



# 天津理工大学

计算机科学与工程学院

## 实验报告

2017 至 2018 学年 第 二 学期

### 实验二 图像几何变换

课程名称	数字图像处理				
学号	20152180	学生姓名	王帆	年级	2015
专业	计算机科学与技术	教学班号	2	实验地点	主 7-212
实验时间	2018 年 4 月 2 日 第 7 节 至 第 8 节				
主讲教师	杨淑莹				

### 实验成绩

软件运行	效果	算法分析	流程设计	报告成绩	总成绩

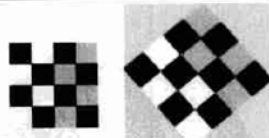


实验（二）	实验名称	图像几何变换	
软件环境	Windows Visual Studio 2017		
硬件环境	PC		
实验目的			
掌握图像的几何变换方法，编程实现图像几何变换。			
实验内容（应包括实验题目、实验要求、实验任务等）			
一、设计新的几何变换方法，分析其几何变换原理。			
二、编程实现图像的几何变换。			
要求：编程实现图像的几何变换功能。			
任务：			
(1) 在左视图中打开一幅 bmp 位图，包括 256 色或真彩色位图			
(2) 制作五个【图像的 XXX 变换】菜单，将消息映射到右视图中，在右视图 中进行图像的几何变换功能。			
实验过程与实验结果			
一、设计新的几何变换方法，分析其几何变换原理。			
图像配准			
什么是图像配准？			
所谓图像配准就是将同一场景的两幅或多幅图像进行对准。如航空照片的配准，以及在很多人脸自动分析系统中的人脸归一化，即要使各张照片中的人脸具有近似的大小，尽量处于相同的位置。一般来说，我们以基准图像为参照，并通过一些基准点（fiducial points）找到适当的空间变换关系 $s$ 和 $r$ ，对输入图像进行相应的几何变换，从而实现它与基准图像在这些基准点位置上的对齐。			
表 4.2 变换类型及其说明			
变换类型	适用情况	最小控制点对儿数	示例
linear conformal	当输入图像中的形状没有改变,但图像经过了平移、旋转以及比例缩放等变换后发生失真时使用本变换。变换后直线仍然是直线,平行线仍为平行的	2 对	
affine	当输入图像中的形状展示出错切效果使用本变换。变换后直线仍然是直线,平行线仍为平行的,但矩形变成了平行四边形	3 对	
Projective	当场景显得倾斜时使用本变换。变换后直线仍然为直线,但平行线不再平行	4 对	

图 1：常见的几种配准变换类型

## 二、编程实现图像的几何变换

### 1. 平移变换

#### 实现步骤:

1. 获取原图像的 Bitmap 对象 objBitmap, 以及其大小参量;
2. 获取平移变换参数 X, Y;
3. 构造一个新 Bitmap 对象 bitmap, 宽和高分别设置为原图宽+X, 原图高+Y
4. 迭代实现对 objBitmap 每一个像素点 (Pixel) 到 bitmap 的赋值操作:  
对每个像素点的横纵坐标分别加上平移变换参数 X, Y, 再以原像素点赋值;
5. 使用 bitmap 构造全局变量 curBitmap, 销毁 bitmap 对象, 使用 curBitmap 初始化右侧显示框。

#### 代码:

//选项: 基本处理-平移

```
private void ToolStripMenuItem_translation_Click(object sender, EventArgs e)
{
    //加载窗体transForm
    transForm transfrm = new transForm();
    //定义窗体所有者
    transfrm.Owner = this;
    transfrm.ShowDialog();
    if (transfrm.flag)
    {
        try
        {
            int temp_x = Convert.ToInt32(transfrm.textBoxX.Text);
            int temp_y = Convert.ToInt32(transfrm.textBoxY.Text);

            //图像处理操作
            int width = objBitmap.Width;
            int height = objBitmap.Height;
            Bitmap bitmap = new Bitmap(width + temp_x, height + temp_y);

            for (int x = 0; x < width; x++)
            {
                for (int y = 0; y < height; y++)
                {
                    bitmap.SetPixel(x + temp_x, y + temp_y, objBitmap.GetPixel(x, y));
                }
            }
            this.pictureBox_new.Image = bitmap;
            curBitmap = new Bitmap(bitmap);
        }
        catch (Exception ex)
        {
        }
    }
}
```

//错误提示

```

        MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK, MessageBoxIcon.Stop);
    }
}
}

```

效果图:

