K-Means Based Prediction of Transcoded JPEG File Size and Structural Similarity

Steven Pigeon
Département d'informatique et de recherche opérationnelle
Université de Montréal
pigeon@iro.umontreal.ca

Stéphane Coulombe
Département de génie logiciel et des technologies de l'information
École de technologie supérieure
stephane.coulombe@etsmtl.ca

Abstract

The problem of efficiently adapting JPEG images to satisfy given constraints such as maximum file size and resolution arises in a number of applications, from universal media access for mobile browsing to multimedia messaging services. However, optimizing for perceived quality (user experience) commands a non-negligible computational cost which in our work, we aim to minimize by the use of low-cost predictors. In previous work, we presented predictors and predictor-based systems to achieve low-cost and near-optimal adaption of JPEG images under given constraints of file size and resolution. In this work, we extend and improve these solutions by including more information about images to obtain more accurate predictions of file size and quality resulting from transcoding. We show that the proposed method, based on the clustering of transcoding operations represented as high-dimensional vectors, significantly outperforms previous methods in accuracy.

1. Introduction

The need for efficient image adaptation arises in a number of contexts, ranging from universal media access with varying browsing conditions (Han et al., 1998; Mohan, Smith, & Li, 1999), to multimedia messaging services (MMS) (Coulombe & Grassel, 2004). In the case of universal access, one uses a mobile device, either a smart-phone, PDA, or a tablet, to access resources or services on the Web. The traditional response has been to use rather crude adaptation strategies (Han et al., 1998) such as simply preparing a single “mobile” version of the resource (Fling, 2009), but this one-size-fits-all solution will leave users at both ends of the device capability spectrum dissatisfied: some will find the mobile version exceeding (or cumbersome for) their devices’ capabilities, while others will find it inadequate and lacking.

In the context of MMS, for another example, a receiving terminal is characterized by its capabilities—or more exactly its *limitations*—such as the maximum resolution of images it can display, the formats it can decode, and the maximum message size it can receive and interpret correctly (Open Mobile Alliance, 2010). Interoperability between MMS users will require server-side adaptation, as the sender’s device may be more capable than the receiver’s, and the receiving device will be unable to display correctly, if at all, a message that exceeds its capabilities. In this context, adaptation will require that the sender’s images are converted to comply with the receiver’s device capabilities, that is, changing the file size (by altering the compression parameters) and resolution of images (by scaling them). Adaptation can also include the case where the compression format itself needs to be changed. But this is seldom a problem since MMS image traffic is mostly composed of JPEG images taken from the devices’ cameras. Accordingly, we will neglect the case where the format also needs to be adapted (for example, from PNG to GIF) and concentrate on the prevalent problem of JPEG to JPEG image adaptation subject to changes in compression parameters (e.g., the quality factor) and scaling (resolution).

Therefore, whether in the context of universal access or multimedia messaging, the challenge is to adapt images to fit given constraints, dictated by the network conditions and the receiving device capabilities, while simultaneously maximizing the user experience and minimizing the computational cost of adaptation. In the context of high-volume service providing, whether for MMS or universal media access, only the fastest adaptation algorithms yielding the best perceived quality can be considered.

Of course, previous studies have addressed the problem of efficient image adaptation, but the solutions they propose are either still computationally expensive (and extensive modifications to existing JPEG manipulation libraries) (Ridge, 2003; Shu & Chau, 2005) or overly rigid, focusing on unrealistically constrained transformations such as scaling by powers of two (Lei & Georganas, 2002; Ratnakar & Ivashin, 2001; Ridge, 2003), or using a small, fixed, number of possible adaptations, without real consideration for the perceived quality resulting from adaptation. For example, Ridge’s method is accurate, but requires the JPEG image to be partly decompressed so that the DCT coefficients are available, on which successive re-quantization passes are performed until the quality factor yielding the largest file not exceeding the constraint is found (Ridge, 2003). Other methods exploit the structure of the DCT to yield fast scaling algorithms in the (partially) compressed domain by manipulating the DCT coefficients directly, but such methods also require the image to be partly decoded so that the DCT coefficients are available, and they are constrained to scaling by powers of two. Furthermore, it is unclear what the expected speed-ups are, as the DCT-coefficient based scaling algorithms are still relatively complex and may compare to an efficient implementation scaling using space-domain filters in terms of computational complexity. But, in our opinion, the principal shortcoming of previous methods is that they do not consider joint changes in compression parameters and scaling as a means of adaptation maximizing perceived quality.

In previous work, we have proposed low-cost predictor-based adaptation systems for the prediction of the file size and perceived quality resulting from an image subjected to simultaneous changes in compression parameters and scaling (Coulombe & Pigeon, 2009, 2010; Steven Pigeon & Coulombe, 2008). These lookup-table based constant time predictors, which we will refer to as JQSP1 (for JPEG Quality and Size Predictor) and JQSP2 in this work, are described in section 3, *Prediction Algorithms*. These predictors, unlike the solutions discussed in the previous paragraph, use only information that is readily available *without* decompressing the images, such as the original file size and the original quality factor (the parameter that controls the aggressiveness of compression) which can be accessed by reading only the file's header. These predictors are used in combination with an adaptation system (Coulombe & Pigeon, 2010; S. Pigeon & Coulombe, 2011, 2012) to predict the best transcoding parameters for adapting a given JPEG image subject to receiving terminal constraints, where "best" is defined as most likely to minimize perceived distortion under the considered viewing conditions as defined by the characteristics of the receiving device.

These predictors, although shown to perform well, do not make use of all the readily available information about the images such as resolution and bits per pixel, both of which are very likely to help formulate even more accurate predictions about file size and quality resulting from a given transcoding operation. However, the table-based schemes in our earlier work do not lend themselves easily to a larger number of parameters, and we believe that the uniform quantization of parameters (which was a key component of the methods' computational efficiency) is also an unwanted restriction for the problem at hand.

Therefore, in this work, we propose to extend and improve the solutions previously presented by the authors by including more information about images in order to help formulate more accurate predictions of file size and quality resulting from transcoding, and by lifting the restrictions of the previous methods, in particular the uniform quantization of parameters. To do so, we propose a method based on the clustering of transcoding operations represented as high-dimensional vectors.

