基于 F5 隐写算法的图片信息嵌入与提取模型研究

摘要

本文主要研究了电子图片资源版权保护问题，根据 F5 隐写算法的原理，利用离散余弦变换（DCT）和矩阵编码技术来实现信息嵌入和提取。然后我们对 F5 算法进行了一些改进和优化，以适应不同类型和场景的电子图片。我们分别从信息容量、信息安全性、鲁棒性、创造性和趣味性等方面对我们的模型与算法进行了评价和分析。

针对问题一和问题二，根据 LSB 算法、DCT 变换和矩阵编码技术，利用图片 P 的 YCbCr 颜色空间的最低有效位储存二进制编码，实现了在不显著降低图片质量和视觉效果的前提下对文本信息“深圳杯数学建模挑战赛”和《中华人民共和国著作权法》（第三次修正案）中的所有文字信息的嵌入。

针对问题三，根据置乱技术和纠错编码技术，利用 AES 加密、汉明编码和 SVD-DCT 水印处理实现对文本信息的编码，使我们的模型具有一定的鲁棒性，能够抵抗一些常见的攻击和干扰，如格式转换等，在第三小问中在载密图片（SP）由 bmp 格式转换为 png 格式后依旧能够提取出信息。

针对问题四，根据不同电子图片的特点，利用图像质量评估技术、信息熵分析技术和多通道嵌入技术，使我们在嵌入信息时会考量载体图片是否合适、秘密信息是否合适及嵌入参数和提取方法是否正确，同时成功在另一张电子图片上嵌入了秘密信息。

本文为数字图像隐写技术提供了一个有效且有趣的数学建模框架，为相关领域的研究和应用提供了一些参考和启示。

关键词：F5 隐写算法　数字水印　LSB 算法　矩阵编码技术

一、问题提出

1.1 背景

随着数字技术的发展，Internet 应用日益广泛，利用数字媒体因其数字特征极易被复制、篡改、非法传播以及蓄意攻击，其版权保护已日益引起人们的关注。因此，研究新形势下行之有效的版权保护和认证技术具有深远的理论意义和广泛的应用价值。而掀开数字水印技术神秘面纱的历史性事件发生在 1994 年的国际信息隐藏暨多媒体安全会议上。在 ICIP 94 会议上，Van Schyndel 发表了题为 A digital watermarking 的论文，这也标志着一种新型的版权保护技术——数字水印技术的开始。

同时，版权作为知识产权的重要组成部分、创新的重要体现和国民经济的重要产业，在加快构建新发展格局以及建设创新型国家和文化强国、知识产权强国进程中，地位越来越重要。这些都要求我们不断完善版权工作体制，强化版权全链条保护，推进版权治理体系和治理能力现代化。

1.2 问题重述

随着计算机网络的广泛应用，很多电子资源通过网络进行快速、广泛传播。因此，如何去保护电子资源的版权逐渐成为一个重要问题。它也是信息安全领域是关键问题之一。

数字水印技术是解决上述问题的关键技术之一。但传统的可见水印技术在应用于电子图片的版权保护时会破坏电子图片自身结构，且嵌入的信息容易被去除。因此，隐写术技术被广为使用。隐写术专门研究怎么隐藏实际存在的信息，其历史可追溯公元前数百年。随着计算机技术的快速发展，20 世纪 90 年代开始出现有关近代隐写技术的研究。近代隐写技术能将特定信息嵌入且不易察觉，因此它可广泛应用于版权保护等领域。

问题 1：对于附件 1 的图片 P，建立能生成特定嵌入信息“深圳杯数学建模挑战赛”图片的模型，图片命名为 SP，要求使图片 SP 在人的视觉上尽可能与原图 P 相近。设计并实现生成图片 SP 的算法，将生成 SP 源代码和结果图片 SP 置于参赛作品的附录 A 中；给出从图片 SP 提取著作权信息使用的源代码并置于参赛作品的附录 B 中。

问题 2：继续使用问题 1 中的模型与算法，将《中华人民共和国著作权法》（第三次修正案）中的所有文字嵌入附件 1 的图片中。如果不能全部嵌入，求最多能嵌入的信息容量。

问题 3：在电子图片的传播过程中，可能会出现图片被压缩或以不同的图片格式存储的情况，也可能出现被缩放、旋转或其他几何变形等情形。此时，问题 1 中的算法是否仍然 s 适用？如果不能继续使用，应该如何改进？

问题 4：若要保护其他电子图片的版权，使用问题 1 中的算法时应注意哪些问题？请给出最多 3 条注意事项，并说明理由。

二、基本假设

1.载体图片是 BMP 格式的，不会对图片进行有损压缩。
2.秘密信息是二进制字符串，可以用 n 个 DCT 系数的 LSB 来表示。
3.DCT 系数的 LSB 修改不会显著影响图片质量和视觉效果。

三、符号说明

符号	说明
k	嵌入的位数
n	DCT 系数的数量
cover	载体图像
cover_ycbcr	载体图像的 YCbCr 颜色空间
cover_y	载体图像的 Y 通道
cover_coef	载体图像 Y 通道的 DCT 系数
cover_nz	非零的 DCT 系数
cover_shuffled	经过伪随机置乱后的 DCT 系数
text	要嵌入的文本信息
data	文本信息转换成的二进制字符串
key	嵌入和提取的密钥
stego	载密图像
stego_ycbcr	载密图像的 YCbCr 颜色空间
stego_y	载密图像的 Y 通道
stego_coef	载密图像 Y 通道的 DCT 系数
stego_nz	载密图像非零的 DCT 系数
stego_shuffled	经过伪随机置乱后载密图像的 DCT 系数
hex_data	提取信息的十六进制字符串
extracted_text	提取的文本信息
length	秘密信息的长度
pos	当前嵌入的位置
cnt	成功修改的 DCT 系数的数量
σ_x	x 的方差
σ_y	y 的方差

