CONTEXT ADAPTIVE THRESHOLDING AND ENTROPY CODING FOR VERY LOW COMPLEXITY JPEG TRANSCODING

Xing Xu†, *Zahaib Akhtar*†, *Ramesh Govindan*†, *Wyatt Lloyd*†, *Antonio Ortega*‡

† Department of Computer Science, ‡ Department of Electrical Engineering
University of Southern California

ABSTRACT

The ever increasing quantity of user generated photos has created a growing storage burden on photo sharing services. This creates the need for compression techniques that take JPEG compressed images as inputs. In this paper we propose two novel very low complexity codecs, ROMP and L-ROMP to recompress JPEG photos. ROMP is a lossless JPEG recompression codec that achieves 15% average gains over JPEG, while L-ROMP is a lossy codec that can achieve 28% average compression gains over JPEG, by applying coefficient thresholding based on a perceptual criterion to a JPEG image before using the entropy coding of ROMP. ROMP and L-ROMP can significantly reduce the storage burden of photo sharing services; besides, internal bandwidth of transferring photos would be reduced as well.

Index Terms**—** JPEG, image compression

1 Introduction

In recent years, there has been a dramatic growth in the popularity of large-scale photo sharing services such as Facebook, Flickr, and Instagram. As of 2013, Facebook alone had 350 million photo uploads per day and stored over 250 billion photos [1]. Furthermore, the easy availability of high-quality digital cameras means that users are taking more photos, at higher resolution, and at higher quality. These trends combined with the user expectation that photos are stored indefinitely result in an ever growing storage requirement. While some of these services serve as archives (e.g., Flickr) and will store images indefinitely at their original resolution and quality, others, in particular image sharing services, are already known to *recompress* images for storage. In this paper we propose techniques to decrease significantly the storage requirements, with minimum impact on quality and access latency.

When a user *uploads* photos to photo sharing services, a transcoder typically reduces the photo file-size by resizing, reducing quality (e.g., through re-quantization), stripping off headers or a combination thereof. Some services may also transcode images to other more storage-efficient formats such as WebP or JPEG2000. When a user sends a request to *download* a previously uploaded photo, the transcoder converts the photo from the format used for storage in backend, to one suitable for delivery to a user device (adjusting image size and quality as needed).

Our proposed approach specifically targets aim at reducing storage costs, but also leads to additional benefits (e.g., reduction of internal bandwidth required for transferring images[a]) and is in general designed so that it can be seamlessly combined with existing system. This has three major implications. First, because JPEG[3] is by far the dominant format for image capture and sharing, our techniques will take JPEG images as input and provide JPEG images as decoded output. Second, most systems already transcode uploaded images to lower quality and resolution. Thus, we will apply our method to these lower quality and resolution images, rather than to the high quality JPEG images generated by cameras. Third, we will aim at very low complexity, in order to minimize latency on the download path and to lower transcoding cost and increase scalability in large scale systems.

[a]A companion submission explores the system level performance and benefits of our approach [2], while in this paper we focus on the image compression tools

Two types of re-encoding approaches could be considered, namely, i) techniques closely associated to JPEG, usually providing lossless performance, e.g., JPEG Progressive, JPEG Arithmetic or PackJPG, ii) alternative coding techniques based on more recently developed codecs, such as WebP or JPEG2000. We do not consider the latter approaches because these codecs are not as widely supported as JPEG among typical client devices, and would require excessive transcoding overhead (e.g., more than a second) if they were used solely for storage (and JPEG images had to be sent to the client devices). This will be discussed in Section 4.

Our approach leverages two unique attributes of photo sharing services. First, the encoder (at image upload) and decoder (at image download) are co-located and are both part of the storage infrastructure; second, constraints on memory usage can be relaxed. Based on these two facts, we introduce a novel JPEG recompression technique, ROMP (Recompression Of Many Photos) that is a direct extension of JPEG, but makes use of a large number of context-dependent Huffman tables. Because encoder and decoder are co-located, these tables can be stored along with the images, leading a negligible storage overhead in a large scale system. These tables allow ROMP to trade-off runtime memory for encoding speed, achieving some of the benefits of context adaptation but without the complexity associated to techniques such as context adaptive arithmetic coding.

