Toward compressed 3D descriptors

Filippo Malaguti, Federico Tombari, Samuele, Salti
DEIS, University of Bologna
Bologna, Italy
filippo.malaguti@studio.unibo.it
{*federico.tombari, samuele.salti*}*@unibo.it*

Danilo Pau
STMicroelectronics
Agrate Brianza, Italy
danilo.pau@st.com

Luigi Di Stefano
DEIS, University of Bologna
Bologna, Italy
{*luigi.distefano*}*@unibo.it*

***Abstract*—Visual search for mobile devices relies on transmitting wirelessly a compact representation of the query image, generally in the form of feature descriptors, to a remote server. Descriptors are therefore compressed, so as to reduce bandwidth occupancy and network latency. Given the impressive pace of growth of 3D video technology, we foresee 3D visual search applications for the mobile and the robotic market to become a reality. Accordingly, our work proposes a study on compressed 3D descriptors, a fundamental building block for such prospective applications. Based on analysis of several compression approaches, we develop and assess different schemes to achieve a compact version of a state-of-the-art 3D descriptor. Through experiments on a vast dataset we demonstrate the ability to achieve compression rates as high as 98% with a negligible loss in 3D visual search performance.**

***Keywords*-3D visual search; 3D descriptor; data compression; surface matching**

I. Introduction

The widespread diffusion of mobile devices equipped with high resolution cameras is increasingly pushing computer vision applications within mobile scenarios. The common paradigm is represented by a user taking a picture of the surroundings with a mobile device to obtain informative feedback on its content. This is the case, e.g., in mobile shopping, where user can shop just by taking pictures of products[1], or landmark recognition for ease of visiting places of interest [1], [10]. As in the aforementioned scenarios visual search needs to be typically performed over a large image database, applications communicate wirelessly with a remote server to send visual information and receive the informative feedback. As a result, a constraint is set forth by the bandwidth of the communication channel, whose use ought to be carefully optimized to bound communication costs and network latency. For this reason, a compact but informative image representation is sent remotely, typically in the form of a set of local feature descriptors (e.g. SIFT [18], SURF [2], etc.) extracted from the image.

Despite the summarization of image content into local features, the size of state–of–the–art descriptors cannot meet bandwidth requirements. Hence, a research trend addressing effective compression of feature descriptors has emerged recently, so as to save communication bandwidth while minimizing the loss in descriptive power. Several techniques aimed at descriptor compression, also known as *compressed* or *compact descriptors*, have been proposed in the literature [4]–[6], [15], The perceived market potential of mobile visual search has also lead to establishment of an MPEG committee which is currently working on the definition of a new standard focused on Compact Descriptors for Visual Search (CDVS)[2].

Techniques for feature detection and description from 3D data have also been proposed in literature [9], [14], [19], [24], [25], [31], [32], the topic recently fostered significantly by the advent of accurate and low-cost 3D sensors, such as the Microsoft Kinect and the Asus Xtion. Popular applications of 3D features include shape retrieval within 3D databases (e.g. Google *3D Warehouse*), 3D reconstruction from range views, recognition and categorization of 3D objects. On the other hand, driven by the developments of 3D video technologies (3D movies, 3D TVs, 3D displays), embedded low-cost 3D sensors have started appearing on a number of diverse mobile devices. For instance, this is the case for new smartphones and tablets, such as the LG Optimus 3D P920, LG Optimus Pad, HTC EVO 3D and Sharp Aquos SH-12C, as well as game consoles, like the 3DS by Nintendo. A study by In-Stat [12] claims that the market for 3D mobile devices is on a steady and fast growth, and that by 2015 it will count more than 148 millions units. Accordingly, new researches are investigating the development of 3D data acquisition technologies specifically conceived for mobile devices [16]. Interestingly, novel technologies for 3D data acquisition are recently being developed for smartphones not equipped with 3D sensors: this is the case of the *Trimensional* Iphone app[3].

Given the predicted fast development of the 3D ecosystem, we envision the demand for new applications that will require querying a 3D database by means of 3D data gathered on-the-fly by mobile device or robots. They will likely adhere to the paradigm of current 2D visual

[1]see e.g. http://www.pixlinq.com/home

[2]http://mpeg.chiariglione.org/working_documents.php

[3]www.trimensional.com

978-0-7695-4873-9/12 $26.00 © 2012 IEEE
DOI 10.1109/3DIMPVT.2012.9

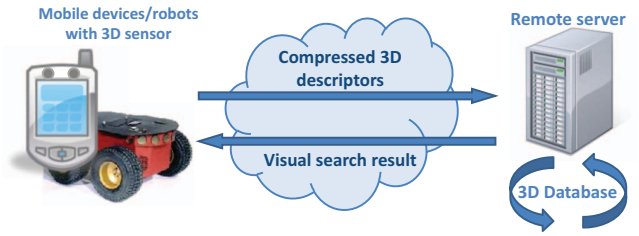


Figure 1: 3D Mobile Visual Search by transmission of compressed 3D descriptors.

search applications, as sketched in Fig. 1. Among already proposed scenarios, we point out here, e.g., the possibility of performing 3D landmark recognition based on a 3D database of buildings [1] or service robots equipped with 3D sensors relying on a cloud computing infrastructure to infer knowledge about objects in their surroundings [11].

