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

实验六 图像的轮廓提取

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	**7-212**
实验时间	**2018 年 4 月 28 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（六）	实验名称	图像的轮廓提取
软件环境	Windows Visual Studio 2017	
硬件环境	PC	

实验目的

实现图像的轮廓提取。

实验内容（应包括实验题目、实验要求、实验任务等）

1. 实现目标图像的轮廓提取

编程：实现目标图像的轮廓提取

2. 实现目标图像的特征提取

编程：实现目标图像的特征提取

实验过程与实验结果

1.实现目标图像的轮廓提取

原理：

在识别图像中的目标时，往往需要对目标边缘作跟踪处理，也叫轮廓跟踪。顾名思义，轮廓跟踪就是通过顺序找出边缘点来跟踪边界的。若图像是二值图象或图像中不同区域具有不同的象素值，但每个区域内的象素值是相同的，则使用追踪算法可完成基于 4 连通或 8 连通区域的轮廓跟踪。

步骤 1：首先按从上到下，从左到右的顺序扫描图像，寻找没有标记跟踪结束记号的第一个边界起始点 A0，A0 是具有最小行和列的边界点。定义一个扫描方向变量 dir,该变量用于记录上一步中沿着前一个边界点到当前边界点的移动方向，其初始化取值为

（1）对 4 连通区域取 dir=3；

（2）对 8 连通区域取 dir=7；

步骤 2：按逆时针方向搜索当前象素的 3*3 邻域，其起始搜索方向设定如下：

（1）对 4 连通区域取（dir+3）mod 4；

（2）对 8 连通区域，若 dir 为奇数取（dir+7）mod 8；若 dir 为偶数去（dir+6）mod 8；

在 3*3 邻域中搜索到的第一个与当前像素值相同的像素便为新的边界点 An,同时更新变量 dir 为新的方向值。

步骤 3：如果 An 等于第二个边界点 A1 且前一个边界点 An-1 等于第一个边界点 A0，则停止搜索，结束跟踪，否则重复步骤 2 继续搜索。

步骤 4：由边界点 A0、A1、A2、……、An-2 构成的边界便为要跟踪的边界。

上述算法是图像轮廓跟踪最基本的算法，它只能跟踪目标图像的内边界

（边界包含在目标点集内），另外，它也无法处理图像的孔和洞。

简单的轮廓提取算法描述如下：

一个简单二值图象闭合边界的轮廓跟踪算法很简单：首先按从上到下，从左到右的顺序搜索，找到的第一个黑点一定是最左上方的边界点，记为 A。它的右，右下，下，左下四个相邻点中至少有一个是边界点，记为 B。从开始 B 找起，按右，右下，下，左下，左，左上，上，右上的顺序找相邻点中的边界点 C。如果 C 就是 A 点，则表明已经转了一圈，程序结束；否则从 C 点继续找，直到找到 A 为止。

