Expanding the Data Capacity of QR Codes Using Multiple Compression Algorithms and Base64 Encode/Decode

Azizi Abas, Dr Yuhanis Yusof, and Farzana Kabir Ahmad
School of Computing, Universiti Utara Malaysia, 06000 Sintok, Kedah.
azizia@uum.edu.my

***Abstract*—The Quick Response (QR) code is an enhancement from one dimensional barcode which was used to store limited capacity of information. The QR code has the capability to encode various data formats and languages. Several techniques were suggested by researchers to increase the data contents. One of the technique to increase data capacity is by compressing the data and encode it with a suitable data encoder. This study focuses on the selection of compression algorithms and use base64 encoder/decoder to increase the capacity of data which is to be stored in the QR code. The result will be compared with common technique to get the efficiency among the selected compression algorithm after the data was encoded with base64 encoder/decoder.**

***Index Terms*—QR Code; Data Compression; Base64 Encoder/Decoder.**

I. INTRODUCTION

A barcode [1] is an optical machine-readable which is consists the data pertaining to the object to which it is given. Primitive bar codes, represent data by varying the widths and space of parallel lines, and they may be referred to as linear or one dimensional code. One dimensional barcode does not hold as much data as compared to the two-dimensional barcode [2]. Figure 1 illustrates the difference between one dimensional barcode and two-dimensional barcode. The design of a two-dimension code (i.e QR Code), in figure 1, shows considerably a greater volume of information than one dimension barcode.

The Quick Response code (QR code) [3][4] is a new technology to keep data and information in a medium range of capacity. It is a popular type of two-dimensional barcodes that was developed by Denso Corporation Japan in 1994. QR Code [5] is registered by the ISO/IEC 18004 of industrial standard. The QR code [6] is widely used in Japan, Europe, America and other developed countries due to effective mode of carrying and the information transmission also includes certain security function. The QR codes are used to track parcel, item tagging, transport ticketing, contact information, website uniform resource locator, identity verification and several types of useful information request. Technically, the QR code is a black and white graphical image which can store information both horizontally and vertically.

The characteristics contained in the QR code are capability in highly speed recognition, robustness in error-correcting capability, able to recognize expression in Kanji and Kana symbols, structured append which is can be splitting up to 16 segments [7], no magnetic tape is used to store information so the cost is reduced [8] and can be scanned in all directional angle.

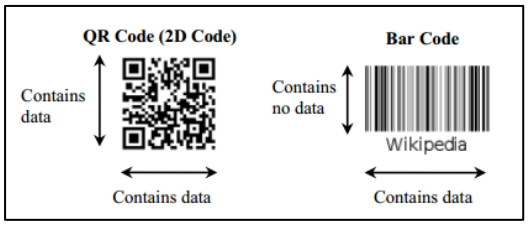

Figure 1: Difference between one dimensional barcode and two-dimensional barcode.

The two dimensional QR code [2] can encode various data including numeric, alphanumeric, symbols, kanji characters and binary 8 bytes. Table 1 shows the basic characteristic of QR Code.

Table 1
The basic characteristic of QR Code

Encodable character set	• Numeric (0-9) • Alphanumeric data (Digits 0 - 9; upper case letters A-Z; nine other characters: space, $ % * + - . / :) • 8-bit byte data • Kanji characters
Color Module	• A dark module is a binary 1 • A light module is a binary 0
Versions	• Version 1 until 40
Error Level Correction	• L -7% or less errors can be corrected. • M 15% or less errors can be corrected. • Q 25% or less errors can be corrected. • H 30% or less errors can be corrected.
Type of QR Code	• Model 1 with maximum version being 14 (73 x 73 modules) and 2 with maximum version being 40 (177 x 177 modules). • Micro with one orientation detecting. • iQR with rectangular code, turned-over code, black-and-white inversion code or dot pattern code (direct part marking). • SQRC with limited specific types of scanners. • LogoQ with combine designability and readability.

II. LITERATURE REVIEW

This section discusses the anthology associated with QR codes and the structure of those codes. The popularity of QR codes depends on its capability symbolizing same amount of data in approximately one tenth the space of a one dimension barcode [1].

