饺子皮图像编辑器

学号：2113157 姓名：李响

在本次 C++大作业中我开发了一款名为“饺子皮图像编辑器”的项目，它实现了图像处理与交互功能。在此，首先将会探讨图像交互与处理功能的实现，然后结合个人的收获和体验进行总结。

图像交互功能的实现贯穿于本程序的整体架构，由多个函数组成，包括 boundingRect()、paint()、mousePressEvent()、mouseMoveEvent()、mouseReleaseEvent()、wheelEvent()和setQGraphicsViewWH()等。这些函数的实现使得图像能够进行自适应展示，用户也能够通过鼠标操作对图像进行拖动和缩放。

图像处理功能，其实现了图像的颜色调、亮度、饱和度的调整，以及模糊和锐化等功能。这些功能的实现主要依靠三个核心算法：饱和度增强算法、模糊算法和锐化算法。

一、图像交互功能及其实现

图像编辑器的交互功能在文件 imagewidget 中通过一系列函数实现，其中包括`boundingRect()`函数、`paint()`函数、`mousePressEvent()`、`mouseMoveEvent()`、`mouseReleaseEvent()`函数，以及`wheelEvent()`函数和`setQGraphicsViewWH()`函数。

·boundingRect()`函数返回图片的大小和位置信息，使得程序可以获取图片的尺寸和位置数据。

```
QRectF ImageWidget::boundingRect() const
{
    return QRectF(-m_pix.width() / 2, -m_pix.height() / 2,
                  m_pix.width(), m_pix.height());
}
```

·paint()`函数在场景中绘制图片，这是实现图片展示的主要函数。

```
void ImageWidget::paint(QPainter *painter, const QStyleOptionGraphicsItem *,
                  QWidget *)
{
    painter->drawPixmap(-m_pix.width() / 2, -m_pix.height() / 2, m_pix);
}
```

·mousePressEvent()`、`mouseMoveEvent()`和`mouseReleaseEvent()`函数实现了图片的拖动功能，使得用户可以方便地在界面上移动图片。

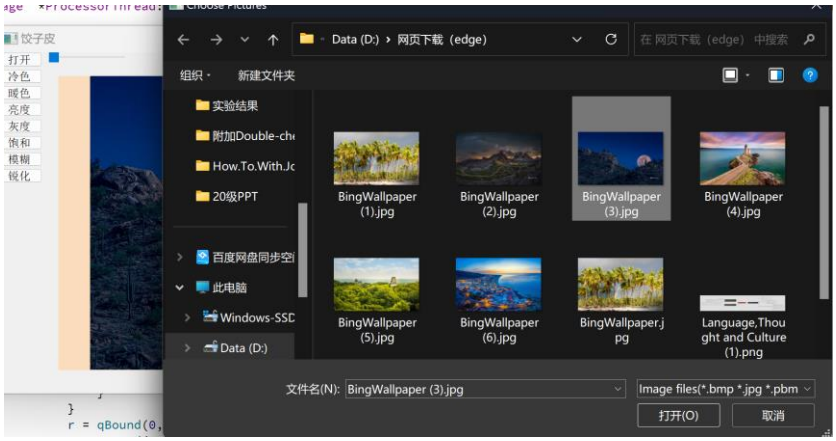
```
void ImageWidget::mousePressEvent(QGraphicsSceneMouseEvent *event)
{
    if(event->button()== Qt::LeftButton)
    {
        m_startPos = event->pos();//鼠标左击时，获取当前鼠标在图片中的坐标，
        m_isMove = true;//标记鼠标左键被按下
    }
    else if(event->button() == Qt::RightButton)
    {
        ResetItemPos();//右击鼠标重置大小
    }

}

void ImageWidget::mouseMoveEvent(QGraphicsSceneMouseEvent *event)
{
    if(m_isMove)
    {
        QPointF point = (event->pos() - m_startPos)*m_scaleValue;
        moveBy(point.x(), point.y());
    }
}

void ImageWidget::mouseReleaseEvent(QGraphicsSceneMouseEvent *)
{
    m_isMove = false;//标记鼠标左键已经抬起
}
```

