SERVICES FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TNC TRANSACTIONS ON NETWORKS AND COMMUNICATIONS

VOLUME 8, NO. 2
ISSN: 2054 -7420

Analysis of Permissions Correlation for Android Apps Using Statistical SVD Approach

[1]Zon Nyein Nway, [2]Htet Htet Win
[1, 2] University of Computer Studies, Yangon, Myanmar;
ms.zonnyeinnway@gmail.com; htethtetwin99.hh@gmail.com;

ABSTRACT

Nowadays, almost all of the users use Android applications in their smart phones for various reasons Since Android is free operating system, android-apps can be easily downloaded via biggest open app stores and third party mobile app markets. But, these applications were not guaranteed whether these are malware apps or not by legitimate organizations. As mobile phones are glued with most of the people, malware applications threaten all of them for their private information. So, the work of analysis for the apps is very important. The proposed system analyzes the correlation patterns of app's permissions that must be used in all android apps by developers by using a statistical technique called singular value decomposition (SVD). The analysis phase uses the numbers of malware samples 50 to 300 from https://www.kaggle.com/goorax/static-analysis-of-android-malware-of-2017. The proposed system evaluates the risk level (High, Medium, and Low) of Android applications based on the correlation patterns of permissions. The system accuracy is 85% for both malware and goodware applications.

Keywords: Android, Permissions, Apps (applications), SVD (Singular Value Decomposition), Risk level, Malware, Goodware.

1 Introduction

When we learn the future of mobile, we have found out that there were 2.4 billion of Internet users at 2013. And there will be 5.4 billion at 2025. As the Internet users growth around the world, on the other hand there also increases the people who connect the Internet via mobile. So, we guess there will be 80% of Internet users who make Internet connections via mobile at 2025. Moreover, 50% of transactions will be made by phone at 2050. When we analyze the anatomy of attack, we have found out there are a lot of attacks. They are device attack, network attack or datacenter attack, etc. Moreover, there are many different ways to attack. For example, in device attack, attackers can attack our mobile phone through browser, phone, sms or applications, etc. Among them, our system is intended to detect applications which use unintended permissions. We would like to inform security issues arising from app stores. Attackers can download an application from official app stores and then repackage that application to third party app stores. And as Android Architecture, there are four layers. The first layer is application layer which can interact with end users. And the second layer is application framework. We can develop an android application through this layer.

DOI: 10.14738/tnc.82.8181
Publication Date: 30th April 2020
URL: http://dx.doi.org/10.14738/tnc.82.8181

Android is the powerful operating systems supporting a large number of Application in smart phones. These applications make life more comfortable. With the repaid growing of Android application every day, there are growing threats for the mobile users by installing more malwares without ability to detect them before installing the applications to the user device. Malware name came from “Malicious Software”, its software designed to secretly access a system without the owner’s device knowledge. A key challenge is to identify a suspected application as anomalous (malware). Therefore, we propose the system that can detect the particular app is malicious or not and cancel the installation if the permissions are unacceptable.

The rest of this paper is organized as follows. In section 2, Related Works is presented. The background theory is described in section 3 and proposes system overview is explained in section 4. In section 5, analysis and the experimental results are shown. The conclusion of this system is in section 6.

2 Related Work

Jin Li, Lichao Sun, Qiben Yan, Zhiqiang Li, Witawas Srisa-an and Heng Ye [1] discussed that Scaling the detection for a large bundle of malware apps remains a challenging task. When there are too many attributes of training datasets, decision tree hits a poor accuracy and the training phase tend to take more time and memory. Instead of extracting and analyzing of all android permissions, develop 3-level of pruning by mining the permission data to identify the most significant permissions. This system achieves over 90% of precision, recall, accuracy, and F-measure using SVM classifier and reduces 4 to 20 analysis times than using all permissions. The limitation of this system is that it cannot grantee advanced obfuscation techniques such as polymorphic and metamorphic malware.

