C 题 最强大脑之图形辨识

摘要

本文通过设立合理的截图相似度比较方法，建立了基于图片的矩阵比较模型。通过对附件里面的图片进行分析，先将图片转化为像素矩阵，利用模拟退火法通过比较像素矩阵的相似度找出附件中相似度最高的截图，并使用矩阵匹配模型来寻找附件中两张截图所在的位置。

对于问题一，我们运用模拟退火法和矩阵比较模型，首先建立起矩阵比较模型，利用 matlab 软件将附件中的图片转化为像素矩阵，从中分别截取 40×40 像素及 20×20 像素的图像的矩阵，再将这两种像素的矩阵，分别运用模拟退火的方法，把矩阵的相似性作为判断因子，求出最优解，根据最优解分别找出 40×40 像素及 20×20 像素相似性最高的截图。

针对问题二，运用矩阵匹配法和借用矩阵比较模型，同样的用 matlab 软件将 62 幅图（包括题目中给出的两幅截图）转化为灰度图，去掉影响矩阵匹配精确度的坐标轴外的空白区域，将剩下的灰度图运用 matlab 软件转化为像素矩阵，为了提高矩阵匹配法的速度和精度，通过二值化将像素矩阵简化为 0-1 矩阵，将 0-1 矩阵通过矩阵匹配法把两幅题目给出的截图逐一匹配其他各项 0-1 矩阵，通过矩阵匹配方法计算和比较 0-1 矩阵的差距值，当差距值最小的时候，即可推导出截图所在原图的准确位置。

本文科学合理地运用了矩阵比较模型，极大地提高了精确度，矩阵比较模型还能和不同而且恰当的方法相结合，又能提高运算速度，具备了很高的普适性。

关键词：矩阵比较模型　0-1 矩阵　像素矩阵　矩阵匹配法　模拟退火法

一、问题重述

1.1 问题背景

江苏卫视播出的“最强大脑”的节目中，有一位选手拥有可以辨识出众多副图画中一小部分截图的能力，让人惊叹，但是有人觉得难度不够，不应该选取每幅图中有着明显特征的截图，而应该用众多的画中相似性最大的部分作为考验选手的截图。

1.2 目标任务

对于问题一：对于给定的附件中的所有图片，建立模型找出其中 40 ×40 像素及 20 ×20 像素的相似性最高的的截图。

对于问题二：建立模型解决给定的截图(附件中文件“截图 1.bmp, 截图 2.bmp”）出自于第几幅图片的何处位置的问题。

二、问题分析

问题一：首先为了搜寻相似度最大的截图，考虑到截图的信息量巨大，如果直接搜寻相似的图像，未免太过主观和容易产生较大的误差，所以我们考虑将图片信息转化为矩阵信息，利用矩阵比较的相关知识和方法，利用矩阵比较模型来进行求解。第一步，利用 matlab 软件将附件中的 60 张图片转化为像素矩阵，在这些像素矩阵中分别截取 40 ×40 像素和 20×20 像素的图像的像素矩阵，将两种像素的像素矩阵分别构成自己的二部图，从而形成一个大矩阵，再运用模拟退火法，从中产生 60 个子矩阵，再将两两比较来计算相似度，保留相似度最大的子矩阵，然后，再次使用同样的方法多次求解，得到的局部最优解就是我们所求的子矩阵，再将子矩阵转化为对应的截图，所的截图即是 40 ×40 像素及 20 × 20 像素的相似性最高的的截图。

问题二：题目要求我们寻找附件中文件“截图 1.bmp, 截图 2.bmp” 出自于第几幅图片的何处位置的问题，相当于将两幅截图和附件中的所有的同等像素的截图进行匹配，于是借用第一问建立起来的矩阵比较模型，首先运用 matlab 软件将附件中的 62 幅图转化为灰度图，此时会产生坐标轴外的空白区域，很明显题目给出的需要寻找两幅截图不是空白区域，所以利用 matlab 软件去掉坐标轴外的空白区域。再将剩下的灰度图转化为像素矩阵，如果此时用灰度矩阵进行比较，为了简化比较，以及无法避免一些含有极少的信息丰富度的截图(很明显题目给出的截图不是含有极少丰富度的截图)带来的误差，所以我们将剩下的像素矩阵，设置一个合适阙值，将灰度矩阵通过二值化的方法，将灰度矩阵转化为 0-1 矩阵。 接下来将得到的附件中文件“截图 1.bmp, 截图 2.bmp”的 0-1 矩阵，