A. Storage Capacity

To date, there is an explosion of information surrounding the community. There is an increased amount of data that comes in various forms such as emails, pictures, and videos, all of which must be accessible in a timely and dependable fashion. This data can be stored in our personal computers or in data centers around the world (cloud computing). Because the growing data requirements, storage is rapidly becoming an important factor in data center IT equipment. A recent survey by Gartner, Inc. (2015) reveals that data growth is the greatest challenge for larger enterprises. The memory storage has kept increasing due to demand of the users.

QR code [11] consists matrix symbols which have arrays of nominally square modules arranged in square pattern. There are 40 versions of QR code that have a specific task or purpose. The difference between each version is the number of modules. In version 1, it consists 21 x 21 module that can store up to 133 encoded characters. However, version 40 has 177 x 177 modules that can store nearly 23648 data modules (2956 encoded characters). Table 2 shows the character capacities by version (1, 20 and 40), error correction level, and mode of QR code.

Table 2
The character capacities by version (1, 20 and 40), error correction level, and mode of QR code

Versions	Error Correction Level	Numeric Mode	Alphanumeric Mode	Byte Mode	Kanji Mode
1	L	41	25	17	10
	M	34	20	14	8
	Q	27	16	11	7
	H	17	10	7	4
20	L	2061	1249	858	528
	M	1600	970	666	410
	Q	1159	702	482	297
	H	919	557	382	235
40	L	7089	4296	2953	1817
	M	5596	3391	2331	1435
	Q	3993	2420	1663	1024
	H	3057	1852	1273	784

According to The International Standard ISO/IEC 18004, the process of basic generation of QR code is as in Figure 2.

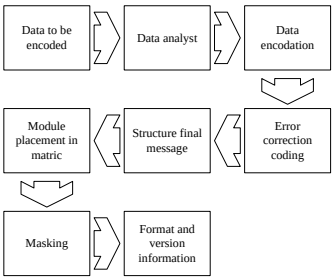

Figure 2: Basic generation of QR code (Courtesy: International Standard ISO/IEC 18004 (Denso Incorporation, 2006))

The output result of the process in Figure 2 is a QR code image. The structure of QR code in Figure 3 shows the interface of QR code and the design along with an explanation of QR code surface. According to Galiyawala [13], there are eight significant parts of a QR code architecture. The parts are (a) Finder pattern (1) - a decoder software is able to recognize the QR code and ensure the correct orientation, (b) Separators (2) - as the separator between finder pattern and code data, (c) Timing pattern (3) - to ensure the decoder software to determine the width of a single module, (d) Alignment patterns (4) - enable the decoder software compensating the image, (e) Format Information (5) - to keep the error correction level of the QR Code and the chosen masking pattern, (f) Data (6) - the 8 bit codewords data, (g) Error correction (7) - the 8 bit codewords error correction, (h) Remainder bits (8) - the empty bits if data and error correction bits cannot be divided into 8 bit codewords without remainder.

Figure 3: The structure of QR code version 2

B. Compression

Compression [8] is an algorithm used to reduce file size which turns storage space into minimal compact data usage. Moreover, it makes the transmission of data over line faster than uncompressed file. The art of compression is to eliminate the redundancy data and squeeze the size using relevant compress process. In general, there are two types of compression (a) Lossless compression - does not lose any part of data and retrieve back the data after decompression, (b) Lossy compression – it does loose some data to achieve higher compression. Table 3 shows the comparison of advantage and disadvantage between lossless and lossy compression.

Table 3
The comparison of advantage and disadvantage between lossless and lossy compression

	Advantage	Disadvantage
Lossly	Use less space Ratio compression is high	Possibility of losing some data
Lossless	One to one input and output	Consume more space and memory

Nowadays, the lossless compression used various encoding schemes such as Lempel-Ziv, Huffman, Deflate, GZip, TTA, FLAC, Zip etc. On the other hand, the lossy encoding scheme utilize MPEG-2, MPEG-3, MPEG-4 codec, psychoacoustics etc. Table 4 shows the description of various lossless compressors schemes.

