**数字内容安全**

**实验报告**

****

**姓 名 詹冲**

**学 号 2023211616**

**指导教师 杨震**

**学 院 网络空间安全学院**

**2025年 4 月 4 日**

实验名称 鲁棒图像感知哈希实验 实验日期： 2025年4月4日 指导老师 杨震 得分

学院 网络空间安全学院 专业 信息安全 班次 2023211801 姓名 詹冲 学号 2023211616

1. **实验目的**

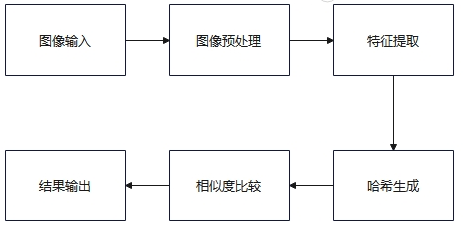
了解鲁棒性内容感知哈希技术的基本特点，设计并实现基于 DCT 的鲁棒性图像感知哈希。了解鲁棒性内容感知哈希技术在数字内容保护中的作用，掌握基于鲁棒性感知哈希的内容保护方法。

1. **实验内容**

本实验实现一种基于视觉特性的图像感知哈希算法，可通过密钥控制哈希序列，保证安全性。

1. **系统整体描述和分功能描述**

**系统整体描述**

****

**分功能描述**

1）分功能1：图像输入

用到的函数：

imread()：读取图像文件并将其转换为 MATLAB 矩阵格式进行处理。

2）分功能2：图像预处理

用到的函数：

rgb2gray()：将 RGB 彩色图像转换为灰度图像，采用加权平均法计算灰度值，减少数据维度。

imresize()：调整图像的大小，可以按指定比例缩放或通过指定目标尺寸进行插值。

3）分功能3：特征提取

用到的函数：

@dct2：用于计算二维离散余弦变换（DCT）。

blkproc()：对图像或矩阵进行块处理，将其划分为指定大小的子块，并对每个子块应用自定义函数。

4）分功能4：哈希生成

用到的函数：

randn()：生成与标准正态分布（均值为0，方差为1）相符合的随机数，返回与输入大小相同的矩阵。

fspecial('gaussian')：生成一个高斯滤波核（卷积核），减少噪声和细节。可指定滤波器大小和标准差，以控制模糊程度。

filter2()：对矩阵或图像执行二维线性滤波操作。

repmat()：将数组或矩阵按指定的行数和列数进行重复，生成一个更大的矩阵。

5）分功能5：相似度比较

用到的函数：

norm()：计算矩阵或向量的范数。

6）分功能6：结果输出

用到的函数：

figure()：创建一个新的图形窗口，用于显示图像或绘制图形。如果已有图形窗口,它会创建一个新的窗口。

nexttile()：在多子图布局中，选择下一个子图区域以进行绘图。

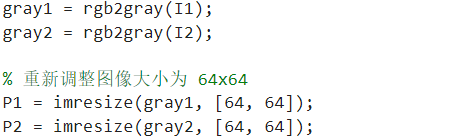
imshow()：显示图像，将图像数据以合适的比例显示在图形窗口中。

title()：为图形或子图添加标题。

1. **实验步骤、结果及分析**

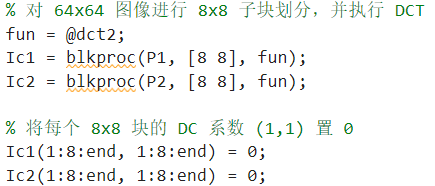
**实验步骤**

1.读入两幅图像，对图像做预处理：如果读入的是彩色图像，将其转换为灰度图像（rgb2gray）；在灰度图像中利用差值方式将图像重采样为64\*64的标准化图表示（imresize）。



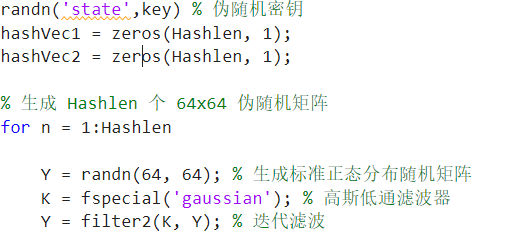
图一

2.对标准化图像进行8\*8子块划分，将标准化图像划分为（64\*64）/（8\*8）=64个子块，依次对各子块进行二维离散余弦变换（dct2）fun=@dct2，Ic=blkproc(I,[88],fun)，并依次将各分块的DC系数，即（1,1）置为0。



