# WORD-HOGS: WORD HISTOGRAM OF ORIENTED GRADIENTS FOR MOBILE VISUAL SEARCH

*Sam S. Tsai, Huizhong Chen, David Chen, and Bernd Girod*

Information Systems Laboratory, Stanford University, Stanford, CA 94305, USA

**ABSTRACT**

Despite being a highly distinctive feature, the potential of text in images for mobile visual search has been largely neglected. Our research reported in this paper strives to improve mobile visual search by incorporating algorithms and compact representations tailored to visual text. We develop a new word patch descriptor, called Word Histogram of Oriented Gradients (Word-HOGs). The descriptor is based on gradient orientation histograms and can be compressed to a very small size using context-based arithmetic coding with lattice quantization. Because of its special structure, we can build image features from the descriptor and use these features for large-scale word patch matching. We show that the Word-HOG descriptor has a word patch matching performance that is better or comparable to the state-of-the-art approaches while being more efficient in feature counts, and it can be highly compressed with negligible loss in retrieval performance..

***Index Terms***— Mobile visual search, Visual text

## 1. INTRODUCTION

Text in images is highly distinctive because it has been designed that way. However, in mobile visual search applications, visual text information is usually treated like any other part of the image. Typically, interest points, such as Laplacian of Gaussian (LoG) [1] or Difference of Gaussian (DoG) [2] points, are detected and descriptors characterizing the image around each interest point are extracted. Then, the descriptors are compressed and sent to an image database for visual search. For text, the detected interest points are at a very small scale or at repetitive locations. When extracting descriptors from these interest points, many descriptors are very similar. Thus, this approach results in a representation that is inefficient and often ineffective. Thus, a new descriptor tailored to visual text is needed..

Our new descriptor should perform well for matching visual text information. It should also be efficient for finding matches in large-scale databases. Furthermore, the size of the descriptor should be compact and the generation or extraction of the descriptor should be computationally efficient. For mobile visual search applications, the client is a camera-phone capable of taking pictures and processing the pictures, but with limited memory. Thus, large visual search databases are placed on a remote server. Hence, query data are sent from the mobile client to the server for matching. It is critically important to reduce the amount of query data sent to minimize the system transmission latency and device power usage [3].

Several approaches to describe word patches have been proposed. In [4], binarized word images were compared with one another based on template matching, which the authors referred to as word spotting. Later, features based on projection profiles [5, 6, 7], foreground or stroke density [8, 9, 10] have been developed. These methods are mostly designed for matching historical scripts by using dynamic time warping [5] or elastic matching [11] to deal with the variations in the script. However, most of these require segmenting the font from the background and are sensitive to incorrect segmentation caused by variation in lighting. Features that do not need to distinguish text from background include wavelet features [12], directional patterns [13], and histogram of oriented gradients [14, 15]. However, the matching methods of these approaches do not scale to large databases. Local image feature-based approaches to word patch matching were investigated [16] for historical printed text. In [17], Schroth combined image features with detected character positions. However, in [16] and [17], an additional detection step is needed after the word patch location is given. While performing well for matching, these methods do not address the query size efficiency. Another possible solution is using Optical Character Recognition (OCR) [18, 19] and the recognized text for description. However, for mobile visual search, this requires running an OCR engine on the mobile device.

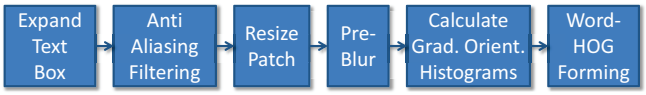
In this paper, we present a new word patch descriptor for mobile visual search. We develop a new descriptor called Word Histogram of Oriented Gradients (Word-HOGs). The Word-HOG is a descriptor that is based on gradient orientation histograms. The descriptor is a concatenation of gradient orientation histograms from sub-blocks within a word patch. From the Word-HOG descriptor, we generate SIFT-like [2] descriptors, which we call WSIFT, and use the Vocabulary Tree (VT)-based approach [20] to perform word patch matching. We show the matching pipeline performs reliable matching while using fewer features than other approaches. We note Word-HOG is similar to [14, 15], however, we differ in how gradient orientation histograms are generated and how word patch matching is performed. Furthermore, we develop a novel compression scheme for Word-HOGs. We use lattice coding [21, 22, 23] to quantize the descriptor and use a context-based arithmetic coder to compress the query. Compressed Word-HOGs performs word patch matching at a high accuracy with only a few tens of bytes.

