Block selective redaction for minimizing loss during de-identification of burned in text in irreversibly compressed JPEG medical images

David A. Clunie
Dan Gebow

Block selective redaction for minimizing loss during de-identification of burned in text in irreversibly compressed JPEG medical images

David A. Clunie[a,*] **and Dan Gebow**[b]
[a]PixelMed, 943 Heiden Road, Bangor, Pennsylvania 18013, United States
[b]MDDX Research and Informatics, 580 California Street, Fl 16, San Francisco, California 94104, United States

Abstract. Deidentification of medical images requires attention to both header information as well as the pixel data itself, in which burned-in text may be present. If the pixel data to be deidentified is stored in a compressed form, traditionally it is decompressed, identifying text is redacted, and if necessary, pixel data are recompressed. Decompression without recompression may result in images of excessive or intractable size. Recompression with an irreversible scheme is undesirable because it may cause additional loss in the diagnostically relevant regions of the images. The irreversible (lossy) JPEG compression scheme works on small blocks of the image independently, hence, redaction can selectively be confined only to those blocks containing identifying text, leaving all other blocks unchanged. An open source implementation of selective redaction and a demonstration of its applicability to multiframe color ultrasound images is described. The process can be applied either to standalone JPEG images or JPEG bit streams encapsulated in other formats, which in the case of medical images, is usually DICOM. [DOI: 10.1117/1.JMI.2.1.016501]

Keywords: deidentification; redaction; anonymization; lossy compression; JPEG; DICOM.

Paper 14164R received Dec. 17, 2014; accepted for publication Mar. 3, 2015; published online Mar. 25, 2015.

1 Introduction

Clinical care of patients requires that every piece of information, including images, be accurately identified. Other applications require that Individually Identifiable Information (III; Information that identifies an individual, or for which there is a reasonable basis to believe the information can be used to identify an individual. Derived from a more specific definition of individually identifiable health information.[1]) can be completely removed. Such applications include research, clinical trials, public distribution for secondary reuse, and preparation and sharing of teaching materials.

In the case of medical images stored or transmitted as files or messages, information about the images (metadata) is usually encoded in a "header." The DICOM standard, for example, defines many attributes that describe the patient, study and characteristics of the acquisition.[2] Standards exist that enumerate those attributes that need to be removed for deidentification of various scenarios.[3]

Usually, the pixel data of a DICOM file do not contain identifying information. If the image is captured as a screen shot or digitized video of a display that contained identifying information or a scanned document or printed film, the pixel data may contain burned-in text. (Pixels in an image that contain text, particularly text that might identify the patient and the date the image was acquired.) Such text needs to be removed (blacked out or "redacted") before the image can be considered to have been adequately deidentified.

*Address all correspondence to: David A. Clunie, E-mail: dclunie@dclunie.com

If the pixel data are not compressed, redaction is a simple matter of replacing the individual pixel values in the offending regions. If the pixel data are stored in a compressed form, traditionally it is decompressed, redacted, and if necessary, recompressed. Decompression without recompression may result in images of excessive or intractable size. Recompression with an irreversible scheme may cause additional loss in the diagnostically relevant regions of the images, hence it may be undesirable.

The European Society of Radiology warns about the dangers of repeated cycles of irreversible compression.[4] The effects of repeated JPEG decompression and recompression can be demonstrated visually.[5]

An echocardiogram (ultrasound image of the beating heart) is one type of image that typically manifests both burned-in identifying information and irreversible compression. The presence of the burned-in text is a historical artifact of the early use of video capture for this modality. Modest irreversible compression is used to keep file sizes reasonable while preserving sufficient image quality for the intended use.[6,7] These images are most commonly encoded as multiframe grayscale or color 8 bits/channel baseline JPEG compressed bit streams encapsulated in DICOM.

