基于模板匹配算法的特征设计——以数字字符识别为例

一、摘要

本项目选取较为简易而具有普适性的数字字符识别为例，在实现数字识别的基础上，进一步探索基于模板匹配算法的特征设计问题，通过设计两种不同的特征提取方案对数字字符进行模板匹配并比较识别正确率，得出较为优良的特征提取方案，并将实验结论进一步推广到多种问题的应用上。

本项目包括了两大部分：

1. 实现基于模板匹配算法的数字字符识别算法
2. 深入探究上述模板匹配过程中的特征设计问题

由于时间和水平限制，本项目在完成第一部分基础上，在第二部分中仅设计了一种方案进行分析比较。若今后有时间精力，将进一步针对数字字符识别的实际问题，尝试创新，设计并比较其他特征提取方案。

二、引言

选取本项目课题的原因有以下几点：

1. 可行性：数字字符识别作为模式识别领域较为基础的方向，可参考资料丰富，且实现方法多样，便于分析比较。
2. 基础性：模板匹配算法是图像识别领域中最基本的方法，适合入手且具有代表性。因此在众多识别数字字符的方法中我选择了模板匹配算法，有助于我了解图像处理和模式识别的基础方法，并在今后的学习中加以应用。
3. 探究性：考虑到模板匹配算法具有相对较高的局限性，对模板以及处理后的检测图像有着较高的要求。因此我进一步考虑从模板特征的选取方向上入手，希望可以进一步优化该算法。

目前基于数字识别问题的特征提取方法方法较为固定。主要方法为：通过将分割后的字符图归一化，选取行、列（或对角线）求得其中的黑像素点（假设字符为黑）个数作为特征矢量，并将其与模板库进行对照，计算出与模板库图案的相关系数进行匹配判断。但在实际数字特征提取过程中，有些数字外形存在很大的相似性，且由于手写数字的个性化问题，可能会造成特征矢量的区别不大而导致识别误差。

三、方案与设计思路

因项目主要分为两部分，现将两部分分开说明：

第一部分：实现基于模板匹配算法的数字字符识别算法

A：整体思路

采用扫描窗的方式对测试图像（来源于网上）进行扫描，对每个扫描窗进行模板匹配，若最接近某个模板，就认为是模板对应的数字。采用的匹配特征是图像的像素分布，即将图像转为向量，利用余弦相似度来衡量匹配程度。为了提高模板匹配的正确率，先对图像进行二值化处理。

B：整体流程

1. 读取训练数据，统一尺寸（为程序处理方便），二值化后得到模板。
2. 对测试图片二值化处理。

3. 采用扫描窗的方式对测试图像进行扫描，对每个扫描窗进行模板匹配。用余弦相似度衡量匹配程度。
4. 对匹配的结果进行分析验证。

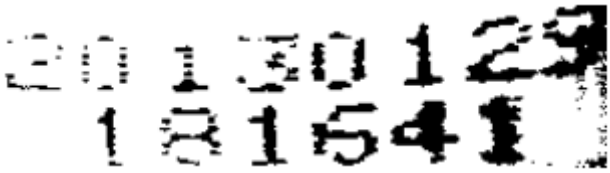
C： 步骤分析

1. **获取模板**
   - 对模板图片直接采用 im2bw 函数将其二值化即可。因为图像较小，图像亮度响度均匀，不用再局部自适应二值化。由于各个模板的大小不相同，这给后续的程序处理带来一定麻烦，解决方案是将所有模板统一到相同的尺寸（最大的模板大小）。
   - 提取出的**数字模板**如下：

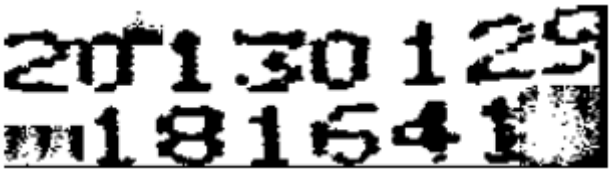
2. **测试图片的二值化**
   - 与数字模板的二值化处理不同，测试图片的不同区域平均亮度不同，直接整幅图像二值化后效果很不理想（如图 2 所示）。因此，考虑采用滑窗的方式，对图像的局部进行自适应1二值化。
   - 处理过程中，**遇到了一个这样的问题**：

     Q: 局部处理时，采取的局部块多大合适?