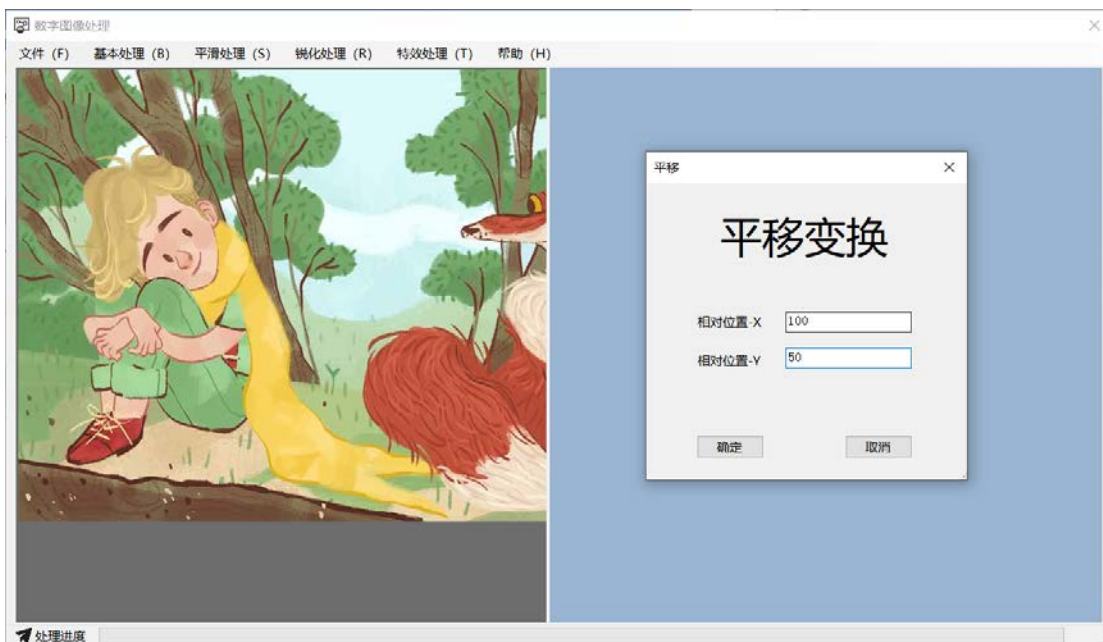


图 2-1 平移变换前

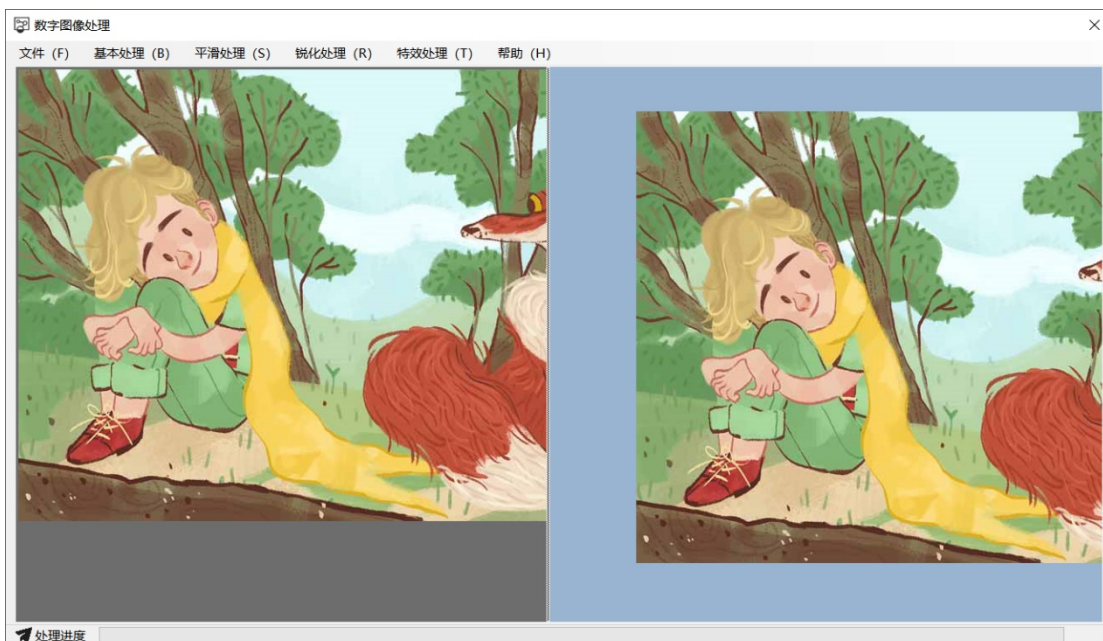


图 2-2 平移变换后

## 2. 镜像变换

### 实现步骤:

1. 获取原图像的 Bitmap 对象 objBitmap, 以及其大小参量;
2. 获取变换类型 (水平/垂直) ;
3. 构造一个新 Bitmap 对象 bitmap, 同原图等宽高;
4. 迭代实现对 objBitmap 每一个像素点 (Pixel) 到 bitmap 的赋值操作:
  - 若水平镜像, 则将每个像素点的水平坐标设置为用图像宽度减去原坐标值;
  - 若垂直镜像, 则将每个像素点的垂直坐标设置为用图像高度减去原坐标值;
5. 使用 bitmap 构造全局变量 curBitmap, 销毁 bitmap 对象, 使用 curBitmap 初始化右侧显示框。

