Jurnal
Teknologi

Full Paper

# A RESOURCE-INDEPENDENT METHOD TO DELIVER AUGMENTED REALITY CONTENT

Syazani Suhaifi, Junaidi Abdullah*, Ku Day Chyi

Faculty of Computing and Informatics, Multimedia University Cyberjaya, Malaysia

**Article history**
Received
*3 December 2013*
Received in revised form
*2 July 2014*
Accepted
*25 November 2014*

*Corresponding author
junaidi@mmu.edu.my

**Graphical abstract**

**Abstract**

Marker Based Augmented Reality requires files and resources that have to be loaded from storage such as a local machine or a web server. We proposed a resource-independent method of transmitting AR content by encoding resources such as 3D model files as QR code and using the QR code itself as the marker. The AR system does not need any online and local storage. We processed the content to best fit it into the QR code. Larger contents are then split to multiple QR codes and the data is joined together by the application on the other end.

*Keywords*: Resource independence, augmented reality, QR code, compression



## 1.0 INTRODUCTION

Resource independent AR is not a concept useful to everybody in general. If the user has high speed broadband or are able to drive to the nearest shop to buy AR app CDs and DVDs this concept is totally irrelevant.

However, if we want to deliver AR content for rural areas where it took 2 hours to go to town and the internet connection is using GPRS or worse 56k dial-up modems, resource independence is very important. The use case scenario for AR in these remote locations is mainly in education and medical purposes. Resource independence is also very useful for content that changes every time. No uploading or updating is necessary on the server or the user's computer.

In order to cater for such resource independence needs, we have created a Resource Independent Marker Based Augmented Reality (RIMBAR).

## 2.0 PREVIOUS WORKS

Similar works have been done by [1]. They created a system called In-Place Augmented Reality. An AR marker is created with the image as the texture and model for the application. They also included a 2D elevation map to form the terrain model. AR behaviors can also be encoded in the system created by [1]. Icons representing transportation are included in the marker image and they used the red lines included in the marker as the path the marker will animate on. [1]'s application however did not embed complicated models. They also highlighted an issue of quality in its textures that the qualities of the textures are dependent on the imaging devices used. In another application, [2] created a system that takes hand-drawn sketches, and converts them into 3D model for Augmented Reality. Notations written down on paper will also give the models physical properties. The works of [1] and [2] is a good example on the possibilities of embedding media on paper. Delivery of content in multiple parts has been tried before by [4]. They came up with a system that uses XML data to provide the content location, provides the ability to do joint models with the minor models attached to a major model on another QR code, and if the loaded model has animation, the QR codes can control the animation. However, the application did not embed the content directly inside the QR codes itself and it has to be downloaded. This however inspired us to take their experiments further and created RIMBAR.

Using QR code as an Augmented Reality marker has been executed several times in the past. The system created by [5] uses a QR code to carry the URLs of the model that their system fetches from a server. Similar approach is used by the system proposed by [3] that collaborates with AR using QR code as marker to fetch data and models from a server. The first proof of concept FLARToolkit with QR codes in 2009 by [6]. However, he did not embed the resources in his proof of concept. These past work have hinted on a possibility that content can be embedded inside QR codes. There is no works being done to embed actual 3d model inside a QR code. The closest work is to embed url that link to a 3d model that resides on a server which require internet connection.

## 3.0 METHODOLOGY

### 3.1 3D Pose Estimations

To render the 3D graphics onto the QR code accurately, we need to estimate the 3d position of the QR code. For this purpose we leveraged the existence of the position detection pattern in a QR code. We then used a method derived from [6] to calculate the 3D pose estimation but instead of using square coordinates, we used the transformation matrix of the "markers". Assuming that the markers are on a flat paper and all the position pattern faces the same direction, then the center of the QR code can roughly be obtained by calculating the average of the 3 transformation matrix of the 3 position detection patterns [6]. We however, calculated the transformation independently instead of using all of the results together as done by [6] and [3]. By independently calculating transformation, the system will be able to cope with loss of detection of some position detection patterns. It will also provide flexibility to the system as the system can track multiple QR code formation.

In order to smooth out temporary loss of detection that happens especially when the marker is moving, a timer is used to prevent the 3D object from disappearing upon detection loss of all the position detection pattern. Such temporary disturbance rarely happens for a long duration. Usually within the range of less than 1 or 2 seconds the application will be able to detect the position detection pattern again.