G.N.Bharathi, T.Anusha, R.S.MeenaKumari and P.Aprna [2] presented that most of the users don’t understand the permission information and they used to ignore these permission information. At the application installation time, we don’t know the apps permission risk and apps may be malicious or not. So, this system assigns a risk score to each app and displays a summary of that information to users. The risk analysis based on the total no: of permissions. The inclusion of risk-score information has significant positive effects in the selection process and can also lead to more curiosity about security related information. The risk calculation is not intended for a user specified application which can be a particular application before installing. Scanning the preinstalled apps are demonstrated as experimental section.

Zarni Aung, Win Zaw[3] proposed that android-based smart phone users can get free applications from Android Application Market. But, these applications were not certified by legitimate organizations and they may contain malware applications that can steal privacy information for users. This system describes the machine learning based malware detection system by using k-mean cluster, decision tree classifier. Both malware and normal applications are classified by using machine learning technique. The limitation is that the system needs to train model with larger dataset to obtain enough samples of malicious applications. And it cannot classify the types of malware application. This system uses a collection of 500 sample android applications and uses Weka tool to analyze the proposed framework.

3 Background Theory

3.1 Mobile Security Issues Arising from App Stores

The malicious applications come into our mobile devices from App Stores.

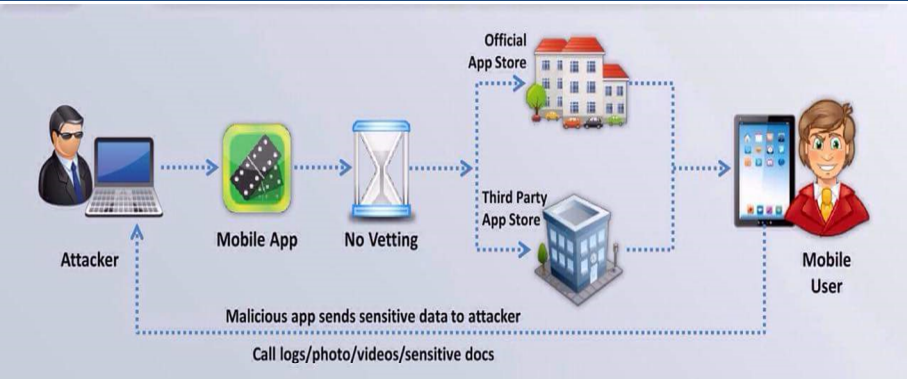


Figure1. Security Issues arising from Mobile App Store

Attackers can download an application from official app stores and then repackage that application to third party app stores.

3.2 Android Permission

The purpose of the permission in android application is to protect the privacy of an Android user. Android applications use a lot of permissions such access the SD card, use the Internet and so on. Google does a great job of declare what permissions an app uses before installing, but does not go into full detail about what exactly those permissions do. When installing an app from the Google Play Store, display dialog box pops up and the user has to agree that the app can do certain things. Every application must have an Android Manifest File. The manifest file provides essential information about the app to the android system.

```
<?xml version="1.0" encoding="utf-8" ?>
- <manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example.lenovo.flu">
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
- <application android:allowBackup="true" android:icon="@mipmap/icon" android:label="@string/app_name" android:supportsRtl="true" android:theme="@style/AppTheme">
  - <activity android:name=".LoginActivity" android:theme="@style/AppTheme.Dark" android:windowSoftInputMode="adjustPan">
    - <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
  </activity>
  <activity android:name=".SignupActivity" android:theme="@style/AppTheme.Dark" android:windowSoftInputMode="adjustPan" />
```

Figure2. Sample Android Manifest.xml file

3.3 Singular Value Decomposition (SVD)

The statistical technique, called Singular Value Decomposition (SVD) is the matrix rank reduction technique. It aims to identify the statistical association of permissions. The results of decomposition of the original matrix are descriptions of permissions and apps based on the hidden correlation space derived from SVD. The method reflects the major associative patterns in the dataset.

Consider the M x N Permission_App matrix W.

If W has rank r <= min (M, N), then W can be decomposed by singular value decomposition:

$$W = USV^T \quad (1)$$

Where:

S is also called the "concept matrix"

U is the "permission-concept similarity matrix"

V is the "app-concept similarity matrix"

M is the number of permissions

N is the number of apps

R is the rank value

3.4 Similarity Measure

Another key factor in the success of the proposed system is the similarity measure between testApp and malware apps. There are three simple and well-known similarity measures to calculate the similarity. They are the Dice, Jaccard and Cosine Coefficients. Among these three similarity measures, the system is used Jaccard similarity to measure the related permissions patterns in testApp and set of trained malware apps.