·wheelEvent()`函数响应鼠标滚轮事件，用于缩放图片，并使图片能够围绕鼠标中心点进行缩放。

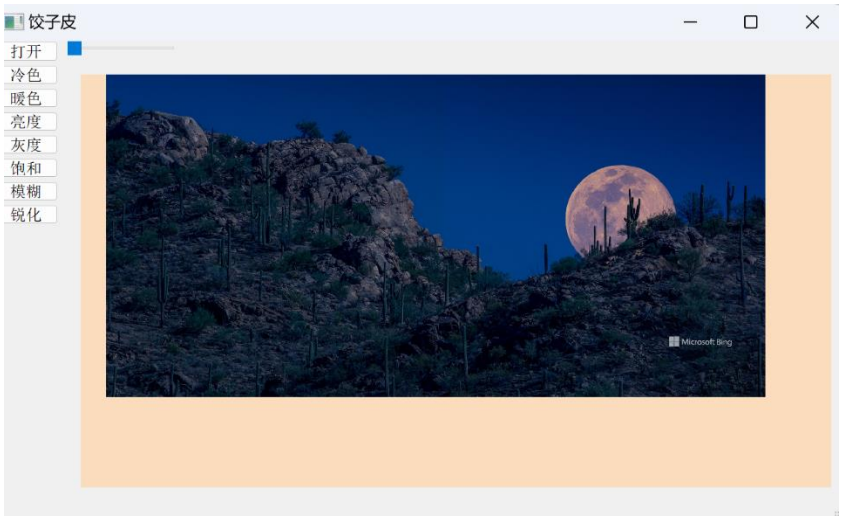
```
void ImageWidget::wheelEvent(QGraphicsSceneWheelEvent *event)//鼠标滚轮事件
{
    if((event->delta() > 0)&&(m_scaleValue >= 50))//最大放大到原始图像的50倍
    {
        return;
    }
    else if((event->delta() < 0)&&(m_scaleValue <= m_scaleDafault))//图像缩小到自适应大小之后就不继续缩小
    {
        ResetItemPos();//重置图片大小和位置，使之自适应控件窗口大小
    }
    else
    {
        qreal qrealOriginScale = m_scaleValue;
        if(event->delta() > 0)//鼠标滚轮向前滚动
        {
            m_scaleValue*=1.1;//每次放大10%
        }
        else
        {
            m_scaleValue*=0.9;//每次缩小10%
        }
        setScale(m_scaleValue);
        if(event->delta() > 0)
        {
            moveBy(-event->pos().x()*qrealOriginScale*0.1, -event->pos().y()*qrealOriginScale*0.1);//使图片缩放的效果看起来像是以鼠标
        }
        else
        {
            moveBy(event->pos().x()*qrealOriginScale*0.1, event->pos().y()*qrealOriginScale*0.1);//使图片缩放的效果看起来像是以鼠标所
        }
    }
}
```

功能展示:

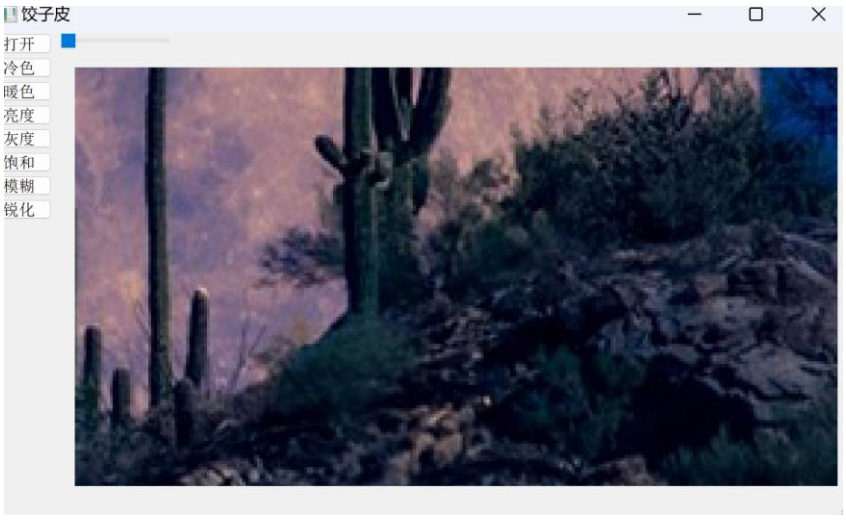
1.图片打开



2.图片拖拽



3.图片放缩



二、图像处理功能及其实现

饺子皮图像编辑器设计了一系列的图像处理功能，包括调整颜色调、亮度、饱和度、模糊

和锐化等，使用户可以根据需要对图片进行多样化的处理。

1.亮度调整：亮度的调整由 AlgoLight 函数实现，它通过增加或减少图片每个像素的 RGB 三个通道的值来调整亮度。当 RGB 三个通道的值增加时，图片会变亮；当 RGB 三个通道的值减少时，图片会变暗。传入的 delta 参数用来控制亮度的调整幅度。

```
QImage *ProcessorThread::AlgoLight(int delta, QImage *origin)
//同时增加三个通道的数值变量，反之就是变暗
{
    QImage *newImage = new QImage(origin->width(), origin->height(), QImage::Format_ARGB32);
    QColor oldColor;
    int r,g,b;
    for(int x=0; x<newImage->width(); x++)
    {
        for(int y=0; y<newImage->height(); y++)
        {
            oldColor = QColor(origin->pixel(x,y));
            r = oldColor.red() + delta;
            g = oldColor.green() + delta;
            b = oldColor.blue()+ delta;
            //we check if the new values are between 0 and 255
            r = qBound(0, r, 255);
            g = qBound(0, g, 255);
            b = qBound(0, b, 255);
            newImage->setPixel(x,y, qRgb(r,g,b));
        }
    }
    return newImage;
```

2.冷暖度调整：冷暖度的调整由 AlgoCool 和 AlgoWarm 函数实现。AlgoCool 函数通过增加蓝色通道的值，使图片呈现出冷色调，AlgoWarm 函数则通过增加红色和绿色通道的值，使图片呈现出暖色调。

```
QImage *ProcessorThread::AlgoCool(int delta, QImage * origin)//如果说暖色调的图片偏黄色，那么冷色调
{
    QImage *newImage = new QImage(origin->width(), origin->height(), QImage::Format_ARGB32);
    QColor oldColor;
    int r,g,b;
    for(int x=0; x<newImage->width(); x++){
        for(int y=0; y<newImage->height(); y++){
            oldColor = QColor(origin->pixel(x,y));

            r = oldColor.red();
            g = oldColor.green();
            b = oldColor.blue()+delta;
            //we check if the new value is between 0 and 255
            b = qBound(0, b, 255);
            newImage->setPixel(x,y, qRgb(r,g,b));
        }
    }
    return newImage;
}


QImage * ProcessorThread::AlgoWarm(int delta, QImage * origin)//当我们说一一幅暖色调的图片的时候通
{
    QImage *newImage = new QImage(origin->width(), origin->height(), QImage::Format_ARGB32);

    QColor oldColor;
    int r,g,b;

    for(int x=0; x<newImage->width(); x++){
        for(int y=0; y<newImage->height(); y++){
            oldColor = QColor(origin->pixel(x,y));
            r = oldColor.red() + delta;
            g = oldColor.green() + delta;
            b = oldColor.blue();
            //we check if the new values are between 0 and 255
            r = qBound(0, r, 255);
            g = qBound(0, g, 255);

            newImage->setPixel(x,y, qRgb(r,g,b));
        }
    }
```