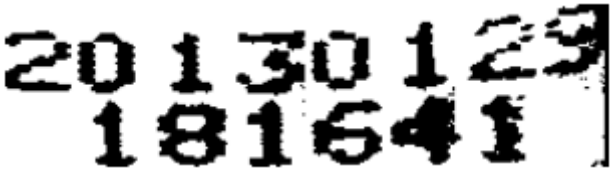
     A: 我最初采用了很小的滑窗进行处理，心想着这样的效果必定是极好的，但得到的结果却比较糟糕（如图 3 所示）。分析后发现，滑窗过小的话，可能滑窗内根本没有数字，这样很可能将背景色全部二值化为黑色。考虑到图像比较"扁长"，采取与图像高度相等的滑窗效果会更好一些。
   - 二值化效果图
     1. 直接二值化：



     2. 采用过小的滑窗二值化：



     3. 采用与测试图像等高的滑窗的二值化：



3. **匹配识别**
   - 对于每个扫描窗，依次采用每个模板与扫描窗进行匹配测试。衡量匹配度的指标是余弦

相似度，即将两幅图像均看成向量，这样一幅二值图像恰好对应一个布尔向量。余弦相似度的计算公式如图所示。

$$similarity = \frac{\vec{v_1} \cdot \vec{v_2}}{\left|\vec{v_1}\right|\left|\vec{v_2}\right|}$$

D：实验结果

- 命令行显示：

```
命令行窗口
    1.bmp        error: 1
    2.bmp        error: 1
    3.bmp        error: 1
    4.bmp        error: 0
    5.bmp        error: 1
    6.bmp        error: 1
    划痕.bmp     error: 1
    噪声.bmp     error: 1
```

- 列表显示：

  匹配结果如表所示，识别结果中荧光表示识别错误的数字，“x”表示该位置未识别出数字。

测试图片	识别结果	识别错误个数	备注
test1.bmp	201301x9181641	1	无滤波
test2.bmp	20130124181641	1	无滤波
test3.bmp	20130124181641	1	无滤波
test4.bmp	20130129181641	0	无滤波
test5.bmp	201301x9181641	1	无滤波
test6.bmp	201301x9181641	1	无滤波
划痕.bmp	20x30129181641	1	中值滤波
噪声.bmp	20130124181641	1	中值滤波

注：针对划痕图片和有椒盐噪声的图片，尝试先对图片进行中值滤波，再进行二值化等操作。我采用的是中值滤波，而不是均值滤波，一方面，根据数字图像处理的知识，消去椒盐噪声的最好滤波方式是中值滤波；另一方面，均值滤波会进一步模糊真实数字的边缘，会给模板的匹配带来更不利的影响。此处肉眼可见图片处理效果，故未进一步与其他滤波方法进行比较。

E：结果分析

1. 普通的没有噪声的图片，不必要的滤波反而可能损害图片的特征（实验中，对普通图片滤波后的效果确不升反降）。
2. 对于带椒盐噪声的图片，尽管滤波对匹配数字本身有一定负面影响，但中值滤波后的效果明显好于没有滤波的效果。

F：深入思考

1. 更换提取的特征。本实验中是把二值化后的图像看成一个向量，利用余弦相似度衡量两个向量的相近程度。如果更换特征，比如可以考虑将图像的强度和重心作为特征。
2. 由此进行第二部分实验。

G：对应代码（见附录）关于代码，直接运行main.m文件即可

- 公用部分：
  - train_preprocess.m：提取模型函数
  - binarization.m：二值化函数
  - recognize.m：模式识别函数
  - analysis_match.m：分析正确性函数
- 差异部分：
  - main.m：主运行函数
  - cal_similarity.m：计算余弦相似度函数

第二部分：深入探究上述模板匹配过程中的特征设计问题

A：整体思路

尝试设计不同的特征并用于第一部分的模板匹配过程。

B：整体流程

设计不同的特征提取方案，并将其移植到第一部分的对应位置。重复运行第一部分的主函数，并观察实验结果，进行正确率比较分析。

C：步骤分析

- 对图像块逐行统计其相关特征。特征提取方式如下。
  1. 强度：

     将模板或扫描窗的每行分为左右两部分。对于每部分，统计其黑色像素点的个数，作为该图像块的强度特征。将这些强度值放在一起，即可得到数字模板或扫描窗对应的强度向量：

$$s=\begin{pmatrix}s_{11} & s_{12} & s_{21} & s_{22} & \cdots & s_{h1} & s_{h2}\end{pmatrix}^T$$