判断是不是边界点很容易：如果它的上下左右四个邻居都是黑点则不是边界点，否则是边界点。

代码：

```
//选项：图像处理-轮廓提取-种子算法
private void ToolStripMenuItem_imgprocess_seed_Click(object sender, EventArgs
e)
{
    try
    {
        int width = objBitmap.Width;
        int height = objBitmap.Height;
        this.pictureBox_new.BackColor = Color.White;
        Bitmap bitmap = new Bitmap(width, height);
        for (int x = 1; x < width - 1; x++)
        {
            for (int y = 1; y < height - 1; y++)
            {
                int a = COMUtil.Binaryzation(objBitmap.GetPixel(x, y));
                if (a == 0)
                {
              int a1 = COMUtil.Binaryzation(objBitmap.GetPixel(x - 1, y - 1));
              int a2 = COMUtil.Binaryzation(objBitmap.GetPixel(x - 1, y));
              int a3 = COMUtil.Binaryzation(objBitmap.GetPixel(x - 1, y + 1));
              int a4 = COMUtil.Binaryzation(objBitmap.GetPixel(x, y - 1));
              int a6 = COMUtil.Binaryzation(objBitmap.GetPixel(x, y + 1));
              int a7 = COMUtil.Binaryzation(objBitmap.GetPixel(x + 1, y - 1));
              int a8 = COMUtil.Binaryzation(objBitmap.GetPixel(x + 1, y));
              int a9 = COMUtil.Binaryzation(objBitmap.GetPixel(x + 1, y + 1));
                    if (a1 + a2 + a3 + a4 + a6 + a7 + a8 + a9 == 0)
                    {
                        bitmap.SetPixel(x, y, Color.FromArgb(255, 255, 255));
                    }
                    else
                    {
                        bitmap.SetPixel(x, y, Color.FromArgb(0, 0, 0));
                    }
```

```
                }
            }
        }
        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);
    }
}

//选项：图像处理-轮廓提取-追踪算法
private void ToolStripMenuItem_imgprocess_track_Click(object sender,
EventArgs e)
{
    try
    {
        int width = objBitmap.Width;
        int height = objBitmap.Height;
        Point currentpoint = new Point { X = -1, Y = -1 };
        int[,] GOTO = new int[8, 2] { { -1, -1 }, { 0, -1 }, { 1, -1 }, { 1,
0 }, { 1, 1 }, { 0, 1 }, { -1, 1 }, { -1, 0 } };
        Bitmap bitmap = new Bitmap(width, height);
        Point point = new Point { X = -1, Y = -1 };
        bool findpoint;
        bool isfind = false;
        int direction = 0;
        for (int j = height - 1; j >= 0 && !isfind; j--)
        {
            for (int i = 0; i < width && !isfind; i++)
            {
                int a = COMUtil.Binaryzation(objBitmap.GetPixel(i, j));
                if (a == 0)
                {
                    isfind = true;
                    point.X = i;
                    point.Y = j;
                    break;
                }
            }
        }
```

```
        currentpoint = point;

        for (int x = 0; x < width; x++)
        {
            for (int y = 0; y < height; y++)
            {
                bitmap.SetPixel(x, y, Color.FromArgb(255, 255, 255));
            }
        }

        isfind = false;
        while (!isfind)
        {
            findpoint = false;
            while (!findpoint)
            {
                int temp_x = currentpoint.X + GOTO[direction, 0];
                int temp_y = currentpoint.Y + GOTO[direction, 1];
                int a = COMUtil.Binaryzation(objBitmap.GetPixel(temp_x,
temp_y));
                if (a == 0)
                {
                    findpoint = true;
                    currentpoint.X = temp_x;
                    currentpoint.Y = temp_y;
                    if (currentpoint == point)
                    {
                        isfind = true;
                    }
                    bitmap.SetPixel(temp_x, temp_y, Color.FromArgb(0, 0, 0));
                    direction--;
                    if (direction == -1)
                        direction = 7;
                    direction--;
                    if (direction == -1)
                        direction = 7;
                }
                else
                {
                    direction++;
                    if (direction == 8)
                        direction = 0;
                }
```

```
            }

        }
        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);
    }
}
```