Table 4
The description of various lossless compressors [14]

Name	Developer	File Extension	Base Algorithm used
GZip (GNU Zip)	Jean-Loup Gailly and Mark Adler	.gz	Deflate algorithm, which is a combination of LZ77 and Huffman coding
Zip	Phil Katz	.zip	Deflate algorithm
LZW	Abraham Lempel, Jacob Ziv, and Terry Welch.	.gif	LZ78 algorithm
Huffman coding	David A. Huffman	.txt	Huffman's algorithm

C. Base64 Encoder

The Base64 [15] is a binary to text encoding scheme that represents binary data in an ASCII string format by translating it into a radix 64 representation. It can transmit data from binary into ASCII characters. Also, it was designed to represent arbitrary sequences of octets in a form that allows the use of both upper- and lowercase letters but that need not be human readable [16]. It can also convert a file to a string format which only contains 64 ASCII characters (i.e., A–Z, a–z, 0–9, +, /) with a special suffix "=" used for padding [17].

According to Rawat, Sahu, & Puthran [18], the base64 encoding undergoes six phases. The first phase divides the input bytes stream into blocks of 3 bytes. Then it divides 24 bits of each 3-byte block into 4 groups of 6 bits, this is followed by mapping each group of 6 bits to 1 printable character, based on the 6-bit value using the base64 character set map as shown in Table 5. Later if the last 3-byte block has only 1 byte of input data, pad 2 bytes of zero (\x0000). After encoding it as a normal block, it overrides the last 2 characters with 2 equal signs (==), so the decoding process knows 2 bytes of zero were padded. If the last 3-byte block has only 2 bytes of input data, pad 1 byte of zero (\ x00). After encoding it as a normal block, override the last 1 character with 1 equal signs (=), so the decoding process knows 1 byte of zero was padded. Finally, carriage return (\r) and new line (\n) are inserted into the output character stream.

Table 5
Character set map by Base64 encoding

Value	Encoding
0-25	A-Z
26-51	a-z
52-61	0-9
62	+
63	/

D. ZXing Library

ZXing [19] (pronounced as "zebra crossing") is an open-source system and multi-format 1D/2D barcode image processing library which is implemented in Java programming language. It can support various encode and decode barcode including QR code. There are five main component libraries for desktop (QR code) which are (a) core – the core image decoding library, (b) javase - J2SE-specific client code, (c) zxingorg – source file in zxing.org/w (d) zxing.appspot.com - web-based barcode generator, (e) glass - Simple google glass application.

This paper will focus on using the methods provided by ZXing library to scan, encode and decode QR codes without communicating with a server. The decode method will use PNG file as a input. During encode and decode processes, the input will use image processing libraries provided by ZXing library.

The ZXing library is easy to integrate into the application because there are a lot of constructors and methods installed in it. Kris Antoni Hadiputra Nurwono and Raymondus Kosala [20] are using ZXing 0.6 as a tool in their research work to develop the mobile barcode reader. Meanwhile, Antonio Grillo etc al. [21] are using ZXing to develop a decoder module for research work prototype that implements the Print&Scan process for High Capacity Color Two Dimensional codes. Thus, the ZXing library is a common type of library in Java which is use to develop QR code application in research work.

E. Compressed QR Code

According to Nancy Victor [2], compressing the data before generating the QR code is more efficient to improve data capacity of QR code. In addition, data capacity can be improved by combining the most distinguish features of compression and QR code generation. This study investigates the idea of encoding compressed data. Figure 4 shows the flow of generating high capacity QR code as proposed by Nancy Victor [2].

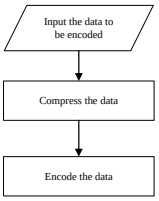

Figure 4: The flow to generate high capacity QR code [2].

III. METHODS

This study focuses on four compression algorithms and its combination. On the other hand, a normal QR code generation is used as a benchmark. The compression algorithms to be tested are the Zip, GZip, LZW, Huffman Coding, LZW-GZip and Huff-Zip. After compressing the data, the compressed data will be embedded to the QR code generator developed using the ZXing image processing library.

A. Experimental Setup

The undertaken experiments includes several hardware and software requirements. The study utilizes Intel i7 processor, 8Gb memory and 800Gb spaces. Meanwhile, the required software includes Windows 7 operating system, NetBean IDE, Notepad, ZXing library, JDK 1.8 compiler, Sun Base64 decoder library, Apache common decode library and compression libraries (GZip, Zip, LZW and Huffman code)..

B. QR Code Encoding Process

The process of encoding involves several parts which are starting with generating a raw input file called constant.txt. The constant.txt file will receive characters starting with one character until thousands of characters. Figure 5 shows the snapshot of constant.txt file.