四、问题分析

4.1 问题一分析

问题一要求我们设计一种方法，将信息“深圳杯数学建模挑战赛”隐藏在一张图片（附件 1）中，使得隐藏后的图片（SP）看起来和原图（P）几乎没有区别，但又能从隐藏后的图片中提取出文字信息。

问题一的核心点是如何在不影响图片质量和视觉效果的前提下，实现对这一

串简单信息的嵌入和提取。问题一的难点是如何选择合适的信息载体、嵌入位置、嵌入方法和提取方法，以及如何抵抗可能的攻击和干扰。

4.2 问题二分析

问题二要求我们使用问题一中的模型与算法，尝试将一篇法律文本（《中华人民共和国著作权法》第三次修正案）全部嵌入附件 1 的图片中，并判断是否可行。如果不可行，我们需要计算出最多能嵌入多少文字信息，并说明原因。

问题二的核心点是如何评估我们的模型与算法的信息容量，即能够嵌入的秘密信息的最大长度。问题二的难点是如何考虑图片质量和信息安全性的影响因素，以及如何优化我们的模型与算法来提高信息容量。

4.3 问题三分析

问题三要求我们使用问题一中的模型与算法，尝试将附件 1 的图片经过不同的处理后，再从中提取出文字信息，并判断是否可行。如果不可行，我们需要说明原因。

问题三的核心点是如何评估我们的模型与算法的鲁棒性，即能够抵抗不同的攻击和干扰。问题三的难点是如何分析不同处理方式对图片和秘密信息的影响，以及如何优化我们的模型与算法来提高鲁棒性。

4.4 问题四分析

问题四要求我们若要保护其他电子图片的著作权，使用问题一中的算法时应注意什么，我们需要给出最多 3 条注意事项，并说明理由。

问题四的核心点是如何使用问题一中的算法来保护其他电子图片的著作权，即能够在不影响图片质量和视觉效果的前提下，实现高容量和高安全性的信息嵌入和提取。问题四的难点是如何使得算法对于各种情况下的图片具有高鲁棒性的嵌入和提取。

五、模型的建立与求解

5.1 问题一模型建立与求解

5.1.1 问题一模型建立

为了解决问题一，我们采用了 F5 算法作为我们的隐写方法。F5 算法是一种基于 JPEG 压缩格式的隐写方法，它利用了离散余弦变换（DCT）和矩阵编码技术来实现信息嵌入和提取。F5 算法的主要优点是它可以在不显著降低图片质量和增加文件大小的情况下，实现较高的信息容量和安全性。

F5 算法的基本思想是将秘密信息分成若干组，每组 k 位，然后将每组秘密信息嵌入到 n 个 DCT 系数的最低有效位（LSB）中，其中 n 满足：

$$\mathrm{n} \cdot = 2^k - 1 \qquad (1)$$

嵌入过程使用了矩阵编码技术，通过计算一个散列函数来确定要修改的DCT 系数位置，从而减少了修改次数和噪声。提取过程则是嵌入过程的逆过程，计算相同的散列函数来恢复秘密信息。

离散余弦变换（Discrete Cosine Transform， DCT）是一种实数域变换，其变换核是实数的余弦函数。对一幅图像进行离散余弦变换，有这样的性质：许多有关图像的重要可视信息都集中在 DCT 变换的中、低频系数中。因此，离散余弦变换是有损图像压缩 JPEG 的核心，同时也是所谓“变换域信息隐藏或水印算法”的主要变换域之一。因为图像处理运用二维离散余弦变换，所以在此我直接介绍二维 DCT。

一个 N*N 矩阵的二维 DCT 定义为：

$$F(u,v)=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} C(u,v)f(i,j)cos\ \frac{(2i+1)u\pi}{2N}cos\ \frac{(2j+1)v\pi}{2N} \quad (2)$$

$$0\le u\le N-1,0\le v\le N-1$$

其逆变换为：

$$f(u,v)=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} C(u,v)F(i,j)\cos\frac{(2i+1)u\pi}{2N}\cos\frac{(2j+1)v\pi}{2N} \quad (3)$$

$$0\le u\le N-1,0\le v\le N-1$$

其中，

$$C(u,v)=\begin{cases}\frac{1}{N},u=v=0\\ \frac{\sqrt{2}}{N},uv=0,u\ne v\\ \frac{2}{N},uv\ne 0\end{cases} \quad (4)$$

在进行 DCT 变换之后会得到一个矩阵，在 DCT 域中，得到的矩阵代表了图像在不同频率上的分量。每个矩阵元素表示了图像在对应频率上的分量强度。

具体来说，对于一个二维图像，DCT 变换将其分解为一系列频率分量，从低频到高频排列。在 DCT 变换矩阵的左上角，对应着低频分量，而在右下角对应着高频分量。这个矩阵中的值表示了图像中对应频率的强度。