2 JPEG (and DICOM Encapsulation) Principles

The JPEG standard[8] describes multiple processes for compression, but the most commonly used method is the sequential discrete cosine transform (DCT) Huffman-encoded "baseline" lossy method. It supports 8 bits/channel and is usually used with either one (grayscale) or three (color) components. This process is used for JPEG files that are displayed in web browsers as well as those produced by consumer digital cameras. It is the process used in DICOM for most 8-bit images and is formally referred to as the "JPEG Baseline (Process 1): Default Transfer Syntax for Lossy JPEG 8 Bit Image Compression" (assigned the unique identifier, "1.2.840.10008.1.2.4.50").[9]

The JPEG process operates on each frame of a multiframe image independently, unlike MPEG, which takes advantage of interframe redundancy. In DICOM, each frame is encoded in one or more "fragments." Frames do not span fragment boundaries. Fragments are encoded between DICOM Sequence Item and Sequence Item Delimiter tags. DICOM toolkits provide the support necessary to encapsulate and unencapsulate individual JPEG frames for use with conventional JPEG codecs.

In brief, the JPEG baseline process works by transforming the RGB colors into YCbCr (a form of chrominance down-sampling that takes advantage of the human visual system's lower acuity for color differences than for luminance. Also referred to as chroma subsampling.[10]) color space, down-sampling the color components, splitting the frame into 8×8 blocks, transforming the pixel values in each block into frequency domain coefficients (with a DCT), quantizing these coefficients, and then entropy coding them to take advantage of redundancies exposed by the earlier steps. This process is illustrated in Fig. 1.

Loss may be incurred at several of these steps. Though there is some loss during the RGB to YCbCr transformation, and further loss in the color down sampling, the majority of the loss takes place in the quantization step. The quantization step is user controllable (e.g., by a so-called "quality factor" that affects selection or population of the quantization tables). The production of smaller (shorter), possibly zero, coefficients in the quantization step is the primary means of reduction in size since the following entropy coding step encodes runs of zeroes more compactly and replaces frequently occurring patterns with shorter symbols (fewer bits).

In more detail, this JPEG process works by:

1. Dividing a multiframe image into one or more frames and encoding each separately.
2. Dividing each frame into separate R, G, and B components and transforming the color space into Y, Cb, and Cr components to take greater advantage of the redundancy between channels (not necessary for grayscale frames).
3. Downsampling the color (Cb and Cr) components, usually by a factor of 2 in the horizontal and/or vertical directions (so called 4∶2∶2 or 4∶2∶0 sampling).
4. Dividing each (possibly down-sampled) component into 8×8 blocks of pixels and performing a DCT on the values in that block to produce an 8×8 matrix of frequency domain coefficients, the top left of which is the lowest (zero) frequency (DC coefficient) and the remainder of which are of increasing frequency (AC coefficients).
5. Quantizing the DC and AC coefficients in each 8×8 block.
6. Interleaving the neighboring 8×8 blocks for all components into minimum coded units (MCUs), such that a single MCU covers the same minimum spatial region accounting for downsampling, e.g., for a grayscale frame an MCU will be one 8×8 block; for a color image with Cb and Cr components downsampled by 2 in both directions, an MCU will include four 8×8 Y blocks and one Cb and one Cr block (see Fig. 2).
7. Defining a fixed number of MCUs that will be encoded in each entropy coded segment (ECS) in a single restart interval (as a means of error resiliency). There is often only a single ECS for the entire frame.
8. For all the MCUs in a single ECS, encoding each block of each MCU in sequential order.
9. For each block, encoding the difference between the DC coefficient of the current block and that of the previous block for the same component using a Huffman code to represent the number of bits to encode followed by that number of bits. Then encoding the AC coefficients of the block scanned in zigzag order using a Huffman code to represent both the number of zeroes and number of bits to encode followed by that number of bits, with a special end of block (EOB) code to indicate all remaining coefficients are zero.
10. Emitting marker segments of fixed or defined variable length to describe the necessary parameters using start of image (SOI), start of frame (SOF), define quantization table (DQT), define Huffman table (DHT), and define restart interval (DRI), with each marker being flagged by a leading 0xFF byte.
11. Emitting the bit stream of each ECS successively [separated by restart interval (RSTn) markers, if used], with each 0xFF byte that happens to occur in the bit stream followed by a "stuffed" zero byte to signal that it is not a marker.