  2. 重心：

     将模板或扫描窗的每行分为左右两部分。对于每部分，寻找其所有黑色像素的坐标，这些坐标的平均值即可认为是其黑色像素的中心，如果该部分没有黑色像素，则认为其中心在图　　像边缘外侧。将这些重心值放在一起，即可得到数字模板或扫描窗对应的重心向量：

$$c=\begin{pmatrix}c_{11} & c_{12} & c_{21} & c_{22} & \cdots & c_{h1} & c_{h2}\end{pmatrix}^T$$

- 获取这些特征向量之后，对于每个扫描窗，我们将其强度向量与重心向量 分别与每个数字模板的强度向量和重心向量对比，求其距离。为了协调不同特征之间的影响，我们可以将两个距离进行组合，得到最终的距离度量，即：

$$d=\alpha\left\|s-s_{pattern}\right\|+(1-\alpha)\left\|c-c_{pattern}\right\|,\ \ \alpha\in(0,1)$$

D：实验结果

- 命令行显示：

- 列表显示：

  匹配结果如表所示，识别结果中荧光表示识别错误的数字，“x”表示该位置未识别出数字。

测试图片	识别结果	识别错误个数	备注
test1.bmp	2813x81x9181641	3	无滤波
test2.bmp	201381xx181641	3	无滤波
test3.bmp	2813812x181641	3	无滤波
test4.bmp	28138129181641	2	无滤波
test5.bmp	88138129181641	3	无滤波
test6.bmp	28138129181241	3	无滤波
划痕.bmp	20xx01x918x64x	5	中值滤波
噪声.bmp	2813812x181x11	5	中值滤波

E：结果分析

相较第一部分的方法来看，根据像素点的统计特性的特征并未在本次试验中起到更优的效果。我分析原因有如下几点：

1. 模板与测试图像都图像进行过归一化处理，且都为印刷字体，因此减小了因像素点个数、笔画粗细等因素引起的误差。因此在这种有前提标准规范的情况下，根据统计学原理，利用所有像素点计算的余弦相似度相较于利用统计特性计算的强度、重心特征向量，更是一个充分统计量，具有良好的无偏估计性，因此识别效果较好。
2. 阿拉伯数字字符在图片中都基本位于中间区域，字符之间的像素点强度和中心较为集中在图形中央，不会有太明显的区别，因此误判可能性高。若针对于英文字母，如“L”、“C”等有明显的重心分布倾向的字符，该特征提取法可能会取得更好的效果。

F：深入思考

1. 针对经过归一化处理且有规范样式的图像识别问题，采取传统的特征方法、利用尽可能多的像素点可以取得更好的效果。
2. 推广到一般的图像识别问题上，若面对测试图片与模板图片不一致、书写样式与模板不一致 或识别图片有明显像素点分布特征的情况，可以考虑采取基于统计特征或分布特征的特征提取方法，能

一定程度上提高效率。

G：对应代码（见附录）关于代码，直接运行main2s.m文件即可

- 公用部分：
  - train_preprocess.m：提取模型函数
  - binarization.m：二值化函数
  - recognize.m：模式识别函数
  - analysis_match.m：分析正确性函数
- 差异部分：
  - main2s.m：主运行函数
  - cal_dist1.m：计算像素点强度函数
  - cal_dist2.m：计算像素点重心函数

四、收获总结

本次项目中：

- 在信号处理方面我使用了图像处理的一些基本处理方法，对图像处理有了一些基础的了解。
- 在特征提取方面，我尝试了多种图像特征提取的方法并进行分析比较。原来我只有对一维的信号进行特征提取的经验，通过这次项目，我了解到了对二维数字图像提取特征的不同方法和选取标准，发现它与一维信号的特征提取还是有比较大的差别。
- 在模式识别方面，我接触并复现了基础的模板匹配算法，并且了解到还有与神经网络等机器学习结合的模式识别方法。这学期刚好也学习了机器学习的知识，希望以后能在模式识别的方法优化方面进一步学习探索。
- 在自己原来所学的数学知识方面，通过分析两种特征提取方法得优劣，我联想到了统计学上的相关知识，这为我对特征算法的设计以及实验结果的分析提供了数学理论的支撑。