The work is structured as follows: the next section, section 2, presents the proposed solution. Section 3 details the validation methodology as well as the algorithms of previous work. Section 4 presents the results from the proposed method as well as the results from the previously presented methods. In section 5, we discuss the results, accuracy, algorithmic complexity, and the memory usage of the algorithms considered. Appendix A discusses the efficient implementation of K-Means and prediction. We conclude in section 6.

2. Proposed Clustering-Based Solution

In this section, we detail the Enhanced JPEG Quality and Size Predictor (EJQSP), the solution we are proposing for the prediction of relative file size and quality resulting from a JPEG image adaptation based on clustering (Hastie, Tibshirani, & Friedman, 2009). We first describe the general problem of clustering, and then we describe its application to our particular objective.

The general clustering problem is as follows: we have n points in $\mathbb{R}^d$ (or other metric space), the $\{x_j\}_{j=1}^{n}$, which we want to partition into $C = \{C_1, C_2, C_3, \ldots, C_m\}$, m disjoint subsets. The partition C of X, per definition, is such that $\bigcup_{i=1}^{m} C_i = X$ and $C_i \cap C_j = \emptyset$, for $1 \le i \ne j \le m$. That is, the union of the disjoint subsets forms C. For each subset C_i, we elect a value, the prototype, denoted $\bar{x}_i$, deemed representative (under a given metric) of the elements of C_i. There are many sensible metrics one can use, but the metric considered here is the usual Euclidean distance, and the prototype $\bar{x}_i$ for subset C_i is given by

$$\bar{x}_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j \ ,$$

the L_2 centroid, and where $|C_i|$ denotes the cardinality of C_i. Let $\bar{X} = \{\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_m\}$ be the set of the prototypes. The goodness of a partition C (and of its corresponding set of prototypes $\bar{X}$), is assessed using the error function

$$E(C) = \sum_{i=1}^{m} \sum_{x_j \in C_i} \left\| x_j - \bar{x}_i \right\|^2 , \qquad (1)$$

and the goal is to find the optimal partition C^* that minimizes $E(C)$, that is,

$$C^* = \arg\min_C E(C) \, . \qquad (2)$$

Applying clustering to our problem of predicting the resulting (relative) file size and quality of an image subjected to changes in quality factor and scaling commands that we represent our exemplars as d-dimensional vectors encoding information about the original image, the transformation applied, and the quantities to be predicted, for example, relative file size and resulting quality. The transformations applied to the image, the change in compression parameters and in resolution, will be referred to as the *transcoding operation*. In this work, the transcoding operation describes only the change in quality factor and scaling; but it could also include more parameters such as, say, the chroma sub-sampling strategy (Pennebaker & Mitchell, 1993).

Let us define more notations. The original compressed image I_j is represented by a tuple (QF_j, w_j, h_j, f_j), where QF_j is the quality factor with which the image was originally compressed (and in this work we suppose that the quality factor complies with the Independent JPEG group definition, that is, it is an integer between 0 and 100 (IJG, 2012)), w_j and h_j are its width and height in pixels, respectively, and f_j, the original compressed file size of image I_j (possibly including extraneous data such as EXIFs and comments (ISO/IEC_CD_10918-5, 2011)). A transcoding operation (QF_{out}, z) describes the desired quality factor with which to recompress the image as well as $0 < z \leq 1$, a scaling factor. Applying a transcoding operation (QF_{out}, z) to an image I_j yields an image with resolution $zw_j \times zh_j$, quality factor QF_{out}, with perceived quality $q\left(I_j, QF_{out}, z\right)$ and resulting *relative* file size $f\left(I_j, QF_{out}, z\right)$, expressed as a ratio of the original file size f_j (let us note that in the original work of (Pigeon & Coulombe, 2008), relative file size rather than absolute file size was used in an effort to abstract the effect of absolute file size from the predictor; and in this work we will adhere to this convention). Both $f\left(I_j, QF_{out}, z\right)$ and $q\left(I_j, QF_{out}, z\right)$ are measured after the actual transcoding of image I_j. The transcoding operations are constrained to be such that $QF_{out} \in \{10,20,\ldots,100\}$ and $z \in \{0.1,0.2,\ldots,1.0\}$. While this is not required for the solution proposed in this work, we used these restrictions for computational reasons in previous work (Steven Pigeon & Coulombe, 2008) and they are included here for a fairer comparison of the methods.

The resulting quality measured between the original and the transcoded image is assessed by a quality metric. Ideally, it would be estimated using a MOS-like measure, but an accurate subjective evaluation of quality is difficult; we will ordinarily rely on simpler measures. In this work, we chose the structural similarity index, or SSIM (Wang, Bovick, Sheikh, & Simoncelli, 2004), to assess the perceived quality of resulting images. While PSNR has been a *de facto* standard for measuring image quality for a long time, SSIM is more robust to transformations that do not affect perceived quality, and is therefore deemed a better estimation of the user experience. Should the transcoded image resolution differ from the original (whenever $z \neq 1$), the transcoded image is scaled back to the original resolution for comparison. In all cases, scaling is performed using the Blackman filter, chosen for its spectral characteristics (Blackman & Tukey, 1959). In previous work, we assessed quality using a more sophisticated approach that depended on the receiving device's screen resolution and canvas (the maximum image size it can manipulate; not necessarily related to screen resolution); but in this work, for the sake of simplicity, we will omit such complications and consider only the case where the images are compared at the original image resolution.

To the original data from the image characteristics and the transcoding operation, we will further add *features*. The added features will consist of transformations of the original data used to put forward characteristics that would be otherwise impossible to discover using K-Means alone. Such features can be created randomly (for example a random linear combination of the original data), but they can also be constructed from *a priori* knowledge. The first feature we will use is the bits per pixel of the original image I_j, denoted b_j, which gives a measure of the image complexity. The second feature is the difference of quality factors, $QF_{out} - QF_j$, which gives information on the expected drop in quality and file size resulting from the change in quality factor. We will discuss features and their selection further in section 5.