3.5 Jaccard Similarity Coefficient

The Jaccard index [5], also known as the Jaccard similarity coefficient (by Paul Jaccard), is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets.

Let D = {D1, D2,……Dn) be the collection of N malware apps. Each malware app Di can be represented by a corresponding set Si such that Si is a set of all the permissions contained in Di. Let us denote that set by Di such that Di = {S1,S2,…….. Sn}.

Some attributes are present in just a few objects of a data set. As they assume zero values in most of the cases, they are called asymmetric. Jaccard Similarity Coefficient measure is used to handle asymmetric binary attributes as only non-zero values are relevant for the calculation.

$$S_{A,B}^{Jaccard} = \frac{|\{Per_A\} \bigcap \{Per_B\}|}{|\{Per_A\} + \{Per_B\}| - |\{Per_A\} \bigcap \{Per_B\}|} \quad (2)$$

Where: Per_A = permissions from trained dataset

Per_B = permissions from user-chosen app

4 Framework of the Proposed System

Since the proposed system is implemented by using vector space model, preprocessing stages and indexing are needed. Therefore, query apps and malware apps can be easily and quickly compared. The results are shown in decreasing order of similarity values with the query apps. When we implement the proposed system, we need to pass the following three steps.

4.1 Data Collection

To implement the proposed system, the first thing we should do is to collect the information about the risky apps as much as we can. We need to analyze the nature of risky apps. According to the literature, there are so many ways to analyze different kinds of apps such as by analyzing signature features, behavior features or anomaly features and so on. Among them, we choose to analyze the apps based on permissions.

There are a lot of permissions that are declared by Google. Moreover, there are also customized permissions. The specific permission has its own task such as reading contacts, or sending sms or getting GPS, etc. Some of them are dangerous. Some of them are normal. Some of them are nothing meaning etc. But when analyzing permissions, it isn't enough to know which permissions are dangerous and which permissions are normal. One application can use as much as permissions according to the developer. And, we cannot conclude an application has high risk or low risk by seeing the permissions only.

So, we need to analyze which correlation patterns of permissions are usually involved in high risk application. To get the correlation patterns of permissions, we choose to apply Singular Value Decomposition (SVD) technique. For the training dataset, we must train malware dataset because we want to test how much risk level has an incoming application based on the correlation pattern of the permissions. We got the required dataset from https://www.kaggle.com/goorax/static-analysis-of-android-malware-of-2017.

4.2 Preprocessing

We need to preprocess the downloaded dataset to be ready to use as the trained dataset in our proposed system. There are two steps for preprocessing phase: tokenization and removing Duplicate Permissions. Tokenizing is the process of breaking of stream of text up into words, phrases, symbols or other meaningful elements. To store the permissions for each application, we extract the required permissions from the json dataset by tokenization. Then we build original matrix (permission-app matrix).

Some permission is frequently occurring and that do not represent any content of the application. Duplicate permissions are list of permissions that the developer includes them unintentionally. So, in this phase, duplicate permissions are removed before building the original malware vector.

4.3 Proposed System Design

The proposed system comprises into two phases. The first one is the training phase. The training phase is starting from extracting permissions of each JSON file to performing statistical singular value decomposition (SVD) approach as follow.

Step 1 Place JSON files under 'download' folder of emulator's internal storage.

Step 2 For each JSON file, extract Permissions and do Preprocessing phase. Create a Permission_App relation (perID and appID)

Step 3 Generate the Boolean permission_app matrix and save it to database

Step 4 Compute S, V, U metrics (using Singular Value Decomposition) and reduce the metrics with k dimension (suppose: k=4).

Step 5 Save S^{-1} and U metrics to the corresponding data files

Step 6 Transpose V (malApp vector) and save it to the corresponding data file

The second one is the testing phase. That testing phase also contains two sub-phases. First sub-phase is finding the permission-correlation pattern of the user chosen application as follow.

