2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

# StereoTag: A Novel Stereogram-Marker-Based Approach for Augmented Reality

Minh Nguyen, Albert (Wai) Yeap
School of Engineering, Computer and Mathematical Sciences
Auckland University of Technology
Auckland, New Zealand
Email: minh.nguyen@aut.ac.nz

***Abstract*—Augmented Reality (AR) is an active and exciting topic aiming to create intuitive computer interface by blending reality and virtual reality. One challenge of AR is to align virtual data with the environment. Typically, one uses a marker-based approach such as a thick-bordered black and white 2D marker which allows one to recover the relative pose (location and orientation) of a camera in real time. However, bar-code markers do not contain any intuitive visual meaning, and they thus look uninteresting and uninformative. We propose a new type of marker, referred to as a StereoTag, which embeds a meaningful stereogram image hiding 3D coded/decoded information. From experiments conducted, our StereoTag is found to be relatively robust under various conditions and thus could be widely used in future AR applications.**

## I. Introduction and Backgrounds

### A. Augmented Reality and Its Common Marker Types

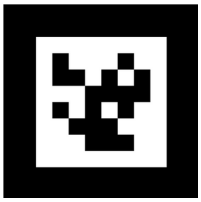
For decades, researchers have been trying to create intuitive virtual environments by blending reality and virtual reality to let general users interact with the digital domain as easily as with the real world. The result is "augmented reality" (AR) whereby virtual objects seamlessly superimpose upon a real environment in three dimensions and in real time, thereby allowing users to interact with the digital contents as easily as with the actual objects. 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 [3].

Creating an effective AR experience requires the use of various tools such as graphics rendering tools (for creating the virtual content), tracking and registration tools (for aligning the real and virtual views), and various display or interaction techniques. However, one central problem of AR applications is determining computer-generated object and its position and orientation to align them accurately with physical objects. Put simply, how does the system know what and where to place the graphic overlay?

At the current stage, graphical content is often placed on pre-defined markers as they provide a convenient way for detecting the encoded contents and calculating the camera poses. An image marker system such as BazAR [4], [5] that uses natural (colour) picture as markers. It works based on computer vision and feature detection techniques. As manual image registration are required, it is suitable for augmented reality applications with a smaller dataset such as children's Magic books [6].

(a) Augmented Reality application running on a smart-phone

(b) Template Marker



(c) Bar-code Marker

(d) Circular Marker

Fig. 1. Examples of Augmented Reality application and Markers.

For robust and unambiguous applications [7], black and white markers with thick borders are more frequently used [8], [9], [10]. There are a few common types of such markers: "template markers" (Fig 1(b)), "bar-code markers" (Fig 1(c)), and "circular markers" (Fig 1(d)) [7]. 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 are recognised, tracked and is used to calculate the position in 3D space. There also exists other fiducial marker designs combining payload with the structure of the tag such as [11], [12]. However, all of them still only hold black and white "random" patterns.

There are many AR frameworks developed to utilise one of

978-1-5090-0357-0/15/$31.00 

the three markers above based patterns, such as the ARToolkit from *www.artoolkit.org*. In an AR application, an image of the real environment is captured using a camera, then search through each image is performed to identify any square shapes (probably markers). Secondly, in case a square is identified, then the application uses some algorithms to calculate the distance and orientation of the camera about the marker square detected. The third step is marker recognition, the symbol inside of the marker is scanned to read bar-code, or matched with templates in memory, to determine which virtual model to be displayed.As the pose of the camera is estimated, the computer graphics model is rendered using the same pose. It is placed as a top layer in the real environment and fixed to the square marker. Lastly, the final output is displayed to the user, who observes graphics overlaid on the real environment.