The exemplars considered are therefore 9-dimensional vectors. The vector associated to an image $I_j = \left(QF_j, w_j, h_j, f_j\right)$ being applied a transcoding operation (QF_{out}, z) is therefore

$$x_j = \left(QF_j, w_j, h_j, b_j, QF_{out}, z, QF_{out} - QF_j, f, q\right), \qquad (3)$$

where, $b_j = {f_j}/{w_j\ h_j}$, and f and q stand for $f\left(I_j, QF_{out}, z\right)$ and $q\left(I_j, QF_{out}, z\right)$, respectively.

Solving eq. (2) exactly is an NP-Hard problem (Aloise, Deshpande, Hansen, & Popat, 2010; Mahajan, Nimbhorkar, & Varadarajan, 2009; Vattani, 2009) and therefore we will seek approximate algorithms such as K-Means (Lloyd, 1982), an algorithm very similar to LBG (Linde, Buzo, & Gray, 1980), which, while stochastic and in general sub-optimal, was shown to converge rapidly to good solutions under most circumstances (Bottou & Bengio, 1995).

The K-Means is an iterative algorithm and proceeds as follows: the initialization consists in randomly picking, without replacement, m vectors from X to serve as initial prototypes, the $\bar{X}_0$. At iteration $t = 1,2,\ldots$, for each vector $x_j \in X$, we find the closest prototype $\bar{x}_{t-1,i} \in \bar{X}_{t-1}$ amongst the prototypes of the previous iteration. That is, we find

$$i = \arg\min_i \left\| x_j - \bar{x}_{t-1,i} \right\|^2.$$

We then assign the vector x_j to the subset $C_{t,i}$. All vectors assigned to a same subset are then used to compute the prototype. Since in our case the metric is the Euclidean distance, the prototype is the average vector within set $C_{t-1,i}$, that is, $\bar{x}_{t,i} = \left|C_{t-1,i}\right|^{-1} \sum_{x_j \in C_{t-1,i}} x_j$ —had we used an L_1 metric, the prototype would have been the vector median, considerably more troublesome to compute (Barni, 1997). The algorithm iterates until satisfactory convergence is obtained, that is, the error $E(C_t)$ is not significantly smaller than the error $E(C_{t-1})$. In our experiments, we set the threshold to a relative difference of $\alpha = 10^{-6}$ or less. **Algorithm 1** details the procedure in pseudo-code, and Appendix A discusses the computational complexity and parallelization of **Algorithm 1**.

Algorithm 1. *K-Means*.

```
Inputs:   $X$, the exemplars
          $m$, the number of prototypes
          $\alpha$, the convergence threshold
$t \leftarrow 0$
$E_0 \leftarrow BIG\_NUM$ { "infinity" }
$\bar{X}_0 \leftarrow m$ random vectors from $X$ (without replacement)
$A \leftarrow \{0,0,\dots,0\}$ { the initial prototype assignment }
for all $x_j \in X$ do
  $a_j \leftarrow \arg\min_i \|x_j - \bar{x}_{0,i}\|^2$ { compute membership }
end for
repeat
  $t \leftarrow t+1$
  $n_t \leftarrow \{0,0,\dots,0\}$ { m elements, the $n_{t,i}$ }
  $\bar{X}_t \leftarrow \{0,0,\dots,0\}$ { m elements, the $\bar{x}_{t,i}$ }
  for all $x_j \in X$ do
    $i \leftarrow a_j$ { reuse membership }
    $\bar{x}_{t,i} \leftarrow \bar{x}_{t,i} + x_j$ { update prototype }
    $n_{t,i} \leftarrow n_{t,i} + 1$ { update the number of exemplars in this subset }
  end for
  for $i = 1$ to $m$ do
    $\bar{x}_{t,i} \leftarrow \bar{x}_{t,i}/n_{t,i}$ { normalize prototype }
  end for
  $E_t \leftarrow 0$
  for all $x_j \in X$ do
    $i \leftarrow a_j \leftarrow \arg\min_k \|x_j - \bar{x}_{t,k}\|^2$ { update membership }
    $E_t \leftarrow E_t + \|x_j - \bar{x}_{t,i}\|^2$ { update total error }
  end for
until converges($E_t$,$E_{t-1}$,$\alpha$)
outputs: $\bar{X}_t$, the prototypes after $t$ iterations.
```

The L_2 norm considered for the minimization of eq. (2) and the lack of a distance matrix in the problem formulation suggests that for best results, the exemplars must lie in an isotropic space; in other words, all dimensions should be spread along similar scales. If exemplars do not lie in such a space, the usual approach is to use principal component analysis or similar techniques to project the exemplars onto a vector space that provides isotropism (Hastie et al., 2009). Results, however, suggest that dimension-wise standardization suffices to provide satisfactory isotropy.

It only remains to decide on the number of prototypes, m. While allowing m to grow arbitrarily large reduces the training error (with zero error when $m = n$, for example), it is not desirable to have a very large m as the table holding the prototypes becomes very large. The number of prototypes has to be only as large as necessary so that at the local scale, the manifold onto which the exemplars lie appears approximately isotropic. The value of m is therefore subject to a number of trade-offs between prediction accuracy (that we should call generalization error, which differ from minimizing eq. (2), since eq. (2) is only concerned with the training exemplars, not all possible exemplars and it is possible that "all" the exemplars are distributed somewhat differently than the training set exemplars), the cost of storing the table, and the time for searching it. We discuss these issues in Appendix A. Fortunately, we will show that m need not be very large to outperform previously proposed predictors (Steven Pigeon & Coulombe, 2008).

3. Prediction Algorithms and Simulations

The data set used in all experiments is composed of approximately 73000 JPEG images collected from the Web using a crawler, with high-profile Web sites as origination points (Steven Pigeon & Coulombe, 2008). The data set was split into two disjoint parts, 90% and 10%, for the training and test sets respectively. As with previous experiments (Steven Pigeon & Coulombe, 2008), each image was subjected to 100 different transcodings (corresponding to all combinations $(QF_{out}, z) \in \{10,20,\ldots,100\} \times \{0.1,0.2,\ldots,1.0\}$), for which we observed resulting file size and perceived quality, yielding approximately 6570000 training exemplars and 730000 test exemplars, each modeled after eq. (3).

