# Android应用相似性检测研究

## 绪论

### 1.1研究背景

随着移动互联网的发展，智能手机已经成为人们生活中不可或缺的部分。智能手机迅速普及有诸多方面的原因，例如便携性，但最重要的是 Android 平台上各种各样性能丰富的智能手机应用程序。这些性能丰富的应用拓展了智能手机的功能，使得用户能够随时随地浏览网页、在线交友、休闲娱乐以及保持通讯，而不是只能打电话和发短信。与传统手机相比较，各种不同功能的应用为智能手机带来的这种便利而充满趣味的使用体验是前所未有的。

为了让智能手机用户能够方便地浏览和安装智能手机应用，平台供应商提供了集中式的应用市场，包括苹果公司的App Store和谷歌公司的Google Play。应用开发者向应用市场提交自己的应用，使其开发的应用提供给成千上万的智能手机用户。平台所有者也能够更有效地控制应用的质量、阻挡恶意应用，从而保护用户的正当权益。与此同时，人们也创造了各种不同用途（比如为了满足区域或本地化需求）的第三方应用市场。例如Cydia和Amazon AppStore分别容纳了成千上万的 iPhone和Android 应用。在中国，还有着众多的第三方应用市场。例如安智市场、机锋市场、木蚂蚁应用市场、安极市场等等。

然而，在这些应用市场上的移动应用程序中，存在数量众多的对合法应用程序肆意进行修改，并用新的签名密钥进行重新打包并重新发布应用的现象。作为一种技术性方法，应用重新打包技术本应该用于良好意图。例如，Aurasium就使用应用重新打包技术来拦截应用与其底层操作系统之间的互动，其目的是为了强制执行用户为该应用指定的安全措施。而ADAM则使用应用重新打包技术对恶意软件样本进行微调，其目的是为了对各种各样的Android 杀毒工具进行压力测试。然而，更常见的情况是，应用重新打包技术被用于恶意目的，由此给应用开发者、手机用户、市场运营商乃至整个Android生态系统都造成不容忽视的风险。例如，恶意软件作者可能会将具有破坏性的载体程序附加到合法的应用程序里面，然后在多个应用市场中通过广告推送的形式来感染不知情的用户，或是，将广告代码植入热门应用程序中，以图通过应用原作者获取广告收入。

在国内，这种情况更加严重。由于Android平台比苹果等手机平台更为开放，用户不仅可以通过谷歌官方应用市场下载手机软件，更多的是通过第三方应用市场，还有的则是手机预装，甚至通过链接和二维码直接下载。国内非官方渠道下载更为普遍，约80%的Android 应用均是从应用宝、百度应用、91、机锋市场等第三方应用市场下载。

在Android恶意软件猖獗的第三方应用市场环境下，用户受到安全威胁和损害的比例大大增加。金山手机毒霸统计显示，有个别应用商店中检测出的恶意软件比例接近 10%，也即是说，用户在此商店中每下载 10个应用就平均有 1 个为恶意软件，面临资费和个人隐私的极大威胁。

业内人士分析，第三方应用商店的不作为主要有三大原因：第一，专业的手机安全技术水平有限，无法发现恶意代码；第二，对安全检测根本不重视，无专业专职人员配备；第三，应用商店最主要的营收来源就是各应用的推广分发费用，应用有下载量就有收入，不关心它是否安全，是否偷取用户资费和隐私。

面临应用重新打包技术肆意泛滥的威胁，最主要的挑战是如何系统地快速准确地从海量的 Android 应用中识别这些经过重新打包的应用？是否能够快速而可度量地检测这些经过重新打包的应用从而对每天提交的大量应用进行有效的管理？能否帮助用户有效判断下载的应用程序是否合法、是否包含恶意负载？

针对这些挑战，本文提出一种Android相似性检测技术，可以快速有效地从海量应用中识别出重新打包的应用，保护应用开发者，防止他们的知识产权受到侵犯。此外，市场运营商能够利用该技术确保他们的应用市场不会充斥着经过重新打包的应用程序。

### 1.2研究现状

针对目前嵌入已知恶意代码和重打包应用，伪造成钓鱼应用带来的安全问题，国内外研究学者提出了一系列解决方法，其中通过检测应用相似性来识别嵌入已知恶意代码应用和重打包应用是目前研究的主流方法．

传统的程序代码的相似度检测的研究在国外起步较早，在20世纪70年代就有很多学者开始研究计算代码相似度的方法。1976 年，Purdue 大学的Ottenstein第一次提出使用属性计数法计算代码相似度的方法，并构造了一个能够计算Fortran 程序相似度的系统，用 Halstead提出的程序度量方法进行程序相似度计算。此方法的核心思想是提取代码的特征属性，Halstead 系统中提取了四个基于长度的软件科学参数，而后 Grier 和 Faidhi 分别使用了 20 和 24种参数来检测代码的相似性。这种方法的优点是效率高，但缺点也很明显，就是无法检测出对程序片段的抄袭，并且该方法之后的研究方向走入了死胡同，仅靠增加属性度量是无法提高系统的检测效果的。之后的学者就从属性计数的方法向基于结构度量的相似性检测方法发展。

DroidMOSS是针对 Android 应用程序重新包装威胁的第一个系统化研究，其基于MOSS。 DroidMOSS 在两个关键方面不同于 MOSS：第一，DroidMOSS 基于Dalvik字节码级别而非源码级别；第二，两个系统都要求使用一个滑动窗口的生成指纹。MOSS使用 k-gram直接构成指纹，而DroidMOSS 计算散列值比较重新包装后的应用的变化。DNADroid采用程序依赖图（PDG）来抽取 Android应用程序特征，通过比较在应用程序对方法之间的程序依赖图来判断 Android应用的相似性。 Potharaju等人使用不同的语法指纹方案构建应用程序指纹，并能模糊处理不同指纹来判断应用相似性。Juxtapp收集静态代码特点和构建位向量，提高 Android应用比较的效率，它也支持增量更新和分布式分析。上述研究使用相似性度量，通过两两比较判定Android程序是否经过重新包装，具有可扩展性问题。

2012年，Saung Li提出采用文件目录结构对应用相似性进行评估，用树结构表示应用的目录结构，计算数之间的编辑距离，由此得到应用之间的距离，该方法可以检测重打包应用、嵌入已知恶意代码应用。但是，该方法的成功得益于恶意开发者基本不改变文件目录结构，如果插入无用文件或者改变文件目录名，则会导致该方法失败。

2014年，焦四辈等提出一种基于应用内特定文件的内容特征计算应用相似性，该方法不受代码混淆的影响，且能有效抵抗文件混淆带来的干扰。但该方法选取的仅仅是有限的资源文件，包括图片文件，音频文件和布局文件。这些文件难以全面描述应用的特征，导致误报率较高

对于Android应用的相似性检测，上述研究存在着以下问题：一是特征提取，上述特征模型难以描述Android 代码的特征，提取算法过于复杂，同时针对采用混淆，变形等技术处理过的应用，检出率差。二是处理效率，Android应用具有海量，发布途径分散等特点，复杂的特征提取与比较算法，特别是基于动态分析的特征提取技术，由于动态分析特征集的不确定性或对于状态改变和行为特征的相似性度量方法过于复杂，使应用的相似性检测因为时间复杂度失去使用价值。三是误报率高，为了平衡检测时间与海量应用，一些技术采用粗粒度的特征提取与比较算法，具有较高的相似性误报率，在实践上还需要大量的人工辅助检测。

### 1.3研究内容

Android应用以压缩文件的形式存在，内部已文件夹的形式存放可执行字节码文件、证书文件和资源文件，其中可执行dalvik字节码存储在class.dex文件中，证书文件是应用的签名文件，位于META-INF目录中；资源文件包括数据库文件、函数库文件，XML文件、图片文件等，其中class.dex文件又可以解压为包含class文件的文件目录。

通过研究压缩文件中的有用信息来比较不同应用的相似性，我们以class类文件的目录结构和界面截图两类信息作为依据来进行判断。理由如下：

（1）由于代码混淆技术主要针对可执行代码，而Android应用程序内类文件的目录结构难以进行混淆，因此基于类文件的目录结构的相似性研究成为方向。

（2）由于钓鱼应用是一种安装在用户移动终端上的恶意应用程序，其通过模拟正规应用的界面，诱导用户输入私密信息，包括自己的账户，密码，再传递到自己的服务器，从而盗取用户信息。对于用户而言，是否信任这个应用，很大程度上取决于该应用的界面与正规应用的界面十分相似或完全一样。因此我们的研究基于视觉相似性的检测来实施。

对于具体的研究过程，步骤如下：

（1）从官方应用网站下载批量正规APK，构成应用库。

（2）对于其中的每个应用，操作如下：

（3）通过解压得到资源文件layout文件夹，drawable文件夹，classes.dex文件。

（4）通过dex2jar将classes.dex转化为类文件目录。

（5）将类文件目录结构通过弱哈希运算，转化为二叉树，再先序遍历，得到字符串，该字符串可以保证插入的垃圾类不会在字符串分散。

（6）对于该字符串进行模糊哈希运算得到类目录结构签名。

（7）将图片和布局资源文件通过动态工具Monkeyrunner得到该应用的界面截图，通过改进感知哈希算法得到界面签名。

（8）将两类签名构成签名库，对于未知的APK，得到签名，与两类签名进行比较相似性，大于阈值，则表示与该应用相似，为该应用的钓鱼应用，否者为非相似应用。

### 1.4论文组织结构

本文内容共分为5章，每一章节的大概内容简述如下：

第一章，绪论。在这一章中，我们介绍了选择这个课题的背景，分析了研究意义和目的。介绍了Android应用相似性的研究现状并分析了本研究的主要工作和内容。最后简单说明了本文的组织结构。

