大连理工大学本科毕业设计（论文）

基于 SVM 的 LSB 信息隐藏算法研究与优化

The Research and Improvement of SVM-based LSB Steganography

学 院（系）：软件学院

专　　业：网络工程

学 生 姓 名：李欣宜

学　　号：201292241

指 导 教 师：孙伟峰　副教授

评 阅 教 师：郭成　　副教授

完 成 日 期：2016 年 6 月 7 日

大连理工大学

Dalian University of Technology

摘　　要

信息隐藏是指将秘密消息隐藏在一个文件中进行交流而不使其他人发觉消息的存在。隐写术作为主要的信息隐藏实现手段，需要在资源有限的前提下，解决容易被隐写分析方法攻击的安全问题，这种安全性方面的提升可以借助引入 SVM 完成。

本文实现了基本的 LSB 隐写方法和一些经典的针对 LSB 的隐写分析方法，并在实现的同时分析了其中的关系，提出一种基于 SVM 的优化方法：提取了 4 个能够较为全面地描述图像块、图像和秘密消息三者间关系的特征作为 SVM 训练和预测时使用的特征集，使用上述实现的隐写分析算法评估这些样本的安全性作为标签集。将 SVM 的训练作为隐写之前的预处理，在隐写时应用 SVM 预测以获得可以安全隐藏消息的位置。这种优化的方法对于缺少能够交换秘钥的安全信道的情况非常有效。

为了验证优化的实现效果，本研究使用图像数据库设计了一组实验对于嵌入率为 5%-50%的伪装图像进行分析，发现相较传统的顺序 LSB 替换，本文的方法确实可以在低嵌入率的情况下以 10%以上的概率提高隐写的安全性。

关键词：信息隐藏；LSB 嵌入；机器学习；支持向量机；

The Research and Improvement of SVM-based LSB Steganography

Abstract

Information hiding refers to the process that hides secret messages into files to communication without drawing attention from others. As a major method to hide information, steganography needs to solve the security problem against steganalysis methods under limited resources. The improvement of steganography on security can be completed by introducing SVM.

In this paper, basic LSB steganography and some classic steganalysis methods are implemented. Then, by analyzing the relationship of steganography and teganalysis, an improved method is proposed: four features are suggested to comprehensively describe the relationship among images, image blocks and secret messages, which are extracted as feature sets for the training and prediction of SVMs. Then the label sets are generated from the security evaluation results of present steganalysis methods mentioned above. The whole training process of SVMs can be considered as preprocessing of steganography. In the embedding process, the sender can find location to hide information safely by using SVMs to predict the security. The improved method can be extremely effective when lack of safe channel to exchange keys.

To verify the performance of this improvement, a group of experiments on image datasets are designed to analyze the stego images with their embedding rates ranging from 5% to 50%. In the experiments, it can be easily observed that comparing to traditional LSB replacement, the method in this paper does improve the steganography security by the probability of at least 10% under low embedding rate situations.

Key Words：Secret hiding；LSB embedding；Machine learning；Support Vector Machine

目　　录

1 绪论

1.1 隐写术的问题背景

隐写是指把一个文件、消息、图像或者视频隐藏到另一个文件、消息、图像或者视频的行为。与密码学不同的是，隐写术旨在隐藏消息或其他形式的信息本身的存在，不引起发送方和接收方以外的人的怀疑而完成信息的交流，而密码学则用于隐藏这些信息的内容，使得非发送方或接收方即使截获消息也无法得到所交流的信息的真实内容。隐写术的主要任务是使发生在公共信道上的秘密信息交流不被察觉，隐藏了秘密信息的图片或其他格式的载体与隐藏之前的原始数据在视觉上以及其他几个重要特征一致。

隐写术是一种秘密通信的艺术，这个术语最早在从几千年前开始，人类就着迷于密文书写，并出于多种原因和动机学习这种和研究这种技术[1]，早期密写通常是指密码学，而随着时代发展现在密写也涵括了隐写术，两者用不同的方式实现密写的目的。无论在哪个时代，隐写术广泛应用于各个领域。早期的隐写实践是使用不可见的墨水在信件中书写消息，显而易见地，军事和政治中，在不被敌方察觉的前提下向友方传递信息的能力十分关键。在数字时代，这种思想发展为在多媒体文件中隐藏其他数字消息。现代隐写术在 1985 年以后随着个人计算机的推广而问世，随着计算机科学技术和数学的发展以及深入研究而迅速发展并投入更广泛的应用，例如电子通信包含了传输层内的隐写码以传送如文档文件或图片文件等多媒体数据。

与密码学相似，隐写术的使用场景决定了它必须满足以下三个要求：

（1） 保密性：不容易被探查到隐藏消息的存在。

（2） 可获得性：不会出现由于修改数据的载体导致秘密消息的丢失，秘密消息可以被恢复。

（3） 完整性： 其他人无法伪造出错误信息。

隐写系统可以被视为加密系统的一个特例[2]，在这个系统中我们要求密文与明文对于其他人来说难以区分。值得注意的是，隐写加密方必须首先合成一个与秘密消息无关的无害的文件。载体合成具有很大的挑战性，高效合成载体的隐写机制很少。对于这个问题，使用通过修改实现隐写的方法很好地避免了这个挑战。这种系统使用一个现有的与秘密消息无关的文件，也就是载体，作为隐写系统的输入的一部分，接下来秘密消息转换为以让人难以觉察的载体的修改，得到与载体极其相似的结果，即伪装（stego）。理论上，载体合成是最自由且强大的隐写方法，因为它可以不受限制地适用于任意的秘密消息。但在实践中，无害文件的合成是一个非常复杂且低效的过程，所以大多数众所

周知的实用系统实用的都是修改载体的方法。当然除了合成和修改两种模式以外，载体选择[3]也是可行的方案，可以类比为传统密码学中使用的编码本，有多个载体或密文分别对应特定的秘密消息。然而，这种模式需要的可用载体数量太大，并不是在所有场景下都实用。所以我们在大多数情况下会选择通过修改进行隐写的模式，也就是说隐写的过程需要使用已经存在的文件作为原始的输入，而多媒体文件（如图像、音频和视频等）往往较大，包含了大规模的数据，可以找到足够的空间隐藏消息同时在不表达出可以被察觉的异常效果，是理想的载体。其中，数字图像的应用场景广泛，修改方便，且容易在互联网快速传播，成为了应用最多的载体。本文也将围绕图像隐写技术展开。

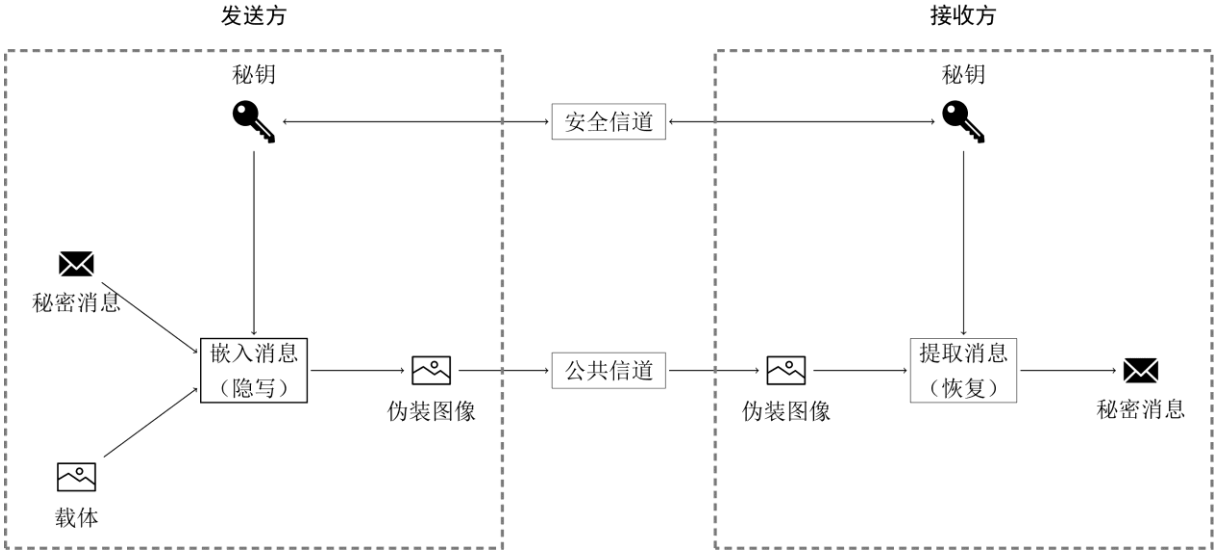

图 1.1　秘钥隐写系统

早期的隐写安全完全依赖于隐写算法，只将载体和秘密消息作为输入而不使用秘钥，被称为纯隐写系统，一旦隐写算法泄露则整个系统被破解。将秘钥引入作为输入的隐写系统则被称为秘钥隐写系统（如图 1.1 所示）。在密码学中，加密者和解密者共享秘钥，关于算法的知识对双方区分伪装和正常消息没有帮助，这个结论即 kerckhoffs 原则，在隐写系统中并不总是成立[4]。符合 kerckhoffs 原则的系统在使用一个秘钥的实例被攻破后，使用其他秘钥的实例的安全性不受影响，与其他系统相比这样的系统具有巨大的优势，因为生成不同的秘钥很容易，而重新设计一个算法却很困难。但事实上，基于 kerckhoffs 原则安全的隐写系统很难设计，同时大多数应用场景下，除了发送方和接收方外其他人对于他们的隐写系统通常一无所知，因此，不遵循 kerckhoffs 原则的系统也是可行的。

1.2 LSB 图像隐写技术

对于一个二进制整数来说，最低有效位（LSB）是最低的比特位（即第 0 位），决定了这个数是奇数还是偶数，这个比特位相比于其他位置的变化对于整个数值变化影响是最小的。

LSB 嵌入方法是一种经典的图像隐写算法。这种方法最早被用于像素图像，在像素图像中，每个像素都是代表该点颜色强度的整数。在灰度图像中，每个点的像素值表达了该点色彩介于黑白之间的程度，而在具有三个色彩信道的 RGB 图像中，每个像素点由三个独立个代表红、绿、蓝三种颜色强度的值合成(R,G,B)单元，这些像素值的取值范围通常是 8bit 的整数值，也就是$[0,255]$。在颜色强度上的微小改动被察觉的可能性很小，LSB 隐写算法正是利用了像素图像的这个特性，舍弃每个像素原来的最低有效位（LSB），并替换为需要隐藏的消息。接收者得到伪装完成的图像后可以通过模 2 操作提取新的 LSB 并将之还原为完整的消息。

除此之外，还有一种经典的基于 LSB 嵌入的改进，被称作 LSB 匹配，体现在随机化了每个样本的篡改。不同于经典 LSB 嵌入直接舍弃最低有效位的操作，LSB 匹配将观察该像素的最低有效位与需要嵌入的信息单元的关系，如果一致则保持最低有效位不变，否则将等概论地对该像素值± 1，具体来说对于偶数值像素$+1$而对于奇数值像素-1。消息的提取同样可以使用模 2 运算直接完成。

1.3 隐写分析

隐写分析是指对于使用隐写术隐藏的秘密消息的探查。隐写分析的主要任务是识别给定的文件中是否隐藏了可疑的秘密消息，最基础的隐写分析算法以可疑文件作为输入，而将二元分类的“是”和“否”作为输出结果。在此基础之上，有些隐写分析也加入了恢复隐藏信息的功能，但是作为隐写分析最根本的任务是探查是否存在隐藏的消息，一旦探查到隐藏消息的存在，甚至不需要恢复出具体的消息，就可以认为该隐写系统被攻破。

图像隐写分析方法根据手段不同主要分为四种：视觉隐写分析，结构隐写分析，统计隐写分析和学习隐写分析。

视觉隐写分析是最容易的分析手段，即观察图像是否存在视觉上的异常痕迹，最常见的实现方式即提取图像的 LSB 平面。通过修改图像的 LSB 而达到隐藏效果的隐写图像相比于自然图像在 LSB 平面体现出一定的异常，特别是在与高位比特平面的图像对比时，不自然的人工痕迹更为明显。

结构隐写分析则寻找在媒体表达或文件格式上泄露的痕迹。由于人为限制，某个版本的隐藏工具会对图像的大小有所要求，比如 Seek and Hide 4.1[5]只会使用320×480像

素的图像作为载体，那么我们可以根据这个特性重点探查320×480像素的图像。再如 JPEG 兼容压缩攻击[6]：JPEG 图像使用的是有损压缩，不是每个像素组都会成为可能的一个给定的 JPEG 解压实现的可能输出，因此这种攻击方法可以针对载体之前使用过 JPEG 压缩的像素图像的嵌入模式进行分析。

统计隐写分析借鉴统计学的方法探查隐写。这种方法通常需要一个统计模型来描述伪装和/或载体的概率分布，因此我们需要预先通过分析相关载体和隐写系统建立一个这样的数据模型。统计假设测验是最典型的统计方法，用以判定一个可疑的文件是可信的载体还是伪装。不仅如此，一些量化的分析使用了参数估计来估测嵌入消息的长度。对于载体的数据建模被证明是困难的，但是对于给定的隐写系统，却往往容易建立良好的关于伪装数据的模型，因此可以有针对地应用于探查某种隐写方法。

但在实际场景中很难单纯通过分析的方法建立一个统计模型，对此我们可以借助计算的方法，使用学习隐写分析，也就是使用穷举的思想得到经验模型。具体的解决方案可以使用模式识别、机器学习或人工智能等领域的方法。

1.4 相关工作

由于这些隐写分析算法的存在，隐写的安全性存在很大的问题，但是随着研究工作的进行，不少已有的工作中提出了需要考察图像本身和秘密消息的特性自适应地选择嵌入位置的方法，这些方法在 LSB 替换和 LSB 匹配等经典的基于 LSB 的图像隐藏方法中都有应用的实例。寻找适合隐藏信息的位置的方法主要是基于像素值差异的，比如在[7]中 Hempstalk 提出了根据一个像素和邻接的四个像素值的差值替换 LSB 的机制，使得隐写者可以在较锐利的边缘嵌入最多的秘密数据并成为了视觉上更不易被察觉的伪装，但是这种方法在隐写时只修改了 LSB 容易被 RS 隐写分析等分析方法探测出来，在安全性方面的表现较弱。在[8]中，Singh 等作者提出一种嵌入方法，先在载体的每个3×3像素大小且无重合的图像块上部署 Laplacian 探测器来寻找边缘，再根据一个阈值θ选出更锐利的边缘，然后再在这些边缘上的图像块的中心像素进行数据隐藏。基于像素值差（PVD）的机制（如[9–10]）是另一种寻找适应隐藏数据的图像块的机制，嵌入像素的数量由一个像素和它的邻接像素的差值决定。差值越大，能被嵌入的秘密比特越多。通常来说，基于 PVD 的方法可以提供更大的嵌入容量，但是这些基于 PVD 的方法在对抗统计隐写分析方面也不理想。在[11]中一种边缘自适应的 LSB 匹配重访问隐写方法被提出，这种方法根据嵌入率调整对边缘区域的定义和选择，考虑了秘密消息大小和图像的关系，寻找这种边缘区域的方法也还是基于像素之间的差值。

然而，这些隐写方法虽然解决了对抗隐写分析的问题，但需要额外依赖于秘钥的安全，即使在选择图像块时使用了一定的优化，但仍需要伪随机数生成器选择在图像块中的隐写位置，对于通信双方的通信资源要求较高。除此之外，还会在图像中嵌入一定量除了秘密消息以外的数据，给嵌入和提取操作带来了一定的难度。

1.5 研究目标

现有的基于 LSB 的隐写方法不计其数，如前文提到的使用秘钥和伪随机数生成器进行随机嵌入，可以在一定程度上抵抗隐写分析，但这样的模式对于通信双方拥有的通信资源有着很高的要求，不能保证在大部分情况下适用。机器学习的方法在隐写领域也被广泛使用，但主要应用的范围往往是侦测隐写技术是否被使用的隐写分析算法，实际上隐写技术本身也可以结合机器学习算法做出相应的优化，以此保证更高的安全性。