Step 1 Accept a test app. Get Permission Values of <uses-permission> Element from Android's Manifest.xml (by getRequestedPermissionList() method)

Step 2 Generate testApp vector (q).

Step 3 Calculate queryApp vector by computing q^T U S^{-1}

The second sub-phase is giving message to the user about the user chosen application's risk level as follow.

Step 1 Fetch V^T from corresponding data file.

Step 2 Compute the similarity values between queryApp vector and malApp vectors (V^T) and save them to temporary similarity result array.

Step 3 Choose the highest similarity value (h) from the similarity result array.

Step 4 If h is greater than or equal to maximum threshold value (0.8000), show the message "The application has high risk permissions." in alert box.

Step 5 Else if h is greater than or equal to minimum threshold value (0.5847), show the message "The application has medium risky permissions." in alert box. Otherwise, show the message "The application has low risk permissions." in alert box.

5 Experiments and Analysis

We have over 4000 malware datasets as described in Section 4.1. But we can't train all of that according to phone storage and emulator performance. Firstly, we must find out how much dataset can be trained. So, we train data beginning from 50 dataset and then by adding 50 JSON files again and again to the dataset. Here I've found out that emulator was hang at trained dataset 200 on 4G RAM. So, we tried to train dataset beginning from 250 dataset on a laptop which has 8G RAM. We took a lot of time to train that amount of dataset. Finally, we have found out that 300 datasets are more suitable on the current situation according to Emulator performance and mobile phone storage.

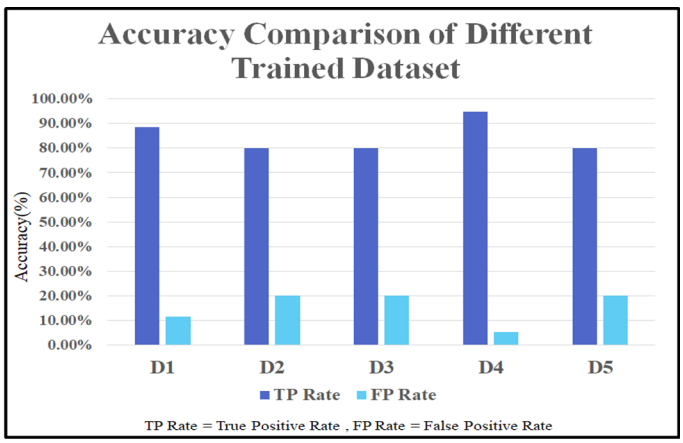


Figure3. Accuracy Comparison of Different Trained Dataset

At that time, we faced another problem that is which 300 datasets will be trained among these 4000 malapps. So, we separated our 4000 datasets by 300. And we trained different 300 datasets on our system to know which dataset has the more features of current environment malapp by testing 95 malapps. Here, we have found out D4 dataset includes the more features of the current environment malapps. So, we choose that D4 dataset to train on our system.

The figure 3 shows Accuracy Comparison of Different Trained Datasets. We trained each of 300 separated datasets from 4000 malapps (D1 has different 300 malware samples. D2 has another different 300 malware samples). But at that figure, we just highlight only the most significant five datasets. After getting the best trained dataset, we must found out which k value will be the best on our trained data according to SVD method. So, we analyze different k value on our trained dataset.

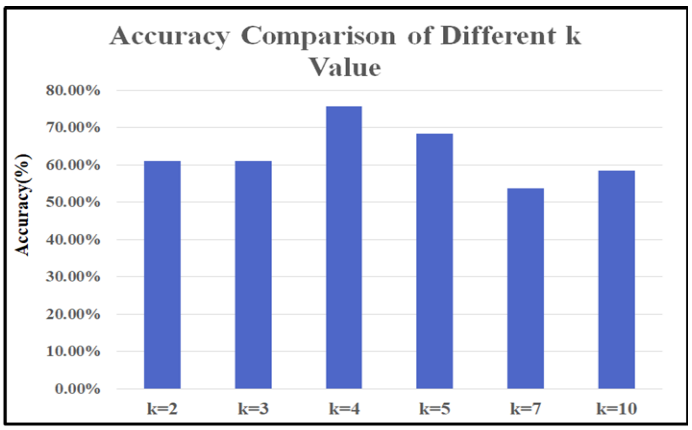


Figure4. Accuracy Comparison of Different k value

According to the figure 4, we've found out k=4 is the best value of the others. This figure shows the overall accuracy of our system at different k value. But we highlight just k=2 to k=5 among a lot of different k value.

