图像处理技术 —— 模板匹配算法

左 力　　2002．3．

认知是一个把未知与已知联系起来的过程。对一个复杂的视觉系统来说，它的内部常同时存在着多种输入和其它知识共存的表达形式。感知是把视觉输入与事前已有表达结合的过程，而识别也需要建立或发现各种内部表达式之间的联系。

匹配就是建立这些联系的技术和过程。建立联系的目的是为了用已知解释未知。

章毓晋《图像工程 下册》P.163

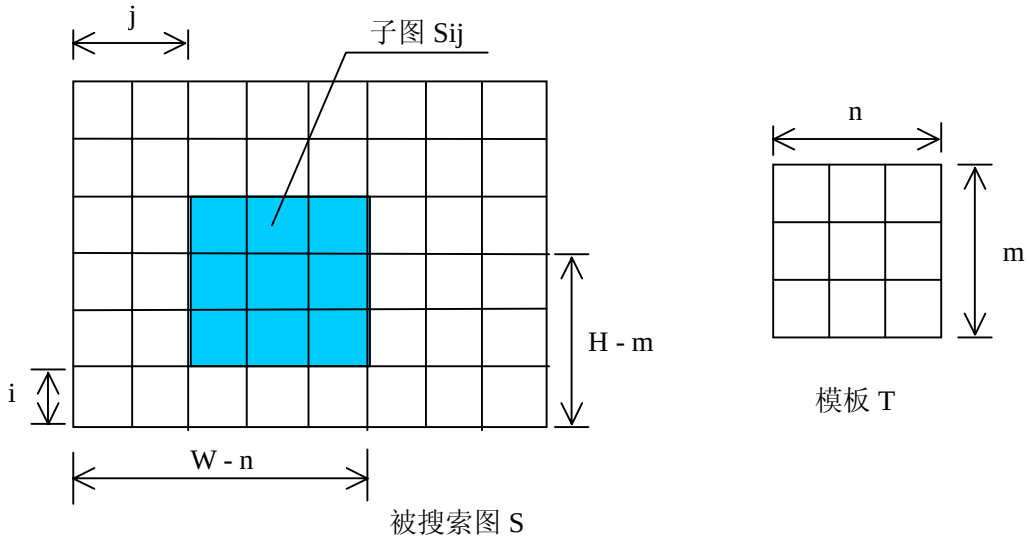
一．模板匹配的基本概念

模板就是一幅已知的小图像。模板匹配就是在一幅大图像中搜寻目标，已知该图中有要找的目标，且该目标同模板有相同的尺寸、方向和图像，通过一定的算法可以在图中找到目标，确定其坐标位置。

以 8 位图像(其 1 个像素由 1 个字节描述)为例，模板 T(m × n 个像素)叠放在被搜索图 S(W × H 个像素)上平移，模板覆盖被搜索图的那块区域叫子图 Sij。i，j 为子图左上角在被搜索图 S 上的坐标。搜索范围是：

$$\begin{cases} 1 \le i \le W - M \\ 1 \le j \le H - N \end{cases}$$

通过比较 T 和 Sij 的相似性，完成模板匹配过程。



注意：图像的数据是从下到上、从左到右排列的。

可以用下式衡量 T 和 Sij 相似性：

$$D(i,j)=\sum_{m=1}^{M}\sum_{n=1}^{N}[S^{ij}(m,n)-T(m,n)]^2$$

$$= \sum_{m=1}^{M}\sum_{n=1}^{N}[S^{ij}(m,n)]^2 - 2\sum_{m=1}^{M}\sum_{n=1}^{N}S^{ij}(m,n)\times T(m,n) + \sum_{m=1}^{M}\sum_{n=1}^{N}[T(m,n)]^2$$

上式的第一项为子图的能量，第三项为模板的能量，都与模板匹配无关。第二项是模板和子图的互相关，随(i, j)而改变。当模板和子图匹配时，该项有极大值。将其归一化，得模板匹配的相关系数：

$$R(i,j) = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N}S^{ij}(m,n)\times T(m,n)}{\sqrt{\sum_{m=1}^{M}\sum_{n=1}^{N}[S^{ij}(m,n)]^2}\sqrt{\sum_{m=1}^{M}\sum_{n=1}^{N}[T(m,n)]^2}}$$

