Car-Rec: A Real Time Car Recognition System

Daniel Marcus Jang and Matthew Turk
University of California, Santa Barbara
Santa Barbara, CA 93106
dmjang80@gmail.com, mturk@cs.ucsb.edu

Abstract

Recent advances in computer vision have significantly reduced the difficulty of object classification and recognition. Robust feature detector and descriptor algorithms are particularly useful, forming the basis for many recognition and classification applications. These algorithms have been used in divergent bag-of-words and structural matching approaches. This work demonstrates a recognition application, based upon the SURF feature descriptor algorithm, which fuses bag-of-words and structural verification techniques. The resulting system is applied to the domain of car recognition and achieves accurate ($> 90\%$) and real-time performance when searching databases containing thousands of images.

1. Introduction

Car recognition is an important domain of object recognition. The ability to recognize cars (models or specific instances) has obvious applications in physical security, military, and law enforcement. Car recognition has previously been the subject of various papers [12] [9] to some success, but the field remains limited in viable implemented applications to provide real-time car recognition.

This paper presents Car-Rec, an approach to car recognition, building on recent technological advances in object recognition, to create a real-time car recognition system with strong recognition performance.

To illustrate the usefulness of a car-recognition system, consider a scenario where a "smart" security camera is stationed at the entrance of an office building parking lot. Throughout a regular work day, a typical employee might drive his/her car into the lot in the morning, exit for lunch, return for the afternoon and leave at night. This behavior would provide many opportunities to build an employee car database.

Given a database of training images representing the set of employee cars, the smart camera should recognize regular employee cars as they enter the lot. It should also be able to distinguish unfamiliar visitor cars from known employee cars. The camera could then notify building security whenever an unfamiliar car enters the lot, focusing security attention on potentially suspicious activity. This system would work in an automated fashion, giving visual hints to human users whenever an interesting or unfamiliar car is detected.

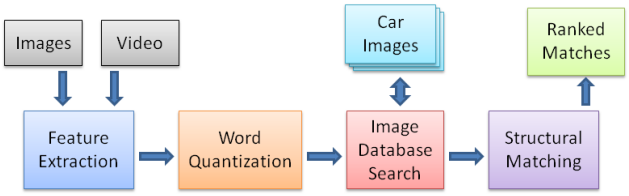

Figure 1. The Car-Rec search framework.

1.1. Car-Rec Framework

Car-Rec builds upon a set of published algorithms used in object recognition, utilizing a recognition framework similar to one used by Sivic and Zisserman [13]. This framework (see Figure 1) consists of four stages:

1. *Feature descriptor extraction*: Speeded-Up Robust Features (SURF) [5] is used to localize interest points in an image and describe their features as a vector of values.

2. *Word quantization*: feature descriptors are efficiently converted from high-dimensional vectors into single-value words. This is accomplished using a vocabulary tree trained on a database of car-related image features.

3. *Word quantization*: feature descriptors are efficiently converted from high-dimensional vectors into single-value words. This is accomplished using a vocabulary tree trained on a database of car-related image features.

4. *Structural matching*: the top results returned from the image database search are scored using a structural verification algorithm. The top matches are returned as a ranked list.

The main contributions of this work are an implementation of a real-time car recognition application and a simple framework, based upon the approach used by Sivic and Zisserman, for general object recognition. Car-Rec combines the benefits of multiple algorithmic approaches for car recognition.

2. Related Works

Shan *et al.* [12] investigated the possibility of identifying or tracking cars across different camera without direct image matching. Their approach compares edge-maps of cars captured by a single camera against a set of exemplar edge-maps selected for that camera. This comparison generates a vector of non-metric distances. A pairing of vector distances created from different cameras are then classified with an SVM to find the probabilities that the images contain the same car. Experimental results with this system show high matching accuracy compared under conditions of high illumination or large perspective differences.

Another approach for car identification is described by Ferencz *et al.* [9]. The main contribution of this work is a classification cascade built by arranging information-rich *hyper-features* extracted from a single vehicle exemplar image. This cascade may then be used to determine whether new query images match the training vehicle.

Car-Rec differs from these prior car identification systems in that it focuses on efficiently searching a large database of car imagery, using quantized feature descriptors to find a small list of likely matches. This list is then re-ranked with a structural verification algorithm.

Feature detection algorithms form the basis for Car-Rec's recognition. Scale Invariant Feature Transform (SIFT) [10] is a feature detector and descriptor algorithm notable for incorporating robust scale and rotational invariance to feature descriptions. SIFT has proved to be very popular in object recognition research and has inspired a variety of feature detector/descriptor algorithms which aim to reduce processing time or increase detection and descriptor robustness.