图二

3.生成N个64\*64伪随机矩阵：首先通过密钥伪随机生成服从标准正态的64\*64矩阵（randn('state',key)，randn），然后用高斯低通滤波器进行迭代滤波（K=fspecial('gaussian');Y=filter2(K,Y);）



图三

4.设DCT敏感度矩阵m

m=[

71.43 99.01 86.21 60.24 41.67 29.16 20.88 15.24;

99.01 68.97 75.76 65.79 50.00 36.90 27.25 20.28;

86.21 75.76 44.64 38.61 33.56 27.47 21.74 17.01;

60.24 65.79 38.61 26.53 21.98 18.87 15.92 13.16;

41.67 50.00 33.56 21.98 16.26 13.14 11.48 9.83;

29.16 36.90 27.47 18.87 13.14 10.40 8.64 7.40;

20.88 27.25 21.74 15.92 11.48 8.64 6.90 5.78;

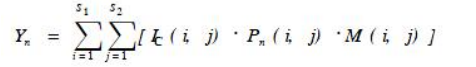
15.24 20.28 17.01 13.16 9.83 7.40 5.78 4.73]

对矩阵m进行周期延拓得到大小为64×64的矩阵M,并将其每个元素作为Ic的对应位置频率系数在特征值计算中的权。

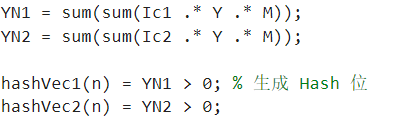


图四

5.取第一个伪随机矩阵，计算



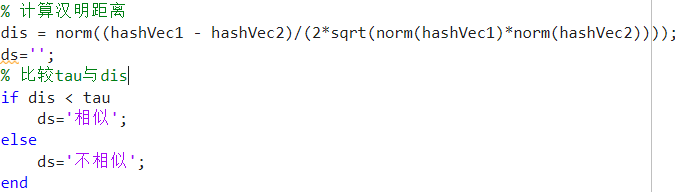
如果YN<0，则H(n)=0，反之则H(n)=1；



图五

6.循环第5步，直到将所有的N个64\*64伪随机矩阵都计算完，最终生成一个N比特的Hash向量。

7.随机选取一副图像，比较它与其他所有图像的hash向量的汉明距离dis，设定一个合适的阈值tau，如果dis<tau，则两幅图像内容一致；反之则是内容不同的两幅图像。汉明距离为dis=norm((hashVec1-hashVec2)/(2\*sqrt(norm(hashVec1)\*norm(hashVec2))));



图六

8.以DemoImages中的图片为测试集，分析实验结果。

上述过程完整代码如下：

phash.m文件

function[ds] = phash(I1,I2,tau,key,Hashlen)

    gray1 = rgb2gray(I1);

    gray2 = rgb2gray(I2);

    % 重新调整图像大小为 64x64

    P1 = imresize(gray1, [64, 64]);

    P2 = imresize(gray2, [64, 64]);

    % 对 64x64 图像进行 8x8 子块划分，并执行 DCT

    fun = @dct2;

    Ic1 = blkproc(P1, [8 8], fun);

    Ic2 = blkproc(P2, [8 8], fun);

    % 将每个 8x8 块的 DC 系数 (1,1) 置 0

    Ic1(1:8:end, 1:8:end) = 0;

    Ic2(1:8:end, 1:8:end) = 0;

    randn('state',key) % 伪随机密钥

    hashVec1 = zeros(Hashlen, 1);

    hashVec2 = zeros(Hashlen, 1);

    % 生成 Hashlen 个 64x64 伪随机矩阵

    for n = 1:Hashlen

        Y = randn(64, 64); % 生成标准正态分布随机矩阵

        K = fspecial('gaussian'); % 高斯低通滤波器

        Y = filter2(K, Y); % 迭代滤波

        % DCT 敏感度矩阵 M（周期延拓到 64x64）

        m = [

            71.43 99.01 86.21 60.24 41.67 29.16 20.88 15.24;

            99.01 68.97 75.76 65.79 50.00 36.90 27.25 20.28;

            86.21 75.76 44.64 38.61 33.56 27.47 21.74 17.01;

            60.24 65.79 38.61 26.53 21.98 18.87 15.92 13.16;

            41.67 50.00 33.56 21.98 16.26 13.14 11.48 9.83;

            29.16 36.90 27.47 18.87 13.14 10.40 8.64 7.40;

            20.88 27.25 21.74 15.92 11.48 8.64 6.90 5.78;

            15.24 20.28 17.01 13.16 9.83 7.40 5.78 4.73

        ];

        M = repmat(m, [8 8]); % 生成 64x64 矩阵

        % 计算哈希值

        YN1 = sum(sum(Ic1 .\* Y .\* M));