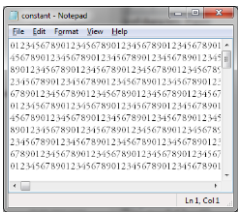

Figure 5: The snapshot of constant.txt file

The process of receiving the characters will end when the Java program generates IOException message called "com.google.zxing.WriterException: Data too big". Then the process will stop. As the flow of the process, after the process of receiving characters is completed, the compression algorithm will compress constant.txt file and will be named by filename extension of compression such as: constant.gz.

For next process, the compressed file name will be decoded by base64 encoder and as a result, the base64 encoder will produce an array of byte data type contains encoded base64 data. The encoded base64 data are converted to a String literal and put into QR code generator method as an input. This process will generate a QR code image. Figure 6 shows the process flow process of encoding the QR code.

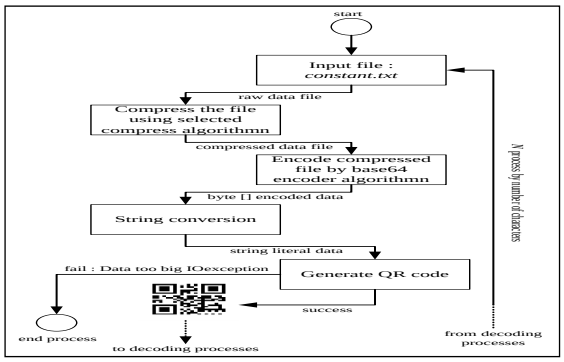

Figure 6: The process flow of encoding the QR code

C. QR Code Decoding Process

When the QR code is generated, the next step is to decode the QR code image. The process starts with binarization of the QR code image. It will return decoded string literal if the process is successfull. If not, the null string literal will be sent and the process is not successful.

The next process is to decode the successful string literal into the Base64 decoder method. As a result, Base64 decoder will generate the compress filename according to the previous compressed algorithm. The compressed filename needs to uncompress back, which is the compressor algorithm will take action to get back normal text filename previously used as an input file. Figure 7 shows the process of decoding the QR code.

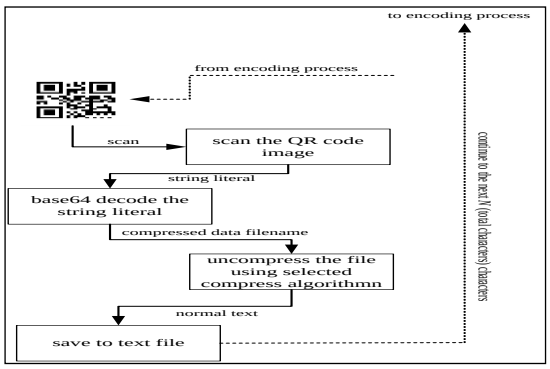

Figure 7: The process flow of decoding the QR code

D. Experiments

The experiment was divided into two phases. In the first phase, the base64 encoder/decoder is not tested due to see the impact of data capacity using ASCII encoder/decoder (normal implementation). But in the second phase, it will include the base64 encoder/decoder.

The first experiment consists random alphanumeric without carriage return and line feed as input data with error correction level H. Meanwhile, the second experiment includes fixed alphanumeric without carriage return and line feed as input data with error correction level L, M, Q and H.

The comparison is based on the total character stored in the produced QR code. Figure 8 shows the raw data used in the experiment.

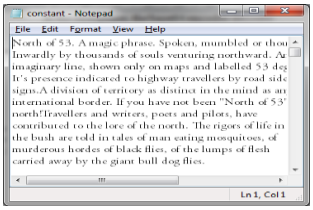

Figure 8: The fixed alphanumeric actual input data

E. Results

This section includes the obtained results of the proposed method. Using the technique of compression and encoding/decoding , may disclose the gap of storage capacity between normal implementation and the proposed method.

a. The First Phase

Results of the first phase is depicted in Table 6 and Table 7. The experiments were carried out twenty times in order to obtain the minimum total character stored in the QR code at error correction level H. The reason of such action is because the input data file contains different characters (due to random character implementation), hence may produce different size of files. . Table 6 includes results based on the maximum number of characters while Table 7 includes data for the minimum size.