The 3D graphics data is stored temporarily in a buffer after it has been parsed. Therefore, the model is still visible even when the QR part containing the encoded data is covered. The system will only purge the data after a set period of time that we assumed that the user is no longer using the marker.

### 3.2 Embedding 3D scene to QR code

The system accepts VRML file format. After comparing with many different file formats, it is found that the VRML format is the smallest and most suitable format as it has very short text for 3d object representation compared to the other file formats.

We used Primitives Models and Custom Models for our testing. Primitives Models are models that is created using the basic VRML primitives such as Box,Sphere, Cylinder and Cone. The Custom Models are models created using a 3D Modelling Software using manipulation techniques such as extrusion, .etc. There are simple (0 – 10 QR), intermediate (10 to 20QR), and complex(>20 QR) models for testing. Table 1 and Table 2 shows the details for both model. category.

**Table 1** Primitives model and its specification

| Primitives | | | |
|---|---|---|---|
| **Shape** | **Category** | **Character Count** | **QR code required** |
| Box | Simple | 521 | 3 |
| Cone | Simple | 540 | 3 |
| Cylinder | Simple | 535 | 3 |
| Sphere | Simple | 420 | 2 |
| Allprimitives | Intermediate | 3971 | 18 |
| Prim Microphone | Intermediate | 3082 | 14 |
| Prim Car | Intermediate | 2980 | 13 |
| Prim Molecules | Complex | 6627 | 29 |
| Prim Forest | Complex | 8704 | 38 |
| Prim Castle | Complex | 11216 | 49 |

Even the smallest VRML file is quite long and beyond the encoding capacity of one QR code. We employed a maximum safety-cap value of 250 characters per QR code in all our test and measurement to avoid the codes from getting harder to detect. We found out through trial and error that the more complex the QR, the harder it is to be detected.

We are able to embed larger 3d model by splitting such graphics data into multiple QR codes. Large amounts of data can be encoded into multiple QR codes. It is possible, with the proposed system and to encode data into any number of QR codes.

**Table 2** Custom model and its specification

| Custom Models | | | |
|---|---|---|---|
| **Shape** | **Category** | **Character Count** | **QR code required** |
| Custom Icosahedron | Simple | 1135 | 5 |
| Custom Hammer | Simple | 1892 | 8 |
| Custom Extruded Box | Simple | 2280 | 9 |
| Custom Arrow | Intermediate | 2359 | 10 |
| Custom Cap | Intermediate | 3489 | 15 |
| Custom Flashlight | Intermediate | 4423 | 19 |
| Custom Candlestick | Complex | 7058 | 30 |
| Custom Spiral Staircase | Complex | 14424 | 62 |
| Custom Teapot | Complex | 29288 | 127 |

### 3.3 RIMBAR Compression Algorithm

To make the size of the data smaller and fit into less QR code we proposed the RIMBAR Compression Algorithm. The overall process is shown in Figure 1.

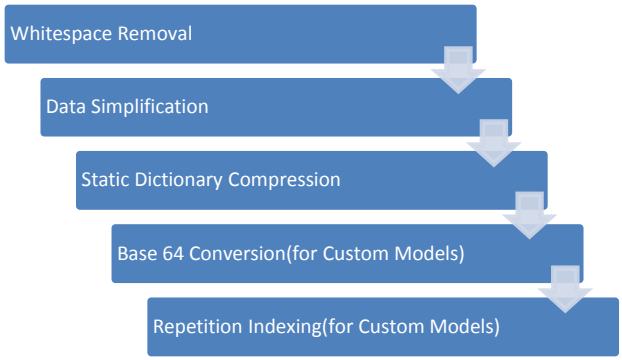


**Figure 1** The RIMBAR Compression

Firstly, the whitespaces in-between the text is removed using the RemoveWhitespace class by [8]. Then the data in the model undergoes a Data Simplification process that only extracts the data needed. Then, Static Dictionary compression method is used to further shorten up the model data. All the VRML nodes are replaced with short letter such as "b1" for "Background" and "c6" for "Coordinate Interpolator". Still however, there are a lot of numerical data that exists in custom made models that has indexed Face Sets for example. To remedy this, we created a dictionary indexed to cater for the repetitive numbers and replace them with alphanumeric representation. Finally, to the numbers that cannot be indexed, we converted the numbers to base64 so that it becomes smaller.