The rest of the paper is organized as follows. Sec. 2 describes how Word-HOGs are extracted from word patches and how they are compressed. Then, Sec. 3 shows how Word-HOGs are used for word patch matching. After that, Sec. 4 presents the experimental results of the proposed approach.

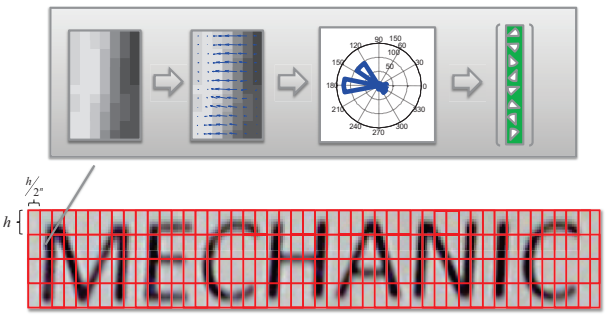
## 2. WORD-HOGS FOR WORD PATCHES

### 2.1. Descriptor Generation

Fig. 1(a) shows a diagram of the Word-HOG extraction process. Given an image, we assume that a rectangular box is tightly fit around the text, where the box is typically found using a text detection algorithm [24]. The box is expanded by a factor of $1/m$, and a word patch is extracted from the location of the expanded box in the image. Then, the patch is resized to a height of $H$ pixels while the aspect ratio is kept the same. Pre-blurring is applied to the word patch using a Gaussian filter, similar to [2]. After that, the patch is divided vertically into 4 rows, as shown in Fig. 1(b), and each row is horizontally divided into sub-blocks that are $1/2^n$ of the row height, where $n$ is an integer ($> 0$). For each sub-block, image gradients are calculated and used to generate a gradient orientation histogram with 8 bins. Both the number of rows and the dimensions of the histogram were determined from patch matching experiments. Once the gradient orientation histogram of each sub-block is generated, the Word-HOG is formed by concatenating the histograms starting from the top-left sub-block and following an order from top to bottom and left to right. Sec. 4.1 will describe how the parameters presented here are trained.




(a) Word-HOG extraction diagram




(b) Gradient histograms are extracted from sub-blocks

**Fig. 1**: The Word-HOG descriptor is generated from gradient orientation histograms of sub-blocks in the word patch.

The Word-HOG descriptor is a variable length descriptor; its length depends on the width of the word patch. Additionally, since the descriptor is formed by gradient orientation histograms, the descriptor can be efficiently compressed using lattice coding techniques [21, 22, 23], as described in the next section.

### 2.2. Lossy Compression of Word-HOGs

To compress the Word-HOG descriptor, lattice coding is first used to lossily compress the histogram of each sub-block. Then, context-based arithmetic coding is used to encode the lattice indices.

**Lattice Coding**: The gradient orientation histogram is the distribution of the image gradient directions within a sub-block. For a normalized gradient orientation histogram of dimension $m$, the histogram vector lies on a probability simplex in $m$-dimensional space. Lattice coding quantizes these histogram vectors to lattice points in the probability simplex using the method described in [23], where a quantization parameter, $n$, controls the density of points on the simplex. The lattice points within the probability simplex are enumerated, hence, only an index is needed to represent the quantized distribution.

