A Tile Based Colour Picture with Hidden QR Code for Augmented Reality and Beyond

H. Tran, H. Le, M. Nguyen, W. Yan.
Auckland University of Technology
No. 2-14, Wakefiled Street, Auckland, 1010 New Zealand

ABSTRACT

Most existing Augmented Reality (AR) applications use either template (picture) markers or bar-code markers to overlay computer-generated graphics on the real world surfaces. The use of template markers is computationally expensive and unreliable. On the other hand, bar-code markers display only black and white blocks; thus, they look uninteresting and uninformative. In this short paper, we describe a new way to optically hide a QR code inside a tile based colour picture. Each AR marker is built from hundreds of small tiles (just like tiling a bathroom), and the unique gaps between the tiles are used to determine the elements of the hidden QR Code. This novel type of AR marker presents not only a realistic-looking colour picture but also contains self-Correcting information (stored in QR code). In this article, we demonstrate that this tile based colour picture with hidden QR code is relatively robust under various conditions and scaling. We believe many nowadays' AR challenges could be solved with this type of marker. AR-enabled medias could then be easily generated. For instance, it would be capable of storing and displaying virtual figures of an entire book or magazine. Thus, it provides a promising AR approach to be used in many different AR applications; and beyond, it may even replace the barcodes and QR Codes in some cases.

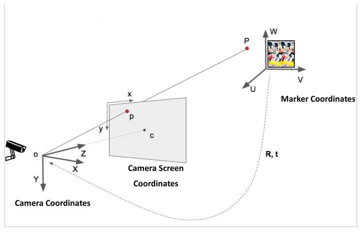
KEYWORDS

Augmented Reality

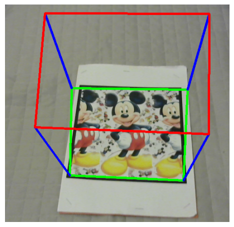
ACM Reference format:
H. Tran, H. Le, M. Nguyen, W. Yan. . 2017. A Tile Based Colour Picture with Hidden QR Code for Augmented Reality and Beyond . In *Proceedings of 23rd ACM Symposium on Virtual Reality Software and Technology, Gothenburg, Sweden, November 2017 (VRST 2017),* 4 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION AND BACKGROUNDS

For decades, researchers have been trying to create intuitive virtual environments by blending reality and virtual reality to let general users interact with the digital domain as easily as with the real world. The result is "augmented reality" (AR) whereby virtual objects seamlessly superimpose upon a real environment in three dimensions and in real time. AR is widely used in medical visualisation, manufacturing, maintenance and repair, path planning, entertainment, and military applications [1, 2]. One of the earliest AR interfaces was created by Sutherland over 50 years ago [19]. For years, many researchers have tried to create AR applications for users to interact easily with both the digital and the real world.



(a) Camera Pose Relationship.

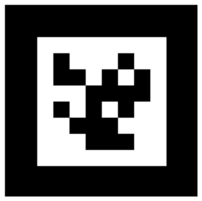


(b) Camera Pose 3D Sample.

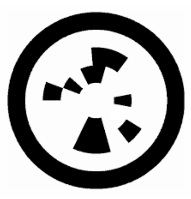
Figure 1: Marker detection and computer graphic rendering.

Creating an effective AR experience requires the use of various tools such as graphics rendering tools, tracking and registration tools, and various display or interaction techniques. There are many long-term problems of AR such as visuality, processing complexity, and the number of allowed virtual items. One central problem of creating AR applications is the determination of computer-generated objects and their position and orientation so that they could be aligned accurately with the physical objects in the real world. In many existing applications, graphical content is often put on pre-defined markers as they provide a convenient way for detecting the encoded contents and calculating the camera poses. For example, an image tag system such as BazAR [7, 8] uses natural (colour) picture as markers. As seen in Fig.1(a), the camera position relative to the marker is calculated to blend the virtual information into the real world environment. The relationship is called pinhole camera model [18] or pose estimation in computer vision.

(a) Template Marker



(b) Bar-code Marker



(c) Circular Marker

Figure 2: Popular types of AR tags.

For robust and unambiguous applications [16], black and white markers with thick borders are often used [5, 10, 15] and these include template markers (Fig 2(a)), barcode markers (Fig 2(b)), and circular marker (Fig 2(c)). These markers are made up of a white/light coloured padding, surrounded by a thick black/dark coloured border and a high contrast pattern of either a template figure, a square or a circular 2D bar code. The pattern is what makes these markers unique. The black border of markers is recognised, tracked and used to calculate the position in 3D space. Some other newly invented fiducial marker designs combining payload with the structure of the tag such as [3] [4], are still collections of black and white squares or dots.

