Tree Histogram Coding for Mobile Image Matching

David M. Chen[1], Sam S. Tsai[1], Vijay Chandrasekhar[1],
Gabriel Takacs[1], Jatinder Singh[2], and Bernd Girod[1]
[1]Information Systems Lab, Stanford University, Stanford, CA 94305
[2]Deutsche Telekom Inc. R&D Lab, Los Altos, CA 94022

Abstract

For mobile image matching applications, a mobile device captures a query image, extracts descriptive features, and transmits these features wirelessly to a server. The server recognizes the query image by comparing the extracted features to its database and returns information associated with the recognition result. For slow links, query feature compression is crucial for low-latency retrieval. Previous image retrieval systems transmit compressed feature descriptors, which is well suited for pairwise image matching. For fast retrieval from large databases, however, scalable vocabulary trees are commonly employed. In this paper, we propose a rate-efficient codec designed for tree-based retrieval. By encoding a tree histogram, our codec can achieve a more than $5\times$ *rate reduction compared to sending compressed feature descriptors. By discarding the order amongst a list of features, histogram coding requires* $1.5\times$ *lower rate than sending a tree node index for every feature. A statistical analysis is performed to study how the entropy of encoded symbols varies with tree depth and the number of features.*

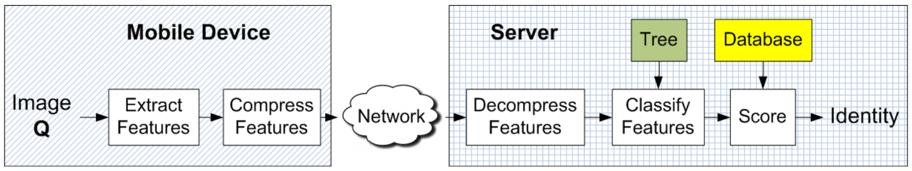
1. Introduction

Robust local image features are commonly used for content-based image retrieval (CBIR). Popular examples include the Scale-Invariant Feature Transform (SIFT) [1], Gradient Location and Orientation Histogram (GLOH) [2], and Speeded-Up Robust Features (SURF) [3]. Fast search through a large database can be achieved using hierarchical approaches such as Scalable Vocabulary Trees (SVTs) [4] or Pyramid Match Kernels (PMKs) [5]. Both the SVT and PMK train a classification tree using hierarchical k-means clustering of feature descriptors. Classification is then performed through tree-structured vector quantization (TSVQ) [6] and counting how often each tree node is visited. This condenses the set of feature descriptors into a tree histogram. Fast search is possible because of the conciseness of the quantized representation.

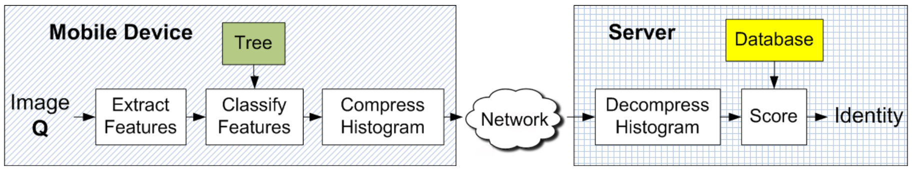
An emerging class of CBIR applications entails transmitting query feature descriptors from a mobile device, such as a camera-phone, over a wireless network to a remote server hosting the database. Compression of the query feature descriptors is necessary for low-latency retrieval. A feature descriptor codec is presented in [7], where each descriptor is transformed by a decorrelating transform, quantized, and entropy-coded using Huffman coding or arithmetic coding. Another codec in [8] generates a hash for each descriptor using random projections. Both systems employ the configuration of Fig. 1(a).

The codecs in [7] and [8] are well suited for pairwise image matching. Neither codec exploits the fact that, in tree-based retrieval, the tree histogram suffices for accurate classification. If the classification tree can be stored at the encoder, then the histogram can be transmitted in place of the descriptors. The compactness of the histogram enables significant bit rate savings compared

1068-0314/09 $25.00 © 2009 IEEE
DOI 10.1109/DCC.2009.33



(a)



(b)

Figure 1. (a) Retrieval pipeline for compressing features and then classifying them at the server. (b) Retrieval pipeline for classifying features on the mobile device and then compressing the tree histogram.

to direct descriptor coding. The retrieval pipeline now changes to Fig. 1(b). Importantly, the encoder-side tree prevents coding-induced quality degradations in feature descriptors prior to classification, as opposed to [7] or [8] which must coarsely quantize or hash the descriptors before transmission.