### 代码:

//选项: 基本处理-镜像-水平镜像

```
private void ToolStripMenuItem_mirror_X_Click(object sender, EventArgs e)
{
    try
    {
        int width = objBitmap.Width;
        int height = objBitmap.Height;
        Bitmap bitmap = new Bitmap(width, height);

        for (int x = 0; x < width; x++)
        {
            for (int y = 0; y < height; y++)
            {
                bitmap.SetPixel(x, y, objBitmap.GetPixel(width - x - 1, y));
            }
        }
        curBitmap = new Bitmap(bitmap);
        bitmap.Dispose();
        this.pictureBox_new.Image = curBitmap;
    }
    catch (Exception ex)
    {
        //错误提示
        MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK, MessageBoxIcon.Stop);
    }
}

//选项: 基本处理-镜像-垂直镜像
private void ToolStripMenuItem_mirror_Y_Click(object sender, EventArgs e)
{
    try
```

```

{
    int width = objBitmap.Width;
    int height = objBitmap.Height;
    Bitmap bitmap = new Bitmap(width, height);
    for (int x = 0; x < width; x++)
    {
        for (int y = 0; y < height; y++)
        {
            bitmap.SetPixel(x, y, objBitmap.GetPixel(x, height-1-y));
        }
    }
    curBitmap = new Bitmap(bitmap);
    bitmap.Dispose();
    this.pictureBox_new.Image = curBitmap;
}
catch (Exception ex)
{
    //错误提示
    MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK, MessageBoxIcon.Stop);
}
}