### 3.4 RIMBAR Format

It became evident later on that even compression of models will not be enough to make the models fit into. Therefore, the RIMBAR Format is created based on the premise that all the models will be created specifically only using Primitives. Primitives are used because results of the previous experiments prove that Primitive-based models are smaller. Table 3 below shows the syntax for the RIMBAR format.

**Table 3** The RIMBAR format

| **RIMBAR Format Syntax** |
|---|
| Metadata<br>⏞<br><Part Number><Part Total>] <Primitive Geometry> \| <Position X><Postion Y><PositionZ> \| <Rotation X><Rotation Y><Rotation Z> \| <Scale X><Scale Y><Scale Z> \| <RGB Red><RGB Green><RGB Blue>\| ;<br><br>Example :<br>2 2]B \| 200 50 250 \| 0 0 90 \| 1 1 1 \| 0 255 255 \| ;<br>C \| 10 50 250 \| 0 0 90 \| 1 1 1 \| 0 256 128 \| ; |

### 3.5 RIMBAR2 Compression Algorithm

To compress the files even further, the RIMBAR 2 Compression is applied. RIMBAR 2 is an algorithm that allows the RIMBAR Format to be shrunk to fit more models into the same amount of QR Code. Since RIMBAR Format has its own exporter, the compression will take place during the writing of the RIMBAR Format. For example, RIMBAR 2 Compression will be applied to the Position X,Y, and Z values as it is written.

Firstly, all the numerical instances inside the model such as coordinates, etc. is converted to base 64 format. Then, the color information is converted to the nearest tens and replaced with a character. For example, "190" is represented as "s".

### 3.6 Recovering 3D Scene from QR Code

To recover the data, we have to piece together the jigsaw puzzle that we created using the developer system. Firstly, the QR code is scanned using a QR code reader. We used Logosware's' QR Code Reader as the QR code reader. The code reader then passes the decoded information back to the system. The system will then look for metadata that is embedded together with the QR code. By using the metadata the system will determine where does the current part of data belong in the whole total set of data. If all the parts of the data are present, a parser will then parse

the document. Parsed information values needed for recreating the scene is then passed on to the 3D creator() function to recreate the 3d scenes in the 3D engine. Finally, the 3D engine augmented the model on top of the live video.

### 3.7 Embedding and Recovering Image Textures to and from QR Code

We are able to encode textures into QR codes by transmitting the pixel color values instead of the whole image. The system iterates through the pixels and builds a list of the color of each pixel. This information is then passed to the QR code encoder to encode as QR code. During recovery, the system takes the data and recreates the pixels according to its original color.

## 4.0 RESULTS

### 4.1 3D Pose Estimations

We found that our current method has a very high degree of robustness to occlusion if compared to the works of [3] and [6]. Previous attempts by [3] have very good occlusion immunity. However, when one of the position detection pattern is occluded, the detection is lost as shown in Figure 2.

**Figure 2** Occlusion immunity of QRAR by [3]

Our method have managed to improve further the occlusion immunity and flexibility of the QR code as it is harder to lose all three detection patterns at the same time. If there is at least one position detection pattern remain uncovered, the model will still appear as shown in Figure 3.

**Figure 3** Occlusion immunity of RIMBAR

### 4.2 Embedding 3D Scene to QR Codes

We are able to embed the 3D scenes to QR code. However, it does take at least two QR Code for the simplest of models to fit. If nothing is done with the data, this will not be practical.

### 4.3 RIMBAR Compression Algorithm

Overall, the RIMBAR compression method works best with Primitives Model with an average of 80% reduction for all models regardless of complexity as shown in Figure 4.

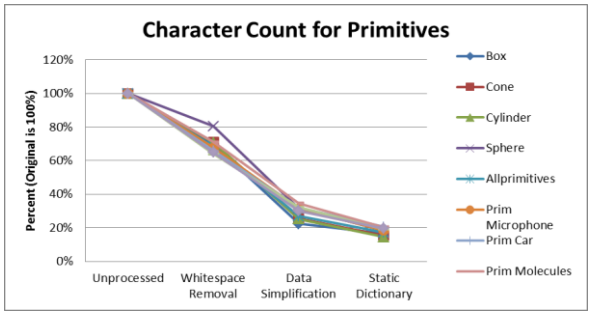


**Figure 4** Character count for primitives

The compression method however is not effective towards custom models. There are reductions towards the Custom Models. However, the reduction percentage is inconsistent and averages at 55% as shown in Figure 5.