Key to the foreseen 3D search scenarios is therefore a novel research topic dealing with compact 3D descriptors, which ought to be developed to effectively support transmission of the relevant local information tokens related to the query 3D scene. In this paper, we pave the way for this new research line by investigating compression techniques for attaining compact representations of 3D local descriptors, thus providing reference methods together with the associated figures of merit. More specifically, we consider a state-of-the-art 3D descriptor, i.e. SHOT [25], in order to devise and evaluate several approaches relying on recent data compression techniques. We carry out experiments on a vast 3D dataset, which allows us to identify the most favorable trade-off between the conflicting requirements of high compression rate and limited performance loss with respect to the original uncompressed descriptor. Results turns out quite satisfactory: an average compression rate of around 98% with a negligible loss in performance can be achieved.

II. Related work

A. Compact Descriptors

As far as 2D compact descriptors are concerned, most techniques proposed so far deal with SIFT [18]. An interesting and thorough review of SIFT compression approaches can be found in [5], where three different categories of compression schemes are identified: hashing, transform coding and vector quantization. In the first, each descriptor is associated with a hash code. These codes are then compared based on their Euclidean or Hamming distance. Examples of such methods are *Locality Sensitive Hashing* [30], *Similarity Sensitive Coding* [27] and *Spectral Hashing* [28]. Instead, transform coding is a technique used for audio and video compression (e.g. JPEG). A conventional transform coder takes an input vector X and transforms it into another vector $Y = TX$ of the same size, then quantizes this new vector. The transformation allows for decorrelating the different dimensions of the original vector, in order to make quantization more effective and reduce the loss in performance. The decoder takes the transformed and quantized vector and applies an inverse transformation to obtain an estimation of the original vector [20]. Example of transform coding schemes are *Karhunen-Love Transform* [7] and *ICA based Transform* [20].

Finally, compression based on vector quantization subdivides the descriptor space into a fixed number of bins (codebook) using clustering techniques such as the k-means algorithm. Successively, instead of a descriptor, its associated codeword ID can be sent. Two examples are *Product Quantization* [13] and *Tree Structured Vector Quantization* [21]. Although generally able to yield small distortions of the original signal, the main disadvantage of such approaches is that the codebook must be present at both the encoder and the decoder side. In our scenario, this requires the codebook to be stored on the mobile device and transmitted, which could be cumbersome due to its size being often considerable. Moreover, if the codebook is modified at run-time, it requires an additional transmission overhead to keep the synchronization between encoder and decoder. Another possibility deals with the use of a data-independent codebook, such as in Type Coding [6] . In this case, the codebook is based on a regular grid defined over the descriptor space, which usually implies more distortion but does not require local storage of the codebook nor any synchronization overhead.

Alternatively to SIFT, one of the most famous compact descriptor is CHoG [6], which reported the best trade-off between compression rate and visual search performance when compared to other compact descriptors [6]. To build the CHoG descriptor, first an uncompressed descriptor (UHoG) is extracted, which, like SIFT, is a vector of histograms of gradient orientations, but carries out spatial binning according to a DAISY configuration [29] instead of a 4x4 square grid. Successively, UHoG is compressed by means of Type Coding [6] to end up with the CHoG descriptor.

An example of compact 3D descriptor was proposed by Papadakis et al. [22]. However, this descriptor is not a local 3D descriptor, but instead a hybrid 2D/3D global descriptor. To compress the descriptor the authors apply scalar quantization followed by Huffman Coding, reaching a compression rate of 92.6%.