Both template and bar-code tags have their pros and cons. Template tag may contain some meaningful picture of the object it is presenting; such as a flying eager in Fig 2(a). As such, one could use feature matching techniques for identifying template markers (by comparing them with marker templates stored in a database). However, such a system must be trained sufficiently for proper template matching but nonetheless template recognition could be unreliable due to the undesired similarity between template markers. [20]. Consequently, in such a system, the number of different templates need to be small for good results.

On the other hand, bar-code and circular markers are encoded in “0” or “1” by arranging the marker region into many black and white bars. Examples are CyberCode [15], Bokode [12], and AprilTag [13]. Decoding techniques are used to decrypt the encoded data. It is relatively easy to detect and recognise bar-code using various feature detection technologies [9]. However, these markers display no useful information for the users. It is thus difficult to know which marker represents which virtual object just by looking at the black and white pattern themselves.

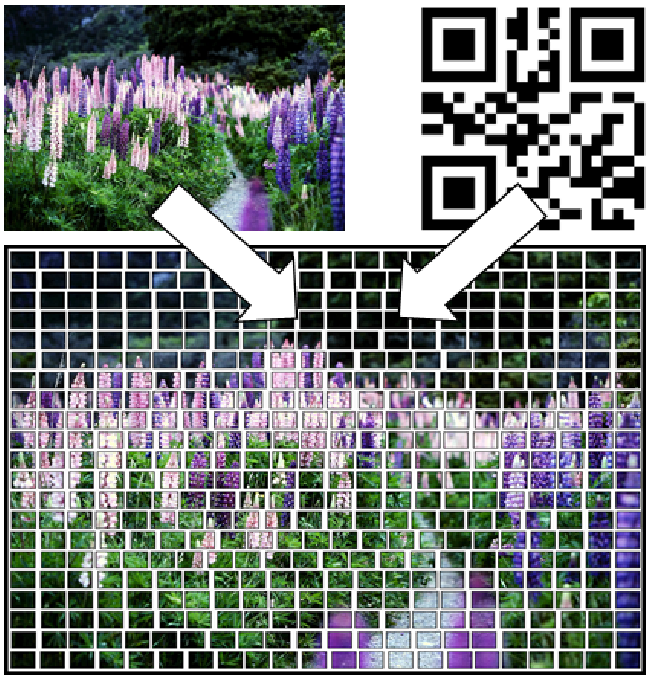


Figure 3: Design of the proposed AR marker.

In this paper, we present a way to optically hide a QR code [17] inside a tile based colour picture. We choose QR Code due to its popularity; moreover, it is self-error corrected and orientation detectable [11]. The proposed AR marker is built from hundreds of small tiles (just like tiling a bathroom) as seen in Fig. 3, and the gaps between the tiles are used to determine the elements of the hidden QR Code. The QR code is then used to determine the graphics to be rendered in the image of an AR application.

2 DESIGN OF A TILE-BASED AR MARKER

Our idea is motivated by the 3D illusion created by looking at a tiled wall in a kitchen or a bathroom. If both eyes are paralleled and virtually looked at a point behind that wall, the tiles appear to float at different layers of depth. The effect is very similar to autostereogram or magic eye pictures described in [21]. The various depths of individual tiles are, in fact, created by the irregular gaps between the tiles. So, if we consider a QR code an $M \times M$ tiled wall and black and white dots are simply many tiles lying at different depth levels; we can build a wall that optically hides a QR code.

Therefore, we design a tile-based AR marker which displays not only a realistic-looking image but also contains numeric information encoded by a QR Code. The tiles are all having the same size, but gaps between them are different. However, for a row of tiles, there are only two sizes of gaps: larger and smaller for binary values of ‘1’ and ‘0’ respectively. The marker is thus pictorial and yet robust enough to be detected under various lighting conditions. In other words, our new AR marker includes the advantageous features of both template and bar-code tags. Moreover, gaps are small compared to tiles; on average, 80-90% of the original picture is retained.

The basic design of our marker is shown in Fig. 3. Only two components are needed for the creation: a colour picture and a QR code. QR codes are now easily obtained by either an online tool such as www.qr-code-generator.com or a public library such as QRcode python library at pypi.python.org/pypi/qrcode.