In this paper, we show that storage of classification trees on mobile devices is feasible and we propose tree histogram encoding in lieu of compressing feature descriptors. Sec. 2 reviews generation of a tree histogram and scoring for an SVT. Rate-efficient lossless encoding of the histogram is presented in Sec. 3. Additionally, Sec. 3 provides a detailed statistical analysis of the encoded symbols. Experimental results in Sec. 4 show that our proposed system requires far fewer bits, over $5\times$ less, to achieve the same matching accuracy as descriptor compression. Additionally, our unordered bag-of-words representation requires $1.5\times$ lower rate than sending a tree node index for every feature.

2. Background on Scalable Vocabulary Tree

The nodes of an SVT are centroids determined by hierarchical k-means clustering of sample feature descriptors. Suppose an SVT has depth D, i.e., there are D levels in the tree including the root node, and max branch factor K, i.e., an interior node has at most K children. A fully balanced tree will have K^L nodes at the L^{th} level, where $L \in \{0, \dots, D-1\}$, and an example is shown in Fig. 2 for $D = 3$ and $K = 3$. In practice, $D = 6$ or $D = 7$ and $K = 10$ are typical selections for good performance [4]. In large trees, the deeper levels can be partly unbalanced because the hierarchical k-means algorithm prunes away very small or degenerate clusters.

There is a simple hierarchical relationship between nodes at different levels of the SVT. Suppose the nodes at the L^{th} level are enumerated $n_0^L, \dots, n_{K^L-1}^L$, and similarly the nodes at the $(L+1)^{\text{st}}$ level are enumerated $n_0^{L+1}, \dots, n_{K^{L+1}-1}^{L+1}$, as exemplified in Fig. 2. Then, the children set of node n_i^L is given by $\mathbf{C}(n_i^L) = \{n_{i \cdot K}^{L+1},\ n_{i \cdot K+1}^{L+1},\ \dots,\ n_{i \cdot K+K-1}^{L+1}\}$.

An image $\mathbf{I}$ is classified by quantizing its feature descriptors according to the SVT by traversing from top to bottom and greedily choosing the nearest node at each level. [9]. We are interested in performing this search on a mobile device. Although a mobile device computes slower than a server, greedy search only requires $O(KD)$ computation per feature and is relatively efficient.

Let the visit count $\mathbf{I}_{n_i^L}$ denote the number of descriptors of $\mathbf{I}$ which are quantized to node n_i^L.

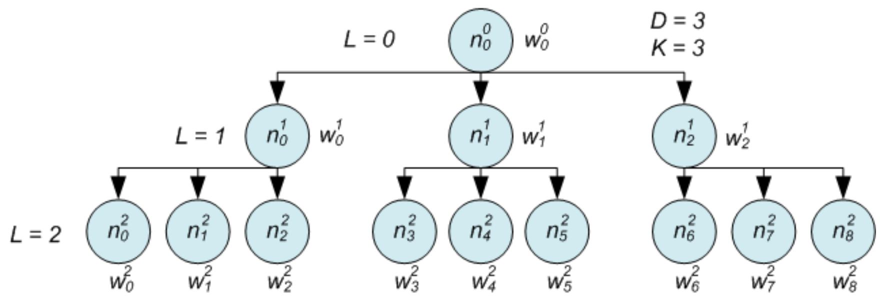

L = 0
L = 1
L = 2
D = 3
K = 3
n_0^0 w_0^0
n_0^1 w_0^1
n_1^1 w_1^1
n_2^1 w_2^1
n_0^2 n_1^2 n_2^2 n_3^2 n_4^2 n_5^2 n_6^2 n_7^2 n_8^2
w_0^2 w_1^2 w_2^2 w_3^2 w_4^2 w_5^2 w_6^2 w_7^2 w_8^2

Figure 2. Fully balanced SVT with depth $D = 3$ and max branch factor $K = 3$.

If n_i^L is a node pruned away by hierarchical k-means, $\mathbf{I}_{n_i^L} = 0$ always. The image is compactly summarized by the tree histogram, which is the set of all visit counts. Because a parent node is visited whenever one of its children is visited, a visit count at level L can be calculated from the visit counts at level $L+1$: $\mathbf{I}_{n_i^L} = \sum_{n_j \in \mathbf{C}(n_i^L)} \left(\mathbf{I}_{n_j} \right)$. The entire tree histogram can thus be reconstructed from the counts at the leaf level. This property enables very efficient coding of the histogram, as shown in Sec. 3.