B. The SHOT descriptor

The SHOT descriptor [25] encodes a signature of histograms of topological traits. A 3D spherical grid of radius r, made out of 32 sectors, is centered at the keypoint to be described and oriented according to a unique local reference frame which is invariant with respect to rotations and translations. For each spherical grid sector, a one-dimensional histogram is computed, built up by accumulating the cosine

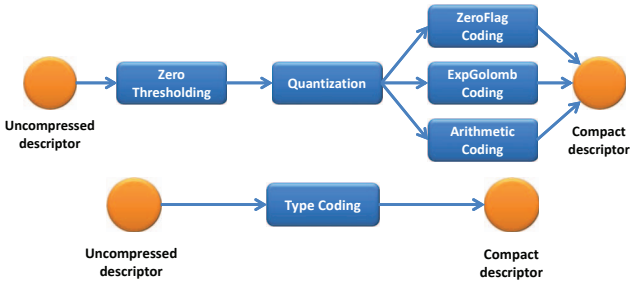


Figure 2: Four different approaches to achieve a compact 3D descriptor.

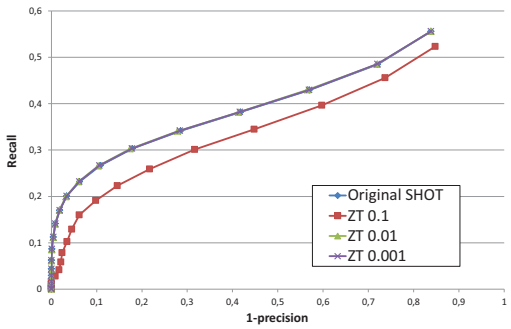


Figure 3: Effects of ZT on SHOT performance (Spacetime dataset).

- discretized into b_S bins - of the angle between the normal at the keypoint and the normal of each of the points belonging to the spherical grid sector for which the histogram is being computed. The final descriptor is then formed by orderly juxtaposing all histograms together according to the local reference frame. To better deal with quantization effects, quadrilinear interpolation is applied to each accumulated element. Finally, to improve robustness with respect to point density variations, the descriptor is normalized to unit length. When color information is available together with depth, as is the case of RGB-D data provided by the Kinect, an additional set of histograms can be computed [26], where the L_1 norm between the color triplet of the center point and that of each point of the current spherical grid sector is accumulated in each histogram, quantized into b_C bins (usually $b_C \neq b_S$). The SHOT code is publicly available as a stand-alone library[4], as well as part of the open source Point Cloud Library[5].

III. Compression schemes for 3D descriptors

As previously mentioned, we consider a recent state-of-the-art proposal [25], [26] to investigate compression schemes suitable for achieving compact 3D descriptors. Purposely, we have analyzed several state-of-the-art algorithms for data compression and derived four approaches, which are outlined in Figure 2. The first three approaches share Zero Thresholding (ZT) followed by quantization: these two steps aim at reducing the redundancy of each descriptor element, and they are both lossy, i.e. the introduced compression can not be reversed. Successively, a lossless compression algorithm is applied, represented either by ZeroFlag, ExpGolomb or Arithmetic Coding, so as to reduce redundancy in the sequence of elements stored in each descriptor. In addition, we investigate a fourth approach based on Type Coding, which is the compression algorithm deployed by CHoG [6].

A. Zero Thresholding

The intuition that, generally, 3D surfaces intersect only a limited portion of a volumetric neighborhood around a keypoint, suggests that a number of proposed 3D descriptors [9], [24], [31], [32] are often quite sparse, i.e. with many values equal (or close) to zero. This is, indeed, the case for SHOT, for which we have experimentally verified this intuition finding that typically more than 50% of the elements are null. This characteristic may be exploited by a lossless compression step, i.e. by using just a few bits to encode the zero value. Moreover, it turns out to be even more effective to threshold to zero also those elements having a small value: we refer to this step as Zero Thresholding (ZT).

	Null Elements	
Th.	*Kinect*	*Spacetime*
0	57.67%	72.16%
0.00001	57.95%	72.44%
0.0001	59.38%	72.73%
0.001	64.49%	75.57%
0.01	76.70%	83.52%
0.1	93.47%	94.89%

Table I: Null elements after Zero Thresholding using different thresholds.

Table I shows the percentage of elements that are less than or equal to a threshold within a set of SHOT descriptors extracted from the two datasets that will be presented in Section IV, namely *Kinect* and *Spacetime*. As demonstrated by the Table, a threshold equal to 0.01 yields a percentage of null elements as high as 83%, while thresholding at 0.1 allows reaching 94%. However, we observed (see e.g. Figure 3) that while the first threshold value does not noticeably affect the performance of the descriptor, the second one causes a significant performance deterioration: thus, we have set the ZT threshold to 0.01.

B. Quantization

The original SHOT descriptor represents each element as a double precision floating-point number[6]. Given the SHOT normalization step, which results in all its elements ranging between 0 and 1, it is possible to quantize each value with a fixed number of bits, thus reducing the descriptor size. Since

[4]http://vision.deis.unibo.it/SHOT
[5]www.pointclouds.org
[6]We assume hereinafter double precision to require 64 bits.

we found that the descriptor performance starts to deteriorate when using less than 4 bits for the quantization step, the experiments in Sec. IV are carried out using 6 and 4 bits. It is worth noting that, depending on the coarseness of the quantization, this step can also account for the previous ZT step (e.g. this occurs in the case of 4 bits, where all values smaller than 1/32 are quantized to 0).

