Cloud-Scale Image Compression Through Content Deduplication

David Perra
perra@cs.unc.edu
Jan-Michael Frahm
jmf@cs.unc.edu

Department of Computer Science
University of North Carolina
Chapel Hill, NC

Abstract

As more images are uploaded to the cloud, it is imperative that pixel-wise redundancy between images be leveraged to minimize each photo's memory footprint. We present an efficient cloud-scale digital image compression scheme for use in cloud image storage systems. Unlike current state-of-the-art systems, our image compression technique takes full advantage of redundant image data in the cloud by independently compressing each newly uploaded image with its GIST nearest neighbor taken from a representative set of uncompressed images. We leverage state-of-the-art video compression techniques in order to efficiently reuse image content which is already stored server-side. Our novel scheme scales to accommodate large datasets and is highly parallelizable for cloud computing. Experimental results demonstrate that our algorithm produces competitive image compression rates while reducing the computational effort by at least an order of magnitude in comparison to competing techniques, all while providing the necessary scalability for cloud-scale image compression.

1 Introduction

The advent of affordable consumer-grade digital cameras has caused the quantity of personal photographs to explode over the past two decades. Since then, consumers have been largely responsible for managing and maintaining their own personal photo collections. In recent years, cloud storage systems such as Google Drive, Microsoft OneDrive, and Facebook have gained popularity as convenient services for hosting personal media files; in 2013, Microsoft claimed that OneDrive had over 250 million users [15], and Facebook revealed that its 1.15 billion users upload over 350 million new photos each day [18]. As the size and number of photos continues to grow, hosting billions or trillions of photos will become a prohibitively expensive task for cloud platforms due to hardware, software, and power constraints. Shi *et al.* [17] postulate that maintaining only 200 billion photos in the cloud would cost over $30 million US dollars each year before even considering server power, cooling, space, and manpower. Thus, finding ways to minimize these rapidly increasing storage costs is a priority for any cloud service.

Ideally, a given image could be recomposed on a pixel or patch basis using several different photographs which have already been stored in the cloud. Personal photo collections are prime candidates for redundant pixel removal since they often depict the same subjects in

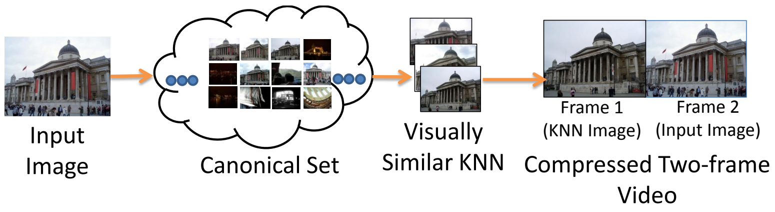

Figure 1: A high-level overview of our proposed technique. A canonical set, consisting of potentially millions of images of varying subjects, can be leveraged to find a set of images which are visually similar to a query image via *k*-nearest neighbors search. Those visually similar images are each used to compress the query image by use of a video codec such as H.265. The two-frame video which yields the best compression to quality ratio will be selected as the output of the pipeline, and the bits associated with the input image will be stored.

common locations. Repeated pixel arrangements across multiple photographs can be identified and reused to prevent storing the same information multiple times. An extension of this idea is to utilize the big photo data on the web to find near-duplicates of a given photo, using the content found in the near-duplicate images to reconstruct the current photo.

In addition to personal photo collections, photo redundancy is especially prevalent in web-hosted datasets, which include specific monuments, landmarks, or frequently geo-tagged areas. Many 3D reconstruction techniques use such datasets for precisely this reason [6]. These datasets tend to be prime candidates for large-scale image compression because they have an incredible amount of photos of the same general subjects and structures.

Conceptually, two images with high appearance redundancy can be seen as two subsequent video frames. This insight enables us to leverage efficient state of the art video compression methods such as H.265 [19]. In fact, these algorithms are implemented in hardware which can be found even in mobile phones. In these schemes, each frame of an image sequence is categorized as an I-frame, P-frame, or B-frame. The amount that each frame can be compressed is dependent upon the frame type. I-frames are compressed independently and can be recovered without information from other images in the sequences. P-frames and B-frames, on the other hand, can reference macroblocks of pixels in one or more reference images, respectively, in order to compress the image. As such, P-frames and B-frames obtain higher compression rates than I-frames [16]. A balance must be struck between compression rate and visual quality; systems which are willing to accept small amounts of compression artifacts would be able to increase compression rates by changing the compression parameters of the H.265 codec. In our proposed work, we set these parameters to maintain the original image quality.