To achieve additional gains, we also introduce L-ROMP (Lossy-ROMP), a fast, lossy recompression technique. Traditional recompression techniques (e.g., re-quantizing by changing JPEG's quality parameter) degrade image quality sub-optimally, especially when incremental storage savings are required (due to rounding effects when applying successive quantization). L-ROMP is based on DCT coefficient thresholding (setting some of the coefficients to 0), has very low complexity and allows full control of the additional distortion introduced. We introduce techniques for L-ROMP that use perceptual masking concepts to minimize the perceptual impact of thresholding.

To evaluate ROMP and L-ROMP, we perform experiments on several image sets including two 10,000 image sets from Facebook. We show that ROMP reduces storage requirements by 15% beyond

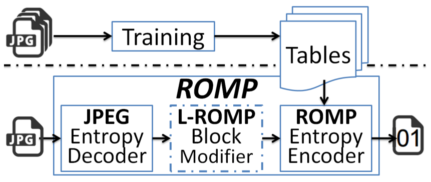

Fig. 1. ROMP Encoding Architecture

the JPEG standard compression while having low decompression time (30-50ms). Its performance dominates all image compression techniques we have been able to find: its competitors either have much higher decoding complexity, or much lower compression gain. L-ROMP can, in some cases, reduce storage requirements by 28% without affecting perceptual quality.

There exist some prior works that focus on lossless compression [4, 5, 6, 7, 8]. Compared to these schemes, ROMP, can achieve competitively high compression at very low complexity. Transcoding from JPEG to other compressed formats, such as WebP [9] or JPEG2000 [10] normally introduces high complexity due to transforming to pixel domain and encoding to the new format. L-ROMP avoids these conversions and always stays in DCT domain, which has advantages in terms of complexity. As will be described, successive changes of quantization parameters in JPEG can introduce significant quality degradations [11], L-ROMP, instead, only introduces small and perceptually lossless changes. L-ROMP is inspired by prior work on thresholding which optimizes the rate/distortion tradeoff [12, 13, 14], but is simpler and more computationally efficient.

2 Context-Adaptive Lossless Coder

In order to satisfy the system design constraints, ROMP makes use of a large set of entropy coding tables generated from a collection of images, where each of the tables is optimized for a specific context. Our approach proceeds block by block and does not involve any transformations or re-orderings of DCT coefficients, essentially using entropy tables that have exactly the same structure as that of a typical JPEG entropy coding table. This ensures that the complexity of the system is very low (see Fig. 1): essentially equivalent to JPEG entropy decoding followed by JPEG entropy coding.

As will be discussed next, in order to improve coding efficiency we take advantage of statistical dependencies that are exploited by other mechanisms (e.g., intra prediction, adaptive arithmetic coding) in state-of-the-art codecs. The key difference is that *by exploiting these dependencies using a large number of fixed pre-computed tables (which increases memory requirements for the codec) we have significantly lower complexity than competing approaches.* Note that this increase in memory is reasonable within a high performance photo sharing system, where there are essentially few constraints on memory usage.

Context-Sensitive Coding. ROMP exploits the freedom to have large coding tables by designing *context-sensitive coding tables* that result in lossless[b] compression. Recall that JPEG's Huffman tables are used to code *runsizes*, that is, information about a *run* of consecutive zeros followed by a non-zero coefficient of a given *size*. Huffman codes for these *runsizes* are designed based on their expected frequency of occurrence based on average image statistics. ROMP *learns context-sensitive* Huffman tables by learning the empirical probability of occurrence of *runsizes* from a corpus of images. This learning leverages the availability of such corpora in a large-scale photo sharing service. A Huffman table will contain the set of variable length codes assigned to each of the possible *runsizes* values, and each table will be optimized for a context based on position and energy information.

Position Dependence. Specifically, we consider *position dependent* tables, so that in principle a different table could be used for *each* of DCT coefficient position along the zig-zag scan. This allows us to use codes that exploit the fact that longer run-lengths are more likely as the frequency increases. This follows from the fact that, for natural images, it is known that non-zero coefficients are increasingly unlikely at higher frequencies [15].

Energy Dependence. Furthermore, we also take into account *energy-dependence* by creating additional contexts (for each position) based on the *energy* of other coefficients within the block (*intra-block* energy) and of neighbouring blocks (*inter-block* energy). Using intra-block energy provides additional information beyond position alone, e.g., in a block with less accumulated energy up to position $p-1$, the coefficient at position p is likely to be smaller. Inter-block energy exploits similarity between neighbouring blocks and is likely to be more effective for high resolution images, for which an 8×8 represents a relatively small portion of the image and neighbouring blocks are more likely to be similar.