Speeded-Up Robust Features (SURF) by Bay, *et al.* [5] and further discussed by Evans [8] is another scale and rotationally invariant feature detector and descriptor algorithm. SURF is notable for fast computation times and increased robustness over SIFT.

SURF owes its relative speed to heavy use of image integrals in both feature detection and description stages. Due to its efficiency and robustness, we use SURF in the first stage of the Car-Rec framework.

Video Google, by Sivic and Zisserman [13], is a feature descriptor-based video search system, and is a major inspiration for Car-Rec, particularly in the algorithm framework. Extracted features from video frames are quantized into a vocabulary and indexed into a database. Ranking of matching documents is achieved by computing the cosine similarity between a query image and the other images in the database. A spatial consistency heuristic is used to decrease the number of false feature matches and rank matching images.

Indexing of a video sequence in Video Google requires re-building of a vocabulary and quantization of each of the features in the searchable frames. This pre-computation of query and search image indexes implies that this application is not suitable for real-time image streams.

Scalable Recognition with a Vocabulary Tree by Nister and Stewenius [11] acheive efficient quantization with a vocabulary tree structure, allowing for the use of vocabularies of a million words or more, with little computational overhead. Car-Rec incorporates vocabulary trees in its word quantization stage.

3. Algorithm Design

The Car-Rec search framework is made up of four stages: feature extraction, word quantization, image database search, and structural matching.

Prior to search, Car-Rec must be trained on a database of car imagery. The current version of Car-Rec accepts only still images for the training database. The database may comprise images of one or many different cars, each separated into folders labeled by car instance. Using this category system, a car may be represented by images of many different poses. The more exemplar images we have of various poses of a car in a database, the greater likelihood we are to accurately match on this particular car.

We assume that the database images contain few background (non-car) features. The database imagery cars used in Section 5 were photographed on neutral, black colored backgrounds. Future work on Car-Rec includes segmentation of cars from noisy backgrounds.

Training with database imagery involves extracting and quantizing features to create quantized word documents. Feature extraction and quantization are described in the following subsections. These quantized word documents are then added to a Lucene index.

Querying the Car-Rec image database follows the four stages of the search framework. Search is typically used with live video, but pre-recorded video or even still images may be used. When querying the database with live video, Car-Rec loops continuously, capturing the most recent video frame to be used as a database query. It then extracts local feature vectors and quantizes these vectors into a list of words. This list of words is used as a query into the

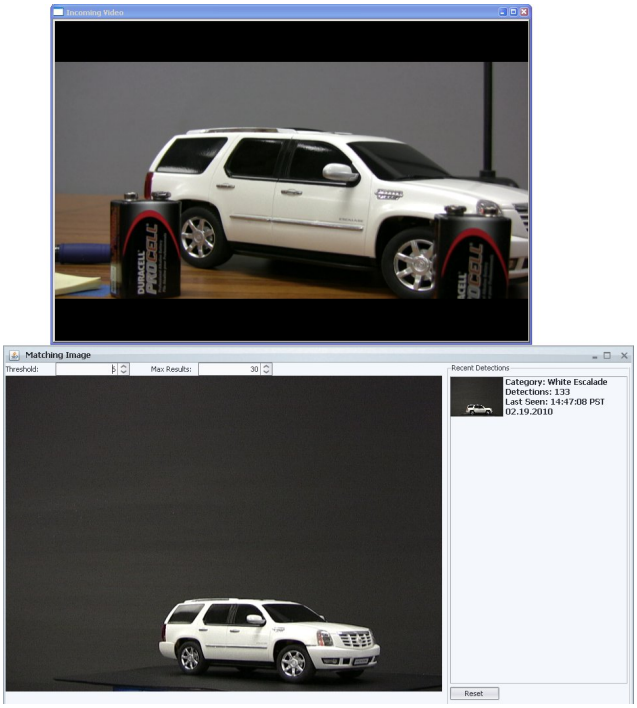

Figure 2. A match on a model car. The top window shows the live video image, and the bottom window shows a best match in a Car-Rec search.

Lucene index of database documents. The top results from the Lucene search are then fed into the structural verification stage of the framework, where they are given structural match scores. Finally, the best match is displayed, and the query image is categorized as an instance of the best match car model (see Figure 2).