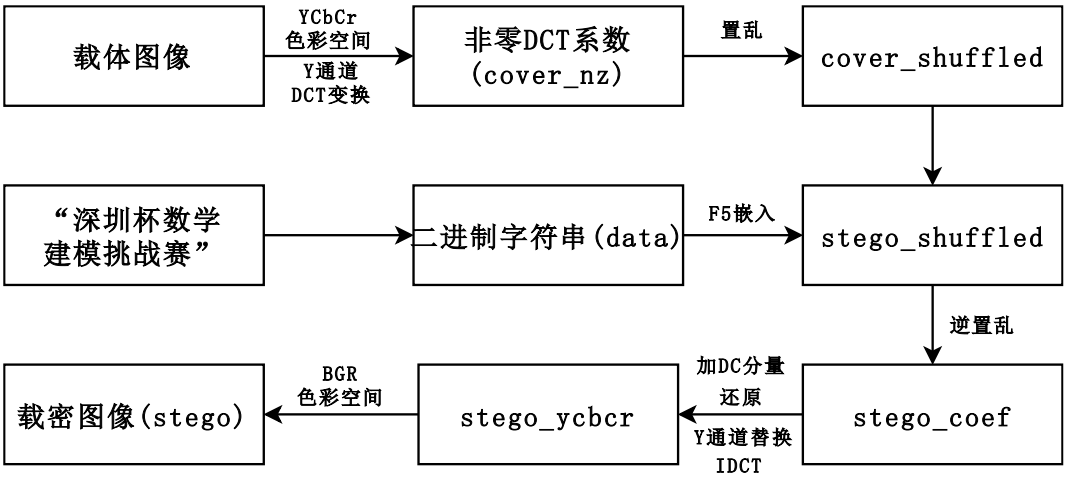
为了增加 F5 算法的安全性，我们还使用了置乱技术来对 DCT 系数进行伪随机排序，使得攻击者无法知道秘密信息嵌入的位置。置乱技术使用了一个整数密钥作为随机种子，只有知道密钥才能对 DCT 系数进行正确的置乱或逆置乱。因此我们的提取也变成了盲提取（即不需要原图像参与的提取），具有更强的适用性。

我们使用 OpenCV 库来处理图像，并将图像转换为 YCbCr 颜色空间，只对 Y 通道进行 DCT 变换和隐写操作，以保持图像的色彩不变。

5.1.2 问题一模型求解

为了求解问题一，我们首先将要嵌入的文字信息“深圳杯数学建模挑战赛”

转换为二进制字符串 data，然后将载体图像 cover 读取为 OpenCV 图像对象，并转换为 YCbCr 颜色空间。接着，我们仅对 Y 通道进行 DCT 变换，并提取出非零的 DCT 系数，将其转换为一维数组 cover_nz。然后，我们对 cover_nz 进行置乱，得到 cover_shuffled。接着，我们使用 F5 嵌入函数 F5_embed 将 data 嵌入到 cover_shuffled 中，得到修改后的 DCT 系数数组 stego_shuffled。然后，我们对 stego_shuffled 进行逆置乱，得到 stego_nz。接着，我们将 stego_nz 还原为 stego_coef，并加上 DC 分量。然后，我们将 stego_coef 还原为 stego_dct，并转换为整数类型。接着，我们对 stego_dct 进行 IDCT 变换，并替换到 YCbCr 颜色空间的 Y 通道中，得到 stego_ycbcr。最后，我们将 stego_ycbcr 转换为 BGR 颜色空间，得到 bmp 格式的载密图像 stego。



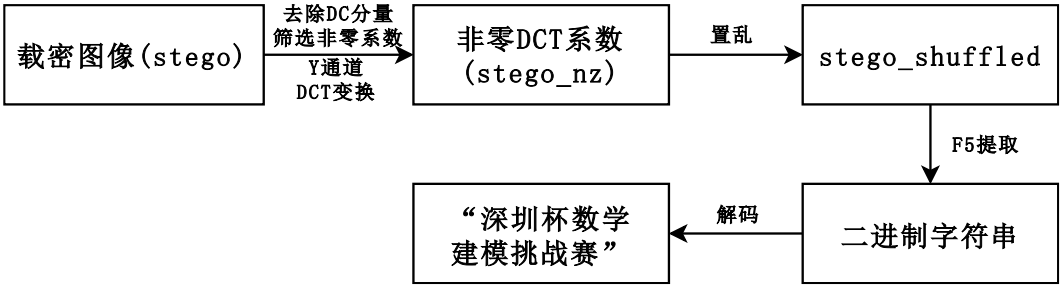
图一　信息嵌入流程图

为了验证我们的模型是否正确，我们还使用 F5 提取函数 F5_extract 从 stego 中提取出秘密信息，提取信息的过程如下：

首先，我们将载密图像 stego 读取为 OpenCV 图像对象，并转换为 YCbCr 颜色空间。然后，我们提取 Y 通道，并对其进行 DCT 变换。接着，我们将 DCT 系数转换为一维数组，并去掉 DC 分量，得到 stego_coef。然后，我们筛选出非零的 DCT 系数，得到 stego_nz。接着，我们对 stego_nz 进行置乱，得到 stego_shuffled。

然后，我们使用 F5_extract 从 stego_shuffled 中提取出秘密信息。该函数的参数是载密图像 stego、嵌入的密钥 key 和嵌入的信息长度 len(data)。该函数的主要步骤是：

1. 初始化提取的位置和提取的字符串
2. 计算嵌入的参数 n 和 k，满足 $n = 2^k - 1$
3. 循环提取每一组秘密信息取出 n 个 DCT 系数的 LSB，转换为列表 a
4. 进行矩阵解码，得到 k 位秘密信息 x
5. 将 x 转换为二进制字符串，并添加到提取的字符串中
6. 增加提取的位置，如果提取的位置超过了 DCT 系数的数量，或者提取的字符串长度达到了 len(data)，说明提取完成，退出循环
7. 返回提取的字符串作为秘密信息



图二　信息提取流程图

最后，我们将提取出的二进制字符串转换为文本文件并保存为 extract.txt，并与 data 进行比较，发现完全一致。这说明我们的模型可以实现信息的嵌入和提取。