五、环境说明和代码附录（参考情况和核心部分将在每块代码前说明）

实验环境：

- 操作系统：Windows 10
- 设计软件：MATLAB R2019a

代码公用部分

- **train_preprocess.m（参考网上开源代码）**

```
function [train_img_bin, pattern_size] = train_preprocess(train_path, idx)
% 对训练模板进行预处理：统一大小，二值化

num  = length(idx);
size_mat = zeros(num, 2);

for i = 1 : num
    eval(['img=imread('''',train_path,num2str(idx(i)),'.bmp'');']);
    size_mat(i,:) = size(img);
end

pattern_size = max(size_mat);
```

```
train_img_bin = ones(pattern_size(1), pattern_size(2), num);
% figure;
for i = 1 : num
    eval(['img=imread(''',train_path,num2str(idx(i)),'.bmp'');']);
    bin_temp = im2bw(img, graythresh(img));
    % 四周补零，统一模板大小
    [h, w] = size(bin_temp);
    dh = floor((pattern_size(1)-h) / 2);
    dw = floor((pattern_size(2)-w) / 2);
    train_img_bin(dh+1:dh+h, dw+1:dw+w, i) = bin_temp;

    % subplot(2,num,i);
    % imshow(train_img_bin(:,:,i));
end

end
```

- **binarization.m（参考网上开源代码）**

```
function img_bin = binarization( img )
% 对输入的图像进行二值化，局部自适应二值化

[H, W] = size(img);
img_bin = zeros(H, W);

step = floor(W / 4);
for k = 1: step : floor(W/step)*step
    % 获取局部图像块
    if k + step - 1 > W
        w = W - k + 1;
    else
        w = step;
    end
    img_blk = img(:, k:k+w-1);
    img_blk_bin = im2bw(img_blk, graythresh(img_blk));
    img_bin(:, k:k+w-1) = img_blk_bin;
end

end
```

- **recognize.m（独自完成 / 模式识别的核心部分）**

```
function match = recognize(test_img, train_img_bin, pattern_size, thres,
idx)
% 根据模式的知识，对测试图像进行模式识别

%% 模板大小
h = pattern_size(1);
w = pattern_size(2);

%% 将测试图像二值化
% figure;
% subplot(2,1,1); imshow(test_img);
```

```
test_img_bin = binarization(test_img);
% subplot(2,1,2); imshow(test_img_bin);

%% 对图片周围延拓
[H_, W_] = size(test_img_bin);
img_resize = ones(H_+20, W_+20);
img_resize(11:H_+10, 11:W_+10) = test_img_bin;
[H, W] = size(img_resize);
mask = zeros(H, W);
match = [];

%% 扫描窗匹配
for i = 1 : H-h+1
    for j = 1 : W-w+1
        % 判断该区域是否已经识别出数字
        mask_blk = mask(i:i+h-1, j:j+w-1);
        if sum(mask_blk(:)) > thres(1) * numel(mask_blk)
           continue
        end
        % 对该区域用每个模板匹配
        img_blk = img_resize(i:i+h-1, j:j+w-1);
        similarity = zeros(length(idx), 1);
        for k = 1 : length(idx)
            similarity(k) = cal_similarity(~train_img_bin(:,:,k), ~img_blk);
        end
        % 根据阈值判定是否成功匹配
        [sort_sim, sort_idx] = sort(similarity, 'descend');
        if (sort_sim(1) > thres(2)) && (sort_sim(1)-sort_sim(2) > thres(3))
            mask(i:i+h-1, j:j+w-1) = 1;
            match = [match; floor(i-10+h/2), floor(j-10+w/2),
idx(sort_idx(1))];
        end
    end
end

% figure;
% subplot(2,1,1); imshow(img_resize);
% subplot(2,1,2); imshow(mask);

end
```