C. ZeroFlag Coding

We propose ZeroFlag Coding (ZFC) as a way to exploit the usually high number of null values present in the descriptor (especially after Zero Thresholding). It effectively encodes sequences of "zeros" by means of an additional flag bit, F, which is inserted before every element different from zero or every sequence of zeros. The flag bit F is inserted according to the following rules:

- $F = 1$ means that the next element is not zero, and it is followed by a fixed number of bits representing the quantized value of this element.
- $F = 0$ means that the next element is a sequence of zeros, and it is followed by a fixed number of bits indicating the length of the sequence.

This approach requires specifying the maximum length of a zero sequence. Since in our experiments a good performance was obtained with a value of the maximum length equal to 16, we have used 5 bits to encode each zero sequence (1 flag bit plus 4 bits to encode the length of the sequence). A zero sequence longer than 16 elements is split into multiple sequences.

D. ExpGolomb Coding

ExpGolomb Coding (EGC), described in detail in [17], is a compression algorithm allowing the use of few bits to represent small values, the number of required bits increasing with increasing numerical values. The algorithm is controlled by a parameter k, which in our investigation is set to 0, so that each null element, particularly frequent in SHOT especially after Zero Thresholding, is represented by just one bit.

E. Arithmetic Coding

The idea behind Arithmetic Coding (AC) is to represent highly frequent values with few bits, the number of bits increasing as the symbol becomes less frequent (or less probable). Frequencies can be estimated through a training stage, wherein the probability distribution associated with symbols is learned. Alternatively, they can also be learned without a specific training stage in an *adaptive* manner, whereby at the beginning all symbols have the same probability, then each frequency is updated every time a symbol is encoded (or decoded). In this last case, there is no overhead due to initial codebook synchronization between encoder and decoder. In our study, we have considered the adaptive version of the AC algorithm, since it is more generally applicable, due to a training stage not being feasible in several application scenarios related to 3D visual search descriptors. A detailed explanation of the AC algorithm can be found in [3], which also provides the implementation of the adaptive version of the algorithm that we used in our experiments[7].

F. Type Coding

Given an m-dimensional symbol, s, Type Coding (TC) associates its nearest neighbor q over a regular m-dimensional lattice. Hence, the index associated with q is transmitted instead of s. The lattice can be built as in [6], the main benefit being that its structure is independent of the data, so that TC does not require storage and transmission of any codebook.

Beside m, Type Coding relies on another parameter, n, which is used to control the number of elements constituting the lattice [6], so that the total number of elements in the lattice coincides with the number of partitions of parameter n into m terms according to the following multiset coefficient:

$$\left(\!\binom{m}{n}\!\right) = \binom{m+n-1}{m-1} \tag{1}$$

The number of bits needed to encode each index is at most [6]:

$$\left\lceil \log_2 \left(\!\binom{m}{n}\!\right) \right\rceil \approx (m-1)\log_2 n \tag{2}$$

To experiment with TC, we followed the approach of subdividing SHOT into equally sized sub-vectors and then applying TC compression to each sub-vector. As TC requires the elements to be encoded to sum up to 1, we appended the set of required normalization factors associated to each sub-vector at the end of the compressed descriptor. Finally, the array formed by the normalization factors is also L_1 normalized between 0 and 1, and then quantized with 8 bits to reduce its storage (this last normalization factor needs not to be stored). This allows the normalization step to be reversed at the end of the decoding stage with a limited loss due to normalization factor compression, as otherwise the information content of the descriptor would be distorted by the different normalization factors.

As SHOT consists of 32 histograms, we evaluated the performance of TC by combining them into sub-vectors consisting of k histograms, with k equal to 1, 2, 4, 8 or 16. Considering, for instance, parameter b_S equal to 10, as proposed in [25], parameter m in (1) can be set to $k\cdot(b_S+1), k = 1, \cdots, 16$[8]. From equation 2 it is possible to determine the size of the compact descriptor, and thus the overall compression rate, for different parameters choices. In Table II we report the data obtained by choosing, for each value of m, the value of n that minimizes the accuracy loss with respect to the uncompressed SHOT descriptor. It

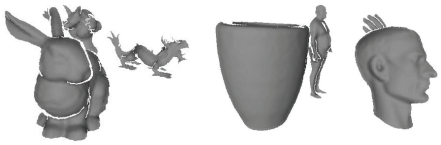
[7]Available at www.bodden.de/legacy/arithmetic-coding

[8]This is due to the number of bins being actually $b_S + 1$ in the authors' implementation used in our evaluation.

m	n	Bit size	Compression rate
11	40	1344	94.03%
22	20	736	96.73%
44	60	848	96.24%
88	100	764	96.61%
176	100	528	97.66%

Table II: TC: Bit size and Compression rate of TC with different parameter choices.