除了上面提到的四大类广泛应用在各种隐写技术的探测的隐写分析方法，针对 LSB 灰度图像隐写还存在着一些高效的特定隐写分析方法，为了改进基于 LSB 的灰度图像隐写技术，必须研究针对其提出的隐写分析方法。在研究这些分析方法的过程中，可以反推针对哪些特性做出一定的改进可以对抗这些方法。

本文在研究了三种针对 LSB 灰度图像隐写分析的方法后，总结了一些可以对抗分析方法隐藏信息的图像特点，在原来的顺序嵌入模式的基础上，提出了一种可行改进的方案。该方案需要提取这些可以评估伪装是否隐写安全的图像（块）的特征，在研究了下文的经典隐写分析方法后，评估模型中创新性地选择了以下一些特征，并将其作为安全性的评价指标，用于训练 SVM 分类器，并将训练完成的 SVM 分类器用于之后的隐写。

1.6 实现工具

本文在 2.1 节实现了基本的 LSB 隐写方法，在 2.2 节实现了三种针对 LSB 图像隐写分析方法，在第三章提出了基于 SVM 的 LSB 隐写优化方案并实现了这个方案。基本的 LSB 隐写的方法使用了 Python 和 MATLAB 编程实现，其中 Python 代码已经在 2.1 节演示基础 LSB 隐写方法时给出，MATLAB 代码在附录中给出。实现优化的过程中提取样本几个特征以及 SVM 的实现也是也使用了 MATLAB 完成的，而实验数据的收集处理和对比作图则使用了 Python 实现。本节将重点介绍这两种工具和实现过程中用到的封包。

1.6.1 Python 简介

Python 是一种高级的、通用的动态编程语言，这种计算机语言面向对象，使用动态类型系统并提供内存分配自动管理机制，使用缩进来表示结构。因此，相比 C++或者 Java，一般来说，实现相同的功能 Python 需要的代码量更少。

Python 有着简单明确的语言特点，同时拥有成熟的标准库，由于其开源的特性，开发者也为 Python 贡献了许多优秀的封包，也因此 Python 作为良好的编程语言被广泛应用于各个领域的开发。

Python 的主流发行版有 Python2.7，Python3.4 及以上，这些发行版在一些语言特性的解释方面略有差别，但对于我们的问题来说编程效率都十分出色。Python 因为拥有为科学计算和图像处理设计的模块所以非常适合用于文中隐写的实现以及实验数据的处理，接下来将一一介绍这些模块。

（1） NumPy 在下文中将被简写为 np 进行使用，该模块为 Python 提供了多维数组和矩阵处理的扩展支持，其核心结构为数据类型 ndarray，使用 Numpy 可以进行高效的大规模的矩阵运算。

（2） SciPy 作为 Python 的算法库和数学工具包，提供了优化、线性代数、积分、微分方程、特殊函数等解决方案，配合 NumPy 则可以以极少量的代码完成绝大多数矩阵的操作运算。

（3） PIL Python 的图像处理库，提供基本的图像处理操作方法，包括本文中使用的图像读取和写入，图像类型转换，以及简单的裁剪、旋转等操作。

（4） Matplotlib Python 的图形框架，可以实现高质量的多种形式的图像绘制，如折线图、散点图和直方图，在 IPython notebook 平台下可以进行交互作图。

另外，现在已经有 Stepic 库实现了 Python 中图像隐写和恢复的功能，基本的隐写和恢复都已被封装为可以调用的函数。然而 Stepic 库实现的图像隐写模块非常简单且无扩展性，无法提供高质量和安全的隐写操作，有着很大的局限，因此只能作为一种实现参考，文中所有的隐写和数据处理方案主要使用的还是上述的四个模块。

1.6.2 MATLAB 简介

MATLAB 是美国 MathWork 公司开发的商业数学软件，软件的名称为矩阵实验室（Matrix Laboratory）的缩写，核心结构为基于矩阵的计算，将大部分数据视为各种类型的矩阵进行运算处理。MATLAB 本身提供了一个计算功能强大的交互式环境，可以实现算法开发，数据可视化，数据分析，数值计算等功能，这些功能均基于其强大的矩阵运算能力，当处理大规模的矩阵运算时，MATLAB 表现出其他工具难以匹敌的效率。其

内置的数学函数基本覆盖科学计算的方方面面，为算法的实现提供了便利。除了交互环境以外，MATLAB 还定义了特定的编程语言，即 m 语言。m 语言与 Python 相似，都具有动态类型系统，结合 MATLAB 内置和第三方扩展的各类工具箱，可以编写少量代码实现强大的数据处理功能。

MATLAB 中的大多数数值数据以矩阵的形式存放，所有变量均可保存为.mat 格式的数据文件，使用 MATLAB 读取和处理这种类型的文件十分方便。m 语言本身提供了很多用以读写文件的函数，尤其对于图像文件的读写、绘制存在许多性能优良的实现方法，因此被选为本文中优化方法的编程语言。由于 SciPy 库的存在，Python 得以高效读写.mat 文件，并将其中的矩阵处理为 ndarray 类型，两种工具间实现了无缝对接。

为了提高 MATLAB 中 SVM 的训练和预测效率，本文借助了 MATLAB 的 LIBSVM 扩展工具箱。LIBSVM 是一个提供了简单实用且快速有效的 SVM 模式识别和回归的工具箱，拥有 C、Java、MATLAB、C#、Ruby、Python、R、Perl、Common LISP、Labview 等数十种语言版本的实现，其中 MATLAB 平台上的实现最为成熟，可以通过运行 LIBSVM 安装包中的 MAKE 文件在 MATLAB 上安装 LIBSVM，安装结束后即可调用封装好的 SVM 相关函数进行使用，如果对默认的训练方法不满意可以调整函数中的相关参数，也就是使用不同类型的 SVM 进行训练。LIBSVM 基本实现了现有的关于 SVM 算法的大部分优化方法，因此对于大规模数据集的训练和预测都非常迅速。

除此之外，由于实验中使用到了很多轮迭代获取数据，虽然 MATLAB 本身对于大规模矩阵的处理能力出众，但为了进一步优化效率，也使用了 MATLAB 的并行计算池，并使用 parfor 语句改写原程序的 for 循环语言，在计算效率上获得了巨大的提升。

1.7 本文结构

本文将按照如下结构组织内容：

第一章为绪论部分，介绍隐写技术的使用背景和发展近况，以及与隐写术相对的隐写分析技术的存在，提出本文需要解决的问题并概述研究和优化的方向。

第二章给出了基本 LSB 替换在灰度图像上的实现隐写的方法，并有针对性地对这种原始的方法使用三种进行伪装图像隐写分析，说明顺序位置的 LSB 在安全性上存在一定的局限，并指出了优化后衡量优化效果的指标，即使用这三种方法得当的评估结果。

第三章开始提出本文的优化场景假设，即无法使用秘钥保证安全性，在此基础上并引入 SVM 挑选适合隐藏数据的位置的方法。其中着重介绍了 SVM 的用法和在图像中选用来评价安全性的四个特征。

第四章针对上面提出的方法使用图像数据库进行实验，以三种隐写方法的评估结果验证位置选择的准确率，由于 SVM 训练为可以在隐写前任意时间完成的数据预处理过程，而隐写时 SVM 预测的过程中时间代价也可以忽略，这里的预测准确率可以直接用于表示方案中隐写性能在安全性上的提升。

结论部分将给出以上工作的总结以及对未来工作的展望。

1.8 本章小结

本章交代了问题的背景和本文中所做的工作的概述。1.1 节是隐写术的理论基础和应用背景，并给出了 1.2-1.3 节中 LSB 隐写和隐写分析的定义。在 1.4 节中，本文研究了一些现有的 LSB 隐写的优化工作，并从中受到了一定的启发，而这些启发驱动了 1.5 节的新优化方向设计，并给出了概述。在 1.6 节中，两种编程语言作为本文的实现工具被引入介绍，并交代了这些工具的使用方法。全文各章内容的概述在 1.7 节给出。

2 LSB 图像隐写和隐写分析的方法及实现

本章中的隐写和隐写分析方法均在 Windows10 系统上完成，使用 MATLAB 和 Python 编程实现，图像示例使用经典图像测试用例的 LENA 图或 UCID 图像数据集中的图像作为演示，实现方式为给出可以描述实验结果的图像，并在本章及附录 A 和附录 B 中提供封装为函数的源代码。

2.1 LSB 图像隐写实现

在灰度图像上，使用 LSB 位隐藏信息的过程非常简单。接下来以256 ×256 像素大小的 8bit 灰度 LENA 图为载体隐藏文本消息演示隐写实现的过程（这里仅给出一种 Python 实现的方法，其他语言和方法的实现可以参考附录 A）并分析其中可能存在的问题。

2.1.1 嵌入过程

首先，对于载体图像中的每个像素，我们对其 LSB 位进行置 0 操作，也就是去掉原图像的 LSB 平面，结果如图 2.1 所示

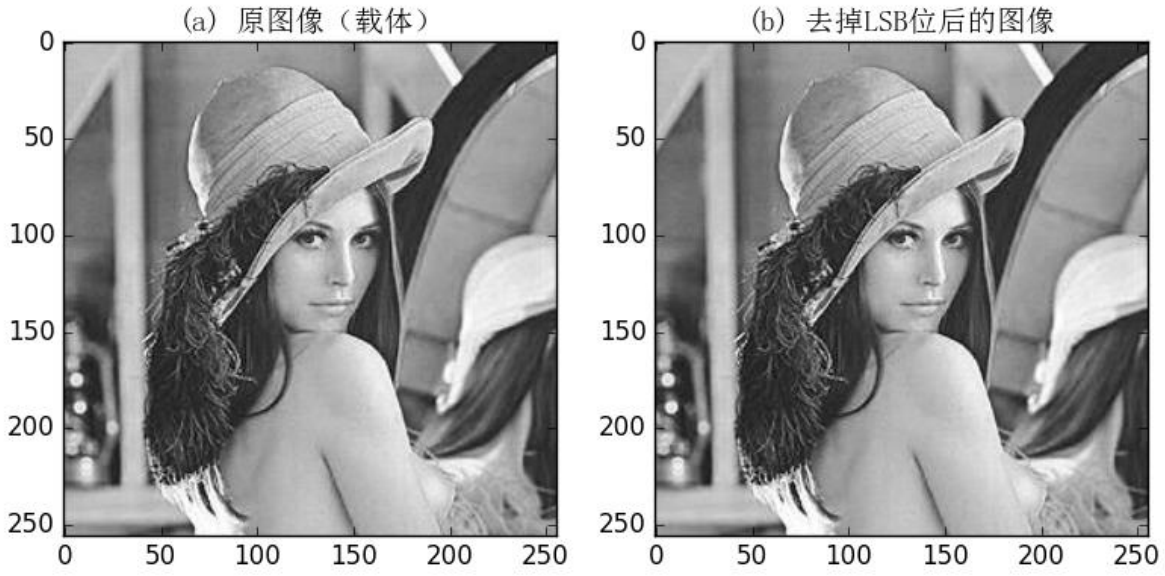

图 2.1 去掉 LSB 位前后图像的变化

我们发现即使截去整个 LSB 平面，在视觉上也很难发现原图和处理后的图像差异，因此可以选择把文本消息。而文本消息可以通过 ASCII 码转换为二进制的形式作为新的 LSB 位。

最基础的基于 LSB 的隐写系统从图像的左上角开始逐位嵌入秘密消息，推进的顺序是由发送者和编程语言决定的最自然的方向。为了方便接受者确定隐藏的消息在何处结束，我们可以将消息的大小作为头部隐藏在前n像素中，这里的n是一个双方已经约定好的整数。在实现过程中一般考虑n位二进制整数恰好能表示图像嵌入消息的最大长度。

实现嵌入过程的 Python 代码如下所示：

```
def convert_message_to_bit(string_text):
    '''将秘密消息 string_text 转换为二进制串'''
    bit_string=[]
    for char in string_text:
        bit_string.extend([int(d) for d in bin(ord(char))[2:].zfill(8)])
    return np.array(bit_string)

def hide_side_information(carrier,length,pro_text_size):
    '''在图像 carrier 的前 pro_text_size 隐藏一些辅助信息（在这里就是秘密消息的长度 pro_text_size） '''
    bit=bin(length)[2:].zfill(pro_text_size)
    length_data=np.array([int(d) for d in bit ])
    side_info=carrier[:pro_text_size]+length_data
    carrier[:pro_text_size]=side_info
return carrier

def hide(cover_pic,secret_text,pro_text_size):
    '''在给定的图像（载体 cover_pic）中嵌入秘密消息 secret_text ，辅助消息长度 pro_text_size '''
    length=len(secret_text)
    cover=np.array(cover_pic)
    size=cover.shape
    cover_data=cover.flatten()
    carrier_data=cover_data & 0b11111110 # 去掉 LSB 平面的载体图像
l=hide_side_information(carrier_data,length,pro_text_size)
    secret_bytes=convert_message_to_bit(secret_text)
secretbytes_length=length*8
carrier_data[pro_text_size:pro_text_size+secretbytes_length]=carrier_data[pro_text_size:pro_text_size+secretbytes_length]+secret_bytes
new_data=carrier_data.reshape(size)
```

```
new_iamge=Image.fromarray(new_data)
return new_iamge
```

最后，隐写者可以获得一张隐藏了秘密消息的伪装图像，并使用公共信道将这种图像作为需要传送的秘密消息给接收方。

2.1.2 提取过程

而接受到伪装图像的接收者已知了n的长度，将先提取前n像素的 LSB 位作为长度l的二进表达，得到秘密消息的长度l，再提取接下来$8l$位的像素的 LSB 位恢复为秘密消息。从伪装图像中获取秘密消息的 Python 代码如下：

```
def binary_array_to_int(arr):
    '''convert a 0-1 array into a decimal integer'''
    bit_string=''.join(arr.astype(np.str))
    return int(bit_string,2)

def reveal(secret_iamge,pro_text_size):
    '''extract secret message from a given picture(stego) return a secret string'''
    im=np.array(secret_iamge).flatten()
    lsb_plane=im & 0b00000001
    length=binary_array_to_int(lsb_plane[0:pro_text_size])
    secret_bytes=lsb_plane[pro_text_size:pro_text_size+length*8].reshape((length,8))
    secret_bit=[binary_array_to_int(x) for x in secret_bytes]
    return ''.join(chr(c) for c in secret_bit)
```

2.1.3 传统方法的缺陷

2.1.1-2.1.2 小节实现的基于 LSB 替换的隐写过程在操作上十分容易，而且通过肉眼观察伪装图像很难察觉到隐藏消息的存在。对于低嵌入率的情况来说直接截取整个 LSB 平面是十分浪费的，所以实际上会保留未嵌入消息的像素的 LSB 平面不变，仅仅修改需要嵌入消息的位置的像素值。

但是当 LSB 平面被单独提取出来的时候，我们很容易发现伪装图像中存在的异常，这也是一种视觉攻击的方法。图 2.2 给出嵌入《葛底斯堡演说》全文前后图像的对比。仅观察整个灰度图像确实难以发现两者的差别，因为在嵌入前后像素的高位没有变化，LSB 位的变化对图像整体的影响很小，但提取出两张图像的 LSB 平面进行对比时，可以明显察觉到伪装图像左上方的异常纹理。因此，我们可以认为这种图像隐写方法非常不安全。

出现这种现象是因为顺序 LSB 嵌入方法使得隐藏的消息总是在同样的位置。显然，作为最经典的图像隐写算法，LSB 嵌入模式许多的变体，针对这个问题，如果随机选择用于嵌入消息的像素点的位置则可以让伪装变得更加隐蔽。在隐写系统中引入伪随机数生成器（PRNG），依据其产生的序列选择嵌入消息的像素位置，可以使伪装消息的分布更加随机。但是值得注意的是这个伪随机数生成器的种子作为系统的秘钥必须在之前就由双方完成交换。