The challenge, then, is to find images which have sufficient visual similarity to allow a video compression format to maximize the bit savings and minimize the visual artifacts and the time taken to encode and decode each photo. To accomplish this, we propose the use of previously uploaded photos in the cloud as a canonical set of images, which can be used to compress the pixel data of future image uploads. This canonical image set is finite in size and, ideally, represents a majority of commonly photographed subjects. It should be noted that images found within the canonical set can also undergo compression themselves.

Our proposed technique takes a newly uploaded, uncompressed query image, and finds its most visually similar counterparts within the canonical set by performing a k-nearest neighbors (KNN) search over a binarized GIST representation of all photos [12]. Then, a video compression scheme such as H.265 can be applied to neighbor-query image pairs, forcing each neighboring image from the canonical image set to be the I-frame in each two-frame video. Finally, we can save the portion of the resulting bit-stream, which corresponds to the query image (along with other metadata describing the canonical image that was used to compress the query image). Figure 1 visualizes our proposed pipeline's processing steps.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 describes our proposed cloud-scale image compression scheme and details the process of retrieving visually similar images. Section 4 presents the results of our work and explores its performance implications. Finally, Section 5 concludes our work.

2 Related Work

Techniques to remove redundant pixel data between images can be broken down into two classes: representative signal techniques [1, 22], and visual clustering techniques [9, 17, 24]. Representative signal techniques work by finding a common low-frequency signal within a set of images; compression rates are highly dependent upon the ability to align all images within the set. As such, representative signal is more useful for image sets in the realms of medical and satellite imagery, which commonly have multiple accurate and photoconsistent images in a given image set.

Visual clustering techniques, on the other hand, do not require tight alignment between images. Instead, they focus on sharing and reusing pixel data between multiple images. Visual clustering techniques model the relationship between images as a directional graph. The nodes of the graph are either the images or a set of image descriptors, whereas the edges represent a quantifiable distance between the nodes (usually using local feature descriptor matching criteria). Following a path through this directional graph describes an image pseudosequence for use in image reconstruction [17, 23, 24]. Image reconstruction consists of one of the following: 1) the warping, correction, and combination of one or more reference images to produce a target image [17], or 2) the stitching and blending of one or more unwarped patches from different reference images to form a target image [23, 24]. Generally speaking, these pseudosequences act as chains of frames, which are used to create interdependent encodings and decodings to maximize bit savings [17]. Many recent visual clustering techniques require a set of local features to identify patches from different images that can be combined to form a target image. However, some techniques in large-scale image completion have reported success with creating directed graphs for images using GIST instead of local image descriptors [7]. Most of the visual clustering compression papers only compare their results to H.265 intra-coding but forgo the benefits of H.265 inter-coding [9, 24], which our approach leverages. Shi *et al.* [17] claim that H.265 inter-coding outperforms H.265 intra-coding. We intend to further explore this space: our approach leverages inter-coding by applying H.265 to form a two-frame video with the query image as the P-frame and the most visually similar image from the canonical set as the I-frame.

In most modern cloud storage systems, photos are saved as JPEG files [20]. One important reason why cloud storage systems haven't migrated to more memory efficient compression techniques is because random access is important to maintaining low latency access times for users. Due to alignment difficulties, representative signal techniques do not translate well to arbitrary photographs across multiple users. However, visual clustering techniques tend to be too slow and have too many decoding steps to provide random, ondemand access of any particular photo. By maintaining one-to-many correspondences between canonical images and query images, and by never compressing query images with photos outside of the canonical set, we can also avoid the time and complexity of multiple encoding and decoding steps.

Systems which create and compress pseudosequences out of a single user's photos obtain high quality and compression rates but are slow performers for compression and image serving. Direct random access of a frame in a video compressed psuedosequence has high overhead because most frames in the pseudosequence will be P-frames or B-frames, and these frames are dependent upon the successful decoding of previous or future frames in the pseudosequence. Additionally, if geometric and photometric transformations need to be performed for each image decoding step then decoding a random image of an H.265 compressed image set will take a non-trivial amount of time to complete.