3.灰度调整：灰度的调整由 AlgoGreyScale 函数实现，它将每个像素的 RGB 三个通道的值设置为三者的平均值，以此将彩色图片转换为灰度图片。

```
QImage * ProcessorThread::AlgoGreyScale(int delta,QImage * origin)
//将彩色图转换成灰度图，我们首先要明白的一点就是，其实标准的灰度图就是每个像素点的三个通道的值一样或者近似，我们的策略就是将每个像素
{
    QImage * newImage = new QImage(origin->width(), origin->height(), QImage::Format_ARGB32);
    for(int y = 0; y<newImage->height(); y++){
        QRgb * line = (QRgb *)origin->scanLine(y);//按行读取，效率比一个一个字节读取要高
        for(int x = 0; x<newImage->width(); x++)
        {
            int average = (qRed(line[x]) + qGreen(line[x]) + qRed(line[x]))/3;
            average += delta;
            average = qBound(0, average, 255);//将像素值范围定在0~255
            newImage->setPixel(x,y, qRgb(average, average, average));
        }
    }
    return newImage;
}
```

4. 饱和度增强算法：`AlgoSaturation(int delta, QImage * origin)`。这个算法通过将图像从 RGB 格式转换到 HSL 格式来增加饱和度。在 HSL 格式中，色相(H)、饱和度(S)和亮度(L)是图像的三个基本属性。增加饱和度只需修改 HSL 格式中的 S 值。

```
QImage *ProcessorThread::AlgoSaturation(int delta, QImage * origin)
//颜色由三个通道组成：红，绿，蓝，尽管如此，RGB不是唯一一个表示色彩的方式，在这里，使用HSL格式表示色彩 - hue（色相），saturation（饱和度），lightne
//饱和的图像拥有更加生动的颜色，通常会比较好看，但不要滥用饱和度，因为很容易出现失真。
{
    QImage * newImage = new QImage(origin->width(), origin->height(), QImage::Format_ARGB32);

    QColor oldColor;
    QColor newColor;
    int h,s,l;

    for(int x=0; x<newImage->width(); x++){
        for(int y=0; y<newImage->height(); y++){
            oldColor = QColor(origin->pixel(x,y));

            newColor = oldColor.toHsl();
            h = newColor.hue();
            s = newColor.saturation()+delta;
            l = newColor.lightness();

            //we check if the new value is between 0 and 255
            s = qBound(0, s, 255);

            newColor.setHsl(h, s, l);

            newImage->setPixel(x, y, qRgb(newColor.red(), newColor.green(), newColor.blue()));
        }
    }

    return newImage;
}
```

5. 模糊算法：`AlgoBlur(int delta,QImage * origin)`。模糊算法使用 5x5 的卷积核来处理图像，通过计算卷积核内的加权平均值，得到每个像素的新值，并将结果存储在新图像中。