利用 matlab 软件设立的矩阵匹配法，将两幅截图的 0-1 矩阵，依次和其他的 0-1 矩阵进行匹配，分别和其他的矩阵配比得到矩阵之间的差距值，当矩阵之间的差距值为最小的时候，所得截图和截图所在的位置即为所求。

三、模型假设

1）假定截图均来自于坐标轴内的图像;
2）假定 60 幅图的的图像特征完全一致;
3）假设截图之后的像素损失不足以影响最后的结果;
4）假设坐标轴以及刻度线对相似性的影响不足以影响最后的结果

四、符号说明

符号	解释说明
$K(d_1,d_2)$	d_1,d_2 两个矩阵的相似度
S_0	像素矩阵子矩阵的左上角标
leveal	阙值
$A(m)_{i*j}$	第 m 幅图坐标为（i，j）的像素值
$B(m)$	第 m 幅图的最小差距值
$C_i(m)$	第 m 幅图最大匹配位置的横坐标
$C_j(m)$	第 m 幅图最大匹配位置的纵坐标
$C_{\min}$	最小差距值
X	原图的横长

Y	原图的纵长
$x1$	截图的横长
$y1$	截图的纵长

五、模型的建立和求解

5.1.1 模型的建立

对第一问可以采用比较矩阵相似度的方法。

由于附件中提供的 60 张图片四周都是空白区域，这对截图相似度的比较有一定的影响，故首先使用 matlab 软件将图片四周空白部分裁去，例如图形 1 转变为如图 1 所示：

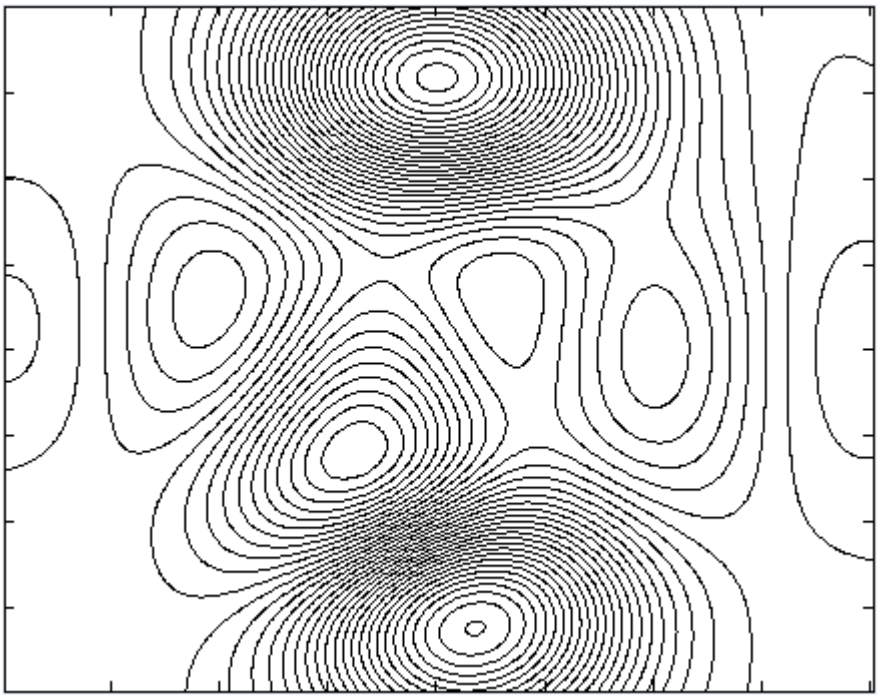

图 1

我们知道如果直接比较图片的相似度非常困难，且误差非常大，但如果能够将其转化为数值间的比较则可以将结果的精度大大提高，由此思路，我们利用

matlab 软件将 60 张图片转换为 60 个像素矩阵 $a(1)_{343*435}$，$a(2)_{343*435}$，……，$a(60)_{343*435}$，再从中截取 60 个相应的 n 阶子矩阵 $a(1)_{n*n}$，$a(2)_{n*n}$，……，$a(60)_{n*n}$，n 取决于截图的大小，从而将问题变成对像素矩阵的子矩阵相似度的比较，据此建立矩阵比较模型。