Table 6
Result of maximum total characters stored in QR code from 20 times tested at error correction level H

No. Test	Normal	Zip	GZip	LZW	Huffmann Coding	Huffman + GZip
1	1271	474	635	434	113	471
2	1271	471	638	434	112	466
3	1271	476	637	433	111	477
4	1271	472	636	436	114	474
5	1271	475	637	433	112	470
6	1271	473	635	438	112	473
7	1271	475	635	433	111	474
8	1271	473	641	438	111	472
9	1271	474	636	438	114	468
10	1271	474	638	439	113	470
11	1271	473	634	433	113	468
12	1271	473	637	438	111	474
13	1271	471	633	441	111	477
14	1271	471	634	433	111	472
15	1271	473	635	437	113	471
16	1271	469	636	440	111	471
17	1271	470	636	438	111	479
18	1271	469	633	437	113	471
19	1271	477	636	436	112	467
20	1271	478	632	433	109	467

Table 7
The summarized minimum total character stored in QR code at error correction level H

Normal	Zip	GZip	LZW	Huffmann Coding	Huffman and GZip
1271	469	632	433	109	466

From the graph in shown Figure 9, it is learned that compression methods do not contribute in extending the storage capacity. . The percentages difference between the

proposed methods and the normal implementgation are (a) Zip – 63%, (b) GZip – 50%, (c) LZW – 66%, (d) Huffmann Coding – 91% (e) Huffmann + GZip – 63%. The smallest difference is the one obtained using GZip compression algorithm while Huffmann Coding produces the largest.

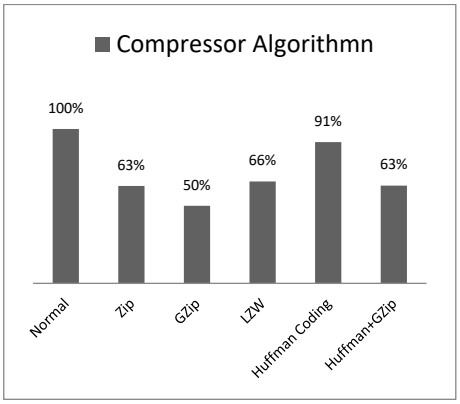

Figure 9: The percentage gap between normal process and the selected compressor algorithms

b. The Second Phase

In the second phase of experiment, the base64 encoder/decoder and fixed character composition were embedded. The results were separated by the error level as shown in Table 8. Each experiment is only performed once as it uses fixed composition characters in the input file where the compressor algorithm will generate same size files.

Table 8
The maximum total characters stored in QR code by error level

Error Level	Normal	Zip	Gzip
H	1270	1560	1784
Q	1662	2114	2405
M	2330	3188	3470
L	2952	4226	4480

LZW	Huffman Coding	Huffman And Gzip	Huffman And Zip
1167	212	1364	1166
1627	282	1827	1639
2441	392	2607	2425
3253	503	3323	3095

From the results in Table 8, the graphs were generated as shown in Figure 10, 12, 13 and 14.

In error level H (High), the highest total characters are GZip compression algorithm . The QR code can hold up to 1784 characters. At the H level, the data are covered by 30% of the codeword in error respectively. The version of QR code created in this experiment is the version 40. Figure 11 shows the generated QR code.

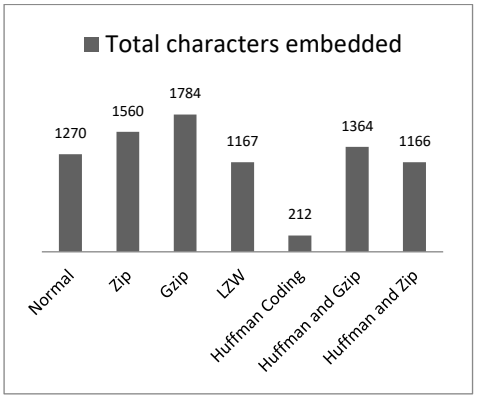

Figure 10: The maximum total characters of normal and selected compression algorithm separated by error correction level H