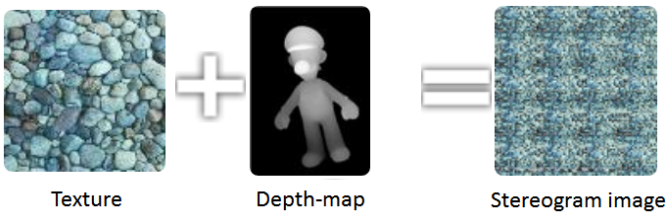
We believe that the step of identifying/recognising markers can be improved due to the following problems:

- Template marker may contain some meaningful picture of the object it is presenting; such as a flying eager in Fig 1(b). Feature matching techniques are used for identifying template markers (by comparing them with marker templates stored in a database). Therefore, it must be trained thoroughly for proper template matching. Template recognition is sometimes unreliable due to the undesired similarity between template markers. [13]. Thus, the number of different templates should be small for good results.
- Bar-code and circular markers are coded in "0" or "1" by dividing the marker region into multiple black and white spaces (bar-codes). Decoding techniques are used to decrypt the encoded data. It is relatively easy to detect and recognise bar-code using computer vision technologies. However, these markers contain meaningless information to the users. It is very hard for general people to know which marker represents which virtual object just by simply looking at the black and white pattern themselves (Fig 1(c)).

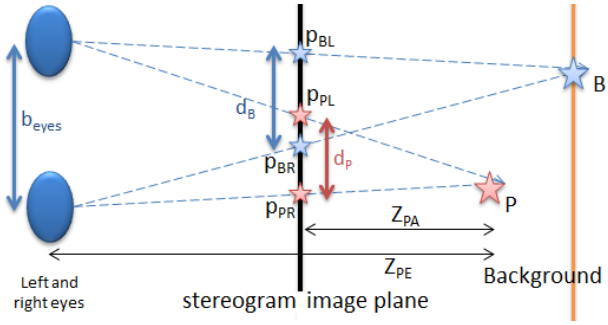
### *B. Stereogram and its Unique Properties*

Stereogram or autostereogram, also known as Magic Eye picture [14], is a two-dimensional (2D) flat image. When viewed it in a particular way (cross-viewing or parallel-viewing); a hidden floating object will appear in three-dimensional (3D). A stereogram picture is generated by repeating patterns many times to present a range of 3D depth forms within certain constraints [15]. The distance between repeating patterns defines the depth of a virtual point perceived by the viewers.

The principle behind 3D illusion perception from a flat stereogram is shown in Fig. 2(b). To perceive the 3D scene from an image of this type, each eye must "see two different points, or focus on a point behind the picture surface. Because the image contains repeating patterns, the brain is forced to think that the two points are the same. It then tricks the brain to perceive this point at a different depth (usually behind) rather than the actual picture surface.




(a) A autostereogram encodes both depth and texture information




(b) Principles behind perceiving a stereogram

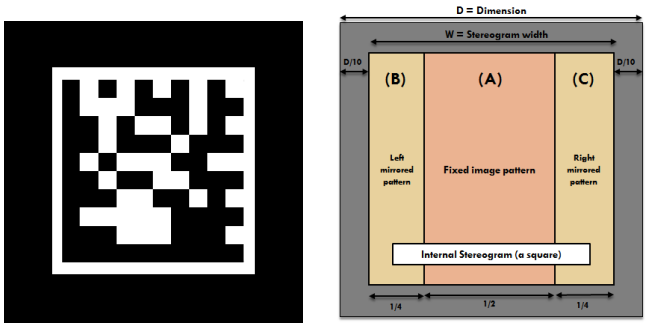
Fig. 2. Principle of Stereogram Images and How to view it.

It is not difficult to generate such stereograms, given a 3D profile $Z(x, y)$ sampled on a grid of $M \times N$, a strip of chosen image pattern $P(i, j)$ of sampled size $M \times N$; the detailed process can be found in [16]. There are also many free tools offered to generate stereogram from supplied depth and image pattern on the Internet, for instance, *http://www.easystereogrambuilder.com*. If image patterns are chosen carefully, we may get interestingly looking results. On the other hand, retrieving 3D information from stereogram images is an Ill-posed Inverse Optical Problem due to the random nature of matching similarity and the structural ambiguity of repetitive patterns. Some early solutions to reconstruct 3D depth from random-dot autostereogram were proposed almost three decades ago, e.g. [17], [18], [19]. In theory, 3D reconstruction can be achieved using stereo vision techniques, e.g. cropping a left and a right image from the picture, then applying an available global stereo matching process [20] to build a disparity/depth map.