考虑到直接比较矩阵的相似度还是具有一定的复杂度，我们在比较矩阵时借鉴了空间矢量的余弦定理，将比较的两个矩阵 d_1，d_2 分别转化为对应的矢量 d_1'，d_2'，利用

$$K(d_1,d_2)=\frac{\sum_{i=1}^{n^2}(d_1'(i)\times d_2'(i))}{\sqrt{\sum_{i=1}^{n^2}(d_1'(i)\times d_2'(i))}\times\sqrt{\sum_{i=1}^{n^2}(d_2(i)\times d_2(i))}}$$

则可以将矩阵的相似度表示出来，其结果越接近于 1，则表示矩阵的相似度越大，

将矩阵依次相互比较则可以得到一个相似度的和：$\sum_{i=1}^{60}\sum_{j=1}^{60}K(a_i,a_j)$

目标式子为：

$$Max(\sum_{i=1}^{60}\sum_{j=1}^{60}K(a_i,a_j))$$

满足上式的一组矩阵即为相似度最大的一组矩阵，在当中两两比较后再转换为对应的图像即可。

5.1.2 模型的求解

将 60 个像素矩阵所产生的子矩阵用 $a(1)_{n*n}$，$a(2)_{n*n}$ … $a(60)_{n*n}$ 来表示，容易得出矩阵间的比较如图 2 所示，其构成了一个二部图：

