# A Resource-Independent Marker-Based Augmented Reality Application

Suhaifi Syazani*, Abdullah Junaidi, and Ku Day Chyi

Multimedia University, Faculty of Information Technology,
Jalan Multimedia 63100 Cyberjaya Selangor Malaysia
{junaidi,dcku,syazani.suhaifi11}@mmu.edu.my

**Abstract.** Creating a marker based Augmented Reality (AR) usually requires a series of files such as marker files and 3d model files. The series of files that Marker Based Augmented Reality requires will limit its capability and reduces its reliability. These files and resources have to be loaded from storage such as a local machine or a web server. We propose a Resource Independent Marker Based Augmented Reality (RIMBAR) 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 marker file or 3d model file. We processed, shortened and convert the content to make it fit into a QR code. Larger contents are then split to multiple markers and the data is joined together at the other end. Currently this system shows potential but further research needs to be done to remove the issues.

**Keywords:** QR Code, Augmented Reality, 3D Model Transmision.

## 1 Introduction

The dependence to the resources that Marker Based Augmented Reality requires will limit its distribution, capability and reliability. If the app needs to download its resource from the internet, the application performance will then be dependent on the speed of the user's internet connection. One solution is to install them on the machine running it. But, if the user wants to use many AR applications, it will be a cumbersome to install each one of them on their machine. 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. 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 frequently. A magazine using a resource independent AR system only needs to generate the QR codes and print it on the magazine. No uploading or updating is necessary on their server. We

* Corresponding author.

have developed a system that is able to distribute AR content via the medium of QR code. The system also has the capability to split the into multiple QR codes if necessary.

## 2 Previous Works

Previous attempts on resource independence have been done by [1] in 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. The system created by [1] also allows embedding of behaviors. 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. Their application however did not embed complicated models as they have to rely to height maps for 3D information. They also highlighted an issue of quality in its textures, that the qualities of the textures are dependant on the imaging devices used. Another attempt is done by [2] using a system that recognizes hand-drawn sketches, and converts them into 3D model for Augmented Reality. Notations written down on paper will also give the models physical properties. Both of the works are very close in achieving resource independance.

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. However, the application did not embed the content directly inside the QR codes itself and it has to be downloaded.

Using QR code as an Augmented Reality marker has been executed several times in the past. A system that uses a QR code to carry the URLs of the model that their system fetches from a server was created by [5] and [3]. The first proof of concept FLARToolkit with QR codes was created by [6] in 2009. 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. Recently, [7] has created a system that can dynamically generate QR code using an ink senstitive to variations of temperature. The ability to change QR code on-the-fly will give our system greater practicality as no QR code needs to be printed.

## 3 Resource Independent Marker Based Augmented Reality (RIMBAR)

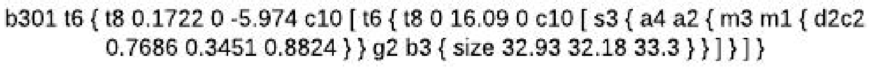
The system is divided into two parts which is the deployment/developer system and the user system.The developer system processes the models and textures. It will process, inspect, and perform necessary conversions. The developer system also determines how many part should the data be split into and to how many QR Codes. The user system contains the FLARToolkit engine, the QR Code decoder and Papervision3D as the 3D engine. The user system will read the QR code, parse it, recreate the 3D scene and augment it to the live feed. The processes involved in Developer Module and Client Module are listed in Table 1.

**Table 1.** The processes involved in the RIMBAR module

| *Developer Module* | *Client Module* |
|---|---|
| 1. Loading the VRML file | 1. Read the QR code |
| 2. Parsing the VRML file | 2. Parse the QR code data |
| 3. Compress the VRML text | 3. Recreate the 3D scene |
| 4. Split the content to multiple QR codes if necessary. | 4. Track the QR code as an AR marker |
| 5. Generate the final QR code containing the model | 5. Overlay 3D model on top of the QR code |