当模板和子图完全一样时，相关系数 R(i, j) = 1。在被搜索图 S 中完成全部搜索后，找出 R 的最大值 Rmax(im, jm)，其对应的子图 Simjm 即为匹配目标。显然，用这种公式做图像匹配计算量大、速度较慢。

另一种算法是衡量 T 和 Sij 的误差，其公式为：

$$E(i,j) = \sum_{m=1}^{M}\sum_{n=1}^{N}|S^{ij}(m,n) - T(m,n)|$$

E(i, j)为最小值处即为匹配目标。为提高计算速度，取一个误差阈值 E0，当 E(i, j)> E0 时就停止该点的计算，继续下一点计算。

试验结果如下：

被搜索图尺寸	模板名称	模板大小	R 算法时间	相关系数	阈值=0	阈值=8000	误差值
					E 算法时间	E 算法时间	
256*256	corner	16*16	0.94 秒	0.9977	1.10 秒	0.55 秒	2521
	eye	13*11	0.55 秒	0.9991	0.60 秒	0.38 秒	1756
	hell	15*22	1.16 秒	0.9973	1.43 秒	0.44 秒	4323
640*480	finger	13*17	4.12 秒	0.9971	4.56 秒	2.30 秒	2117
	nose	11*17	3.41 秒	0.9787	3.85 秒	2.20 秒	4724

注：以上试验是在赛扬 600 PC 机上用 VC6.0 进行的。

结果表明：被搜索图越大，匹配速度越慢；模板越小，匹配速度越快。误差法速度较快，阈值的大小对匹配速度影响大，和模板的尺寸有关。

二．改进模板匹配算法

我在误差算法的基础上设计了二次匹配误差算法：

第一次匹配是粗略匹配。取模板的隔行隔列数据，即四分之一的模板数据，在被搜索图上进行隔行隔列扫描匹配，即在原图的四分之一范围内匹配。由于数据量大幅度减少，匹配速度显著提高。

为了合理的给出一个误差阈值 E0，我设计了一个确定误差阈值 E0 的准则：

$$E_0 = e_0 * (m+1)/2 * (n+1)/2$$

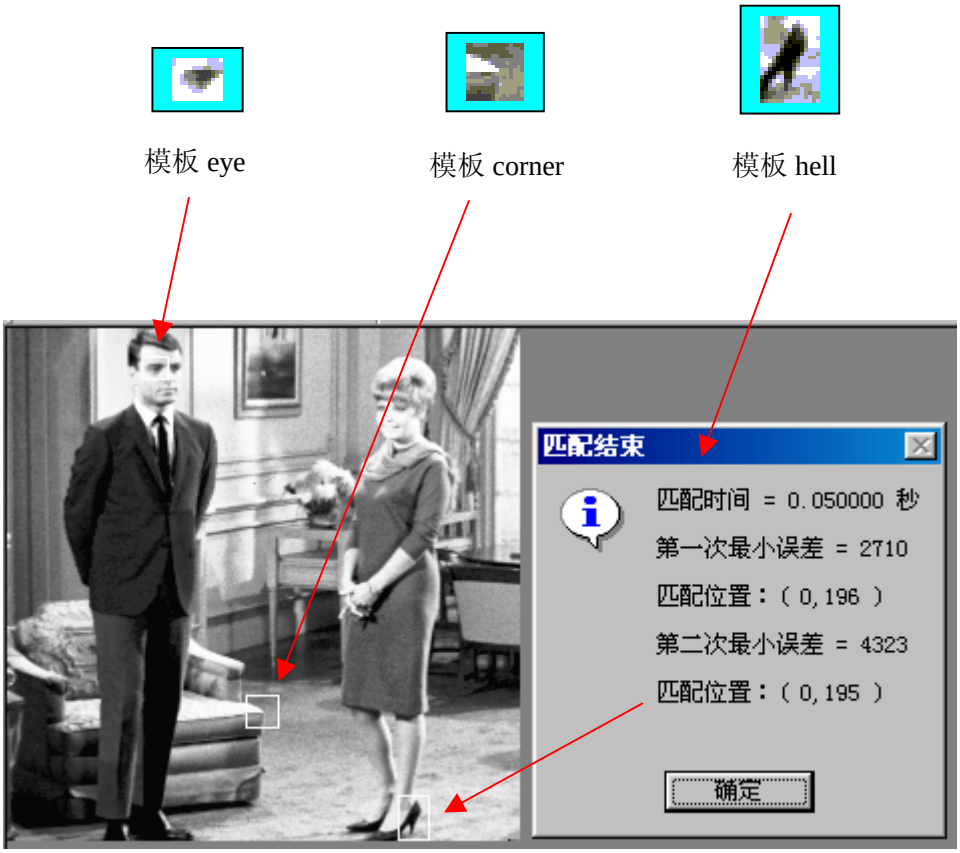
式中：e0 为各点平均的最大误差，一般取 40~50 即可；
m,n 为模板的长和宽。