**Context-based Arithmetic Coding**: Once the Word-HOG descriptor is quantized using lattice coding, a set of lattice indices is produced. These indices are compressed using entropy coding to further reduce the query size. Since the sub-block size is typically smaller than the character strokes, a single stroke would appear in several sub-blocks. For sub-blocks that contain the same stroke, the direction of the image gradients is similar. We exploit this relationship in sub-blocks by using context-based arithmetic coding. Starting from the top-left, the indices of each sub-block are entropy coded while using the previously coded horizontal sub-block as context, as shown in Fig. 2.

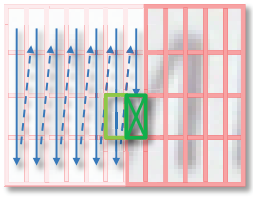


**Fig. 2**: The horizontal neighboring sub-block is used as context for the context-based arithmetic coder for Word-HOGs.

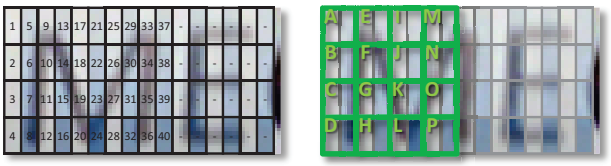
## 3. WORD PATCH MATCHING WITH WORD-HOGS

To enable fast comparisons in large-scale word patch matching, WSIFTs are generated from Word-HOGs and used in a VT to retrieve word patches. Sec. 3.1 describes how WSIFT descriptors are generated, and Sec. 3.2 explains how they are used in VTs.

### 3.1. Generating WSIFT Descriptors

WSIFT is a gradient orientation histogram-based descriptor generated from a $4 \times 4$ spatial grid with square spatial bins, similar to SIFT [2]. Within each spatial bins, image gradients are calculated and used to generate a gradient orientation histogram. Since Word-HOGs are constructed of gradient orientation histograms from rectangular sub-blocks, they can be used to assemble WSIFT descriptors. Let us consider the generation of the first WSIFT descriptor in the left-most position of a word patch as shown in green in Fig. 3(b). To generate the histogram for spatial bin A, we add up the histograms of sub-blocks 1 and 5 of the Word-HOG, as labeled in Fig. 3(a). Similarly, for spatial bin B, we add up the histograms of sub-blocks 2 and 6. The process is repeated for all $4 \times 4$ spatial bins. Then, the histograms are concatenated into to a 128-dimensional vector, and the vector is normalized to have unit length forming the WSIFT descriptor. After that, we move one horizontal sub-block step to the right to generate the next WSIFT descriptor. We generate a series of WSIFT descriptors along the horizontal direction.

Readers familiar with the implementation details of SIFT will notice some subtle differences. We do not use magnitude weighting



(a) Sub-block indices of word patch (b) Location of Left-most WSIFT

**Fig. 3**: (a) Part of the original image overlaid with sub-block indices. (b) Spatial grid of the first WSIFT descriptor

but use only the gradient orientation to generate histogram directly. We do not use soft binning for spatial bins but only for angular bins. Furthermore, we omit the Gaussian weighting that de-emphasizes the bins further away from the detected SIFT keypoints.

### 3.2. Matching with Vocabulary Trees

Given a set of word patches, we first train a word patch matching database. Word-HOGs are extracted from every word patch and used to generate their WSIFT descriptors. Then, hierarchical K-means clustering (or Tree Structured Vector Quantization) is used to build a VT with depth of $D$ on the database WSIFT descriptors. After that, we quantize all the WSIFT descriptors with the VT using a greedy scheme [25] to generate the inverted indices used for fast lookup [20]. This step completes the database training process.

A query is initiated when a query word patch is given. On the client, a query word patch is extracted from the image and the Word-HOG descriptor is extracted from the word patch. Then, the Word-HOG is compressed and transmitted to the server where the database is located. On the server, the Word-HOG is decoded and the WSIFT descriptors are generated from the Word-HOG. Lastly, the WSIFT descriptors are quantized through the VT and the inverted indices are used to generate a list of matching database word patches.