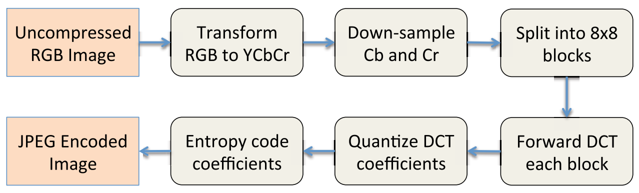

Fig. 1 JPEG baseline sequential encoding process.

The decoding process is the reverse of the encoding process.

3 Redaction

3.1 *Conventional Redaction*

If the pixels in a JPEG image contain identifying information in the form of burned-in text that needs to be removed to protect privacy, the naive approach is to decompress the image, redact a region encompassing the offending pixels (e.g., replace with black or background values), and possibly recompress the image. More generally, regions other than burned-in text can also be redacted (e.g., recognizable facial features, if not

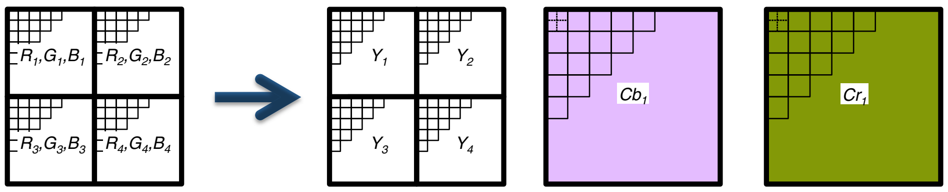

Fig. 2 Chrominance down-sampling by a factor of two in both horizontal and vertical directions after color space transformation to produce one MCU consisting of four luminance, one Cb and one Cr 8×8 block for four original 8×8 RGB blocks.

a significant proportion of the entire image and not needed for the diagnostic task or other intended use).

If recompression is not performed after decompression and redaction, then the loss will be no greater in this process than any other use of the image that requires decompressing (such as displaying it). However, the decompressed image may be considerably larger than the original compressed image, and in some cases, may be of impractical size.

Recompression of JPEG images entails further loss than was present in the original compressed image, even when using the same codec, quantization tables, or without quantization (100% “quality factor”) and without chrominance downsampling, since several of the steps in the process involve rounding errors (specifically, the YCbCr transformation and the forward and inverse DCT operations).

3.2 *Block Selective Redaction*

Since only a limited number of pixel regions need to be redacted to remove the identifying information and the JPEG process divides the image into MCUs of small numbers of 8×8 blocks, it is obvious that only those blocks affected need to be modified and all other blocks could be left alone. This approach completely avoids any loss at all in regions that do not intersect the redacted regions.

A new block can then replace every original block that is entirely included in a redacted region.

There are various choices of values to use for replacement blocks. Since it is sufficient to replace the block with a solid color, all the AC coefficients can be set to zero. The choice of DC coefficient value is complicated by the fact that the DC coefficient is not itself encoded, but rather the difference from the DC coefficient of the preceding block is encoded. This means that replacing every block with “all zero” AC coefficients, but leaving the DC difference, though expedient, does not produce a “black” block, but rather a block that is entirely one shade of gray or color, that of the average value of the original block. This approach provides for the most compact replacement with the least requirement for preceding context, and the least impact on succeeding blocks.

Alternatively, the DC difference value could be replaced with the value necessary to result in an actual DC coefficient of zero (black), quantized appropriately (considering the quantization tables and the component bit depth), and then the effect of that change propagated to the immediately succeeding block (once a run of successive redacted blocks had finished). This additional complexity has not been found necessary for the current implementation.

