# Storing Metadata as QR Codes in Multimedia Streams

Athanasios Zigomitros[1] and Constantinos Patsakis[2]

[1] Department of Informatics, University of Piraeus, Piraeus, Greece & Institute for the Management of Information Systems, "Athena" Research Center, Greece
azigomit@unipi.gr

[2] Distributed Systems Group, School of Computer Science and Statistics, Trinity College, College Green, Dublin 2, Ireland
patsakik@scss.tcd.ie

**Abstract.** With the continuous adoption of the web and the increase of connection speeds, people are more and more sharing multimedia content. The main problem that is created by this approach is that the shared content become less and less search-friendly. The information that is shared, cannot be easily queried, so a big part of the web becomes inaccessible. To this end, there is a big shift towards adopting new metadata standards for image and video that can efficiently help with queries over image and videos.

In this work we extend our proposed method of embedding metadata as QR codes in gray scale images, to color video files with a slightly modified algorithm to make the decoding faster. We then examine the experimental results regarding the compressed file size, using a lossless encoding and the distortion of the frames of the video files. Storing the metadata inside the multimedia stream with QR format has several advantages and possible new uses that are going to be discussed.

**Keywords:** QR codes, video metadata, LSB.

## 1 Introduction

Human nature makes us more attracted to image stimuli. Therefore, a huge economic sector is focused on capitalizing image processing. Apart from the press, which uses still images and plays a significant part in the news, advertisement and other areas, large amounts of money are invested in video. The biggest part of the entertainment industry is currently focused on video, whether this is cinema films, or TV shows. However, the displayed information has different target audience, using different mediums each time to consume it. Hence, there is a plethora of video file formats available trying to cover the needs of each use-case. Nevertheless, video is used in many other aspects of our everyday living. Typical examples are surveillance or traffic cameras, teleconferences etc.

It is quite clear that since a video is a set of images, the amount of stored information is quite big, which introduces a great difficulty in processing it.

But the size of videos is not the actual problem in managing the stored information. One of the biggest challenges in video is data mining, extracting useful information or just specific instances. Contrary to text data, where we usually have efficient methods to search for the occurrences of a string or of a pattern, searching in videos is far more demanding and unfortunately with significantly smaller success rate. Brightness, luminosity, angle of recording, movement, closure, lens distortions are few of the factors that have to be taken into consideration when processing video streams, creating great demands in processing and computing.

Due to the processing cost in terms of resources and time, the collected and processed data must be stored in a convenient format which allows further exploitation. Several approaches have been proposed in the literature [1,2,3], in most cases an external database is used where all this information is stored, as in the case of MPEG-7 [4]. Yet, in other cases, this information is stored inside the file as comments or additional tags. According to the AVI file format specification, the metadata are stored at the start of AVI files. The video filetype MP4 is based on the ISO base media file format known as ISO/IEC 14496-12 (MPEG-4 Part 12) which allows metadata to be stored anywhere in the file, but practically they are stored either at the beginning or at the end of the file, as the raw captured audio/video data is saved continuously. Apparently, in both cases this information can easily be removed or lost over file format changes or editing.

The current article extends the work of [5] and proposes metadata embedding in videos as QR codes in the LSB of each frame. The experimental results illustrate that the use of the proposed method introduces a minimal image distortion, while enabling tracing in some cases of distortion on each frame. Furthermore, it introduces many other new possibilities for novel applications.

## 2 Related Work

### 2.1 Storing Metadata

When the applications are based on portability, metadata are embedded in the file, while the approach of a database is used, when the need is for easy reuse, enabling intensive queries on video content, through a central multimedia management system. Compared to the use of databases, embedding metadata in the video files has several additional advantages. It protects the file against loss or unavailability of the central database, while providing a search-engine friendly interface, as the embedded metadata can also be extracted by search engines, making them multimedia aware. This way, they may catalog the multimedia content that is offered by streaming for example sites. Finally, the metadata is not lost when a user downloads and keeps a local copy of the file to their computer.

To allow interoperability several metadata standards have been recommended, with the most well known being, the Qualified Dublin Core (DC), Public Broadcasting Core (PB Core), Extensible Metadata Platform (XMP) and Moving Picture Experts Group 7 (MPEG7).