(a) Spacetime (b) Kinect (c) Stanford



(d) Virtual Stanford (e) Virtual Watertight

Figure 4: Examples of scenes belonging to the evaluated datasets.

can be noticed that the choice $m = 176$, $n = 100$, i.e. $k = 16$, provides the highest compression rate. Therefore, in our experiments we used these values, so as to favor compactness of the descriptor. However, it is worth pointing out that: i)the computational complexity (thus, encoding and decoding time) of TC grows with m and n, and ii)the algorithm uses, internally, integers represented with a large number of bits, which may be difficult to handle, both in software as well as in hardware. With the choice $m = 176$, $n = 100$, the resulting descriptor consists of two 256 bits integers: to handle them, we had to use a specific software library for large sized-integers[9], which causes a significant increase of the computational burden. As for experiments including color information, m and n have been set according to the same principle, in particular $m = 16 \cdot (b_S + 1)$ for the shape part and $m = 16 \cdot (b_C + 1)$ for the color part, and $n = 100$.

IV. Experimental results

The described approaches for achieving compact 3D descriptors are evaluated and compared here in terms of performance and compression rate with respect to the uncompressed SHOT. We address the case of 3D data as well as that of RGB-D data.

[9]Available at https://mattmccutchen.net/bigint/

Data	Dataset	b_S	b_C	r	bits
Shape-only	Kinect	10	-	30	22528
	Spacetime	10	-	15	22528
	Stanford	10	-	15	22528
	Virtual Stanford	10	-	60	22528
	Virtual Watertight	10	-	60	22528
Shape + Color	Kinect	15	5	30	45056
	Spacetime	10	30	15	86016

Table III: SHOT Parameters: from left to right, number of shape and color histogram bins, radius of the support (in mean mesh resolution units), total number of bits of the uncompressed descriptor.

A. Datasets

Experiments are carried out over five different datasets, two of which also contains color information and will be used in the experiments concerning RGB-D descriptors. Three of these datasets are those that were originally used in the experimental evaluation of SHOT [25], [26]:

- *Spacetime* dataset[10], containing 6 models and 15 scenes acquired with the Spacetime Stereo technique.
- *Kinect* dataset[10], containing 6 models and 17 scenes acquired with a Microsoft Kinect device.
- *Stanford*[10], containing 6 models and 45 scenes built assembling 3D data obtained from the Stanford repository[11].

Two additional datasets, namely *Virtual Stanford* and *Virtual Watertight*, were built using, respectively, 6 models from the Stanford repository and 13 models from the Watertight dataset[12]. The scenes in these datasets have been created by randomly placing 3 to 5 models close to each other, then rendering 2.5D views in the form of range maps, with the aim of mimicking a 3D sensor such as the Kinect. To this end, we implemented a Kinect simulator which first generates depth-maps from a specific vantage points by ray casting, then adds Gaussian noise [8], [23] and quantizes the z-coordinates [23], with both the noise variance and the quantization step increasing with distance. Finally, we applied bilateral filtering to the depth maps to reduce noise and quantization artifacts.

All these datasets include, for each scene, ground-truth information (i.e. the list of model instances present in the scene, together with their rotation and translation with respect to the original model). Figure 4 shows sample scenes from each dataset.