To reduce the computation cost, for a given *runsize* that occurs at zigzag position p of the n-th block, we use the average of the observed coefficient sizes in a block as an estimate of intra-block energy:

$$intra(n,p) = \frac{1}{p-1}\sum_{i=1}^{p-1}\frac{SIZE(b_n(i))}{max_{SIZE}(i)} \tag{1}$$

where b_n denotes the n-th block, and $b_n(i)$ denotes the coefficient at position i, $SIZE(\cdot)$ denotes the bits required to represent the amplitude of the coefficient, $max_{SIZE}(i)$ is the observed maximum coefficient size for position i of images in the training set. Similarly, the inter-block energy value is estimated based on the average sizes of coefficients in nearby blocks: F nearby zigzag positions in B[c] adjacent prior blocks (coefficients at positions $p, p+1, \cdots, p+F-1$ of blocks $b_{n-1}, b_{n-2}, \cdots, b_{n-B}$):

$$inter(n,p) = \frac{1}{B\cdot F}\sum_{i=n-B}^{n-1}\sum_{j=p}^{p+F-1}\frac{SIZE(b_i(j))}{max_{SIZE}(j)} \tag{2}$$

Learning the Context-Sensitive Tables. ROMP uses a triple $< p,i,e >$ to define context: zigzag position p, intra-block energy i and inter-block energy e. It generates a Huffman table for each of these contexts from a training set of images. For *runsize* that occurs in any image in the training set, ROMP first determines its context triple and then gathers it together with other *runsizes* belonging to the same context. After gathering all the *runsizes* for each context, ROMP can generate a table for this context based on the number of occurrences of each, including *runsizes* that have 0 occurrence; if all the *runsizes* in one context are 0 occurrence, then a default Huffman table will be used. ROMP pre-defines 20 different energy levels for both intra-energy and inter-energy, which leads to ~64[d] $\times 20\times 20 = 25600$ different contexts and Huffman tables to be learned. These tables are quite different: take the example of End-Of-Block (EOB) symbol, which always takes 4 bits in default JPEG huffman table, ROMP's table of context $< 5,0,0 >$ uses 1 bit for it, while context $< 2,5,9 >$ uses 9 bits! These different Huffman tables allow ROMP to achieve better compression over standard JPEG.

Using Context-Sensitive Tables for En-/Decoding. Given the learned Huffman tables, the en-/decoding of ROMP is easy: ROMP

[b]Lossless with respect to the uploaded JPEG image.

[c]We use $F = 5$ and $B = 3$ in ROMP.

[d]There are 64 different zigzag positions in a 8×8 block.

parses the decoded symbols and for every *runsize* it computes the corresponding triple $< p, i, e >$, based on causal information, and uses the table corresponding to that context to encode *runsize*.

In summary ROMP operates as follows (Fig. 1): 1) From a training set of images, ROMP learns a Huffman table for each unique context (i.e., for each unique combination of position, intra- and inter-block energy), 2) When an image is uploaded, ROMP decodes it using default JPEG table, then uses the learned context-adaptive entropy tables to re-code the image and 3) Before delivering the image to the user, ROMP reverses its context-adaptive entropy code, and then applies the default JPEG entropy code.

3 Fast Near-Lossless Thresholding

ROMP's entropy coder is lossless with respect to the uploaded JPEG. In this section, we describe L-ROMP, which introduces loss (or *distortion*) in uploaded images as a way of achieving further savings in photo storage.

As discussed previously, users upload high quality JPEG images and many photo sharing services, e.g., Facebook, change the JPEG quality parameter (QP) to a lower level in order to ensure predictable storage usage, a step that introduces additional distortion [11]. Fig. 2 shows how generating a JPEG image with QP_B in two steps (i.e., encoding first with QP_A and then requantizing to QP_B) can be significantly worse than encoding directly the original with QP_B. In Fig. 2, "raw" corresponds to encoding the original raw image (raw$\rightarrow QP_B$), while "JPEG" corresponds to re-encoding a JPEG image (raw$\rightarrow QP_A \rightarrow QP_B$). This penalty arises from the rounding effects when a quantized coefficient is first reconstructed, then divided by a different quantization parameter and again quantized.