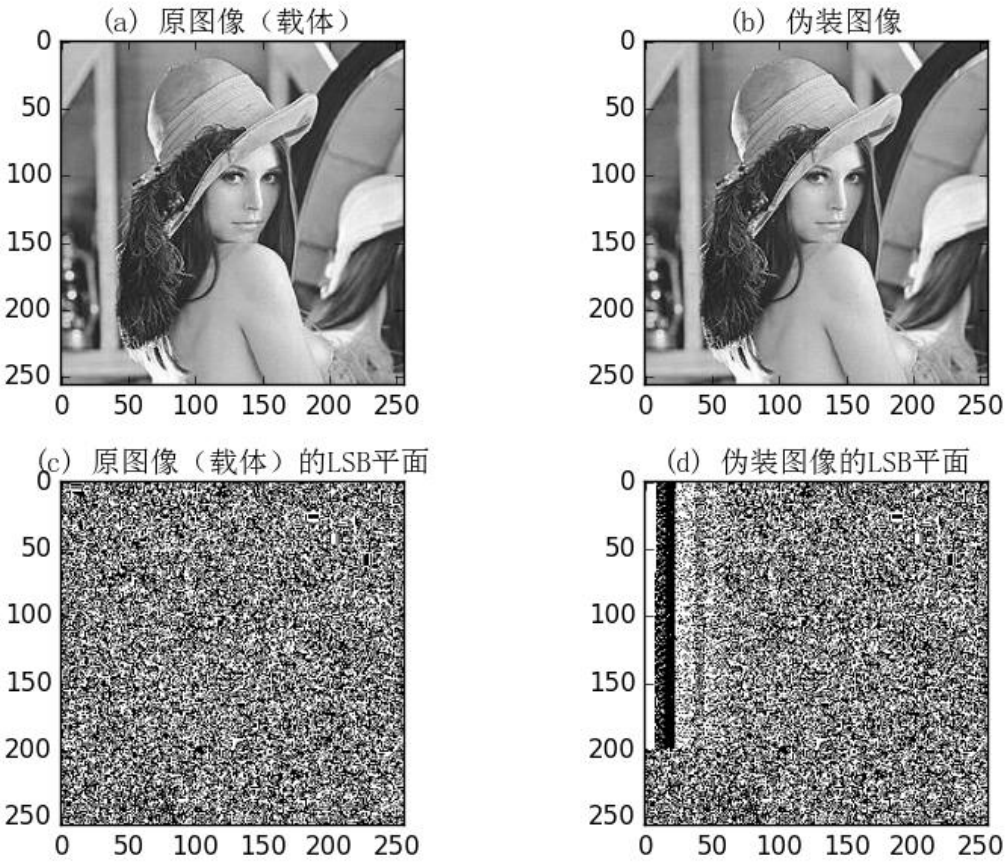

图 2.2　嵌入秘密消息前后的图像对比

然而，交换秘钥的安全信道的条件往往不太容易满足，无法保证隐藏数据可以以理想的方式随机分布于图像中。同时，LSB 隐写作为最为经典的隐写方法，除了上文中定性的视觉攻击，还存在许多有效的隐写分析方法，这些方法的存在使得在无秘钥情况下 LSB 隐写安全存在很大的威胁。

2.2 针对 LSB 图像的隐写分析方法的原理和实现

为了了解在哪些方面可以提升 LSB 灰度图像隐写的安全性，本节研究了三种特定的隐写分析算法原理，将其作为优化的评估标准，并实现了这些隐写方法（见附录）。

针对基于 LSB 的图像隐写分析的方法主要有χ^2测验[12]，样本对分析[13]和 RS 隐写分析[14]，他们的主要特性如表 2.1 所示。

表 2.1 三种针对基于 LSB 的隐写分析方法的比较

方法	原则	特征	作为侦测分类器
χ^2测验	将图像进行分片并对每个分片使用χ^2测试	χ^2系数	基于阈值
RS 隐写分析	使用翻转和掩码来识别 R,S,U 组并画出 RS 图进行估计	在不同的掩码中 R,S,U 组出现的频率	基于阈值
样本对分析	计算样本对的频率，解方程估测消息长度	每个特定样本对出现的频率	基于阈值

2.2.1 χ^2测验

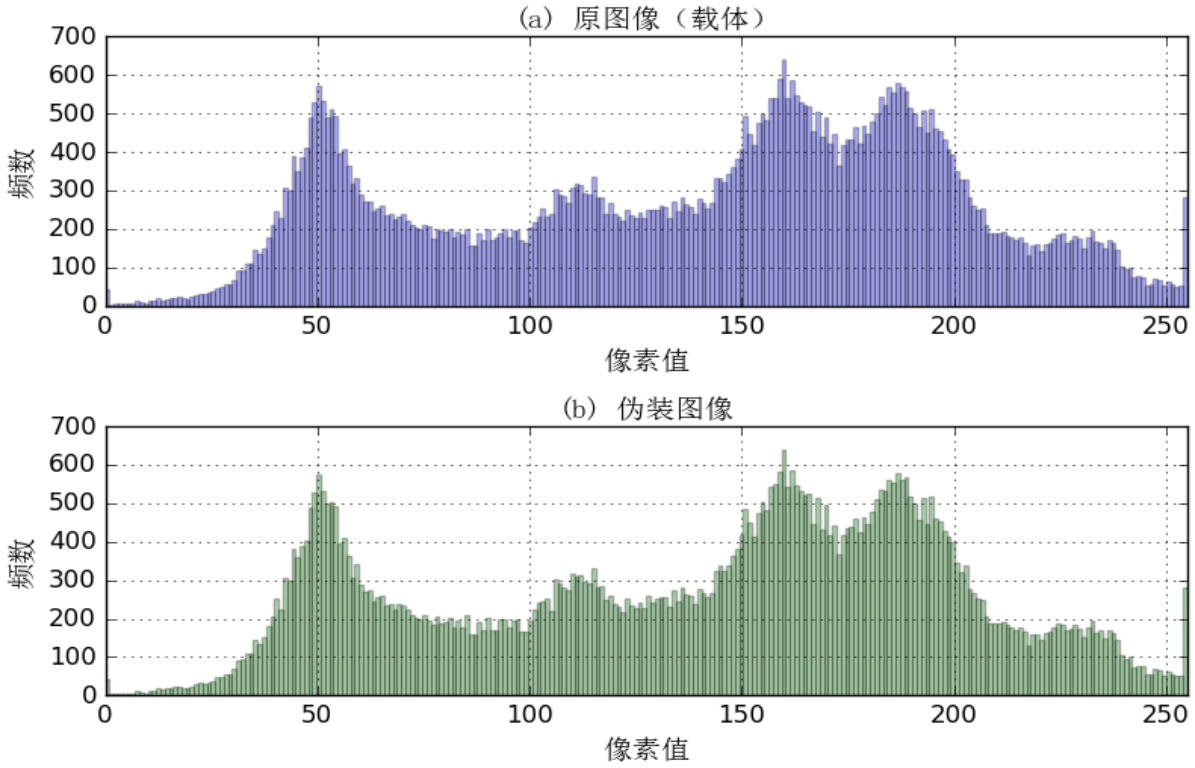

图 2.3 伪装图像的直方图

χ^2测验，也就是值对（PoV）攻击，是统计隐写分析的先驱。χ^2攻击的设计基于图像直方图来检测 LSB 嵌入。

为了理解值对，我们可以先观察载体和使用 LSB 满容量嵌入得到的伪装的直方图，即图 2.3。直方图统计了每个灰度值出现的次数，可以从图中看出伪装图像的直方图中有更多的成对出现的大致等高的条方。

回顾 LSB 嵌入的过程，载体图像中一个值为$2i$的像素可能在依然保留值$2i$或者改变为$2i+1$，同理，一个值为$2i+1$的像素也可能保持不变或者变为$2i$，因此对于每一个$i=1\ldots127$的$(2i,2i+1)$都形成了一个值对，LSB 嵌入可以改变一个值对内的两个像素值但无法将一个像素值从一个值对改变到另一个值对。如果嵌入的消息是一个随机消息或者加密过的消息，也就是需要嵌入的位近似于 0 和 1 的均匀分布，那么对于伪装图像的每一个i来说，$2i$和$2i+1$也是近似于均匀分布的。

基于上述特性，我们可以使用统计假设测验来检查可疑的图像是否为包含嵌入消息的伪装图像。这里选择使用χ^2测验来检验像素值$2i$和$2i+1$出现的频率是否符合均匀分布。令h_m表示颜色强度值m出现的次数，在（满容量）伪装图像中我们期望一个值对中的两个值出现的频率符合均匀分布，因此对所有$l\in[0,127]$有$E(h_{2l})=(h_{2l}+h_{2l+1})/2$，根据标准$\chi^2$测试[15]有

$$S_{PoV}=\sum_{l=0}^{127}\frac{\left[h_{2l}-\frac{1}{2}\left(h_{2l}+h_{2l+1}\right)\right]^2}{\frac{1}{2}\left(h_{2l}+h_{2l+1}\right)} \tag{2.1}$$

查询χ^2分布表我们可以计算对应的p值，用以度量图像为伪装图像的概率，根据p值大小决定该图像是否包含隐藏消息。

这样的χ^2测验一开始只适用于探查是否存在连续的 LSB 替换，对于低嵌入率的情况效果不佳，且对于使用了伪随机数决定嵌入像素的隐写系统无效。随着现有研究的发展，χ^2测验的思想被推广，可以通过多次渐进迭代、从特定位置开始探查、增大样本大小等方法[16]改进用于随机位置的 LSB 嵌入以及其他 LSB 衍生系统的探查。

2.2.2 样本对分析

样本对分析（SPA）方法跟踪 LSB 嵌入前后的样本对构成的多个集合，并分析多个集合间的关系计算嵌入消息的长度。

假设图像用连续的样本$s_1,s_2,\ldots,s_N$表示，其中N为划分的样本总数，一个样本对(s_i,s_j)，$1\le i,j\le N$，集合P为图像中所有样本构成的集合，我们定义D_n是包含类似于$(u,u+n)$或$(u+n,u)$的P的子集，其中$0\le n\le255$。对于$0\le m\le255$，定义C_m为前 7

比特的差值m的样本对。我们再定义$X_{2m+1}=D_{2m+1}\cap C_{m+1}$以及$Y_{2m+1}=D_{2m+1}\cap C_m$，对于$0\le m\le 126$且$X_{255}=\emptyset$有$Y_{255}=X_{255}$。如果$P$中的样本对离散均匀分布，那么对于任意的满足$0\le m\le 126$的$m$都有

$$E\left|X_{2m+1}\right|=E\left|Y_{2m+1}\right| \tag{2.2}$$

这也是 SPA 方法的关键理论基础。

显然集合X_{2m+1}中的样本对都形如$(2k-2m-1,2k)$或$(2k,2k-2m-1)$，Y_{2m+1}中的样本对则形如$(2k-2m,2k+1)$或$(2k+1,2k-2m)$，如果我们考虑 LSB 嵌入中样本对的翻转，我们有四种修改模式：00,01,10,11，其中 1 表示样本对中 LSB 发生改变的样本，而 0 表示保持完好的样本，所以对于每一个$0\le m\le 126$，多元子集C_m都被划分为X_{2m-1}，D_{2m}和Y_{2m+1}，那么显而易见，这些C_m在嵌入后都是相近的，而X_{2m-1}，D_{2m}和Y_{2m+1}却不是。所以我们继续把D_{2m}分割为X_{2m}和Y_{2m}，这时X_{2m}包含形如值为$(2k-2m,2k)$或$(2k+1,2k-2m+1)$的样本对，Y_{2m}包含形如包含形如值为$(2k-2m+1,2k+1)$或$(2k,2k-2m)$的样本对，$1\le m\le 126$的多重集合C_m通过上述操作可以被分为X_{2m-1}，X_{2m}，Y_{2m}和Y_{2m+1}，成为C_m的变多重集合，那么在 LSB 嵌入后，我们可以获得一个描述变多重集合间转换的有限状态机。

根据[13]中提到的方法我们可以用如下的方程计算嵌入消息的长度p:

$$\frac{p^2}{4}\left(2\left|C_0\right|-\left|C_{j+1}\right|\right)-\frac{p}{2}\left[2\left|D'_0\right|-\left|D'_{2j+2}\right|+2\sum_{m=0}^{j}\left(\left|Y'_{2m+1}\right|-\left|X'_{2m+1}\right|\right)\right]+\sum_{m=0}^{j}\left(\left|Y'_{2m+1}\right|-\left|X'_{2m+1}\right|\right)=0 \tag{2.3}$$

我们令$j=126$即假设只使用最低一位进行嵌入，则可以估计嵌入消息长度的最小值。

2.2.3 RS 隐写分析

RS 方法挖掘伪装图像空间的相关性。相比于用于嵌入操作的原始载体，LSB 嵌入后得到的伪装图像的空间相关性往往有所下降，我们可以利用这个特点探查隐写的存在。RS 方法的主要思想如下：先将图像划分为大小相等的小图像块（群)，对得到的每个小图像块都分别进行非负翻转和非正翻转操作，再对比统计经过这些操作以后小图像块的空间相关性的变化，Fridrich 等人[9]经过统计分析认为，LSB 嵌入前后空间相关性增加的小图像块数量和相关性减小的小图像块数量各自呈现一定的关系，可以根据这样的关系判断图像中是否有经过 LSB 嵌入的隐藏消息。在本小节中将使用如图 2.4 中给出的大小为384×512的灰度图像作为载体进行实验，因为它有着作为典型随机 LSB 平面以及其他自然图像的良好特性，方便我们观察相应的实验结果。

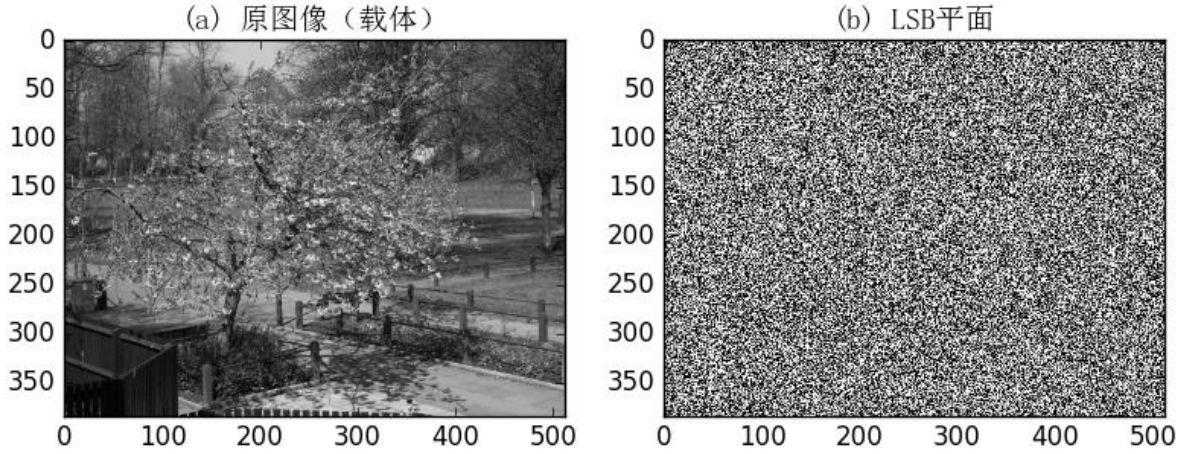

图 2.4 性质良好的自然图像

我们规定三种翻转操作：正翻转F_1，负翻转F_{-1}和零翻转F_0。对于任意的$0 \le i \le 126$，正翻转F_1是像素值在$2i$和$2i+1$间的转换，如$0 \leftrightarrow 1,2 \leftrightarrow 3,\ldots,254 \leftrightarrow 255$，是使用 LSB 嵌入秘密消息过程中发生的翻转操作。负翻转F_{-1}则是像素值在$2i$和$2i-1$间的转换，如$1 \leftrightarrow 2,3 \leftrightarrow 4,\ldots,253 \leftrightarrow 254$。零翻转$F_0$则保持原像素值不变。原始的 LSB 嵌入过程可以描述为如下过程：当嵌入的位与像素的 LSB 位相同时对该像素进行零翻转F_0，否则使用正翻转F_1。