The first predictor we presented in (Steven Pigeon & Coulombe, 2008), denoted here JQSP1, uses a table look-up scheme to formulate its predictions. First, the original quality factor QF_{in} (corresponding to QF_j in this work), the desired output quality factor QF_{out}, and the scaling z are uniformly quantized to the desired precision, giving $\widetilde{QF}_{in}$, $\widetilde{QF}_{out}$, and $\tilde{z}$ (the tilde notation denotes quantized values throughout this paper). In (Steven Pigeon & Coulombe, 2008), we constrained both $\widetilde{QF}_{in}$ and $\widetilde{QF}_{out}$ to $\{10,20,\ldots,100\}$, and $\tilde{z}$ to $\{0.1,0.2,\ldots,1.0\}$, effectively indexing a $10 \times 10 \times 10$ array containing, in each cell, the relative file size and quality predictions corresponding to the tuple $\left(\widetilde{QF}_{in}, \widetilde{QF}_{out}, \tilde{z}\right)$. The predictions are simply formulated as the centroid (the average) of all training exemplars whose quantized parameters fall into a same cell.

The predictor introduced in (Coulombe & Pigeon, 2010), denoted here JQSP2, unlike the JQSP1, does not predict resulting file size nor quality, but the transcoding parameters maximizing quality under the constraint of file size. This allows the predictor to formulate finer transcoding operations than the coarser JQSP1, despite being optimized on the same training set. JQSP2 formulates predictions as follows: given an original quality $\widetilde{QF}_{in}$, a scaling factor $\tilde{z}$ and a target file size $\tilde{f}_{max}$, the algorithm formulates the prediction of the transcoding parameter $\widehat{QF}_{out}$ and resulting quality $\hat{q}$ (the hat notation will denote predictions) as the centroid of, for each unique image in the training set with original quality factor of $\widetilde{QF}_{in}$, the transcoding parameters maximizing file size without exceeding the constraint. The size of the table can be adjusted by varying the quantization on $\widetilde{QF}_{in}$, $\tilde{z}$, and $\tilde{f}$. In (Coulombe & Pigeon, 2010), we have that $\widetilde{QF}_{in} \in \{10,20,\ldots,100\}$, $\tilde{z} \in \{0.1,0.2,\ldots,1.0\}$, and $\tilde{f}$ varies from 0.001 to 1.0 by increments of 0.001, thus minimizing errors introduced by the quantization on f—since f represents *relative* file size, even a small fraction of the *original* file size may actually correspond to a large portion of the *target* file size.

The variant of K-Means described by Algorithm 1 is not very sensitive to the initial conditions (the randomly chosen $\bar{X}_0$) but it is not impervious to them either; its stochastic nature will require, to obtain a truly satisfying minimum, several independent optimizations. In our experiments, we opted for 30 independent optimizations (using different initial conditions but the same training set) and chose the optimization with the smallest error, as defined by eq. (1). The efficient implementation of K-Means is discussed in Appendix A.

All algorithms share the same training set (90% of the exemplars) and the same test set (the remaining 10%). Even if the exemplars are modeled after eq. (3), algorithms JQSP1 and JQSP2 use only QF_j (or QF_{in} in the original papers), QF_{out}, z, f, and q, ignoring the resolution information, w_j and h_j, and the features, b_j, $QF_{out} - QF_j$. Algorithm EJQSP, based on K-Means, will compute m clusters from the training set using the vectors as given by eq. (3). Tests were conducted on the remaining 10% of the exemplars using the three prediction algorithms, and we compared the actual resulting file size f and observed quality q against the predictions $\hat{f}$ and $\hat{q}$. The differences are reported as average absolute error as the quantities lie in $[0,1]$, using a mean square error would yield exceedingly small quantities (as, for example, $0.1^2 = 0.01$), thus exaggerating the method's performance.

4. Results

The JQSP1, JQSP2, and the proposed EJQSP predictors are compared, as described in the previous section, using the same test exemplars. For each test exemplar (formed according to eq. (3)), the predictors were asked to compute the predicted file size $\hat{f}$ and quality $\hat{q}$, and the errors $\hat{f} - f$ and $\hat{q} - q$ were measured. The EJQSP predictor was trained using arbitrary, but not unlikely, values of m, namely, 10, 25, 50, 100, 250, 500, 1000, 2000, 3000, 4000, 5000, 10000, and 20000. The values are chosen not only to show that the prediction accuracy increases as the number of prototypes increases, but also to show the graceful behavior of algorithm EJQSP as the number of allowed prototypes is *reduced*, and that EJQSP is amenable to trade-offs.