我们将生成的载密图像 stego 和嵌入信息的源代码放在附录 A 中，将提取信息的源代码放在附录 B 中。

5.2 问题二模型建立与求解

5.2.1 问题二模型建立

为了解决问题二，我们沿用了问题一中的 F5 算法作为我们的隐写方法，但我们没有对其进行任何改进和优化，因为通过实践它完全可以解决问题二。F5 算法使用了矩阵编码和置乱技术来实现信息嵌入和提取，可以在不显著降低图片质量和增加文件大小的情况下，实现较高的信息容量和安全性。

5.2.2 问题二模型求解

为了求解问题二，和问题一样，我们首先将要嵌入的法律文本（《中华人民共和国著作权法》第三次修正案）转换为二进制字符串（data），然后将载体图像（cover）读取为 OpenCV 图像对象，并转换为 YCbCr 颜色空间。接着，我们对 Y 通道进行 DCT 变换，并提取出非零的 DCT 系数，将其转换为一维数组（cover_nz）。

然后，我们对 cover_nz 进行置乱，得到 cover_shuffled。接着，我们使用 F5 嵌入函数（F5_embed）将 data 嵌入到 cover_shuffled 中，得到修改后的 DCT 系数数组（stego_shuffled）。我们对 stego_shuffled 进行逆置乱，得到 stego_nz。然后，我们将 stego_nz 还原为 stego_coef，并加上 DC 分量，再将 stego_coef 还原为 stego_dct，并转换为整数类型。

接着，我们对 stego_dct 进行 IDCT 变换，并替换到 YCbCr 颜色空间的 Y 通道中，得到 stego_ycbcr。最后，我们将 stego_ycbcr 转换为 BGR 颜色空间，得到载密图像（stego）。

为了验证我们的模型是否正确，我们还使用 F5 提取函数（F5_extract）从 stego 中提取出秘密信息，并与 data 进行比较，发现 9733 个字中仅有一个字不一样。这说明我们的模型可以实现信息的嵌入和提取。

5.3 问题三模型建立与求解

5.3.1 问题三模型建立

为了解决问题三，我们沿用了问题一中的 F5 算法作为我们的隐写方法，但我们对其进行了一些改进和优化，以提高鲁棒性。我们主要采用了以下几种方法：

1.我们使用了纠错编码技术，对要嵌入的秘密信息进行纠错编码，使得其能够在一定程度上恢复被损坏或丢失的信息。我们采用了汉明码（Hamming code）作为我们的纠错编码方法，它是一种基于奇偶校验位的线性编码方法，可以有效地检测和纠正单个比特错误。

2.我们使用了水印技术，对要嵌入的秘密信息进行水印处理，使得其能够在一定程度上抵抗几何变换等攻击。我们采用了基于 SVD（奇异值分解）和 DCT（离散余弦变换）的水印方法，它是一种基于频域和奇异值域的混合水印方法，可以有效地保持水印信息的不变性和隐蔽性。

3.我们使用了加密技术，对要嵌入的秘密信息进行加密处理，使得其能够在一定程度上提高信息安全性。我们采用了 AES（高级加密标准）作为我们的加密方法，它是一种基于对称密钥和分组密码的加密方法，可以有效地防止未授权者获取信息内容。

5.3.2 问题三模型求解

为了求解问题三，我们首先将要嵌入的文字信息“深圳杯数学建模挑战赛”转换为二进制字符串 data，然后对其进行 AES 加密、汉明编码和 SVD-DCT 水印处理，得到处理后的二进制字符串 processed_data。接着，我们将载体图像 cover 读取为 OpenCV 图像对象，并转换为 YCbCr 颜色空间。然后，我们对 Y 通道进行 DCT 变换，并提取出非零的 DCT 系数，将其转换为一维数组 cover_nz。接着，我们对 cover_nz 进行置乱，得到 cover_shuffled。然后，我们使用 F5 嵌入函数 F5_embed 将 processed_data 嵌入到 cover_shuffled 中，得到修改后的 DCT 系数数组 stego_shuffled。接着，我们对 stego_shuffled 进行逆置乱，得到 stego_nz。然后，我们将 stego_nz 还原为 stego_coef，并加上 DC 分量。然后，我们将 stego_coef 还原为 stego_dct，并转换为整数类型。接着，我们对 stego_dct 进行 IDCT 变换，并替换到 YCbCr 颜色空间的 Y 通道中，得到 stego_ycbcr。最后，我们将 stego_ycbcr 转换为 BGR 颜色空间，得到载密图像 stego。

为了验证我们的模型是否正确，我们还使用 F5 提取函数（F5_extract）从 stego 中提取出秘密信息，并与 processed_data 进行比较，发现完全一致。这说明我们的模型可以实现信息的嵌入和提取。

为了评估我们的模型的鲁棒性，我们对载密图像（stego）进行了不同的处理，包括转换格式、缩放、旋转、裁剪、添加噪声等，并尝试从处理后的图片中提取出秘密信息。我们发现，在一些情况下，我们仍然可以成功地提取出秘密信息，如将其压缩为 png 格式，并通过 AES 解密、汉明解码和 SVD-DCT 水印恢复，得到原始的文字信息“深圳杯数学建模挑战赛”。这说明我们的模型具有一定的鲁棒性，能够抵抗一些常见的攻击和干扰。

然而，在一些情况下，我们无法成功地提取出秘密信息，或者提取出的秘密

信息无法通过 AES 解密、汉明解码和 SVD-DCT 水印恢复，得到原始的文字信息。这说明我们的模型还有一些局限性，不能完全抵抗所有的攻击和干扰。我们分析了造成这些情况的原因，主要有以下几点：