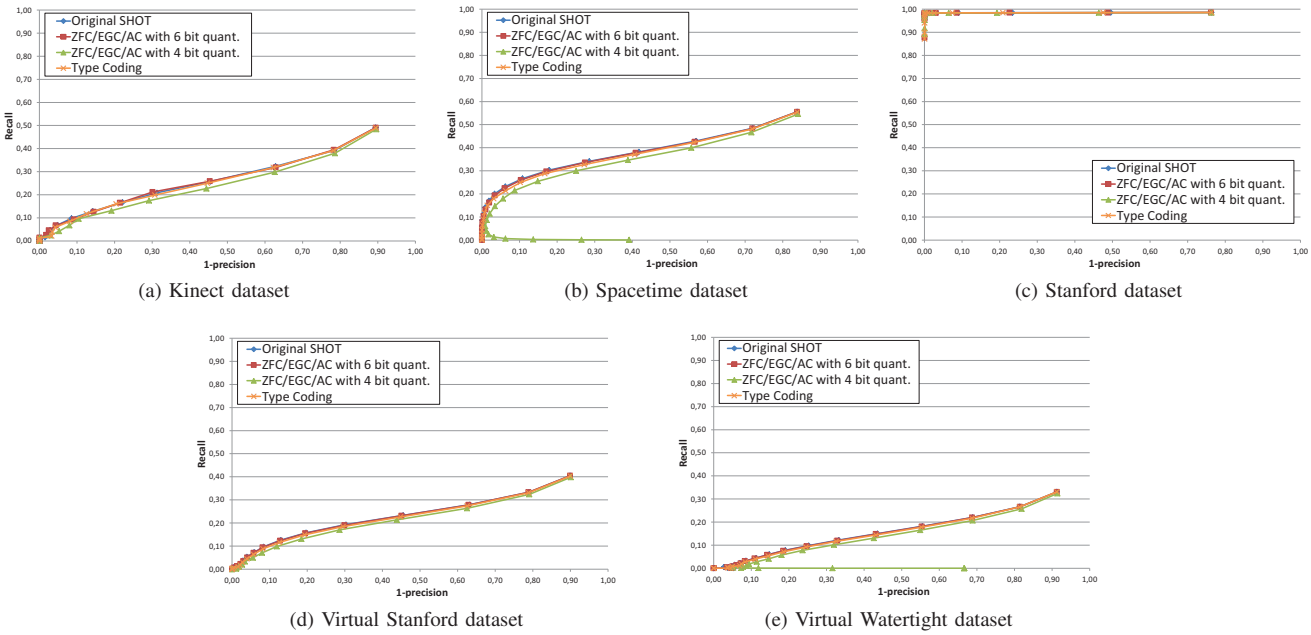
B. Methodology

To evaluate the performance of compact descriptors, we follow the methodology proposed in [25], [26]. Specifically, we first extract a predefined number of keypoints from

[10]http://www.vision.deis.unibo.it/SHOT/

[11]http://graphics.stanford.edu/data/3Dscanrep/

[12]http://watertight.ge.imati.cnr.it/



(a) Kinect dataset (b) Spacetime dataset (c) Stanford dataset

(d) Virtual Stanford dataset (e) Virtual Watertight dataset

Figure 5: Performance yielded by descriptors on 3D data.

	ZFC		EGC		AC		TC
	6-bits	**4-bits**	**6-bits**	**4-bits**	**6-bits**	**4-bits**	
Kinect	97.18	97.79	97.27	97.97	97.78	98.80	97.66
Spacetime	97.52	98.02	97.43	98.03	98.05	98.92	97.66
Stanford	95.77	96.77	96.67	97.77	96.94	98.47	97.66
Virtual Stanford	97.12	97.75	97.25	97.96	97.73	98.79	97.66
Virtual Watertight	97.56	98.06	97.46	98.03	98.07	98.95	97.66
Avg.	97.03	97.68	97.22	97.95	97.71	98.79	97.66

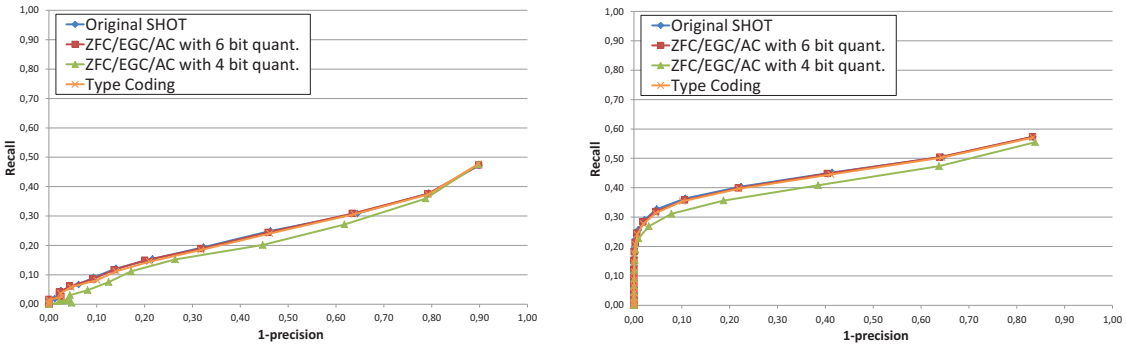
Table IV: Compression rates (%) yielded by the descriptors on 3D data.

each model via random sampling, then rely on ground-truth information to select the scene points that exactly match those extracted from models. To simulate the presence of outliers, we randomly extract a predefined number of keypoints from clutter, which do not have a correspondent among the models. For each keypoint, the SHOT descriptor is computed. For the SHOT parameters, we have tuned the size of the radius r and the number of shape and color bins (b_S and b_C) so as to adapt them the specific characteristics of the dataset. The tuned values, listed in Table III, are used by all the considered compact descriptors.