我们定义作用在群$G=(x_1,x_2,\cdots,x_n)$的函数f为平滑度函数，表示为$f(x_1,\ldots x_n)=\sum_{i=1}^{n-1}|x_i-x_{i+1}|$，图像的平滑度函数函数是每个像素与其邻接像素的绝对差值的和，反映了图像的空间相关性。平滑函数的值越大，图像的混乱程度越大，图像空间的相关性越小。

我们对于每个组都使用掩码为M的翻转操作，得到的翻转结果为$F(G)=\left(F_{M(1)}(x_1),F_{M(2)}(x_2),\ldots F_{M(n)}(x_n)\right)$其中$M(i)\in\{-1,0,1\}$。根据翻转前后像素组$f$值大小的变化我们可以定义三类像素组$R$,$S$和$U$：如果$f\left(F(G)\right)>f(G)$则$G$为正则组，记为$G\in R$；如果$f\left(F(G)\right)<f(G)$则$G$为奇异组，记为$G\in S$；除此以外，若$f\left(F(G)\right)=f(G)$则$G$为不变组，记为$G\in U$。掩码$M\in\{0,1\}$表示的非负翻转操作下的正则组的相对数量记为$R_M$，奇异组的相对数量记作$S_M$，同理对于掩码为$-M\in\{0,-1\}$的情况，即非正翻转操作得到的正则则的相对数量记为$R_{-M}$，奇异组的相对数量表示为$S_{-M}$。

Fridrich 等人[6]提出零假设是对于典型的未经过 LSB 嵌入的载体图像满足R_M和R_{-M}的大小近似相等，同样S_M和S_{-M}也是近似相等的关系，也就是经过非负翻转后空间相关性减小的小图像块和经过非正翻转后空间相关性减小的小图像块数量大致相等，并且对

于相关性增大的小图像块这种类似的关系也成立。同时，对于未经 LSB 嵌入的图像还有$R_M > S_M$和$R_{-M} > S_{-M}$，这意味着对于自然图像，非零翻转必然使图像空间的像素相关性呈现下降趋势。对于经过 LSB 嵌入消息的伪装图像，应用非负翻转后的空间相关性相较于应用非正翻转有很大提升，所以有$R_M < R_{-M}$和$S_M > S_{-M}$。对图 2.4 中的示例图像使用以5×5 作为分组大小进行翻转操作，得到各组数量随着嵌入率变化如图 2.5 所示，在各分组数量的变化趋势符合以上分析结果。

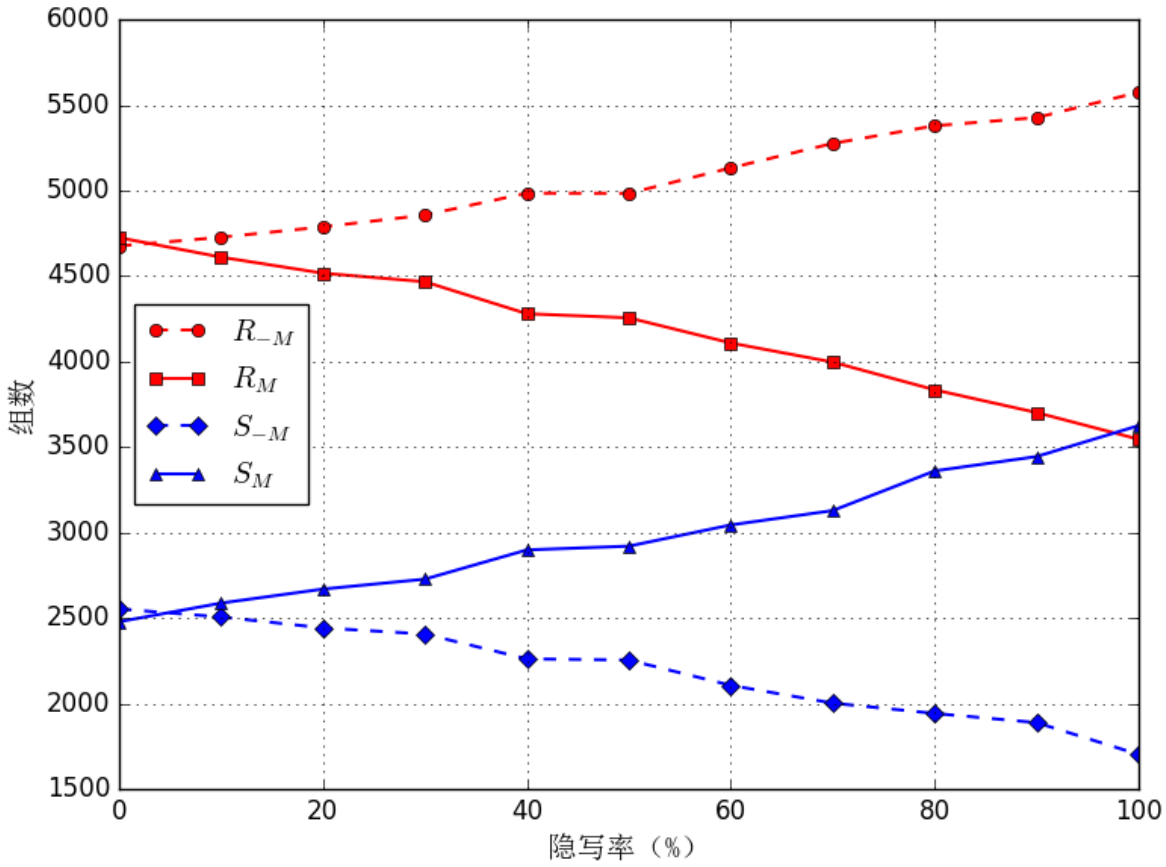

图 2.5　图像中的R_M，S_M，R_{-M}和S_{-M}的数量随着嵌入率的变化关系

由于 LSB 嵌入本质上应用的也是一种非负翻转的实例，所以在统计意义上，随着嵌入率的上升，R_M和S_M的差值会有所减小。然而在对伪装图片进行非正翻转的过程中，有些像素可能经历过正负两次翻转，致使其与原始值偏离得更远，因此R_{-M}和S_{-M}不会随着嵌入率的上升而发生显著变化。根据R_M和S_M的差值可以量化估计嵌入消息的长度，但是由于计算量较大这里不作介绍，RS 隐写分析的主要实现是捕获两类翻转后图像空间相关性的非对称性变化的特点发现图像中 LSB 隐写痕迹。

2.3 关于对抗隐写分析方法的思考

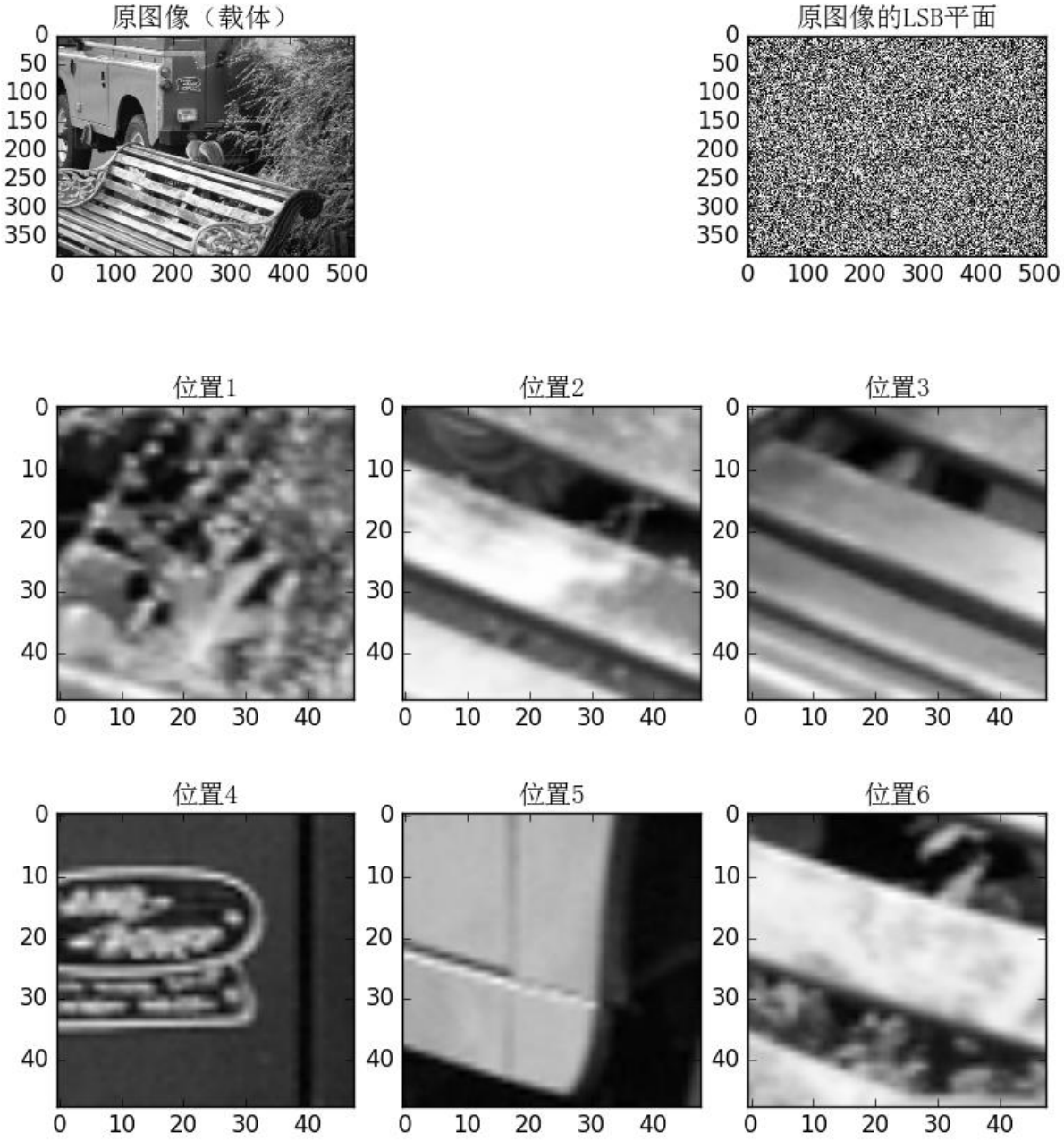

图 2.6 选择载体图像上不同的位置

一般来说，变换隐藏信息的位置可以在一定程度上缓解这些攻击带来的威胁。图 2.6 中给出一个384×512大小的载体图像，图像各个位置表达了不相同的事物，如果要嵌入一个 384B 长度的消息，则可以选择一个48×48大小的图像块嵌入其 LSB 平面。图

2.5 给出了 6 个不同位置的48×48大小图像块，隐藏在这 6 个位置产生的效果不同。而如何评价这 6 个位置是否适合隐藏，则需要提取这 6 个图像块的特征进行分析最后判断哪个适合隐藏而哪个不适合，这种分析的过程与 SVM 的预测过程十分相似。

2.4 本章小结

本章对于 LSB 图像隐写和针对 LSB 图像的隐写分析进行了研究，在 2.1 节用 Python 实现了基本的 LSB 图像隐写方法后，在 2.2 节使用 MATLAB 实现了三种隐写分析的方法对于 2.2 节的隐写方法进行安全评估，发现基本的 LSB 隐写方法确实在安全性上存在很大的缺陷，容易被现有的隐写分析方法攻破。

接下来我们在 2.3 节提出对于隐写者来说，需要知道的信息仅仅是选择的图像块是否适合隐藏秘密消息这个二元分类问题，因此，在 LSB 图像隐写的基础上引入 SVM 预测安全性可以给隐写者提供安全上的优化，而需要选择哪些特征来量化图像的性质，需要哪种类型的 SVM 分类器，以及怎样让收到图像的接收方了解秘密消息被隐藏在什么位置，这些优化方案中的具体设置细节则将在第三章详细论述。

3 基于 SVM 的 LSB 隐写优化原理

通过前面的介绍，我们知道，如果通信双方在嵌入和提取之前已经完成了秘钥交换，那么可以引入伪随机数生成器，以秘钥作为种子确定随机的隐藏位置序列，可以提高的 LSB 嵌入的安全性。然而，需要注意的是这里的秘钥必须通过安全信道提前完全交换，实际应用中是并不是一定有条件实现这个要求的，因此在没有额外的资源的情况下，保证隐藏的位置完全随机且接收方可以准确提取是不可能的。

本文所做的工作是在与原始的 LSB 给定的资源条件下实现一个更安全高质量的隐写系统，如上文所述，我们假设在我们的应用场景中无法使用秘钥等开销较大的信息作为系统的基础，但允许双发约定前l_1位作为隐藏信息长度的嵌入位置以及紧接着的l_2位长度的像素嵌入一个坐标，这种假设基于一些无法完成秘钥交换和更新的情况，但l_1和l_2的大小决定了其容易通过约定俗成的方式让双方获得。同时这样的限制让我们无法随机地、离散地选择嵌入位置，只能按照像素的自然顺序嵌入消息，但相比于原始的 LSB 嵌入方法，新的方案可以选择从图像的一个特定位置开始顺序隐写，相当于从原图像中分割出了一个最小图像块进行顺序嵌入。

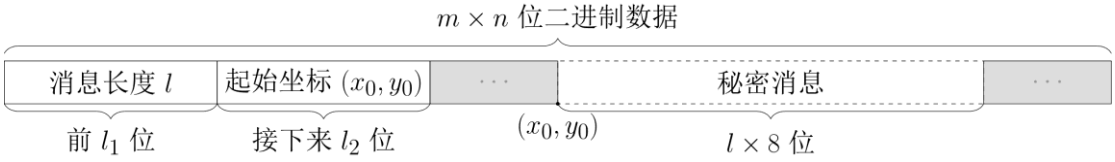

图 3.1 图像 LSB 平面各个位置的分配方案

无论使用哪种隐写方案，当嵌入率上升到一定程度时，都无可避免地会被隐写分析方法探查出异常的存在；同理，当嵌入率接近 0 时，即使使用最简单的隐写算法也很难被探查出来。本文仅讨论像素值为 8 比特的灰度图像（实际对于 RGB 图像也同理），假设图像大小为$m \times n$像素，只在图像中嵌入$l_1 + l_2$位的辅助信息，包括秘密消息的长度和起始隐写像素的坐标，只要满足$l_1 + l_2 \ll m \times n$，已有的通用 LSB 隐写分析算法对于探查这些消息的存在没有帮助，所以在这里我们可以认为在图像的前$l_1 + l_2$位嵌入这些辅助消息是安全的。所以，我们对图像各个像素的 LSB 的分配方法如图 3.1 所示，虚线表示的用于嵌入消息的图像块在空间上不一定连续，仅仅在理解上作为一个连续的像素集进行表示，灰色的位置表示没有用于嵌入消息的像素，在隐写的过程中不对其进行任何操作。

接下来需要完成的任务就是使用 SVM 分类器找到整个图像中可以安全嵌入消息的图像块，为了完成这个任务，需要先随机选择一些图片并在位置随机的图像块中嵌入消息，并记录每个图像块i的相关特征$\mathbf{x}_i$，再用经典的隐写分析方法评估得到的伪装图像是否安全记为y_i，其中$y_i \in \{-1,1\}$，将$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_N, y_N)\}$作为训练集来训练 SVM，多次训练，根据准确率找到性能最佳的 SVM，依据其判断在给定图像中的指定位置嵌入消息是否安全。