### *C. Our Proposed Stereogram Marker*

In this paper, we propose a new technique that optically encodes binary data in a stereogram image. The stereogram can be printed on a black-border square and used as a marker in Augmented Reality applications. The proposed marker is named "StereoTag", its design is shown in Fig. 3(b). Each StereoTag is a square with a dimension $D$ measured in pixels or millimetres; border thickness is 10% of $D$. The quadrilateral property of the squares can be used to detect their four straight lines and four corners. These are crucial for calculating the poses of the markers and cameras in an AR application. The internal image is a stereogram (size $W \times W$) made of three regions. The central area is a fixed image (region $A$ that fills

up 50% of the stereogram) and two repeated patterns on both sides of region $A$ (region $B$ and region $C$ with 25% each).




(a) Traditional 2D Barcode maker (b) Our proposed marker

Fig. 3. Our proposed autostereogram marker (right) v.s. a traditional 2D Barcode maker (left).

Compare to traditional markers such as the 2D barcode marker shown in Fig. 3(a). Our proposed StereoTag presents two significant advantages:

- **Virtually Informative Picture:** The stereogram embedded inside each marker is made from meaningful image patterns, rather than "random black and white squares.
- **Flexibility of Image Pattern:** The decoded information is independent of image patterns, we can use a broad range of images to construct stereograms that encode the same numeric information.

### D. Structure of the Paper

The following sections will cover the detail of our proposed techniques and organised as follows. Sec. II outlines the design and implementation to achieve a reliable StereoTag. Sec. III demonstrates initial results made from some conducted experiments. Sec. IV then concludes the paper.

## II. Design and Implementation

The design and implementation described in this section aim to answer the following questions:

- How do we build a detectable StereoTag (Make).
- How do we know what binary information the StereoTag is encoded (Solve).

The formal question was briefly answered in Sec. I-C. More detail will be further discussed in Sec. II-A; and solution to the latter question will be outlined in Sec. II-B.

### A. Encode Binary Information in Stereogram

Fig. 4 demonstrates basic steps of creating our proposed marker (StereoTag). As described, our StereoTag has a thick black border so that it is easily and reliably detectable under various circumstances. The inside stereogram can encode some optical machine-readable representation of data such as "one dimensional" (1D) or "two-dimensional" (2D) bar-codes.

There are numerous 1D bar-code (that is made up of lines and spaces of various widths) and 2D barcode (or matrix code) standards. Code11, Code 32, Code 49, Code 93, Code