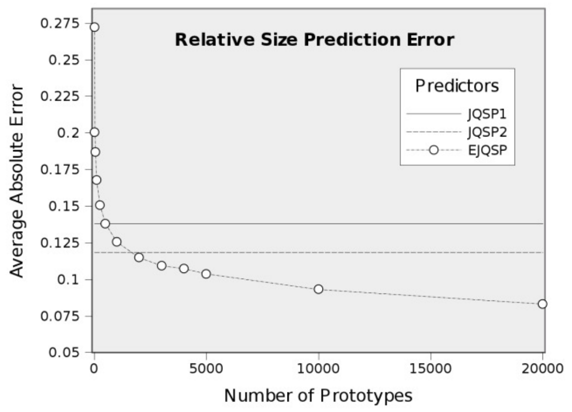


Figure 1. **Average absolute error for relative size prediction.**

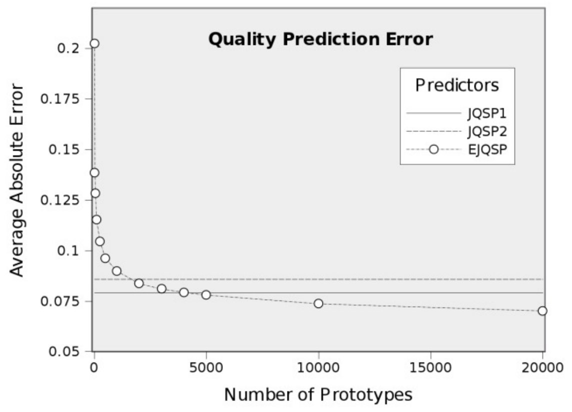


Figure 2. **Average absolute error for quality prediction.**

Examining Fig. 1, we see that the EJQSP predictor breaks even, on the accuracy of the prediction of the resulting file size, with predictor JQSP1 using only 500 prototypes and with JQSP2 using 2000. The performance difference continues to increase as the number of prototypes grows, to a point where, at 20000 prototypes, EJQSP has an error $\approx 40\%$ smaller than JQSP1, and $\approx 27\%$ smaller than JQSP2. While the prediction error is $\approx 10\%$ smaller with 20000 prototypes than with 10000, the gain is obtained at the cost of doubling the run-time, as we will discuss further in section 5. Fig. 2 shows similar behavior for quality prediction. EJQSP breaks even with JQSP2 using approximately 1500 prototypes and with JQSP1 at 4000; however, EJQSP ultimately yields an error that is $\approx 20\%$ smaller than JQSP2, and $\approx 12\%$ less than JQSP1. Both Figs. 1 and 2 show EJQSP accuracy increases smoothly with the number of prototypes (and the large number of exemplars allows the use of a model with a rather high capacity without risks of over-fitting (Hastie et al., 2009)).

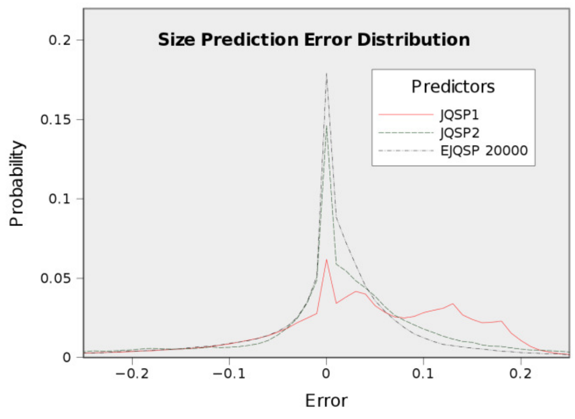


Figure 3. **Distribution of errors on file size prediction.**

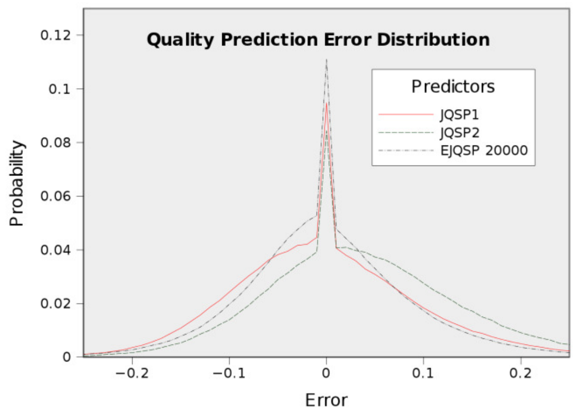


Figure 4. **Distribution of error on quality prediction.**

Figure 3 shows the distribution of errors $\hat{f} - f$ (the error of the predicted relative file size $\hat{f}$ against the observed file size, after transcoding, f) for the different predictors in this study. Examining Fig. 3, we can see that the distribution of errors from the JQSP1 predictor, despite peaking near zero, exhibits a strong skewness resulting in a definite propensity to overestimate the resulting file size. An algorithm using JQSP1 as its predictor will therefore tend to take (*a posteriori*) conservative decisions, and will likely fail to use the full file size budget, leading to images with a lesser quality than might have been actually possible. Predictor JQSP2 also exhibits such a skewness, but to a much lesser degree, and shows a stronger peak around zero, explaining its greater accuracy (as shown in Fig. 1). Lastly, EJQSP, with 20000 prototypes, shows a general behavior comparable to JQSP2, but again with a much stronger peak at zero than either JQSP1 or JQSP2.

If Fig. 3 shows that the individual relative file size predictors behave differently, Fig. 4 shows that for the resulting distributions of quality prediction errors $\hat{q} - q$ (the error of the predicted quality $\hat{q}$ against the observed quality q) are quite similar. For all predictors, the distribution of the errors shows two distinct components: a component similar to a skewed Gaussian (Azzalini,

1985) and a strong peak around zero. While JQSP2 formulates better relative file size prediction than JQSP1, the situation is reversed for quality prediction; a fact that is reflected in Fig. 2, where the overall error on quality prediction of JQSP1 is $\approx 8\%$ smaller than that of JQSP2. This can be explained by the fact that the predictor JQSP2 was designed to avoid overshooting the predicted file size but has no special provision regarding quality prediction. EJQSP, like JQSP1, tends to underestimate resulting quality, however, the central peak and a more compact distribution of errors around that peak show that it yields, overall, better predictions than previous predictors.

5. Discussion

For all algorithms, training and test exemplars must be obtained. In our experiments, it meant subjecting every image from the database obtained by crawling to 100 different transcodings—varying over combinations of $\widetilde{QF}_{out}$ and $\tilde{z}$—to yield a sufficiently large (and dense) pool of transcoding examples. This process is of course very expensive as one has to perform the actual transcoding and assess resulting quality, but the operation can be performed off-line (and incrementally) as trends in image characteristics will vary over time, at application-specific pace.

Training the JQSP1 predictor is $O(n)$ in the number of exemplars, n. Training consists, for each exemplar $x_j \in X$, in quantizing the QF_j, QF_{out}, and z, into $\widetilde{QF}_j$, $\widetilde{QF}_{out}$, and $\tilde{z}$ to index an entry in the array and accumulate the partial sums to compute the centroids. The finalization of the centroid computation is proportional to the number of entries in the array, that is, $O\left(\left|\widetilde{QF}_j\right| \left|\widetilde{QF}_{out}\right| |\tilde{z}|\right)$, where, by abuse of notation, $\left|\widetilde{QF}_j\right|$, $\left|\widetilde{QF}_{out}\right|$, and $|\tilde{z}|$ denote the number of distinct values each can take. The cost of the normalization is therefore negligible compared to the cost of computing the partial sums, since the number of entries in the array will be very small compared to the number of exemplars used for training; furthermore, the array cannot become very large, not only because of memory consumption, but also to avoid the problem of context dilution (Hastie et al., 2009), where it becomes increasingly likely that only a few exemplars (or even none at all) are mapped to a given entry.