        YN2 = sum(sum(Ic2 .\* Y .\* M));

        hashVec1(n) = YN1 > 0; % 生成 Hash 位

        hashVec2(n) = YN2 > 0;

    end

    % 计算汉明距离

    dis = norm((hashVec1 - hashVec2)/(2\*sqrt(norm(hashVec1)\*norm(hashVec2))));

    ds='';

    % 比较tau与dis

    if dis < tau

        ds='相似';

    else

        ds='不相似';

    end

end

main.m文件

clc; clear;

% 创建一个新的图形窗口，标题为"图片相似度比较"，窗口大小为1200x2000

figure('NumberTitle', 'off', 'Name', '图片相似度比较', 'Position', [100, 100, 1200, 2000]);

key = 1234;

tau = 0.5;

Hashlen = 512;

% 获取'./DemoImages'文件夹中的所有图片文件路径

img\_path = dir('./DemoImages/\*');

img\_path = img\_path(~[img\_path.isdir]);

fileList = fullfile({img\_path.folder}.', {img\_path.name}.');

randIndex = randi(length(fileList));

selectedImagePath = fileList{randIndex};

% 读取基准图像

I1 = imread(selectedImagePath);

% 在当前窗口中创建第一个子图，并显示基准图像

nexttile;

imshow(I1);

title('基准图');

% 初始化相似图片计数器

cnt = 0;

% 遍历文件列表中的每一张图片（从第二张开始，因为第一张是基准图）

for i = 1:length(fileList)

    I2 = imread(fileList{i});

    % 使用phash算法比较基准图I1和当前图像I2的相似度

    ds = phash(I1, I2, tau, key, Hashlen);

    % 如果当前图片与基准图相似，显示该图并增加计数器

    if strcmp(ds, '相似')

        cnt = cnt + 1;

        nexttile;

        imshow(I2);

        title(ds);