综上所述，整个隐写系统在使用过程的框架如图 3.2 所示

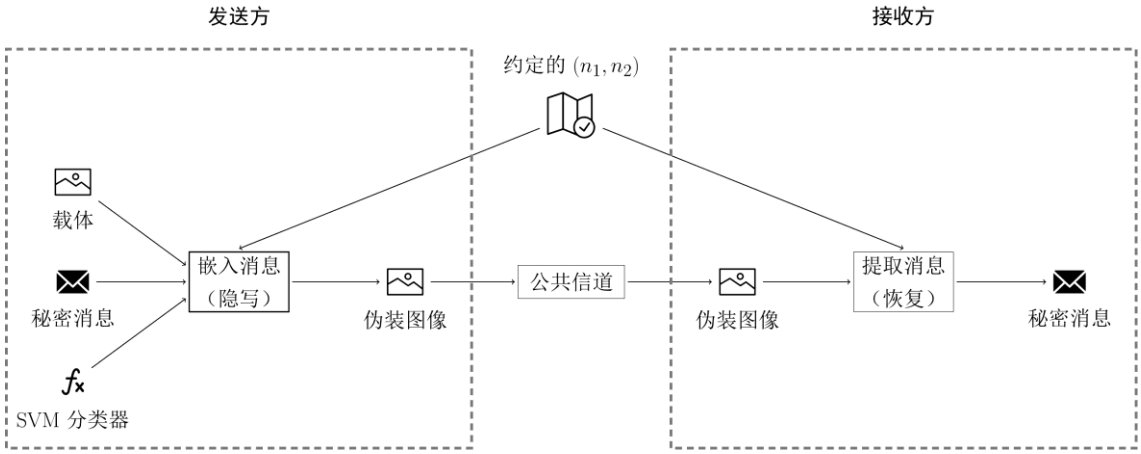

图 3.2　基于 SVM 优化的 LSB 隐写体系框架

3.1　支持向量机

通过以上对隐写分析方法的介绍，容易发现单纯的 LSB 替换或者 LSB 匹配等基于 LSB 嵌入的隐写方案在安全性方面存在一定的缺陷，易受到这些隐写分析方法的攻击。针对这个问题，我们可以引入学习的概念，寻找适合隐藏密码消息的像素的位置，相比按照顺序连续嵌入或使用伪随机数生成器寻找嵌入位置的隐写方法，通过学习现有样本和及对应的评估结果的关系，相比于传统的对抗隐写分析的方法可以省略复杂的分析过程，基于经验得到直观的选择嵌入位置的方法。这里的过程可以被简述为生成一些在不同图像上选择不同位置的 LSB 嵌入的伪装图像，将它们的隐写分析结果连同位置和图像的几个特征作为训练集，使用支持向量机（SVM）进行学习，完成有监督学习得到的分类器用于对需要嵌入消息的图像的位置（像素）进行二元分类，输出结果为适合隐藏消息的位置的集合和不适合隐藏消息的位置的集合。

能实现分类效果的机器学习模型算法数不胜数，每种算法都存在各自独有的优势和劣势。本文选择 SVM 是出于其能最大限度区分出每个像素点是否适合隐藏秘密数据的

考虑，不仅如此，SVM 理论和实践发展较为成熟，关于 SVM 本身的优化和参数调节的研究丰富，易于实现，在不同应用领域的适用性都比较理想，适合用于支持 LSB 隐写方法的改进优化。

对于一个给定的容量为N的训练样本集$D=\{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),\dots(\mathbf{x}_N,y_N)\}$，其中$y_i\in\{-1,1\}$，如果存在可以根据$y$的值线性划分整个样本空间的超平面，那这样的超平面往往不止一个，但是需要挑选最佳的超平面作为分类器。 SVM 方法选择这个最佳的超平面的标准是使两类样本向量中距离超平面最近的向量到超平面的距离之和最小，这个距离之和也被称为间隔，同时，这些到每个类的样本中到超平面距离最近的向量则被称为是支持向量。

SVM 的输出结果是超平面的法向量$\mathbf{w}$和样本空间中的截距b，这两个系数可以唯一确定一个可用于样本分类的超平面，对于任意的向量$\mathbf{x}$，代入函数$f(\mathbf{x})=sign(\mathbf{w}^T\mathbf{x}+b)$可以得到分类结果。

SVM 的训练过程是找到这样一个可以分割不同类样本的超平面并使其中的间隔最大，可以表示为优化过程：

$$\begin{aligned}&\max_{\mathbf{w},b}\quad \frac{2}{\|\mathbf{w}\|}\\&s.t.\quad y_i\left(\mathbf{w}^T\mathbf{x_i}+b\right)\geq 1,i=1,2,\dots,N\end{aligned}\tag{3.1}$$

但是直接从此式入手求解过程有些复杂，使用 Lagrange 乘子法可以把这个问题转化为对应的对偶形式，即凸二次规划问题：

$$\begin{aligned}&\max_{\boldsymbol{\alpha}}\quad \sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\mathbf{x}_i^T\mathbf{x}_j\\&s.t.\quad \sum_{i=1}^{N}\alpha_i y_i=0\\&\qquad \alpha_i\geqslant 0,\qquad i=1,2,\dots N\end{aligned}\tag{3.2}$$

求解得到N维的 Lagrange 乘子向量$\boldsymbol{\alpha}=[\alpha_1,\alpha_2,\dots\alpha_N]$,系数$\mathbf{w}=\sum_{i=1}^{N}\alpha_i y_i\mathbf{x}_i$，根据 KKT 条件中的$\alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i+b)-1)=0$约束，找到一个对应的 Lagrange 乘子$\alpha_i$为 0 的向量$x_i$，也就是支持向量，代入求得$b$的值。这样简化后的过程可以通过简单的编程完成，也可以使用 SMO 算法[17]等技巧提升计算速度。

原始的 SVM 方法如上所述，可以解决大部分的简单分类问题，但是依然存在一定的局限，如对于线性不可分的训练集无法得到闭解，以及对于最大间隔的追求可能导致出现过度拟合。所以在基本的 SVM 之上可以引入核技巧或者软间隔的概念提升 SVM 的适用性。

3.1.1 核函数

对于线性不可分的情况，我们可以将n维向量$\mathbf{x}$变换为新的d维特征空间的向量$\phi(\mathbf{x})$，一般来说低维无法解决的问题往往向高维寻求解决方法，因此，通常有$d \geq n$。如果我们规定新的特征空间中两个向量的内积可以用函数$k(\cdot,\cdot)$来表示，即$k(\mathbf{x},\mathbf{x}') = \phi(\mathbf{x})^T \cdot \phi(\mathbf{x}')$，这个函数被称为核函数。使用核函数代换原来的对偶问题中目标函数的低维内积操作为$\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j)$，以及最后的训练结果也就是分类函数

$$\begin{aligned} f(\mathbf{x}) = \quad & sign\left(\mathbf{w}^T\phi(\mathbf{x}) + b\right) \\ & sign\left(\sum_{i=1}^{N}\alpha_i y_i k(\mathbf{x}_i,\mathbf{x}_j) + b\right) \end{aligned} \tag{3.3}$$

这种计算技巧被称为是核技巧，在原特征向量空间无法解决或者解决效果不理解的 SVM 问题中，应用特征空间变换操作$\phi(\cdot)$可以把样本映射到高维的特征空间寻找可行的解，但是由于我们很难直接计算高维内积，所有可以直接使用现有的核函数计算内积，也等价于应用了该核函数对应的空间变换操作。在设置 SVM 参数的过程中，选择核函数即选择空间变换的具体形式，如应用可以把原向量映射到无限维空间的高斯核（下文提到 RBF 核为其中的一种特例）就相当于选择训练和预测过程中把样本变换为无限维。所以，选择一个合适的核函数对于训练效果和预测性能是至关重要的，表 3.1 列出了常见的核函数：

表 3.1 常见核函数

核函数	表达式	参数
线性核	$k(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$	无
多项式核	$k(\mathbf{x}_i,\mathbf{x}_j) = (\gamma\mathbf{x}_i^T\mathbf{x}_j + r)^d$	$d \geq 1$为多项式的最高次数，$\gamma > 0$为系数，r为常数项
RBF 核	$k(\mathbf{x}_i,\mathbf{x}_j) = \exp\left(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$	系数$\gamma > 0$
sigmoid 核	$k(\mathbf{x}_i,\mathbf{x}_j) = \tanh\left(\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2 + r\right)$	系数$\gamma > 0$，常数项$r < 0$

除此之外，经过证明，当且仅当$n \times n$上对称函数$f(\cdot,\cdot)$对于任意的n维向量$\mathbf{x}$组成的数据集$D = \{x_1, \ldots x_m\}$的运算得到的$m \times m$维核矩阵$\mathbf{K}_{i,j} = f(\mathbf{x}_i,\mathbf{x}_j)$是半正定矩阵时，这个函数$f$有对应的空间变换映射的，可以作为 SVM 的核函数使用[18]。因此，除了这些常用的核函数，在实际应用中也可以通过对常用核函数进行线性组合、求直积和其他方法设计实用的核函数。

3.1.2 软间隔 SVM

总结上文，在处理线性不可分的样本集时，将样本空间进行变换，也就是通过核技巧找到可以把异类样本完全分开的曲面。然而事实上训练过程中很难确定能完全正确分类训练集的核函数，即使找到了这样的核函数也有很大是过度拟合的结果，在预测时表现出的效果并不理想。所以，我们可以在训练分类器的时候，对于样本分类的正确性作出一定的妥协，以获得可行的分类函数以及更大的间隔，期望得到更健壮的 SVM 分类器，这就是软间隔 SVM 的思想。

为了表现违反分类规则的样本在该分类下的“偏离程度”，我们使用线性的 hinge 损失函数来量化

$$l_{hinge}\left(y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)\right)=\max\left(0,1-y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)\right) \tag{3.4}$$

并把这一项乘以一个系数C与原最小化优化目标函数作和得到新的目标函数，经过与硬间隔 SVM 相似的求解方法得到一个几乎与硬间隔完全完全相同的凸二次规划表达式，除了α_i的约束由原来的$\alpha_i \geq 0$缩小为$0 \geq \alpha_i \geq C$。参数C的作用是调节训练过程中对于间隔大小和对训练集样本分类准确率的权重之比，一般来说，C越大，训练过程中对于准确率的偏好越高，对于违反分类的情况的容忍度越低，最后得到的间隔越小，当 C 趋向于无穷大时，这样的 SVM 也就退化为了对分类错误零容忍的硬间隔 SVM。

在下文的方法中，为了得到高性能的 SVM 分类器，我们的方法将选择引入核函数的软间隔 SVM 对图像样本以及其隐写效果进行分类和训练，再对给定图像预测出适合隐藏秘密消息的像素位置。

3.2 图像块选择

首先为了确定图像块的大小，我们规定使用的图像块为$n \times n$的正方形，因为在同等面积的情况下使用正方形的像素块隐藏数据比较均匀，一般来说可以在一定程度上抗视觉攻击。假设 8bit 存储的秘密消息的长度为l，载体图像大小为$M \times N$，那么正方形图像块的边长的必须满足

$$(n-1)^2 < 8l \leqslant n^2 \tag{3.5}$$

容易求得$n = \lceil 2\sqrt{2l} \rceil$其中$n < \min(M, N)$，因此，这样的模式对嵌入率提出更多的限制。

在使用这个新的隐写系统时，首先根据消息长度计算图像块边长l，确定图像块的大小，并将l的信息嵌入图像的前l_1位。接下来需要在整个图像中寻找适合隐藏消息的图像块，SVM 训练完成后，可以使用启发式的方法或者单纯运用随机的方法找到几个图像

块作为 SVM 的输入，从中挑选适合被 SVM 分类器判定为适合隐藏数据的图像块进行消息的嵌入，并将起始坐标(x_0, y_0)嵌入图像前l_1后紧接的l_2位。

3.3 特征选择

为了评价一个给定的图像块是否适合隐藏给定的消息，本文选择了四个特征来描述这个图像块。这四个特征分别是方差、整体差异度、sc 匹配度和平滑度，它们体现的特征包括图像块像素值的多样度、与整个图像在像素值分布上的相似性、图像块 LSB 平面与秘密消息的匹配程度以及图像块本身的连续程度，相对于上文提到的几种隐写分析方法针对图像某个单一特征做出评估，我们的这些特征更能完整评价图像块是否可以用于隐藏数据。这些特征的采集方法实现可以参见附录 C 的 MATLAB 源代码。

3.3.1 方差

本文对于图像块的方差理解和统计学中经典的方差概念一致，用以反映一个样本集中所有样本的离散程度。这里将图像块中每个像素作为一个样本，整个图像块作为一个样本集进行计算，$m \times n$像素的图像块B的方差可以按照如下公式计算：

$$var(B) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\left(x_{i,j} - \bar{x}\right)}{m \cdot n - 1} \tag{3.6}$$

这里的$x_{i,j}$表示坐标为(i, j)的像素的值，$\bar{x}$为该图像块所有像素的值的平均值。

一般来说，越平滑的图像块方差越小，其中包含的像素之间的差异越小，越不适合隐藏数据。

3.3.2 整体差异度

整体相似度反映了图像块与被提取的原图像的关系，我们用像素值的频率来描述图像和图像块的一些特征，使用它们之间的差值的平方和，也就是频率差值向量的二范数的平方来放大图像块与原图像之间的差异。对于大小为$m \times n$图像块B和大小为$M \times N$图像I，频率向量fre分别用公式 3.8 计算：

$$\begin{aligned} fre(B)_i &= \frac{\sum_{k=1}^{m}\sum_{j=1}^{n} p\left(x_{k,j} = i\right)}{m \cdot n}, \quad 0 \le i \le 255, i \in Z \\ fre(I)_i &= \frac{\sum_{k=1}^{M}\sum_{j=1}^{N} p\left(x_{k,j} = i\right)}{M \cdot N}, \quad 0 \le i \le 255, i \in Z \end{aligned} \tag{3.7}$$

其中$p(\cdot)$函数是一个值域为$\{0,1\}$的逻辑判断函数，当自变量表达式的值为真时其p函数的值为 1，否则为 0。$fre(\cdot)$的值为一个 256 维向量，上式给出的是向量中每一个值的计算方法，这里的索引值从 0 开始计数。 接下来我们计算这两个向量中每个对应的像素值间的差值的平方和作为整体差异度

$$D_{B,I}=\sum_{i=0}^{255}\left[\, fre(B)_i - fre(I)_i \,\right]^2 \tag{3.8}$$

通过上述的计算，我们可以容易发现，对于这个可以反映像素值分布差异的特征，不同于其他特征仅考虑用于隐藏消息的位置自身的特性，整体差异度还考虑了选取的图像块在整个原图像中扮演的角色，也就是说与整体的融合程度。从概念来看，在图像块大小一定的情况下整体差异度的值越大说明该图像块在视觉上越突出，但不能简单地反映是否适合隐藏图像，只能作为一个用以辅助参考的特征。具体来说这样的“突出”需要分具体情况探讨在哪方面突出，隐藏数据的数据是否会因此在视觉攻击时暴力，都需要结合其他特征，如平滑度评价是否为不宜嵌入数据的突出的平滑区。另外，虽然在计算频率时有归一化的处理，但整体差异度的取值仍然很大程度地受到图像块大小的影响，这种计算方式的在图像块过小的时候将失去意义。

3.3.3 sc 匹配度

sc(secret-cover)匹配度是指图像块本身的 LSB 平面与秘密消息的二进制表达之间相似程度，可以用来描述秘密消息和图像内容的关系。我们在介绍隐写术的时候已经提到为了不让人觉察到隐藏消息的存在，秘密消息的内容和图像内容往往是无关的，所以在这里说的内容并非其表达的内容，这样的内容因为多媒体文件的存储和表达的原因，很难在高层表现上让人察觉两种的联系，但可以通过底层实现探究图像和消息的关系，具体来说就是把它们都转换为二进制的表达形式，度量这些二进制数据间的相似程度。这种做法看似毫无意义，但是考虑到我们对秘密消息的隐写和恢复的过程也是通过将图像和消息都转换为二进制数据来实现的，以秘密消息的二进制形式来代替原来的二进制 LSB 平面，所以计算它们之间的相似性对于评价图像块是否适合隐藏特定的数据非常重要。

我们将长度为l的秘密消息转换为 8bit 的二进制数据并以 0/1 向量表示为$\overrightarrow{M_{Binary}}$，同时以自然顺序取图像块中前$l\times 8$像素的 LSB 位并构建 0/1 向量$\overrightarrow{B_{LSB}}$，这两个向量中对应位置的元素相同的个数与整个向量长度之比记为 SC 匹配度，按照公式 3.9 计算

$$sc\_match=\frac{\sum_{i=1}^{8l} p\left(M_{Binary}(i)=B_{LSB}(i)\right)}{8l} \tag{3.9}$$