3.1. Feature Detection and Description

Car-Rec uses SURF [5] for feature detection and description. A feature detection algorithm searches a digital image for interesting features, and localizes found features to subpixel keypoints on the image. SURF tends to detect blobs of pixels which contrast from surrounding pixels as interest points. SURF extracts oriented gradient information from the local area surrounding the keypoint to generate a feature descriptor vector. These descriptors are robust to in-plane rotations and changes in scale.

A benefit to using the local descriptor approach in image search is that partial occlusions are not a major problem. For example, if a car is partially occluded by surrounding foliage, it may be recognized with the few visible features. SURF can extract hundreds or thousands of feature descriptors in an image, but a structural matching algorithm building on these features may need only a few strong matching features between images for recognition, allowing flexibility in matching under partial occlusion.

3.2. Bag-of-Words Model

A large portion of Car-Rec's application follows the bag-of-words model [7]. This model relies on statistical relevance of words, while disregarding spatial or structural details of documents, in order to simplify indexing and search. In text-based search, this means that the ordering of words in a document is ignored. For example, the phrase "Alice chases Bob" would be equivalent to "Bob chases Alice," even though the word orderings impart different denotations to the phrases.

The bag-of-words model may be applied to computer vision by treating local image features as words. But, since feature descriptors are high dimensional vectors, each must first be quantized into a word from a vocabulary (see Section 3.3). The naive quantization approach would be to compare every image feature with every vocabulary feature. This linear nearest neighbor search is slow and precludes the use of very large vocabularies.

Many approaches have been devised to increase the speed of nearest neighbor search on feature descriptors, such as the KD-tree modification best-bin-first [6], vocabulary trees [11], or simply limiting the size of the vocabulary to a reasonably small size. Car-Rec uses vocabulary trees for quantization due to their strong performance and intuitive design.

3.3. Vocabulary Trees and Word Quantization

A vocabulary tree is built from a pool of feature descriptors. These descriptors should be extracted from imagery similar to the target database. The vocabulary trees used in the following experiments (see Section 5) were built from a pool of over 7 million features taken from over 10,000 car-related imagery compiled from Flickr [2].

To build a vocabulary tree, the pool of feature descriptors is separated into k number of clusters with k-means clustering. k may be arbitrarily chosen, and the trees in the experiments section use $k = 2$ in order to keep branch factors small, and increase control over the final tree size. The initial clusters represent the immediate k branching children of the tree root. These clusters are then separated into k sub-clusters, which become the second level of branch nodes. This process is repeated recursively until a desired tree depth is reached.

Each vocabulary tree node contains a cluster centroid vector and a unique integer ID. A feature is quantized to a word by first comparing this feature against root's children. The child node with the nearest centroid vector (using Euclidean distance) is chosen. Its child nodes are then compared with the feature. This process continues until a leaf node is reached. The ID of this leaf is then assigned to the feature and treated as its "word".

When compared to naive linear search, the performance

time of feature quantization is squashed from linear time to logarithmic time. The maximum number of comparisons required to quantize a feature on a vocabulary tree is $branchfactor * depth$. For example, a full binary tree with 18 levels (260,144 words) requires only 36 comparisons for quantization.

3.4. Image Database and Lucene Search

Images are converted into documents of unordered words using the vocabulary tree for quantization. Database and query imagery are both are converted into this document format. Car-Rec uses Lucene [1], an open-source search engine library, to create an index of the quantized words documents and query this database index. Lucene uses the boolean and vector space models for scoring, and words are weighted using term frequency-inverse document frequency (tf-idf).

tf-idf is a weight applied to each word in a document, based on the number of times it appears in the document, divided by the number of documents in which it appears. tf-idf weighting is an attempt to make frequently used words in a specific document more relevant in a search, while ensuring that overly common words across the entire database are not favored.

Car-Rec uses Lucene in a straightforward fashion: an image document is used as a direct query against the database index to return the top matching results. Lucene returns matching documents from a database of thousands of images in fractions of a second. With Lucene search, we are using the bag-of-words model; structure information will not be used until the structural verification stage (described in Section 3.5).

A list of hypothesized matches returned by Lucene may be satisfactory in some circumstances, especially if the user is only interested in getting a fuzzy list of matches and has the time to visually inspect each match. This could be particularly useful in a large-scale Internet-based image search where users are presented with a visual listing of the top matching results, akin to the interface of Tineye [3]. However, in a situation where an image search must clearly define a specific match, a structural matching algorithm is vital.