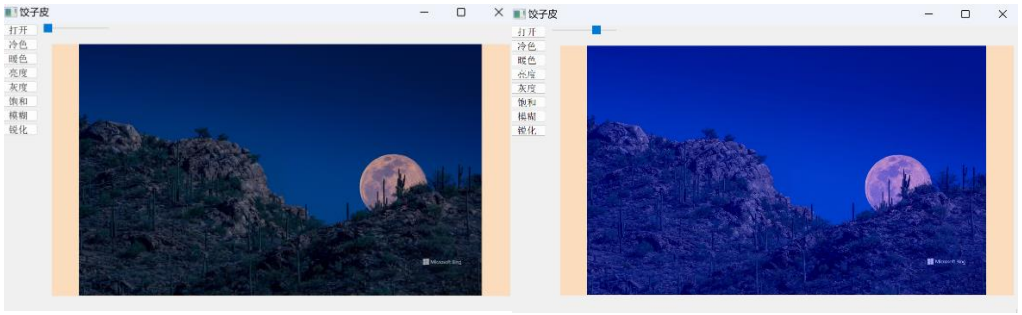
```
QImage  *ProcessorThread::AlgoBlur(int delta,QImage * origin)
{
    QImage * newImage = new QImage(*origin);

    int kernel [5][5]= {{0,0,1,0,0},
                        {0,1,3,1,0},
                        {1,3,7,3,1},
                        {0,1,3,1,0},
                        {0,0,1,0,0}};
    int kernelSize = 5;
    //int sumKernel = 27+delta;
    int sumKernel = 27;
    int r,g,b;
    QColor color;

    for(int x=kernelSize/2; x<newImage->width()-(kernelSize/2); x++){
        for(int y=kernelSize/2; y<newImage->height()-(kernelSize/2); y++){

            r = 0;
            g = 0;
            b = 0;

            for(int i = -kernelSize/2; i<= kernelSize/2; i++){
                for(int j = -kernelSize/2; j<= kernelSize/2; j++){
                    color = QColor(origin->pixel(x+i, y+j));
                    r += color.red()*kernel[kernelSize/2+i][kernelSize/2+j];
                    g += color.green()*kernel[kernelSize/2+i][kernelSize/2+j];
                    b += color.blue()*kernel[kernelSize/2+i][kernelSize/2+j];
                }
            }

            r = qBound(0, r/sumKernel, 255);
            g = qBound(0, g/sumKernel, 255);
            b = qBound(0, b/sumKernel, 255);

            newImage->setPixel(x,y, qRgb(r,g,b));

        }
    }
    return newImage;
}
```

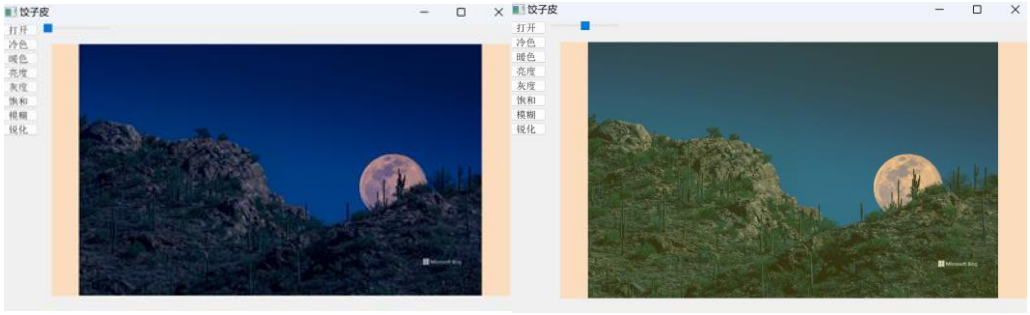
6. 锐化算法：`AlgoSharpen(int delta,QImage * origin)`。锐化算法使用 3x3 的卷积核来处理图像，这个算法的目的是增强图像的细节和边缘，使图像看起来更加清晰。

```
QImage  *ProcessorThread::AlgoSharpen(int delta,QImage * origin)
{
    QImage * newImage = new QImage(* origin);
    int kernel [3][3]= {{0,-1,0},
                        {-1,5,-1},
                        {0,-1,0}};
    int kernelSize = 3;
    int sumKernel = 1;
    int r,g,b;
    QColor color;
    for(int x=kernelSize/2; x<newImage->width()-(kernelSize/2); x++){
        for(int y=kernelSize/2; y<newImage->height()-(kernelSize/2); y++)
        {
            r = 0;
            g = 0;
            b = 0;
            for(int i = -kernelSize/2; i<= kernelSize/2; i++)
            {
                for(int j = -kernelSize/2; j<= kernelSize/2; j++)
                {
                    color = QColor(origin->pixel(x+i, y+j));
                    r += color.red()*kernel[kernelSize/2+i][kernelSize/2+j];
                    g += color.green()*kernel[kernelSize/2+i][kernelSize/2+j];
                    b += color.blue()*kernel[kernelSize/2+i][kernelSize/2+j];
                }
            }
            r = qBound(0, r/sumKernel, 255);
            g = qBound(0, g/sumKernel, 255);
            b = qBound(0, b/sumKernel, 255);
            newImage->setPixel(x,y, qRgb(r,g,b));
        }
    }
    return newImage;
}
```