L-ROMP avoids re-quantizing coefficients, but introduces distortion by carefully setting *some* non-zero (quantized) coefficients to zero, a specific example of thresholding [12]. While more general forms of thresholding have been explored in other contexts, we are not aware of it being considered as an alternative to re-quantization in large photo sharing services. The intuition behind thresholding is that, by setting a well-chosen non-zero coefficient to zero, we decrease the number of *runsizes* to be encoded or generate an earlier end of block, in both cases reducing the overall rate. Optimization of coefficient thresholding has been considered from a rate-distortion perspective in the literature [16, 12]. Here we use a simplified version where only coefficients of size equal to 1 (i.e., 1 or -1) can be removed. This means that the distortion increase for any coefficient being removed will be the same[e]. Thus, we only need to decide if for a given coefficient the bit-rate savings are sufficient to remove it. We make the decision by introducing a *rate threshold* and only thresholding a coefficient if the bits saving by doing so would exceed this threshold.

However, setting too many coefficients to zero within a block can introduce local artifacts (e.g., blocking). Thus, L-ROMP uses a *perceptual threshold* T_p that limits the percentage of non-zero coefficients that will be set to zero. By doing this L-ROMP can guarantee that the block-wise SSIM respect to the original JPEG is always higher than $1 - \frac{T_p}{2-T_p}$. For example, if we use $T_p = 0.1$ (i.e., we can threshold at most 10% of the non-zero coefficients), then block-wise SSIM metric is guaranteed to be higher than 0.947. The proof of such bound is based on results from [17] and is omitted for brevity.

[e]Note that the actual MSE will be different if two coefficients have same value 1, but different frequency weights in the quantization matrix. However, by ignoring this difference we take into account the different perceptual weighting given to each frequency and obtain better perceptual quality.

Finally, L-ROMP can be easily introduced into ROMP's pipeline: before applying the context-sensitive entropy coding, L-ROMP's thresholding can be applied to each block (Fig. 1). No changes are required to ROMP's entropy coder.

4 Evaluation

In this section, we evaluate ROMP and L-ROMP, by comparing their compression and complexity performance to other state-of-the-art alternatives. Our evaluations use an implementation of ROMP and L-ROMP on top of `libjpeg-turbo` [18], a fork of the Independent JPEG Group `libjpeg` [19] SIMD [20] accelerated C library JPEG codec. Our implementation is optimized to reduce complexity, for example, by caching intermediate results. We compare ROMP and L-ROMP to all lossless JPEG codecs we are aware of that have publicly available implementations, including *JPEG Standard*, *JPEG Optimized*, *JPEG Progressive*, *JPEG Arithmetic*, *MozJPEG* [21] and *PackJPG* [22, 8]. For lossy codecs, we include WebP and JPEG2000 into comparison.

Our evaluation uses two sets of images. First we use two *Facebook image sets* each containing 10,000 images: one set has *medium* size images that are no bigger than 960×960, while the other set has *large* images that are no bigger than 2048×2048. Both sets are real user uploaded images, and we aim at providing convincing evaluation results. We also use the *Tecnick image set* [23]: 100 images of maximum resolutions 1200×1200 in raw format (PNG), which allows us to transcode them to JPEGs of different resolutions and different quality parameters. We use Tecnick because it is commonly used in image processing benchmarks and it allows us to evaluate on varying resolutions and quality parameters. For ROMP and L-ROMP, we randomly select a group of images for training and *test on the rest of the images*, e.g., for Facebook sets, we train on 1,000 and test one 9,000 images.

We evaluate our recompression schemes and alternatives on two metrics: compression ratio, and encoding/decoding time. *Compression ratio* is computed with respect to the original image's size in JPEG Standard: $\frac{s-s'}{s}$, where s' is the size of an image generated by a scheme and s is the file size of the image coded with JPEG. The *encoding time* is the time to recompress from JPEG and the *decoding time* is the time to decompress back to JPEG.