Zou *et al.* [24] attempt to manage and compress personal photo collections by building a minimum spanning tree (MST) out of the photo collection and using paths from the MST's root as compressible image pseudosequences. They define their similarity metric by an all pairs patch-wise SSD computation. Each image pseudosequence is then subject to H.265 compression. Their technique performs well for small photo sets, but the all-pairs nature of the algorithm does not allow for scalability. Additionally, focusing only upon individual users' photo collections does not allow use of the redundant data from other users' photo collections, which may contain more visually similar photos. In contrast, our approach leverages millions of pre-existing images in the cloud in order to pick the best candidate for H.265 compression. We avoid building an MST by compressing images using a large canonical image set, so our algorithm is more robust to adding and removing photos; there are no album specific encoding or decoding dependencies generated by our algorithm.

Most recently, Shi *et al.* [17] propose an image reconstruction technique that achieves state of the art results and shows robustness to small, sparse datasets with challenging geometric variations between images. However, if millions of users' photos are leveraged during compression, it is much more realistic to assume that a given photo has a visually similar neighbor in the canonical image set. This should drastically improve H.265's performance. Although our technique is less robust to challenging photometric and geometric deformations, it attempts to avoid these deformations altogether by taking full advantage of the sheer number of photos present in the cloud. This also reduces the run time of our technique by orders of magnitude.

Given that our approach aims to find a visually similar image pair out of millions or billions of existing photo uploads, brute-force local feature matching is impractical. Although Shi *et al.* [17] perform a clustering step to find visually similar images before performing local feature matching, their clusters can grow in an unconstrained manner. Shi *et al.* [17] cluster based on SIFT descriptors which bears high computational cost. Weiss *et al.* [21] use spectral hashing in order to quickly identify visually similar photos from within a dataset. On the other hand, Douze *et al.* [5] showed that GIST descriptors are an efficient and scalable choice for finding near-duplicate images in web-scale image sets. Frahm *et al.* [6] also showed that GIST is effective for performing viewpoint grouping based upon appearance clustering. For these reasons, we opt to represent all images as GIST descriptors for the purposes of finding near-duplicates of a query image from within our canonical image set.

Yue *et al.* [23] propose an image reconstruction technique which leverages local patches from more than one thousand images. Patches are mapped from a canonical set of 1491 images to various patches on the query image by using large-scale SIFT matching. While their technique produces excellent visual results, their local feature extraction and matching operation isn't scalable. Additionally, their small canonical set limits the number of photos that they can reconstruct. Their technique fails if no visually similar photos exist since a lack of similar pixel patches prevents target image reconstruction, whereas the compression rate of our technique gracefully degrades under the same circumstances. Moreover, our choice of GIST over SIFT allows us to match a query image to visually similar images within a large canonical set, improving our scalability.

3 Proposed Cloud Image Compression Technique

The primary goal of our approach is to efficiently compress a user's photos at cloud-scale by reusing similar image data that already exists in the cloud (we refer to this data as the canonical image set). In the following sections, we describe how to obtain a canonical set and how it can be leveraged for cloud-scale image compression. We explain how to build and use a single canonical set for a single geographic region, although it is straightforward to allow our technique to operate with multiple canonical sets for multiple geographic regions by using input image geotag metadata or by inferring geographic information from the input image's contents [4, 8].

3.1 Building a Canonical Set

Our key insight is that many photographs uploaded to the cloud are highly likely to have similar pixel patches, especially near landmarks and other commonly geotagged areas – even within the home of the user. Thus, we assume that the canonical set is a randomly selected, finite set of photos that is composed of tens or hundreds of millions of images depicting commonly photographed subjects and structures. Constructing such a set can be done, for example, by randomly sampling all photos currently stored in the cloud. Alternatively, techniques like Frahm *et al.* [6] and Raguram *et al.* [14] can be used to construct such a canonical set through iconic scene graphs. This process should naturally yield many views of popular subjects as more photos of those subjects are uploaded to the cloud. A sufficiently large canonical set contains enough photos to have a visually similar image for a large majority of photos uploaded in the future. Similarly, we foresee complementing the general canonical set with a user-specific canonical set if desired.