这里使用$M_{Binary}(i)$和$B_{LSB}(i)$分别表示$\overrightarrow{M_{Binary}}$和$\overrightarrow{B_{LSB}}$的第$i$个元素，函数$p$的定义与整体差异度中的函数$p$一致，用以判定两个二进制表达中相同的位。

一般来说，对于性质良好的自然图像，sc 匹配度越高，隐藏的数据就越能被察觉。在嵌入率不变的情况下，随着 sc 匹配度的提高，隐写过程对于载体图像的修改量越少，得到的图像越接近原图像，如果原图是典型的自然图像，也能很好地符合一些自然图像特征的图像，那么得到到伪装图像也接近自然图像，很难通过现有的隐写分析手段探查出隐藏消息的存在。我们可以想象一种极端的情况，要嵌入的秘密消息（包括消息长度等辅助信息）的二进制表达形式恰好与原图像的 LSB 平面完全一致，也就是 sc 为 100%时，消息的嵌入过程不需要对图像做任何修改，载体图像与伪装图像完全一致，所有的隐写分析方法对于载体和伪装的检测结果都相同，显然无法发现隐藏消息。同理，sc 匹配度为 0 时也相当于对图像用于嵌入消息所有像素的 LSB 位都进行了修改，这时伪装图像被检测出异常的概率就非常高。然而，这两种情况都太过理想。对于典型的隐写应用场景来说，秘密消息是特定的，图像以及可以选择的图像块也是有限的，且二者作为承载信息的数据都表现出一定的规律性，极少出现匹配率极高和极低的情况。为了模拟隐写场景，我们在实验中使用 0-255 间的随机整数构成数组作为秘密消息，经过测试，sc 匹配度往往在 0.4-0.6 之间。

3.3.4 平滑度

平滑度的概念是借鉴了前文 RS 隐写分析中提到的平滑度计算，两者有相似之处，在本文中使用的平滑度相对于 RS 隐写分析中提出的平滑度函数在定义上更为完善。原始的平滑度函数中将像素矩阵降维处理压平为向量，然后根据一个方向计算差值的绝对值之和，如随着i的增大累加$|x_i - x_{i+1}|$，这样的简单计算一个像素与前后像素的差值的模式不能完全反映该像素与周围像素的关系，因此，这里我们需要全面改进这种计算方式。

对于非边缘的像素x_i，周围一共有 8 个像素，这 8 个像素位于该像素的 8 个不同方向，这种扩展的差值计算也被应用在一些隐写分析算法中[19]，可以根据方向表示为关于x_i的集合

$$S_i = \left\{x_{i\leftarrow}, x_{i\rightarrow}, x_{i\uparrow}, x_{i\downarrow}, x_{i\nearrow}, x_{i\swarrow}, x_{i\nwarrow}, x_{i\searrow}\right\} \tag{3.10}$$

那么对于在边缘的像素点，相应的S_i集合则会缺少一些方向的数据，仅仅保留周围存在的像素。那么我们可以使用公式 3.11 的求和方式计算$m \times n$大小的图像块B的平滑度

$$Smooth(B)=\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{x'\in S_{ij}}\left|x_{ij}-x'\right|}{m\cdot n} \tag{3.11}$$

此时得到的平滑度函数充分考虑了像素点和各个方向的邻接像素的关系，但是，我们容易看出，有些差值也因此被重复计算，而且直接利用以上公式求值的计算量较大，因此我们可以使用矩阵偏移的方式计算平滑度。

令$B((x_1,y_1),(x_2,y_2))$表示以(x_1,y_1)为起始坐标（左上角），(x_2,y_2)为终止坐标（右下角）的像素矩阵，那么整个图像块B可以表示为$B((1,1),(m,n))$，我们可以使用如下公式计算每个方向上的像素绝对差矩阵

$$AD_{\rightarrow}=\left|B\left((1,1),(m,n-1)\right)-B\left((1,2),(m,n)\right)\right| \tag{3.12}$$

$$AD_{\uparrow}=\left|B\left((1,1),(m-1,n)\right)-B\left((2,1),(m,n)\right)\right| \tag{3.13}$$

$$AD_{\nearrow}=\left|B\left((1,1),(m-1,n-1)\right)-B\left((2,2),(m,n)\right)\right| \tag{3.14}$$

$$AD_{\nwarrow}=\left|B\left((2,1),(m,n-1)\right)-B\left((1,2),(m-1,n)\right)\right| \tag{3.15}$$

对由公式 3.12-3.15 得到四个矩阵中每个元素作求和运算，则可以避免重复计算。另外，直接对矩阵进行运算也大大提高了计算效率。

平滑度沿用了 RS 隐写分析中对于图像混乱程度的定义，与命名的直观含义相违背的是，平滑度越大，各个相邻像素间的差异越大，代表的图像的混乱程度越大。平滑度的含义与方差有一定的重合，都在一定程度上反映了图像各个像素间的差异性，但需要注意的是这两个特征体现了图像不同方面的离散程度，方差仅考虑图像作为一个无序像素集时各个像素之间的关系，而平滑度则考虑了像素在图像中的位置及其与周围像素的关系，两个特征存在一些联系但并不正相关，假设有一个图像块仅由同样大小的黑色色块（像素值 0）和白色色块（像素值 255）拼接而成，且交接的边界很小，对其计算两种特征，我们会发现它的方差很大但是平滑度不一定很大，尤其是对于图像本身较大的情况。

3.4 SVM 的训练和预测

本文中对于 SVM 的使用分为两个阶段，一个为使用数据集建立模型的训练阶段，另一个是将其作为模型对样本进行分类的预测阶段，后者包括也包含训练过程中进行检验和调整时进行的测试性预测，但主要的使用场景是在训练完全完成后的应用阶段，训练和应用的整体流程如图 3.3 所示。

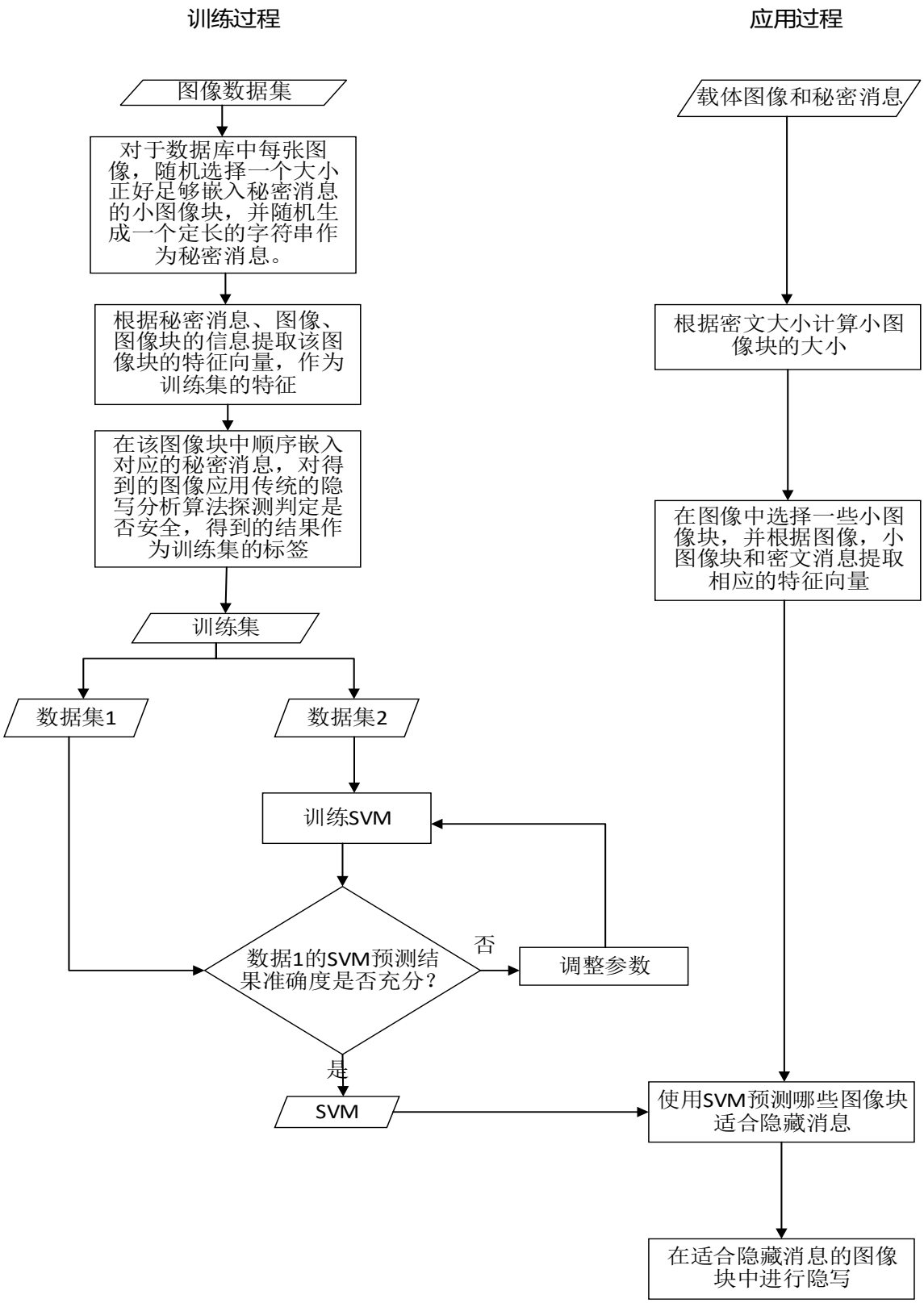

图 3.3　SVM 的训练和应用流程图

在我们的优化方案中，已经完成训练的 SVM 连同秘密消息以及载体图像是作为嵌入消息过程的输入使用的，因此对于整个隐写系统来说，训练一个性能良好的 SVM 是预处理过程中应该完成的，且训练好的 SVM 是可以用于下一次隐写的。

首先我们需要以随机采样的方法构造训练集、检验集和测试集。应用场景假设在预处理过程中我们有足够多的图像资源和计算资源，生成长度已知的随机数组作为嵌入消息，那么构造训练集的过程可以随机的在每个已有的图像上选择若干个图像块，对于每个小图像i按照前文提到的方法提取对应的四个特征作为该样本的特征向量$\mathbf{x}_i$，在小图像块嵌入消息后，使用基于阈值的 RS 隐写分析算法判断得到的整个伪装图像是否可以被认定为隐藏了消息的可疑图像，将隐写分析的结果作为样本的标签y_i，这些信息采集的具体编程实现可以参考附录 C。

在预设的资源足够的预处理阶段，使用这样的采样方法可以构造出足够大的样本集。我们可以把这样的样本集简单分割为训练集、检验集和测试集，也可以按照一定的指标对样本分组进行测试，如嵌入率。

值得注意的是，采样过程中为了使样本尽可能多样化，我们对于每个样本使用的是随机数组，然而实际上消息隐写的实践中消息本身不可能完全随机无序，在二进制表达上会体现出一定的规律性，如字母的 ASCII 编码的二进制形式第一位为 1，所以由字母构成的字符串在转化为字节流时也会每隔 8 位出现 1，我们在这里不考虑这种规律给我们训练上带来的影响，因为本文的目的是训练出隐写技术适用范围更广泛的 SVM。

3.5 本章小结

本章详细讨论了基于 SVM 的 LSB 隐写优化方案。在本章的开始提出了整个引入 SVM 后的隐写系统的架构。

本章在 3.1 小节具体讨论了 SVM 在样本上的优化过程以及核函数和软间隔在提升 SVM 性质方面的作用。3.2 节探讨了图像块大小的计算方法，而图像块位置选择需要依据的特征则在 3.3 中给出定义和计算的方法，这些特征提取方法的实现也在文章中给出。

在以上理论和实现基础论述结束后，3.4 节具体讨论了 SVM 在整个隐写体系中所扮演的角色，说明了预处理阶段和隐写阶段 SVM 分别将经历的训练和预测过程。这个部分也是基于 SVM 优化的核心内容，但是这种优化中 SVM 具体的参数设置实验中观测和验证。根据直觉通过选择良好的特征确实可以使隐写系统在安全性上得到一定的优化，但优化的程度则需要根据不同情况对于不同的隐写分析算法进行讨论。本章仅仅探究了基于 SVM 的 LSB 图像隐写优化的理论基础，在理论分析上给出了可行性，而实际这种

系统是否可行以及优化效果是否理想则需要通过在大量样本上的实验观察。实验的设置和结果分析部分将在第四章给出。

4 实验与结果分析

4.1 数据集合实验平台设置

本文的实验中使用了 v2 版本的 UCID 图像数据集[20]作为图像资源。UCID 数据集包含了 1338 张未经压缩的 RGB 彩色图像，每张图像的大小为384×512或512×384像素，格式为 tif，图像的主题丰富，包括了风景、建筑、动植物和人物。为了简化操作突出实验的效果，我们把这些图像都转化为了灰度图像，实际上我们的机制对于像素图像的处理原理相同，使用 RGB 图像的效果也类似，因此不再重复实验。

实验中使用的操作系统为 Windows10，数据的部分预处理和收集工作使用 Python3.5 完成，大部分的运算则在 MATLAB2015b 实现，其中 SVM 相关部分的训练和预测借助了 LIBSVM3.21[21]工具箱完成。

4.2 实验设置

因为嵌入率对于隐写分析效果的影响巨大，而使嵌入率 5%-50%是隐写的常见情况，为了在隐写系统中充分使用我们训练出来的模型，实验中我们设置了步长为 5%的嵌入率从 5%上升到 50%的 10 组数据样本量为 200 的样本合成为一个样本量为 2000 的训练集，用另外 2000 个嵌入率分布相同的样本作为检验集。这些样本的采集过程则是迭代地在不同的 200 张灰度图像上随机选择与嵌入率相符的大小的图像块进行隐写，并提取特征和分析结果作为样本的数据。使用训练集训练不同参数的 SVM 后，我们通过它们在几个检验集上的表现调整训练参数，综合考虑支持向量的个数和准确率选择最佳的 SVM 分类器。

4.3 实验结果分析

我们使用了 80 组不同的参数对 SVM 进行训练，这些参数包含了核函数的类型，次数，系数，以及软间隔 SVM 的代价，它们的关于训练集的错误率E_0和检验集的错误率E_1如散点图所示，图 4.1 中散点的大小由软间隔 SVM 的参数，代价C的大小决定，颜色由多项式核的次数和 RBF 核的系数共同决定，从散点图 4.1 可以发现对于低次的多项式核来说，系数的变化引起的训练效果几乎没有影响，但对于代价的变化非常敏感。将样本特征空间映射到无穷多维的 RBF 核在训练集的表现上始终维持了较低的错误率，但代价较高且系数较大的 RBF 核 SVM 在测试集上的错误率就出现了明显的上升，这是一种过度拟合的现象。总之，这些 SVM 的训练结果符合 3.1 中给出的推断。