3.5. Structural Verification

Though the bag-of-words model allows for efficient searches, accuracy is lost by ignoring the positional layout of matching keypoints. In the structural verification stage, we improve search quality by re-incorporating and analyzing this structure information. There are numerous ways of verifying the structural consistency between matched images.

A simple possibility would be to use a Hough accumulator to bin keypoints based purely on 2D location. This has the effect of matching an object between images with the same scale and orientation but different positions. For example, consider a camera at street-level, with a perpendicular view of a street and cars passing through the scene. Assume, for simplicity, that background features are ignored. If we compare an image of a moving car entering the scene from the left side and an image of the same car as it exits to the right, the only major difference between the two images is the car's location.

The translational distance (measured in x and y offsets) between car features in the two images should be similar to one another and, after quantization, will be placed in the same accumulator bin. To account for minor errors associated with quantization and feature location detection, the features are placed in the two nearest bins in both x and y directions; thus each feature is placed in a total of 4 bins. Detection of the car's features amounts to an iteration through the accumulator to find the largest bin.

The 2D Hough accumulator is a good indicator of car matches, and the loss of orientation information is not much of a problem for this domain. If car imagery is taken from street level, cars will virtually always appear upright.

Scale differences between images may be overcome with a small modification to this algorithm. Each matching keypoint in one image may be scaled by a factor before finding the translation offsets. The algorithm can make multiple guesses at the relative scale differences between query and database images. Scale becomes another dimension of the accumulator bins. For example, if a query image car is twice as large as the matching exemplar image car, each of the query image feature locations can be scaled by a factor of 0.5 before calculating the location offsets to simulate matching two identically-sized car images. In practice, 10 different scale factors are used, ranging from 0.25 to 3.0.

The size of the largest accumulator bin is treated as a ranking score. This structural matching algorithm is a simplified modification of Lowe's algorithm [10], which, in turn, is based upon the Generalized Hough transform [4].

4. Implementation

Car-Rec is primarily written in Java, with some C. The vocabulary trees, word quantization, structural matching, most of the graphical user interface are in Java.

Image loading, video frame grabbing, live video display, and SURF feature extraction use functionality provided by OpenCV 2.0, an open source computer vision library, and are written in C.

The Lucene search functionality is provided by Lucene 2.9.0, an open-source Java library for text-based search.

Figure 3. The 16 different poses of a car in database A.

Figure 4. Side-views of 8 car types with name abbreviations. These cars made up the categories of image database C and were used as query categories in all experiments.

5. Experimental Results

The following experiments measure the performance of Car-Rec in accuracy and speed. For practical reasons, it was not possible to test Car-Rec on real car imagery, so toy cars were used. This afforded the luxury of building multiple car imagery databases on a variety of cars from arbitrary angles, without background clutter. 20 *Matchbox* cars were used for database imagery. These ranged from realistic, well-known car models to several more cartoonish varieties. Table 1 summarizes the three different image databases used in the following experiments.

Database A represents a comprehensive car database with 20 car types, and 16 view angles of each type, for a total of 2650 images. Each of the cars were photographed from 16 view angles on a rotating turntable (Figure 3). Each pose angle was captured with eight images. The redundancy in using eight images per pose is useful due to the tendency for SURF to detect certain different interest points in similar images. Each set of car images was labeled with a unique category indicating its model, with 128 images per category.

Database B is a subset of 160 images database A, containing only single view images. This test database represents the kind of database that may be used in situations where the car pose is known to be static, such as a security checkpoint.

Database C is another subset of database A, with 16 pose angles of eight cars, for a total of 1024 images. It is used in best match accuracy experiments, where the database images share the same categories as the query images.

Database	# Cars	Poses	# Images
A	20	16	2650
B	20	1	160
C	8	16	1024

Table 1. Car image databases

All of the experiment queries were made with non-database images created with various poses and scales. Eight car types (Figure 4) were used in the queries. These query cars were chosen due to their realistic nature. All experiment images (database and query) are 720x480 resolution.

5.1. Precision and Recall

This experiment queried car images against database B. This database consists of 20 car model categories with eight side-view images in each cateogry, for a total of 160 images.

The initial query set for each of the eight cars (Figure 4) consisted of non-database sets of four images taken at five different scales. This resulted in a total of 20 query images of each car. The single side-view poses in database and query imagery was useful in quantifying precision and recall. There was no ambiguity on correct car matches with widely differing poses (i.e. a spurious match made between an image of the front of a gray minivan and an image of the rear of the same vehicle).