From the VT matching, a list of ranked matching patches is generated. However, since the VT matching stage uses only image features and not their locations, the matching does not guarantee features are in geometrically consistent order. Thus, words that have the same set of characters but in different order can be highly ranked in the list. To differentiate them, we use Geometric Verification (GV) to ensure that the matching features are geometrically consistent [26].


**Fig. 4**: Block diagram of the geometric verification process.

We use a GV process as shown in Fig. 4. Similar to previous approaches [2], features from two feature sets are paired based on the descriptor $L_2$ distance. But, instead of using distance ratio test to reject false matches, distance threshold test is used with threshold $\theta$. This helps retain similar feature matches that appear in reoccurring letters in the same word. To further rule out the false matches, longest common sequence search is performed on the matching feature pairs [16]. Lastly, a horizontal line projection model is estimated from the locations of the matching feature pairs. The matching GV score is given as $c_1 \times c_2$, where $c_1$ and $c_2$ is the percentage of features that have matched up between the two images. This score is used to rerank the final list.

## 4. EXPERIMENTAL RESULTS

This section shows experimental results on large-scale word patch matching using a database of 960K word images. These word images are generated synthetically with Arial font using a combined dictionary from Tesseract [1] and SCOWL [2] with a total of 400K words. We generate lower case (except for names), upper case, and sentence case versions of each word to obtain the set of 960K images. The adoption of the Tesseract dictionary is for the purpose of comparison. From the dictionary, we randomly select 500 words and print them on paper and take pictures of the words using camera-phones to generate query word patches. In total, we generate 1000 training word patches and 1000 testing word patches. Sample word patches are shown in Fig. 5. To obtain the tight boxes in the query word patches, we perform adaptive binarization [27] on the image and fit a rectangular box to the connected components.

We evaluate the performance of the word patch matching by looking at the retrieval performance of searching the same word from the large database of word patches given a query word patch. We calculate the recall at different ranks of the retrieved list and the precision of the top match in the retrieved list.

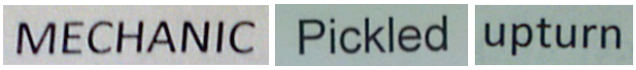



**Fig. 5**: Example query word patches.

### 4.1. Training

We train the Word-HOG descriptor and the word patch matching pipeline using the 1000 training query word patches and a subset of the database image with 120K images. Optimizing for accuracy and speed, we find the best parameters for the Word-HOG descriptor as follows: the expanded box margin $m = 0.95$, the resized patch height before Word-HOG extraction $H = 32$, the pre-blurring Gaussian filter has $\sigma = 3$, and the sub-block width is $1/2^n$ with $n = 1$. The best parameter for the VT tree depth $D = 4$ while the distance threshold for GV $\theta = 0.6$. These settings are used for the remaining experiments.

### 4.2. Large-scale Word Patch Matching

Fig. 6 shows the retrieval performance at two stages within the word patch matching pipeline. The VT uses only image descriptors hence reorders similarity based on appearance. Thus, after VT matching, the accuracy at the top is fairly low. In the GV step, we rerank the 1000 top scoring entries. With GV, we enforce the order of the matching features and improve the performance. Overall, we achieve a recall of 0.84 for the top match. The average overall retrieval process time is a couple hundred milliseconds on a server using a single core of a 2.4GHz Xeon processor.