Blocks that overlap but are not entirely enclosed by the redaction regions introduce another complication. Theoretically, it would be possible to fully decompress each of those blocks, redact only the affected pixels in the spatial domain, then recompress that block. This would result in some loss in the unredacted pixels of that block but without discarding the information completely. The process would affect the DC coefficient prediction value; therefore, the delta would need to be propagated to all subsequent blocks as well. For the current implementation, it has proven sufficient to replace only entire blocks; in effect, each redacted region is expanded to 8×8 luminance (Y) component boundaries (and 8×16, 16×8 or 16×16 chrominance (Cb, Cr) component boundaries depending on the horizontal and vertical chrominance downsampling in the original, if any). Note that even if only a subset of luminance blocks in an MCU overlap with the redaction region, the nonoverlapping luminance blocks in the MCU are still affected, since the corresponding downsampled chrominance blocks will be reduced to DC coefficients.

4 Implementation

A conventional JPEG “codec” (coder/decoder) used to compress or decompress an image does not normally provide a mechanism to access the individual compressed MCUs and blocks in the manner necessary to implement block selective processing. Frames are usually compressed or decompressed in their entirety.

The complexity of processing a compressed JPEG bit stream to identify and extract each MCU and block is modest, particularly if any changes to the bit stream necessary for redaction can be confined strictly to the block level without requiring prior context from previous blocks or propagation of changes to successive blocks.

Accordingly, a new JPEG parser was written in Pure Java (based on a translation of an earlier C++ JPEG dumping tool by one of the authors[11]). The JPEG parser reads and deciphers the parameters of each JPEG marker segment and extracts the bytes comprising each ECS. The parser was configured to also copy the marker segments to the output stream. When the end of each ECS is reached, the buffered bytes are decoded by processing each successive MCU, each component within that MCU, and each block for that component. Since the blocks are present in the bit stream in successive order with no delimiters, it is necessary to decode the bit stream to extract the Huffman codes for the DC and (up to 63) AC coefficients and further extract the number of bits specified by those codes, count the number of coefficients processed (accounting for the specified number of zeroes in a run) and recognize when a block has been terminated by the special EOB code. The decoded values are then discarded since they were only decoded in order to maintain synchronization with the bit stream; only the corresponding original encoded bit values are copied to the output.

Prior to deciphering each block, its location is computed relative to the prespecified regions to be redacted. If the block intersects a redaction region, it is entirely replaced by an appropriately encoded block that is the decoded DC coefficient difference value (unchanged), followed by an AC coefficient EOB code (meaning all remaining values in the block are zero). If the block does not intersect a redaction region, then the original encoded bit values are copied to the output unchanged as they are being decoded.

The net result is an output bit stream that is identical to the original except that all blocks that intersect with a redaction region are replaced by a block whose pixels all have the value that is the same as the average value of the preceding block (whether redacted or not).

5 Results

Visually, the effect on the text is shown in Fig. 3, which consists of close up views of an original image containing text, a traditionally redacted image with the selected redaction region replaced by black, and an image to which selective block redaction has been applied. This example illustrates the effect of retaining the original DC coefficient difference value rather than attempting to replace each redacted block with a specific DC value, as well as how the redaction region is expanded to a block boundary.

Figure 4 shows entire images illustrating the effect of decompression and recompression on the pixel values of the nonredacted areas, compared with the application of selective block redaction process which has no effect at all on pixels outside the redacted areas. Note that both difference images are presented both without windowing as well as windowed with a very narrow window (center 4 width 11) to highlight any differences in the areas of potential diagnostic significance.

6 Discussion

Before writing a new implementation, the process described was prototyped using the jpegtran utility from the Independent JPEG Group (IJG)'s libjpeg tools written in C,[12] as well as a commercially available consumer JPEG editing tool.[13]

Tom Lane and Guido Vollbeding provided, in jpegtran, a mechanism to first losslessly transcode JPEG images between processes such as baseline to progressive (version 6, 2-Aug-95), then to losslessly rotate JPEG images without fully decompressing them (version 6b, 27-Mar-1998). The jpegtran tool was extended to also allow lossless cropping (by Ben Jackson, later included in version 7, 27-Jun-2009), resizing (version 8, 10-Jan-2010), pasting of other JPEG images in place (Guido Vollbeding's "droppatch," described in his 21 Jan 2000 post)[14] and most recently, a specific "wipe" option to support redaction (in v9a, 19-Jan-2014).