示例图：

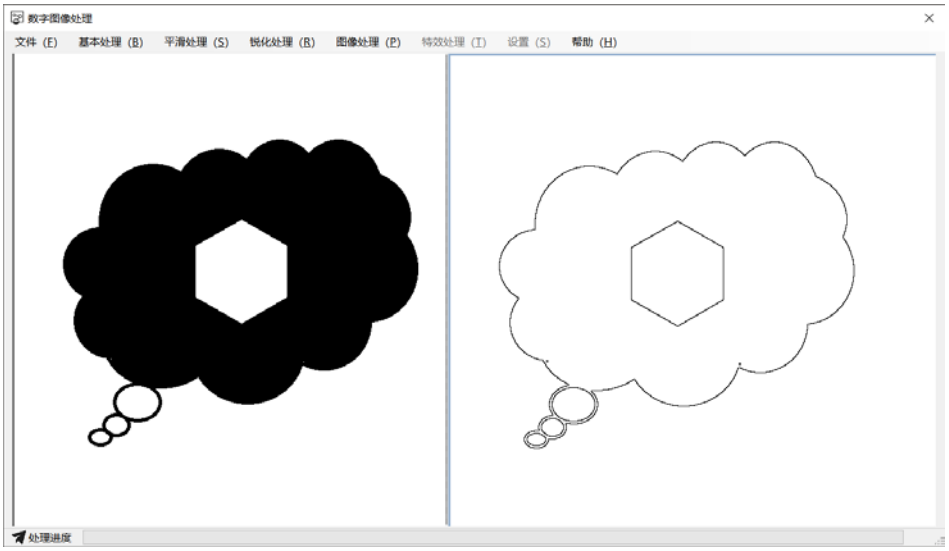

图 1-1 整体轮廓提取

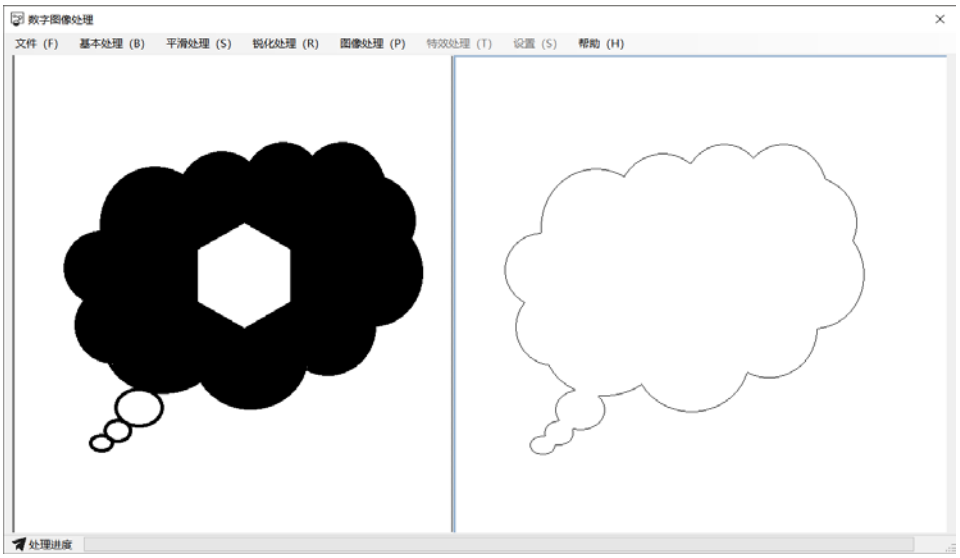

图 1-2 外轮廓提取

2.实现目标图像的特征提取

原理：

图像特征提取是计算机视觉和图像处理中的一个概念。它指的是使用计算机提取图像信息，决定每个图像的点是否属于一个图像特征。

SIFT（Scale-invariant feature transform）是一种检测局部特征的算法，该算法通过求一幅图中的特征点（interest points,or corner points）及其有关 scale 和 orientation 的描述子得到特征并进行图像特征点匹配，获得了良好效果。

SIFT 特征不只具有尺度不变性，即使改变旋转角度，图像亮度或拍摄视角，仍然能够得到好的检测效果。整个算法分为以下几个部分：

1. 构建尺度空间；
2. LoG 近似 DoG 找到关键点<检测 DOG 尺度空间极值点>；
3. 除去不好的特征点；
4. 给特征点赋值一个 128 维方向参数；
5.关键点描述子的生成；
6.根据 SIFT 进行匹配。