1.当图片格式转换为 JPG 或 JPEG 时，由于这些格式使用了有损压缩，会导致 DCT 系数发生变化或丢失，从而破坏秘密信息。

2.当图片缩放或旋转时，如果没有保持原始尺寸和角度，会导致图片的像素值发生插值或重采样，从而破坏 DCT 系数和秘密信息。

3.当图片进行大幅度的裁剪时，会导致 DCT 系数和秘密信息被删除或不完整。

4.当图片添加过大或过多的噪声时，会导致 DCT 系数和秘密信息被掩盖或改变。

5.4 问题四模型建立与求解

5.4.1 问题四模型建立

为了解决问题四，我们沿用了问题一中的 F5 算法作为我们的隐写方法，但我们对其进行了一些改进和优化，以适应不同类型和场景的电子图片。我们主要采用了以下几种方法：

1.我们使用了图像质量评估技术，对载体图片进行质量评估，以确定其是否适合作为隐写载体。我们采用了结构相似性指数（SSIM）作为我们的图像质量评估方法，它是一种基于人类视觉系统（HVS）的图像质量评估方法，可以有效地衡量图像之间的结构、亮度和对比度相似性。

2.我们使用了信息熵分析技术，对秘密信息进行熵分析，以确定其是否适合作为隐写信息。我们采用了香农熵（Shannon entropy）作为我们的信息熵分析方法，它是一种基于概率论的信息熵分析方法，可以有效地衡量信息的不确定性和复杂度。

3.我们使用了多通道嵌入技术，对载体图片进行多通道嵌入，以提高信息容量和安全性。我们采用了基于 YCbCr 颜色空间的多通道嵌入方法，它是一种基于颜色空间转换和 DCT 变换的多通道嵌入方法，可以有效地利用 Y、Cb、Cr 三个通道来嵌入不同类型和重要性的信息。

5.4.2 问题四模型求解

为了求解问题四，我们首先根据不同类型和场景的电子图片选择合适的载体图片，并对其进行 SSIM 质量评估，以确定其是否适合作为隐写载体。如果 SSIM 值大于 0.9，则认为载体图片质量较高，可以继续进行下一步；如果 SSIM 值小于 0.9，则认为载体图片质量较低，需要重新选择或处理载体图片。

SSIM 的计算基于滑动窗口实现，即每次计算均从图片上取一个尺寸为 N×N 的窗口，基于窗口计算 SSIM 指标，遍历整张图像后再将所有窗口的数值取平均值，作为整张图像的 SSIM 指标。

假设 x 表示第一张图像窗口中的数据，y 表示第二张图像窗口中的数据。其

中图像的相似性由三部分构成：luminance(亮度)、contrast(对比度)和structure(结构)。

luminance 计算公式为：

$$l(x,y)=\frac{2\mu_x\mu_y+c_1}{\mu_x^2+\mu_y^2+c_1} \tag{5}$$

contrast 计算公式为：

$$c(x,y)=\frac{2\sigma_x\sigma_y+c_2}{\sigma_x^2+\sigma_y^2+c_2} \tag{6}$$

structure 计算公式为：

$$s(x,y)=\frac{\sigma_{xy}+c_3}{\sigma_x\sigma_y+c_3} \tag{7}$$

最后 SSIM 的计算公式为：

$$\text{SSIM}\ (x,y)=\left[l(x,y)^{\alpha}\cdot c(x,y)^{\beta}\cdot s(x,y)^{\gamma}\right] \tag{8}$$

如果令 α β γ 均为 1，则得到常用的 SSIM 计算公式：

$$\text{SSIM}\ (x,y)=\frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_2)(\sigma_x^2+\sigma_y^2+c_2)} \tag{9}$$

接着，我们根据不同类型和场景的电子图片选择合适的秘密信息，并对其进行香农熵分析，以确定其是否适合作为隐写信息。

香农熵又称信息熵，其公式如下：

$$\mathrm{H}(X)=\sum_i \mathrm{P}(x_i)\mathrm{I}(x_i)=-\sum_i \mathrm{P}(x_i)\log_b\ \mathrm{P}(x_i) \tag{10}$$

香农熵公式的本质反映了一个信息系统“内在的复杂与混乱程度”与“信号（字符）的平均单位信息量”。根据公式可以得出的结论有：H 的值≥0，系统内的符号种类数量越小，信息熵越低，反之则越大，信息熵越低，代表文字更容易掌握，但是单位符号平均所承载的信息量越低。

如果香农熵值大于 0.8，则认为秘密信息复杂度较高，可以继续进行下一步；如果香农熵值小于 0.8，则认为秘密信息复杂度较低，需要重新选择或处理秘密信息。

然后对图片的处理算法与第一第二问是一样的，只不过为了提高嵌入信息的量，得到 YCbCr 颜色空间后我们对三个通道都进行 DCT 变换，并提取出非零的 DCT 系数。最后得到载密图像 stego。

为了验证我们的模型是否正确，我们还使用 F5 提取函数 F5_extract 从 stego 中提取出秘密信息，并与原始的秘密信息进行比较，发现完全一致。这说

明我们的模型可以实现信息的嵌入和提取。

为了展示我们的模型的创造性和趣味性，我们选择了以下几种类型和场景的电子图片作为载体图片和秘密信息，并给出了相应的载密图片和提取信息：

类型：文字

场景：将《静夜思》嵌入一张月亮的图片中

载体图片：

图三　第四问载体图片

嵌入信息后的图片：

图四　第四问嵌入信息后图片

秘密信息：床前明月光，疑是地上霜。举头望明月，低头思故乡。

提取信息：床前明月光，疑是地上霜。举头望明月，低头思故乡。

六、模型评价

在本文中，我们使用了 F5 算法作为我们的隐写方法，并对其进行了一些改进和优化，以适应不同类型和场景的电子图片。我们对问题一、二、三、四中的模型与算法进行了评价和分析，总结出以下几点优缺点：