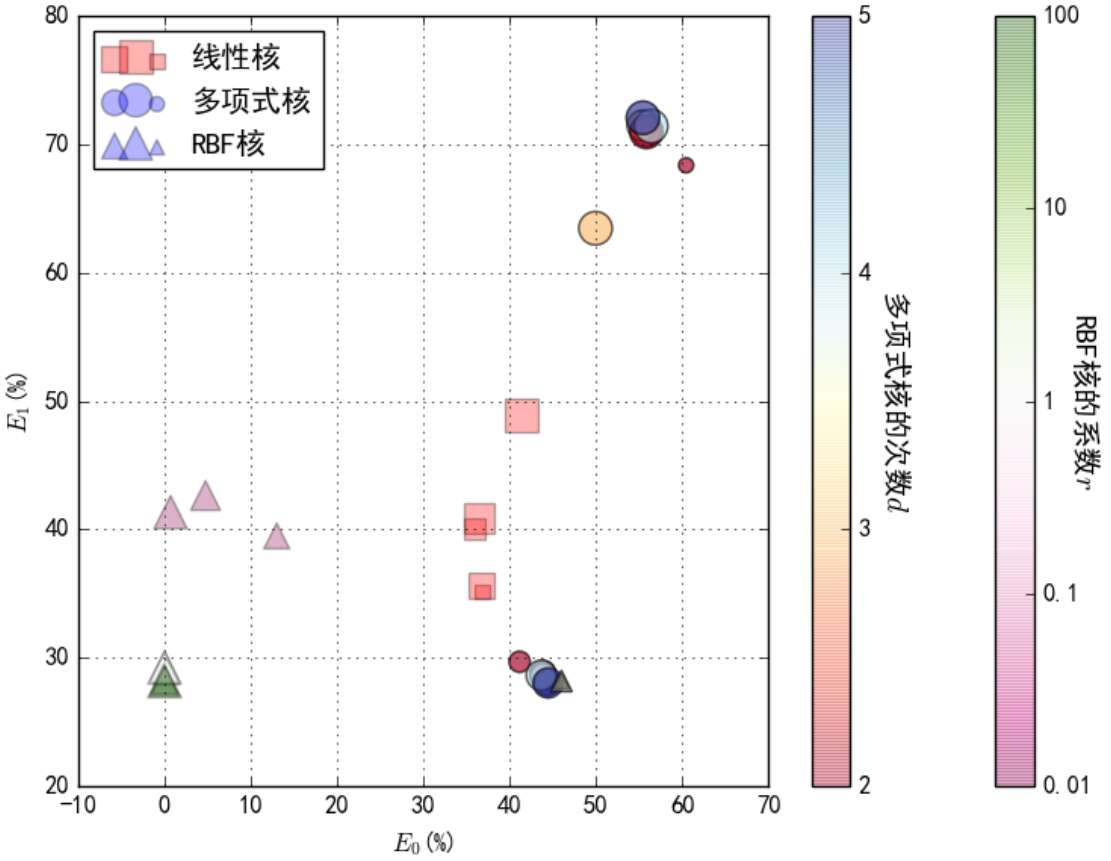

图 4.1　以不同参数训练得到的 SVM 分类器的错误率

我们选择其中比较性质较好的 6 组典型的参数得到的 SVM（见表 4.1）进行后续实验。

表 4.1　6 个典型的 SVM

编号	核类型	代价C	次数d	系数γ	训练错误率E_0	检验错误率E_1	支持向量数
1	线性核	0.01	1	1	36.8	35.15	1566
2		1		1	36.75	35.65	1045
3	多项式核	0.1	2	1	41.15	29.7	208
4		0.1	4	1	43.8	28.65	90
5		10	4	0.1	43.65	28.65	83
6	RBF 核	1	无穷大	1	0.2	28.95	2000

接下来我们使用这 6 个 SVM 分类器对于嵌入率为 5%-50%的 10 组在 1338 张图像中随机选择 5 个图像块嵌入消息得到的样本集进行分类测试，结果图 4.2 所示

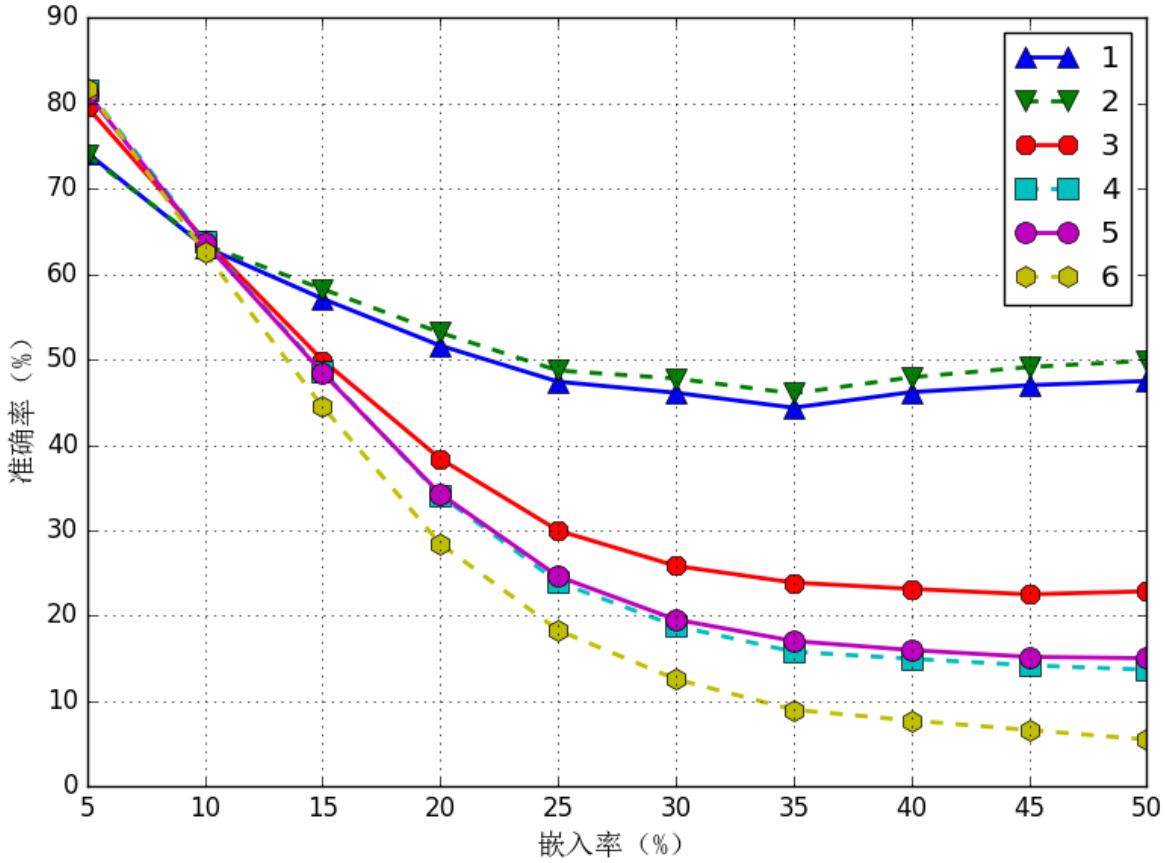

图 4.2　不同嵌入率下 6 种 SVM 对于 RS 隐写分析的预测准确率

从图 4.2 中可以明显观察得到，所有分类器的预测准确率都随着嵌入率的升高而表现出下降的趋势，但不同的是低次低代价C的线性核和多项式对应的准确率的下降相对于高次高代价C的多项式核以及 RBF 核更为平缓，虽然在低嵌入率时前者的表现较差，但即使嵌入率升高，前者的表现依然稳定。这是因为高次核以及高代价有很大概率会导致过度拟合，随着测试样本的变化，预测的准确率会发生很大的变化。同时，当嵌入率增大时，图像块的大小也随之增大，包含的信息量也随之增加，有限的选取特征对于图像块的描述力度减小，图像块对于用以隐藏消息是否适合的不确定度也有所增加，之前的分类器的决策水平随之降低，所以会出现准确度下降的现象。但是从整体上看，我们可以认为至少当嵌入率为 5%~25%时，我们的分类器确实可以以高准确率帮助隐写者判定适合隐藏数据的位置。

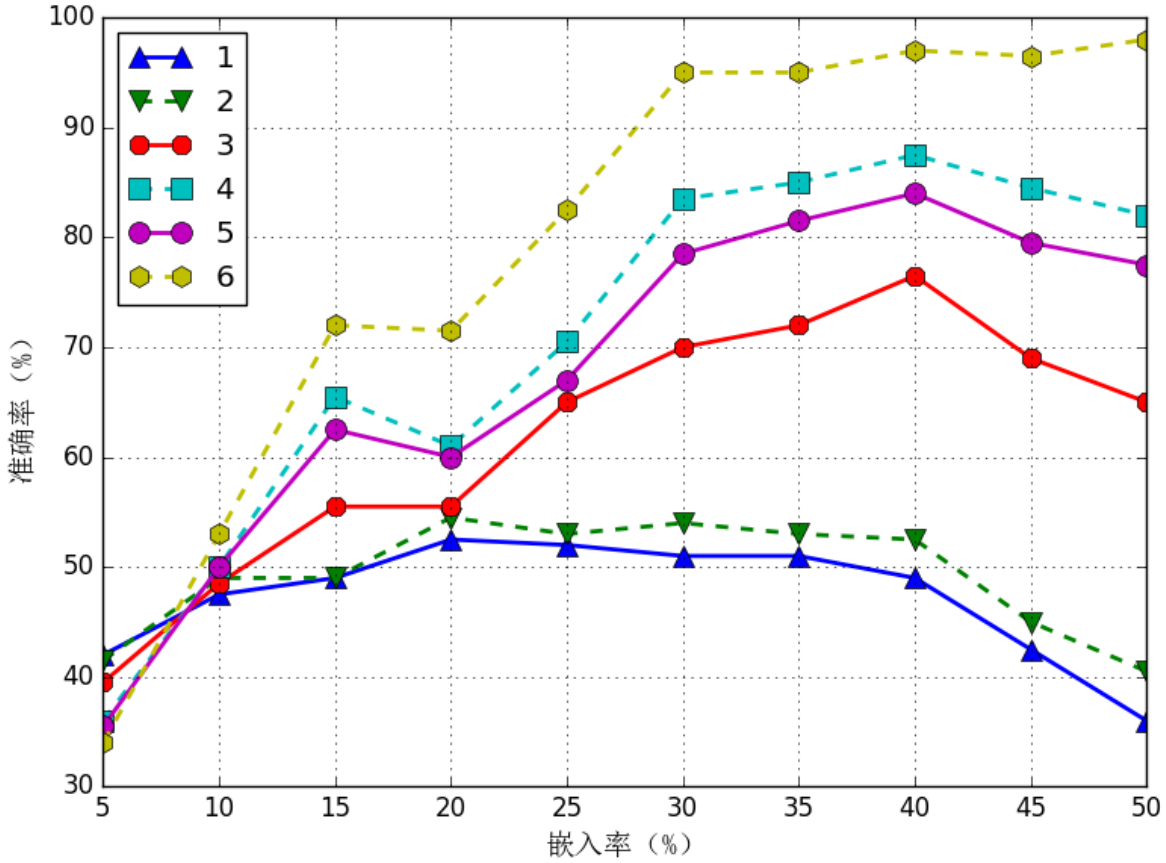

图 4.3　不同嵌入率下 6 种 SVM 对于样本对分析的预测准确率

接下来使用这 6 个 SVM 在 5%-50%的 10 组样本量均为 200 的图像块上预测隐写的安全性，并使用样本对分析（SPA）的方法检验预测的准确率，可以绘制出图 4.3。

从图 4.3 可以观察到，在随着嵌入率在 5%-25%间升高，所有 SVM 分类器的准确率都有一定的上升趋势，当嵌入率到达 25%以后，低次低代价的核在准确率上的表现开始出现回退，而高次核的表现出现很大的提高，这种结果正好与使用 RS 隐写预测的结果相反，也说明了高次核的使用对于对抗 SPA 攻击非常有效。通过第 3 章的研究，我们可以很容易发现 RS 隐写分析主要考虑特征变化自身单纯的变化，而 SPA 分析则将很多可能出现的特征结合考虑，分析的过程更为复杂，这种情况下使用高次核可以获得更好的预测效果。

低次核与高次核都有各自的优势和劣势，需要考虑具体的使用场景进行应用。当然，在以后的工作中，我们也可以考虑根据应用场景使用对多个低次核和高次核进行加权聚合得到一个更强大的 SVM。

接下来又随机抽样得嵌入率为 5%-50%且每个嵌入率对应的样本量均为 200 的 10 组伪装图像样本并应用χ^2测验评估这些图像是否存在隐写，然后使用这 6 个 SVM 对其进行预测，预测的准确率如图 4.4 所示。

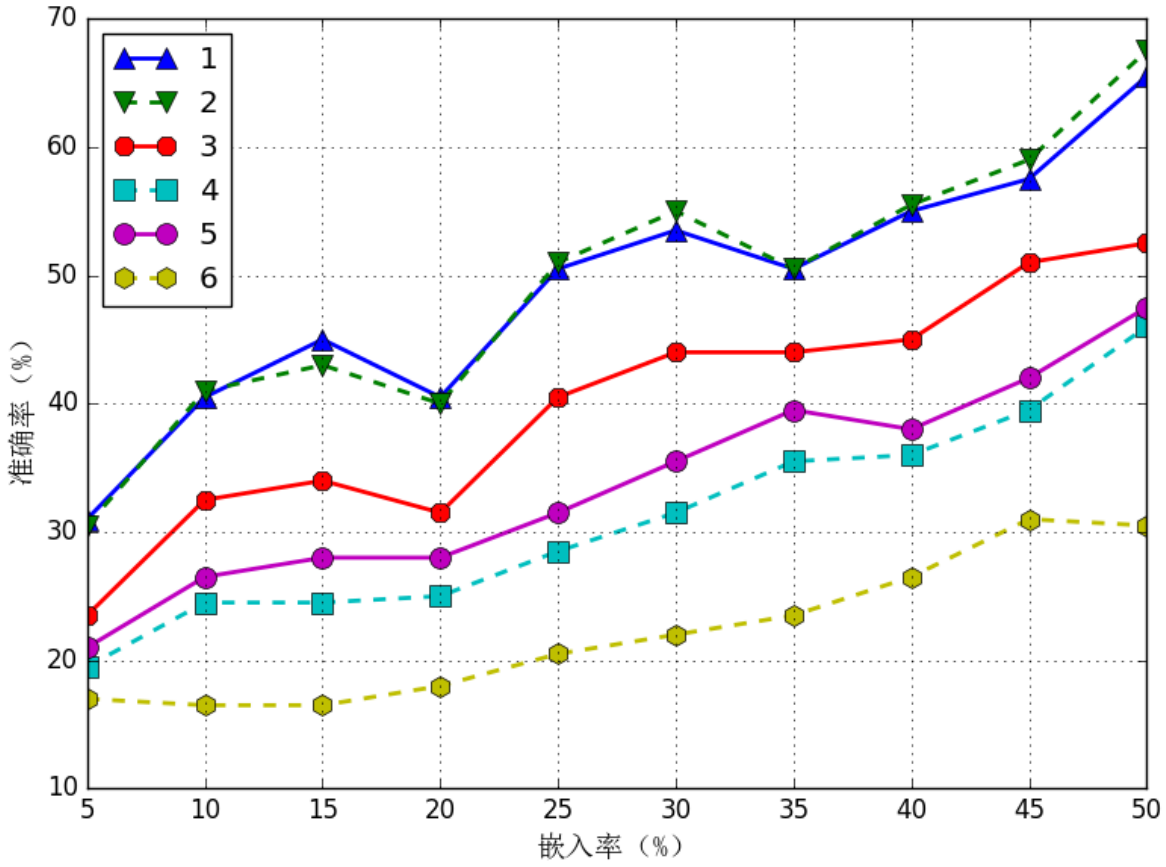

图 4.4　不同嵌入率下 6 种 SVM 对于χ^2测验分析方法的预测准确率

从图 4.4 中可以看到对于χ^2测验的准确率的变化趋势与 RS 隐写分析（图 4.2）以及 SPA 分析（图 4.3）都存在相似之处。一方面，对于χ^2测验的检验结果如 RS 隐写分析一样，表现最佳的是低次低代价核，而高次高代价的核表现始终不太理想；另一方面，新的机制在对于χ^2测验的预测准确率也随着嵌入率的上升表现出提高的趋势，这种趋势与对于 SPA 方法的预测准确率相似。相对来说，χ^2测验也是一种简单的分析方法，因此使用低次核去模拟分类过程得到的结果更佳；而χ^2测验相较 RS 隐写分析对于嵌入率更加敏感，因此 SVM 的分类在高嵌入率的情况下更有优势，此时使用 SVM 可以快速找到不适合隐藏的位置，对安全性的保障有所提高。但与此同时，SVM 也可能更难找到一个适合隐藏的位置，需要从更大的图像块样本中寻找特征较好的，在高嵌入率情况下尽管