Compression Ratios. Table 1 (left) shows the compression ratio for all low complexity schemes[f] for Facebook image sets. We see that ROMP provides the highest compression ratio among lossless codecs. It achieves a 15% compression ratio over JPEG Standard. L-ROMP achieves a 28% compression ratio. Facebook currently uses JPEG Optimized, so the adoption of ROMP and L-ROMP would result in 13% and 26% compression ratios respectively beyond the current deployment. We further examine the distribution of compression ratios of images and find that ROMP provides a compression ratio over 13% for 95% of images while L-ROMP provides a compression ratio over 24% for 95% of images. This demonstrates both schemes provide good compression for almost all images on average.

We also evaluate using images of different resolutions and quality parameters, of the Tecnick image set. Table 1 (left) shows the most common parameters: resolution of 1200×1200, and quality parameter of 75. For ROMP and L-ROMP, we observe consistent compression results as compared to Facebook image sets (15%

[f] PackJPG, WebP and JPEG2000 are high complexity schemes that add unacceptable latency to photo downloads, so we do not evaluate them for the Facebook image sets.

		Compression Ratio (100%)			**Complexity (ms)**	
		Facebook		**Tecnick**	**Tecnick**	
		Medium	*Large*		*Enc.*	*Dec.*
Lossy	L-ROMP	28.13%	28.27%	28.04%	37	30
	WebP	N/A	N/A	29.71%	288	371
	JPEG2000	N/A	N/A	28.34%	624	719
Lossless	ROMP	15.47%	15.61%	15.02%	47	37
	PackJPG	N/A	N/A	21.53%	168	178
	Arithmetic	10.21%	11.63%	9.96%	40	40
	MozJPEG	6.48%	6.61%	5.85%	209	25
	Progressive	5.02%	5.07%	4.67%	91	26
	Optimized	2.93%	3.40%	1.71%	23	15

Table 1. **Left**: Compression ratio over JPEG Standard for Facebook image sets (medium and large) for low complexity schemes, and for Tecnick image set (1200×1200, QP=75). **Right**: Encoding and decoding complexity comparison on Tecnick image set (1200×1200, quality parameter is 75).

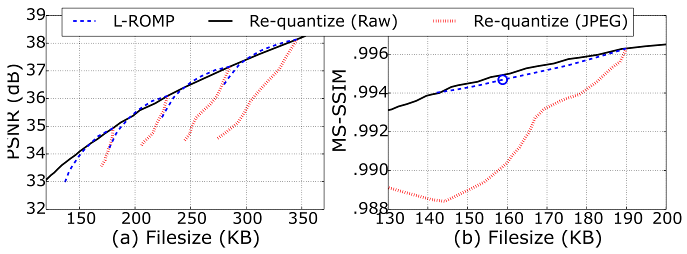

Fig. 2. Rate/distortion performance of L-ROMP, compared to re-quantization from raw image and from JPEG image using Tecnick images of 1200×1200. **(a)**: Using PSNR as the quality metric, and showing performance on JPEG images of four QPs (70,80,86,90). **(b)**: Using MS-SSIM metric, and focusing on JPEG images of QP=75; "o" marks the perceptually lossless setting of L-ROMP.

and 28%). For lossy alternatives, to make the comparison fair, we compare the compression ratio of the same achieved PSNR: we tune each lossy alternative's "quality degradation" parameter to provide the same PSNR of L-ROMP's perceptually lossless setting. Note that in all cases images are first encoded using JPEG and then re-encoded with one of the algorithms under consideration in order to mimic the settings in a photo sharing system where JPEG images are uploaded. WebP and JPEG2000 achieve similar compression ratios compared to L-ROMP[g], but with unacceptably high complexity as will be discussed next. In terms of lossless performance, we see that ROMP's compression ratio is over 15%; more importantly, ROMP's compression ratio increases as the quality parameter increases beyond 75, achieving up to 7% higher compression gain than JPEG Arithmetic, and 10% higher compression than Progressive and MozJPEG: given the trend towards higher quality images, this is a desirable property. PackJPG has higher compression gain, but its complexity is also significantly higher.