Dissimilarity between two images is measured by comparing their tree histograms, after entropy weighting. Nodes visited by fewer images possess greater discriminative or informative value than nodes visited by many images. The SVT system in [4] assigns to node n_i^L the weight $w_i^L = \ln\left(N/N_i^L\right)$ for $N_i^L > 0$ and $w_i^L = 0$ for $N_i^L = 0$, where N is the total number of database images and N_i^L is the number of database images with at least one descriptor having visited node n_i^L. The weighted tree histogram for image $\mathbf{I}$ is a vector of all entropy-weighted visit counts through the entire SVT:

$$\bar{\mathbf{I}} \;=\; \left[w_i^L \cdot \mathbf{I}_{n_i^L} \right]_{L=0,1,\cdots,D-1 \text{ and } i=0,1,\cdots,k^L-1} . \tag{1}$$

For a query image $\mathbf{Q}$ and the m^{th} database image $\mathbf{D}_m$, whose weighted tree histograms are defined by replacing $\mathbf{I}$ in (1) with $\mathbf{Q}$ or $\mathbf{D}_m$, the mutual dissimilarity score is

$$d\left(\mathbf{Q}, \mathbf{D}_m\right) = \left\| \frac{\overline{\mathbf{Q}}}{\|\overline{\mathbf{Q}}\|_1} - \frac{\overline{\mathbf{D}_m}}{\|\overline{\mathbf{D}_m}\|_1} \right\|_1 . \tag{2}$$

By measuring (2) for each database image, the closest matches can be determined.

3. Compression of Tree Histograms

A prerequisite for tree histogram coding is the storage of an SVT in the memory of the mobile device. First, in Sec. 3.1, we show the memory requirement can be fulfilled by mobile devices on the market today. Then, a rate-efficient tree histogram encoder will be presented in Sec. 3.2, followed by a statistical analysis in Sec. 3.3.

3.1. Storage of Classification Tree on Mobile Devices

Each SVT node is represented by a centroid of the same dimensionality as the feature descriptor. In a tree of depth D and max branch factor K, the maximum number of nodes is $(K^D - 1)/(K - 1)$, achieved by a fully balanced tree. SURF descriptors are 64-dimensional and require half the memory of 128-dimensional SIFT descriptors. For this important reason, we choose SURF instead of SIFT for our retrieval system.

Table 1. Memory requirements for SVTs and availability on several mobile phones.

Tree Structure	Memory Requirement
SURF SVT, $D = 6$, $K = 10$	7 MB
SURF SVT, $D = 7$, $K = 10$	68 MB

Mobile Phone	RAM	Memory Card
Nokia N95	64 MB	8 GB
Sony Ericsson W960	128 MB	8 GB
Samsung i8510	128 MB	8 GB
Palm Treo Pro	128 MB	8 GB
Apple iPhone	128 MB	8 GB

Table 1 lists the memory requirements for SVTs of max branch factor $K = 10$ and depth $D = 6$ or $D = 7$. Each centroid is represented with 8-bit precision per dimension. Also listed are the memory capacities for several mobile phones, each equipped with a digital camera [10]. Only one tree, designed for the database the user is currently interested in, needs to reside in random access memory (RAM). For all of the listed phones, a tree of depth $D = 6$ and max branch factor $K = 10$ fits in RAM, and for all but one, a tree of depth $D = 7$ also fits in RAM. With memory availability continuously improving, next generation mobile devices will more easily meet the storage requirement.

Trees built for other image databases not required for the current application can be stored on a memory card, where memory is more much abundant than in RAM. Further memory conservation can be achieved using a recently proposed lossless memory encoding scheme [11], allowing savings as much as 60 percent. Alternatively, a tree can be pruned by the BFOS algorithm [12] to a smaller subtree.

The server and the mobile device must agree on the same version of the SVT. Initially, an SVT can be trained on the server using a large image database and sent to the mobile device. Over time, if newer images are added to the database, the SVT on the server can be efficiently updated using the scheme in [13]. The incremental modifications to the SVT in [13] can also be compactly communicated to the mobile device for synchronization.

3.2. Runlength Encoding of Tree Histogram

Our rate-efficient lossless encoder for the tree histogram exploits the hierarchical relationship between the nodes discussed in Sec. 2. The leaf histogram is sufficient for reconstructing the entire tree's histogram. Thus, our encoder transmits only the leaf histogram, while the decoder can easily reconstruct histograms at higher levels.

A typical query image $\mathbf{Q}$ produces several hundred SURF features. By comparison, a fully balanced SVT of depth $D = 7$ and max branch factor $K = 10$ contains $N_{\text{leaf}} = 10^6$ leaf nodes. Therefore, most of the leaf nodes are not visited by the query image, resulting in many zero counts. This motivates the encoding of the length of zero-runs between positive-count nodes. The sequence of symbols for communicating the positive-count node identities is

$$R_1 = n_{1^+}^{D-1},\ R_2 = n_{2^+}^{D-1} - n_{1^+}^{D-1},\ \ \ldots,\ \ R_{N^+} = n_{N^+}^{D-1} - n_{N^+-1}^{D-1}\ , \tag{3}$$

where $n_{i^+}^{D-1} \in \{1, 2, \ldots, N_{\text{leaf}}\}$ is the i^{th} positive-count leaf node and N^+ is the total number of positive-count leaf nodes visited by the descriptors of $\mathbf{Q}$. A second sequence of symbols of the same length encodes the actual counts at these positive-count nodes:

$$S_1 = \mathbf{Q}_{n_{1^+}^{D-1}},\ S_2 = \mathbf{Q}_{n_{2^+}^{D-1}},\ \ \ldots,\ \ S_{N^+} = \mathbf{Q}_{n_{N^+}^{D-1}}\ . \tag{4}$$

The R and S sequences are further encoded by an arithmetic coder [14].