Assume that we have a QR code with dimension $M \times M$, to be mapped on a colour picture of dimension $W \times H$ pixels. In order to build a tiled walled, that neatly fit in the image, we need to have $(M + 1) \times M$ rectangular tiles. Each tile should have the same size of w_t, h_t pixels. We firstly calculate the components of the vertical cap (g_y), as follow:

$$g_y = \frac{H - M \times h_t}{M} \tag{1}$$

The horizontal big gap $g_x b$ for black QR dot and horizontal small gap $g_x s$ for white QR dot are calculated differently for each horizontal line i^{th} of the QR code. Assume that a QR scan-line i has a black dots and b white dots: $a + b = M$, and the big gap $g_x b$ is n times the small gap $g_x s$. They are calculated as follow:

$$g_x s(i) = \frac{W - (M + 1) \times w_t}{n + 1} \tag{2}$$

$$g_x b(i) = n \times g_x s(i) \tag{3}$$

When all the components (tiles and gaps) are defined, they can be placed accordingly on top of the provided picture. Each tile is a transparent glass with a black frame; the gaps are filled with white cement. After that, a black border is placed with width = 1% of the image on top. This rectangular border is used for the marker recognition, detection, and segmentation (the quadrilateral property of the squares can be used to detect their four straight

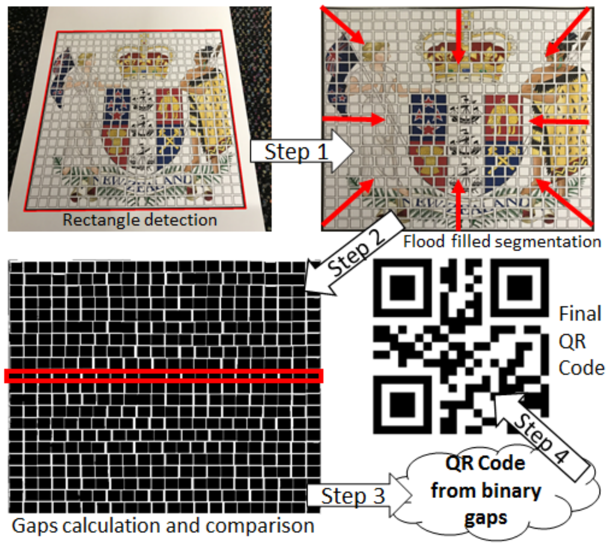


Figure 4: Steps of detection and decryption process.

lines and four corners). The final result is what shown in Fig. 3: a tiled picture of a flower garden, that also hides the QR code storing an "aut" word. Another example is shown in Fig. 4-top-left, the marker was printed on a sheet of paper. This marker will be used to demonstrate the next section of Marker Detection and Decryption.

3 DETECTION AND DECRYPTION

The detection and decryption of aforementioned AR Marker are demonstrated in Fig. 4. The procedure includes four steps: (1) rectangular marker detection, (2) gap region segmentation, (3) binary gap classification, and (4) QR Code reconstruction from gapfis distances.

3.1 Rectangular marker detection

This is a common AR-related problem which has been solved effectively by Contour Approximation Method [6]. First, we find closed contours on the input image and unwarp the image inside it to a square shape. A contour detection is applied on the picture; the contour traces the perimeter of the pattern polygons that have the four corner characteristic. The steps to detect black-border markers are outlined below:

- Convert the input image from RGB to greyscale
- Perform an adaptive binary thresholding method.
- Detect contours in the image. If there are four vertices in the contour, it should be a quadrilateral.
- Apply Perspective Transform [14] to retrieve the internal image of the tag (Fig. 4-top-left). Once an image's border is detected, we can obtain the internal image for further decryption.

3.2 Gap Region Segmentation

The internal tiled image is detected and resize to a suitable dimension (1024×1024 pixels in this case). With the knowledge of the white colour gaps between tiles, and each tile has a thin black frame; a simple flood filling algorithm can be applied. This is also called seed fill algorithm, that determines the area connected to a given node in a multi-dimensional array. Shadows and lights can change the appearance of the photo; for a robust segmentation, we set eight seed points at the corners and boundaries of the image, as seen in Fig. 4-top-right. At each seed point, the below flood fill algorithm applied:

```
def floodFill(x, y, fillColor, interiorColor):
 getPixel(x,y,colour)
 if colour is similar to interiorColor:
  setPixel(x, y, fillColour)
  floodFill(x+1, y, fillColor, interiorColor)
  floodFill(x−1, y, fillColor, interiorColor)
  floodFill(x, y+1, fillColor, interiorColor)
  floodFill(x, y−1, fillColor, interiorColor)
```