Preliminary experiments with manual extraction of JPEG baseline process bit streams from DICOM multiframe echocardiograms, followed by the use of jpegtran with droppatch, repeated with BetterJPEG, confirmed that the approach was viable.

Andrew Senior developed a C++ JPEG Redaction Library[15] for use with the SecureSmartCam project intended for redaction of detected faces; see also Chattopadhyay for related discussion.[16] Invertible encryption of redacted information has also been described, in which facial regions are recognized in JPEG images and their blocks selectively replaced.[17]

The earliest description that could be found of manipulation of JPEG blocks in the transformed (frequency) domain for editing purposes is the "subtitling" operation described by Smith.[18] Miller describes an early approach to manipulation of JPEG blocks without fully decompressing them in a patent.[19] That patent claims a specific process that entails maintaining pointers to original and edited blocks and is intended for random access. Miller also specifically addresses the need to handle the differential DC coefficient using an "edge table." Our implementation neither uses a pointer array nor an edge table nor changes the DC coefficients as claimed in Miller. Later patents related to lossless manipulation of JPEG blocks focus primarily on rotation and other geometric transformations. A survey of the prior art is provided in one such patent on rotation.[20] There exists a considerably body of related work on image manipulation in the transformed domain.[21–29] Selective use of blocks in the transformed domain for other purposes, such as searching, indexing, or selective region decompression has also been described.[30–33]

Pure Java frameworks are popular for deidentification, whether they use an automated bulk process controlled by templates, such as RSNAs Clinical Trial Processor (CTP),[34] or an interactive user-controlled process, with a tool such as DicomCleaner.[35] Accordingly, it was necessary to implement a Pure Java block selective redaction process rather than to modify existing platform-specific code.

The implementation described here would seem to be the first published that applies the well known process of selective JPEG block processing for the purpose of deidentification of DICOM encapsulated JPEG images, such as is required for clinical trials and public reuse of medical images. The software also uses the traditional DICOM "header" deidentification mechanisms already present in the PixelMed toolkit,[36] of which it is an extension. The JPEG bit stream processing is factored out as a separate package,[37] allowing it to be used with other Java DICOM toolkits, such as dcm4che,[38] which is used in CTP.

The current implementation has several limitations. Entire blocks in redacted regions are processed rather than attempting to decompress and redact only subparts of blocks, a theoretical possibility described earlier. In practice, this has been sufficient, since the identifying information is usually located far enough from the diagnostic information in the image for this not to matter. Further, partial block redaction might render the block vulnerable to recovery methods such as those described by Zhong-Yang Ho.[39] Using the DC coefficient difference value unchanged results in a patchwork of color blocks rather than a single value (such as black) for the redacted blocks. Conceivably, if the offending text was rendered in a very large font relative to the 8×8 block size, then the redaction might fail to obscure the text, but this has not proven to be an issue in practice, since the

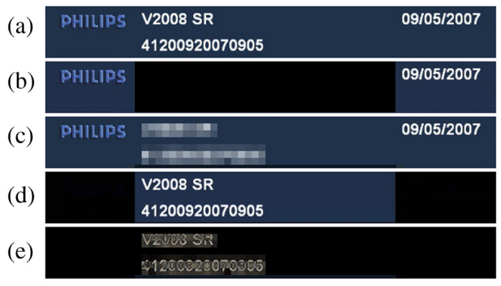

Fig. 3 Closeup views of original and redacted text. (a) Original. (b) Decompressed, redacted, recompressed (with identity quantization tables). (c) Selective block redaction. (d) Difference original versus recompressed. (e) Difference original versus selective block redaction.