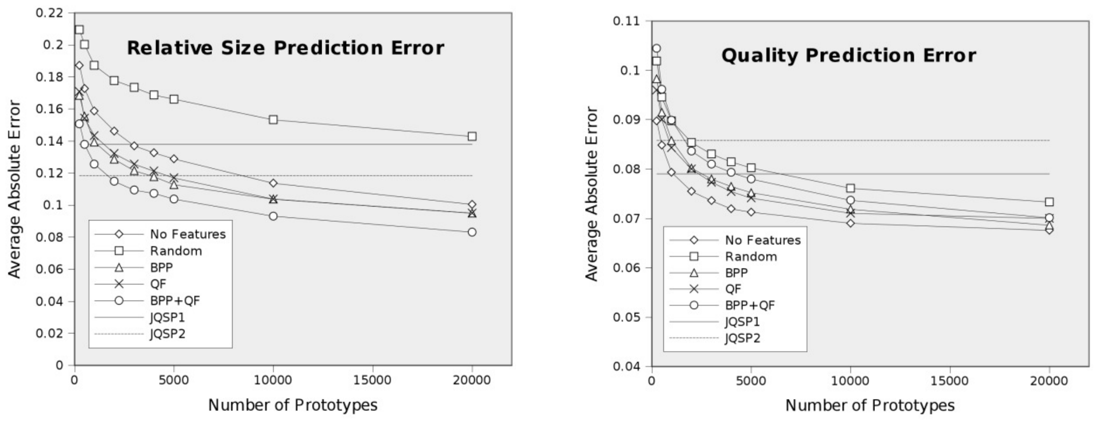
The operating principle of JQSP2 is quite different as it formulates, given an original quality factor QF_j, a scaling z, a prediction on the QF_{out} needed to meet, without exceeding, a target file size (it also predicts resulting file size and prediction as a by-product), and has linear-time complexity for training. JQSP2's design makes it less likely to overshoot significantly on file size (Coulombe & Pigeon, 2010). The training phase constructs the table by going through all the exemplars in the training set and for each exemplar, it updates partial sums indexed by $\widetilde{QF}_j$, $\tilde{z}$, and $\tilde{f}$; which is performed in $O(n)$. The partial sums are then normalized at the cost of $O\left(\left|\widetilde{QF}_j\right||\tilde{z}|\left|\tilde{f}\right|\right)$, which is again negligible compared to the scanning of the training set and the updates of the partial sums.

EJQSP formulates its prediction by clustering, as described in section 2. However, solving eq. (2) exactly is NP-Hard (Aloise et al., 2010; Mahajan et al., 2009; Vattani, 2009), and one has to revert to an approximate algorithm such as the Linde-Buzo-Gray (Linde et al., 1980) or K-means (Lloyd, 1982). An iteration for K-means is $O(d\ m\ n)$ for m prototypes and n exemplars in $\mathbb{R}^d$. Since $O(\lg n)$ iterations seem to be sufficient to bring K-Means to a converging solution, even for moderate m and n, we get an over-all complexity of $O(d\ m\ n\ \ \lg n)$.

Predicting resulting file size and quality or transcoding operations from algorithm JQSP1 and JQSP2 is a constant-time process as it suffices to quantize the appropriate parameters and use the quantized versions to index a table containing the desired prediction. In both cases, we assume that quantizing the parameters is also a constant time operation; or at least, constant in the sense that its complexity does not depend on the number of exemplars used for training and only loosely on the table density. One can think of a quantization that is essentially a "rounding" of values, which certainly can be performed in constant-time. Other, possibly PDF-optimized (Graf & Luschgy, 2000), quantization schemes could be used, and in this case the cost would be at most proportional to the entropy of the quantized values. The clustering method proposed in this work does not require quantization (at least, not explicitly as a preprocessing stage) but was, for the sake of comparability with previous work, trained using the same training exemplars which have quality factors and scalings constrained to a small set of possible values (see section 3). The clustering-based predictor, EJQSP, therefore simply takes the original data, computes the features (in constant time) and searches for the prototype closest to the test exemplar (ignoring quantities f and q, as those we want to predict and are unknown in a prediction-time exemplar); which is essentially nearest neighbor search between one point, the exemplar, and the m prototypes. Although with some preprocessing (Mahajan et al., 2009) or approximate search (Indyk & Motwani, 1998) nearest neighbor search can be made sub-linear, we consider it requires linear time; therefore, if the prototypes lie in $\mathbb{R}^d$ (including features), the search is $O(m\ d)$.

Let us note that while the table size in either algorithm JQSP1 or JQSP2 is limited upwards by the problem of context dilution—where, at some point, there are not enough exemplars to fill all the entries in the table or with sufficiently low variance—algorithm EJQSP degrades gracefully because even if it is presented an exemplar that resembles no other exemplar it has seen during the training phase, it can still formulate a prediction using the closest prototypes. In the worst case, JQSP1 and JQSP2 could fail because the corresponding cell in the array is empty. To avoid this problem, arrays in algorithms JQSP1 and JQSP2 must be kept small enough so that each cell is susceptible to receive a sufficiently large number of exemplars.

The input can be transformed or augmented using features to make information available to the prediction algorithm, information that would be otherwise hard to discover (in a machine-learning sense). Finding good transformations or good features is not, in general, a trivial task. *A priori* knowledge can help us add a very small number of very effective features. In our experiments, we devised two such features. The first feature is the number of bits per pixel of the image, denoted b_j for image I_j, will help distinguish images of the same resolution but of different file sizes. The rationale is that the ratio of the file size to the resolution is indicative of the intrinsic complexity of the image. At equal resolution, an image representing only a featureless blue sky will have a rather small file size, while an image representing a complex natural scene—say a picture taken in a forest—will likely have a much larger file size. They will also behave differently under transformations, and this knowledge will help the predictor formulate more accurate predictions. In the same way, the quality factor difference feature, $QF_{out} - QF_j$, encodes the drop in quality incurred by the transcoding operation, and will help extract information about transcodings that have a similar drop in quality onto similar hyperplanes, thus helping prediction, which is the primary criterion for retaining a feature over another.



a) b)

Figure 5. a) Features' influence on relative file size prediction and b) on quality prediction.