Since the tree histogram is a bag-of-words representation, the order among the feature descriptors is discarded. There are $N_{\text{desc}}!$ different orders among N_{desc} descriptors. If each order is equally likely, then it would take $\log_2(N_{\text{desc}}!)$ bits to communicate a particular order. By Stirling's approximation, for large N_{desc},

$$\log_2(N_{\text{desc}}!) \approx N_{\text{desc}}\left(\log_2(N_{\text{desc}}) - 1/\ln 2\right) + 0.5\log_2(N_{\text{desc}}) \quad . \tag{5}$$

An alternative codec that transmits the node identity for every descriptor would have to spend this extra number of bits, which our codec successfully avoids.

3.3. Statistical Analysis of Encoded Symbols

For our analysis, we first model how likely each leaf node is to be visited by a new feature descriptor. Fig. 3 shows a histogram measured for the Zurich Building Database (ZuBuD) and CD Database (CDD) images. The contents of both databases will be explained in Sec. 4. Although this distribution is difficult to model, a uniform distribution is an acceptable approximation. Further, we assume that feature descriptors are independently classified. As we will show in this section, these assumptions lead to reasonably accurate models for the R and S symbols.

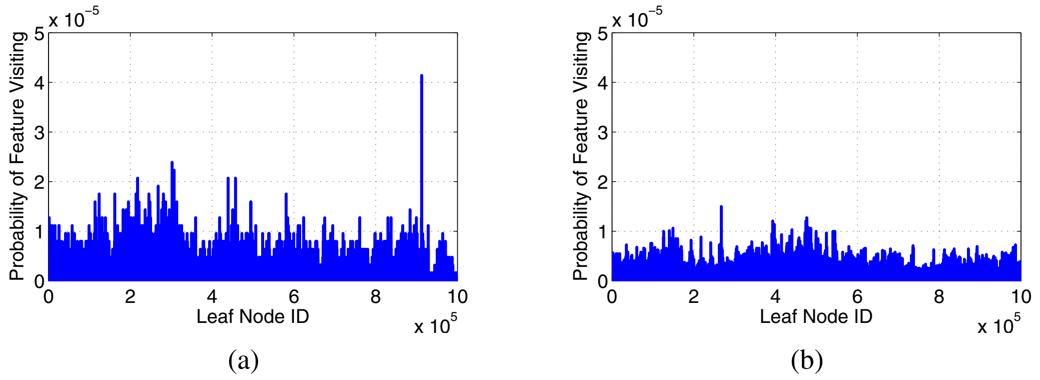


Figure 3. Probability of visiting each leaf node in an SURF SVT for (a) ZuBuD and (b) CDD images.

3.3.1. Distributions for Positive-Count Node Identities. Suppose the query image **Q** has N_{desc} feature descriptors and the SVT has N_{leaf} leaf nodes. Assuming the leaf nodes are visited uniformly and the descriptors are classified independently, the probability that a descriptor reaches a particular leaf node is

$$p_{\text{visit}} = 1/N_{\text{leaf}} \quad . \tag{6}$$

The probability that a leaf node is not visited by any of the N_{desc} descriptors is

$$p_{\text{empty}} = (1 - p_{\text{visit}})^{N_{\text{desc}}} \quad . \tag{7}$$

Each runlength symbol R_i represents the number of consecutive leaf bins that must be visited, starting at the previous nonempty leaf bin, before reaching the next nonempty leaf bin. If $R_i = r$, then $r - 1$ empty leaf bins and 1 nonempty leaf bin are observed in the process. For large N_{leaf}, R_i is nearly a geometric random variable:

$$\begin{aligned} p_{R_i}(r) &= \frac{(p_{\text{empty}})^{r-1}(1-p_{\text{empty}})}{\sum_{r'=1}^{N_{\text{leaf}}}(p_{\text{empty}})^{r'-1}(1-p_{\text{empty}})} \quad , \\ r &\in \{1, 2, \cdots, N_{\text{leaf}}\} \quad , \end{aligned} \tag{8}$$

where the denominator of $p_{R_i}(r)$ quickly approaches 1 as N_{leaf} becomes large. The model PMF $p_{R_i}(r)$ is plotted in Fig. 4 for $i = 1, 50$ and for parameters specific to ZuBuD and CDD. Also shown are the actual histograms for ZuBuD and CDD, which agree well with the model PMFs.

