6.1.4　运动模糊

1．效果介绍

运动模糊（Motion Blur）滤镜是一种抓取物体运动状态的效果滤镜，主要应用物体运动时曝光的摄影手法，模拟出在摄像中拍摄运动物体的间接曝光功能，从而使图像产生出一种动态效果。它通常用来制造物体掠过或移动的效果。

2．实现原理

运动模糊滤镜沿特定方向，并以特定的强度进行模糊处理，它的实现方法较为复杂。

首先，要解决图像运动模糊方向的角度问题。在数学中，*Y* 轴向上为正；然而在图像处理中，*Y* 轴向下为正。所以在获取用户指定的运动方向角度后，应先将其沿正方向旋转 180°。

接着，要解决图像在指定方向上的位移问题。运动模糊不是简单地将图像在指定方向上移来移去，而在距离限定的范围内，按某种方式复制并叠加像素。其实，可以形象地看成，将一幅图像的多张副本叠放在指定方向上，然后取其平均值。

最后要解决的问题就是图像的透明度。在本滤镜效果处理中，需要处理像素颜色的 Alpha 分量，这样最终产生的模糊效果才更理想。

3．编程思路

① 通过对话框获取用户指定的角度和距离。

② 根据角度和距离计算出图像运动路线。

③ 循环处理图像数据区内所有像素。

④ 将当前像素点在②中的运动路线上进行累加，即叠加处理。注意，此步是关键，请仔细阅读例 6-6 中的代码。

⑤ 将④得出的新色彩作为结果颜色输出。

⑥ 循环步骤③、④、⑤，直到处理完原图的全部像素点。

4．实例程序

本例程序中，将主要用到例 6-6 这一段程序。

例 6-6 的功能就是实现运动模糊效果。

例 6-6

```
/// <summary>
/// 运动模糊
/// </summary>
/// <param name="b">位图流</param>
/// <param name="angle">角度方向[0, 360]</param>
/// <param name="distance">距离[1, 200]</param>
/// <returns></returns>
public Bitmap MotionBlur(Bitmap b, int angle, int distance)
{
```

```
        angle %= 360;

        if (distance < 1) distance = 1;
        if (distance > 200) distance = 200;

        // 弧度、距离
        double radian = ((double)angle + 180.0) * Math.PI / 180.0;
        int dx = Convert.ToInt32((double)distance * Math.Cos(radian));
        int dy = Convert.ToInt32((double)distance * Math.Sin(radian));

        // 首点、尾点
        Point start = new Point(-dx / 2, dy / 2);
        Point end = new Point(dx / 2, -dy / 2);

        //获取由首尾点组成的线段中的所有点的集合（本书不显示具体代码，详细代码查阅软件代码）
        Point[] points = Function.GetLinePoints(start, end);

        int width = b.Width;
        int height = b.Height;

        // 目标图像
        Bitmap dstImage = new Bitmap(width, height);

        BitmapData srcData = b.LockBits(new Rectangle(0, 0, width, height),
          ImageLockMode.ReadOnly, PixelFormat.Format32bppArgb);
        BitmapData dstData = dstImage.LockBits(new Rectangle(0, 0, width, height),
          ImageLockMode.ReadWrite, PixelFormat.Format32bppArgb);

        int stride = srcData.Stride;
        System.IntPtr srcScan0 = srcData.Scan0;
        System.IntPtr dstScan0 = dstData.Scan0;
        int offset = stride - width * BPP;

        unsafe
        {
          byte* src = (byte*)srcScan0;
          byte* dst = (byte*)dstScan0;

          int X = 0, Y = 0;

          for (int y = 0; y < height; y++)
          {
            for (int x = 0; x < width; x++)
            {
              long bSum = 0;
              long gSum = 0;
```

```
            long rSum = 0;
            long aSum = 0;
            int div = 1;
            int validPoint = 0;

            foreach (Point point in points)
            {
             X = x + point.X;
             Y = y + point.Y;

             if ((X >= 0 && X < width) && (Y >= 0 && Y < height))
             {
              src = (byte*)srcScan0 + Y * stride + X * BPP;

              if (src[3] > 0)
              {
               bSum += src[0] * src[3]; // B * A
               gSum += src[1] * src[3]; // G * A
               rSum += src[2] * src[3]; // R * A
               aSum += src[3];       // A
               div += src[3];
              }

              validPoint++;
             }
            } // point

            dst[3] = (byte)(aSum /= validPoint);
            dst[2] = (byte)(rSum /= div);
            dst[1] = (byte)(gSum /= div);
            dst[0] = (byte)(bSum /= div);

            dst += BPP;
           } // x

           dst += offset;
          } // y
         }

         b.UnlockBits(srcData);
         dstImage.UnlockBits(dstData);

         b.Dispose();

         return dstImage;
        } // end of MotionBlur
```

5．重点讲解

在例 6-6 中，局部变量 bSum、gSum、rSum 和 aSum 是在处理像素点移动时叠加亮度之用。局部变量 div 用来记录像素的 Alpha 状况，而 validPoint 用来记录在像素点移动时，线段上的有效点数。本例采用求和取平均的思路来设置最终的亮度值。

6．效果演示

依次选择【特效滤镜】→【模糊】→【运动模糊】菜单命令，弹出“运动模糊”参数设置对话框，如图 6-5 所示。

图 6-5

在对话框中，可以看到两个选项：“角度”和“距离”。角度用于控制运动模糊的方向，产生往哪一个方向的运动效果；距离用于设定像素移动的距离。用鼠标在角度盘上旋转，进行角度方向选择，其调节范围是[0, 360]。用鼠标拖动距离滑条，进行距离调节，其调节范围是[1, 200]，值越大，模糊效果越强。当调节好所需的值后，单击【确定】按钮，将看到如图 6-6 所示的前后对比效果图。这里设置的参数为角度：180°，距离：30。

图 6-6

从图 6-6 可以看出，经运动模糊处理之后，图中的轿车在水平方向上产生了模糊的重影，产生很大的视觉冲击力，给人一种很动感的效果。

7．重要小结

图像模糊处理有着很广的用途，常见的方法就是对图像进行滤波处理。

比如，气象卫星在高空拍摄大气运动状况时，由于种种环境因素，使得最终得到的卫星云图会出现少量的杂点，这时完全可以通过图像模糊的原理，来弱化或去除这些杂点。

对图像进行滤波处理的内容，笔者将在 8.3 中作进一步讲解。