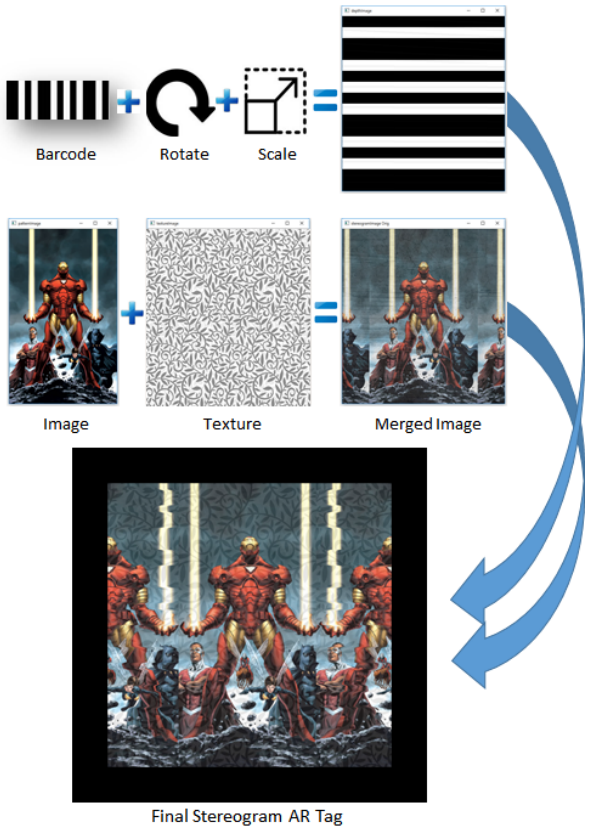



Fig. 4. Process of making a detectable stereogram-based AR marker.

128, EAN-8, EAN-13 [21] are some of the most popular 1D barcodes. Well-known 2D bar-codes are VSCode, Aztec Code, Data Matrix, Maxi Code, PDF417, Visual Code, ShotCode, and QR Code [22]. Thanks to their popularities, there exists many libraries to create and read these bar-codes. In theory, we could utilise any of them to embed inside our stereogram image. At this stage, we choose one of the most simple of 1D bar-code – the "Pharmacode".

*1) Pharmacode:* Also known as Pharmaceutical Binary Code [23], it is frequently used in the pharmaceutical industry as a packing control system. Pharmacode can represent only a single integer which is encoded in binary (using narrow bars and thick bars only). The minimum Pharmacode is 2 bars and the maximum is 16 bars, so the smallest number that could be encoded is 3 with two narrow bars, and the biggest is 131070 ($\approx 2^{17}$) with 16 thick bars. Each number corresponds to the IDs of computer graphics models in the database.

*2) Pharmacode as a disparity map (depth):* A disparity map holding depth of each pixel is needed to create stereogram. Assume that we have a Pharmacode storing a number between 3 and 131070. The Pharmacode block is 90 degree clockwise rotated, scaled, and converted to a corresponding disparity map $D*$ (a $W \times W$ matrix where each element is a number ranging between 0 and $d_{MAX}$). The map $D*$ is used to code to the distance in how many pixels are between a

pair of corresponding points in a stereogram. As Pharmacode normally contains black bars on white background, there are only two intensity levels: 0 and $d_{MAX}$. We can encode black coloured pixels in Pharmacode block as a 0-level disparity point, and white coloured pixels is coded as N-level disparity point. A final disparity map is a grey-scaled image with intensity ranged between $[0, d_{MAX}]$. $d_{MAX}$ represent the maximum shift of a point from its origin in stereogram image; for convenience, we choose $d_{MAX} = \frac{W}{64}$.

*3) A stereogram that optically hides a Pharmacode:* The central stereogram image will have its dimension of $W \times W$. In stereograms, two pixels with the same colour act as virtual conjugate pixel projections in 3D and relate to one pixel in the disparity map; these two pixels are the correspondent points.

First, we choose an image $I_{pattern}$ to be placed on the marker, the image will need to be resized to $W \times H$. It is then copied to the central region of the stereogram. This central picture is kept untouched (region $A$ - Fig. 3(b)) and will be used to repeat itself on the left and the right regions (regions B and C). In other words, pixels with identical/same colours to the left and right of a selected pattern are added with different horizontal shifts according to the disparity map $D*$. The following implementations describe the process in more detail:

1) From left to right:

```
for (y = 0; y < image.height; y++) {
 for (x = startX; x < image.width/4; x++) {
  depth = getValue(matrixCode, y, x);
  xToDraw = x + W_pattern - depth;
  xToGet = x;
  if(xToDraw>=startX+W_pattern){
   newPixel=getValue(stereogram,y,xToGet);
   setValue(stereogram,y,xToDraw,newPixel);
  }
 }
}
```

2) From right to left:

```
for (y = 0; y < image.height; y++) {
 for (x = startX/4; x >= 0; x--) {
  depth = getValue(matrixCode, y, x);
  xToDraw = x + depth;
  xToGet = x + W_pattern;
  newPixel = getValue(stereogram,y,xToGet);
  setValue(stereogram,y,xToDraw,newPixel);
 }
}
```

3) At this point, an autostereogram with a hidden Pharmacode is made. The image is then decorated with a thick black border to generate our final AR StereoTag. Three examples of the StereoTag markers are shown in Fig. 7(b), 7(c), and 7(d). All of them are hiding the same Pharmacode shown in Fig. 7(a).

