
    return gesture;
}
```

Putting all together

Now we have code that detect coloured object and code that detect gesture. Following code shows how those function are used together.

```
//detection motion from captured image from camera
void ColorMotionDetector::detectMotion( const QImage& image) {

    QRect rect = detectColor( image);
    Gesture gesture = detectGesture( rect );

    if( gesture != Invalid ) {
        emit gestureDetected( gesture );
    }
}
```

Following is vary simple gesture handler, which just print which handled gesture.

```
void MyWidget::gestureDetected( Gesture gesture) {

    if( gesture ==  SwipeLeft) {
        mSwipeLabel->setText("Left swipe");
    } else if( gesture == SwipeRight) {
        mSwipeLabel->setText("Right swipe");
    }
}
```

Summary

This article describes process to detect coloured object in image and how to detect gesture from motion of object.

You can download my application's code from here.

Hope you enjoyed the article.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.