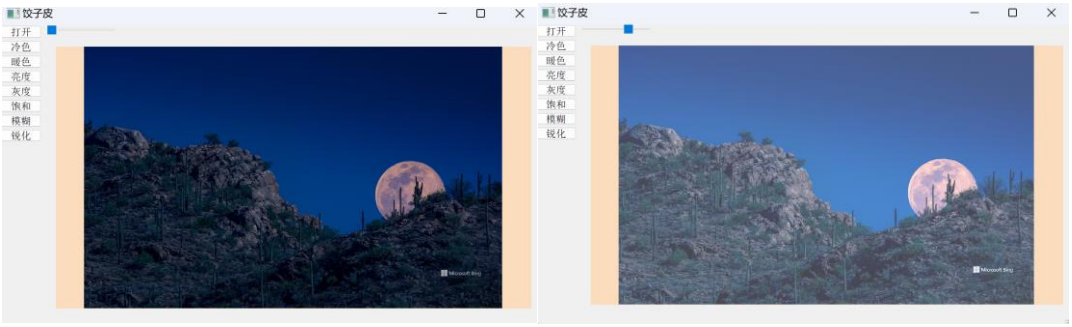
功能展示

1.冷暖色调整

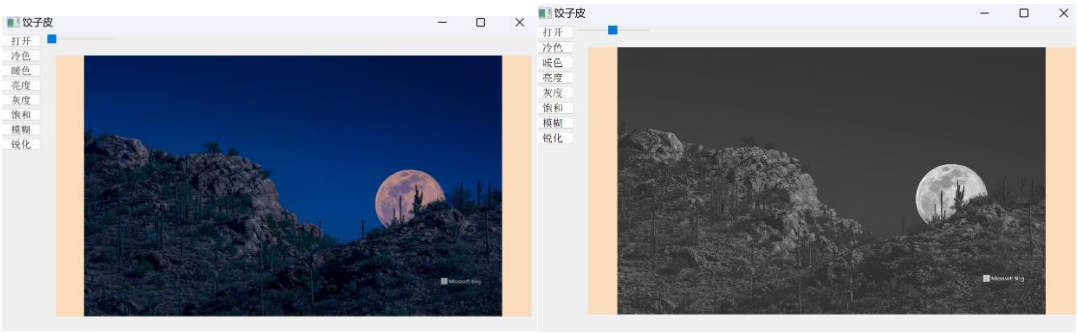




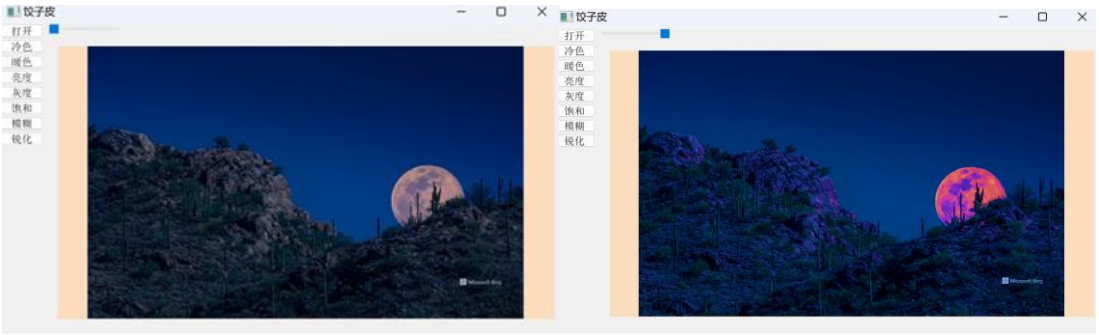
2.亮度调整



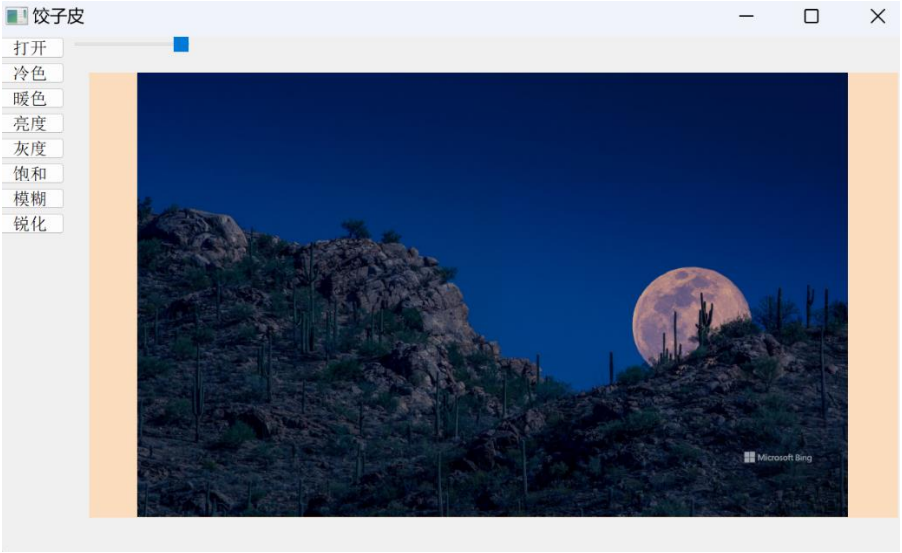
3.灰度调整



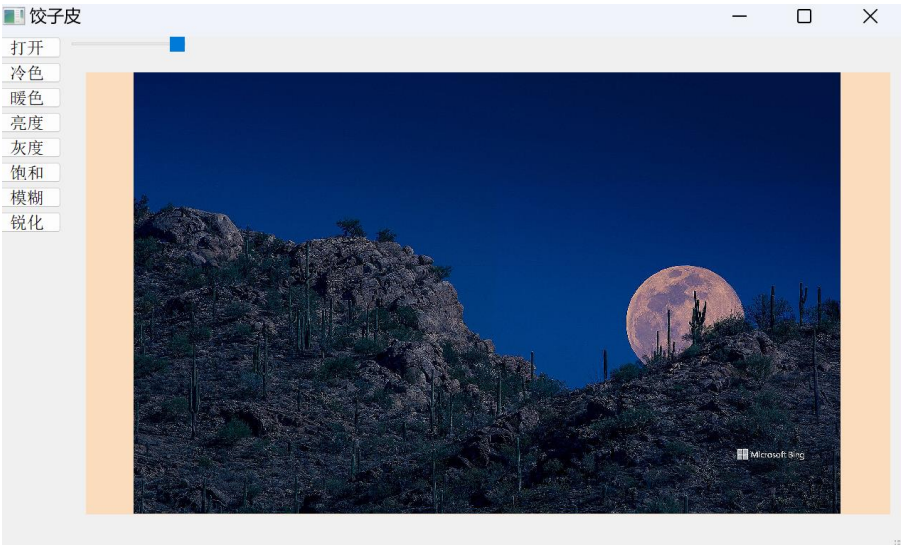
4.饱和度调整



5.模糊度调整



6.锐化



三、收获与总结

通过本次项目，我对计算机视觉和图像处理有了更深的理解。在实现图像处理算法的过程中，我学习了如何转换图像格式，如何使用卷积核对图像进行处理，以及如何调整图像的基本属性。这些知识都在实际的编程中得到了应用和巩固。另外，我也对 Qt 框架有了更深的理解，学会了如何使用 QGraphicsItem 和 QGraphicsView 等类进行图像的显示和交互。

同时，这个项目也锻炼了我的编程能力和解决问题的能力。在面对各种问题和困难时，我学会了如何搜索资料、如何分析问题，以及如何寻找有效的解决方案。我相信这些能力将在今后的学习和工作中发挥重要的作用。