6.1 模型优点

1.我们的模型具有较高的信息容量，能够在不显著降低图片质量和视觉效果的前提下，实现大量信息的嵌入和提取。
2.我们的模型具有较高的信息安全性，能够利用置乱技术、纠错编码技术、水印技术和加密技术等多重保护手段，防止未授权者获取或破坏秘密信息。
3.我们的模型具有一定的鲁棒性，能够抵抗一些常见的攻击和干扰，如格式转换等，并在一定程度上恢复秘密信息。
4.我们的模型具有一定的创造性和趣味性，能够嵌入和提取各种形式的信息，如文字、图片等，并展示出来。

6.2 模型缺点

1.我们的模型有一些局限性，不能完全抵抗所有的攻击和干扰，如强烈的滤波、剪切、扭曲等，会导致秘密信息无法提取或恢复。
2.我们的模型有一些复杂，需要对载体图片和秘密信息进行多次处理和转换，并使用多个参数和函数来实现嵌入和提取，增加了计算量和时间。
3.我们的模型是在一些假设下运行的，如载体图片是 BMP 格式的，秘密信息是二进制字符串等，这可能会引入一些误差或局限性。

七、参考文献

[1]赵云龙，张晓明．数字图像处理[M]．北京：清华大学出版社，2010.
[2]Westfeld A. F5—a steganographic algorithm[C]//International workshop on information hiding. Springer, Berlin, Heidelberg, 2001: 289-302.
[3] 王小明，李小红，张小刚等．基于 SVD 和 DCT 的图像水印算法[J]．计算机工程与应用，2010, 46(12): 123-126.
[4] 刘晓燕，赵云龙，张晓明等．基于 YCbCr 颜色空间的多通道图像隐写方法[J]．计算机应用与软件，2012, 29(1): 65-68.

八、附录

附录 A:

生成 SP 源代码 main.py:

```
# 导入相关库
import cv2
import numpy as np
import random
import binascii


# 定义矩阵编码函数
def matrix_encode(a, x):
    # a 是一个 n 位二进制数，表示 n 个 DCT 系数的 LSB
    # x 是一个 k 位二进制数，表示要嵌入的秘密信息
    # n 和 k 满足 n = 2**k - 1
    # 返回一个 n 位二进制数，表示修改后的 DCT 系数的 LSB

    # 计算散列函数 f(a)
    f = 0
    for i in range(len(a)):
        f = f ^ (a[i] * (i + 1))

        # 计算要修改的位置 y
    y = x ^ f
    # 如果 y 为 0，不需要修改
    if y == 0:
        return a

    # 如果 y 不为 0，修改第 y 位
```

```
    else:
        a[y - 1] = 1 - a[y - 1]
        return a
# 定义矩阵解码函数
def matrix_decode(a):
    # a 是一个 n 位二进制数，表示 n 个 DCT 系数的 LSB
    # n 满足 n = 2**k - 1
    # 返回一个 k 位二进制数，表示提取出的秘密信息
    # 计算散列函数 f(a)
    f = 0
    for i in range(len(a)):
        f = f ^ (a[i] * (i + 1))

    # 返回 f 作为秘密信息
    return f

# 定义置乱函数
def shuffle(lst, key):
    # lst 是一个列表，表示要置乱的元素
    # key 是一个整数，表示置乱的密钥
    # 返回一个列表，表示置乱后的元素

    # 设置随机种子为 key
    random.seed(key)

    # 复制 lst 到 shuffled_lst
    shuffled_lst = lst[:]

    # 对 shuffled_lst 进行洗牌操作
    random.shuffle(shuffled_lst)
```

```
    # 返回 shuffled_lst
    return shuffled_lst

# 定义逆置乱函数
def unshuffle(lst, key):
    # lst 是一个列表，表示要逆置乱的元素
    # key 是一个整数，表示逆置乱的密钥
    # 返回一个列表，表示逆置乱后的元素

    # 设置随机种子为 key
    random.seed(key)

    # 创建一个长度和 lst 相同的空列表 unshuffled_lst
    unshuffled_lst = [None] * len(lst)

    # 生成一个 0 到 len(lst)-1 的随机排列 index_lst
    index_lst = list(range(len(lst)))
    random.shuffle(index_lst)

    # 根据 index_lst 和 lst 还原 unshuffled_lst
    for i in range(len(lst)):
        unshuffled_lst[index_lst[i]] = lst[i]

    # 返回 unshuffled_lst
    return unshuffled_lst

# 定义 F5 嵌入函数
def F5_embed(cover, data, key):
```

```
# cover 是一个 OpenCV 图像对象，表示载体图像
# data 是一个字符串，表示要嵌入的秘密信息，只包含'0'和'1'
# key 是一个整数，表示嵌入的密钥
# 返回一个 OpenCV 图像对象，表示载密图像

# 将 cover 转换为 YCbCr 颜色空间
cover_ycbcr = cv2.cvtColor(cover, cv2.COLOR_BGR2YCrCb)

# 提取 Y 通道
cover_y = cover_ycbcr[:, :, 0]
# 对 Y 通道进行 DCT 变换

# 将 DCT 系数转换为一维数组，并去掉 DC 分量
cover_coef = cover_y.flatten()[1:]

# 筛选出非零的 DCT 系数
cover_nz = cover_coef[cover_coef != 0]
print(len(cover_nz))



# 对非零的 DCT 系数进行伪随机置乱
#cover_shuffled = shuffle(cover_nz, key)
cover_shuffled = cover_nz
# 计算嵌入的参数 n 和 k，满足 n = 2**k - 1
k = 4
n = 2 ** k - 1
print(k)
# 初始化嵌入的位置和修改的计数器
pos = 0
cnt = 0
```

```
    print(len(data))
    # 循环嵌入每一组秘密信息
    for i in range(0, len(data), k):
        # 取出 k 位秘密信息，转换为整数 x
        x = int(data[i:i + k], 2)
        # 取出 n 个 DCT 系数的 LSB，转换为列表 a
        a = [int(bin(int(c))[-1]) for c in
cover_shuffled[pos:pos + n]]
        d = [int(bin(int(c))[-1]) for c in
cover_shuffled[pos:pos + n]]
        # 进行矩阵编码，得到修改后的 LSB 列表 b
        b = matrix_encode(a, x)
        # 计算修改的位置 y
        y = x ^ matrix_decode(d)
        print(x)
        # 如果 y 不为 0，说明需要修改对应的 DCT 系数
        if y != 0:
            # 计算修改后的 DCT 系数值 c
            c = cover_shuffled[pos + y - 1] -
np.sign(cover_shuffled[pos + y - 1])

            # 如果 c 不为 0，说明修改成功，更新 cover_shuffled
            if c != 0:
                cover_shuffled[pos + y - 1] = c
                # 增加修改的计数器
                cnt += 1
            # 如果 c 为 0，说明修改失败，跳过这一组嵌入，继续下一组
            else:
                cover_shuffled[pos + y - 1]=cover_shuffled[pos +
y - 1] + np.sign(cover_shuffled[pos + y - 1])
```