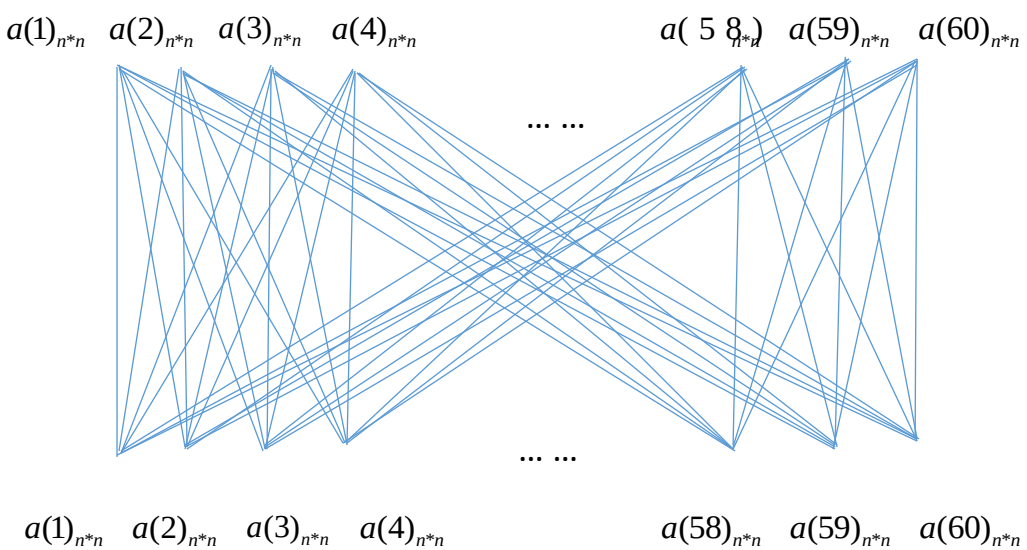

图 2

如图可知由于每一个像素矩阵所产生的子矩阵数量都非常大，如果采取依次产生相互比较未免太过复杂，可以利用模拟退火法来求出其近似最优解。

基于模拟退火法的矩阵比较模型：

升温：

将 60 个像素矩阵 $a(1)_{n*n}$， $a(2)_{n*n}$，$\cdots a(60)_{n*n}$ 合并为一个大矩阵 E：

$$E_{343*26100}=[a(1)_{343*435},a(2)_{343*435}.....a(60)_{343*435}]$$

随机产生 60 组子矩阵的左上角标 (x_n,y_n)，应满足

$$x_n\in[1,344-m]，\ y_n\in[1,435n+1-m]$$

其中 m 代表矩阵的阶数，根据角标可以得到所对应的子矩阵，此时用

$$K(d_1,d_2)=\frac{\sum_{i=1}^{n^2}(d_1^{'}(i)\times d_2^{'}(i))}{\sqrt{\sum_{i=1}^{n^2}(d_1^{'}(i)\times d_2^{'}(i))}\times\sqrt{\sum_{i=1}^{n^2}(d_2(i)\times d_2(i))}}$$

两两比较子矩阵的相似度，可得到一个值 $\sum_{i=1}^{60}\sum_{j=1}^{60}K(a_i,a_j)$，此为该组角标对应矩阵相似度的和，重复上述过程足够多次，保留 $\sum_{i=1}^{60}\sum_{j=1}^{60}K(a_i,a_j)$ 最大的一组角标，用 S_0 表示，即为升温后所得数据。

退火：

用 matlab 软件随机产生一个[1，60]范围的数 N， N 表示在 E 中取第 N 个小矩阵，同时随机产生一组矩阵的角标 (x_N,y_N)，将一个和 S_0 相同的矩阵 S 的第 N 组角标用 (x_N,y_N) 替换，再次利用

$$K(d_1,d_2)=\frac{\sum_{i=1}^{n^2}(d_1^{'}(i)\times d_2^{'}(i))}{\sqrt{\sum_{i=1}^{n^2}(d_1^{'}(i)\times d_2^{'}(i))}\times\sqrt{\sum_{i=1}^{n^2}(d_2(i)\times d_2(i))}}$$

计算此时矩阵的相似度，若得到的 $\sum_{i=1}^{60}\sum_{j=1}^{60}K(a_i,a_j)$ ＞ S_0 的 $\sum_{i=1}^{60}\sum_{j=1}^{60}K(a_i,a_j)$ 则将 S_0 用 S 替换，否则若其满足 $e^l>n$ 时同样将 S_0 用 S 替换，其中 l 表示 S 的

$\sum_{i=1}^{60}\sum_{j=1}^{60}K(a_i,a_j)$与$S_0$的$\sum_{i=1}^{60}\sum_{j=1}^{60}K(a_i,a_j)$之差，　n表示一个[0，1]范围的随机数。

重复上述过程足够多次所得S_0则为一组近似最优解。

接下来利用S_0中角标对应矩阵使用$K(d_1,d_2)=\frac{\sum_{i=1}^{n^2}(d_1'(i)\times d_2'(i))}{\sqrt{\sum_{i=1}^{n^2}(d_1'(i)\times d_2'(i))}\times\sqrt{\sum_{i=1}^{n^2}(d_2(i)\times d_2(i))}}$两两比较得出与其它矩阵相似度最大的一个矩阵，此即为相似度最大的截图所对应的矩阵。

利用该方法可以得到40×40像素的相似度最大的截图如图3所示：

图3

20×20像素相似度最大的截图如图4所示：

图4

5.2.1　利用二值化取0-1矩阵

针对问题二，首先，我们截取原图的实际的部分，即为坐标轴以内的图像，再通过PS软件，定位到图像的实际截图位置为（74，33），然后再运用matlab，将除了两幅截图以外的60幅图转化为434×342的图像。