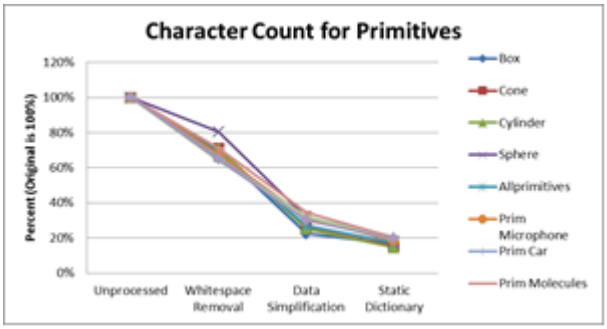


**Figure 5** Character count for primitives

The most effective process is the Data Simplification process, followed by whitespace removal and Static Dictionary for both Primitives Model and Custom Models.

The results of the RIMBAR Compression are then compared side-by-side with the results of compressing the 3D models using text compression techniques. The results are shown in Figure 6 and Figure 7. Both results show that RIMBAR Compression is more effective in compressing the models compared to the other text compression methods. LZ77 is the closest competition for the RIMBAR Algorithm by achieving 73 percent reductions. However, it only works on primitives of a certain complexity. Negative result means that the

outcome of the compression is bigger than the original data.

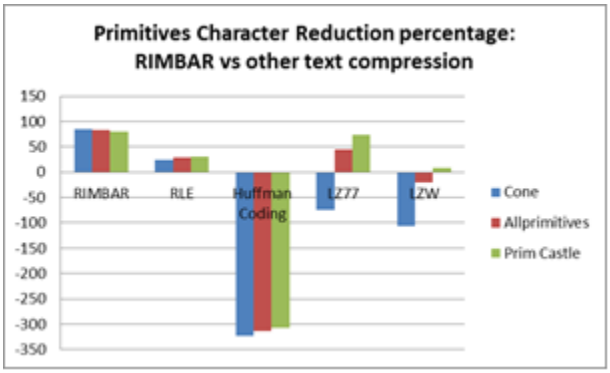


**Figure 6** Comparison of character reduction using RIMBAR algorithm and other text compression methods for primitives

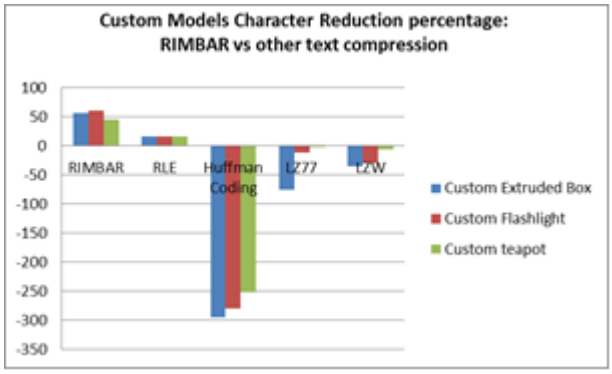


**Figure 7** Comparison of character reduction using RIMBAR algorithm and other text compression methods for custom models

### 4.4 RIMBAR Format

It is found that RIMBAR Format is smaller than both the VRML files and also the files that are compressed using the RIMBAR Compression. Table 4 shows the full results.

**Table 4** Comparison between RIMBAR format, RIMBAR cmpression and raw VRML

| | VRML | RIMBAR Format | RIMBAR Compression |
|---|---|---|---|
| Box | 521 | 35 | 86 |
| Cone | 540 | 35 | 84 |
| Cylinder | 535 | 36 | 78 |
| Sphere | 420 | 37 | 77 |

### 4.5 RIMBAR Format

It is found that RIMBAR Format with RIMBAR 2 Compression is smaller than both the models with RIMBAR Compression and the uncompressed RIMBAR Format. Table 5 shows the full results. In general, the RIMBAR 2 algorithm is able to decrease the character count of the models.