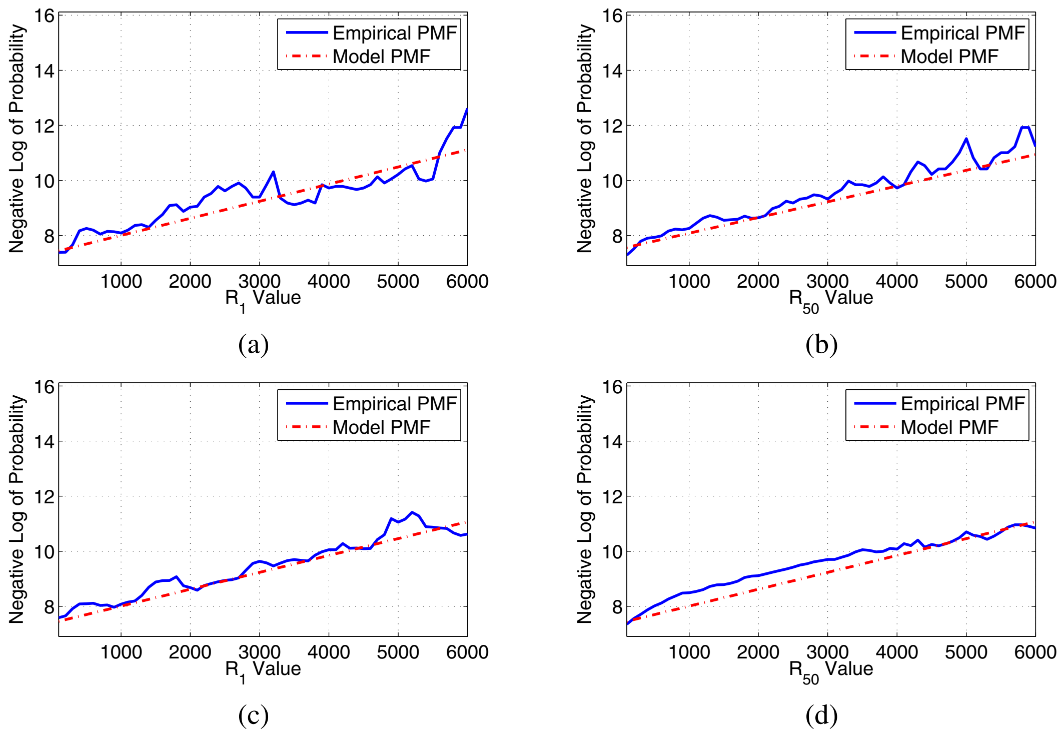


Figure 4. Probability distributions of R symbols for (a)-(b) ZuBuD and (c)-(d) CDD images.

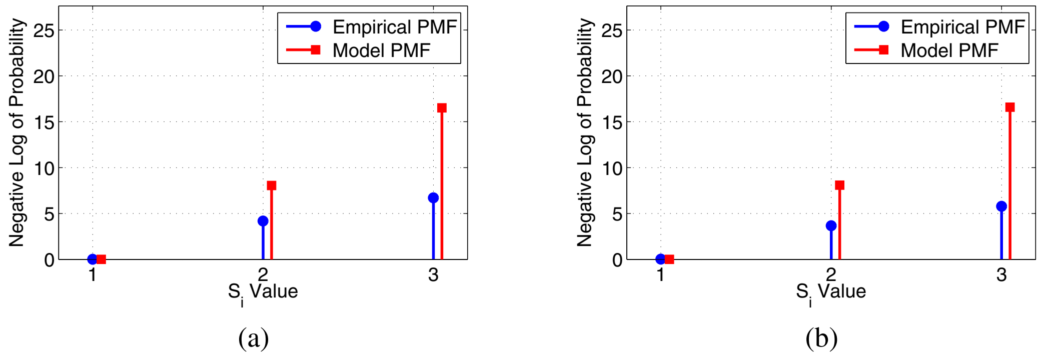


Figure 5. Probability distributions of S symbols for (a) ZuBuD and (b) CDD images.

3.3.2. Distribution for Positive Node Counts. The number of features s which visit a positive-count node nearly follows a binomial distribution. The only difference is that we must condition on the event $\{s > 0\}$:

$$p_{S_i}(s) \;=\; \frac{1}{1 - p_{\text{empty}}} \binom{N_{\text{desc}}}{s} (p_{\text{visit}})^s \, (1 - p_{\text{visit}})^{N_{\text{desc}} - s} \;, \qquad (9)$$
$$s \;\in\; \{1,\, 2,\, \ldots,\, N_{\text{desc}}\} \;.$$

Normalization by $(1 - p_{\text{empty}})$ accounts for the fact that each encoded node is nonempty. In Fig. 5, the model PMF in (9) is compared against the actual histogram for ZuBuD and CDD. Importantly, almost all of the node counts S_i have value one, which means each S_i symbol has very low entropy.