第二章，基于类目录文件的相似性技术研究。

## 二、基于类目录文件结构的相似性研究

### 2.1 APK的反编译

#### 2.2.1 APK

APK是安卓安装包（AndroidPackage）的缩写，它的文件格式类似塞班的Sis或Sisxp[14]。APK文件安装十分方便，直接通过安卓模拟器或在安卓手机上直接打开就能自行安装。APK文件是将安卓SDK编译的工程打包，变成一个安装的程序文件，后缀格式直接就是.apk，究其本质其实是个压缩文件zip格式，只不过后缀名被修改。既然作为一个压缩文件，它就可以被解压得到Dex文件，Dex文件是非常重要关键的，这一点在下文将会提及，Dex文件是DalvikVM executes的简称，是安卓Dalvik执行程序，由Dalvik的字节码编写而成而非Java Me字节码。

APK的文件结构主要由5类文件构成[15]，其中META-INF目录存放了签名信息，它是APK完整性的保证；res目录存放了资源文件；AndroidManifest.xml是文件的全局配置文件；classes.dex的内容是Dalvik字节码，是程序执行时所必要的代码文件；resources.arsc则是编译之后的二进制资源文件。

#### 2.2.2 反编译

APK的反编译就是得到一个APK[16]，对它进行解压，对源文件和资源进行处理再进行编译，以得到开发APK时的源代码，这样一来可以对APK自行设置，包括一些自定义的界面或是APK的汉化，当然作为一个开发者，也可以参考APK的代码，为自己的开发编写产生灵感。

最经典的反编译工具是apktool，不过它的操作需要调用cmd指令，于是在apktool的基础上出现了一些有可视化界面的反编译工具。

一个完整的反编译过程是将APK应用程序的后缀改回Zip，解压得到classes.dex文件，利用dex2jar将dex文件转为jar包，最后利用jd-gui工具反编译jar文件 。这样就可以得到APK开发者在开发应用程序时的源代码了，对于那些基于特征代码的恶意软件检测方案，特征代码值就可以从中搜寻整理得出。然而本文的安卓应用相似性的检测是通过文件的目录结构来做出判断，所以需要将jar包通过Unzip再次解压，得到APK的classes.dex的完整结构。

### 2.2 特征信息的提取

#### 2.2.1 树状目录转化为线性结构

经过上一步的解压转换，已经得到了classes.dex文件中的类文件目录，这些文件的结构也清晰可见，然而光靠眼睛看到文件结构是不行的，人眼是无法对复杂的文件结构进行分析的，必须将其转换为计算机可以理解的形式比如字节码。

图2-9 文件遍历方式

于是，编写程序遍历整个文件目录，取所有文件名及文件夹名的首尾字符，组成一个长字符串作为该APK类文件目录的特征。

程序在遍历文件夹时，先判断当前对象是否为文件夹，若是文件夹，则继续遍历文件夹中的内容并插入到遍历的队列中，若该文件夹中还存在文件夹，也会继续打开次文件夹获取其中的内容；若当前对象不是文件夹，则将当前内容插入队列，并获取下一个对象。所以图2-7中的遍历顺序为A-B-H-C-D-G-I-E-F。

#### 2.2.2 文件名的弱哈希

由于不同的类文件的名称存在长短不一的问题，对于组成的线性字符串结构，则文件名长的类影响大，文件名短的类影响小，为了解决这个问题，需要将所有的文件名归一化，这里采用hash的方式解决。

对于常见的散列算法，如MD5，SHA-1的输出大小都是128bits，160bits，这会使得得到的字符串的长度很大，不利于后期的比较处理

##### 2.2.2.1 散列函数

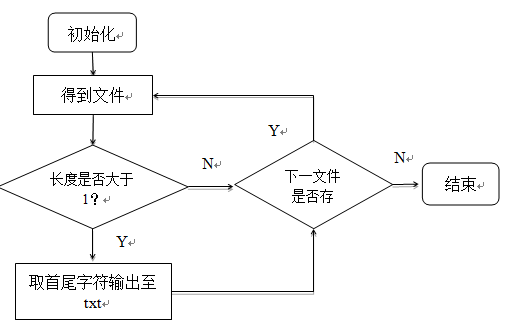
所有散列函数都有如下一个基本特性：如果两个散列值是不相同的（根据同一函数），那么这两个散列值的原始输入也是不相同的。这个特性是散列函数具有[确定性](https://zh.wikipedia.org/wiki/%E7%A1%AE%E5%AE%9A%E6%80%A7)的结果，具有这种性质的散列函数称为单向散列函数。但另一方面，散列函数的输入和输出不是唯一对应关系的，如果两个散列值相同，两个输入值很可能是相同的，但也可能不同，这种情况称为“哈希碰撞”，这通常是两个不同长度的输入值，刻意计算出相同的输出值。输入一些数据计算出散列值，然后部分改变输入值，一个具有强混淆特性的散列函数会产生一个完全不同的散列值。

典型的散列函数都有非常大的[定义域](https://zh.wikipedia.org/wiki/%E5%AE%9A%E4%B9%89%E5%9F%9F)，比如[SHA-2](https://zh.wikipedia.org/wiki/SHA-2)最高接受(264-1)/8长度的[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82)[字符串](https://zh.wikipedia.org/wiki/%E5%AD%97%E7%AC%A6%E4%B8%B2)。同时散列函数一定有着有限的[值域](https://zh.wikipedia.org/wiki/%E5%80%BC%E5%9F%9F)，比如固定长度的[比特](https://zh.wikipedia.org/wiki/%E6%AF%94%E7%89%B9)串。在某些情况下，散列函数可以设计成具有相同大小的定义域和值域间的[单射](https://zh.wikipedia.org/wiki/%E5%8D%95%E5%B0%84)。散列函数必须不具有可逆性。

而对于一个文件夹的目录结构，里面存在的文件夹的名称可能的值域空间大小为几千或几万个。对于散列值得要求是必须为字符串，每个字符串的空间为52，则三个字符串的值域空间（52^3-1）的大小基本可以满足要求

通过上述弱哈希运算，得到文件及文件夹名的首尾字符，排成一个长字符串作为APK的特征字符串，程序的具体工作流程见下图2-10。要特地说明的是，如果文件或文件夹名的字符长度为1，则将其忽略不添入特征字符串。原因是许多Android应用程序的开发者会对自己的类名进行混淆操作，批量改为类似于a.class、b.class、c.class这样的名字，所以这种文件名对于APK的特征采集是没有帮助的，因此可以考虑将这种没有特征的类从字符串中删除

接下来要做的就是如何对这一长字符串进行相似性的比较。



### 2.3 相似性比较

#### 2.3.1 编辑距离

莱文斯坦距离，又称Levenshtein距离，是[编辑距离](https://zh.wikipedia.org/wiki/%E7%B7%A8%E8%BC%AF%E8%B7%9D%E9%9B%A2)（[edit distance](https://en.wikipedia.org/wiki/edit_distance)）的一种。指两个[字串](https://zh.wikipedia.org/wiki/%E5%AD%97%E4%B8%B2)之间，由一个转成另一个所需的最少编辑操作次数。许可的编辑操作包括将一个字符替换成另一个字符，插入一个[字符](https://zh.wikipedia.org/wiki/%E5%AD%97%E7%AC%A6)，删除一个字符。

例如将kitten一字转成sitting：

sitten （k→s）

sittin （e→i）

sitting （→g）

[俄罗斯](https://zh.wikipedia.org/wiki/%E4%BF%84%E7%BE%85%E6%96%AF)科学家[弗拉基米尔·莱文斯坦](https://zh.wikipedia.org/w/index.php?title=%E5%BC%97%E6%8B%89%E5%9F%BA%E7%B1%B3%E5%B0%94%C2%B7%E8%8E%B1%E6%96%87%E6%96%AF%E5%9D%A6&action=edit&redlink=1)在1965年提出这个概念。

#### 2.3.2 相似度公式

在上一节中，可以通过编辑距离算法得到两个字符串之间的编辑距离，然而编辑距离只能说明两个字符串之间需要通过几次变换可以变成相同的字符串，所以必须根据这个编辑距离找一个公式，得到一个能够说明两个字符串相似度的数值。

可以举两个确切的例子来得到这个公式。

若两个字符串分别为和；

若两个字符串分别为和

和的编辑距离为3，和的编辑距离为5，而显然可以观察出上两个字符串的相似度大于百分之50，下两个字符串的相似度小于百分之50,，同时相似度与两个字符串的长度也是有关系的，于是可以得到两者的相似度公式 ，其中为两字符串的编辑距离，与是两字符串的各自长度。

### 2.4 小结

基于类目录文件结构的方法进行相似性的分析，基于的是静态的分析方法，优点是简单方便，将目标应用的特征字符串与库中的字符串进行编辑距离的匹配，快速找到相似度高的应用，但是基于一个特征的判别方法还是过于单调简单，而且如果被恶意开发者利用，也会存在攻破的可能，因此接下来的章节会基于截图的相似性来进行相似性的判断