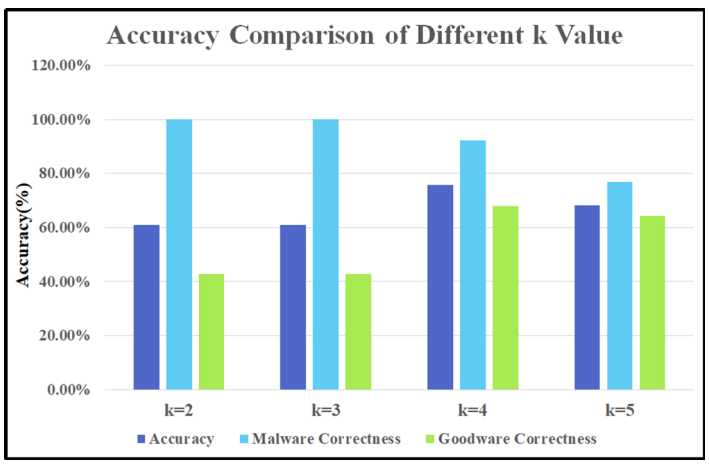


Figure5. Accuracy Comparison of Malware and Goodware Correctness for Different k Value

In figure 5, the correctness of malware is 100% at k=2. But the correctness of goodware is too low. At k=3, also like k=2. At k=4, the correctness of malware decreases a little, but the correctness of goodware is

significantly high. So, the accuracy of the system also increases significantly than others. At later k value, the correctness of malware is lower.

So, finally we choose k=4 on the trained dataset 300 according to our analysis of Figure 5 Accuracy Comparison of Malware and Goodware Correctness for Different k Value.

When we analyze the accuracy of our proposed system on 134 applications (96 malwares and 24 goodwares), our system identifies 83 apps as malwares and 13 apps as goodwares on 96 malwares. On the other hand, the proposed system identifies 19 apps as goodwares and 5 apps as malwares on 24 goodwares. So, the overall accuracy of our proposed system is 85%.

6 Conclusion

Focus of attackers and malware writers has changed to mobile devices due to the increased adoption of mobile devices for business and personal purposes. Therefore, App stores are common target for attackers to distribute malware and malicious apps. The important point is that the attacker can add the normal permissions to the manifest.xml file to hide their targeted dangerous permissions. So, only the numbers of normal permissions and dangerous permissions are not guaranteed whether the apps are safe or not. This system proposes to analyze the correlation pattern of permissions for detecting the anomalous Android applications. However, the growing amount and diversity of Android malware has significantly weakened the effectiveness of the conventional defense mechanisms, and thus Android platform often remains unprotected from new and unknown malware.

REFERENCES

[1]. "Significant Permission Identification for Machine Learning Based Android Malware Detection", L. Jin, S. Lichao, Y. Qiben, L. Zhiqiang, S.A. Witawas, Y. Heng, 2017.

[2]. "Effective Permission Analysis and Complete Security for Android Application", G.N. Bharathi, T. Anusha, R.S. MeenaKumari, P. Aprna.

[3]. "Permission-Based Android Malware Detection", Zarni Aung, Win Zaw, March 2013.

[4]. "Anomalous Android Application Detection with Latent Semantic Indexing", H. Shahriar, V. Vlincy, 2016 IEEE 40th Annual Computer Software and Applications Conference.

[5]. "Ethical hacking and counter measures by EC-Council".

[6]. "Spam SMS Detection on Android Smart Phone", Hay Mar Sam, July 2015.

[7]. "Information Retrieval for Pharmacy using location – Based Mobile System", Myat Su Aung, December 2016.

[8]. "Stock Trend Prediction from Mobile Device Through Web Services", Kay Thi Aung, December 2015.