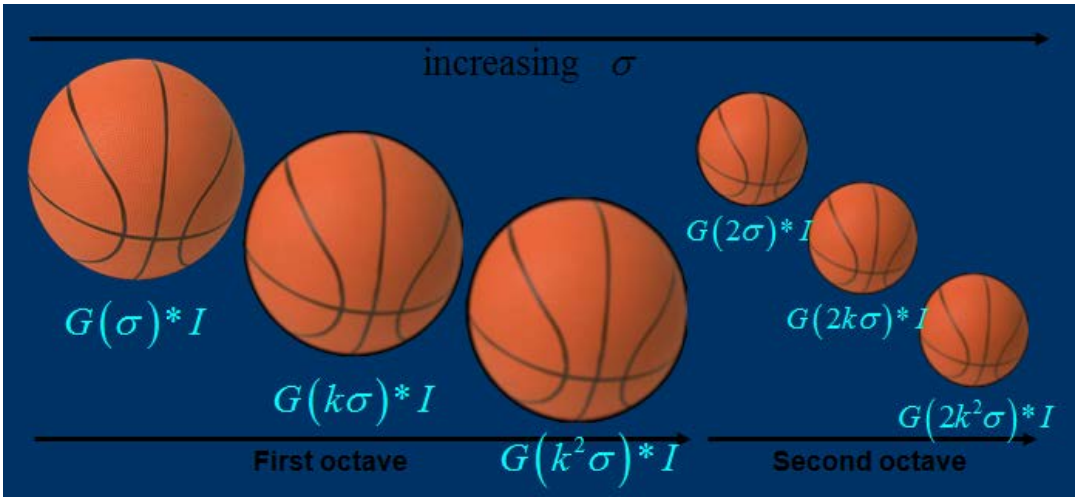

图 2-1 不同 σ 下图像尺度空间

代码：

```
private void ToolStripMenuItem_ imgprocess_getfeature_Click(object sender,
EventArgs e)
{
    try
    {
    //Load Image
    Mat c_src1 = imread("..\\Images\\3.jpg");
    Mat c_src2 = imread("..\\Images\\4.jpg");
    Mat src1 = imread("..\\Images\\3.jpg", CV_LOAD_IMAGE_GRAYSCALE);
    Mat src2 = imread("..\\Images\\4.jpg", CV_LOAD_IMAGE_GRAYSCALE);
    if (!src1.data || !src2.data)
    { std::cout << " --(!) Error reading images " << std::endl; return -1; }

    //sift feature detect
    SiftFeatureDetector detector;
```

```
    std::vector<KeyPoint> kp1, kp2;

    detector.detect(src1, kp1);
    detector.detect(src2, kp2);
    SiftDescriptorExtractor extractor;
    Mat des1, des2;//descriptor
    extractor.compute(src1, kp1, des1);
    extractor.compute(src2, kp2, des2);
    Mat res1, res2;
    int drawmode = DrawMatchesFlags::DRAW_RICH_KEYPOINTS;
    drawKeypoints(c_src1, kp1, res1, Scalar::all(-1), drawmode);//在内存中画出
特征点
    drawKeypoints(c_src2, kp2, res2, Scalar::all(-1), drawmode);
    cout << "size of description of Img1: " << kp1.size() << endl;
    cout << "size of description of Img2: " << kp2.size() << endl;

    //write the size of features on picture
    CvFont font;
    double hScale = 1;
    double vScale = 1;
    int lineWidth = 2;// 相当于写字的线条
    cvInitFont(&font, CV_FONT_HERSHEY_SIMPLEX | CV_FONT_ITALIC, hScale,
vScale, 0, lineWidth);//初始化字体，准备写到图片上的
    // cvPoint 为起笔的x，y坐标
    IplImage* transimg1 = cvCloneImage(&(IplImage)res1);
    IplImage* transimg2 = cvCloneImage(&(IplImage)res2);

    char str1[20], str2[20];
    sprintf(str1, "%d", kp1.size());
    sprintf(str2, "%d", kp2.size());


    const char* str = str1;
    cvPutText(transimg1, str1, cvPoint(280, 230), &font, CV_RGB(255, 0,
0));//在图片中输出字符

    str = str2;
    cvPutText(transimg2, str2, cvPoint(280, 230), &font, CV_RGB(255, 0,
0));//在图片中输出字符

    //imshow("Description 1",res1);
    cvShowImage("descriptor1", transimg1);
    cvShowImage("descriptor2", transimg2);
```

```
    BFMatcher matcher(NORM_L2);
    vector<DMatch> matches;
    matcher.match(des1, des2, matches);
    Mat img_match;
    drawMatches(src1, kp1, src2, kp2, matches, img_match);//,Scalar::all(-
1),Scalar::all(-1),vector<char>(),drawmode);
    cout << "number of matched points: " << matches.size() << endl;
    imshow("matches", img_match);
    cvWaitKey();
    cvDestroyAllWindows();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK, Message
BoxIcon.Stop);
    }
}
```