Dublic Core[1] is a simple standard for describing digital files, using a set of 15 elements. The main goal is the quick search and retrieval of information in digital libraries. The Qualified Dublin Core has 3 additional elements, enables users to add their own qualifiers for the semantic description of the media resulting fine-tunning search results. Metadata in Dublin Core are often stored as name-value pairs within META tags as follows:

```
<META NAME="DC.Title" CONTENT="Metadata - Dublin Core">
<META NAME="DC.Creator.Address" CONTENT="iris@jarmin.com">
<META NAME="DC.Subject" CONTENT="metadata, metatags, Dublin Core,
guidelines, web design, resources, HTML authoring">
<META NAME="DC.Description" CONTENT="A quick guide to Dublin Core
metadata for web designers.">
<META NAME="DC.Date.Created" CONTENT="2000-02-01">
<META NAME="DC.Date.Modified" CONTENT="2000-02-09">
<META NAME="DC.Type" CONTENT="Text.Homepage.Educational">
<META NAME="DC.Format" CONTENT="text/html">
<META NAME="DC.Language" CONTENT="en">
<META NAME="DC.Identifier"
CONTENT="http://www.jarmin.com/meta/dcore.html">
```

The Dublin Core has formally been embedded in ISO 15836, ANSI/NISO Z39.85-2007 and IETF RFC 5013.

**Table 1.** Dublin Core Elements

| | | |
|---|---|---|
| Title | Date | Relation |
| Creator | Type | Coverage |
| Subject | Format | Rights |
| Description | Identifier | Audience* |
| Publisher | Source | Provenance* |
| Contributor | Language | RightsHolder* |

**Note:** The elements with * are the Qualified Dublin Core additional elements.

Public Broadcasting Core (PB Core)[2] were published in 2005 as a metadata standard for multimedia. This standard has been developed by the public broadcasting community and has been funded by the Corporation for Public Broadcasting to serve the U.S. public broadcasting community. PBCore is mainly based on Dublin Core, introducing several additional elements useful for media. It has been adopted by many users in public media, as well as film archives, academic institutions, and other multimedia collections and archives.

MPEG-7 is also an ISO standard, namely ISO/IEC 15938, but unlike the preceding MPEG standards (MPEG-1,MPEG-2,MPEG-4) which deal with the

[1] http://www.dublincore.org/documents/dces

[2] http://www.pbcore.org

coded representation of the multimedia content, the purpose of MPEG-7 is the multimedia metadata structures, used to annotate and describe multimedia content using the XML. MPEG-7 is designed to allow fast and efficient searches on multimedia, based on their content. The MPEG-7 metadata may be physically located with the multimedia files, in the same data stream or on the same storage system as an external file, however the descriptions could also be stored in a remote database.

Adobe has introduced the non-proprietary Extensible Metadata Platform (XMP), which became an ISO standard for metadata, namely ISO 16684-1. XMP is based on Resource Description Framework (RDF) and can be considered as a subclass specific for multimedia files and currently, it is being supported by many vendors. In XMP-enabled applications, information about a projects can be captured during the content-creation process and the embedding is made within the file or into a multimedia content-management system, with the first case being the most widely used. However, since many file formats might not support the XMP standard, the metadata is stored in a separate file. Embedded XMP metadata have been reported by Adobe to remain even if the file is converted to a different format.

One significant problem of all these standards is that by uploading videos on a video sharing website like YouTube, files are transcoded by the platform, stripping them of all stored metadata. Therefore, a huge amount of important information is removed, almost disabling queries over videos. Moreover, by removing metadata, media interaction on the web is becoming less and less usable, so frameworks like Popcorn.js[3] enable developers to create time-based interactive web media.

Addressing to this problem, our work focuses mostly on descriptive metadata and how they can be embedded on the content and not just the file. The distortion of image is necessary but is insignificant and the disadvantage of the limited payload of the QR code can still be enough for various applications. Our goal is not to make the other metadata formats obsolete but to offer an alternative way on how the metadata can be transmitted.

## 2.2 QR Codes

QR code [6], an abbreviation for Quick Response code, developed by Denso Wave in 1994, is a two dimensional barcode that can store more information compared to the traditional barcode. They have recently become popular because of their use in modern smart phones and mobile devices. One of their big advantages, besides the ability to store more data, QRs have the ability of error correction. Even if they are partially distorted, due to physical distortion of the code, lack of light sources etc, one can still restore the embedded data.