Indeed, features can be heuristic, formulated from *a priori* knowledge, or even random; but in all cases, features are to be evaluated, and added only if they ameliorate prediction. Fig. 5 presents our experiments for feature selection. In these experiments, we compare clustering using different vector lengths. First, we compare with vectors representing only the images' basic features (that is, using only QF_j, w_j, h_j, z, and QF_{out}), then we compare using vectors formed from the basic features plus a uniform random feature (a random variable uniformly drawn on $[0,1]$, different for each vector), then with the basic vectors augmented with b_j, then with the basic vectors augmented by $QF_{out} - QF_j$, and finally with vectors augmented by both b_j and $QF_{out} - QF_j$. In Fig. 5, the experiments are respectively denoted as no features, random, BPP, QF, and BPP+QF. In Fig. 5 a), we compare the effects of the different features (or combinations of features) on the relative file size prediction error. We see that no feature is preferable to adding a random feature (thus showing it is not very useful to select random features) but that successively adding the proposed features helps the clustering achieving better relative file size prediction. Indeed, it is with both added features that prediction error is minimized, at 0.083, using 20000 prototypes. The situation is different when we consider quality prediction, as shown in Fig. 5 b). Again, the random feature is worse; but using no features is slightly better than using both features. However, using a large number of prototypes, the difference in quality prediction error between using both features and no feature is likely not significant (in the order of 0.003) whereas the difference in relative file size prediction is more important (in the order of 0.01). The experiments show that feature selection, especially when predicting more than one quantity, may lead to application-specific trade-offs. In our feature selection, we explicitly favored precision on relative file size prediction in view of applications such as (Pigeon & Coulombe 2011).

The memory usage is also worth discussing. As JQSP1 and JQSP2 training procedures consist in streaming in exemplars one after the other and discarding them after usage, the memory usage is limited to the table needed to store intermediate results. The size of the table is determined by the quantization imposed on the parameters. For JQSP1, for example, the quantization on QF_j, QF_{out}, and z, as used in previous works, yields a $10 \times 10 \times 10$ table with a small number of values stored in each entry (the cumulated file sizes, qualities, and number of exemplars mapping to this entry), which is unlikely to pose problems in a server-type environment. The memory needed by JQSP2 is also determined by the quantization of its input, QF_j, z, and desired file size f_{max}, and so its memory usage can also be quite moderate, even if the target file sizes are rather finely quantized. For EQJSP, the memory usage during training can be made $O(m\ d)$ for m prototypes if one streams the n exemplars from external storage, but one would likely keep all exemplars in memory for faster iterations, yielding $O\big((m+n)d\big)$ storage. During run-time, where only prediction is needed, the storage is brought back to $O(m\ d)$, which is also likely to be essentially negligible in a server-type environment.

Of course, there are trade-offs between storage, algorithmic complexity, and prediction accuracy to consider. Since one would think that storage, especially in a server-type environment, will be negligible even for large m, the main trade-off for EJQSP will be between algorithmic complexity and accuracy. Accuracy plays an important role in applications such as MMS adaptation where the task is not to adapt a single image, but a series of images which are part of the same message (S. Pigeon & Coulombe, 2011, 2012). In this type of optimization problem, errors propagate and do not necessarily cancel out, and it is therefore preferable to favor accuracy over the price of slightly increased complexity (which we will mitigate in Appendix A) so that the final adaptation quality is not jeopardized.

6. Conclusion

Despite formulating predictions in (at most) linear time in the number of prototypes, algorithm EJQSP accuracy outperforms the constant-time JQSP1 and JQSP2 algorithms we presented in previous work. Algorithm EJQSP, using 20000 prototypes, yields significantly better prediction of resulting file size and quality of JPEG images subject to transcoding operations. It yields $\approx 40\%$ smaller prediction error on file size and $\approx 12\%$ on quality than algorithm JQSP1, while yielding

$\approx$ 27% smaller prediction error on file size and $\approx$ 20% on quality than algorithm JQSP2. The new predictor, with its reduced error, can be combined with systems such as those presented in (Coulombe & Pigeon, 2009, 2010; S. Pigeon & Coulombe, 2011, 2012) to yield more efficient and more precise transcoding systems, whether for universal media access, mobile browsing, or multimedia messaging services.

Acknowledgement

This work was funded by Vantrix Corporation and by the Natural Sciences and Engineering Research Council of Canada under the Collaborative Research and Development Program (NSERC-CRD 326637-05).

Appendix A. Efficient implementation of K-Means

Translating **Algorithm 1** directly to a programming language such as C would yield a rather straightforward implementation of K-means, where the program scans the exemplars one by one, finds the nearest prototypes, assigns them to the corresponding partition, then uses the assignment to update the prototypes, repeating the procedure until error ceases to decrease significantly. A careful, but sequential, implementation of this procedure will fail to exploit the inherent parallelism of the algorithm. For example, searching for the nearest prototype can be performed in parallel for every exemplar very efficiently because there are no dependencies between the exemplars: either the values are read-only (the x_j, the $\bar{x}_{t,i}$) or write-only (the a_j), thus dispensing us entirely of the need for mutual exclusion mechanisms. The complexity of one iteration can therefore be reduced from $O(d\;m\;n)$ to $O\left(d\;m\;\frac{n}{c}\right)$, where c is the number of available cores (or hardware threads). Other parts can also be parallelized, such as the update of the prototypes, but not as easily; one would partition the problem by groups of n/c exemplars and hold c series of m partial sums to be combined once all the exemplars are processed.

Parallelism can be obtained by various means, mostly depending on the programming language chosen, in our case C++, such as POSIX Threads (pthreads) (Carver & Tai, 2005), Boost threads (Williams, 2007), but the simplest mean by far is to use OpenMP (Mattson, Sanders, & Massingill, 2005). OpenMP is a compiler extension that reduces the parallelization of a classical C (or C++) program to little more than the addition of a few well-placed compiler-specific `#pragma`s that specify parallel for-loops, memory fences, and reductions.