之后，运用matlab，将附件中的62幅图像转化为与颜色无关的灰度图，并且得到62个像素矩阵，在这里，我们运用matlab软件，对每一个像素矩阵找到一个合适的阙值，然后实现图像的二值化。

$$A(m)_{i*j}=\begin{cases}1 & \mathrm{A(m)}_{i*j}>leavl\\ 0 & \mathrm{A(m)}_{i*j}<leavl\end{cases}$$

$$(m=1,2,3.....60;i=1,2,3.....X;\ j=1,2,3.....Y)$$

进而得到我们想要的60个0-1矩阵。

$$\begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 1 \\ \vdots & \ddots & \vdots & \cdots & \vdots & \iddots & \vdots \\ 1 & 1 & 0 & \vdots & \vdots & \vdots & 1 \\ \vdots & \vdots & \vdots & 1 & \vdots & 0 & \vdots \\ 1 & 0 & 1 & \vdots & 0 & \vdots & 1 \\ \vdots & \iddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ 1 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix}_{343\times435} \quad \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 1 \\ \vdots & \ddots & \vdots & \cdots & \vdots & \iddots & \vdots \\ 1 & 1 & 0 & \vdots & \vdots & \vdots & 1 \\ \vdots & \vdots & \vdots & 1 & \vdots & 0 & \vdots \\ 1 & 0 & 1 & \vdots & 0 & \vdots & 1 \\ \vdots & \iddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ 1 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix}_{343\times455}$$

接着，利用相同的方法，先对所求的两张截图灰度化，紧接着通过图像的二值化得到相应的像素矩阵，再转化为 0-1 矩阵。

。 $\begin{pmatrix} 1 & \cdots & 0 & \cdots & 1 \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ 1 & \cdots & 0 & \cdots & 1 \end{pmatrix}_{40\times 4}$

截图 1.bmp

$\begin{pmatrix} 1 & \cdots & 0 & \cdots & 1 \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ 1 & \cdots & 1 & \cdots & 0 \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ 1 & \cdots & 0 & \cdots & 1 \end{pmatrix}_{40\times 4}$

截图 2.bmp

5.2.2 矩阵匹配模型的建立和求解

得到 0-1 矩阵之后，然后将 60 幅图的 0-1 矩阵导入 txt.1 文件中，然后导入附件中文件“截图 1.bmp，截图 2.bmp”的 0-1 矩阵到 txt.2 文件中，将运用 matlab 读取 txt.1 和 txt.2 的矩阵，然后在 matlab 中运用矩阵匹配方法，将附件中文件“截图 1.bmp，截图 2.bmp ”的 0-1 矩阵和其他 60 幅图的 0-1 矩阵进行逐项匹配。

$$B_m = Min\left(\sum_{i0=1}^{x1}\sum_{j0=1}^{y1}\left|A(m)_{(i+i0-1)*(j+j0-1)} - A(m1)_{i0*j0}\right|\right)$$

(i=1,2,3.....X-x1;j=1,2,3,4.....Y-y1;m1=1,2;m=1,2,3.....60)

当截图与某一特定区域相似度越大时，其 0-1 矩阵的差距值越小，经过矩阵匹配方法的运算，最终我们得到一个 60 位数组 B_m，即 60 幅图的最小差距值。同时，我们得到一个二维数组，每一维包含两个值，即最大匹配区域的横坐标 i_m 和纵坐标 j_m。

$$C(m) = (i_m, j_m) \quad (m = 1,2,3.....60) \qquad \cdots\cdots ①$$

接下来，我们从这 60 位数组中经过对比，找出最小差距值。

$$C_{\min}=\mathrm{Min}(\mathrm{B}_m) \qquad m=(1,2,3,4,5...60) \cdots\cdots ②$$

最后得到最小差距值为$C_{\min}$，根据式子②即可得到相似度最大的图序号B_m，再根据式子①最终得到区域的起始位置i_m j_m 即为最大匹配区域。

通过 matlab 软件的矩阵匹配方法的计算如下：

针对图 1，计算得到$C_{\min}$=228，B_m=60，i_m=267，j_m=83，即在附件中图片 tuxing60.bmp 内，通过运用 PS 软件，得到我们要的截图，如下是两幅图的比较：