They have a square crossword form, where every small square is called module. According to the data to be stored, the QRs range from $21 \times 21$ up to $177 \times 177$ modules. Since they can be easily read and correct a big extend of possible

[3] `http://popcornjs.org/`

**Table 2.** QR code data capacity

| | |
|---|---|
| Numeric code only | Max. 7,089 characters |
| Alphanumeric | Max. 4,296 characters |
| Binary (8 bits) | Max. 2,953 bytes |
| Kanji/Kana | Max. 1,817 characters |

**Table 3.** Error correction of QR codes

| | |
|---|---|
| Level L | 7% of codewords can be restored. |
| Level M | 15% of codewords can be restored. |
| Level Q | 25% of codewords can be restored. |
| Level H | 30% of codewords can be restored. |

errors, modern mobile devices are shipped with special libraries in order to read the encoded messages. Therefore, one may take a photo of a QR that is placed on a magazine or in a museum tag with his smartphone and instantly be redirected to a web page containing additional information. Their capacity, as well as their error correction capabilities are illustrated in Tables 2 and 3.

The watermarking images with QR codes has already drawn the attention of the research community in several works such as [7,8,9,10,11,12,13,14]. Moreover, there is an application of QR code embedding in audio [15]. Even if the idea of embedding metadata in the video content have already proposed it has only been used in a specific domain, that of UAV videos [16,17] and without the use of a 2D barcode. To the best of our knowledge, only one robust watermarking scheme that embeds QR codes in video exist so far [18] but the scope of that work is the copyright protection of the content and not the additional functionality that metadata have to offer.

## 3 Proposed Method

### 3.1 Application Scenario

Smooth integration and platform independence is a major issue in software development. The wide range of video formats and players makes co-operation between applications and cross-platform development of content-aware applications very difficult.

Let's assume that we have a media player that is playing a video. The displayed information in many cases would be beneficial input for other running applications and services. It would be very handy for example to be able to extract several metadata from the video, so that the browser displays relevant information about the actors or the place that the video takes place. Even for targeted-advertisement applications, this information is very important, as this would enable them to display content related advertisement.

To allow this information flow, developers could create a buffer, where video players send their metadata, so that other applications can extract them, independently of how this information is stored. It is clear that this creates a problem when we are talking about Operation System-independent development and multi application environments. Moreover, one could block access to this buffer, crippling this feature.

### 3.2 Core Idea

To this end, we propose embedding the metadata information as part of the video, so that independently of how a user is watching a video, from which combination of application/operating system, this information can be retrieved. One method to achieve this is by embedding the metadata in a watermark-like way in the LSB. The idea is that applications could take a screenshot of the user's desktop, keep the LSB, trace the watermark and extract the information.

It is clear that the aforementioned method can be applied to any operating system and any multimedia player, with trivial overhead. Moreover, by embedding metadata in this form, the metadata are:

– Stored with the file.
– Are always synced with the content.
– Remain after processing the video, which introduce low noise.
– Remain if a video is split or videos are merged.

Since we want to allow the information to be retrieved from a screenshot, where the video can be placed in any part of the screen, there is an obvious need for clear patterns that can easily be traced. Moreover, it would be very beneficial if the pattern could be easily recognized by existing libraries and if it could allow error correction. The criteria above have led us to use QR codes in the LSB as a test-bed, to explore the efficiency, the storage, the performance and distortion of our solution.

### 3.3 Implementing the Proposed Method

In order to test the proposal to the extremes, we embedded three QR codes in each frame of the videos, one in every color component. With the use of Least Significant Bit (LSB) we alter the bits of every color component to create a binary image of a QR, containing the desired metadata. The payload of the QR depends on the version of the QR, which can also be resized to be smaller in the embedding and in the extraction process to be inversely resized. This scheme is able to retain the metadata in simple transformations and lossless compression of the multimedia stream. Figures 1 and 2 illustrate the embedding process, as along with a frame where information has been embedded with the proposed method.

To embed the metadata, the algorithm reads the movie file properties and then uses the smallest size of height or width, to create a square space in the center of the frame for all the color components where the QR code will be embedded. Then using LSB, the message is embedded in each color component.