3.3 Binary Gap Classification

Flood fill segmentation can separate gaps and tiles and a typical result is shown in Fig. 4-bottom-left. We do not expect all the gaps are detected due to many constraints such as noises and lighting condition. However, there is an assumption that a majority of the gaps are segmented. The tiles can be reconstructed to find the marker orientation. All the components described in Sec. 2: the dimension $M \times M$ of QR Code, the size of each tile w_t, h_t pixels, the vertical cap (g_y), the horizontal big gap $g_x b$ and horizontal small gap $g_x s$; can also be estimated statistically.

After segmentation, the result is similar to what shown in Fig. 4-bottom-left, every horizontal and vertical scan line are analysed to create collections of:

- Number of horizontal and vertical black and white gaps.
- Sizes of horizontal and vertical black and white gaps.

If the statistics mode value (the data value that appears most often) of horizontal white gaps for all scan lines is one value higher than the mode value of vertical white gaps for all scan lines, then the marker is at correct orientation or up-side-down orientation. If not, a rotation needs to apply on the image.

The mode value of all horizontal black gaps is the estimate of the width of each tile w_t. Thus, the mode value of all vertical black gaps is the height h_t of each tile. Vertical gap g_y is found from the statistics mode of all vertical white gaps.

Knowing the height of each tile, each line of QR code (shown as the red region in Fig. 4-bottom-left) can be processed. Gap sizes are harvested horizontally, the known width w_t of each tile is used to control the quality to make sure no gaps are missing. From the collection of tile gaps, they can be separated into two group: big with size $g_x b$ and small gaps with size $g_x s$. Big gaps represent black dots, and small gaps represent white dots of the QR Code.

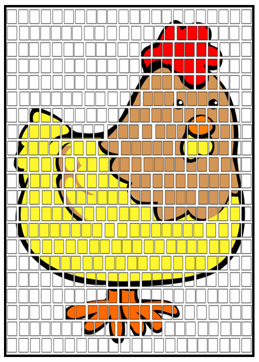
3.4 QR Code Reconstruction and Decryption

Gaps are detected correctly help reconstruct the full QR code as shown in Fig. 4-bottom-right. The QR Code can now be used to extract hidden data and also the orientation of the tag (correct or upside-down orientation). There are several libraries that are capable for the decode such as ZBar[1], ZXing[2], Quiec[3], or Libdecodeqr[4].

Each of those libraries can read an image frame, automatically detect the boundaries of the QR code, decode it to a meaningful text, and also determine the orientation of the code. In other words, the four corners of the QR Code are defined, it helps specify the orientation (direction) of the AR marker. Such information is crucial for the rendering of computer graphics on the marker.

4 INITIAL RESULTS AND LIMITATIONS

Two other simple pictorial AR markers are shown in Fig. 5. They are cartoon images, both hide the same QR Code of the text "aut". These markers are printed on papers, and we use a smart-phone to acquire their photos. The markers are placed at various distances (0.5, 1.0, 1.5, 2.0 metres away) from the camera under indoor fluorescent lighting condition. The detection and decryption processes are carried out.


(a) A chicken AR Marker

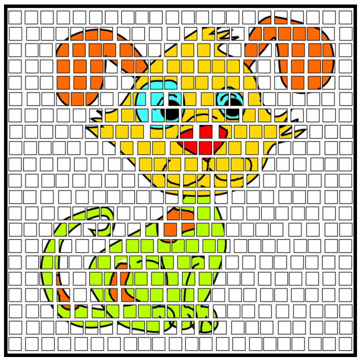

(b) A dog AR Marker

Figure 5: Examples of other AR Markers.

After some initial analysis of the results, we have the following conclusions. The rectangular marker detection and segmentation is relatively efficient and robust. In most cases, it can quickly separate the marker out from the background. The decryption, on the other hand, is not as accurate as we expected, especially when the marker is located at a distance (more than 1.5 metres away). In these cases, the small gaps appear too thin, which stop the flood filling algorithms to spread towards the centre of the image. Light condition and camera resolution also affect the results. With good lightings, high-resolution cameras, and the marker is not too far away; the QR Code decryption is working efficiently. What is more, we found that the higher the size ratio between large gaps and small gaps, the more accurate the detection was.