```

效果图：

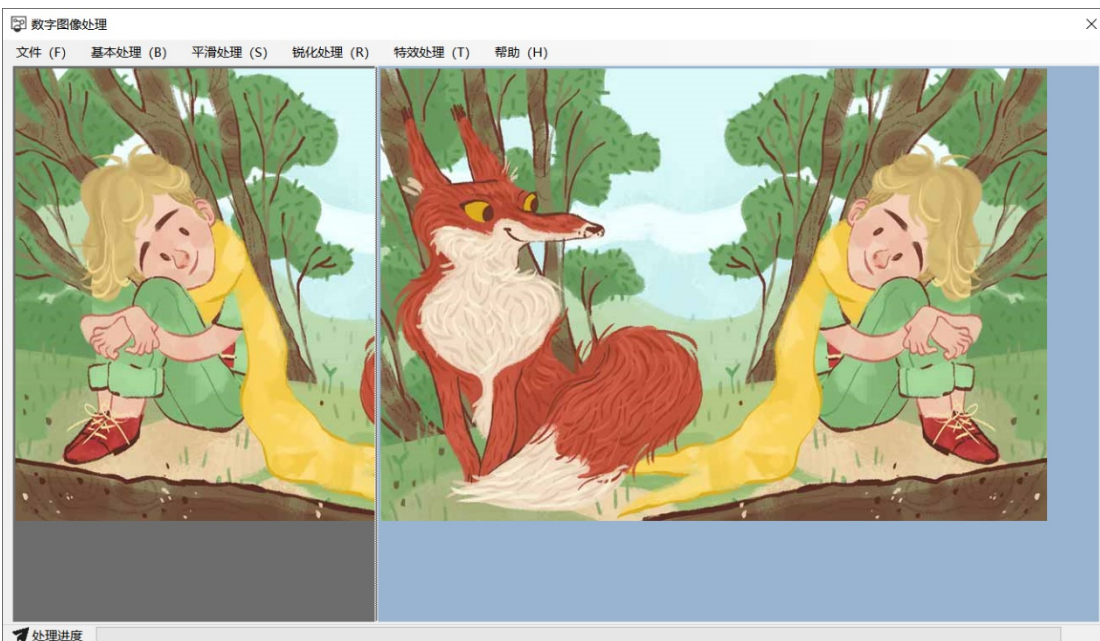


图 3-1 水平镜像后



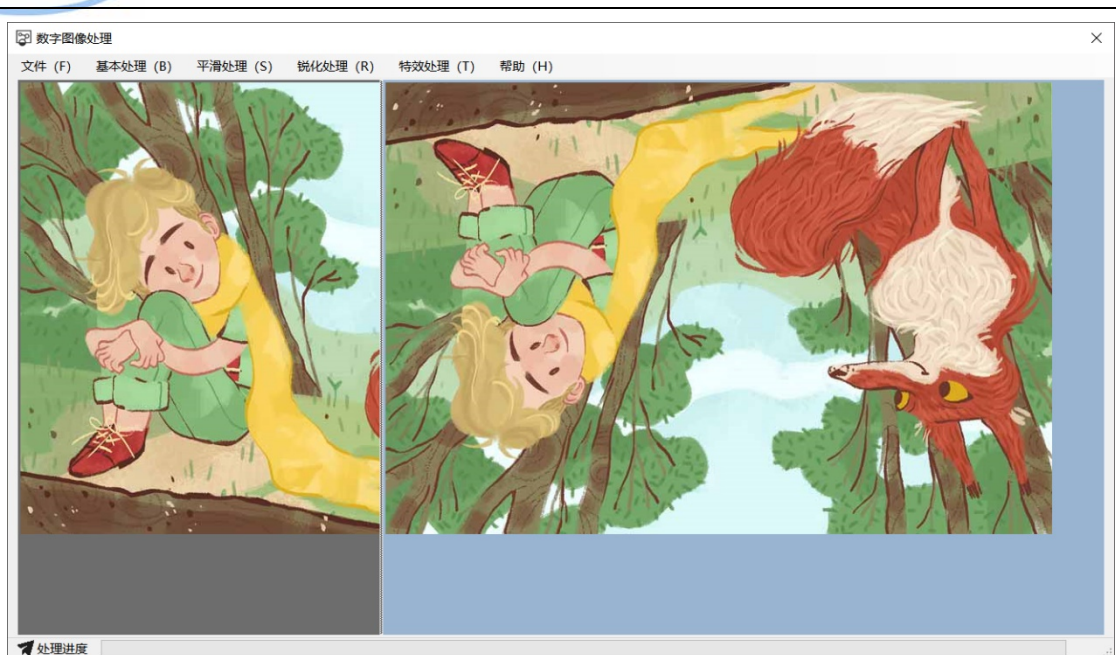


图 3-2 垂直镜像后

### 3.缩放变换

#### 实现步骤:

- 1.获取原图像的 Bitmap 对象 objBitmap, 以及其大小参量;
- 2.获取缩放比例系数 (X, Y) ;
- 3.构造一个新 Bitmap 对象 bitmap, 宽高为宽\*X+1, 高\*Y+1;
- 4.迭代实现对 objBitmap 每一个像素点 (Pixel) 到 bitmap 的赋值操作:  
将每个像素点的坐标设置为原坐标/比例系数, 如横坐标/X, 纵坐标/Y;
- 5.使用 bitmap 构造全局变量 curBitmap, 销毁 bitmap 对象, 使用 curBitmap 初始化右侧显示框。

#### 代码:

//选项: 基本处理-缩放

```
private void ToolStripMenuItem_zoom_Click(object sender, EventArgs e)
{
    try
    {
        zoomForm zoomfrm = new zoomForm();

        zoomfrm.ShowDialog();
        if (zoomfrm.flag)
        {
            double bilvX = Convert.ToDouble(zoomfrm.textBoxX.Text);
            double bilvY = Convert.ToDouble(zoomfrm.textBoxY.Text);
            int width = objBitmap.Width;
            int height = objBitmap.Height;
```

```

        Bitmap bitmap = new Bitmap((int)(width * bilvX) + 1, (int)(height *
        bilvY) + 1);

        for (int x = 0; x < width * bilvX; x++)
        {
            for (int y = 0; y < height * bilvY; y++)
            {
                bitmap.SetPixel(x, y, objBitmap.GetPixel((int)(x / bilvX),
                (int)(y / bilvY)));
            }
        }
        curBitmap = new Bitmap(bitmap);
        bitmap.Dispose();
        this.pictureBox_new.Image = curBitmap;
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK, MessageBoxIcon.Stop);
}
}

```

效果图:

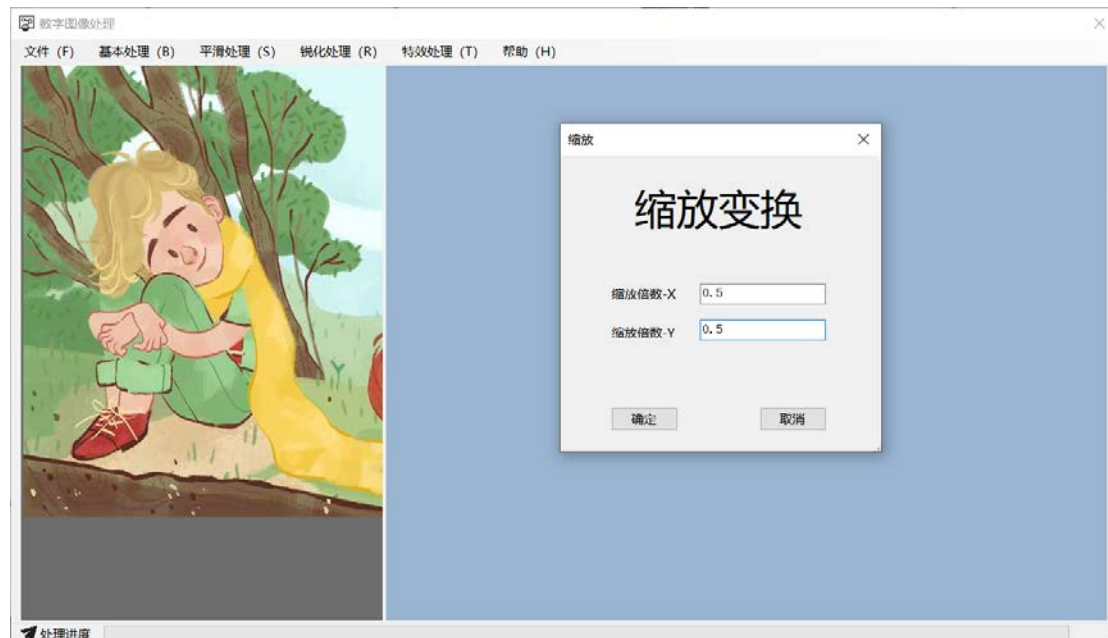


图4-1 缩放变换前



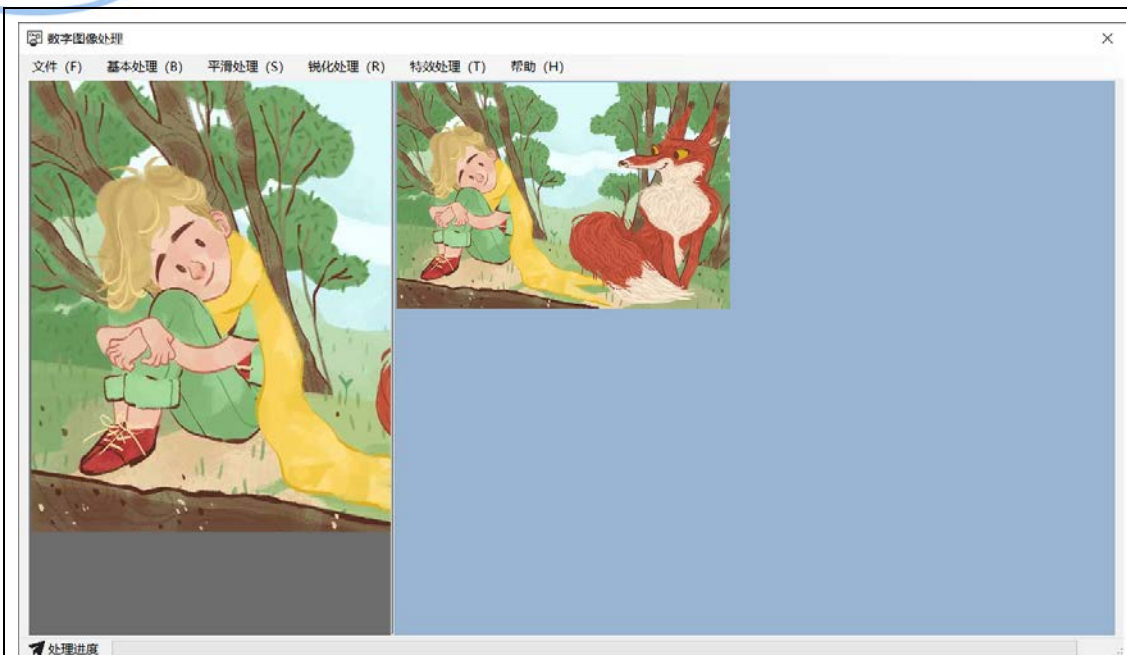


图4-2 缩放变换后（缩小为原来的一半）

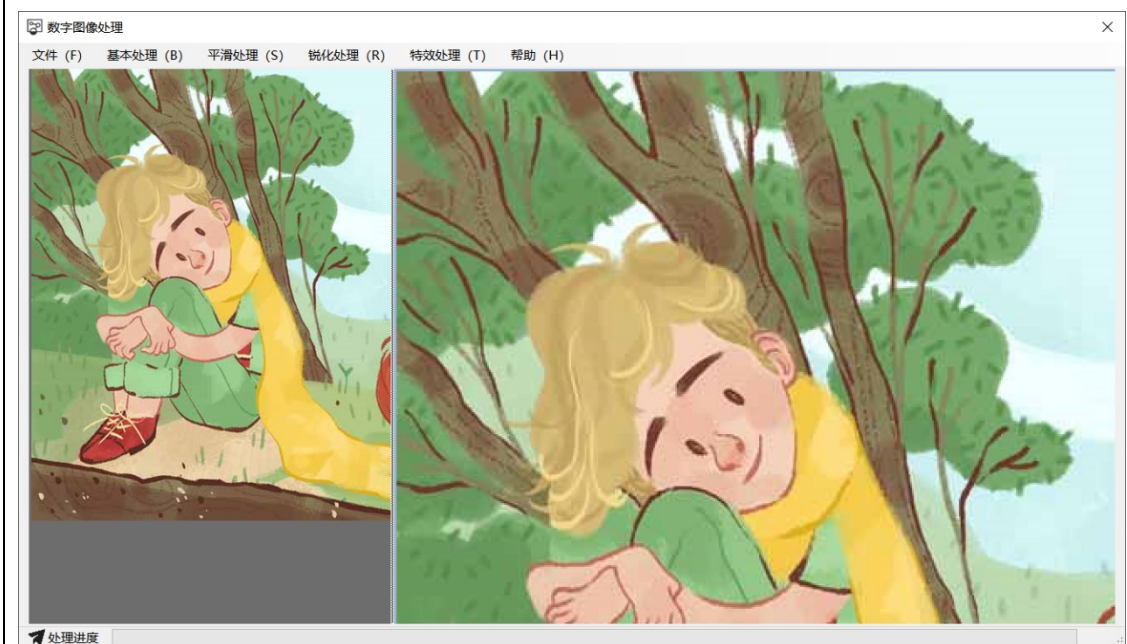


图4-3 缩放变换后（放大为原来的2倍）

#### 4.转置变换

##### 实现步骤:

- 1.获取原图像的 Bitmap 对象 objBitmap, 以及其大小参量;
- 2.构造一个新 Bitmap 对象 bitmap, 宽, 高分别为原图像高和宽;
- 3.迭代实现对 objBitmap 每一个像素点 (Pixel) 到 bitmap 的赋值操作:  
将每个像素点的坐标分别设置为原纵坐标, 原横坐标, 如: (y, x);
- 4.使用 bitmap 构造全局变量 curBitmap, 销毁 bitmap 对象, 使用 curBitmap 初始化右侧显示框。

##### 代码:

//选项：基本处理-几何变换-转置

```
private void ToolStripMenuItem_transposition_Click(object sender, EventArgs e)
{
    try
    {
        int width = objBitmap.Width;
        int height = objBitmap.Height;
        Bitmap bitmap = new Bitmap(height, width);

        for (int x = 0; x < width; x++)
        {
            for (int y = 0; y < height; y++)
            {
                bitmap.SetPixel(y, x, objBitmap.GetPixel(x, y));
            }
        }
        curBitmap = new Bitmap(bitmap);
        bitmap.Dispose();
        this.pictureBox_new.Image = curBitmap;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK, MessageBoxIcon.Stop);
    }
}
```