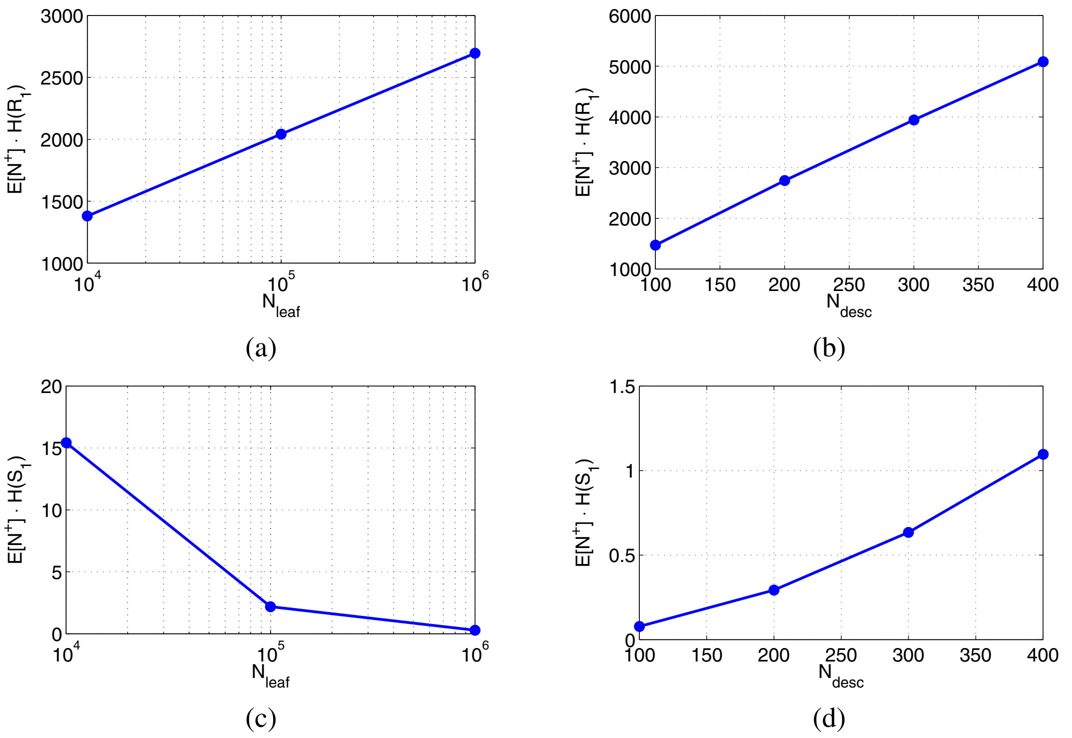


Figure 6. Joint entropy in bits for (a) R symbols versus N_{leaf}, for $N_{\text{desc}} = 220$, (b) R symbols versus N_{desc}, for $N_{\text{leaf}} = 10^6$, (c) S symbols versus N_{leaf}, for $N_{\text{desc}} = 220$, and (d) S symbols versus N_{desc}, for $N_{\text{leaf}} = 10^6$. Statistics reported for ZuBuD images.

3.3.3. Entropy of Distributions. We study how the parameters N_{leaf} and N_{desc} affect the distributions of node identities and counts, and consequently the symbol entropies. Results are shown for ZuBuD images, but results for CDD images are qualitatively very similar. First, N_{leaf} depends on the depth D and max branch factor K of the tree. A fully balanced tree has $N_{\text{leaf}} = K^{D-1}$. Deeper trees have more leaf nodes and cause flatter $p_{R_i}(r)$ distributions, and in turn, the entropy $H(R_i)$ increases. At the same time, deeper trees increase the number of positive-count nodes N^+, which follows a binomial distribution:

$$p_{N^+}(n) \;=\; \binom{N_{\text{leaf}}}{n} (1 - p_{\text{empty}})^n \, (p_{\text{empty}})^{N_{\text{leaf}} - n} \;, \qquad (10)$$
$$n \;\in\; \{0,\, 1,\, \ldots,\, N_{\text{leaf}}\} \;. \qquad (11)$$

The expectation is $E\left[N^{+}\right] = N_{\text{leaf}}\left(1 - p_{\text{empty}}\right)$. Fig. 6(a) plots $E\left[N^{+}\right] \cdot H\left(R_i\right)$, which is an estimate of the joint entropy of the R symbols, versus N_{leaf}. As the number of leaf nodes increases, the expected joint entropy increases.

As the number of query feature descriptors N_{desc} increases, the $p_{R_i}(r)$ distribution becomes peakier and has lower entropy. An opposing and stronger effect, however, is that $E[N^{+}]$ increases, because p_{empty} decreases. The overall effect on $E\left[N^{+}\right] \cdot H\left(R_i\right)$ is that it increases with N_{desc}, as can be seen from Fig. 6(b).