Figure 5 suggests smaller vocabularies result in better overall performance, because a much higher number of images from the database are ranked with structural verification. The best performance of all is achieved by running structural match verification against every image in the database. This would occur in the most extreme cases of tiny vocabularies with one or a few words. Such an exhaustive search would obviously nullify any time benefit of using Lucene for a bag-of-words search. As the number of structurally verified images increases, so does the total search time.

The Lucene search acts as an intermediate form of search, pushing reasonable matches to the top of the list for verification, and is important for real-time operation. A reasonable compromise must be found between structurally verifying only one or two images (short search times) and structural matching against the entire database (longer search times).

Neither precision nor recall must be perfect in order to satisfy Car-Rec's recognition tasks. Only the best scoring match is considered for category classification, as discussed in the following section.

5.2. Best Match Accuracy

The structural matching stage generates a score indicating the largest cluster of structurally matching features between a query and retrieved image.

Image database C (1024 images) is searched for matches. Images in database C are separated into categories of car types at 16 angles. Query images are made from a subset of these cars at various poses and scales. The best match is simply the highest scoring database match category returned from the structural matching stage. Here, we are concerned with match accuracy in difficult scenarios where cars may be viewed in different poses. As long as the query image is correctly categorized by model (disregarding specific poses of matching query and database images), we consider this an accurate match.

The number of structurally verified Lucene results is a user-defined parameter that affects overall Car-Rec accuracy. Two confusion matrices are shown, using different structural verification values. Table 2 shows accuracy with four structurally verified matches and table 3) uses 50 matches.

The 2^{18} word vocabulary was used for these results. When 4 Lucene results are structurally verified, 6 of 8 car types are 92% accurate or better. Structurally verifying 50 results pushes all eight car type accuracy to 91% or higher, seven of which are 98% or higher in match accuracy.

Next, we consider how the eight car query imagery matches against image database A, the largest database.

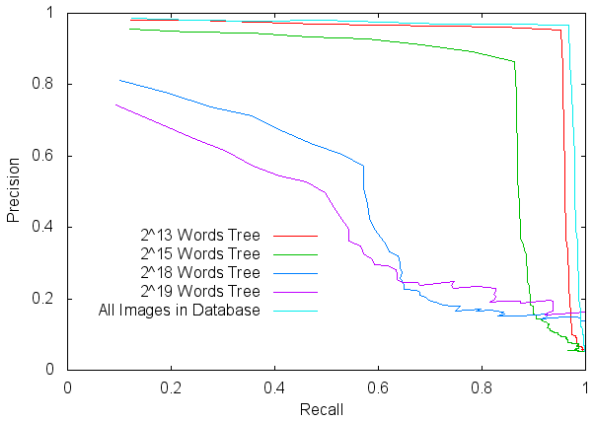

Figure 5. Precision and Recall for searches with structural verification using database B.

#4	GC	BV	BS	GM	MC	CD	RF	YS
GC	69	7	1	0	0	0	1	2
BV	5	83	0	0	3	0	0	1
BS	5	5	93	1	0	2	3	1
GM	3	2	0	98	1	2	2	1
MC	2	1	2	1	95	0	0	0
CD	8	2	2	0	1	94	2	2
RF	5	0	0	0	0	2	92	1
YS	3	0	2	0	0	0	1	92

Table 2. Confusion matrix for accuracy on structurally verified 4 top matches

#50	GC	BV	BS	GM	MC	CD	RF	YS
GC	91	1	0	0	0	0	0	0
BV	1	97	0	0	0	0	0	0
BS	0	1	100	0	0	0	0	1
GM	1	0	0	100	0	0	0	0
MC	1	0	0	0	100	0	1	0
CD	2	1	0	0	0	100	0	0
RF	3	0	0	0	0	0	99	0
YS	1	0	0	0	0	0	0	99

Table 3. Confusion matrix for accuracy on structurally verified 50 top matches

	1 match	10 matches	30 matches
2^{13} words	65.75%	81.63%	88.25%
2^{15} words	67.75%	85.75%	91.38%
2^{18} words	65.13%	85.13%	92.2%

Table 4. Average accuracy of searches with different vocabularies and varying numbers of structurally verified matches.

There are 20 car types and 2650 images in this database (Table 4). The extra car categories provide more opportunities for incorrect matches, stressing Car-Rec's performance more than the 8 car database.

The accuracy for all three vocabularies is over 80% with 10 Lucene matches. These results show Car-Rec's performance is scalable to databases containing thousands of images.