After computation of the descriptors, each vector is first encoded and then decoded. This is done also for the models descriptors, so as to account for the distortions brought in by compression. Successively, the matching stage compares the descriptors extracted from each model to those identified in each scenes based on the Euclidean distance in the descriptor space. More precisely, according to the methodology adopted in [25], we match descriptors based on the ratio of distances criterion [18]. Correspondences are then compared with the ground-truth to compute the number of True Positives and False Positives at different values of the matching threshold, thus attaining *Precision-Recall* curves. It is important to point out that, as shown in Figure 2, the three compact descriptors based on ZeroFlag, ExpGolomb and Arithmetic Coding have identical performance (i.e. identical *Precision-Recall* curves) due to their lossy stages being exactly the same. Accordingly, we will always plot a single Precision-Recall curve for the three methods, and compare them in terms of their different compression rates.

C. Results

Fig. 5 shows the *Precision-Recall* curves for the evaluation of compact SHOT descriptors on the five datasets using 3D shape information only. In particular, each chart reports the performance of the uncompressed SHOT descriptor together with those provided by ZFC/EGC/AC (using 6 and 4 bits for quantization) as well as by Type Coding. Table IV reports the achieved compression rates. These results show that ZFC, EGC and AC pipelines using 6 bit quantization, as well as Type Coding, are notably effective, achieving high compression rates (between 96 and 98%) with a negligible loss in performance compared to the uncompressed SHOT descriptor. Among the compared approaches, AC and Type Coding yield the best compression rates, with AC based on 6 bit quantization performing slightly better than Type Coding (average compression rate 97.71% vs. 97.66%). Moreover, as discussed previously, Type Coding with parameters tuned



(a) Kinect dataset

(b) Spacetime dataset

Figure 6: Performance yielded by descriptors on RGB-D data.

	ZFC		EGC		AC		TC
	6-bits	**4-bits**	**6-bits**	**4-bits**	**6-bits**	**4-bits**	
Kinect	97.63	98.12	97.61	98.13	98.24	99.13	97.75
Spacetime	98.51	98.77	97.98	98.27	98.92	99.45	98.47
Avg.	98.07	98.45	97.80	98.20	98.58	99.29	98.11

Table V: Compression rates ($\%$) yielded by descriptors on RGB-D data.

to achieve a performance level comparable to AC turns out to be significantly less efficient. In particular, with our implementation, encoding with Type Coding is on the average between 3 and 4 times slower than encoding with ZT, quantization and AC (i.e. $0.26ms$ vs. $0.07ms$ per descriptor), while decoding can be up to two orders of magnitude slower ((i.e. $0.58ms$ vs. $0.05ms$ per descriptor). Therefore, the pipeline based on AC seems the preferred choice to attain a compact SHOT descriptor for 3D shape data.

These findings are confirmed by the results of the experiments on RGB-D data (i.e. using both 3D shape and color), reported in Fig. 6. and Table V. Again, 6-bits ZFC, EGC and AC as well as Type Coding exhibit a performance level indistinguishable from the uncompressed SHOT while providing excellent compression rates. Also with RGB-D data, 6-bits AC seems the best compact descriptor, due to its higher average compression rate (98.58% vs. 98.11% of Type Coding) and lower computational complexity with respect to Type Coding.

We point out that the only other compact 3D descriptor we are aware of, i.e. the hybrid 2D/3D global descriptor proposed in [22], achieves an inferior compression rate (92.6%) than our proposals.

Finally, we ran experiments using a state-of-the-art 3D keypoint detector [32] instead of random sampling. The results confirmed the trend related to random keypoint selection, as regards both compression rates as negligible accuracy loss with respect to the uncompressed descriptor (the reader is referred to the supplementary material for the details concerning the experiment using [32]).

V. Conclusions

This paper has demonstrated how the use of suitable compression techniques can greatly reduce the redundancy of a state-of-the-art 3D descriptor, providing dramatic shrinking of the descriptor size with a negligible loss in performance. Among considered ones, the approach based on Arithmetic Coding is preferable to Type Coding, the latter being the compression method deployed by the most popular image descriptor (i.e. CHOG). A key intuition behind the devised compression pipelines deals with leveraging on the sparsity of the considered 3D descriptor, a feature that is likely to be advantageous also with several other 3D descriptors relying on a volumetric support. We hope that this study may contribute to foster a novel research direction as well as the related application scenarios, so that compact 3D descriptors may be used for searching and knowledge discovering in large remote databases given query 3D data sensed by next-generation mobile devices and robots.