## 基于界面相似性的研究

对于相似应用，一个主要的特征，就是构造与目标应用非常相似的登陆，支付等关键界面，诱骗用户输入敏感信息，从而达到恶意开发者的目的。

基于这种相似性，提出了通过动态的截取被检测界面的人际交互界面，利用图像哈希感知算法计算与目标应用的相似程度。

### 3.1 图像感知哈希技术

#### 3.1.1 感知哈希的基本概念（王亚男）

感知哈希的研究起源于数字水印技术，借鉴了传统密码学哈希和多媒体认证的相关领域的概念和理论，通过对图像感知信息的简短摘要和基于摘要的匹配，来支持图像内容的识别和认证。这些消息摘要最初被一些学者称作数字指纹，因为它体现了载体信息的唯一特征，可以像人的指纹一样用于标识载体信息的身份；也有一些学者称这种摘要为鲁棒哈希，软哈希。2001年，Ton Kalker提出了感知哈希的概念，他通过“感知”一词来强调感知哈希关注的事感知的相似性，于是感知哈希一词出现，并被很多学者引用。但是直到今天，这些摘要仍然没有确定的称呼（在本文中统称为感知哈希）。

感知哈希试图通过解决多媒体感知特性与传统哈希函数的矛盾，为多媒体内容认证、识别、检索等信息内容服务提供安全可靠的技术支撑。如今已经有许多研究机构和公司致力于对感知哈希算法的研究和应用。Philips工作室主要研究感知哈希算法，并试图提出感知哈希算法的性能标准；Microsoft和Fraunhauser也在感知哈希算法的研究和应用上做了大量工作；Auditude和Relatable则将汉纸哈希算法应用于广告投放监视，盗版跟踪和网络多媒体内容服务等方面，开发出了各种不同的应用模式。随着感知哈希的发展，越来越多的公司和研究结构都对感知哈希的前景看好。研究感知哈希，不仅对多媒体技术的发展有着重要而深远的意义，而且具有关阔的应用价值和经济意义。

#### 3.1.2 感知哈希的定义

对于感知哈希的定义，在很多文献中都有提及，归结起来可以概括为：感知哈希是一种基于媒体作品的可感知特征的内容摘要提取技术，提取的摘要对盖面媒体感知内容的操作敏感而对不改变媒体感知内容的处理鲁棒。同样，图像感知哈希就是通过提取图像I的感知特征，并将其映射成长度较小的比特序列h为消息摘要的一种技术。所提取的感知特征，既能对改变图像感知内容的操作敏感，有能够对不改变图像内容的操作鲁棒。

在文献中，Ton Kalker指出了感知哈希是这样的一个函数：“它能（1）将大数据量的多媒体对象映射为长度较小的序列；（2）将感知相近的媒体对象映射为数学相近的哈希值。”因此，我们可以将感知哈希定义为如下的单向函数PH:



其中，h表示有媒体对象I提取得到的感知哈希值，PH为感知哈希函数。在检测中，只需要通过计算新的感知哈希值h’=PH(I’),并验证h’是否等于h，或两者的感知距离在规定阈值之内即可。

感知距离是感知内容相似性的度量，它可以通过计算汉明距、欧式距离或者其他任何可能的距离计算方式求得。现有的大多数图像感知哈希算法，都采用汉明距来计算感知距离，从而判断图片的真实性。

为了实现感知哈希的安全性，很多感知哈希算法引入了密钥来提供随机性，因此感知哈希也可以定位为：



即提供了两个输出参数的单向函数。其中KEY表示密钥。

本文研究的内容对于安全性没有太多的要求，更加关注的是运行的效率，因此采用第一种感知哈希函数的定义方式。

#### 3.1.3 感知哈希的性质

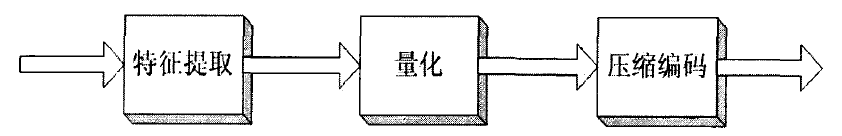
与传统密码学中的哈希函数相同，图像感知哈希同样具备摘要性、单向性和抗碰撞性；同时，由于感知哈希起源于数字水印，因此也继承了数字水印对图像感知内容鲁棒的特点。所谓鲁棒，就是指只要图像感知内容不变，原图像的感知哈希值应该与目标感知哈希值相同，或两者的感知距离在某一个指定的阈值内。另外，为了保证感知哈希在图像认证等领域的安全性，很多算法也引入了密钥类实现这种安全性。

下面对感知哈希函数的PH的性质做个总结：设是三个多媒体信息（假设为三幅图像），分别用三个密钥生成感知哈希，对应的感知哈希分别为为感知距离，标识哈希值间的感知差异。当感知距离大于等于感知阈值时，则判定两个哈希值所对应的媒体信息的感知内容不同。感知哈希函数PH应满足一下基本性质：

1. 感知鲁棒性：感知内容相同或相近的媒体对象应该映射得到相同或者相近的感知哈希值。即：如果 ，那么 。
2. 摘要性：感知哈希值所占用的存储空间尽可能的小，即感知哈希值长度L要尽量的简短。
3. 安全性
4. 单向性：可以根据媒体对象计算出感知哈希值，但是根据感知哈希值，无法推算出媒体的有效信息。即：已知 但是.
5. 区分性：又称作抗碰撞性或者两两独立性，是指两个感知内容不同的媒体对象所计算出的感知哈希值之间的感知距离，应大于TP。即：如果I1与I2是两幅完全不同的图像，那么 。
6. 随机性：相同的媒体对象，用不同的密钥应产生不同的感知哈希值。在密钥未知的情况下，感知哈希值应该不容易被伪造或被估计。即：如果，且，那么。

#### 3.1.4 感知哈希的基本流程（张鹤超）

现有的感知图像哈希算法的基本流程可以归纳为三步，如图所示：



其中，第一步：从图像中提取特征向量；第二步：量化提取出的特征向量的到二进制hash串；第三步：压缩二进制hash串得到最终的哈希序列。在特征提取阶段，一副二维的图像被映射为一维的特征向量，该特征向量必须是从图像中提取的感知特征，也就是说对于人类视觉系统来说，相同的图像应该有在某种距离度量下相近的特征向量。反之，两幅截然不同的图像在该种距离度量下应该具有较大的距离。第二步的输出结果通常称为中间哈希值，目前存在的文献主要都是研究中间哈希值的生成，二中间哈希值的进一步压缩可以采用目前广泛应用的各种压缩编码方法。

#### 3.1.5 常见图像感知哈希算法

感知图像哈希算法是近年来兴起的热门研究点，与传统的图像哈希算法不同，感知哈希算法是一种基于图像内容哈希算法． 其以图像的视觉特征为基础，通过散列函数生成图像的固定长度或不定长度的一串“指纹”用于比较． 由于感知哈希算法具有良好的实时性以及适应大数据量等优点，其在图像检索、图像认证等领域已有初步的应用．

用于进行感知哈希的图像特征可分为全局特征和局部特征两种．全局特征包括灰度特征、频率特性、颜色特征、纹理特征和线特征等，哈希算法主要有阈值控制的灰度哈希算法、基于离散余弦变换的频率阈值哈希算法以及与其它特征结合的多维全局特征哈希算法，但由于全局特征过于单一，往往对光照、旋转、噪声等局部特征不敏感，使得该哈希算法抗攻击能力不够理想． 局部特征主要包括近年来来广泛使用的Harris、SIFT 以及SUＲF 特征． SIFT，尺度不变特征转换(Scale － invariant Feature Transform)是由David Lowe 在1999 年提出，其优点为具有尺度、光照以及旋转不变性，但由于其具有较高的算法复杂度，在实时系统中并不能得到很好的应用。

##### 3.1.5.1 基于DCT的图像哈希算法

离散余弦变换(DCT)是与傅里叶相关的一种变换，类似于傅里叶变换但是只用实数． 对于图像其DCT 变换公式如下：



其中f 为图像像素点，F 为 DCT 阈矩阵，C为余弦系数矩阵． 由于DCT 具有接近K － L 变换的效果，又具有快速算法，故常用于图像压缩以及编码． DCT 的快速算法一般分为两种:直接算法以及间接算法． 直接算法主要包括DCT 变换的矩阵分解和递归算法;间接算法主要利用DCT与DFT(离散傅里叶变换)的关系，通过DFT 的快速算法FFT实现，本文采用后者． 该算法流程如下:

a) 对图像像素进行抽样，将图像缩小为32 × 32 像素．

b) 对彩色图像进行灰度变化，转化为灰度图像．

c) 计算该图像的DCT 变换，得到32 × 32 变换系数矩阵．

d) 保留矩阵中左上角8 × 8 系数子矩阵．

e) 计算8 × 8 子矩阵中系数的均值，计为m．

f) 按照从左到右从上到下的顺序遍历8 × 8 子矩阵，哈

希规则如下:



其中x 为8 × 8 子矩阵中的元素即DCT 变换系数，h( x)为该元素对应的哈希码． 故按照如上步骤可以得到64 位的哈希编码．

g) 将64 位2 进制编码转化为16 位16 进制编码，该编码即为该图像的哈希编码．

按照如上步骤得到的16 位哈希码主要是图像的低频信息，保留了图像概貌特征． 通过计算两幅图像该编码的Hamming距离，可得到两幅图像的相似度． 但是该算法滤除了图像的高频信息，故失去了对图像局部变化的敏感性． 下面的

SUＲF 算子正是为这一问题作为补充．

##### 3.1.5.2 sift（维基百科）

尺度不变特征转换(Scale-invariant feature transform 或 SIFT)是一种[机器视觉](https://zh.wikipedia.org/wiki/%E6%9C%BA%E5%99%A8%E8%A7%86%E8%A7%89)的算法用来侦测与描述影像中的局部性特征，它在空间尺度中寻找极值点，并提取出其位置、尺度、旋转不变数，此算法由[David Lowe](https://zh.wikipedia.org/w/index.php?title=David_Lowe_(computer_scientist)&action=edit&redlink=1) 在1999年所发表，2004年完善总结。  后续的论文中也有许多基于 SIFT 改进的论文，例如 [SURF](https://zh.wikipedia.org/wiki/%E5%8A%A0%E9%80%9F%E7%A8%B3%E5%81%A5%E7%89%B9%E5%BE%81) 将 SIFT 的许多过程近似，达到加速的效果；PCA-SIFT利用[主成分分析](https://zh.wikipedia.org/wiki/%E4%B8%BB%E6%88%90%E5%88%86%E5%88%86%E6%9E%90)降低描述子的维度，减少内存的使用并加快配对速度。