示例图：

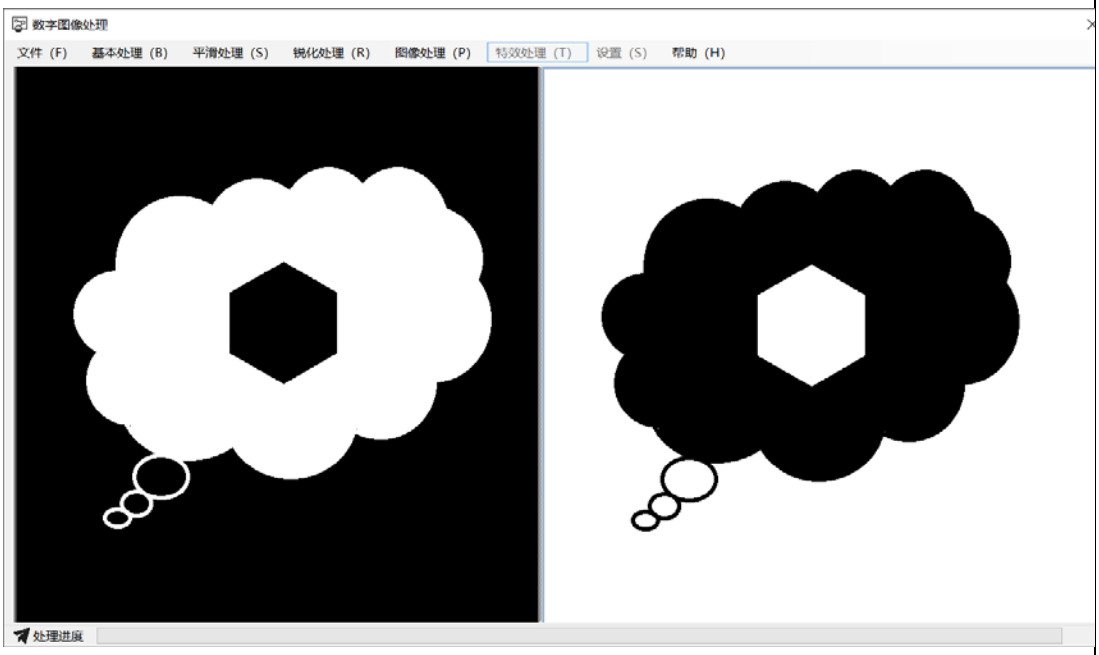

图 2-2 提取特征处理

附录

基本概念及一些补充

什么是局部特征？

- 局部特征从总体上说是图像或在视觉领域中一些有别于其周围的地方
- 局部特征通常是描述一块区域，使其能具有高可区分度
- 局部特征的好坏直接会决定着后面分类、识别是否会得到一个好的结果

局部特征需具备的特性

- 重复性
- 可区分性

•准确性

•数量以及效率

•不变性

局部特征提取算法-sift

•SIFT 算法由 D.G.Lowe 1999 年提出，2004 年完善总结。后来 Y.Ke 将其描述子部分用 PCA 代替直方图的方式，对其进行改进。

•SIFT 算法是一种提取局部特征的算法，在尺度空间寻找极值点，提取位置，尺度，旋转不变量

•SIFT 特征是图像的局部特征，其对旋转、尺度缩放、亮度变化保持不变性，对视角变化、仿射变换、噪声也保持一定程度的稳定性。

•独特性好，信息量丰富，适用于在海量特征数据库中进行快速、准确的匹配。

•多量性，即使少数的几个物体也可以产生大量 SIFT 特征向量。

•可扩展性，可以很方便的与其他形式的特征向量进行联合。

尺度空间理论

•尺度空间理论目的是模拟图像数据的多尺度特征

•其基本思想是在视觉信息图像信息处理模型中引入一个被视为尺度的参数，通过连续变化尺度参数获得不同尺度下的视觉处理信息，然后综合这些信息以深入地挖掘图像的本质特征。

描述子生成的细节

•以极值点为中心点，并且以此点所处于的高斯尺度 sigma 值作为半径因子。对于远离中心点的梯度值降低对其所处区域的直方图的贡献，防止一些突变的影响。

•每个极值点对其进行三线性插值，这样可以把此极值点的贡献均衡的分到直方图中相邻的柱子上

归一化处理

•在求出 4*4*8 的 128 维特征向量后，此时 SIFT 特征向量已经去除了尺度变化、旋转等几何变形因素的影响。而图像的对比度变化相当于每个像素点乘上一个因子，光照变化是每个像素点加上一个值，但这些对图像归一化的梯度没有影响。因此将特征向量的长度归一化，则可以进一步去除光照变化的影响。

•对于一些非线性的光照变化，SIFT 并不具备不变性，但由于这类变化影响的主要是梯度的幅值变化，对梯度的方向影响较小，因此作者通过限制梯度幅值的值来减少这类变化造成的影响。

PCA-SIFT 算法

•PCA-SIFT 与标准 SIFT 有相同的亚像素位置，尺度和主方向。但在第 4 步计算描述子的设计，采用的主成分分析的技术。

•下面介绍一下其特征描述子计算的部分：

•用特征点周围的 41×41 的像斑计算它的主元，并用 PCA-SIFT 将原来的 2×39×39 维的向量降成 20 维，以达到更精确的表示方式。

•它的主要步骤为，对每一个关键点：在关键点周围提取一个 41×41 的像斑于给定的尺度，旋转到它的主方向；计算 39×39 水平和垂直的梯度，形成一个大小为 3042 的矢量；用预先计算好的投影矩阵 n×3042 与此矢量相乘；这样生成一个大小为 n 的 PCA-SIFT 描述子。

参考资料

1. 图像的轮廓提取算法与代码 - CSDN 博客

https://blog.csdn.net/raoqiang19911215/article/details/21329961

2. 图像轮廓跟踪原理 - CSDN 博客

https://blog.csdn.net/rns521/article/details/6909780

3. SIFT 特征提取分析 - CSDN 博客

https://blog.csdn.net/abcjennifer/article/details/7639681

4. 图像特征检测(Image Feature Detection) - Wuya - 博客园

https://www.cnblogs.com/xrwang/archive/2010/03/03/ImageFeatureDetection.html