[1] Tesseract Open Source OCR, http://code.google.com/p/tesseract-ocr/
[2] Spell Checker Oriented Word List, http://wordlist.sourceforge.net/

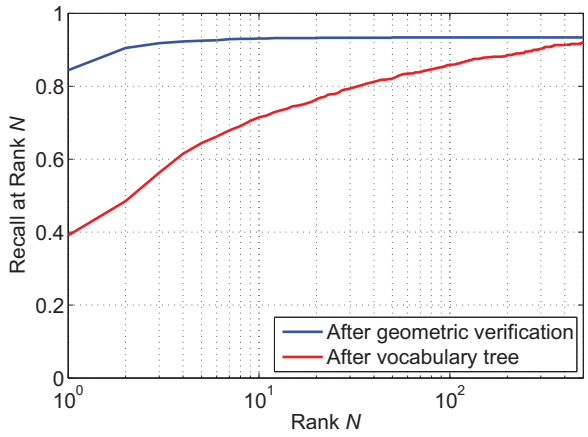



**Fig. 6**: Retrieval performance at two stages in the pipeline.

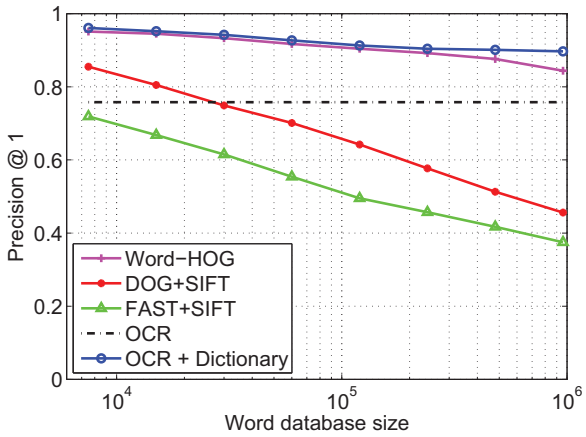



**Fig. 7**: Comparison of various word patch matching methods.

### 4.3. Retrieval Performance Comparisons

We compare the proposed pipeline with the following additional methods. (1) **FAST+SIFT**: we extract SIFT descriptors from FAST interest points in the image and use the VT retrieval pipeline with GV. This is based on [16]. (2) **DOG+SIFT**: we extract SIFT from DoG interest points and use the pipeline in [26] for matching. (3) **OCR**: Tesseract OCR is used to recognized the word patch. (4) **OCR+Dictionary**: Words from a dictionary that only contains the words in the database are retrieved using the recognized characters from OCR and ranked based on editing distance.

Fig. 7 shows the retrieval performance of the described methods with database sizes ranging from 7.5K to 960K. The figure also shows the recognition accuracy of Tesseract, which is $\sim 0.76$. **FAST+SIFT** performs as well as Tesseract for small databases, but not as well when the database is large. **DOG+SIFT** performs better than **FAST+SIFT**, but the trend is similar. Our approach is better than **DOG+SIFT** and **FAST+SIFT** and close to the approach of using **OCR+Dictionary**, which performs the best.

For the three methods that use the VT for matching words, we look at the number of features used in the VT matching. The average number of query image features used is $88, 60, 33$ for **FAST+SIFT**, **DOG+SIFT**, **Word-HOG**, respectively. With just half of what **FAST+SIFT** or **DOG+SIFT** is using, **Word-HOG** can perform much better in retrieval.

### 4.4. Matching with Lossy Compression

We further want to investigate how the query data size affects the final word matching or recognition performance. We set up three systems that use different query data to perform word matching for comparison: (1) **Compressed Word-HOG**: We use compressed Word-HOG as query data with different quantization parameters that results in different sizes. (2) **Compresed WSIFT**: We use compressed WSIFT as query data. The WSIFT is extracted from the Word-HOG and compressed using lattice coding. The same retrieval pipeline is used for (1) and (2). (3) **JPEG**: We use JPEG compressed word patch as query data. For recognition, we use Tesseract to recognize the text. Then, the recognized text is used to retrieve words from a dictionary containing only the words in the database.