    end

end

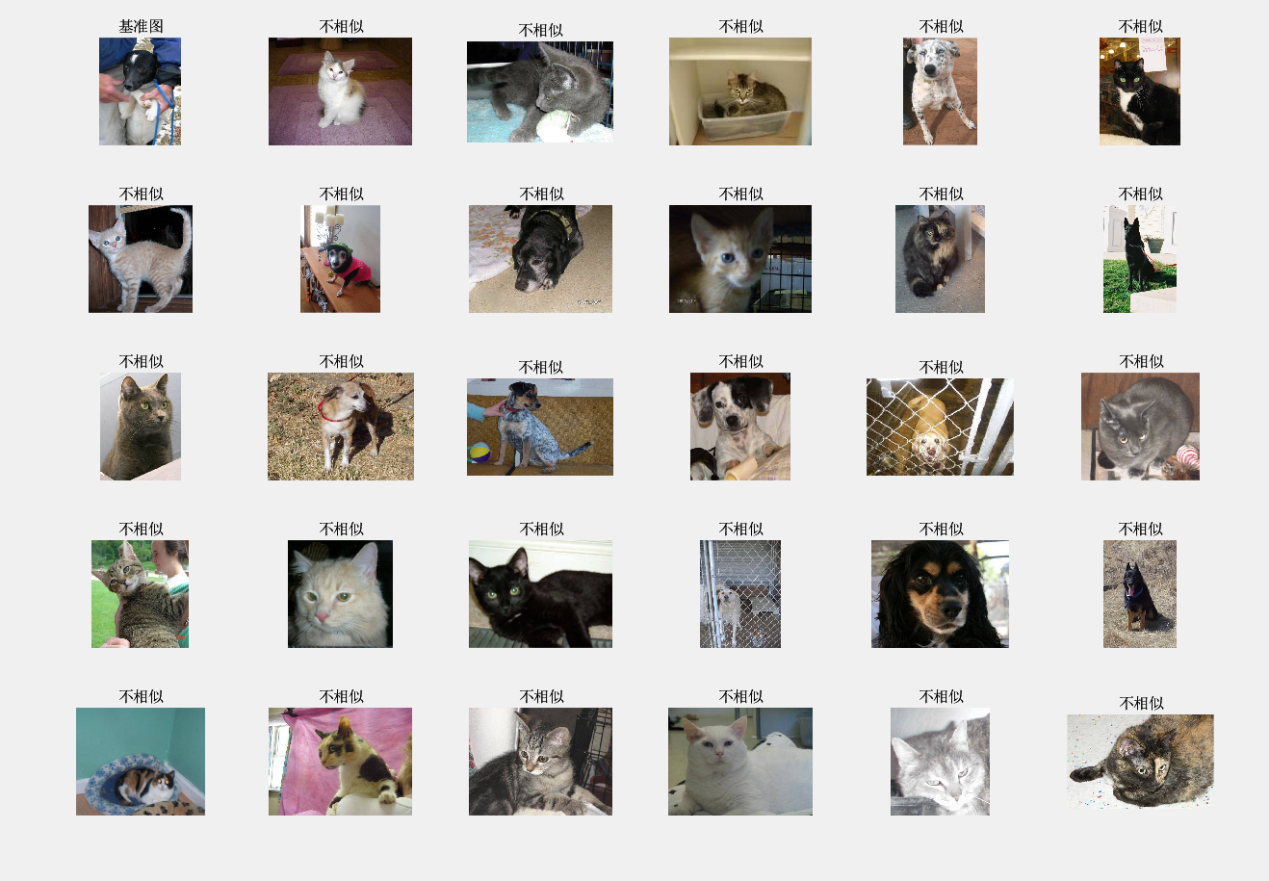
% 输出相似图像的数量

disp(cnt);

**实验结果及分析**

**实验结果**

以DemoImages中的图片为测试集，由于数据集过多，此处只显示了前三十张图片与基准图的对比。结果如下图所示。



**实验分析**

（1）关于 tau 的选取，tau 是一个经验值，其选取依据是根据测试集的准确率决定，给出选取过程。

由于给出的数据集中几乎没有相似的图片，所以我们以相似图片作为标准，如果相似图片在很小的区间，但是不能常为0时，认为此时效果很好，所有不断改变tau值，并且控制变量：N=512，key=1234，得到以下表格和图表。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 相似图像个数 | | | | |  |  |
| tau | 第一次 | 第二次 | 第三次 | 第四次 | 第五次 | 平均值 | 正确率 |
| 0.5 | 177 | 117 | 148 | 105 | 185 | 147 | 0.51 |
| 0.49 | 63 | 95 | 79 | 100 | 69 | 82 | 0.7267 |
| 0.48 | 26 | 20 | 42 | 65 | 23 | 36 | 0.88 |
| 0.47 | 14 | 40 | 3 | 13 | 38 | 22 | 0.9267 |
| 0.46 | 14 | 19 | 11 | 7 | 18 | 14 | 0.9533 |
| 0.45 | 3 | 5 | 8 | 3 | 10 | 6 | 0.98 |
| 0.44 | 0 | 0 | 0 | 1 | 3 | 1 | 0.9967 |

（2）关于 N 的选取，N 值越大 Hash 精度越高,与不同图像 Hash 碰撞的概率就越小,但鲁棒性会降低,因而需设定合适的 N 值，以满足 Hash在精度和鲁棒性之间的折衷，给出选取过程。

哈希的位数决定了其能够表示不同图像的数量时，最大为2n个。通过实验可以发现，在tau和key固定的情况下，哈希位数越大，图像感知越精细，图像相似度判断的准确性越高。但这也意味着计算机的容错能力降低，计算出的汉明距离变小，因此需要通过调整tau训练出更精准的判别阈值，从而提高训练成本。因此，折衷方案是根据图像数量N选择合适的哈希位数n，使得2n大于N，以保证足够的精度和合理的计算开销。

1. **实验中遇到的问题及改正的方法**

在本次图像相似度比较实验中，我在实现和测试过程中遇到了一些实际问题，也在逐步调试中找到了对应的解决办法。以下是我遇到的几个主要问题及个人的处理思路：

1.图像太小，显示不清楚

一开始将300张图像都显示在同一个 figure 中，每张图像占用的区域都很小，导致无法清晰辨认。后来我尝试增大subplot的分布行列、调整图像大小，但效果仍不理想。最终我改为只显示与基准图像相似的图像，这样不仅节省空间，也更聚焦于比对结果。

2.哈希位数选取

哈希长度（Hashlen）最开始我是随便设的1000，但实验过程中发现哈希位数越大判断越精准，同时容错率也下降了，有些本该算“相似”的图像被错判了“不同”。于是我尝试调低 Hashlen 到512，同时重新微调了tau 的阈值。通过对比几组数据，最终选了一个在精度和鲁棒性之间比较平衡的值。

3.不熟悉 MATLAB 语法和函数用法

作为这次实验使用 MATLAB 的初学者，我在写代码时踩了不少坑。比如一开始不理解dir 的输出结构，弄不清.name和.folder的拼接方式。此外，不同 MATLAB 版本中有些函数参数也有变动，比如某些图像处理函数的默认行为变了，需要额外指定参数才能兼容，这也让我在调试时费了些时间。