效果图：

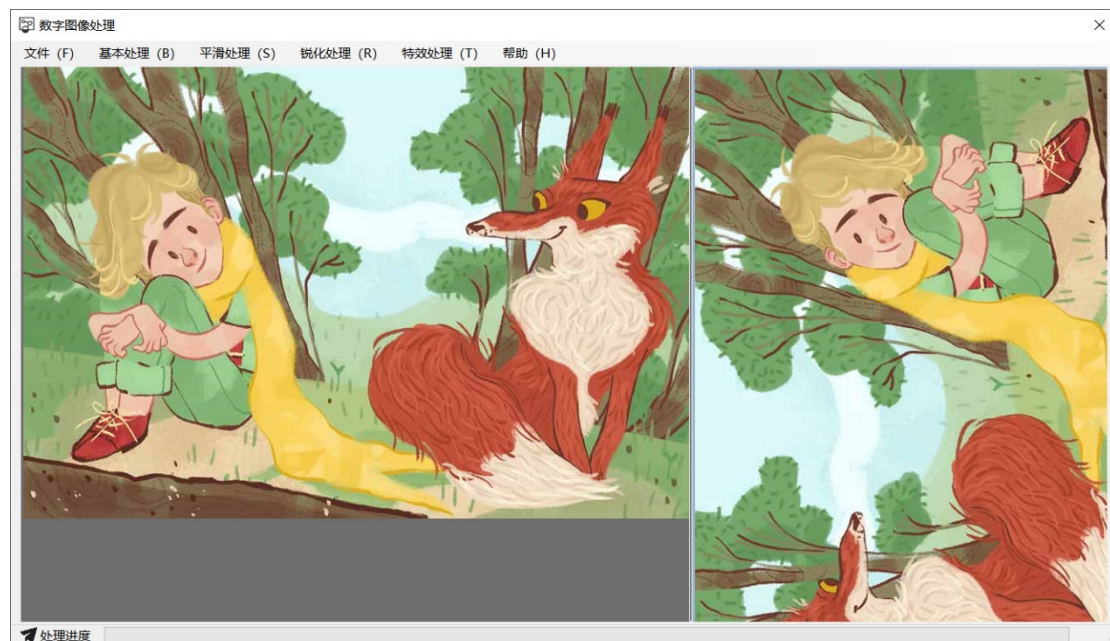


图5 转置变换后

## 5.旋转变换

### 实现步骤:

- 1.获取原图像的 Bitmap 对象 objBitmap, 以及其大小参量;
- 2.获取旋转角度 rotation;
- 3.构造一个新 Bitmap 对象 bitmap, 通过 GetRotateRectangle 方法求得 bitmap 的长和宽;
4. 根据旋转后的宽高定义 Bitmap(rotateImage), 定义 Graphics, 将 Graphics 按 rotateImage 的矩形区域中心进行旋转变换:
  - (1)将 Graphics 的原点移至矩形的中点, 假设坐标为(x,y);
  - (2)将 Graphics 绕当前原点旋转 N 度;
  - (3)将 Graphics 沿(-x, -y)移回。
- 5.将 srcImage 绘制到 rotateImage 中心 (即两个中心点重合);
- 6.重置 Graphics;
- 7.使用 bitmap 构造全局变量 curBitmap, 销毁 bitmap 对象, 使用 curBitmap 初始化右侧显示框。

### 代码:

```
/// <summary>
/// 计算矩形绕中心任意角度旋转后所占区域矩形宽高
/// </summary>
/// <param name="width">原矩形的宽</param>
/// <param name="height">原矩形高</param>
/// <param name="angle">顺时针旋转角度</param>
/// <returns></returns>
public static Rectangle GetRotateRectangle(int width, int height, float angle)
{
    double radian = angle * Math.PI / 180; ;
    double cos = Math.Cos(radian);
    double sin = Math.Sin(radian);
    //只需要考虑到第四象限和第三象限的情况取大值(中间用绝对值就可以包括第一和第二象限)
    int newWidth = (int)(Math.Max(Math.Abs(width * cos - height * sin),
Math.Abs(width * cos + height * sin)));
    int newHeight = (int)(Math.Max(Math.Abs(width * sin - height * cos),
Math.Abs(width * sin + height * cos)));
    return new Rectangle(0, 0, newWidth, newHeight);
}

/// <summary>
/// 获取原图像绕中心任意角度旋转后的图像
/// </summary>
/// <param name="rawImg"></param>
/// <param name="angle"></param>
/// <returns></returns>
public static Bitmap GetRotateImage(Bitmap srcImage, int angle)
```

```
{
    angle = angle % 360;
    //原图的宽和高
    int srcWidth = srcImage.Width;
    int srcHeight = srcImage.Height;
    //图像旋转之后所占区域宽和高
    Rectangle rotateRec = GetRotateRectangle(srcWidth, srcHeight, angle);
    int rotateWidth = rotateRec.Width;
    int rotateHeight = rotateRec.Height;
    //目标位图
    Bitmap destImage = null;
    Graphics graphics = null;
    try
    {
        //定义画布，宽高为图像旋转后的宽高
        destImage = new Bitmap(rotateWidth, rotateHeight);
        graphics = Graphics.FromImage(destImage);
        //要让graphics围绕某矩形中心点旋转N度，分三步
        //第一步，将graphics坐标原点移到矩形中心点,假设其中点坐标 (x,y)
        //第二步，graphics旋转相应的角度(沿当前原点)
        //第三步，移回 (-x,-y)
        //获取画布中心点
        Point centerPoint = new Point(rotateWidth / 2, rotateHeight / 2);
        graphics.TranslateTransform(centerPoint.X, centerPoint.Y);
        graphics.RotateTransform(angle);
        graphics.TranslateTransform(-centerPoint.X, -centerPoint.Y);
        Point Offset = new Point((rotateWidth - srcWidth) / 2, (rotateHeight -
srcHeight) / 2);
        graphics.DrawImage(srcImage, new Rectangle(Offset.X, Offset.Y, srcWidth,
srcHeight));
        graphics.ResetTransform();
        graphics.Save();
    }
    catch (Exception ex)
    {
        throw ex;
    }
    finally
    {
        if (graphics != null)
            graphics.Dispose();
    }
    return destImage;
}
```