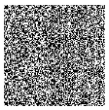

Figure 11: The generated version 40 QR code

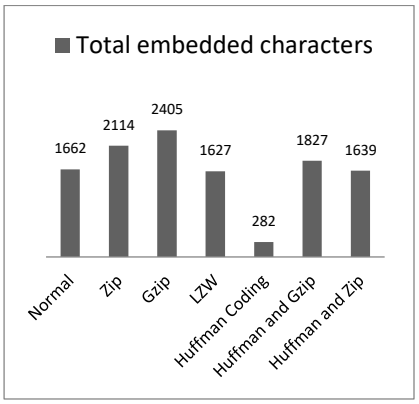

Figure 12: The maximum total characters of normal and selected compression algorithm separated by error level Q

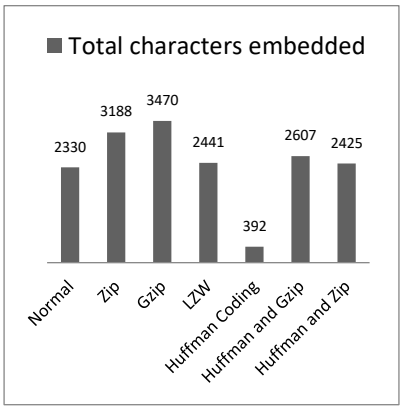

Figure 13: The max total characters of normal & selected compression algorithms separated by error level M

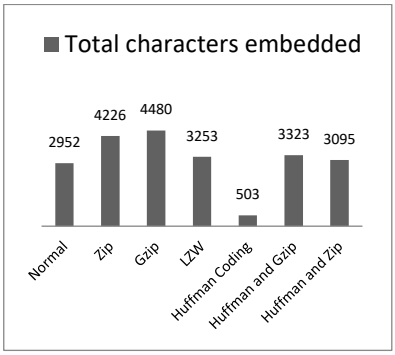

Figure 14 The maximum total characters of normal and selected compression algorithm separated by error level L

The error correction level Q (quartile) consists 20% of the codeword in error. The GZip compression algorithm still can hold the highest characters compared to normal and other selected compression algorithm. It covered 2114 characters and the different between error correction level H and Q is 621 characters.

In the error correction level M (medium), the total characters that QR code can hold increases. This is because the total correction is decreased to 15%. It increases up to 3470 characters.

The last error correction level H (high) produces the largest number of characters that a QR code can hold The QR code at this level can hold until 4480 characters and the size of the file is 4.375 kilobytes.

The undertaken experiments reveal that GZip is the best compression algorithm as it is able to compress the maximum text data and able embedded in QR code compared to other selected compression. Nevertheless, it must be encoded using base64 encoder/decoder not ASCII encoder. Table 8 shows the gap of a number of characters between normal and selected compression algorithms.

Gzip algorithm can exceed more than 40% of data compression compared to other algorithms. Clearly here that GZip is the best text data compression at all levels of error-correction followed by Zip algorithm. Meanwhile, Huffman Coding is not suitable for text data compression because it gives a negative percentage compared to normal text. Table 9 shows the gap percentage between total normal characters and selected compression algorithm with four error level correction.

Table 9
The total number of characters gap between normal and selected compression algorithm with four error level correction

Total Normal	Zip	Gzip	LZW	Huffman Coding	Huffman and Gzip	Huffman and Zip
1270	290	514	-103	-1058	94	-104
1662	452	743	-35	-1380	165	-23
2330	858	1140	111	-1938	277	95
2952	1274	1528	301	-2449	371	143

Table 10
The gap percentage between total normal character and selected compression algorithm with four error level correction

Normal (characters)	Zip (%)	Gzip (%)	LZW (%)	Huffman Coding (%)	Huffman and Gzip (%)	Huffman and Zip (%)
1270	23	40	-8	-83	7	-8
1662	27	45	-2	-83	10	-1
2330	37	49	5	-83	12	4
2952	43	52	10	-83	13	5

IV. CONCLUSION

This study investigates mechanism to extend the data storage in a QR code. The undertaken method is to compress the text data and utilizes the Base64 to encode/decode the QR code. The high data density helps to minimize the space used for printing in the QR code images. Disadvantage regarding the QR code is the users must provide smartphone embedded with a camera and the correct software for encode and decode. Furthermore, Base64 is limited to 64 characters representation to encode and decodes process and not human readable. The main advantage of base64 is able to transmit data from binary, into (most commonly) ASCII characters.