图 5 原图

图 6 截图

针对图 2，计算得到$C_{\min}$=144，B_m=1，i_m=95，j_m=205，即在附件中图片 tuxing1.bmp 内，如下是两幅图的比较：

图 7 原图

图 8 截图

六、模型的评价

6.1 模型优点

1）数学化，将抽象的图像比较转化为具体的矩阵，可以对于不同的要求对矩阵进行不同的数学转化。

2）多元性，得到了不同的要求下的矩阵，结合不同的方法可以解决多元的问题，比如本题的第一问，结合模拟退火法解决找寻一定像素的最大相似性的截图问题，第二问，结合矩阵匹配法解决解决给定截图搜索问题。

3）开放性，该模型可以允许在图像转化成具体的矩阵时候，可以没有限制的不断地插入不同的方法，达到转化成自己想要的矩阵，并且在得到矩阵之后，可以对矩阵进行不同的数学方法处理，而且可以多次使用不同地方法进行处理。

4）利用矩阵代替图像巧妙地避免了相似度比较的复杂度，使用模拟退火法极大地减少了任务的工作量。

6.2 模型缺点

1）由于采取了矩阵进行匹配的方法，时间复杂度比较高，运算时间相对比较长。

2）由于矩阵比较模型基于模拟退火法，故所得结果存在一定的波动性。

七、模型的改进和推广

本题中，我们采用了图像转化为矩阵然后以矩阵为基础结合适当的方法来解决不同的问题，其中包含灰度图到像素矩阵的转化，矩阵匹配模型的建立，模拟退火等方法来进行解答，在矩阵匹配的过程中，我们采取了逐项比较的方法，降低了算法的复杂度，但在一定程度上又增加了其时间复杂度，由于计算机识图是通过识别数字矩阵的方式，因此该方法可以较为方便地找出截图的原处位置。在模型的改进方面，在面对特定的问题的时候，可以选取不同的算法和矩阵建立模型，比如解决音频的配比，使用 KMP 算法和相关矩阵进行模式匹配，这样可以进一步缩短时间复杂度。提高程序性能。

在模型的推广方面，在诸如音频匹配方面，也可以部分采用现有算法和具体矩阵进行结合。

八、参考文献

【1】 阮一峰，如何识别图像边缘，http://www.ruanyifeng.com/blog/2016/07/edge-recognition.html，2016-07-22；

【2】 阮一峰，相似图片的搜索原理，http://www.ruanyifeng.com/blog/2011/07/principle_of_similar_image_search.html，2011-07-22；