准确率提升但隐藏数据的效率出现下降，也是所有隐写方法都无法避免的问题。当然，值得注意的是χ^2测验的评估结果非常依赖于载体图像的性质，在载体图像的选择有限的时，也就是大部分实际的应用场景中，这个预测结果的意义不大，对于优化的评估主要参考依据应该是对于 RS 隐写和 SPA 隐写的准确率，我们可以发现，挑选合适的 SVM 可以使 LSB 隐写在安全性方面至少获得 10%的提升。

4.4 本章小结

本章基于第二章和第三章给出的方法及优化给出实验和结果分析。4.1 节说明了实验所使用平台、编程语言、工具箱等信息，4.2 节为实验中嵌入率、样本量等全局参数的设置并说明了设置的原因。4.3 节则详细分析了实验的结果。实验中使用了 80 组不同的参数对 SVM 进行训练，最后选出 6 组性质较为典型的 SVM 进行关于χ^2测验，样本对分析和 RS 隐写分析的预测准确率实验，发现对于不同的隐写方法，6 组 SVM 在不同隐写率下的预测表现有所不同，这些差异符合第二章中关于隐写分析方法的理论。但是，无论对于哪种隐写分析方法，优化方案中引入的 SVM 总能在选择合适的参数的情况下使得准确率上获得客观的提高，因此可以认为本文中引入 SVM 的 LSB 信息隐藏方法确实使隐写在安全性上获得优化。

结　论

通过本次研究和改进 LSB 信息隐藏的方法，我深入地学习了隐写术、隐写分析以及机器学习相关领域的知识，并作出相应的思考将其有机结合提出了一些创新性的内容，为了实现这些内容我学习了相关的编程技术和工具的使用，并学到了如何收集数据进行分析。期间遇到了不少问题，我通过查阅资料、与他人交流和使用学校的硬件资源等方式解决了这些问题并顺利完成了本文，收获不不少的知识和技能，更重要的是了解到了怎样将自己的灵感转化为可以实现的技术。

本文对 LSB 隐写系统进行了深入的研究并探索了现有模式的局限，提出了一种适用于缺少安全信道交换秘钥时对 LSB 隐写在安全上的改进，但是这种改进的代价是需要寻找适合的数据集训练分类器，消耗一定的计算资源得到可以长期使用的选择隐写位置的 SVM 分类器，相较传统的顺序隐写方法，该分类器经过试验评估，确实可以在一定程度上提高隐写的安全性。

随着数据挖掘的研究日益多元和深入，在本文之前就有不少结合隐写相关技术和机器学习方法的成果，但这些成果主要集中在隐写分析领域，几乎没有使用机器学习方法对于隐写技术本身进行优化的尝试，文中提出的隐写系统填补了这方面的空白，在机器学习和隐写术两个领域间建立了一种新的联系，将为未来的相关工作者提供一些思路。

当然，这种优化的模式由于研究时间和资源有限以的原因，以及本人的技术水平不足，仍然存在一些局限，因此未来还有很多工作需要完成，例如前面提到的可以探索出一种有效的加权聚合方法获得鲁棒性更强准确率更高的分类器，另外，也可以引入有效的启发式算法最快寻找可能的性质最好的图像块。当然，除了定性的分类方法，也可以考虑使用 SVM 回归或其他回归方法定量地定位适合隐藏秘密数据的区域。

参 考 文 献

[1] SCHAATHUN H G. Machine learning in image steganalysis[M]. Chichester West Sussex UK: Wiley-IEEE Press, 2012.

[2] BACKES M, CACHIN C. Public-Key Steganography with Active Attacks[C]//Theory of Cryptography: Second Theory of Cryptography Conference, TCC 2005, Cambridge: Springer, 2005, 3378: 210.

[3] FILLER T, FRIDRICH J. Complete characterization of perfectly secure stego-systems with mutually independent embedding operation[C]//Acoustics, Speech and Signal Processing, 2009. Taipei: IEEE, 2009: 1429-1432.

[4] FRIDRICH J, GOLJAN M, SOUKAL D. Searching for the stego-key[C]// Electronic Imaging 2004. San Jose, California, USA：International Society for Optics and Photonics, 2004: 70-82.

[5] WAYNER P. Disappearing cryptography: Information hiding: Steganography and watermarking (2Nd edition)[M]. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.

[6] FRIDRICH J, GOLJAN M, DU R. Steganalysis based on JPEG compatibility[J]. Proc Spie, 2001, 4518: 275-280.

[7] HEMPSTALK K. Hiding behind corners: Using edges in images for better steganography[C]//Proc. computing women’s congress, hamilton, new Zealand. 2006.

[8] SINGH K M, SINGH L S, SINGH A B, et al. Hiding secret message in edges of the image[C]//2007 International Conference on Information and Communication Technology. Dhaka, Bangladesh: 2007.

[9] SWANSON M D, ZHU B, TEWFIK A H. Robust data hiding for images[C]// Digital Signal Processing Workshop: proceedings, September 1-4, 1996, Hotel Alexandra, Loen, Norway: IEEE, 1996: 37-40.

[10] WU D-C, TSAI W-H. A steganographic method for images by pixel-value differencing[J]. Pattern Recognition Letters, Elsevier, 2003, 24(9): 1613－1626.

[11] LUO W, HUANG F, HUANG J. Edge adaptive image steganography based on LSB matching revisited[J]. Information Forensics and Security, IEEE Transactions on, IEEE, 2010, 5(2): 201－214.

[12] WESTFELD A, PFITZMANN A. Attacks on steganographic systems, Lecture notes in computer science 1768[J]. Journal of Intelligent Information Systems, 2000, 15(1):5-6.

[13] DUMITRESCU S, WU X, MEMON N. On steganalysis of random LSB embedding in continuous-tone images[C]//Image Processing International Conference on. Rochester, New York, USA: IEEE, 2002: 641 - 644.

[14] FRIDRICH J, GOLJAN M, DU R. Detecting LSB Steganography in Color and Gray-Scale Images[J]. IEEE MultiMedia, 2001 1(4): 22-28.

[15] JOHNSON R, BHATTACHARYYA G. Statistical concepts and methods[M]. Wiley series in probality and mathematical statistics, 1977.

[16] PROVOS N, HONEYMAN P. Hide and seek: An introduction to steganography[J]. Security & Privacy, IEEE, 2003, 1(3): 32-44.

[17] PLATT J. Sequential minimal optimization: A fast algorithm for training support vector machines[R]. MSR-TR-98-14, Microsoft Research, 1998: 21.

[18] SCHÖLKOPF B, SMOLA A J. Learning with kernels: Support vector machines, regularization, optimization, and beyond[M]. MIT press, 2002.

[19] 闫晓蒙,张涛,奚玲,等.一种针对 LSB 匹配隐写的负载定位新算法[J].数据采集与处理, 2016, 31(1):145 - 151.

[20] SCHAEFER G, STICH M. UCID: an uncompressed color image database.[J]. Proceedings of SPIE - The International Society for Optical Engineering, 2004, 5307(1):472-480.

[21] CHANG C-C, LIN C-J. LIBSVM: A library for support vector machines[J]. ACM Transactions on Intelligent Systems and Technology, 2011, 2(3): 27.

附录 A　实现 LSB 隐写的源代码（MATLAB/Mathematica）

Mathematica 实现 LSB 顺序隐写和恢复：

```
StegCover[Carrier_Image, text_] :=
 Block[{CarrierData, TruncatedCarrier, pixelChannels, SecretBits,
   LifeLength, SecretData},
  CarrierData = ImageData[Carrier, "Byte"];
  TruncatedCarrier = BitAnd[CarrierData, 2^^11111110];
  pixelChannels = Apply[Times, Dimensions[CarrierData]];
  SecretBits =
   Flatten[IntegerDigits[
     ToCharacterCode[
      ToString[text, InputForm, CharacterEncoding -> "ASCII"]], 2,
     8]]; LifeLength = IntegerDigits[Length[SecretBits], 2, 48];
  SecretData =
   Fold[Partition,
    PadRight[Join[LifeLength, SecretBits], pixelChannels],
    Reverse@Rest[Dimensions[CarrierData]]];
  Image[TruncatedCarrier + SecretData, "Byte"]]

StegUncover[CoverImage_Image] :=
 Block[{SecretData, LifeLength, secretBytes},
 SecretData = Flatten[BitAnd[ImageData[CoverImage, "Byte"], 1]];
 LifeLength = FromDigits[Take[SecretData, 48], 2];
 secretBytes = Partition[Take[Drop[SecretData, 48], LifeLength], 8];
 ToExpression[
  FromCharacterCode[FromDigits[#, 2] & /@ Take[secretBytes]]]
 ]
```

MATLAB 实现特定位置隐写：

```
function stego=lsb_embed_steg(cover,secret_text,x,y,block_width)
stego=cover;
block=cover(x:x-1+block_width,y:y-1+block_width);
block_byte=reshape(block,1,[]);
protext=dec2bin(length(secret_text),24)-'0';
x_pos=dec2bin(x,10)-'0';
```

```
y_pos=dec2bin(y,10)-'0';
stego(1,1:24)=bitand(stego(1,1:24),254)+uint8(protext);
stego(1,25:34)=bitand(stego(1,25:34),254)+uint8(x_pos);
stego(1,35:44)=bitand(stego(1,35:44),254)+uint8(y_pos);
secret_byte=uint8(reshape(dec2bin(secret_text,8),1,[]))-'0';
block_byte(1,1:length(secret_text)*8)=bitand(block_byte(1,1:length(secret_text)*8),254)+sec
ret_byte;
stego(x:x-1+block_width,y:y-1+block_width)=reshape(block_byte,size(block));
end
```

附录 B　三种隐写分析方法的 MATLAB 源代码

χ^2测验：

```
function [fre_odd,fre_even]=color_dist(im)
idensity=0:255;
[fre,~]=hist(im(:),idensity);
fre_odd=fre(1:2:end);
fre_even=fre(2:2:end);
end

function p=chi_square_test(im)
[h_odd,h_even]=color_dist(im);
s=(h_even-1/2*(h_even+h_odd)).^2./(1/2*(h_even+h_odd));
s(isnan(s))=0;
p=chi2cdf(sum(s),128);
end
```

SPA 方法：

```
X = double(X);
[M,N] = size(X);

u = X(:,1:N-1);
v = X(:,2:N);

Xc = length(find( (mod(v(:),2)==0 & u(:) < v(:)) | (mod(v(:),2)==1 & u(:) > v(:))));
Zc = length(find( u(:) == v(:) ));
Wc = length(find(floor(u(:)/2) == floor(v(:)/2) & u(:)~=v(:) ));
Vc = M*(N-1) - (Xc + Zc + Wc);

a = (Wc + Zc)/2;
b = 2*Xc - M*(N-1);
c = Vc + Wc - Xc;

D = b^2-4*a*c;
if a > 0
    p1 = (-b+sqrt(D))/(2*a);
```

```
    p2 = (-b-sqrt(D))/(2*a);
else
    p1 = -1; p2 = -1;
end

beta_hat = real(min(p1,p2));
```

RS 隐写分析：

```
function dif=f_smooth(im,x,y,G_x_length,G_y_length) %RS 隐写分析中用到的平滑度函数
dif_x=sum(abs(im(x:x-1+G_x_length,y:y-2+G_y_length)-im(x:x-1+G_x_length,y+1:y-
1+G_y_length)));
dif_y=sum(abs(im(x:x-2+G_x_length,y:y-1+G_y_length)-im(x+1:x-
1+G_x_length,y+1:y+G_y_length)));
dif=sum(dif_x)+sum(dif_y);
end

function    [RSU_FM,RSU_M]=RSAnalysis(img,G_x_length,G_y_length)%RS 分析，
x_length,G_y_length 为小图像块（组）的长和宽，返回负翻转和负反转的正则组、奇异
组和不变组的个数。
[r,c]=size(img);
digit_img=double( img(:,:,1) );
img_M=digit_img;
img_FM=digit_img;
for  x= 1 : r
    for y= 1 :c
        if rem(digit_img(x,y), 2 )==0
            img_M(x,y)=img_M(x,y) + 1;
            img_FM(x,y)=img_FM(x,y)-1;
        else
            img_M(x,y) =img_M(x,y) - 1;
            img_FM(x,y)=img_FM(x,y) + 1;
        end
    end
end
```

```
RSU_FM=zeros(1,3);
RSU_M=zeros(1,3);

for x= 1 :G_x_length:r-G_x_length
    for y=1 :G_y_length:c-G_y_length
        f_G=f_smooth(digit_img,x,y,G_x_length,G_y_length);
        f_FMG=f_smooth(img_FM,x,y,G_x_length,G_y_length);
        f_MG=f_smooth(img_M,x,y,G_x_length,G_y_length);
        RSU_FM=RSU_FM+[f_FMG>f_G,f_FMG<f_G,f_FMG==f_G];
        RSU_M=RSU_M+[f_MG>f_G,f_MG<f_G,f_MG==f_G];
    end
end
end
```

附录 C　特征提取的 MATLAB 源代码

```
function fre=color_frequency(im)%返回给定图像中的色彩分布矩阵
[r,c]=size(im);
idensity=0:255;
[fre,~]=hist(im(:),idensity);
fre=fre/(r*c);
end

function diff_square=local_diff(img,block)%整体差异度
fre_global=color_frequency(img);
fre_local=color_frequency(block);
differences=fre_global-fre_local;
diff_square=sum(differences.*differences);
end

function smo=image_smooth(block)%平滑度特征
[r,c]=size(block);
diff_x2=sum(sum(abs(block(1:end-1,:)-block(2:end,:))));
diff_x1=sum(sum(abs(block(:,1:end-1)-block(:,2:end))));
diff_x3=sum(sum(abs(block(1:end-1,1:end-1)-block(2:end,2:end))));
diff_x4=sum(sum(abs(block(2:end,1:end-1)-block(1:end-1,2:end))));
smo=(diff_x1+diff_x2+diff_x3+diff_x4)/(r*c);
end

function similarity=sc_match(block,secret)%sc 匹配度
secret_bit=uint8(reshape(dec2bin(secret,8),1,[]))-'0';
lsb=reshape(bitand(block,1),1,[]);
lsb_embed=lsb(1,1:length(secret_bit));
difference=double(lsb_embed)-double(secret_bit);
similarity=1-sum(abs(difference))/length(secret_bit);
end
```

致　　谢

这篇文章的完成除了自我奋斗以外，也考虑了很多相关人员的建议，在此毕业离别之际我想感谢他们的无私帮助。

首先我要感谢我的毕业设计指导老师孙伟峰老师，在我的毕设选题和完成的过程中提供了很多建议，并具体帮助我规划各个时间段的具体任务，鼓励我多做出尝试。接下来我将感谢各位评阅我的论文和参加我的答辩的老师们，感谢你们的不吝赐教，让我可以客观地评价自己的工作。同时，我也衷心感谢四年大学生涯中或多或少帮助过我的所有老师们。

我能如期完成实验数据的收集也归功于大连理工大学软件学院创新实践中心树模组为我提供了搭建好环境的计算机，对此我表示非常感谢，希望他们能继续为学院的创新工作助力。

除此之外，我的同学们在我毕设期间经常与我交流激动，给了我很多启发和激励，我很庆幸在我的大学生涯能拥有这些同学们，感谢他们给我带来的快乐和动力。

当然，完成这些工作的前提主要是我站在了巨人的肩膀上，很惭愧，我只做了一些微小的工作。所以，我在此感谢计算机界和其他领域的前辈为我提供的平台，尤其是宾汉姆顿大学的 Fridrich 教授，为本文最后实验结果检测部分提供了 SPA 方法的 MATLAB 源代码。同时，我的工作也得益于软件，工具箱，封包构成的平台的成熟发展，感谢他们做出的工具极大的提高了实验的效率，特别是国立台湾大学的团队开发的 LIBSVM。在此，我对上文提到的所有参考文献的作者表示衷心的感谢。