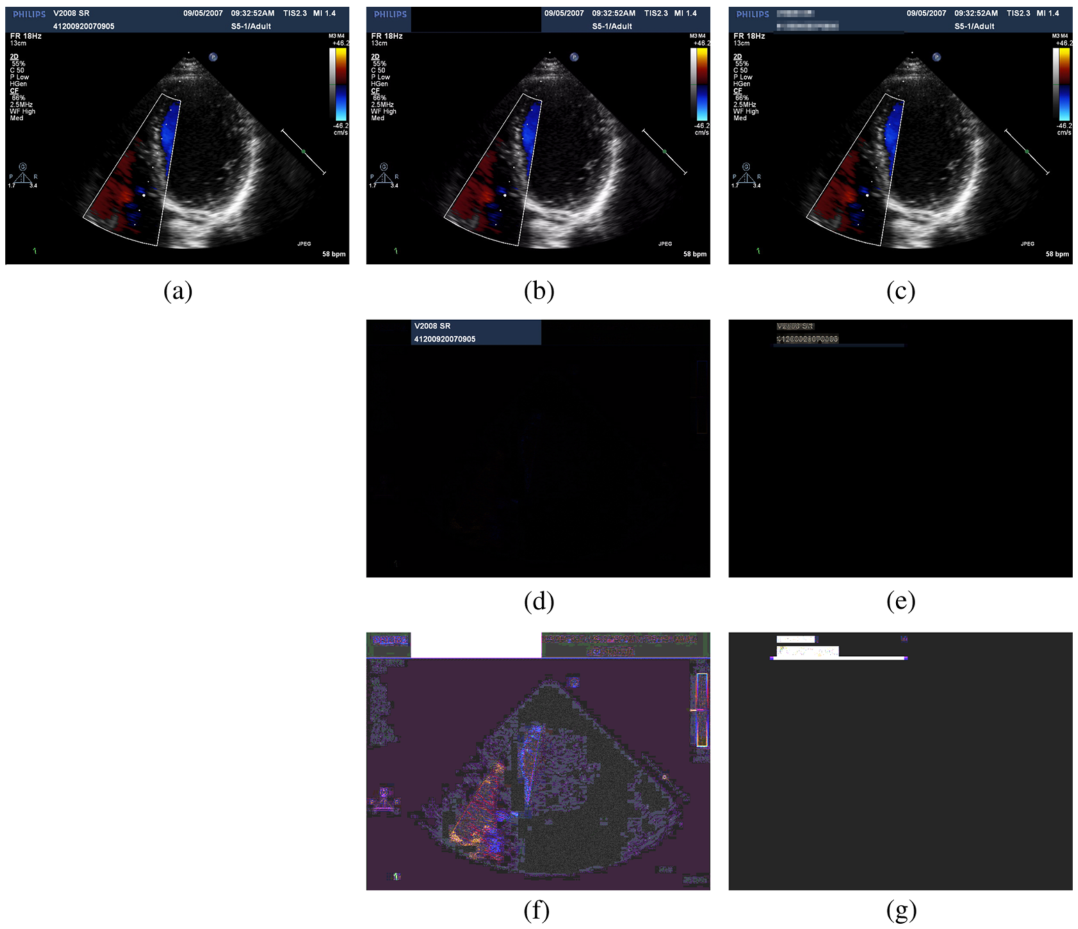

Fig. 4 Entire images with original and redacted text. (a) Original. (b) Decompressed, redacted, recompressed (with identity quantization tables). (c) Selective block redaction. (d) Difference original versus recompressed. (e) Difference original versus selective block redaction. (f) Narrowly windowed (4/11) difference original v. recompressed. (g) Narrowly windowed (4/11) difference original versus selective block redaction.

font strokes are usually of the order of a single pixel in width. If redaction of areas other than text is required, such as of facial features, particularly in high-resolution images, the decision not to change the DC coefficient to a single value may need to be revisited. Only the 8-bit per component JPEG process is currently implemented since it is the most common format in which images with burned-in text are encountered. JPEG also defines an extended, 12-bit, process, which is sometimes used for compressing CT and MR and X-ray images, but in general, these do not contain burned-in text; extension of the implementation to support greater component precision would be very straightforward, particularly since only the entropy coded is affected and a higher precision DCT is not needed.