【3】 姜启源，谢金星，叶俊，数学模型（第三版）：北京高等教育出版社，2003；

【4】 胡山鹰，非线性规划问题的全局优化的模拟退火模法，清华大学报(自然科学报)，2003-09；

九、附录

matlab 代码（问题一）：

40*40 像素截图：

```
load matrix.mat
```

```
rand('state', sum(clock));

S = zeros(60, 2);
S0 = [];
Sum = 0;

for i = 1 : 1000
    for j = 1 : 60
        x = 1 + round(303 * rand(1));
        y = 1 + round(395 * rand(1)) + (j - 1) * 435;
        while matrix(x : x + 39, y : y + 39) == ones(40, 40) * 255
             x = 1 + round(303 * rand(1));
             y = 1 + round(395 * rand(1)) + (j - 1) * 435;
        end
        S(j,:) = [x, y];
    end
    temp = 0;
    for k = 1 : 59
        for l = k + 1 : 60
           a = matrix(S(k, 1) : S(k, 1) + 39, S(k, 2) : S(k, 2) + 39);
           b = matrix(S(l, 1) : S(l, 1) + 39, S(l, 2) : S(l, 2) + 39);
           a1 = a(:)';
           b1 = b(:)';
           temp = dot(a1, b1) / (sqrt(sum(a1 .* a1)) * sqrt(sum(b1 .* b1)))
+ temp;
        end
    end

    if  temp > Sum;
        Sum = temp;
        S0 = S;
    end
end
S0

for i = 1 : 10000
    n = 1 + round(59 * rand(1));
    x = 1 + round(303 * rand(1));
    y = 1 + round(395 * rand(1)) + (n - 1) * 435;
    while matrix(x : x + 39, y : y + 39) == ones(40, 40) * 255
         x = 1 + round(303 * rand(1));
         y = 1 + round(395 * rand(1)) + (n - 1) * 435;
    end
```

```
    S = S0;
    S(n,:) = [x, y];

    temp = 0;
    for k = 1 : 59
        for l = k + 1 : 60
           a = matrix(S(k, 1) : S(k, 1) + 39, S(k, 2) : S(k, 2) + 39);
           b = matrix(S(l, 1) : S(l, 1) + 39, S(l, 2) : S(l, 2) + 39);
           a1 = a(:)';
           b1 = b(:)';
           temp = dot(a1, b1) / (sqrt(sum(a1 .* a1)) * sqrt(sum(b1 .* b1)))
+ temp;
        end
    end

    if temp > Sum
       Sum = temp;
       S0 = S;
    elseif exp(temp - Sum) > rand(1)
        Sum = temp;
        S0 = S;
    end
end
S0

n = 0;
result = 0;
zero = 0;
for i = 1 : 60
    zero = size(find(matrix(S0(i, 1) : S0(i, 1) + 39, S0(i, 2) : S0(i,
2) + 39) == 0), 1) + zero;
end
zero = zero / 60;
for i = 1 : 60
    if size(find(matrix(S0(i, 1) : S0(i, 1) + 39, S0(i, 2) : S0(i, 2) +
39) == 0), 1) < zero
        continue;
    end
    temp = 0;
    for j = 1 : 60
        a = matrix(S0(i, 1) : S0(i, 1) + 39, S0(i, 2) : S0(i, 2) + 39);
        b = matrix(S0(j, 1) : S0(j, 1) + 39, S0(j, 2) : S0(j, 2) + 39);
        a1 = a(:)';
        b1 = b(:)';
```

```
        temp = dot(a1, b1) / (sqrt(sum(a1 .* a1)) * sqrt(sum(b1 .* b1)))
+ temp;
    end
    if temp > result
        result = temp;
        n = i;
    end
end
n
```

20*20 像素截图：

```
load matrix.mat
rand('state', sum(clock));

S = zeros(60, 2);
S0 = [];
Sum = 0;

for i = 1 : 1000
     for j = 1 : 60
        x = 1 + round(323 * rand(1));
        y = 1 + round(415 * rand(1)) + (j - 1) * 435;
        while matrix(x : x + 19, y : y + 19) == ones(20, 20) * 255
            x = 1 + round(323 * rand(1));
            y = 1 + round(415 * rand(1)) + (j - 1) * 435;
        end
        S(j,:) = [x, y];
    end
    temp = 0;
    for k = 1 : 59
        for l = k + 1 : 60
           a = matrix(S(k, 1) : S(k, 1) + 19, S(k, 2) : S(k, 2) + 19);
           b = matrix(S(l, 1) : S(l, 1) + 19, S(l, 2) : S(l, 2) + 19);
           a1 = a(:)';
           b1 = b(:)';
           temp = dot(a1, b1) / (sqrt(sum(a1 .* a1)) * sqrt(sum(b1 .* b1)))
+ temp;
        end
    end

    if  temp > Sum;
        Sum = temp;
        S0 = S;
```

```
    end
end
S0

for i = 1 : 10000
    n = 1 + round(59 * rand(1));
    x = 1 + round(323 * rand(1));
    y = 1 + round(415 * rand(1)) + (n - 1) * 435;
    while matrix(x : x + 19, y : y + 19) == ones(20, 20) * 255
        x = 1 + round(323 * rand(1));
        y = 1 + round(415 * rand(1)) + (n - 1) * 435;
    end

    S = S0;
    S(n,:) = [x, y];

    temp = 0;
    for k = 1 : 59
        for l = k + 1 : 60
           a = matrix(S(k, 1) : S(k, 1) + 19, S(k, 2) : S(k, 2) + 19);
           b = matrix(S(l, 1) : S(l, 1) + 19, S(l, 2) : S(l, 2) + 19);
           a1 = a(:)';
           b1 = b(:)';
           temp = dot(a1, b1) / (sqrt(sum(a1 .* a1)) * sqrt(sum(b1 .* b1)))
+ temp;
        end
    end

    if temp > Sum
       Sum = temp;
       S0 = S;
    elseif exp(temp - Sum) > rand(1)
        Sum = temp;
        S0 = S;
    end
end
S0

n = 0;
result = 0;
zero = 0;
for i = 1 : 60
    zero = size(find(matrix(S0(i, 1) : S0(i, 1) + 19, S0(i, 2) : S0(i,
2) + 19) == 0), 1) + zero;
```

```
end
zero = zero / 60;
for i = 1 : 60
    temp = 0;
    if size(find(matrix(S0(i, 1) : S0(i, 1) + 19, S0(i, 2) : S0(i, 2) +
19) == 0), 1) < zero
        continue;
    end
    for j = 1 : 60
        a = matrix(S0(i, 1) : S0(i, 1) + 19, S0(i, 2) : S0(i, 2) + 19);
        b = matrix(S0(j, 1) : S0(j, 1) + 19, S0(j, 2) : S0(j, 2) + 19);
        a1 = a(:)';
        b1 = b(:)';
        temp = dot(a1, b1) / (sqrt(sum(a1 .* a1)) * sqrt(sum(b1 .* b1)))
+ temp;
    end
    if temp > result
        result = temp;
        n = i;
    end
end
n
```