Metadata extraction from the multimedia stream process is very straightforward. If we have access to the actual video file, we load each frame and compute the modulo 2 for every color component, resulting to a binary table. This table is then processed by the QR reader to return the data to the user. Otherwise, an application takes a screenshot of the user's desktop and computes the modulo 2 for every color component, resulting again to a binary table, which is parsed to a QR reader. It is clear that due to the QR pattern, the metadata can be extracted without any problem.

The stored metadata have no limitations on the format and the content to be stored. In the current implementation, vcard format, URLs, geotagging information etc are supported. Depending on the video, users could store movie subtitles. This way, multiple could simultaneously be supported without any synchronization problem.

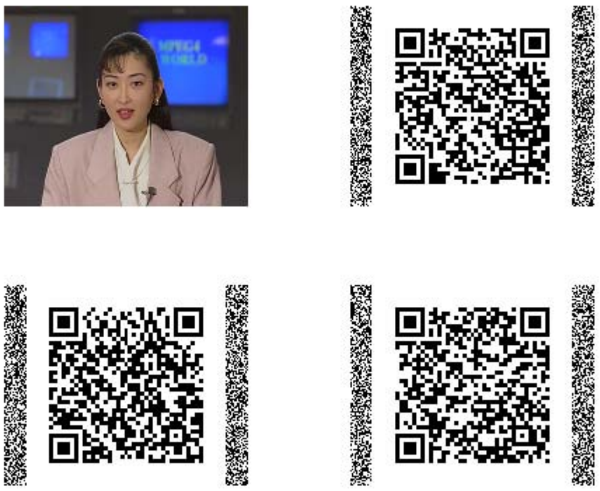

**Fig. 1.** A frame with it's embedded QR code

## 4 Experimental Results

The experiments were conducted using Matlab. The videos that were used can be found at [19]. All 14 video sequences are in the uncompressed YUV4MPEG format and in two different resolutions. Seven Cif video files are 352 by 288 with 300 frames each, while the other seven Qcif files are 176 by 144 with 300 frames each with the same content. The quality measurement of distortion were made using the Matlab package Image quality measures, developed by Athi Narayanan. The metadata that were stored in each QR, were a random 120 character stream

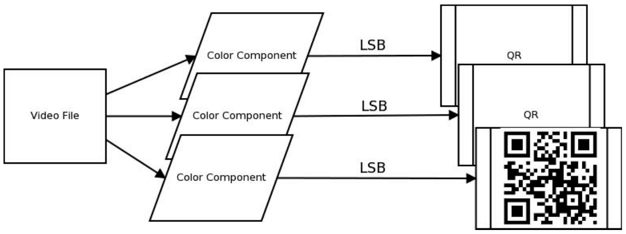

**Fig. 2.** Embedding process

**Table 4.** Video quality measurements of cif files

| | Mean Sq. Error | PSNR | Normalized cross-correl. | Structural Content | Average Diff. | Max Diff. | Normalized Abs. Error |
|---|---|---|---|---|---|---|---|
| akiyo | 0,5051 | 51,0974 | 0,9984 | 1,0033 | 0,2189 | 1 | 0,0048 |
| bowing | 0,4998 | 51,1433 | 0,9986 | 1,0028 | 0,2012 | 1 | 0,0043 |
| coastguard | 0,4985 | 51,1544 | 0,9985 | 1,0029 | 0,2005 | 1 | 0,0042 |
| container | 0,5018 | 51,1252 | 0,9985 | 1,0029 | 0,2049 | 1 | 0,0035 |
| foreman | 0,4985 | 51,1541 | 0,9987 | 1,0026 | 0,2 | 1 | 0,0034 |
| hall_monitor | 0,5032 | 51,1138 | 0,9985 | 1,003 | 0,209 | 1 | 0,0038 |
| news | 0,4991 | 51,1486 | 0,9985 | 1,003 | 0,2008 | 1 | 0,006 |
| **Average** | 0,5009 | 51,1338 | 0,9985 | 1,0029 | 0,2050 | 1 | 0,0043 |

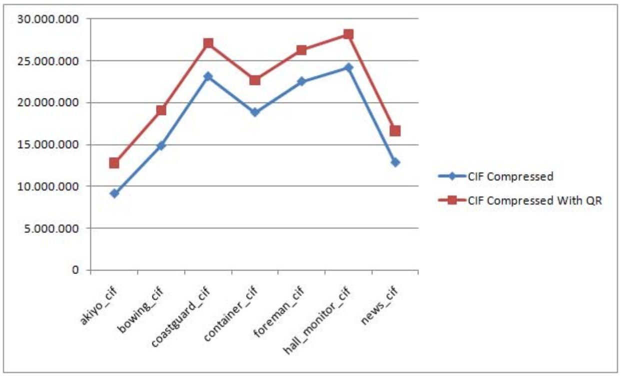

**Fig. 3.** Filesize difference of original vs QR embedded cif files

in both sizes of the video files (360 characters per frame). We chose YUVsoft codec [20] for compression in order to have a lossless video format [21].