其应用范围包含[物体辨识](https://zh.wikipedia.org/w/index.php?title=%E7%89%A9%E9%AB%94%E8%BE%A8%E8%AD%98&action=edit&redlink=1)、机器人地图感知与导航、影像缝合、3D模型建立、手势辨识、影像追踪和动作比对。

此算法有其专利，专利拥有者为 [英属哥伦比亚大学](https://zh.wikipedia.org/wiki/%E8%8B%B1%E5%B1%AC%E5%93%A5%E5%80%AB%E6%AF%94%E4%BA%9E%E5%A4%A7%E5%AD%B8)。

SIFT能够找出独特的关键点，此关键点不会受移动、转动、缩放、[仿射变换](https://zh.wikipedia.org/wiki/%E4%BB%BF%E5%B0%84%E5%8F%98%E6%8D%A2)、亮度等外在因素的影响而改变其特性。因此能够有效应用在物体辨识上，其步骤包含：

* 输入侦测物，并执行SIFT算法找出输入影像中不变的特征。
* 这些特征会与SIFT特征数据库作描述子配对，配对将透过[最近邻居法](https://zh.wikipedia.org/wiki/%E6%9C%80%E8%BF%91%E9%84%B0%E5%B1%85%E6%B3%95)来完成。为了增加可信度，将会移除最近距离与第二近距离大于0.8的配对，这将能够有效移除背景造成的错误配对。此外为了提升最近邻居法的计算速度，应用best-bin-first算法能够有够高概率找出最近的距离并加快搜寻速度。
* 去除错误的配对后，仍有不同物体的成功配对，因此需将成功的配对加以分类，将相同物体的分类分在一起并移除不同物体的分类，因此应用了[霍夫变换](https://zh.wikipedia.org/wiki/%E9%9C%8D%E5%A4%AB%E8%AE%8A%E6%8F%9B)。如此一来变能够辨认拥有相同角度的特征点，这些特征点有很高概率是同一个物件的，因此能够分出各个特征群。
* 对于每个被挑选出来的特征群，使用最小平方法求得输入影像与训练资料间最佳的仿射变换参数。运用此参数对各个特征点作比对，调整参数直到特征点皆能正确仿射没有错误发生为止。

##### 3.1.5.3 surf

SURF (Speeded Up Robust Features, 加速稳健特征) 是一个稳健的图像识别和描述算法，首先于2006年发表在ECCV大会上。这个算法可被用于[计算机视觉](https://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA%E8%A7%86%E8%A7%89)任务，如[物件识别](https://zh.wikipedia.org/w/index.php?title=%E7%89%A9%E4%BB%B6%E8%AF%86%E5%88%AB&action=edit&redlink=1)和[3D重构](https://zh.wikipedia.org/w/index.php?title=3D%E9%87%8D%E6%9E%84&action=edit&redlink=1)。他部分的灵感来自于[SIFT](https://zh.wikipedia.org/wiki/%E5%B0%BA%E5%BA%A6%E4%B8%8D%E5%8F%98%E7%89%B9%E5%BE%81%E5%8F%98%E6%8D%A2) 算法。SURF标准的版本比SIFT要快数倍，并且其作者声称在不同图像变换方面比SIFT更加稳健。

SURF使用[海森矩阵](https://zh.wikipedia.org/wiki/%E6%B5%B7%E6%A3%AE%E7%9F%A9%E9%98%B5)的行列式值作特征点侦测并用[积分图](https://zh.wikipedia.org/wiki/%E7%A7%AF%E5%88%86%E5%9B%BE)加速运算；SURF 的描述子基于 [2D 离散小波变换](https://zh.wikipedia.org/w/index.php?title=Haar-like_features&action=edit&redlink=1) 响应并且有效地利用了[积分图](https://zh.wikipedia.org/wiki/%E7%A7%AF%E5%88%86%E5%9B%BE)。

SURF算法的概念及步骤均建立在SIFT之上，但详细的流程略有不同。SURF算法包含以下三个步骤:特征点侦测、特征邻近描述、描述子配对。后文会对surf算法进行详细介绍

### 3.2 Android登陆界面特征分析

在手机平台上，用户登录界面是手机应用展示给用户的门户界面。随着应用开发者越来越重视用户体验，多数手机应用登录界面的设计非常简洁，以实现简单、快捷操作。图 1 列出了几款应用的登 录界面，可以发现这些登录界面的设计有着较为一 致的特点。首先，由于登录界面功能单一，不需要 用户进行复杂操作，因此控件数量少，而且界面布 局简单;其次，作为应用的门户，登录界面上都会 放置一个醒目的应用图标，对于设计简单的登录界 面来说，界面上的应用图标具有很强的特征性，用 户在很大程度上是通过应用图标来辨识该界面是属 于哪个应用;此外，登录界面背景图片设置简洁， 颜色单一，这使得界面上的控件和图标轮廓清晰， 特征明显。

登录界面的简单设计使得攻击者容易通过伪造 界面来进行钓鱼攻击。攻击者伪造登录界面的理想 情况是钓鱼界面与目标界面完全相同。在这种情况 下可以将界面图像完全相同作为标准，通过进行界 面图像比对实现对钓鱼登录界面的检测。而现实情 况是伪造的登录界面很难与目标登录界面完全相 同，如果仅以界面图像是否完全相同作为检测标准， 检测效果将会大打折扣。即使攻击者可以伪造出与 目标登录界面完全相同的钓鱼界面，在面对上述检 测时，只需使钓鱼界面与目标界面高度相似，就会 绕开检测。而且，用户也很难识别出高度相似的登 录界面之间的差异，使得该类钓鱼登录界面容易获 取用户的信任，导致隐私信息泄露。图2 a是目标 登录界面，图 2 b、c 是伪造的登录界面，三幅图存 在差异，但人眼很难识别出。

对于上述情况，本文提出的方法是:描述当前 登录界面与目标登录界面的相似度，如果相似度在 预设的范围内，则认为该登录界面与目标界面相似， 并判断当前登录界面是恶意软件的钓鱼界面。

本文提出通过对当前界面图像和目标界面图像 进行特征检测和匹配来描述两个界面之间的相似 度。对图像进行特征检测和匹配是计算机视觉应用 的一个重要组成部分。在一幅图像中，存在一些关 键点可以反映图像的局部特征，通过描述这些关键 点并进行匹配，就可以依据匹配结果来判断图片的 相似性。关键点的检测和匹配流水线可以分为三个 阶段[3]:首先是特征提取阶段，需要在图像中找到 适合的、稳定的特征点;第二步是进行特征描述， 把检测到的关键点周围区域转化为一个紧凑和稳定 的描述子;第三步是进行特征匹配，将待比较的图 像特征描述子进行匹配，在匹配的过程中，匹配效 果不但取决于特征描述算法，还取决于匹配策略， 简单的蛮力匹配策略会使结果存在误报和漏报现 象，因此需对匹配策略进行优化，筛选合适的匹配， 再根据优化后的匹配结果计算界面相似度。

针对手机应用的登录界面，通过对界面图像进行特征检测和匹配来描述相似度是非常合适的。原因在于，登录界面布局简洁，背景颜色单一，控件边沿轮清晰，这使得大多数特征点都分布在控件以及界面的应用图标区域，特征点的集中有利于检测时钓鱼界面与目标界面的特征匹配。但是，由于待检测界面与目标界面并非完全一致，存在控件尺寸较目标界面有微调，控件位置有偏移等情况。此外，屏幕视图旋转为横屏也会对检测带来干扰。因此，所选择的特征点检测算法需要具有尺度缩放和旋转、平移保持不变的特性。这样不但可以提高特征检测的稳定性和准确性，同时也使得攻击者很难通过微调界面的方式来绕过检测。

### 3.3 SURF算法

SURF (Speeded Up Robust Features)也是一种类似于SIFT的兴趣点检测及描述子[算法](http://lib.csdn.net/base/datastructure)。其通过Hessian矩阵的行列式来确定兴趣点位置，再根据兴趣点邻域点的Haar小波响应来确定描述子，其描述子大小只有64维，是一种非常优秀的兴趣点检测算法。

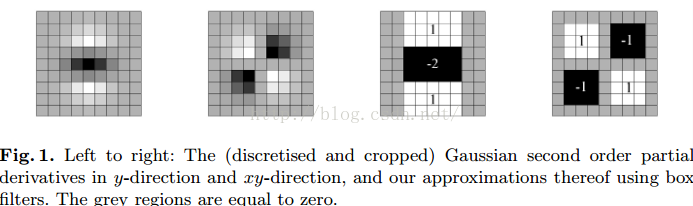
#### 3.3.1 ****FAST-Hessian检测****

首先同SIFT方法一样，SURF也必须考虑如何确定兴趣点位置，不过SIFT采用是DOG来代替LOG算子，找到其在尺度和图像内局部极值视为特征点，而SURF方法是基于Hessian矩阵的，而它通过积分图像极大地减少运算时间，并称之为FAST-Hessian。（这里提一下，SIFT通过DOG来近似LOG，也实际上相当于计算Laplacian，即可以视为Hessian矩阵的迹，而SURF则利用的近似Hessian矩阵的行列式）