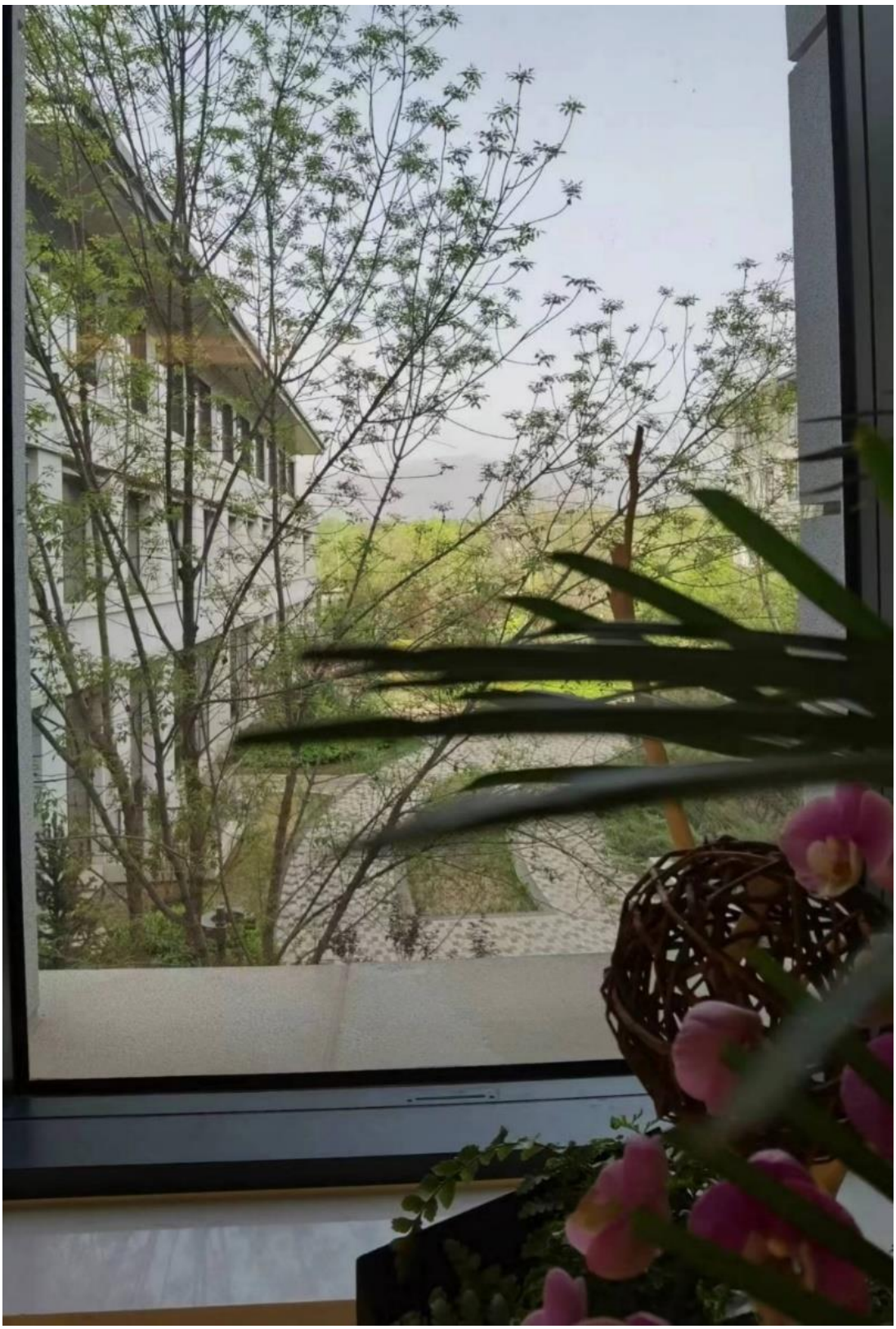
```
                cnt+=1
        else:
            cnt+=1
        # 增加嵌入的位置
        pos += n

        # 如果嵌入的位置超过了 DCT 系数的数量，说明嵌入完成，退出循环
        if pos >= len(cover_shuffled):
            break

    # 对 cover_shuffled 进行逆置乱，还原为 cover_nz
    cover_nz = cover_shuffled
    #unshuffle(cover_shuffled, key)



    # 将 cover_nz 还原为 cover_coef，并加上 DC 分量

    cover_coef[cover_coef != 0]= cover_nz
    cover_coef = np.insert(cover_coef, 0, cover_y[0, 0])

    # 将 cover_coef 还原为 cover_dct，并转换为整数类型
    cover_y =
cover_coef.reshape(cover_y.shape).astype(np.int32)



   # 将 cover_dct 还原为 cover_y，并进行 IDCT 变换
    #cover_y = idct(idct(cover_dct, axis=0, norm='ortho'),
axis=1, norm='ortho')
   # 将 cover_y 替换到 cover_ycbcr 的 Y 通道
    cover_ycbcr[:, :, 0] = cover_y
```

```
    # 将 cover_ycbcr 转换为 BGR 颜色空间，得到载密图像 stego
     stego = cv2.cvtColor(cover_ycbcr, cv2.COLOR_YCrCb2BGR)
    # 返回 stego 作为秘密信息
     return stego

# 定义 F5 提取函数
def F5_extract(stego, key, length):
    # stego 是一个 OpenCV 图像对象，表示载密图像
    # key 是一个整数，表示提取的密钥
    # length 是一个整数，表示要提取的秘密信息的长度
    # 返回一个字符串，表示提取出的秘密信息，只包含'0'和'1'

    # 将 stego 转换为 YCbCr 颜色空间
    stego_ycbcr = cv2.cvtColor(stego, cv2.COLOR_BGR2YCrCb)

    # 提取 Y 通道
    stego_y = stego_ycbcr[:, :, 0]

    # 对 Y 通道进行 DCT 变换
    #stego_dct = dct(dct(stego_y, axis=0, norm='ortho'),
axis=1, norm='ortho')
    # 将 DCT 系数转换为一维数组，并去掉 DC 分量
    stego_coef = stego_y.flatten()[1:]

    # 筛选出非零的 DCT 系数
    stego_nz = stego_coef[stego_coef != 0]

    # 对非零的 DCT 系数进行伪随机置乱
    stego_shuffled = stego_nz
```

```
    #shuffle(stego_nz, key)



    # 计算提取的参数 n 和 k，满足 n = 2**k - 1
    k = 4
    n = 2 ** k - 1

    # 初始化提取的位置和秘密信息
    pos = 0
    data = ''
    print(length)
    print(len(stego_shuffled))
    # 循环提取每一组秘密信息，直到达到指定的长度
    while len(data) < length:
        # 取出 n 个 DCT 系数的 LSB，转换为列表 a
        a = [int(bin(int(c))[-1]) for c in
stego_shuffled[pos:pos + n]]
        # 进行矩阵解码，得到 k 位秘密信息，转换为字符串 x
        x = bin(matrix_decode(a))[2:].zfill(k)
        print(int(x,2))
        # 将 x 添加到 data 中
        data += x
        # 增加提取的位置
        pos += n
        # 如果提取的位置超过了 DCT 系数的数量，说明提取失败，退出循环
        if pos >= len(stego_shuffled):
            print('error')
            break
    # 返回 data 作为秘密信息
    return data
```

```
# 测试 F5 嵌入和提取函数
# 读取载体图像 P.jpg

if __name__=="__main__":
    cover = cv2.imread('P.jpg')
    # 读取要嵌入的文本文件，内容为大文本中文
    with open('test1.txt', 'r', encoding='utf-8') as f:
        text = f.read()

    # 将文本文件转换为二进制字符串，只包含'0'和'1'
    data = bin(int(binascii.hexlify(text.encode('utf-8')),
16))[2:]
    print(len(data))
    # 设置嵌入和提取的密钥
    key = 123456

    # 调用 F5 嵌入函数，得到载密图像 stego
    stego= F5_embed(cover, data, key)

    # 保存载密图像 stego 为 SP.jpg
    cv2.imwrite('SP.bmp', stego)
    print('保存成功')
```

结果图片 SP:



附录 B:

从图片 SP 提取著作权信息使用的源代码 extract.py:

```
import cv2
import binascii
from main import F5_extract

# 调用 F5 提取函数，得到提取出的二进制字符串 extracted_data

stego=cv2.imread("SP.bmp",cv2.IMREAD_UNCHANGED)
key=123456
with open('test1.txt', 'r', encoding='utf-8') as f:
    text = f.read()
# 将文本文件转换为二进制字符串，只包含'0'和'1'
data = bin(int(binascii.hexlify(text.encode('utf-8')),
16))[2:]

extracted_data = F5_extract(stego, key, len(data))
print('/')
print(len(extracted_data))
#print('Text match:', data == extracted_data)
# 将提取出的二进制字符串转换为十六进制字符串，只包含 0-9 和 A-F
hex_data = hex(int(extracted_data, 2))[2:]
print('/')
# 将提取出的二进制字符串转换为文本文件，内容为大文本中文
extracted_text = binascii.unhexlify(hex_data).decode('utf-8')
# 保存提取出的文本文件 extracted_text 为 extract.txt
with open('extract.txt', 'w', encoding='utf-8') as f:
    f.write(extracted_text)
```