Complexity. Table 1 (right) compares the encoding and decoding complexity of ROMP, and L-ROMP against other alternatives. For encoding time, ROMP is comparable to JPEG Arithmetic, and much faster than other competitors. Compared to ROMP, L-ROMP's additional step of thresholding does not induce any extra complexity. Decoding time is the more relevant metric for photo sharing services because it affects user-perceived delay. ROMP's decoder is slightly faster than JPEG Arithmetic, comparable to JPEG Progressive and MozJPEG, and much faster than other schemes: 4.8× faster than PackJPG, 10× faster than WebP and 20× faster than JPEG2000. L-ROMP's decoder is identical to ROMP, but after thresholding the image becomes smaller, which makes it ~20% faster than ROMP, a more significant advantage comparing to alternatives. Interestingly, decoding is faster than encoding for most of the schemes; but for all the high complexity alternatives (PackJPG, WebP, JPEG2000), the decoding is actually slower.

This experiment shows that ROMP and L-ROMP achieve both high compression ratios ($15-28\%$) and very low complexity ($<$ 50ms encoding/decoding time for a 1200×1200 image). By contrast, the other high-compression scheme, PackJPG, has a compression ratio of 20%, but its decoding time is over 178ms, more than 5 times the complexity of ROMP[h]. Other lossy codecs have even higher complexities. We also conduct experiments on higher quality and resolution images to show how decoding time scales with each of these factors. In general, we observe that schemes with low decoding complexity scale well, including ROMP and L-ROMP; the decoding time for PackJPG, WebP and JPEG2000 scale poorly with image size and image quality. For example, PackJPG takes 531 ms to decode a 2048×1536 image, WebP needs 811 ms and JPEG2000 requires almost 3000 ms, limiting in photo sharing services unless the client can decode these formats.

L-ROMP Quality Evaluation We start the quality evaluation of L-ROMP, by comparing its rate-distortion performance to re-quantizing, from raw image, and from images in JPEG (Fig. 2). We use two different metrics: PSNR and MS-SSIM. For plot **(a)**, L-ROMP degrades PSNR more gracefully than simply re-quantizing to change the JPEG quality parameter. This happens because the latter suffers from cumulative errors due to successive re-quantization. We see that with conservative thresholds, L-ROMP's curve is actually higher than re-quantizing from the raw image curve, illustrating the efficiency of L-ROMP's trading distortion for bits-saving. For plot **(b)**, we use MS-SSIM as the metric and we focus on transcoding images of JPEG quality parameter of 75. As ROMP tracking the curve of re-quantizing from the raw image closely, it offers further validation for the assertion that L-ROMP provides a more graceful degradation in quality. We also conducted a subjective evaluation locally at our end, by developing a comparison tool [24] that can choose thresholds and shows the image at the chosen setting as well as size reduction. It allows us to compare resulting images from conservative thresholds to aggressive thresholds, and figure out the perceptually lossless setting that offers the most bits-saving. In particular, we find that using rate threshold of 2.0 and perceptual threshold of 0.4 provides maximum bits-saving without noticeable quality distortion (marked in Fig. 2 (b) using "o").

5 Conclusion

Motivated by the need for additional tools for managing storage in large photo sharing services, this paper explores the problem of image recompression and proposes two low complexity recompression schemes, ROMP and L-ROMP, that produce perceptually lossless compression with gains of 15-28% on a large corpus of images from Facebook. Compression gains of this magnitude can substantially reduce storage requirements at these services; this brings collateral benefits including internal bandwidth reduction, etc.

[g]Note that L-ROMP's algorithm is optimized for perceptual quality, not PSNR. L-ROMP can be tuned to optimize for PSNR, and that would lead to relatively higher compression ratio as compared to WebP and JPEG2000.

[h]In our companion submission, [2] we show that this difference in complexity can have a non-negligible impact on end-user latency

6 References

[1] Facebook, Ericsson, and Qualcomm, *A Focus on Efficiency*, September 2013.

[2] Xing Xu, Zahaib Akhtar, Ramesh Govindan, Wyatt Llyod, and Antonio Ortega, "Reducing storage in large-scale photo sharing services using low latency recompression," in *14th USENIX Conference on File and Storage Technologies (FAST 16)*, Santa Clara, CA, Feb. 2016, USENIX Association.

[3] Gregory K. Wallace, "The JPEG Still Picture Compression Standard," in *Commun. ACM*, New York, NY, USA, Apr. 1991, vol. 34, pp. 30–44, ACM.

[4] Ingo Bauermann and Eckehard Steinbach, "Further Lossless Compression of JPEG Images," in *In Proceedings of PCS 2004 Picture Coding Symposium, CA*, 2004.