As N_{leaf} increases, the distribution $p_{S_i}(s)$ in (9) becomes peakier and contains lower entropy. More leaf nodes make it more probable that each nonempty node has a count of unity. The opposing and weaker effect is that as N_{leaf} increases, $E[N^{+}]$ increases. Overall, as N_{leaf} increases, $E\left[N^{+}\right] \cdot H\left(S_i\right)$ decreases, as shown in Fig. 6(c).

Lastly, as N_{desc} increases, $p_{S_i}(s)$ becomes flatter and contains higher entropy, and $E[N^{+}]$ also increases. The expected joint entropy $E\left[N^{+}\right] \cdot H\left(S_i\right)$ increases with N_{desc}. This behavior is plotted in Fig. 6(d).

An important observation is that the expected joint entropy of the S symbols is very low compared to the expected joint entropy of the R symbols. As we will experimentally confirm in Sec. 4, almost all of the bit budget is spent on encoding the node identities, with very few bits needed to encode the actual node counts.

4. Experimental Results

The performance of tree histogram coding is first compared against descriptor coding from [7], which applies a decorrelating transform, scalar quantization, and Huffman or arithmetic coding. To achieve the same matching accuracy, similar bit rates as descriptor encoding are needed for random projections in [8]. We also compare against tree-structured vector quantization (TSVQ), which transmits the index of the leaf node reached by each feature descriptor.

Our experiment tests two image databases. The first database is the Zurich Building Database (ZuBuD), consisting of 1,005 database images of 640×480 pixels resolution, representing five views of 201 different building facades in Zurich [15]. The ZuBuD query set consists of 115 images of 320×240 pixels resolution. The second database is our own CD Database (CDD), consisting of 10,597 database images of 500×500 pixels resolution, representing 10,597 different clean CD covers. Out of this database, 50 CD covers are randomly chosen, printed on paper, inserted into jewel cases, and photographed in different environments, resulting in 50 query images of 640×480 pixels resolution [16]. On average, each ZuBuD and CDD query image produces $N_{\text{desc}} = 220$ and $N_{\text{desc}} = 370$ SURF features, respectively.

4.1. Comparison of Different Codecs

For each database, an SVT of depth $D = 7$ and max branch factor $K = 10$ is trained using SURF descriptors extracted from the database images. After SURF features are extracted from a query image and classified through the SVT, the leaf histogram is transmitted using the encoding method of Sec. 3.2. Upon receipt of the leaf histogram and reconstruction of the entire tree's histogram, the server can compare the query image against database images using the dissimilarity score in (2).

For each query image, the identity of the database image attaining the minimum dissimilarity score is reported. The match rate, which is the fraction of correctly identified query images, is given in Table 2. For fair comparison, we force the match rates to be equal among the different schemes and then examine their different bit rates. For histogram coding, the bit rate is the number

Table 2. Image match rates and bit rates for descriptor coding, tree-structured vector quantization, and tree histogram coding.

Database	Codec	Match Rate	Bits/Image	Bits/Feature	Bits for $\{R_i\}$	Bits for $\{S_i\}$
ZuBuD	Descriptor	0.98	15,890	72.2	—	—
ZuBuD	TSVQ	0.98	4,390	19.9	—	—
ZuBuD	Histogram	0.98	2,890	13.1	2,850	40
CDD	Descriptor	0.96	26,620	71.9	—	—
CDD	TSVQ	0.96	7,380	19.9	—	—
CDD	Histogram	0.96	4,780	12.9	4,730	50

of bits in the arithmetic-coded bitstream for the R and S symbols. For descriptor coding, the bit rate is the number of bits needed to send entropy-coded, scalar-quantized transform coefficients. TSVQ transmits the leaf node reached by each feature. For a tree with N_{leaf} leaf nodes, using a fixed-length code, TSVQ requires a bit rate of $N_{\text{desc}} \cdot \log_2(N_{\text{leaf}})$ bits per image.

For tree histogram coding, good retrieval performance is achieved by sending just several thousand bits per image. Most of the bit budget is spent on encoding the R symbols, with very few bits needed to communicate the S symbols, which is in good agreement with our analysis in Sec. 3.3. The bit rate for the R symbols is close to our model prediction, while the bit rate for the S symbols is higher than predicted because our model overestimates the number of unity-count leaf nodes. To achieve the same matching accuracy, descriptor coding requires over $5\times$ the bit rate used for tree histogram coding.

As discussed in Sec. 3.2, tree histogram coding saves $R_{\text{order}} = \log_2(N_{\text{desc}}!)$ bits per image by discarding the order in a set of feature descriptors. In contrast, TSVQ requires about $1.5\times$ higher rate than tree histogram coding because TSVQ implicitly transmits the order. When we add the order rate R_{order} to tree histogram coding's rate, the result is approximately the rate required by TSVQ.