The visual impact of this distortion was not traceable by the human eye, while the PSNR values remained close to optimal the experimental results are illustrated in Figures 3,4 and Tables 4,5. It is clear, that the implementation that was made was targeting to show that even in the extreme cases, the method can still be applied without significant quality cost. A more fine-grained approach would be to store one QR that would be split among the three color components, so that their XOR for example resulted to the needed QR. Additionally, one

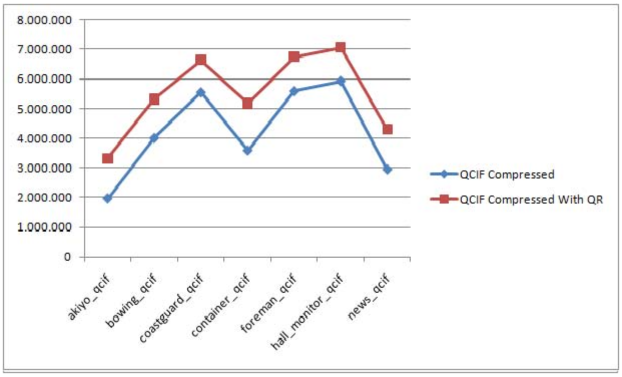

**Fig. 4.** Filesize difference of original vs QR embedded qcif files

**Table 5.** Video quality measurements of qcif files

| | Mean sq. error | PSNR | Normalized cross-correl. | Structural Content | Average Diff. | Max Diff. | Normalized Abs. error |
|---|---|---|---|---|---|---|---|
| akiyo_qcif | 0,5043 | 51,1043 | 0,9984 | 1,0033 | 0,2173 | 1 | 0,0048 |
| bowing_qcif | 0,4978 | 51,1603 | 0,9986 | 1,0028 | 0,2004 | 1 | 0,0043 |
| coastguard_qcif | 0,4968 | 51,1686 | 0,9985 | 1,0029 | 0,1996 | 1 | 0,0042 |
| container_qcif | 0,4991 | 51,1487 | 0,9986 | 1,0029 | 0,2022 | 1 | 0,0035 |
| foreman_qcif | 0,4966 | 51,1708 | 0,9987 | 1,0025 | 0,1989 | 1 | 0,0034 |
| hall_monitor_qcif | 0,5014 | 51,1288 | 0,9985 | 1,003 | 0,2078 | 1 | 0,0037 |
| news_qcif | 0,4991 | 51,1493 | 0,9984 | 1,0031 | 0,2019 | 1 | 0,006 |
| Average | 0,4993 | 51,1473 | 0,9985 | 1,0029 | 0,2040 | 1 | 0,0043 |

could allow the embedding of distorted QR, up to a certain threshold. Clearly such approach has even less distortion and stores less information, nevertheless it does not highlight the balance between the amount of information that can be stored in each frame versus the quality of the video.

## 5 Conclusions

Definitely, the proposed solution invokes an obvious disadvantage, that of partially distorting the center of the frame with LSB embedding QRs. Nevertheless, the experiments indicate that the distortion is quite small. On the other hand, the proposed method for embedding metadata has several advantages over currently used methods. The most obvious one is that the metadata are embedded in the video file, more precisely inside the frames. This enables the transfer of metadata in case of file conversion to another lossless video format. Moreover, the metadata are kept even if the video files are split and merged, without synchronization problems.

It should be highlighted that even though we introduce a new approach for storing metadata, we do not propose the use of any metadata standard. We believe that this decision should be left to the developer that adopts such solution.

The only constraint is that the size of metadata which can be stored, obviously depends on the size of the video, in terms of resolution and length.

The proposed method enables easy queries on stored metadata, and their storage follows a well known and standardized industry format, which is noise tolerant. The extraction is so easy that taking a screenshot from any frame, with any application, one may retrieve the stored metadata.