A parallel implementation can mitigate the complexity of prediction as well. Reducing the (sequential) complexity from $O(m\;d)$ to $O\left(\frac{m}{c}d\right)$ may mean a significant speed-up especially that c can be quite large in modern server-type shared-memory CPUs. Furthermore, as the operation is read-only, there is no need for mutual exclusion, and synchronization is limited to waiting for all sub-problems to terminate and combine (sequentially) the *c* sub-answers into the final answer.

References

Aloise, D., Deshpande, A., Hansen, P., & Popat, P. (2010). NP-Hardness of Euclidean Sum-of-Squares Clustering. *Machine Learning, 75*(2), 245-248.

Azzalini, A. (1985). A Class of Distributions that Includes the Normal Ones. *Scand. J. Stat., 12*, 171-178.

Barni, M. (1997). A fast algorithm for 1-norm vector median filtering. *IEEE Trans Image Process, 6*(10), 1452-1455. doi: 10.1109/83.624972

Blackman, R. B., & Tukey, J. W. (1959). *The Measurement of Power Spectra, from the Point of View of Communications Engineering*: Dover.

Bottou, L., & Bengio, Y. (1995). Convergence Properties of the K-Means Algorithms *Advances in Neural Information Processing Systems* (Vol. 7, pp. 585-592): The MIT Press.

Carver, R. H., & Tai, K.-C. (2005). Modern Multithreading Implementing, Testing, and Debugging Multithreaded Java and C++/Pthreads/Win32 Programs (pp. 480 p. ill.).

Coulombe, S., & Grassel, G. (2004). Multimedia Adaptation for the Multimedia Messaging Service. *IEEE Communication Magazine, 42*(7), 120-126.

Coulombe, S., & Pigeon, S. (2009). *Quality-Aware Selection of Quality Factor and Scaling Parameters in JPEG Image Transcoding*. Paper presented at the Procs. IEEE 2009 Computational Intelligence for Multimedia, Signal, and Video Processing (CIMSVP).

Coulombe, S., & Pigeon, S. (2010). Low-Complexity Transcoding of JPEG Images with Near-Optimal Quality Using a Predictive Quality Factor and Scaling Parameters. *IEEE Trans. Image Processing, 19*(3), 712-721.

Fling, B. (2009). *Mobile Design and Development: Practical Concepts and Techniques for Creating Mobile Sites and Web Apps - Animal Guide*: O'Reilly Media, Inc.

Graf, S., & Luschgy, H. (2000). Lecture notes in mathematics #1730. *Foundations of quantization for probability distributions* (pp. 230 p.). Berlin ; New York: Springer.

Han, R., Bhagwat, P., LaMaire, R., Mummert, T., Perret, V., & Rubas, J. (1998). Dynamic Adaptation in an Image Transcoding Proxy for Mobile Web Browsing. *IEEE Personal Communications Magazine, 5*(6), 8-17.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*: Springer.

IJG. (2012). The Independent JPEG Group. Retrieved March 10, 2012, from http://www.ijg.org/

Indyk, P., & Motwani, R. (1998). *Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality*. Paper presented at the Procs. 30th Annual ACM Symposium on Theory of Computing (STOC'98).

ISO/IEC_CD_10918-5. (2011). Information Technology, Digital Compression and Coding of Continuous-Tone Still Images: JPEG File Interchange Format (JFIF). rev. 2011/04/04.

Lei, Z., & Georganas, N. D. (2002). *Accurate bit allocation and rate control for DCT domain video transcoding*. Paper presented at the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2002).

Linde, Y., Buzo, A., & Gray, R. M. (1980). An Algorithm for Vector Quantizer Design. *IEEE Trans. Comm., 28*(1), 84-95.

Lloyd, S. P. (1982). Least Squares Quantization in PCM. *IEEE Trans. Inf. Theory, 28*(2), 129-137.

Mahajan, M., Nimbhorkar, P., & Varadarajan, K. (2009). The Planar K-Means Problem is NP-Hard *WALCOM: Algorithms and Computation* (pp. 274-285).

Mattson, T. G., Sanders, B. A., & Massingill, B. (2005). *Patterns for parallel programming*. Boston: Addison-Wesley.

Mohan, R., Smith, J. R., & Li, C.-S. (1999). Adapting Multimedia Internet Content for Universal Access. *IEEE Trans. Multimedia, 1*(1), 104-114.

Open Mobile Alliance. (2010). Enabler Test Specification (for Conformance) for MMS Candidate Version 1.3.

Pennebaker, W. B., & Mitchell, J. L. (1993). *JPEG still image data compression standard*. New York: Van Nostrand Reinhold.

Pigeon, S., & Coulombe, S. (2008). *Computationally Efficient Algorithms for Predicting the File Size of JPEG Images Subject to Changes of Quality Factor and Scaling*. Paper presented at the Procs. 24th Queen's University Biennial Symposium on Communications.

Pigeon, S., & Coulombe, S. (2011). *Optimal quality-aware predictor-based adaptation of multimedia messages*. Paper presented at the Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2011 IEEE 6th International Conference on. 10.1109/IDAACS.2011.6072803

Pigeon, S., & Coulombe, S. (2012). Optimal quality-aware predictor-based adaptation of multimedia messages. In R. Duro (Ed.), *Digital Image, Signal and Data processing*: Rivers Pub.

Ratnakar, V., & Ivashin, V. (2001). File Size Bounded JPEG Transcoder, US Patent 6,233,359.

Ridge, J. (2003). Efficient Transform-Domain Size and Resolution Reduction of Images. *Signal Processing: Image Communication, 18*(8), 621-639.

Shu, H., & Chau, L.-P. (2005). *Frame size selection in video downsizing transcoding application.* Paper presented at the Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on.

Vattani, A. (2009). *K-Means Require Exponentially Many Iterations Even in the Plane*. Paper presented at the Procs. 25th Symposium on Computational Geometry.

Wang, Z., Bovick, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Trans. Image Processing, 13*(4), 600-612.

Williams, A. (2007). Boost Threads. Retrieved March 10, 2012, from http://www.boost.org/doc/libs/1_48_0/doc/html/thread.html