Table 3. Image match rates and bit rates for tree histogram coding and varying tree depth.

Database	Tree Depth	Match Rate	Bits/Image	Bits for $\{R_i\}$	Bits for $\{S_i\}$
ZuBuD	$D = 7$	0.98	2,890	2,850	40
ZuBuD	$D = 6$	0.94	2,210	2,150	60
ZuBuD	$D = 5$	0.91	1,990	1,880	110
CDD	$D = 7$	0.96	4,780	4,730	50
CDD	$D = 6$	0.92	3,540	3,450	90
CDD	$D = 5$	0.76	2,800	2,580	220

4.2. Variation in Tree Depth

For memory conservation, it is desirable to reduce the depth of the tree. A shallower tree with fewer leaf nodes also reduces the entropy of the tree histogram, as shown in Sec. 3.3. On the other hand, a deeper tree generally improves matching accuracy. We reduce the depth from $D = 7$ to $D = 5$ and observe the change in match rates and bit rates, shown in Table 3. As predicted by our analysis in Sec. 3.3, specifically Fig. 6, the bit rate for the R symbols decreases and the bit rate for the S symbols increases as the tree becomes shallower. Also, as expected, the matching accuracy becomes worse as tree depth decreases. Since CDD has a much larger database than ZuBuD, classification of CDD query images suffers more when the tree size decreases. If the mobile device has sufficient memory, it is preferable to use a deeper tree to attain better classification, especially if the target database is large.

5. Conclusion

In this paper, we demonstrated that when using a scalable vocabulary tree, it is much more rate-efficient to transmit the tree histogram than individual feature descriptors. By discarding order, histogram coding also requires lower rate than tree-structured vector quantization. The proposed system losslessly compresses the histogram by runlength coding of nonempty leaf nodes. Our statistical analysis accurately models the underlying random process generating the histogram, showing how tree depth and the number of query features affect the entropy of encoded symbols. We outperform a previous approach, which compresses feature descriptors by transform coding, by more than $5\times$ for the same image matching accuracy. This is of great interest for mobile image matching applications, where the response time due to transmission over a slow wireless link can be reduced dramatically. Future work should explore other beneficial applications of incorporating the classification tree at the encoder.

References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, November 2004.

[2] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, October 2005.

[3] H. Bay, T. Tuytelaars, and L. J. V. Gool, "SURF: Speeded up robust features," in *European Conference on Computer Vision*, 2006, pp. I: 404–417.

[4] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *IEEE Computer Vision and Pattern Recognition*, 2006, pp. II: 2161–2168.

[5] K. Grauman and T. J. Darrell, "Pyramid match hashing: Sub-linear time indexing over partial correspondences," in *IEEE Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[6] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Norwell, MA, USA: Kluwer, 1992.

[7] V. Chandrasekhar, G. Takacs, D. Chen, S. S. Tsai, J. Singh, and B. Girod, "Transform coding of image feature descriptors," in *SPIE Visual Communications and Image Processing*, 2009.

[8] C. Yeo, P. Ahammad, and K. Ramchandran, "Rate-efficient visual correspondences using random projections," in *IEEE International Conference on Image Processing*, 2008.

[9] G. Schindler, M. Brown, and R. Szeliski, "City-scale location recognition," in *IEEE Computer Vision and Pattern Recognition*, 2007, pp. 1–7.

[10] "GSM Arena." [Online]. Available: http://www.gsmarena.com/

[11] L. Yang, R. P. Dick, H. Lekatsas, and S. Chakradhar, "RAM for free," *IEEE Spectrum Magazine*, vol. 45, no. 8, pp. 38–43, August 2008.

[12] P. A. Chou, T. D. Lookabaugh, and R. M. Gray, "Optimal pruning with applications to tree-structured source coding and modeling," *IEEE Transactions on Information Theory*, vol. 35, no. 2, pp. 299–315, January 1989.

[13] T. Yeh, J. J. Lee, and T. J. Darrell, "Adaptive vocabulary forests for dynamic indexing and category learning," in *IEEE International Conference on Computer Vision*, 2007, pp. 1–8.

[14] M. Nelson and J.-L. Gailly, *The Data Compression Book*. New York, NY, USA: MIS:Press, 1996.

[15] H. Shao, T. Svoboda, and L. V. Gool, *ZuBuD - Zurich Buildings Database for Image Based Recognition*, April 2003. [Online]. Available: http://www.vision.ee.ethz.ch/showroom/zubud/

[16] D. Chen, S. Tsai, J. Singh, and B. Girod, *CDD - CD Cover Database - Query Images*, April 2008. [Online]. Available: http://msw3.stanford.edu/$\sim$dchen/CDD/index.html