A very important feature is that by embedding information this way, developers can extract the information without the need for cross-platform frameworks, publicly accessible registers etc. Therefore, the solution can be adopted by any application in every operating system, that enables it to take screenshots of user's desktop.

Our future work will focus on extending this embedding process in order to survive possible attacks. Moreover, we are planning to work on certain lossy compression formats and less noisy solutions that can support several image and geometrical transformations, beyond rotation, flipping, which are currently supported.

## References

1. Jain, R., Hampapur, A.: Metadata in video databases. SIGMOD Rec. 23(4), 27–33 (1994)
2. Catarci, T., Donderler, M., Saykol, E., Ulusoy, O., Gudukbay, U.: Bilvideo: A video database management system. IEEE MultiMedia 10(1), 66–70 (2003)
3. Kosch, H.: Mpeg-7 and multimedia database systems. ACM SIGMOD Record 31(2), 34–39 (2002)
4. Mpeg 7, `http://mpeg.chiariglione.org/standards/mpeg-7` (accessed June 10, 2013)
5. Zigomitros, A., Patsakis, C.: Cross format embedding of metadata in images using QR codes. In: Tsihrintzis, G.A., Virvou, M., Jain, L.C., Howlett, R.J. (eds.) IIMSS 2011. SIST, vol. 11, pp. 113–121. Springer, Heidelberg (2011)
6. QR codes, `http://www.qrcode.com/en/` (accessed June 20, 2013)
7. Zhang, S., Yoshino, K.: Dwt-based watermarking using qr code
8. Vongpradhip, S., Rungraungsilp, S.: Qr code using invisible watermarking in frequency domain. In: 2011 9th International Conference on ICT and Knowledge Engineering (ICT & Knowledge Engineering), pp. 47–52. IEEE (2012)
9. Lee, H.-C., Dong, C.-R., Lin, T.-M.: Digital watermarking based on JND model and QR code features. In: Pan, J.-S., Yang, C.-N., Lin, C.-C. (eds.) Advances in Intelligent Systems & Applications. SIST, vol. 21, pp. 141–148. Springer, Heidelberg (2012)
10. Hsu, F.-H., Wu, M.-H., Wang, S.-J.: Dual-watermarking by qr-code applications in image processing. In: 2012 9th International Conference on Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC), pp. 638–643. IEEE (2012)
11. Xie, R.-S., Wu, K.-S., Xu, G.-P., Ouyang, M.: Research on anti-counterfeiting quick response 2d barcode techniques based on digital watermark. Journal of Shanghai Jiaotong University (Science), 1–5 (2013)
12. Islam, M., Alzahir, S., et al.: A novel qr code guided image stenographic technique. In: 2013 IEEE International Conference on Consumer Electronics (ICCE), pp. 586–587. IEEE (2013)

13. Felix Gunalan, G., Nithya, J.: Qr code hiding using histogram shifting method
14. Chen, J.-H., Chen, C.-H.: Image tamper detection scheme using qr code and dct transform techniques
15. Poomvichid, T., Patirupanusara, P., Ketcham, M.: The qr code for audio watermarking using genetic algorithm
16. Marcinak, M.P., Mobasseri, B.G.: Digital video watermarking for metadata embedding in uav video. In: IEEE Military Communications Conference (MILCOM 2005), vol. 3, pp. 1637–1641. IEEE (2005)
17. Philp, A., Bradley, B., Stach, J., Rodriguez, T.: Real-time application of digital watermarking to embed tactical metadata into full motion video captured from unmanned aerial systems. In: IS&T/SPIE Electronic Imaging, International Society for Optics and Photonics, p. 725410 (2009)
18. Prabakaran, G., Bhavani, R., Ramesh, M.: A robust qr-code video watermarking scheme based on svd and dwt composite domain. In: 2013 International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME), pp. 251–257. IEEE (2013)
19. Xiph, `http://media.xiph.org/video/derf/` (accessed June 28, 2013)
20. YUVsoft codec, `http://www.yuvsoft.com/download/lossless-codec/index.html` (accessed June 13, 2013)
21. Lossless Video Codecs Comparison, `http://compression.ru/video/codec_comparison/pdf/msu_lossless_codecs_comparison_2007_eng.pdf` (accessed June 21, 2013)