**Table 5** Character count between the RIMBAR format, models after RIMBAR compression, and RIMBAR format with RIMBAR 2 compression

| | RIMBAR Compression | RIMBAR Format | RIMBAR Format + RIMBAR 2 Compression |
|---|---|---|---|
| Box | 86 | 35 | 25 |
| Cone | 84 | 35 | 25 |
| Cylinder | 78 | 36 | 26 |
| Sphere | 77 | 37 | 26 |

### 4.6 Embedding and Recovering Image Textures to and from QR Code

In context of transmitting large images, our current method is not at all practical if compared to the methods used by [1] because 2 QR Codes is required to transmit a very small 6x6 pixel image as shown in Figure 8.

**Figure 8** Two QRs needed for 6x6 pixel image

## 5.0 DISCUSSION

The high occlusion immunity by RIMBAR has its roots in its flexibility in averaging the transformation matrix of multiple markers as individuals instead of rigidly relying on a set of 3 transformation points as done by [6] or by having 4 points of the QR codes as reported by [3] and [5].

From the results of this experiment overall, the best model to use for RIMBAR is model based on Primitives. The best format to use is the RIMBAR Format + RIMBAR 2 Compression. The RIMBAR Format does not allow decimal points as one of the measures of reducing file spaces. As a result of this limitation, there might be minute difference in transformational value from the original file.

The Huffman coding technique might work in other applications however it does not work in the context of RIMBAR. This is because RIMBAR needs to encode the information in String format. The result of Huffman coding is in binary format. Hence, the character count is increased in some cases up to 400% of the original size.

We have such a bad result in transmitting images due to the amount of data that we need to transmit. Currently, we have to send the RGB color information

of each pixel into the QR code. This takes up a lot of space.

This research also illustrates the possibilities of encoding media into printed 2D barcodes which might be very useful for any publishers wishing to deliver interactive content instantly.

## 6.0 FURTHER RESEARCH

Usage of High Capacity Color Barcodes such as Microsoft Tag will largely improve the ability of this system to carry more information without the need to split the data (into multiple barcodes). At the time of writing, there is no Microsoft Tag reader for PC. Microsoft Tag reader is however available for smartphones with Android, iOS, and Symbian. In our attempt to build mobile solution, using Microsoft Tag instead of QR code is very much possible.

## 7.0 CONCLUSION

We proposed an AR system that encodes resources such as 3D scenes to QR code and using the QR code itself as the marker. We found that we are able to fit considerably large amount of scene data. But in order to transmit textures and images, our current method is not feasible and not recommended.

## References

[1] Bergig, N. H. O., J. El-Sana, K. Kedem, M. Billinghurst. 2008. In-Place Augmented Reality. IEEE International Symposium on Mixed and Augmented Reality 2008. 15-18 Sept. 2008. 135-138.

[2] Bergig, N. H. O., J. El-Sana, Mark Billinghurst. 2009. In-Place 3D Sketching for Authoring and Augmenting Mechanical Systems. 8th IEEE International Symposium on Mixed and Augmented Reality 2009 Science and Technology Proceedings. 87-94.

[3] Jian-tung, W., Chia-Nian, S., Hou, T. W., & Fong, C. P. 2010. Design and implementation of augmented reality system collaborating with QR code. International Computer Symposium (ICS), 2010. Tainan, Taiwan. 16-18 Dec. 2010. 414 - 418

[4] Kan, T.-W., &Teng, C.-H. 2010. A framework for multifunctional Augmented Reality based on 2D barcodes. ACM SIGGRAPH 2010 Posters. Los Angeles, California. 25-29 July 2010.

[5] Kan, T.-W., Teng, C.-H., & Chou, W.-S. 2009. Applying QR Code in Augmented Reality Applications. Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry. Yokohama, Japan. 2009.

[6] MakC. 2009. Augmented Reality and QR Codes. [Online]. From: http://makc3d.wordpress.com/2009/10/30/augmented-reality-and-qr-codes/2009.

[7] Adobe. 2012. Stage 3D. [Online]. From: http://www.adobe.com/devnet/flashplayer/stage3d.htm.

[8] G.Skinner. 2007. AS3 String Utils – RemoveWhitespace Class. [Online]. From: http://tinyurl.com/lvsyyeg. [Accessed on 15 May 2012].