//选项：基本处理-几何变换-旋转

```
private void ToolStripMenuItem_rotation_Click(object sender, EventArgs e)
{
    try
    {
        rotationForm rotationfrm = new rotationForm();
        rotationfrm.ShowDialog();
        if (rotationfrm.flag)
        {
            int angle = Convert.ToInt32(rotationfrm.textBox_degree.Text);
            Bitmap bitmap = COMUtil.GetRotateImage(objBitmap, angle);
            curBitmap = new Bitmap(bitmap);
            bitmap.Dispose();
            this.pictureBox_new.Image = curBitmap;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK,
        MessageBoxIcon.Stop);
    }
}
```

效果图：

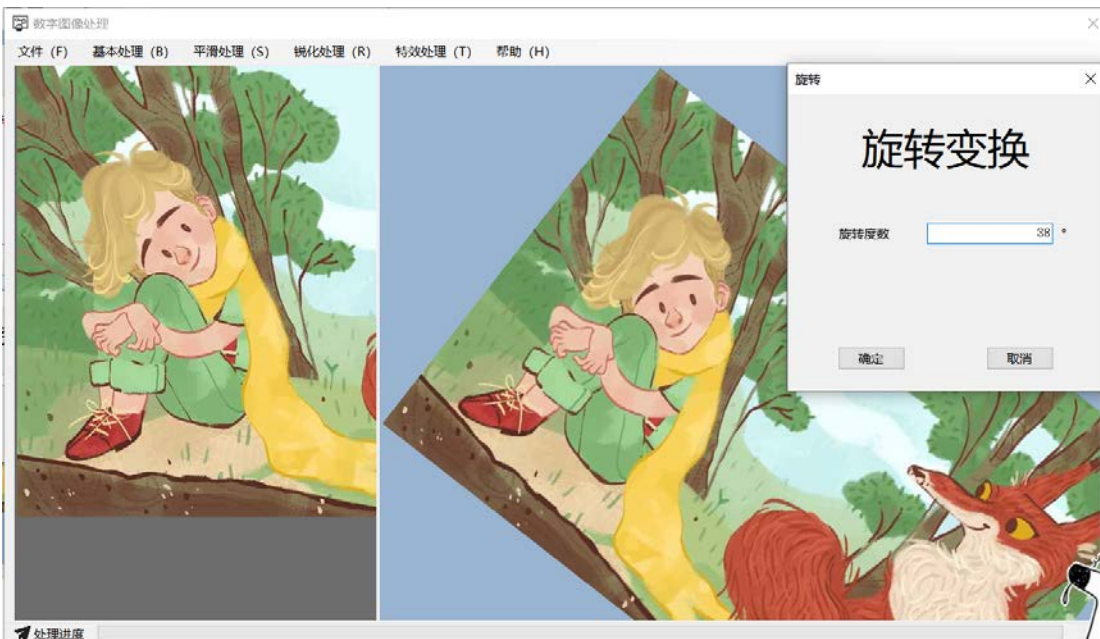


图 6 旋转变换后（绕图像中心旋转 38°）

## 附录

参考资料:

1. 《数字图像处理 Visual Studio C++技术实现》杨淑莹, 张桦, 陈胜勇/著;
2. 《C#程序设计》王贤明等 编著;
3. C#中基于 GDI+(Graphics)图像处理系列之任意角度旋转图像 - CSDN 博客  
<https://blog.csdn.net/lhtzbj12/article/details/54099572>;
4. C#如何释放已经加载的图片\_百度知道  
<https://zhidao.baidu.com/question/405810916.html>;
5. C#中 OpenFileDialog 获取文件名和文件路径的常用方法 - CSDN 博客  
<https://blog.csdn.net/zjm750617105/article/details/47867311>;
6. C#, 单元测试入门 - 清风笑 - 博客园  
<https://www.cnblogs.com/KevinMO/articles/5657747.html>
7. C# 静态方法和数据 - Mr&H - 博客园  
<https://www.cnblogs.com/hjxzjp/p/7861813.html>
8. C#图像处理入门(-bitmap 类和图像像素值获取方法) - 浮云的等待 - 博客园  
<https://www.cnblogs.com/GmrBrian/p/6830106.html>
9. Visual C# 中实现窗体间的数据传递 - CSDN 博客  
<https://blog.csdn.net/cngkqy/article/details/2051033>