首先我们考虑一个Hessian矩阵：



这里的Lxx是指图像经过高斯二阶梯度模板卷积之后得到的，像素点关于x方向的二阶梯度。SURF方法考虑将高斯二阶梯度模板用盒函数来近似，如下图：



如此以来，我们可以通过积分图像非常方便地计算高斯二阶梯度，得到其近似：



因为是近似，我们也需要平衡两者之间的相关比，这里我们假设，尺度为1.2的高斯模板可以用9\*9的盒函数模板代替，然后计算下式归一化尺度的模板比值，这里的是指Frobenius范数：



最后Hessian矩阵的行列式，我们可以近似为：



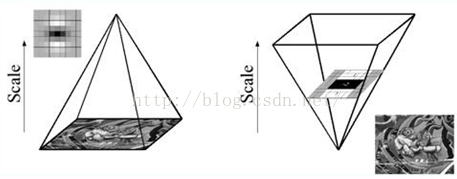
由此，这里的0.9是归一化比值，所以在任何尺度下，我们都可以通过这个比来补偿近似造成的误差，因此任何尺度下，我们都可以计算近似Hessian行列式的值。

#### 3.3.2 SURF的尺度空间

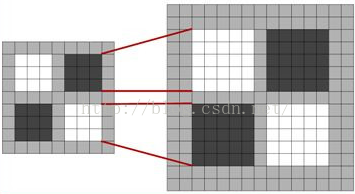
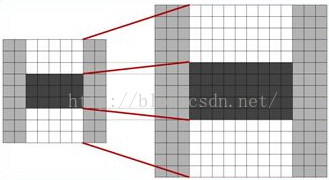
尺度空间通常通过高斯金字塔来实施，图像需要被重复高斯平滑，然后经过不断子采样，一层一层直到塔顶，如sift方法。而SUFR通过盒函数和积分图像，我们就不需要像SIFT方法那样，每组之间需要采样，而且还需要对当前组应用同上层组相同的滤波器，而SURF方法不需要进行采样操作，直接应用不同大小的滤波器就可以了。

为什么可以这样呢？因为都是为了得到不同尺度的图像，而sift通过采样操作比图像卷积操作计算量更少，而对于SURF来说，不存在这样的问题，因为盒函数和积分图像的操作计算量也非常小。另一方面，因为不需要采样，所以也不会出现混叠现象。

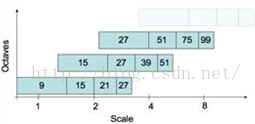
下图说明了这一情况，左图是sift算法，其是图像大小减少，而模板不变（这里只是指每组间，组内层之间还是要变的）。而SURF算法（右图）刚好相反，其是图像大小不变，而模板大小扩大。



SURF也是将金字塔分为组（Octaves），而每组分为若干层。其将9\*9大小的滤波器结果作为初始尺度组，即指的高斯尺度为1.2。那么接下来的每组，是通过逐渐增大的模板来进行滤波图像，一般情况下，滤波器的大小以9\*9，15\*15，21\*21，27\*27等变化，随着尺度增加，滤波器大小之间的差别也在增加。因此，对于每组来说，其滤波器大小增加数（15-9）是以双倍增长的（如6到12再24）。与此同时，提取兴趣点的采样间隔也是在以双倍增长的（这样可以获得小的尺度变化范围）。下面是模板的变化图：



下图反映了组及层之间尺度变化，及滤波模板长度变化过程，我们可以发现层间采样间隔以2倍扩大，所以随着层尺度增加，其尺度变化的粒度减少了，但是我们发现第一组每一层的尺度变化粒度太大了，所以在这里我们需要引入尺度空间更为精细的插值操作。

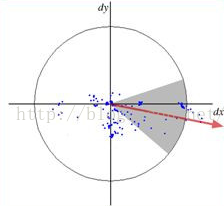


尺度空间搭建好了之后，同sift运算一样，我们找到在尺度及图像空间的3\*3\*3的范围内进行非极大值抑制，找到局部极值点（Hessian行列式），最后再应用尺度和图像空间的插值操作，以获得精确的兴趣点位置

#### 3.3.3 特征点主方向获得

为了获得旋转不变性，我们需要识别兴趣点区域的一个主方向。SIFT方法采用的是计算兴趣点附近3\*1.5Ϭ大小的圆形区域内方向直方图，选择最大的方向为主方向。而SURF方法则是通过计算其在x,y方向上的haar-wavelet响应，这是在兴趣点周围一个6s半径大小的圆形区域内。当然小波变换的大小也同尺度参数s有关，其步长为s，其大小为4s。

一旦区域内所有小波响应被计算，再对所有小波响应进行高斯加权（以兴趣点为中心，尺度为2.5s），然后建立小波响应dx,dy的坐标系（dx是小波在x方向上的响应，而dy是小波在y方向上的响应），将区域内的每点在这个坐标系来表示，如下图所示，选择一个60度的扇区（下图灰色区域），统计这个扇区所有响应的总和，就获得了一个总的方向（下图红箭头），旋转整个扇区，找到最长的矢量方向即为主方向。



#### 3.3.4 SURF描述子

同sift算法一样，SURF也是通过建立兴趣点附近区域内的信息来作为描述子的，不过sift是利用邻域点的方向，而SURF则是利用Haar小波响应。

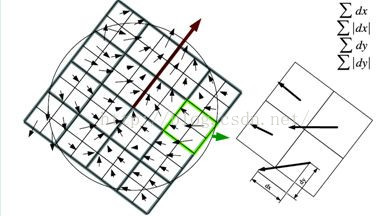
SURF首先在兴趣点附近建立一个20s大小的方形区域，为了获得旋转不变性，同sift算法一样，我们需要将其先旋转到主方向，然后再将方形区域划分成16个（4\*4）子域。对每个子域（其大小为5s\*5s）我们计算25（5\*5）个空间归一化的采样点的Haar小波响应dx和dy。

之后我们将每个子区域（共4\*4）的dx,dy相加，因此每个区域都有一个描述子（如下式），为了增加鲁棒性，我们可以给描述子再添加高斯权重（尺度为3.3s，以兴趣点为中心）