The matching performances retrieving from a database of 120K patches of the three methods are show in Fig. 8. We see **Compressed Word-HOG** performs well at low bit rates. At a query size of 50 bytes, the recognition accuracy is 0.87. The **Compressed WSIFT** query size is generally $> 7$ times larger than the **Compressed Word-HOG** query size. **JPEG** performs the worst when the query size is small. For reference, **OCR+Dictionary** shows the retrieval result of using the recognized word from the original word patch with dictionary lookup.

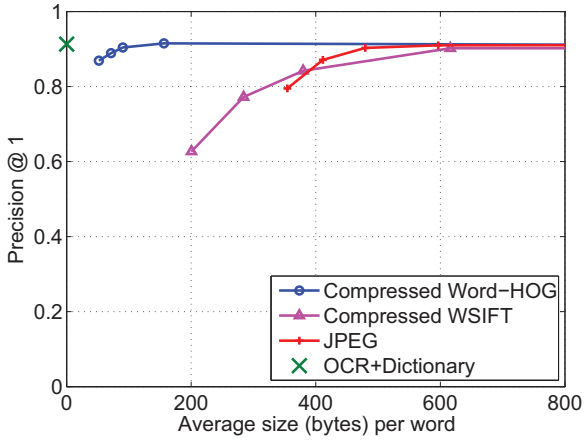



**Fig. 8**: Retrieval performance vs. query data size.

## 5. CONCLUSIONS

We have developed a new word patch descriptor for mobile visual search, called Word-HOG, or Word Histogram of Oriented Gradients. The descriptor is gradient orientation histogram-based which we compress efficiently using lattice coding with a context-based arithmetic coder. The descriptor is used for word patch matching with vocabulary tree and geometric verification. The matching pipeline has a retrieval performance that is much better than state-of-the-art algorithms for general visual features and is roughly on par with OCR followed by dictionary lookup. However, computing Word-HOGs is much faster than performing Tesseract OCR, an important advantage for mobile implementations. The newly developed descriptor is highly compressible, at a query size of just tens of bytes, there is only a negligible drop in matching performance.

## 6. REFERENCES


[1] Tony Lindeberg, "Feature detection with automatic scale selection," *International journal of computer vision*, vol. 30, no. 2, pp. 79–116, 1998.

[2] David G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] Bernd Girod, Vijay Chandrasekhar, David M. Chen, Ngai-Man Cheung, Radek Grzeszczuk, Yuriy Reznik, Gabriel Takacs, Sam S. Tsai, and Ramakrishna Vedantham, "Mobile visual search," *Signal Processing Magazine, IEEE*, vol. 28, no. 4, pp. 61–76, 2011.

[4] Raghavan Manmatha, Chengfeng Han, and Edward M. Riseman, "Word spotting: A new approach to indexing handwriting," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1996, pp. 631–637.

[5] Toni M. Rath and Raghavan Manmatha, "Word image matching using dynamic time warping," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2003, vol. 2, pp. 521–527.

[6] Tony M. Rath and Rudrapatna Manmatha, "Word spotting for historical documents," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 9, no. 2-4, pp. 139–152, 2007.

[7] Thomas Konidaris, Basilios Gatos, Kostas Ntzios, Ioannis Pratikakis, Sergios Theodoridis, and Stavros J. Perantonis, "Keyword-guided word spotting in historical printed documents using synthetic data and user feedback," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 9, no. 2-4, pp. 167–177, 2007.

[8] Gaofeng Meng, Nanning Zheng, Yonghong Song, and Yuanlin Zhang, "Document images retrieval based on multiple features combination," in *International Conference on Document Analysis and Recognition*. 2007, vol. 1, pp. 143–147, IEEE.

[9] Yaodong He, Zao Jiang, Bing Liu, and Hong Zhao, "Content-based indexing and retrieval method of chinese document images," in *International Conference on Document Analysis and Recognition*. 1999, pp. 685–688, IEEE.

[10] Chew Lim Tan, Weihua Huang, Zhaohui Yu, and Yi Xu, "Imaged document text retrieval without ocr," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 6, pp. 838–844, 2002.