### B. Marker-based Detection and Decryption

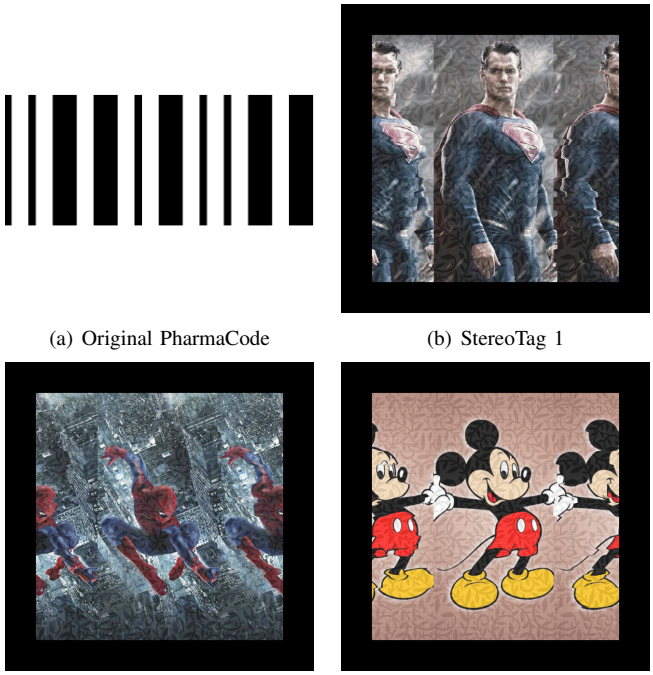
*1) Detection:* An AR system can estimate the pose of the camera using the four corners (coplanar but non-collinear points) of detected marker due to known constraints (a square border) [24]. First, we need to find closed contours on the input image. The inside image is checked for a correct marker model. With a calibrated camera, the system can render virtual objects in the right place. Some image processing steps are used to detect thick-border markers, which are outlined below:

- Step 1: Convert the input image from RGB to greyscale.
- Step 2: Perform an adaptive binary thresholding method.
- Step 3: Detect contours using line detection, line fitting and line sorting methods.
- Step 4: Search for all possible markers and extract internal images.

Once marker's border is detected, we can estimate both the correct scale and pose of the camera: its location $(x, y, z)$ and its rotation angles $(\alpha, \beta, \gamma)$. These image-processing tasks are well known and are therefore not discussed here; further detail can be found in [25].

*2) Decryption of StereoTag:* The internal stereogram image of the detected StereoTag is used to identify which binary information was encoded, to render correct graphic on the screen at marker location. This step is equivalent to a stereo reconstruction process applied on two stereo images $C_1$ and $C_2$ with $C_1$ is the left half of the stereogram and $C_2$ is the right half of the stereogram. The disparity levels ranged between 0, $d_{MAX}$ are known from the width of the internal stereogram: $d_{MAX} = \frac{W}{64}$.

*3) Stereo Reconstruction:* Stereo Reconstruction or Stereo Matching is the process to extract depth information from a stereo pair of images. Given a known disparity range, it can output a disparity map, which characterises the observed 3D surface.



(a) Original PharmaCode (b) StereoTag 1

(c) StereoTag 2 (d) StereoTag 3

Fig. 5. Three examples of our StereoTag markers created from the same Pharmacode and various images found on the Internet.