- **analysis_match.m（独自完成）**

```
function [line1, line2, error_idx] = analysis_match(match)
% 根据模式匹配的结果，分析识别的正确性

aera = [
    10 1 36 25
    14 34 39 56
    9 71 39 96
    11 103 38 134
    12 139 36 164
    7 178 35 202
    2 208 31 240
    1 243 32 274
```

```
    46 38 72 60
    45 67 72 100
    45 107 72 131
    43 137 72 169
    42 175 72 208
    40 212 72 237];
truth = [2,0,1,3,0,1,2,9,1,8,1,6,4,1];

result = (-1) * ones(1,14);
for i = 1 : size(aera,1)
    for j = 1 : size(match,1)
        if match(j,1)>aera(i,1) && match(j,1)<aera(i,3) && ...
                match(j,2)>aera(i,2) && match(j,2)<aera(i,4)
            result(i) = match(j,3);
        end
    end
end

line1 = result(1:8);
line2 = result(9:14);
error_idx = find(truth ~= result);

end
```

第一部分（差异）

- **main.m（独自完成）**

```
clear all; close all; clc;

%% 设置样本路径
train_path = 'train/';
test_path = 'test/';

%% 预处理训练样本,获得模型
idx = [0,1,2,3,4,6,8,9];
[train_img_bin, pattern_size] = train_preprocess(train_path, idx);

%% 设置识别过程中的预置参数
thres = [0.01, 0.73, 0.05];

%% 对图片中的数字进行识别
test = {'1','2','3','4','5','6','划痕','噪声'};
size_mat = zeros(length(test), 2);
for i = 1 : length(test)
    eval(['test_img=rgb2gray(imread(''',test_path,test{i},'.bmp''));']);
    size_mat(i,:) = size(test_img);

    if i == 7 % 带有划痕图片
        test_img = medfilt2(test_img); % 中值滤波
        thres = [0.01, 0.70, 0.05]; % 调整参数
    end
    if i == 8 % 带有噪声图片
        test_img = medfilt2(test_img);
        thres = [0.01, 0.72, 0.05];
```

```
    end
    % 根据模式进行匹配
    match = recognize(test_img, train_img_bin, pattern_size, thres, idx);
    % 分析匹配结果
    [line1, line2, error_idx] = analysis_match(match);
    fprintf('%s.bmp\t\terror: %d\n', test{i}, length(error_idx));
    % disp(line1); disp(line2);
end

%% 处理大小不一致的图片
mean_size = round(mean(size_mat)); % 图像平均大小
supp = {'补充1', '补充2'};
thres = [0.01, 0.72, 0.1];
for i = 1 : length(supp)
    eval(['test_img=rgb2gray(imread(''',test_path,supp{i},'.bmp''));']);
    test_img = imresize(test_img, mean_size);
    match = recognize(test_img, train_img_bin, pattern_size, thres, idx);
    [line1, line2, error_idx] = analysis_match(match);
    fprintf('%s.bmp\terror: %d\n', supp{i}, length(error_idx));
    % disp(line1); disp(line2);
end
```

- **cal_similarity.m（参考网上开源代码 / 特征提取的核心部分）**

```
function similarity = cal_similarity(img1, img2)
% 计算两个相同大小的二值图像的余弦相似度

vec1 = double(img1(:));
vec2 = double(img2(:));
assert(numel(vec1)==numel(vec2), '请确保所比较矩阵的元素个数相等！');

similarity = vec1' * vec2 / (norm(vec1) * norm(vec2));
end
```

第二部分（差异）

- **main2s.m（独自完成）**

```
clear all; close all; clc;

%% 设置样本路径
train_path = 'train/';
test_path = 'test/';

%% 预处理训练样本,获得模式的知识
train_idx = [0,1,2,3,4,6,8,9];
[train_img, centroid] = train_preprocess(train_path, train_idx);

%% 对图片中的数字进行识别
test_idx = {'1','2','3','4','5','6','划痕','噪声'};
thres = [
    0.2, 0.3, 0.4
    0.2, 0.3, 0.35
```

```
    0.2, 0.3, 0.4
    0.2, 0.3, 0.4
    0.2, 0.3, 0.38
    0.2, 0.3, 0.4
    0.2, 0.3, 0.4
    0.2, 0.3, 0.4
    ];
for i = 1 : length(test_idx)
    % for tt = 0.3 : 0.02 : 0.5

 eval(['test_img=rgb2gray(imread(''',test_path,test_idx{i},'.bmp''));']);
        if i == 8
            test_img = medfilt2(test_img); % 中值滤波
        end
        test_img = ~binarization(test_img); % 测试图像二值化
        % figure; subplot(2,1,1); imshow(test_img);

        % 根据模式进行匹配
        match = recognize(test_img, train_img, centroid, train_idx,
thres(i,:));

        % 分析匹配结果
        [line1, line2, error_idx] = analysis_match(match);
        % fprintf('tt: %f\t', tt);
        fprintf('%s.bmp\t\terror: %d\n', test_idx{i}, length(error_idx));
    % end
end
```