The smallest tree has the lowest accuracy when more than one Lucene result is structurally verified. The differences between the moderately-sized tree (2^{15} words) and the large tree (2^{18} words) are subtle. The moderate-sized vocabulary performing slightly better with fewer Lucene matches, and the larger vocabulary performing better with 30 structurally verified Lucene results. These results suggest that, while a vocabulary should not be very small, there are diminishing returns for ever-increasing vocabulary sizes.

	2^{15} Words	2^{18} Words
Feature Extraction (ms)	126.00	124.88
Word Quantization (ms)	0.91	1.012
Lucene Search (ms)	96.42	96.11
Structural Matching (ms)	97.46	100.37
Total Search (ms)	320.79	322.41

Table 5. Average search times for different vocabularies, broken down by framework stage.

5.3. Search Times

Search times are measured across the various stages of Car-Rec search. This benchmark was run on image database A (2650 images), with a set of 800 query images. The structural verification stage re-ranked 30 top Lucene results. Table 5 shows average measurements for various stages of the algorithm, along with the average total time.

Real-time performance is achieved in Car-Rec, with single image searches taking an average of 321 ms with 30 structural matches. Here, we consider real-time to mean the system works fast enough to facilitate human response in a physical security situation. The number of structural matches may be varied to suit the performance needs of the user. Car-Rec can be expected to run about 7-8 frames per second at best, with an average of approximately 3 frames per second on the test computer. Results may vary depending on the number of features extracted from query and database imagery.

6. Conclusions

Car recognition is a challenging problem, but one with many potential solutions. Car-Rec exploits the non-deformable physical structure and unique local features of cars for recognition. The experimental results show that this system provides strong performance in recognition accuracy and search times.

One of Car-Rec's strengths is its modularity. Most stages of the Car-Rec framework may be modified to utilize different algorithms in a straightforward fashion: SIFT may be used in place of SURF feature extraction, or Lucene search and the structural verification algorithm may be swapped out for alternative choices. This modular framework can be applied to general image recognition tasks, particularly with similar image search in large image databases.

Handling imagery with background noise information is an important area for further development. We are currently working with a prototype motion detection algorithm for segmenting moving cars from the background. Alternatively, where motion information is not available, a car detection algorithm may be used to localize the car in a scene.

Can Car-Rec's accurate performance scale up to large databases of hundreds or thousands of different cars models? This is impossible to answer with certainty until tests with such databases are performed, but larger databases may require larger vocabularies. The trade-off to larger vocabularies is increased mismatches between similar descriptor vectors. We believe Car-Rec might work reasonably well with high-quality databases of 100-200 cars, but match accuracy will eventually decrease to unreasonable levels. Future development in Car-Rec should take this potential limitation into consideration.

Finally, Car-Rec must be tested and developed with real car imagery. Toy cars may appear surprisingly realistic, but should only be treated as a temporary substitute for the real thing.

With the strong experimental results and high versatility, continued development on Car-Rec should result in a viable system for real-world use.

References

[1] Apache Lucene. lucene.apache.org/java/.

[2] Flickr Photo Sharing, 2010. www.flickr.com.

[3] Tineye Reverse Image Search, 2010. www.tineye.com.

[4] D. H. Ballard. Generalizing the Hough Transform to Detect Arbitrary Shapes. *Readings in C. V.*, pp. 714–725, 1987.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[6] J. S. Beis and D. G. Lowe. Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces. *IEEE Conf. C. V. Patt. Recog.*, pp. 1000-1006, 1997.

[7] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual Categorization with Bags of Keypoints, *Worksh. on Stat. Learn. in C.V., ECCV*, pp. 1–22, 2004.

[8] C. Evans. Notes on the OpenSURF Library, Jan. 2009. http://www.cs.bris.ac.uk/publications/papers/2000970.pdf.

[9] A. Ferencz, E. G. Learned-Miller, and J. Malik. Building a Classification Cascade for Visual Identification from One Example. *ICCV 05*, pp. 286293, Wash., DC, 2005.

[10] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[11] D. Nister and H. Stewenius. Scalable Recognition with a Vocabulary Tree. *CVPR 06*, pp. 2161–2168, Wash., DC, 2006.

[12] Y. Shan, H. S. Sawhney, and R. Kumar. Vehicle Identification between Non-Overlapping Cameras without Direct Feature Matching. *ICCV 05*, pp. 378-385, Wash., DC, 2005.

[13] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. *ICCV 03*, pp. 1470−1477, Wash., DC, 2003.