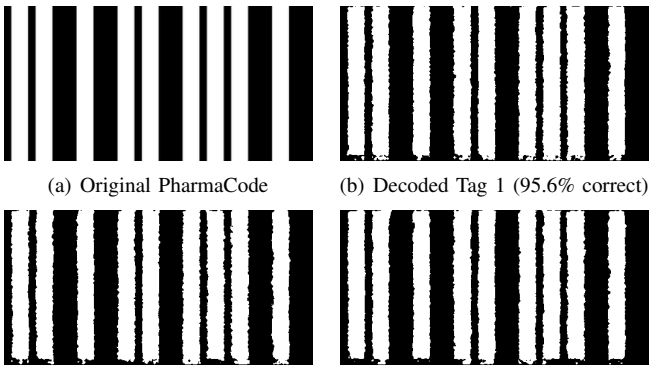
Assume we have two stereo images $C_1$ and $C_2$, and a disparity map $D^*$. Map $D^*$ specifies the computational stereo vision reconstruction of continuous optical surface from the pair $C_1$ and $C_2$. Let $R$ be a finite lattice supporting the disparity maps $D^* = [d_{x,y} : (x, y)] \in R$. The 3D location of every binocularly visible surface point is determined from coordinates, $(x_1, y)$ and $(x_2, y)$, of the corresponding pixels in the images via their horizontal distance: $d_{x,y} = x_1 x_2$. The reconstructed maps are either projected to one of the images (left/right), or to Cyclopean coordinates: $x = \frac{1}{2}(x_1 + x_2)$.

Many stereo matching algorithms estimate disparity maps $D^*$ by minimizing the globally energy $E(D|C_1, C_2)$ passing through the two images $C_1$ and $C_2$. The energy is a weighed sum of the mismatch and non-smoothness terms as shown below:

$$E(D|C_1, C_2) = E_{mm}(D|C_1, C_2) + \beta E_{ns}(D)$$

Block Matching Stereo (BMS) is a local stereo matching algorithm is created by K. Konolige Bierling [26]. It computes stereo matches that minimise the Squared Absolute Differences (SAD) over local neighbourhoods. Semi-Global Block Matching (SGBM) is a semi-global implementation of BMS [27], SGBM combines a SAD-based local cost-function and a smoothness-term in a global energy function. Both of BMS and SGBM are available in the OpenCV package[1].

Due to the availability and reliability, we have employed SGBM for the decryption purpose. The product of SGBM is a normalised grey-scaled disparity image. After an anti-clockwise rotation of 90 degrees, we will retrieve the hidden Phamacode. Fig. 6(b), 6(c), 6(d) display three decoded results of the three tags that are previously shown in Fig. 5. Incorrect pixels (near the borders) are presented in the decoded barcodes. However, the overall shapes of thick and narrow bars are well reserved. In fact, approximately 95% of the three decoded tags are the same as the original Pharmacode (with pixel-by-pixel comparison).



(a) Original PharmaCode (b) Decoded Tag 1 (95.6% correct)

(c) Decoded Tag 2 (95.0% correct) (d) Decoded Tag 3 (94.6% correct)

Fig. 6. Three examples of rebuilt Tags from the samples shown in Fig. 5.

This Pharmacode can be scanned and recognised effectively by many available tools to achieve a unique number (ID of a graphic model). This ID will be used to search for a unique 3D figure in a database, which can be rendered on the screens of computers or mobile devices.

## III. Experimental Results

Is our proposed StereoTag sensitive to different lighting conditions, noises, and scaling? We test these criteria using the one of the StereoTags – "Superman" tag shown in Fig. 7(b). We carry out three experiments to observe the changes in quality of decoded Pharmacodes after *(1) alternating the brightness and contrast of a marker, (2) scaling original image, (3) adding a noise and raindrops*.

We use IrfanView tool[2] to alternate the sizes, apply effects, and add noises to the samples before checking the decoded results. Some examples are shown in Fig. 7.