- **cal_dist1.m（参考自网上开源代码 / 特征提取的核心部分）**

```
function dist = cal_dist1(img_blk, train_img)
% dist1(k) = cal_dist1(img_blk, train_img{k});

[~, w] = size(img_blk);
halfw = floor(w/2);

% 逐行统计
test_sum = [sum(img_blk(:,1:halfw),2); sum(img_blk(:,halfw+1:end),2)];
train_sum = [sum(train_img(:,1:halfw),2); sum(train_img(:,halfw+1:end),2)];
% 归一化
test_sum = test_sum / norm(test_sum);
train_sum = train_sum / norm(train_sum);

dist = norm(test_sum - train_sum);

end
```

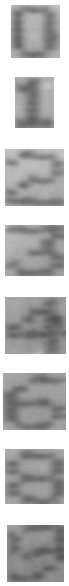
- **cal_dist1.m（参考自网上开源代码 / 特征提取的核心部分）**

```
function dist = cal_dist2(img_blk, centroid)
% dist2(k) = cal_dist2(img_blk, centroid{k});

[h, w] = size(img_blk);
```

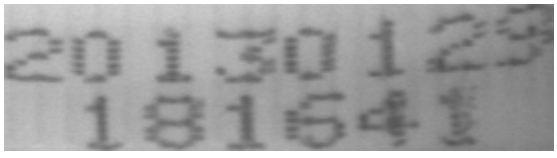
```
halfw = floor(w/2);

imgcent1 = zeros(h, 1);
imgcent2 = zeros(h, 1);
for i = 1 : h
    left = find(img_blk(i,1:halfw) == 1);
    right = find(img_blk(i,halfw+1:end) == 1);
    if ~isempty(left)
        imgcent1(i) = mean(left);
    end
    if ~isempty(right)
        imgcent2(i) = mean(right);
    else
        imgcent2(i) = halfw;
    end
end
imgcent = [imgcent1; imgcent2];
% 归一化
imgcent = imgcent / norm(imgcent);
dist = norm(imgcent - centroid);

end
```

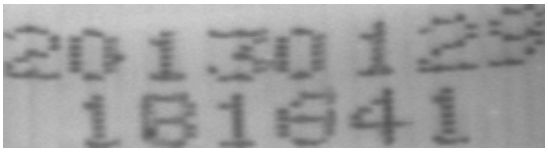
六、图像材料

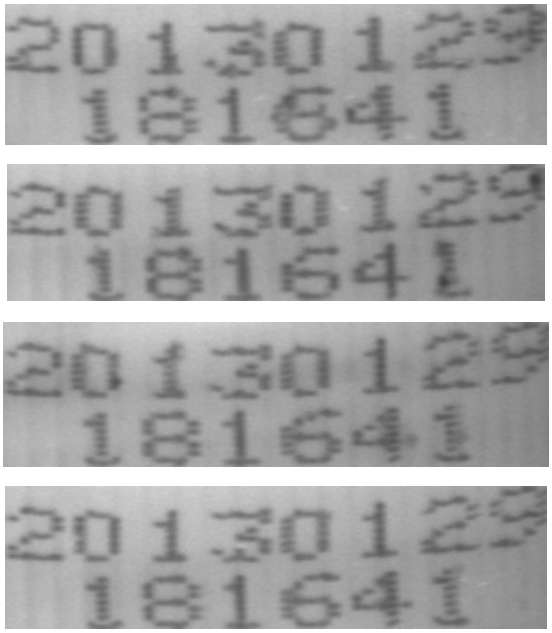
模板



测试







七、致谢

感谢严老师信号处理和模式识别课程的教导，让我们对全新的领域有了初步的认识，也让我们有更多的可能性去融合学科，为以后解决交叉复杂问题提供支撑。