## 3.1 Compressing the VRML File

The file on its own is quite long and beyond the encoding capacity of one QR code. Therefore the 3d formats undergo a process of shortening. Firstly, the whitespaces inside the strings are removed. The empty-spaces are counted as one character by the QR code encoder. Unnecessary empty-spaces will decrease considerably the amount of data encodable. Then, the words inside the strings are replaced with shortened codes. The system will check series of words inside an XML file, and if the words are found inside the string, it will then be replaced with a shortened code. Figure 1 below shows a sample of VRML string after shortening.



```
b301 t6 { t8 0.1722 0 -5.974 c10 [ t6 { t8 0 16.09 0 c10 [ s3 { a4 a2 { m3 m1 { d2c2
0.7686 0.3451 0.8824 } } g2 b3 { size 32.93 32.18 33.3 } } ] } ] }
```

**Fig. 1.** Sample of VRML string after shortening

**Table 2.** Result of the shortening process of Box.wrl model

| | *Character Count* | *QR code required to fit the data* |
|---|---|---|
| Raw unedited VRML | 466 | 3 |
| Whitespace removal | 229 | 1 |
| Dictionary replacement | 135 | 1 |
| Final Outcome | 135 | 1 |
| Reduction of Character Count | 71.03% | |
| Reduction of QR | 67% | |

However, the shortening process as we found out later does not totally solve the problem. Although the results are promising as shown in Table 1, it is not enough for our goal to fit in complex models. If one box is 135 characters long, one QR code will be full if we put two boxes. That is not practical. Both formats contain a plethora of information. However, not all of them are needed for Papervision3D to recreate the scene. Therefore, we created our own proprietary format. This particular format carries the exact information that the Papervision3D needs to reconstruct the scene. We called this format the QRF format. Figure 2 shows a sample of the QRF format.

```
MDL,g3  b3,m1 d2c2 0.7686 0.3451 0.8824| ,t16 0.1722 0 -5.974| ,s11 32.93 32.18 33.3|
```

**Fig. 2.** The same VRML as used in Figure 1 after conversion to QRF

The use of QRF allows for a significant decrease in the size of the VRML string as it only contains the information needed to reconstruct the scene. Using the QRF format allows more complex scenes to be embedded into QR code.

## 3.2 Multiple QR Codes

There is a need to split contents as we have set the limit of 250 characters per QR code. Indeed the capacity of QR can take much more than 250. But, as the numbers increase, we found that it becomes harder for the QR code reader to read.

We are able to embed larger 3d scene by splitting such information into multiple QR codes. Part information metadata is attached to the split string. Whenever the program decodes the QR codes, the metadata will inform the system the number of parts does the system need to decode and recombine before processing the final output. Once all the parts are present, a parser will parse the codes and recreate the 3d scene. The process is shown in Figure 3.

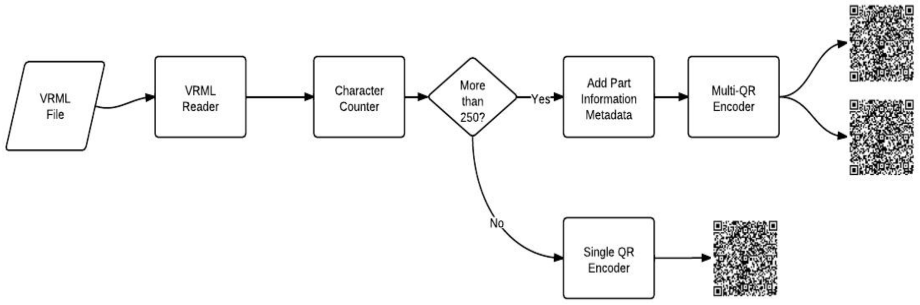


**Fig. 3.** Process of QR splitting

By splitting the information, large amounts of data can be encoded into multiple QR codes. It is possible, with the help of this system and metadata to encode data into any number of QR codes. However, too many QR codes part will be very cumbersome to scan.

## 3.3 Recovery of 3D Models

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' 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 the part that the current data belong to 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 3dcreator() function to recreate the 3d scenes in Papervision 3D. Finally, Papervision3D augmented the model on top of the live video.

### 3.4 Embedding and Recovery of Image Textures into 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 table or 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. However we found out that in context of transmitting large images, this 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 image as shown in Figure 4.