第二次匹配是精确匹配。在第一次误差最小点(imin, jmin)的邻域内，即在对角点为(imin-1, jmin-1), (imin+1, jmin+1)的矩形内，进行搜索匹配，得到最后结果。

下表是相关法、误差法、二次匹配误差法这三种模板匹配算法对两幅图像进行模板匹配的结果比较，二次匹配误差法的速度比其它算法快了 10 倍左右。

被搜索图尺寸	模板名称	模板大小	卷积法时间	误差法时间	二次匹配法时间
256*256	corner	16*16	0.94 秒	0.55 秒	0.06 秒
	eye	13*11	0.55 秒	0.38 秒	0.03 秒
	hell	15*22	1.16 秒	0.44 秒	0.05 秒
640*480	finger	13*17	4.12 秒	2.30 秒	0.22 秒
	nose	11*17	3.41 秒	2.20 秒	0.16 秒

使用二次匹配误差法对 256*256 像素的被搜索图进行模板匹配的结果如下：



从上图结果可看出，第一次匹配位置是偶数，因为是隔行隔列进行搜索的；第二次则是精确位置。

三．二次匹配误差法的主要代码

```
#define AvEthreshold    40          // 各点平均误差
int    Ethreshold;                  // 误差阈值
```

```
//计算误差阈值
    Ethreshold=AvEthreshold*((lTemplateHeight+1)/2)*((lTemplateWidth+1)/2);

// 第一次粗略匹配，找出误差最小位置（nMaxHeight，nMaxWidth）
// 仅使用模板中隔行隔列的数据，在被搜索图中隔行隔列匹配
    nMinError = 99999999;
    for (i = 0; i < lHeight - lTemplateHeight +1; i=i+2)
    {
        for(j = 0; j < lWidth - lTemplateWidth + 1; j=j+2)
        {
            nError = 0;

            for (m = 0; m < lTemplateHeight; m=m+2)
            {
                for(n = 0; n < lTemplateWidth; n=n+2)
                {
                 // 指向被搜索图像倒数第 i+m 行，第 j+n 个象素的指针
                 lpSrc    = (char *)lpDIBBits + lLineBytes * (i+m) + (j+n);

                 // 指向模板图像倒数第 m 行，第 n 个象素的指针
                 lpTemplateSrc    = (char *)lpTemplateDIBBits + lTemplateLineBytes * m + n;

                 pixel = (unsigned char)*lpSrc;
                 templatepixel = (unsigned char)*lpTemplateSrc;
                 nDelta=(int)pixel-templatepixel;
                 if (nDelta<0)                    // 做绝对值运算
                 {nError = nError-nDelta;}
                 else{nError = nError+nDelta;}
                }

                if (nError>(Ethreshold)) break;   // 误差大于阈值，进入下一点计算
            }

            if (nError < nMinError)       //与最小误差比较
            {
                nMinError = nError;     // 记录最小误差及其坐标
                nMaxWidth = j;
                nMaxHeight = i;
            }
        }
    }
```