Acknowledgments

MDDX Research and Informatics funded this project. The code is released under a BSD-style Free and Open Source Software (FOSS) license. Andrew Senior and Guido Vollbeding provided helpful feedback on their implementations.

References

1. United States Government, "Social Security Act," 42 U.S.C. 1320d(6)) section 1171(6), http://www.ssa.gov/OP_Home/ssact/title11/1171.htm (31 January 2015).
2. DICOM Standards Committee, *Digital Imaging, and Communications in Medicine (DICOM)—PS3.3 Information Object Definitions*, National Electrical Manufacturers Association, Rosslyn, Virginia, http://dicom.nema.org/medical/dicom/current/output/chtml/part03/PS3.3.html (2014).
3. DICOM Standards Committee, *Digital Imaging, and Communications in Medicine (DICOM)—PS3.15 Security, and System Management Profiles*, National Electrical Manufacturers Association, Rosslyn, Virginia, http://dicom.nema.org/medical/dicom/current/output/chtml/part15/PS3.15.html (2014).
4. European Society of Radiology, "Usability of irreversible image compression in radiological imaging. A position paper by the European Society of Radiology (ESR)," *Insights into Imaging* **2**(2), 103–115 (2011).
5. M. Brown, "JPEG iterations: effects of repeated JPEG compression," (2013), http://www.nezumi.demon.co.uk/photo/jpeg/2/jpeg2.htm (10 December 2014).
6. T. H. Karson et al., "JPEG compression of digital echocardiographic images: impact on image quality," *J. Am. Soc. Echocardiogr.* **8**(3), 306–318 (1995).
7. T. H. Karson et al., "Digital storage of echocardiograms offers superior image quality to analog storage, even with 20:1 digital compression: results of the digital echo record access study," *J. Am. Soc. Echocardiogr.* **9**(6), 769–778 (1996).
8. ISO/IEC 10918-1/ITU T.81, "JPEG standard for digital compression, and encoding of continuous-tone still images. Part 1—requirements and implementation guidelines," (1984), http://www.w3.org/Graphics/JPEG/itu-t81.pdf (10 December 2014).
9. DICOM Standards Committee, *Digital Imaging, and Communications in Medicine (DICOM)—PS3.5 Data Structures, and Encoding*,