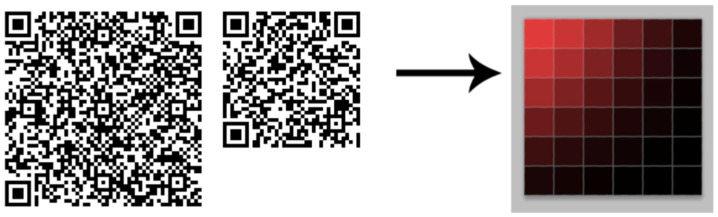


**Fig. 4.** Two QR codes that carries the pixel values of a 6 by 6 pixel image

### 3.5 3D Pose Estimation

To estimate the position of the 3D graphics, we employed a normal marker tracking approach with a twist to make it robust. Firstly, we generated a marker using the position detection pattern on the QR code. In order to place the 3D object at the center of the QR we used the transformation matrix from all 3 position detection pattern. Similar to the works by [6], we assume that the markers are on a flat paper and the position of the 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. Instead of selective averaging we averaged all the components of the transformation matrix. Overall, the method works with one QR code, and continues to work with multiple QR codes. If there is 4 QR codes, averaging the transformation matrix will result in the marker being in the middle of the QR code formation. Our approach provides a new level of robustness to occlusion as shown in Figure 5 below.

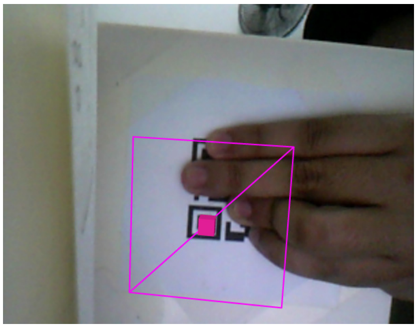


**Fig. 5.** RIMBAR still works with almost all the QR code covered

By using our method, almost the whole QR code can be covered. If there is at least one position detection pattern remain uncovered, the model will still appear. 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 loss detection of all the position detection pattern. Such temporary disturbance rarely happens for a long time. Usually within the range of less than 1 or 2 seconds before the application can detect the position detection pattern again.

## 4 Future Research

Coping with temporary occlusion with Kalman filter as demonstrated by [3] will be a useful feature for this system and we intend to implement it in our second version of the system. 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.

This system if it can be used as mobile apps will prove to be very useful as mobile phones and tablets are easier to carry around especially when the user goes to a remote area. The application is already built using Flash to ease the transition to mobile apps by using AIR for Android and AIR for iOS.

## 5 Conclusion

We proposed a Resource Independent Marker Based Augmented Reality (RIMBAR) by encoding resources such as 3D model files to QR code and using the QR code itself as the marker. We found that this method can transmit models in an acceptable manner. But in order to transmit textures and images, it is not feasible.

## References

1. Hagbi, N., Bergig, O., El-Sana, J., Kedem, K., Billinghurst, M.: In-Place Augmented Reality. In: IEEE International Symposium on Mixed and Augmented Reality (2008)
2. Bergig, O., Hagbi, N., El-Sana, J., Billinghurst, M.: In-Place 3D Sketching for Authoring and Augmenting Mechanical Systems. In: EEE International Symposium on Mixed and Augmented Reality 2009 Science and Technology Proceedings, ISMAR (2009)
3. Jian-tung, W.C.-N., Shyi Hou, T.W., Fong, C.P.: Design and implementation of augmented reality system collaborating with QR code. In: 2010 International Computer Symposium (ICS), pp. 414–418 (2010)
4. Kan, T.-W., Teng, C.-H.: A framework for multifunctional Augmented Reality based on 2D barcodes. In: ACM SIGGRAPH 2010 Posters, p. 1 (2010)
5. Kan, T.-W., Teng, C.-H., Chou, W.-S.: Applying QR code in augmented reality applications. In: Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry, pp. 253–257. ACM
6. MakC:Augmented reality and QR codes, http://makc3d.wordpress.com/2009/10/30/augmented-reality-and-qr-codes/7
7. Peiris, R.L., Fernando, O.N.N., Bee, C.S., Cheok, A.D., Ganesan, A.G., Kumarasinghe, P.: dMarkers: ubiquitous dynamic makers for augmented reality. In: Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry, pp. 217–224. ACM