References

[1] G. Baatz, K. Köser, D. M. Chen, R. Grzeszczuk, and M. Pollefeys. Leveraging 3d city models for rotation invariant place-of-interest recognition. *International Journal of Computer Vision (IJCV)*, 96(3):315–334, 2012.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 110(3)(3):346–359, 2008.

[3] E. Bodden, M. Clasen, and J. Kneis. Arithmetic coding revealed - a guided tour from theory to praxis. Technical report, Sable Research Group, School of Computer Science, McGill University, 2007.

[4] M. Calonder, V. Lepetit, and P. Fua. Brief: binary robust independent elementary features. In *European Conference on Computer Vision (ECCV)*, 2010.

[5] V. Chandrasekhar, M. Makar, G. Takacs, D. Chen, S. S. Tsai, N. M. Cheung, R. Grzeszczuk, Y. Reznik, and B. Girod. Survey of sift compression schemes. In *International Conference on Pattern Recognition (ICPR)*, 2010b.

[6] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, Y. Reznik, R. Grzeszczuk, and B. Girod. Compressed histogram of gradients: a low bitrate descriptor. *International Journal on Computer Vision (IJCV)*, 94(5)(5), May 2011.

[7] V. Chandrasekhar, G. Takacs, D. M. Chen, S. S. Tsai, and B. Girod. Transform coding of feature descriptors. In *Visual Communications and Image Processing Conference (VCIP)*, 2009.

[8] D. Falie and V. Buzuloiu. Noise characteristics of 3d time-of-flight cameras. In *International Symposium on Signals, Circuits and Systems (ISSCS)*, 2007.

[9] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *European Conference on Computer Vision (ECCV)*, 2004.

[10] Google. Google goggles. *http://www.google.com/mobile/goggles*, 2011.

[11] E. Guizzo. Robots with their heads in the clouds. *IEEE Spectrum*, 48(3):16–18, 2011.

[12] In-Stat. Number of 3d mobile devices will surpass 148 million in 2015. *http://www.instat.com/press.asp?ID=3232&sku=IN1105069SI*, 2011.

[13] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2010.

[14] A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *Pattern Analysis and Machine Intelligence (PAMI), IEEE Transactions on*, 21(5):433–449, 1999.

[15] M. Johnson. Generalized descriptor compression for storage and matching. In *British Machine Vision Conference*, 2010.

[16] A. Kirmani, A. Colaco, F. Wong, and V. Goyal. Codac: A compressive depth acquisition framework. In *Int. Conf. Acoustics, Speech, and Signal Proc.*, 2012.

[17] L. Li and K. Chakrabarty. On using exponential-golomb codes and subexponential codes for system-on-a-chip test data compression. *J. Electron. Test.*, 20(6):667–670, 2004.

[18] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[19] A. S. Mian, M. Bennamoun, and R. A. Owens. On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. *International Journal of Computer Vision (IJCV)*, 89(2-3):348–361, 2010.

[20] M. Narozny, M. Barret, and D. T. Pham. Ica based algorithms for computing optimal 1-d linear block transforms in variable high-rate source coding. *Signal Processing*, 88(2):268–283, 2008.

[21] D. Nistr and H. Stewnius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[22] P. Papadakis, I. Pratikakis, T. Theoharis, G. Passalis, and S. Perantonis. 3d object retrieval using an efficient and compact hybrid shape descriptor. *Eurographics Workshop on 3D Object Retrieval*, 2008.

[23] J. Smisek, M. Jancosek, and T. Pajdla. 3d with kinect. In *ICCV:CDC4CV Workshop*, pages 1154–1160, 2011.

[24] F. Tombari, S. Salti, and L. Di Stefano. Unique shape context for 3d data description. In *ACM workshop on 3D object retrieval*, 2010.

[25] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *European conference on Computer Vision (ECCV)*, 2010.

[26] F. Tombari, S. Salti, and L. Di Stefano. A combined texture-shape descriptor for enhanced 3d feature matching. *Int. Conf. Image Processing (ICIP)*, 2011.

[27] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[28] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Neural Information Processing Systems (NIPS)*, 2008.

[29] S. Winder, G. Hua, and M. Brown. Picking the best daisy. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[30] C. Yeo, P. Ahammad, and K. Ramchandran. Rate-efficient visual correspondences using random projections. *Int. Conf. Image Processing (ICIP)*, 2008.

[31] A. Zaharescu, E. Boyer, and K. Varanasi. Surface feature detection and description with applications to mesh matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[32] Y. Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *ICCV:3DRR Workshop*, 2009.