Generalized Descriptor Compression for Storage and Matching

Matthew Johnson
http://www.matthewajohnson.org/

Nokia Point and Find
35th Floor
1 Market Plaza
San Francisco, CA

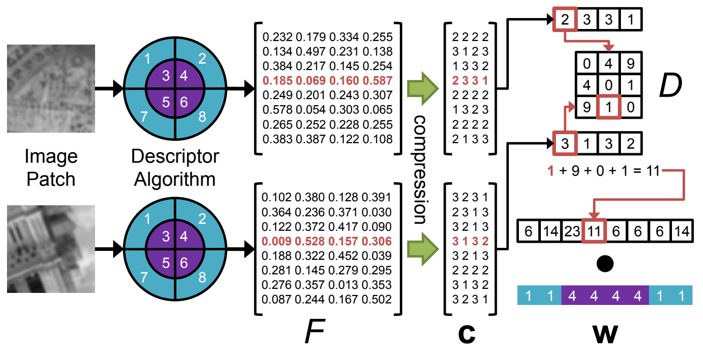

Figure 1: **Method Overview** In this figure, we can see an overview of the entire compression and matching method. A descriptor algorithm (in this case, a GLOH-like scheme) is used to compute a matrix F for two image patches, in which each row is a normalized histogram which corresponds to a cell in a mask over the patch. F is then compressed to form the vector $\mathbf{c}$, where each index represents a row from F. Each element in this vector is then used to compute per-histogram distances by way of a distance matrix D. The dot product of this vector and a weight vector $\mathbf{w}$ produce the final distance. F, D and $\mathbf{w}$ constitute our proposed *canonical form* of a descriptor algorithm.

Algorithm	*Mode*	*Error Rate*	*Bytes Used*
ZNCC	Reference	52.2	3600
SIFT	Reference	7.29	512
NSIFT	Canonical	**3.27**	512
CSIFT	Compressed	4.29	48
MU-SURF	Reference	30.5	256
NSURF	Canonical	23.0	256
CSURF	Compressed	**22.8**	20
GLOH	Reference	4.09	1088
NGLOH	Canonical	**2.47**	1088
CGLOH	Compressed	2.91	136

Figure 2: **Descriptor Retrieval Results** The error rate is computed at 95% detection. The last column denotes the number of bytes used to encode the descriptor in our experiments. The algorithm modes refer to either a reference implementation of the algorithm, our canonical form of the algorithm, or the compressed version of our canonical form.

With the advent of smart phones that incorporate high quality cameras, mobile computer vision has become an area of increased research. While many efforts have focused on thin phone clients which interact with a remote server that uses conventional computer vision techniques to solve image matching and recognition problems, the current generation of phones are sufficiently powerful that they can execute computer vision algorithms locally. In particular, ubiquitous augmented reality on the mobile phone becomes a feasible undertaking, as the complementary sensors (*e.g.* GPS, accelerometer) work along with the camera to provide a rich user experience. Such applications heavily utilize interest point and descriptor based algorithms for matching, registration and tracking. However, these algorithms present unique challenges in a mobile computer vision scenario:

Storage Space Memory is in short supply on mobile handsets, and thus the large memory footprints of descriptors like SIFT, SURF and GLOH make the storing of databases in memory unfeasible, and writing to and from the disk on a mobile phone is too slow for the realtime recognition required for an augmented reality application.

Bandwidth The local image descriptors computed for a reference image or object number in the thousands and each require a large number of bits to represent. Given the bandwidth available in a mobile scenario, providing a localized database from a remote server as a user moves from one geographical area to another becomes untenable.

Computation The most computationally intensive component of any image matching or registration system is descriptor database retrieval. With thousands of descriptors in the image and thousands more in the database, millions of distance computations can potentially take place.

In order to allow mobile computer vision to exploit the substantial body of computer vision research depending upon existing local image descriptor algorithms despite the challenges listed above, we propose a general and efficient method of compression for descriptors, and a technique by which those descriptors can be compared without requiring decompression. An overview of our method can be seen in Figure 1.

We propose a *canonical descriptor form* uniquely suited for compression which captures the structure shared by most local image descriptors. This form consists of three parts, as seen in Figure 1:

1. An $M \times N$ matrix, F, in which each row is normalized such that $\sum_j^N |F[i,j]| = 1 \quad \forall i \in M$.
2. D is a $N \times N$ distance lookup matrix in which $D[i,j] = f(2^{-i}, 2^{-j})$, where f is a decomposed portion of a distance metric.
3. A weight vector, $\mathbf{w}$, of length M determining how much a particular row should contribute to the overall distance.

First, the F matrix is computed for an image patch. Each row of this matrix is compressed using tree coding, resulting in a vector $\mathbf{c}$. Each index of $\mathbf{c}$ encodes an entire row from F using $O(N \log N)$ bits, where N is the length of the original histogram. $\mathbf{c}$ vectors are used to index the distance matrix D to compute a distance between descriptors without requiring decompression.

The normalized rows of F can be compressed using methods developed for the lossy compression of probability distributions, such as those proposed by Gagie in [1]. These algorithms use binary trees to assign a depth code to each element of the distribution which corresponds to a negative power of 2. The algorithm we use is based on Huffman Trees and guarantees that $D(P||Q) < 1$, where P is the original distribution, Q is the compressed distribution and $D(P||Q)$ is the Kullback-Leibler divergence. The results achieved by this compression can be seen in Figure 2.

The distance lookup matrix D contains the pre-computed values of a decomposed distance metric. For example, in the case of the L2 norm $f(i,j) = (2^{-i} - 2^{-j})^2$. For two compressed vectors $\mathbf{c_x}$ and $\mathbf{c_y}$, the distance for rows $i \in M$ can be computed as $\sum_j D\left[\mathbf{c_x}[i,j], \mathbf{c_y}[i,j]\right]$ for columns $j \in N$. One result of normalizing each row of F separately and computing the distance between each row directly is that any weighting of cells in the descriptor is lost. Our canonical form recaptures this effect by way of the weighting vector $\mathbf{w}$. The values of $\mathbf{w}$ weight the row distances and as such recapture the effect of the cell-weighting used in many descriptors.

The conversion to canonical form, which computes the distance between rows in F separately using D before weighting the result with $\mathbf{w}$ results in an improvement in performance for all descriptor algorithms, as can be seen in Figure 2. While compression of this canonical form results in a slight loss in accuracy, the compressed version still performs better than the reference implementation while requiring far less storage space and having a significantly reduced computational cost at runtime due to the use of a lookup matrix in distance computations.

[1] T Gagie. Compressing probability distributions. *Information Processing Letters*, 97(4):133–137, February 2006.