所以最后在所有的16个子区域内的四位描述子结合，将得到该兴趣点的64位描述子

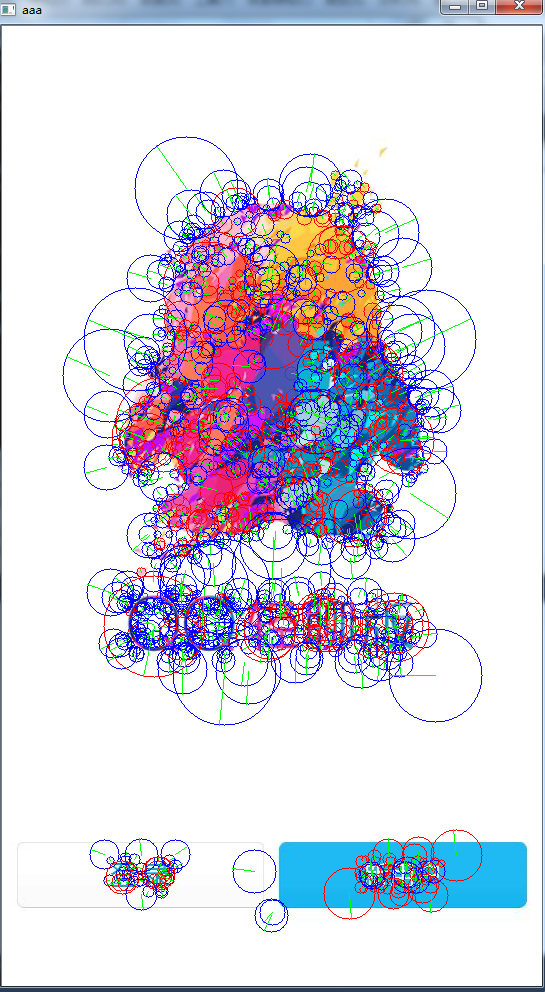


由于小波响应对于光流变化偏差是不变的，所以描述子具有了光流不变性，而对比性不变可以通过将描述子归一化为单位向量得到。

### 3.4 SURF算法改进

#### 3.4.1 特征点过滤

之前的surf匹配过程中，如果按照一般的过程，平均每个图像会提取到几百上千的特征点。

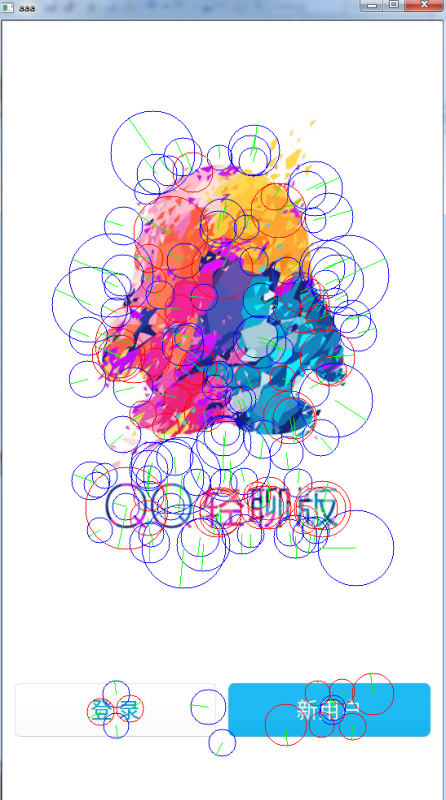
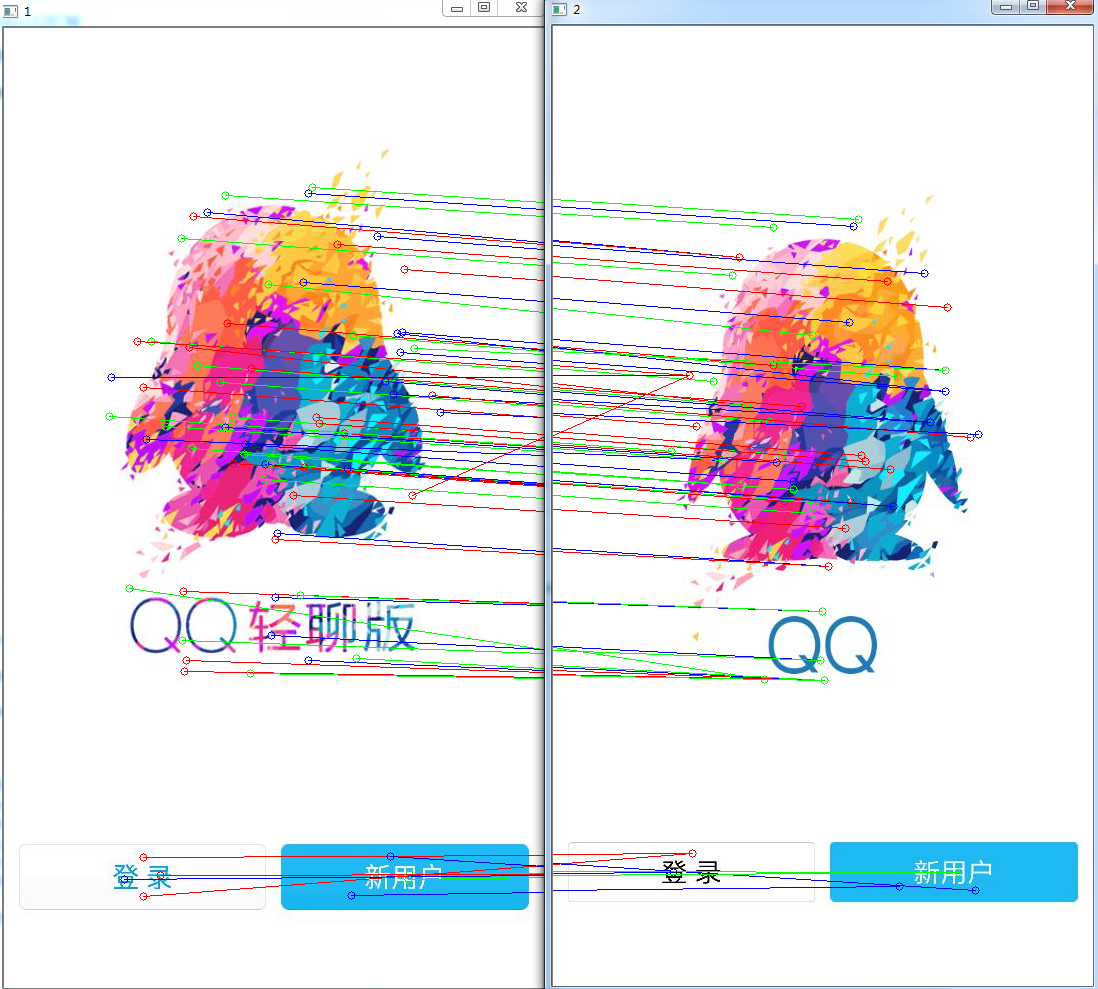
其中大圆圈代表轮廓特征，小圆圈代表细节特征。

改进：由于Android界面关心的是轮廓，因此可以对特征点进行删选，选择大的圆圈。

其中前三列为特征点数，第四列相似度 = 匹配点/max(qq特征点数，qq轻聊版特征点数)。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | qq轻聊版 | qq | 匹配点 | 相似度 |
| 全部 | 1300 | 1094 | 434 | 0.333846 |
| R>4 | 1033 | 882 | 350 | 0.338819 |
| R>8 | 386 | 339 | 146 | 0.378238 |
| R>12 | 221 | 180 | 87 | 0.393665 |
| R>15 | 145 | 113 | 69 | 0.475862 |
| R>20 | 79 | 53 | 27 | 0.341772 |

通过调节圆圈半径的大小，发现半径为 15 时达到最优效果 ，得到的特征点和匹配情况如下：

#### 3.4.2 匹配过滤

按照之前的方法进行比较，发现存在误判的情况：有约-星星



这个是surf算法本身没办法避免的问题，必须通过匹配点的过滤来解决。

google上找到一个方法，通过 计算各个线段的长度，然后 过滤线段长度（lenght<2\*minlength）来处理 。

然后对于如上的这个图片，没有任何过滤作用，因为最长最短比值没有达到2，并且这个比率也很难设定。

改进：标准差策略来进行匹配过滤

因此考虑计算线段的长度，得到一个长度的数组，计算数组的标准差来判断聚集程度

C:\Users\Robin\AppData\Local\Temp\enhtmlclip\Image(5).png

再计算qq轻聊-qq的标准差

C:\Users\Robin\AppData\Local\Temp\enhtmlclip\Image(6).png

发现  有约-星星  比  qq轻聊-qq 的标准差大很多，存在更大的误判。

因此考虑将标准差作为判断误判情况的判断因子：

        标准差大：误判多

        标准差小：误判少

则修正后相似度的表达式：

相似度 =匹配点数/max(图1特征点数，图2特征点数)\* （标准长度-标准差）/标准长度

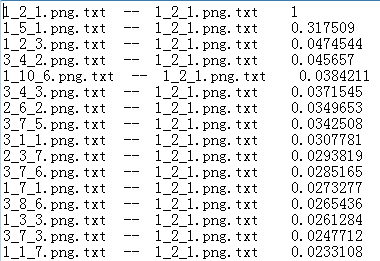
其中标准长度为图片的大小（斜边的分辨率）

#### 3.4.3 相似度的阈值选择

  现在目前得到的是50个app的手动截图，共计194张图片，现在计算所有两两图片的相似度，发现大部分的相似度都小于10%，高于10%的一些情况如下：

截图的第一列为第一幅图的特征点的文件，第二列为第二幅图的特征点的文件，第三列为两幅图的相似度。

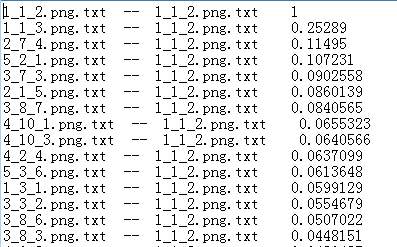
1、



相似度达到30% 的两个图片，可以判定为相似。

2、





相似度达到25%的两个图片，可以判定为不太相似。

综上：可以将阈值判定在两者之间的任何值，或者通过其他的图片来判断阈值。

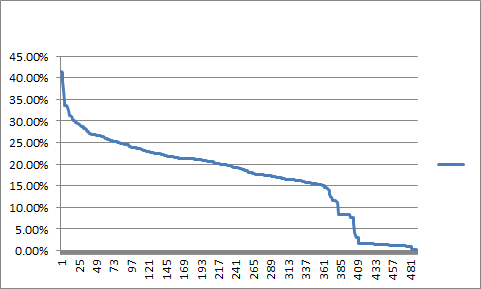
## 实验结果分析

### 4.1 基于类目录文件的实验结果

本实验以社交类的500个应用为测试对象，根据已知的正常应用和相似应用来判定阈值，通过这些阈值来确定整体对象中的重打包应用，严重相似应用，部分相似应用，再通过拓扑图直观的显示相似应用之间的关联。

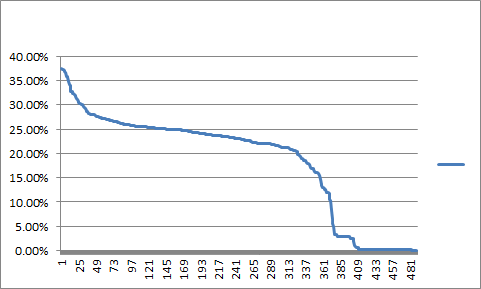
#### 4.1.1 正常应用

如下为微信应用与其他499个应用之间的相似度，图片显示的是按照相似度降序排列得到的相似度，列表显示的是相似度最高的前10个应用的相似度，发现微信应用与其他应用的相似度较低，全部低于42%



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3\_4\_Faceu激萌.apk.txt | 3625 | 4728 | 4883 | 41.54% |
| 36\_7\_王者荣耀爱玩助手.apk.txt | 3625 | 5823 | 5558 | 41.17% |
| 4\_6\_视吧.apk.txt | 3625 | 3061 | 4078 | 39.01% |
| 3\_3\_WhatsApp.apk.txt | 3625 | 8919 | 7894 | 37.07% |
| 24\_5\_葫芦侠精选.apk.txt | 3625 | 9094 | 8359 | 34.28% |
| 38\_6\_微笑直播.apk.txt | 3625 | 11459 | 10016 | 33.60% |
| 48\_7\_微笑直播.apk.txt | 3625 | 11459 | 10016 | 33.60% |
| 49\_2\_网络测速器.apk.txt | 3625 | 2616 | 4149 | 33.52% |
| 8\_10\_微信语音表情.apk.txt | 3625 | 2888 | 4351 | 33.20% |
| 20\_6\_蜜色直播.apk.txt | 3625 | 11762 | 10303 | 33.04% |