One important detail observed by Hays *et al.* [7] is that, because GIST does not encode high-level semantic information, a sufficiently large canonical image set must be used in order to allow visually similar images to be consistently returned during a visual similarity search. Their empirical results show that significant amounts of data enables GIST to perform well as a mechanism to find visually similar images. In short, larger canonical image sets will contain visually similar images with higher probability at the expense of slightly longer search times. Note that we have empirically observed that, even with the search time increase, search times for a visually similar image are dwarfed by the time taken to execute H.265 on the frames.

An initial construction of a canonical image set will probably not contain visually similar images for all future queries. Picking an entirely new, larger canonical set is impractical as it would require many images to be recompressed against new canonical images. Even if the current image upload rates remained constant, this would be computationally prohibitive and likely cause further degradation of image quality. As such, it's necessary to devise methods

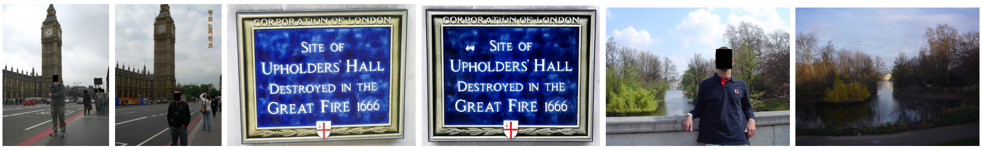

Figure 2: Examples of query images are depicted as the left-side image of each image pair. Each query image's nearest neighbor (as returned by our KNN code, leveraging a 512 bit binarized GIST descriptor) returned from the canonical set is depicted as the right-side image of each photo pair. Photos coutesy of Flickr.

for growing a pre-existing canonical set in a controlled manner while still maximizing the bit savings.

One promising way of growing the canonical set is to make a best-effort attempt at compressing all query images and then analyze each query image's resulting compression rate. If the compression rate is not high enough then that query image could be added to the canonical set and used as a potential reference image for future uploads. Alternatively, it may be the case that several new image queries do not compress well. In this scenario, it may be worthwhile to add one or more iconic images from this set of new image uploads to the canonical set; this iconic image could then be used as an I-frame in another query photo's compression.

3.2 Finding a Visually Similar Image from the Canonical Set

Given a recently uploaded image, Q, we want to retrieve the k most visually similar images, N_k, from the canonical image set, C. To do this, we need to pre-process the canonical data set in order to allow for a very fast k-nearest neighbors (KNN) query. We compute a GIST representation of each image of the canonical set in parallel using a CUDA-optimized GIST implementation [11]. Still, storing a 368-float GIST descriptor for every element of the canonical set, C, is not memory efficient when handling multiple millions of images. Additionally, when performing a KNN operation, it's important to fit as many GIST descriptors into GPU-memory as possible in order to minimize the computation time. Thus, we compress the GIST descriptors through a binarizing process using a locality sensitive scheme [2, 13]. Each GIST descriptor is reduced to a 512 bit binarized string; we pick a 512 bit feature vector because it has been shown to produce a good balance between descriptor size and discriminative ability [6].

After precomputing all binarized GIST descriptors for the elements in the canonical set, C, we can perform KNN on the query image, Q. We compute the query image's 512 bit binarized GIST representation and use it to find its nearest neighbors among the binarized GIST descriptors of the images in the canonical set. The KNN output will describe a set of images from the canonical set, which are visually similar to the query image, N_k. Ideally, the nearest neighbors will be near-identical to the query image. Each of these k most visually similar images will be used to compress the query image by use of a video codec, as described in subsection 3.3. An example query image and its most visually similar image from a canonical set is depicted in Figure 2.

3.3 Compression details

To combat the issues associated with compressing and decompressing images using long pseudosequences, we require all photos in the canonical set to act as I-frames (frames which only require intra-coding) for the H.265 compressed output. This allows all images in the canonical set to remain disjoint. Then, when a query image Q finds its visually nearest neighbor N from within the canonical set, C, the nearest neighbors N_k will act as an I-frame and the query image Q will act as a P-frame. This establishes a one-to-many correspondence between images in the canonical set and query images and prevents query images from becoming dependent upon images not in the canonical set.

Note that our approach does not require H.265 — it simply stands as a design choice in our system. This technique works with different types of video compression schemes (although results will vary based on the individual parameters of those compression schemes).

In the next section, we discuss the experimental setup of our approach and compare our results to the state of the art.