(a) Tag with +100 levels Brightness (b) Tag with +100 levels Contrast

(c) Tag with +100 levels Noises (d) Tag with +100 raindrops Effect

Fig. 7. Four examples of StereoTag markers after various effects added by IrfanView.

### A. Lighting conditions:

Different brightness and contrast levels are applied on top of the "Superman" tag (Top rows of Fig. 7). Results achieved from the affected images are compared with the original Pharmacode, the percentage of same colour pixels are collected as accuracy. Quantitative results are shown in Tab. I. Overall, the rebuilt Phamacodes are still relatively accurate: between 92% and 96%.

### B. Different scaling:

We also rescale the "Superman" tag to different sizes: $640 \times 640$, $320 \times 320$, $320 \times 200$, $200 \times 160$, $100 \times 100$ pixels. For

[1]http://opencv.org/

[2]http://www.irfanview.com/

TABLE I
QUALITY AFFECTED BY CHANGES IN BRIGHTNESS AND CONTRAST

| Brightness | Accuracy | Contrast | Accuracy | Size | Accuracy |
|---|---|---|---|---|---|
| 0 | 95.6% | 0 | 95.6% | 640x640 | 95.6% |
| +100 | 94.3% | +50 | 95.7% | 320x320 | 95.3% |
| -100 | 95.5% | -50 | 95.1% | 200x200 | 94.5% |
| +200 | 92.2% | +100 | 94.4% | 160x160 | 94.2% |
| -200 | 93.2% | -100 | 93.5% | 100x100 | **89.4%** |

TABLE II
QUALITY CHANGED BY NOISES, BLURS, AND RAINDROPS

| Noise | Accuracy | Blur | Accuracy | Rain drops | Accuracy |
|---|---|---|---|---|---|
| 0 | 95.6% | 0 | 95.6% | 0 | 95.6% |
| +25 | 94.8% | +25 | 95.7% | +25 | 92.3% |
| +50 | 92.0% | +50 | 95.7% | +50 | 89.7% |
| +75 | **87.0%** | +75 | 95.8% | +75 | **87.3%** |
| +100 | **82.1%** | +100 | 95.8% | +100 | **85.5%** |

each tag sample, we collect the accuracy again; results are shown in Tab. I. The accuracy falls slightly below 90% only when image size is as small as $100 \times 100$ pixels.

### *C. Noises, blurs, and other effects:*

We apply some other IrfanView's built-in effects: noises, blurs, and raindrops on the "Superman" tag (Bottom rows of Fig. 7). The accuracy of rebuilt Phamacodes at each test is collected in Tab. II. From the results, image blurs do not decrease the quality; while raindrops do create some effects: only 85% of pixels are correctly recovered at 100 raindrop levels. Random noises in the image are the strongest factors. Thus, only 82% of the overall shape retained after 100 levels of noise. However, if the levels of noises, blurs, and raindrops are kept under 50, accuracies are likely to be above 90%.

## IV. CONCLUSIONS

Augmented Reality (AR) is an exciting topic which helps to blend reality with virtual reality. One challenge of AR is to place correctly virtual data within the real environment. Using bar-code marker is one of the successful solutions. However, bar-code normally contains either black and white vertical bars or square dots; thus, it looks uninteresting and uninformative. In this paper, we propose a new marker type (called StereoTag), which is not only presenting a realistic-looking image but also encoding a broad range of numeric data in its stereogram patterns.

From some experiments conducted, our proposed StereoTag is robust under various lighting conditions, insensitive to noises and relatively accurate. Thanks to these superiors, StereoTag could be a promising approach to be used in future AR applications.

## REFERENCES


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

[2] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, B. MacIntyre, Recent advances in augmented reality, Computer Graphics and Applications, IEEE 21 (6) (2001) 34–47.

[3] I. E. Sutherland, The ultimate display, Multimedia: From Wagner to virtual reality.