Further enhancement of QR code data density is focusing capability to increase the data into more than 52% as a maximum result from the GZip compression algorithm tested before.

Secure QR code can also be implemented by using encryption techniques [2]. More advance data compression technique can be introduced to get more data capacity from normal QR code.

REFERENCES

[1] Pandya K. H. and Galiyawala H. J. 2014. A Survey on QR Codes: in context of Research and Application. *International Journal of Emerging Technology and Advanced Engineering*. 4(3): 258-262.

[2] Victor N. 2012. Enhancing the Data Capacity of QR Codes by Compressing the Data before Generation. *International Journal of Computer Applications (0975 –8887)*. 60(2): 17-21.

[3] An Lin J. and Shann Fuh C. 2013. 2D Barcode Image Decoding," *The Scientific World Journal*. 2013(3): 1.

[4] Chen W., Yang G., and Zhang G. 2012. A Simple and Efficient Image Pre-processing for QR Decoder, in *2nd International Conference on Electronic & Mechanical Engineering and Information Technology (EMEIT-2012)*. 234-238.

[5] Qianyu J. 2014. Exploring the Concept of QR Code and the Benefits of Using QR Code for Companies.

[6] Xiaoyang Y., Yang S., Yang Y., Shuchun Y., Hao C., and Yanxia G. 2013. An Encryption Method for QR Code Image Based on ECA. *International Journal of Security and Its Applications.* 7(5): 397-406.

[7] Swetake Y. 2014. How to create QRcode, *swetake.com*, Online]. Available: http://www.swetake.com/qrcode/qr1_en.html.

[8] Goel S. and Singh A. K. 2014. Cost Minimization by QR Code Compression, *International Journal of Computer Trends and Technology (IJCTT)*. 15(4): 157-161.

[9] Belussi L. F. F. and Hirata N. S. T. 2013. Fast Component-Based QR Code Detection in Arbitrarily Acquired Images. *Journal Math Imaging Vis.* 45: 277-292.

[10] Christy Pettey L. G. Gartner Survey Shows Data Growth as the Largest Data Center Infrastructure Challenge. *Gartner, Inc. (NYSE: IT)*, 2015. [Online]. Available: http://www.gartner.com/newsroom/id/1460213.

[11] Harish N. and Gurav S. 2014. Embedding a Large Information In QR Code Using Multiplexing Technique, *Taraksh Journal of Communications*.1(6): 6-9.

[12] Incorporation D. 2006. Information technology automatic identification and data capture techniques bar code symbology QR Code.

[13] Galiyawala H. J. and Pandya K. H. 2014. To Increase Data Capacity of QR Code Using Multiplexing with Color Coding: An example of Embedding Speech Signal in QR Code. in *2014 Annual IEEE India Conference (INDICON)*. 2-7.

[14] Gailly J.-L. and Adler M., Gzip, *Wikipedia*, 2015. [Online]. [Accessed: 21 Oct 2015. Available: https://en.wikipedia.org/wiki/Gzip.

[15] Hobbes T., Base64, 2014. [Online]. Available: Accessed: 20-May-2015. http://en.wikipedia.org/wiki/Base64.

[16] Josefsson S. 2006. The Base16, Base32, and Base64 Data Encodings.

[17] Xu C., Chen Y., and Chiew K. 2010. An Approach to Image Spam Filtering Based on Base64 Encoding and N -Gram Feature Extraction.

in *22nd International Conference on Tools with Artificial Intelligence*. 171-177.

[18] Rawat D., Sahu R., and Puthran Y. 2015. Optimizing the Capacity of QR Code to Store Encrypted Image. *International Journal of Emerging Trends in Engineering Research (IJETER)*. 3(1): 1-4.

[19] Trivedi H. ZXing ('Zebra Crossing'), 2014. [Online]. Available: Accessed: 20 May 2015. http://androidcustomviews.com/new/zxing-zebra-crossing/.

[20] Nurwono K. A. H. and Raymondus K. 2009. Color quick response code for mobile content distribution. *the 7th International Conference*. 267-271.

[21] Grillo A., Lentini A., Querini M., and Italiano G. F. 2010. High Capacity Colored Two Dimensional Codes. in *Proceedings of the International Multiconference on Computer Science and Information Technology*. 5(1): 709-716.