Matlab 代码（问题二）：

裁剪图片：

```
close all;
clc;
clear;
file_path='D:\matalab7\work\project1\1\';
img_path_list=dir(strcat(file_path,'*.bmp'));
img_num=length(img_path_list);
if img_num>0
    for j=1:img_num
        image_name=img_path_list(j).name;
        image=imread(strcat(file_path,image_name));
        im2=imcrop(image,[74,33,434,342]);
        imwrite(im2,['D:\output1' , '\', 'mm' num2str(j) '.bmp']);
    end
end
```

灰度图二值化：

```
close all;
clc;
```

```
clear;
file_path='D:\output1\';
img_path_list=dir(strcat(file_path,'*.bmp'));
img_num=length(img_path_list);
if img_num>0
    for j1=1:img_num
        image_name=img_path_list(j1).name;
        im2=imread(strcat(file_path,image_name));
        %im3=rgb2gray(im2);
        level=graythresh(im2);
        im=im2bw(im2,level);
        [x,y]=size(im);
        fd=fopen(strcat('D:\outtext\abc',int2str(j1),'.txt'),'wt');
        for i=1:x
        for j=1:y;
        if j==y
        fprintf(fd,'%g\n',im(i,j));
        else
            fprintf(fd,'%g\t',im(i,j));
        end
    end
        end
fclose(fd);
    end
end
```

矩阵匹配：

```
clear all;
clc;
i0=1;
i1=1;
i2=1;
i3=1;
i4=1;
i5=1;
i6=1;
i7=0;

num=0;
hash=100;
for i0=1:1:60
im2=imread(strcat('D:\output1\mm',int2str(i0),'.bmp'));
level=graythresh(im2);
A=im2bw(im2,level);
```

```
B=load('D:\d2.txt');

[y,x]=size(A);
[y1,x1]=size(B);

for i1=1:1:x-x1
    for i2=1:1:y-y1
        for i3=1:1:x1
            for i4=1:1:y1
                if A(i2+i4-1,i1+i3-1)==B(i4,i3)
                    num=num+1;
                end
            end
        end
        if num>=hash
           hash=num;
            num=0;
            i5=i1;
            i6=i2;
            i7=i0;
        else
             num=0;
        end
    end
end
end
s0=sprintf('第:%f 副图\n',i7); disp(s0);
s1=sprintf('最后结果为:%f\n',hash); disp(s1);
s2=sprintf('横坐标为:%f\n',i5); disp(s2);
s3=sprintf('纵坐标为:%f\n',i6); disp(s3);
```