[4] V. Lepetit, P. Lagger, P. Fua, Randomized trees for real-time keypoint recognition, in: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, Vol. 2, IEEE, 2005, pp. 775–781.

[5] V. Lepetit, P. Fua, Keypoint recognition using randomized trees, Pattern Analysis and Machine Intelligence, IEEE Transactions on 28 (9) (2006) 1465–1479.

[6] M. Billinghurst, H. Kato, I. Poupyrev, The magicbook-moving seamlessly between reality and virtuality, Computer Graphics and Applications, IEEE 21 (3) (2001) 6–8.

[7] S. Siltanen, Theory and applications of marker-based augmented reality, 2012.

[8] J. Rekimoto, Y. Ayatsuka, Cybercode: designing augmented reality environments with visual tags, in: Proceedings of DARE 2000 on Designing augmented reality environments, ACM, 2000, pp. 1–10.

[9] T.-W. Kan, C.-H. Teng, W.-S. Chou, Applying qr code in augmented reality applications, in: Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry, ACM, 2009, pp. 253–257.

[10] T.-Y. Liu, T.-H. Tan, Y.-L. Chu, Qr code and augmented reality-supported mobile english learning system, in: Mobile multimedia processing, Springer, 2010, pp. 37–52.

[11] F. Bergamasco, A. Albarelli, A. Torsello, Pi-tag: a fast image-space marker design based on projective invariants, Machine vision and applications 24 (6) (2013) 1295–1310.

[12] F. Bergamasco, A. Albarelli, L. Cosmo, E. Rodola, A. Torsello, An accurate and robust artificial marker based on cyclic codes.

[13] A. Tikanmäki, J. Röning, Markers–toward general purpose information representation, in: IROS2011 workshop: knowledge representation for autonomous robots, 2011.

[14] N. T. Enterprises, The Magic Eye, Volume I: A New Way of Looking at the World, Andrews McMeel Publishing, 1993.

[15] C. W. Tyler, M. B. Clarke, Autostereogram, in: SC-DL tentative, International Society for Optics and Photonics, 1990, pp. 182–197.

[16] R. Kimmel, 3d shape reconstruction from autostereograms and stereo, Journal of Visual Communication and Image Representation 13 (1) (2002) 324–333.

[17] R. Szeliski, G. Hinton, Solving random-dot stereograms using the heat equation, CVPR, 1985.

[18] R. Szeliski, Cooperative algorithms for solving random-dot stereograms, Carnegie-Mellon University, Department of Computer Science, 1986.

[19] N. Qian, T. J. Sejnowski, Learning to solve random-dot stereograms of dense and transparent surfaces with recurrent backpropagation, in: Proceedings of the 1988 Connectionist models summer school, Morgan Kaufmann, 1989, pp. 435–443.

[20] D. Scharstein, R. Szeliski, Middlebury stereo vision page, Online at http://www. middlebury. edu/stereo 6.

[21] R. C. Palmer, P. Eng, The Bar Code Book: A Comprehensive Guide to Reading, Printing, Specifying, Evaluating and Using Bar Code and Other Machine-readable Symbols, Trafford Pub., 2007.

[22] H. Kato, K. T. Tan, Pervasive 2d barcodes for camera phone applications, Pervasive Computing, IEEE 6 (4) (2007) 76–85.

[23] D. Sharma, V. Sharma, B. Shrivastava, R. K. Songara, P. Sharma, Regulatory aspect of barcode technology.

[24] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, Cambridge university press, 2003.

[25] R. Szeliski, Computer vision: algorithms and applications, Springer Science & Business Media, 2010.

[26] M. Bierling, T. R. Hsing, Displacement estimation by hierarchical block matching, in: Proceedings of the SPIE Visual Communications and Image Processing, Vol. 1001, Massachusetts, USA, 1988, pp. 942–953.

[27] H. Hirschmüller, Stereo processing by semiglobal matching and mutual information, Pattern Analysis and Machine Intelligence, IEEE Transactions on 30 (2) (2008) 328–341.