5 CONCLUSION AND FUTURE WORKS

This short paper describes a new presentation of pictorial AR marker that can optically hide a QR code and can be effectively used in many AR applications. This pictorial AR marker itself contains a tile-based colour image concealing a self-correcting QR code. The codes are optically encoded by the small gaps between individual tiles; thus, most of the picture features depicted on the card are reserved. Blank and monotone regions of the original image do not affect the detection and decryption results. This tag can also be used in collectable trading cards to turn a traditional game environment into an interactive AR experience. From some limited experiments, these proposed tags are relatively robust if used with high-resolution cameras (this assumption is not too rare for today's technology). Therefore, this design could be a promising approach for use in applications where a traditional barcode marker would distract from the content presented.

At this stage, we only encode QR Codes busing horizontal gaps. However, this design can, in fact, allow the users to use vertical tile gaps. In the future, we will investigate this direction, to build an AR marker that encodes two QR Codes, vertically and horizontally.

[1]https://github.com/ZBar/ZBar
[2]https://github.com/zxing/zxing
[3]https://github.com/dlbeer/quirc
[4]https://github.com/josephholsten/libdecodeqr

REFERENCES

[1] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. 2001. Recent advances in augmented reality. *Computer Graphics and Applications, IEEE* 21, 6 (2001), 34–47.

[2] Ronald T Azuma. 1997. A survey of augmented reality. *Presence: Teleoperators and virtual environments* 6, 4 (1997), 355–385.

[3] Filippo Bergamasco, Andrea Albarelli, Luca Cosmo, Emanuele Rodola, and Andrea Torsello. 2016. An Accurate and Robust Artificial Marker based on Cyclic Codes. (2016).

[4] Filippo Bergamasco, Andrea Albarelli, and Andrea Torsello. 2013. Pi-Tag: a fast image-space marker design based on projective invariants. *Machine vision and applications* 24, 6 (2013), 1295–1310.

[5] Tai-Wei Kan, Chin-Hung Teng, and Wen-Shou Chou. 2009. Applying QR code in augmented reality applications. In *Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry*. ACM, 253–257.

[6] Jin-Hun Kim. 1998. Contour approximation method for representing a contour of an object. (June 30 1998). US Patent 5,774,595.

[7] Vincent Lepetit and Pascal Fua. 2006. Keypoint recognition using randomized trees. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28, 9 (2006), 1465–1479.

[8] Vincent Lepetit, Pascal Lagger, and Pascal Fua. 2005. Randomized trees for real-time keypoint recognition. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 2. IEEE, 775–781.

[9] Tony Lindeberg. 1998. Feature detection with automatic scale selection. *International journal of computer vision* 30, 2 (1998), 79–116.

[10] Tsung-Yu Liu, Tan-Hsu Tan, and Yu-Ling Chu. 2010. QR code and augmented reality-supported mobile English learning system. In *Mobile multimedia processing*. Springer, 37–52.

[11] Yue Liu, Ju Yang, and Mingjun Liu. 2008. Recognition of QR Code with mobile phones. In *Control and Decision Conference, 2008. CCDC 2008. Chinese*. IEEE, 203–206.

[12] Ankit Mohan, Grace Woo, Shinsaku Hiura, Quinn Smithwick, and Ramesh Raskar. 2009. Bokode: imperceptible visual tags for camera based interaction from a distance. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 98.

[13] Edwin Olson. 2011. AprilTag: A robust and flexible visual fiducial system. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 3400–3407.

[14] Learning OpenCV. 2008. Computer vision with the OpenCV library. *GaryBradski & Adrian Kaebler-OfiReilly* (2008).

[15] Jun Rekimoto and Yuji Ayatsuka. 2000. CyberCode: designing augmented reality environments with visual tags. In *Proceedings of DARE 2000 on Designing augmented reality environments*. ACM, 1–10.

[16] Sanni Siltanen. 2012. *Theory and applications of marker-based augmented reality.*

[17] Tan Jin Soon. 2008. QR code. *Synthesis Journal* 2008 (2008), 59–78.

[18] Peter Sturm. 2014. Pinhole camera model. In *Computer Vision*. Springer, 610–613.

[19] Ivan E Sutherland. 1965. The ultimate display. *Multimedia: From Wagner to virtual reality* (1965).

[20] Antti Tikanmäki and Juha Röning. 2011. Markers–toward general purpose information representation. In *IROS2011 workshop: knowledge representation for autonomous robots*.

[21] Christopher W Tyler and Maureen B Clarke. 1990. Autostereogram. In *SC-DL tentative*. International Society for Optics and Photonics, 182–197.