4 Experimental Results

In our experiments we use a canonical set of approximately 13 million random images of London, which were downloaded from Flickr. No particular size, data, or content constrains were placed on the downloaded photos. Indeed, noise in the dataset can exist through mislabeled or mis-tagged photos. However, because we randomly sample images of London to generate the canonical set, noise in the dataset will not have a significant impact upon our results. Our choice to only use a London dataset is simply a choice of scope for this paper; work done by Crandall *et al.* [3] suggests a way of building and organizing a global canonical set.

Our method's canonical set is preprocessed as described in Section 3.2. A gaming PC was used to conduct all of our experiments, and all code (both KNN and compression) was parallelized and multi-threaded in order to maximize compression throughput. Images returned by the KNN operation are not guaranteed to be the same size or aspect ratio as the query image. Under these circumstances, we simply allow each codec to resize and rescale the KNN images to make them the same size as the query image.

4.1 Analyzing the Efficiency of KNN Over a Large Canonical Set

In our first experiment, we analyze our k-nearest neighbor operation over the binarized GIST descriptors of the canonical image set, C. The goal of this experiment is to determine how large k must be to allow our KNN operation to return a visually similar image, maximizing the compression rate of a query image.

We vary the number of retrieved nearest neighbors, k, in order to evaluate how the compression results change and to determine how many nearest neighbors must be returned before we can be confident that we've found the most similar image in the canonical set. For each trial, we compress each query image (7665 total) with each of its k-nearest neighbors by using H.265 compression and then record the total elapsed time (for both KNN and compression) as well as the peak signal-to-noise ratio (PSNR) and the best compression rate for each photo. The results are provided in Table 1. Our results show that the nearest neighbor generated by our KNN operation is typically the most visually similar image from the canonical set. Seeking out additional neighbors above k=1 produces diminishing returns, allowing for a small percentage of additional compression at the expense of significantly higher run-times. This experiment shows that small values of k are sufficient when focusing upon compression speed and scalability while maintaining high image quality.

Table 1: Comparing Different Values of k in KNN

	k=1	**k=4**	**k=9**
Average PSNR	40.46	40.0	40.2
Average Bit Savings (% size reduction)	74%	76.0%	76.5%
Time per Image (seconds)	0.17	0.65	1.5

4.2 Comparing Our Proposed Scheme to the State of the Art

In our second experiment, we show our technique's performance in comparison to other state-of-the-art techniques. To demonstrate the scalability of our algorithm, we take our query images to be 76,526 images [1]. These images were not a part of the canonical set.

Table 2 shows how our technique performs with respect to the works of Shi *et al.* [17], Zou *et al.* [24], and Yue *et al.* [23]. The timings presented in Table 2 are end-to-end times starting with the query image submission, the KNN operation, the compression, and the final image recovery after decoding. Our own timings also include the decoding and PSNR measurements; the competing techniques made no mention of whether they measured PSNR as a part of their reported timings. No code was made publicly available for the competing techniques so we simply present the run-times reported in their respective publications.

Our technique scales to operate upon tens thousands of query images while achieving state-of-the-art run-times and competitive compression rates. Because the other techniques [17, 23, 24] use all-pairs local feature matching in order to establish relationships between the images, their run-times would be significantly worse if they had been subject to datasets composed of millions of images.

Table 2: Comparison to State-of-the-art Techniques

	Ours	**[17]**	**[24]**	**[23]**
Canonical Set Size	**13,000,000**	7	150	1,491
Images Compressed	**76,526**	7	150	10
Average PSNR	**40.46**	39	38.5	21.32
Average Bit Savings (% size reduction)	74%	**96%**	75%	81.5%
Time per Image (seconds)	**0.17**	256	8	>10

4.3 Reducing the Canonical Set

The act of minimizing the canonical set is highly dependent upon the compression goals for the canonical set. Ideally, the canonical set would be composed of exactly the number of images needed to represent all possible input photo appearances. In practice, this kind of

[1] Note that we do not use the competing works' datasets for direct comparison. In the case of Shi *et al.* [17], the dataset links in the paper were broken and the authors were not responsive to our inquiries. Zou *et al.* [24] did not make their test photo collections publically available. Yeu *et al.* [23] used a very sparse dataset which did not contain enough images to construct a proper canonical image set (see Section 3.1).

canonical set is impossible to construct. Instead, the canonical set should be focused upon approximating the ideal canonical set as best as possible. Hence, our third experiment aims to explore the question "How small of a canonical set can be used in practice?".