National Electrical Manufacturers Association, Rosslyn, Virginia, http://dicom.nema.org/medical/dicom/current/output/chtml/part05/PS3.5.html (2014).
10. Wikipedia Contributors, "Chroma subsampling," *Wikipedia, The Free Encyclopedia* (2015), http://en.wikipedia.org/wiki/Chroma_subsampling (31 January 2015).
11. D. Clunie, "JPEG dump, a utility in Dicom3tools," (2014), http://www.dclunie.com/dicom3tools.html (11 December 2014).
12. Independent JPEG Group, "LIBJPEG: free library for JPEG image compression," (2014), http://www.ijg.org/ (10 December 2014).
13. Better JPEG Team, "Edit JPEG photos without losing quality," (2014), http://www.betterjpeg.com/ (10 December 2014).
14. JPEG Club, "JPEGTRAN," (2012), http://jpegclub.org/jpegtran/ (10 December 2014).
15. "Senior A. JPEG redaction library," (2014), https://github.com/asenior/Jpeg-Redaction-Library (10 December 2014).
16. A. Chattopadhyay and T. E. Boult, "PrivacyCam: a privacy preserving camera using uCLinux on the blackfin DSP," in *Computer Vision and Pattern Recognition (2007)*, Minneapolis, MN, IEEE Computer Society, Los Alamitos, CA (2007).
17. T. E. Boult, "PICO: privacy through invertible cryptographic obscuration," in *Computer Vision for Interactive and Intelligent Environment*, Lexington, KY, pp. 27–38, IEEE Computer Society, Los Alamitos, CA (2005).
18. B. C. Smith and L. A. Rowe, "Algorithms for manipulating compressed images," *IEEE Comput. Graph. Appl.* **13**(5), 34–42 (1993).
19. R. F. Miller and S. M. Blonstein, "Compressed image virtual editing system," U.S. Patent No. 5,327,248 (1994).
20. F. A. Micco and M. E. Banton, "Method and apparatus for image rotation with reduced memory using JPEG compression," U.S. Patent No. 5,751,865 (1998).
21. R. L. de Queiroz, "Method and apparatus for processing of a JPEG compressed image," U.S. Patent No. 5,867,598 (1999).
22. R. L. de Queiroz, "Fast decompression and rotation for devices with asymmetric resolution," U.S. Patent No. 6,175,653 (2001).
23. V. Ratnakar, V. Ivashin, and V. Bhaskaran, "Image transformations in the compressed domain," U.S. Patent No. 6,298,166 (2001).
24. R. V. Klassen, "Rotated read-out of compressed images," U.S. Patent No. 6,353,682 (2002).
25. M. Boliek, E. L. Schwartz, and M. J. Gormish, "Method and apparatus for editing an image while maintaining codestream size," U.S. Patent No. 7,079,690 (2006).
26. S. Boler et al., "Lossless manipulation of media objects," U.S. Patent No. 7,142,225 (2006).
27. K. Srinidhi, "Method and apparatus for processing a compressed image in an order other than the order of which it was compressed," U.S. Patent No. 7,856,147 (2010).
28. S.-F. Chang, "New algorithms for processing images in the transform compressed domain," *Proc. SPIE* **95**, 445–454 (1995).
29. S.-F. Chang, "Compressed-domain techniques for image/video indexing and manipulation," in *IEEE Int. Conf. Image Processing (ICIP95), Special Session on Digital Library and Video on Demand*, Washington, DC, IEEE Computer Society, Los Alamitos, CA (1995).
30. C.-W. Ngo, T.-C. Pong, and R. T. Chin, "Exploiting image indexing techniques in DCT domain," *Pattern Recognit.* **34**(9), 1841–1851 (2001).
31. F. Arman, A. Hsu, and M.-Y. Chiu, "Image processing on compressed data for large video databases," in *Proc. First ACM Int. Conf. on Multimedia*, pp. 267–272, ACM, Anaheim, California (1993).
32. B. C. Smith, "A survey of compressed domain processing techniques," (1995), http://www.uky.edu/~kiernan/DL/bsmith.html (10 December 2014).
33. B. C. Smith and L. A. Rowe, "Compressed domain processing of JPEG-encoded images," *Real Time Imaging* **2**(1), 3–17 (1996).
34. RSNA, "CTP—the RSNA clinical trial processor," (2014), http://mircwiki.rsna.org/index.php?title=CTP-The_RSNA_Clinical_Trial_Processor (10 December 2014).
35. D. Clunie, "DicomCleaner," (2014), http://www.dclunie.com/pixelmed/software/webstart/DicomCleanerUsage.html (10 December 2014).
36. D. Clunie, "PixelMed Java DICOM Toolkit," (2014), http://www.pixelmed.com/index.html#PixelMedJavaDICOMToolkit (10 December 2014).
37. D. Clunie, "PixelMed Java DICOM Toolkit," (2014), http://www.pixelmed.com/index.html#PixelMedJavaJPEGSelectiveBlockRedactionCodec (10 December 2014).
38. G. Zeilinger et al., "DCM4CHE, a DICOM implementation in JAVA," (2014), http://www.dcm4che.org/ (10 December 2014).
39. N. Z.-Y. Ho, "Residual information of redacted images hidden in the compression artifacts," *Lect. Notes Comput. Sci.* **5284**, 87–101 (2008).

David A. Clunie is a radiologist and medical informatics consultant. He received his MBBS from the University of Melbourne in 1983. He is the editor of the DICOM standard. His current research interests include deidentification and quantitative result encoding for clinical and preclinical research. He is a member of SPIE.

Dan Gebow is a scientist with a specialty in developing biomedical research applications. He received his doctorate from Harvard University. He is currently the CEO of MDDX Research and Informatics, a San Francisco-based imaging research organization. His current interest is focused on creating long-term DICOM repositories and part of an image-centric clinical trial management system.