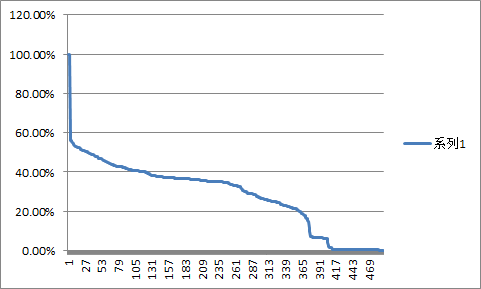
如下为微信应用与其他499个应用之间的相似度，图片显示的是按照相似度降序排列得到的相似度，列表显示的是相似度最高的前10个应用的相似度，发现微信应用与其他应用的相似度较低，全部低于38%



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9\_5\_喵播-美女直播视频交友.apk.txt | 18392 | 13936 | 20238 | 37.40% |
| 21\_4\_碉少堡论坛.apk.txt | 18392 | 19156 | 23552 | 37.27% |
| 21\_7\_四川麻将社区.apk.txt | 18392 | 19156 | 23552 | 37.27% |
| 17\_6\_文玩迷.apk.txt | 18392 | 18686 | 23304 | 37.15% |
| 22\_4\_文玩迷.apk.txt | 18392 | 18642 | 23278 | 37.14% |
| 41\_2\_陌陌美女秀.apk.txt | 18392 | 20212 | 24410 | 36.77% |
| 10\_5\_Up直播.apk.txt | 18392 | 13900 | 20452 | 36.67% |
| 38\_6\_微笑直播.apk.txt | 18392 | 11459 | 19185 | 35.73% |
| 48\_7\_微笑直播.apk.txt | 18392 | 11459 | 19185 | 35.73% |
| 20\_6\_蜜色直播.apk.txt | 18392 | 11762 | 19428 | 35.57% |

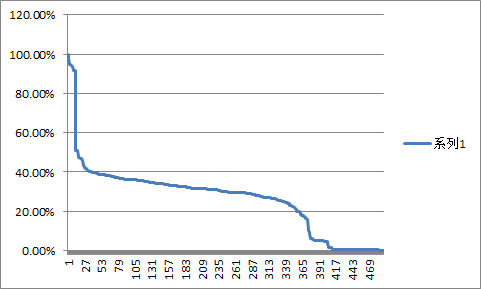
#### 4.1.2 相似应用

现在应用库中找几个与其他应用相似度很高的应用，如初夜约，与其他的某些应用的相似度可以达到100%，这种应用就是完全重打包的应用



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 13\_6\_夜色有约.apk.txt | 7541 | 7541 | 0 | 100.00% |
| 26\_4\_夜色约爱.apk.txt | 7541 | 7541 | 0 | 100.00% |
| 32\_9\_秒约爱.apk.txt | 7541 | 7541 | 0 | 100.00% |
| 6\_1\_初夜约.apk.txt | 7541 | 7541 | 0 | 100.00% |
| 12\_8\_爱流量.apk.txt | 7541 | 6873 | 6312 | 56.21% |
| 22\_5\_水晶直播.apk.txt | 7541 | 7451 | 6610 | 55.91% |
| 44\_9\_天籁K歌之王.apk.txt | 7541 | 11063 | 8274 | 55.53% |
| 35\_5\_微信表情.apk.txt | 7541 | 7642 | 6861 | 54.81% |
| 49\_7\_QQ空间助手.apk.txt | 7541 | 8005 | 7066 | 54.55% |
| 31\_9\_虫虫公会.apk.txt | 7541 | 6815 | 6580 | 54.17% |

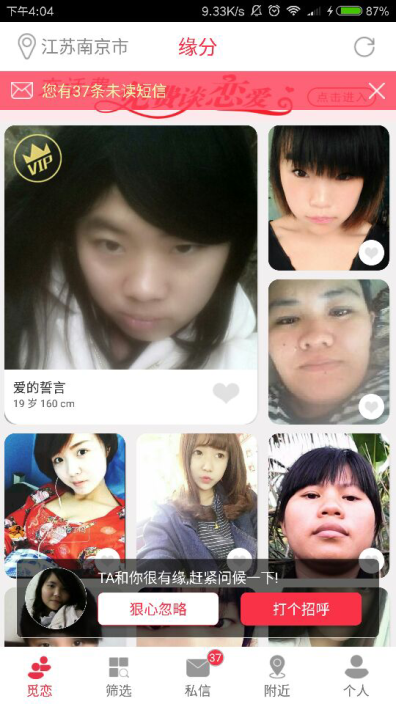
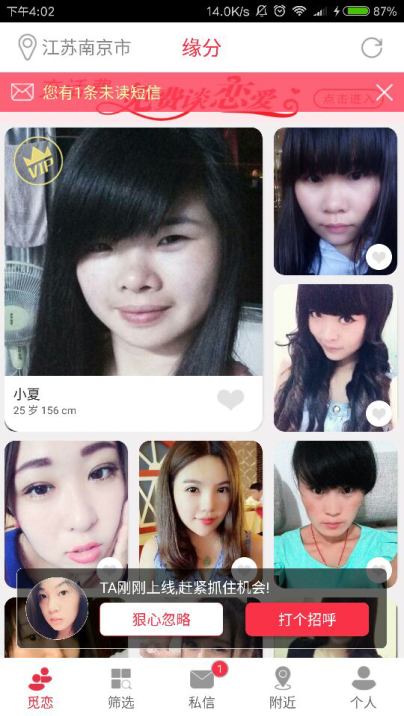
对于某些应用，如同城夜约，与其他的某些应用的相似度可以达到90%以上，这就属于严重相似的应用



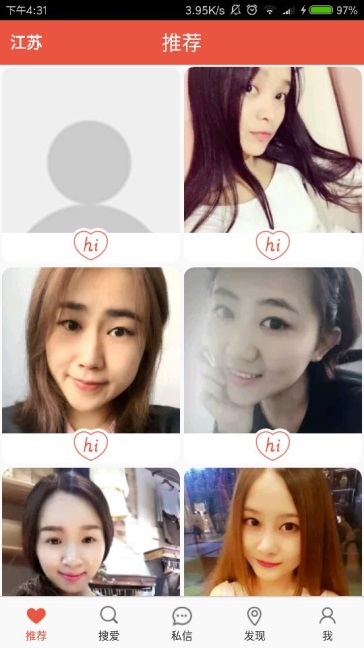
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 23\_9\_同城夜聊.apk.txt | 8799 | 8935 | 368 | 97.92% |
| 33\_1\_夜夜约爱.apk.txt | 8799 | 8922 | 935 | 94.72% |
| 43\_4\_恋爱吧.apk.txt | 8799 | 8596 | 949 | 94.54% |
| 40\_3\_午夜爱吧.apk.txt | 8799 | 9272 | 989 | 94.53% |
| 47\_8\_寂寞单身约爱.apk.txt | 8799 | 9023 | 1004 | 94.37% |
| 39\_8\_热恋同城.apk.txt | 8799 | 8874 | 1049 | 94.06% |
| 29\_2\_求恋爱.apk.txt | 8799 | 9294 | 1127 | 93.77% |
| 16\_6\_成人社交.apk.txt | 8799 | 9294 | 1505 | 91.68% |
| 23\_10\_成人社交.apk.txt | 8799 | 9294 | 1505 | 91.68% |
| 46\_5\_寂寞成人约.apk.txt | 8799 | 9294 | 1505 | 91.68% |

#### 4.1.3 应用验证

对于以上检验的到完全重打包应用，选取其中的三个应用，安装到手机上，得到截图如下，通过截图可以判断，三个应用除了数据源存在差异，在功能和内容布局上完全一致，证明就是完全重打包应用

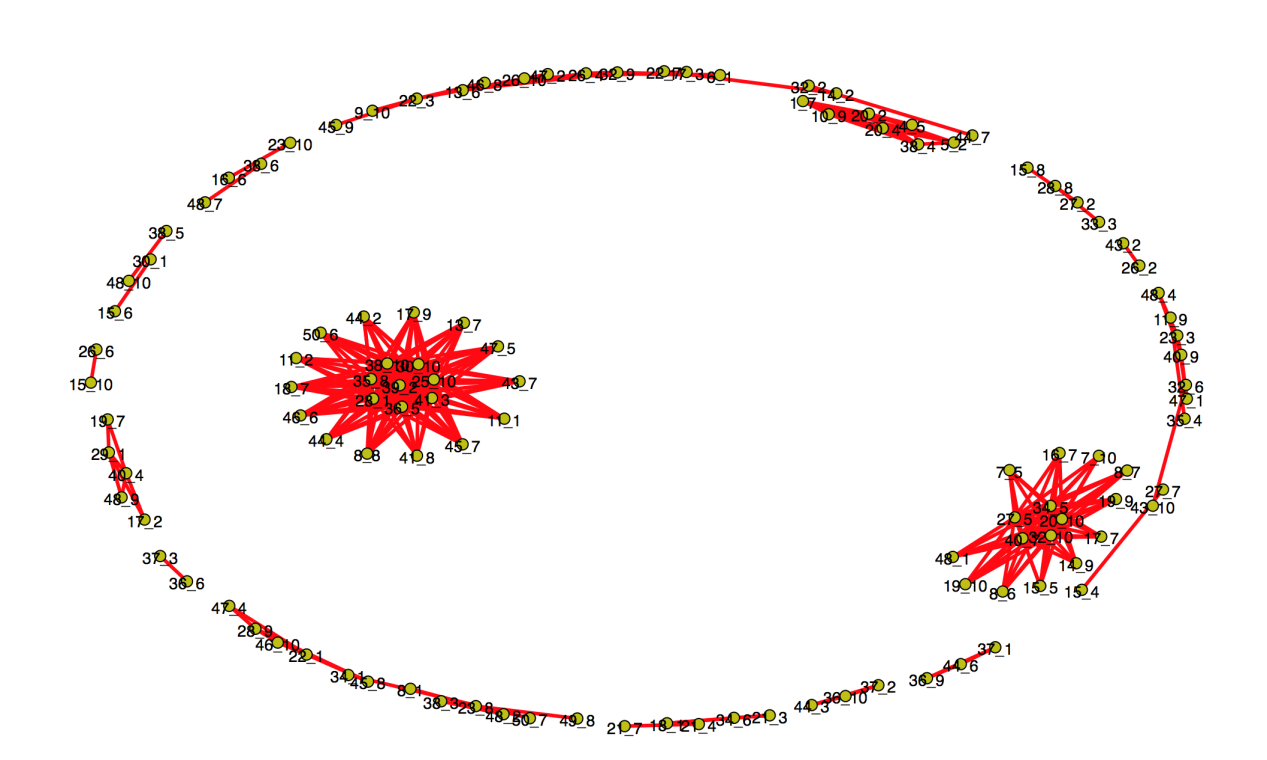
 

对于以上检验得到的严重相似应用，其中截图1的应用与其他两个应用的相似度都超过90%，在图中发现就是将图片的格式改为图片加简介的方式进行展示。而对于其他功能也并没有任何修改，证明为严重相似应用