[9]. "SMS Security with Aes Algorithm on Android OS", Phyo Su Khin, February 2017.

[10]. "Analyzing Terror Attacks using Latent Semantic Indexing", I. Toure, A. Gangopadjyay, 2013 IEEE.

[11]. "A Survey of Android Malware Characteristics and Mitigation Techniques", V. Cooper, H. Shahriar, and H. Haddad, Proc. Of the 11th International Conference on Information Technology: New Generations, IEEE CPS, Las Vegas, USA, April 2014, pp.327-332.

[12]. "Using Latent Semantic Analysis to Identify Similarities in Source Code to Support Program Understanding", A. Marcus and J. Maletic, Proc. Of 12th IEEE International Conference on Tools with Artificial Intelligence, November 2000, pp. 46-53.

[13]. "Effective Permission Analysis and Complete Security for Android Application", G.N.Bharathi, SSRG International Journal of Computer Science and Engineering – (ICCREST'17) – Special Issue – March 2017, ISSN: 2348 - 8387.

[14]. Foracchia, M., *Extraction and quantitative description of vessel features in hypertensive retinopathy fundus images*, in *Book Abstracts 2nd Int. Workshop on Computer Assisted Fundus Image Analysis*, E. Grisan, Editor 2001. p. 6-6.

[15]. Xiaohong, G., et al. *A method of vessel tracking for vessel diameter measurement on retinal images*. in *Image Processing, 2001. Proceedings. 2001 International Conference on*.

[16]. Lowell, J., et al., *Measurement of retinal vessel widths from fundus images based on 2-D modeling.* Medical Imaging, IEEE Transactions on, 2004. 23(10): p. 1196-1204.

[17]. Hong, S., et al., *Optimal scheduling of tracing computations for real-time vascular landmark extraction from retinal fundus images.* Information Technology in Biomedicine, IEEE Transactions on, 2001. 5(1): p. 77-91.

[18]. Pinz, A., et al., *Mapping the human retina.* Medical Imaging, IEEE Transactions on, 1998. 17(4): p. 606-619.

[19]. Zana, F. and J.C. Klein, *A multimodal registration algorithm of eye fundus images using vessels detection and Hough transform.* Medical Imaging, IEEE Transactions on, 1999. 18(5): p. 419-428.

[20]. Can, A., et al., *A Feature-Based, Robust, Hierarchical Algorithm for Registering Pairs of Images of the Curved Human Retina.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002. 24: p. 347-364.

[21]. Fritzsche, K., et al., *Automated Model Based Segmentation, Tracing and Analysis of Retinal Vasculature from Digital Fundus Images*, in *State-of-The-Art Angiography, Applications and Plaque Imaging Using MR, CT, Ultrasound and X-rays*2003, Academic Press. p. 225-298.

[22]. Hoover, A. and M. Goldbaum, *Locating the optic nerve in a retinal image using the fuzzy convergence of the blood vessels.* Medical Imaging, IEEE Transactions on, 2003. 22(8): p. 951-958.

[23]. Foracchia, M., E. Grisan, and A. Ruggeri, *Detection of optic disc in retinal images by means of a geometrical model of vessel structure.* Medical Imaging, IEEE Transactions on, 2004. 23(10): p. 1189-1195.

[24]. Huiqi, L. and O. Chutatape, *Automated feature extraction in color retinal images by a model based approach.* Biomedical Engineering, IEEE Transactions on, 2004. 51(2): p. 246-254.

[25]. Fritzsche, K.H., *Computer vision algorithms for retinal vessel detection and width change detection*, 2004, Rensselaer Polytechnic Institute: Troy, NY, USA.

[26]. Houben, A.J.H.M., et al., *Quantitative analysis of retinal vascular changes in essential and renovascular hypertension.* Journal of hypertension, 1995. 13(12).

[27]. Wasan, B., et al., *Vascular network changes in the retina with age and hypertension.* Journal of hypertension, 1995. 13(12).

[28]. Koozekanani, D., et al., *Tracking the Optic Nerve Head in OCT Video Using Dual Eigenspaces and an Adaptive Vascular Distribution Model.* Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, 2001. 1: p. 934.