[11] Yann Leydier, Frank Lebourgeois, and Hubert Emptoz, "Text search for medieval manuscript images," *Pattern Recognition*, vol. 40, no. 12, pp. 3552–3567, 2007.

[12] Ediz Saykol, Ali Kemal Sinop, Ugur Gudukbay, Ozgr Ulusoy, and A. Enis etin, "Content-based retrieval of historical ottoman documents stored as textual images," *Image Processing, IEEE Transactions on*, vol. 13, no. 3, pp. 314–325, 2004.

[13] Hiromichi Fujisawa and Cheng-Lin Liu, "Directional pattern matching for character recognition revisited," *algorithms*, vol. 13, pp. 14, 2010.

[14] José A. Rodríguez and Florent Perronnin, "Local gradient histogram features for word spotting in unconstrained handwritten documents," in *International Conference on Frontiers in Handwriting Recognition*, 2008.

[15] Kengo Terasawa and Yuzuru Tanaka, "Slit style hog feature for document image word spotting," in *International Conference on Document Analysis and Recognition*. 2009, pp. 116–120, IEEE.

[16] Ismet Zeki Yalniz and Raghavan Manmatha, "An efficient framework for searching text in noisy document images," in *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*. 2012, pp. 48–52, IEEE.

[17] G. Schroth, S. Hilsenbeck, R. Huitl, F. Schweiger, and E. Steinbach, "Exploiting text-related features for content-based image retrieval," in *Multimedia (ISM), 2011 IEEE International Symposium on*. 2011, pp. 77–84, IEEE.

[18] "Tesseract ocr engine," http://code.google.com/p/tesseract-ocr/.

[19] Sam S. Tsai, Huizhong Chen, David Chen, Vasu Parameswaran, Radek Grzeszczuk, and Bernd Girod, "Visual text features for image matching," in *IEEE International Symposium on Multimedia*. 2012, pp. 408–412, IEEE.

[20] David Nistér and Henrik Stewénius, "Scalable recognition with a vocabulary tree," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006, vol. 2, pp. 2161–2168.

[21] Vijay Chandrasekhar, Gabriel Takacs, David Chen, Sam Tsai, Radek Grzeszczuk, and Bernd Girod, "Chog: Compressed histogram of gradients a low bit-rate feature descriptor," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 2504–2511.

[22] Vijay Chandrasekhar, Yuriy Reznik, Gabriel Takacs, David Chen, Sam Tsai, Radek Grzeszczuk, and Bernd Girod, "Quantization schemes for low bitrate compressed histogram of gradients descriptors," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. 2010, pp. 33–40, IEEE.

[23] Vijay Chandrasekhar, Gabriel Takacs, David M. Chen, Sam S. Tsai, Yuriy Reznik, Radek Grzeszczuk, and Bernd Girod, "Compressed histogram of gradients: A low-bitrate descriptor," *International Journal of Computer Vision*, vol. 96, no. 3, pp. 384–399, 2012.

[24] Huizhong Chen, Sam S. Tsai, Georg Schroth, David M. Chen, Radek Grzeszczuk, and Bernd Girod, "Robust text detection in natural images with edge-enhanced maximally stable extremal regions," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*. 2011, pp. 2609–2612, IEEE.

[25] Grant Schindler, Matthew Brown, and Richard Szeliski, "City-scale location recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–7.

[26] Sam S. Tsai, David Chen, Vijay Chandrasekhar, Gabriel Takacs, Ngai-Man Cheung, Ramakrishna Vedantham, Radek Grzeszczuk, and Bernd Girod, "Mobile product recognition," in *Proceedings of the international conference on Multimedia*. 2010, pp. 1587–1590, ACM.

[27] Nobuyuki Otsu, "A threshold selection method from gray-level histogram," *IEEE Trans.on Systems, Man, and Cybernetics*, vol. 9, pp. 62–66, 1979.