We compare our results in Section 4.2 to results obtained from two smaller canonical sets (one of size 500,000, and another of size 3,000,000, both of which consist of randomly chosen images of London). The nearest neighbor for each query image is found from the reduced canonical set and is used to carry out the compression described in Section 3. We evalute each canonical set by using tens of thousands of query images. Our results, shown in Table 3, suggest that the size of the canonical set does not affect the compression results as much as one would intuitively believe. We observed that reducing the canonical set a factor of 24 only reduced the compression rate by 11 percent. However, the execution time for the compressions dropped by 29.4% and the overall size of the canonical set was approximately 24 times smaller. Hence, systems which require faster compression timings or which may constrain the size of the canonical set can still use our method to great effect. As the canonical set shrinks, our time per image decreases and our compression rate gracefully degrades. Note that the time per image does not decrease linearly since the H.265 operations are the bottleneck; fortunately, this bottleneck can be eliminated by employing commercially available H.265 compression hardware [10].

It should be noted that there is a theoretical limit to how small the canonical set can become before our technique's compression decays to an unacceptable level. To reiterate, smaller canonical sets imply sparser sampling of the domain of the GIST descriptor. Hays *et al*. [7] independently observed that KNN over GIST descriptors degrades quickly as the number of samples decreases. However, because we are focused upon the non-degenerate case where big data is available, we leave the exploration of our technique under further canonical set size constraints to future work. Even under these unexplored circumstances, our compression will still degrade gracefully.

Table 3: Comparing Different Canonical Set Sizes

Canonical Set Size	13,000,000	3,000,000	500,000
Average PSNR	40.46	40.3	40.2
Average Bit Savings (% size reduction)	74%	69%	63%
Time per Image (seconds)	0.17	0.15	0.12

5 Conclusion

In this paper, we present a highly scalable algorithm, which is capable of reducing redundant image data in the cloud by leveraging previously uploaded photos. Unlike previous techniques, which use exhaustive all-pairs local feature matching in order to identify images with similar visual qualities, our approach represents images as high-dimensional points by using a binarized GIST descriptor. These binarized GIST descriptors lend themselves to an efficient GPU-enabled k-nearest neighbors implementation which can quickly identify visually similar photos within a canonical set of millions of images. Our technique is shown to provide competitive compression rates and can identify and remove duplicate pixel information at significantly higher speeds that enable online operation of our system at cloud-scale.

References

[1] Samy Ait-Aoudia and Abdelhalim Gabis. A comparison of set redundancy compression techniques. *EURASIP J. Appl. Signal Process.*, 2006:216–216, January 2006. ISSN 1110-8657. doi: 10.1155/ASP/2006/92734. URL http://dx.doi.org/10.1155/ASP/2006/92734.

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS '06. 47th Annual IEEE Symposium on*, pages 459–468, Oct 2006. doi: 10.1109/FOCS.2006.49.

[3] David J. Crandall, Lars Backstrom, Daniel Huttenlocher, and Jon Kleinberg. Mapping the world's photos. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 761–770, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-487-4. doi: 10.1145/1526709.1526812. URL http://doi.acm.org/10.1145/1526709.1526812.

[4] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei A Efros. What makes paris look like paris? In *ACM Transactions on Graphics (SIGGRAPH)*, volume 31, page 101. ACM, 2012.

[5] Matthijs Douze, Hervé Jégou, Harsimrat Sandhawalia, Laurent Amsaleg, and Cordelia Schmid. Evaluation of gist descriptors for web-scale image search. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, CIVR '09, pages 19:1–19:8, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-480-5. doi: 10.1145/1646396.1646421. URL http://doi.acm.org/10.1145/1646396.1646421.

[6] Jan-Michael Frahm, Pierre Fite-Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, and Marc Pollefeys. Building rome on a cloudless day. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision âĂŞ ECCV 2010*, volume 6314 of *Lecture Notes in Computer Science*, pages 368–381. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15560-4. doi: 10.1007/978-3-642-15561-1_27. URL http://dx.doi.org/10.1007/978-3-642-15561-1_27.

[7] James Hays and Alexei A. Efros. Scene completion using millions of photographs. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM. doi: 10.1145/1275808.1276382. URL http://doi.acm.org/10.1145/1275808.1276382.