[5] N. Ponomarenko, K. Egiazarian, V. Lukin, and J. Astola, "Additional Lossless Compression of JPEG Images," in *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, Sept 2005, pp. 117–120.

[6] Mahmud Hasan, Kamruddin Md. Nur, and Hasib Bin Shakur, "An Improved JPEG Image Compression Technique based on Selective Quantization," in *International Journal of Computer Applications*, October 2012, vol. 55, pp. 9–14, Full text available.

[7] I. Matsuda, Y. Nomoto, K. Wakabayashi, and S. Itoh, "Lossless Re-encoding of JPEG Images Using Block-Adaptive Intra Prediction," in *16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland*, August 2008.

[8] M. Stirner and G. Seelmann., "Improved Redundancy Reduction for JPEG Files," in *Proc. of Picture Coding Symposium (PCS 2007), Lisbon, Portugal, November 7-9, 2007*, 2007.

[9] Google, *WebP: A New Image Format for the Web*, Available at `https://developers.google.com/speed/webp/`.

[10] D.S. Taubman and M.W. Marcellin, "JPEG2000: Standard for Interactive Imaging," in *Proceedings of the IEEE*, Aug 2002, vol. 90, pp. 1336–1357.

[11] Stéphane Coulombe and Steven Pigeon, "Low-Complexity Transcoding of JPEG Images with Near-Optimal Quality Using a Predictive Quality Factor and Scaling Parameters," in *Image Processing, IEEE Transactions on*. 2010, vol. 19, pp. 712–721, IEEE.

[12] Kannan Ramchandran and Martin Vetterli, "Rate-Distortion Optimal Fast Thresholding with Complete JPEG/MPEG Decoder Compatibility.," in *IEEE Transactions on Image Processing*, 1994, vol. 3, pp. 700–704.

[13] Fu-Wing Tse and Wai-Kuen Cham, "Image Compression Using DC Coefficient Restoration and Optimal AC Coefficient Thresholding," in *Circuits and Systems, 1997. ISCAS '97., Proceedings of 1997 IEEE International Symposium on*, Jun 1997, vol. 2, pp. 1241–1244 vol.2.

[14] Viresh Ratnakar and Miron Livny, "Extending RD-OPT with Global Thresholding for JPEG Optimization," in *Proceedings of the 6th Data Compression Conference (DCC '96), Snowbird, Utah, March 31 - April 3, 1996.*, James A. Storer and Martin Cohn, Eds. 1996, pp. 379–386, IEEE Computer Society.

[15] Edmund Y Lam and Joseph W Goodman, "A Mathematical Analysis of the DCT Coefficient Distributions for Images," in *Image Processing, IEEE Transactions on*. 2000, vol. 9, pp. 1661–1666, IEEE.

[16] Antonio Ortega and Kannan Ramchandran, "Rate-Distortion Methods for Image and Video Compression," in *Signal Processing Magazine, IEEE*. 1998, vol. 15, pp. 23–50, IEEE.

[17] Sumohana S. Channappayya, Alan C. Bovik, Robert W. Heath Jr., and Constantine Caramanis, "Rate Bounds on SSIM Index of Quantized Image DCT Coefficients," in *2008 Data Compression Conference (DCC 2008), 25-27 March 2008, Snowbird, UT, USA*, 2008, pp. 352–361.

[18] Darrell Commander, *libjpeg-turbo*, Available at `http://libjpeg-turbo.virtualgl.org/`.

[19] Independent JPEG Group, *libjpeg*, Available at `http://libjpeg.sourceforge.net/`.

[20] Michael J. Flynn, "Some Computer Organizations and Their Effectiveness," *IEEE Trans. Comput.*, vol. 21, no. 9, pp. 948–960, Sept. 1972.

[21] Mozilla, *mozjpeg*, Available at `https://blog.mozilla.org/research/2014/03/05/introducing-the-mozjpeg-project/`.

[22] Hochschule Aalen, *PackJPG.*, Available at `http://www.elektronik.htw-aalen.de/packjpg/`.

[23] Tecnick.com LTD, *Tecnick Testimages*, Available at `http://www.Tecnick.com`.

[24] Xu, Xing and Akhtar, Zahaib and Govindan, Ramesh and Llyod, Wyatt and Ortega, Antonio, *Demo: Finding the Perceptually Lossless Setting for* L-ROMP, Available at `http://68.181.99.224/ROMP.html`.