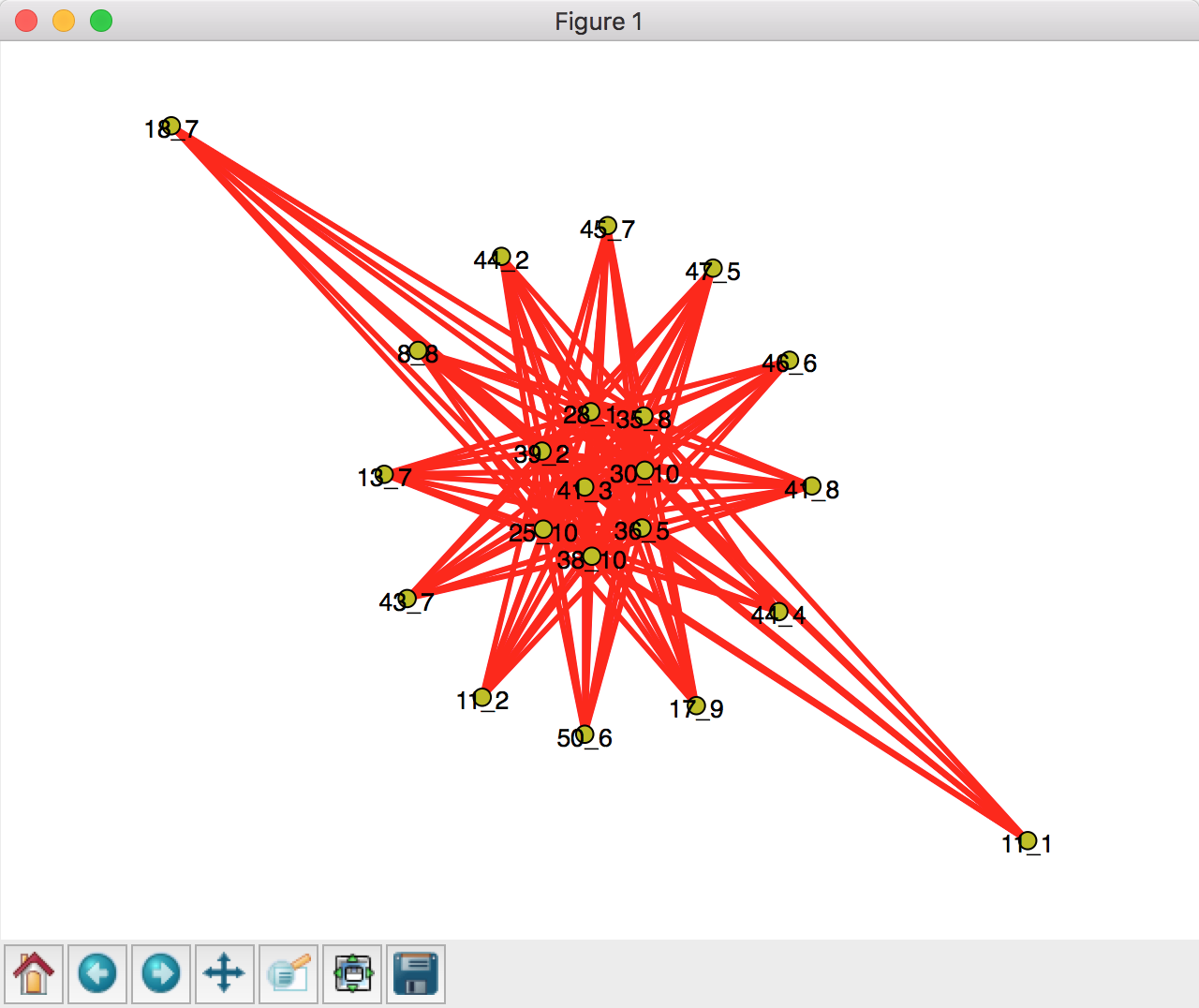
#### 4.1.4 拓扑图展示

对于一个应用，通过单一的指标很难去判断是否是相似应用，因为相似应用往往是成簇存在的，对于应用，更好的判断标准应该是判断与相似应用簇的关系，而通过拓扑图可以很清晰的观测改应用与其他相似应用的关联，对于没有显示在拓扑图中的应用，表明该应用在该类中是独立应用



该图中显示了社交类的500个应用的apk的完全相似应用的关联图。该图中显示共有103个应用相互之间存在完全相似。通过这个图中，还可以发现有三大应用簇聚集在一起。

1、



其标示对应的apk名称和度数如下：

25\_10 才米公社 28

36\_5 湖南涂料网 28

39\_2 湘财证券 28

35\_8 开鑫贷 28

30\_10 模拟炒股 28

41\_3 员工贷 28

28\_1 铜掌柜理财 28

38\_10 微记账 28

45\_7 掌上建筑咨询 8

46\_6 每日一股炒股票 8

47\_5 中大辅料城 8

11\_2 齐鲁银行 8

11\_1 天眼查-企业信息信用查询 8

18\_7 福彩双色球彩票 8

50\_6 分期贷 8

44\_2 山西商贸行业网 8

13\_7 中彩票 8

44\_4 祺天优贷 8

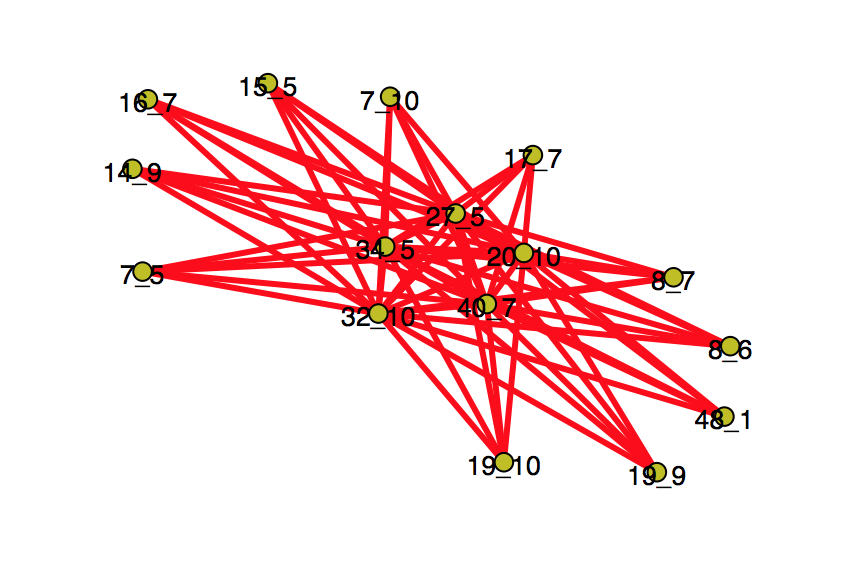
41\_8 鸿运彩票-注册送充值送 8

43\_7 点牛金融 8

8\_8 信达天下 8

17\_9 储蓄罐 8

2、



其标示对应的apk名称和度数如下：

34\_5 工业照明 19

20\_10 金融工场 19

32\_10 广金所 19

27\_5 借了吗 19

40\_7 生命云服务 19

17\_7 东吴证券同花顺 5

14\_9 钱站 5

16\_7 操盘英雄 5

15\_5 金融工场 5

19\_10 新新贷理财 5

7\_5 小方 5

48\_1 金行家 5

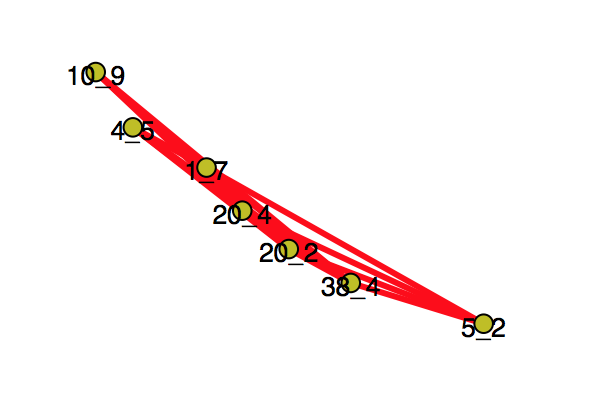
8\_6 银天下&middot;贵金属 5

8\_7 福建农信 5

19\_9 粤国际 5

7\_10 直销银行 5

3、



其标示对应的apk名称和度数如下：

38\_4 钱牛牛 9

20\_4 智e付 9

20\_2 算局七星彩奖表 9

1\_7 中国建设银行 9

10\_9 网易1元购 4

5\_2 智远一户通 4

4\_5 益盟操盘手经典版 4