[8] James Hays and Alexei A. Efros. Im2gps: estimating geographic information from a single image. In *in IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[9] Yang Lu, Tien-Tsin Wong, and Pheng-Ann Heng. Digital photo similarity analysis in frequency domain and photo album compression. In *Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia*, MUM '04, pages 237–244, New York, NY, USA, 2004. ACM. ISBN 1-58113-981-0. doi: 10.1145/1052380.1052413. URL http://doi.acm.org/10.1145/1052380.1052413.

[10] P.K. Meher, Sang Yoon Park, B.K. Mohanty, Khoon Seong Lim, and Chuohao Yeo. Efficient integer dct architectures for hevc. *Circuits and Systems for Video Technology, IEEE Transactions on*, 24(1):168–178, Jan 2014. ISSN 1051-8215. doi: 10.1109/TCSVT.2013.2276862.

[11] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, March 2008. ISSN 1542-7730. doi: 10.1145/1365490.1365500. URL http://doi.acm.org/10.1145/1365490.1365500.

[12] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001. ISSN 0920-5691. doi: 10.1023/A:1011139631724. URL http://dx.doi.org/10.1023/A%3A1011139631724.

[13] Maxim Raginsky and Svetlana Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In Yoshua Bengio, Dale Schuurmans, John D. Lafferty, Christopher K. I. Williams, and Aron Culotta, editors, *NIPS*, pages 1509–1517. Curran Associates, Inc., 2009. ISBN 9781615679119. URL http://dblp.uni-trier.de/db/conf/nips/nips2009.html#RaginskyL09.

[14] Rahul Raguram, Changchang Wu, Jan-Michael Frahm, and Svetlana Lazebnik. Modeling and recognition of landmark image collections using iconic scene graphs. *Int. J. Comput. Vision*, 95(3):213–239, December 2011. ISSN 0920-5691. doi: 10.1007/s11263-011-0445-z. URL http://dx.doi.org/10.1007/s11263-011-0445-z.

[15] Omar Shahine. Over 250m people using skydrive, May 2013. URL http://blog.onedrive.com/over-250m-people-using-skydrive/.

[16] T. Shanableh, E. Peixoto, and E. Izquierdo. Mpeg-2 to hevc video transcoding with content-based modeling. *Circuits and Systems for Video Technology, IEEE Transactions on*, 23(7):1191–1196, July 2013. ISSN 1051-8215. doi: 10.1109/TCSVT.2013.2241352.

[17] Zhongbo Shi, Xiaoyan Sun, and Feng Wu. Photo album compression for cloud storage using local features. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 4(1):17–28, March 2014. ISSN 2156-3357. doi: 10.1109/JETCAS.2014.2298291.

[18] Cooper Smith. Facebook users are uploading 350 million new photos each day, September 2013. URL http://www.businessinsider.com/facebook-350-million-photos-each-day-2013-9.

[19] Gary J. Sullivan, Jens-Rainer Ohm, Woojin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Trans. Circuits Syst. Video Techn.*, 22(12):1649–1668, 2012. URL http://dblp.uni-trier.de/db/journals/tcsv/tcsv22.html#SullivanOHW12.

[20] Gregory K. Wallace. The jpeg still picture compression standard. *Commun. ACM*, 34(4):30–44, April 1991. ISSN 0001-0782. doi: 10.1145/103085.103089. URL http://doi.acm.org/10.1145/103085.103089.

[21] Yair Weiss, Antonio Torralba, , and Robert Fergus. Spectral hashing. *NIPS*, 9(1), 2008.

[22] Chi-Ho Yeung, O.C. Au, Ketan Tang, Zhiding Yu, Enming Luo, Yannan Wu, and Shing-Fat Tu. Compressing similar image sets using low frequency template. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6, July 2011. doi: 10.1109ICME.2011.6011954.

[23] Huanjing Yue, Xiaoyan Sun, Jingyu Yang, and Feng Wu. Cloud-based image coding for mobile devices 2014;toward thousands to one compression. *Multimedia, IEEE Transactions on*, 15(4):845–857, June 2013. ISSN 1520-9210. doi: 10.1109/TMM.2013.2239629.

[24] Ruobing Zou, O.C. Au, Guyue Zhou, Wei Dai, Wei Hu, and Pengfei Wan. Personal photo album compression and management. In *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, pages 1428–1431, May 2013. doi: 10.1109/ISCAS.2013.6572124.