International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 8 (2016) pp. 423–433

Multi-agent-based Two-dimensional Barcode Decoding Robust against Non-uniform Geometric Distortion

Kazuya Nakamura, Kohei Kamizuru, Hiroshi Kawasaki, and Satoshi Ono

Department of Information Science and Biomedical Engineering,
Graduate School of Science and Engineering, Kagoshima University
Kagoshima, Japan
k3420880@kadai.jp, {sc111015,kawasaki, ono}@ibe.kagoshima-u.ac.jp

Abstract: **Two-dimensional (2D) codes are subject to distortion when printed on non-rigid materials, such as papers and clothes. Although general 2D code decoders correct uniform distortion such as perspective distortion, it is difficult to correct non-uniform and irregular distortion of the 2D code itself. To overcome this problem, an agent-based approach is presented here to reconstruct the 2D code. In this approach, auxiliary lines are placed on a 2D code and used to recognize distortion. First, 2D code area is identified through feature patterns composed by the auxiliary lines, and Convolutional Neural Network (CNN) is used to discriminate the patterns. Then, many agents simultaneously trace the lines referring to the various image features and the neighborhood agents. The feature weights are optimized by Genetic Algorithm. The experimental results indicate that agents successfully tracked auxiliary lines right up to occluded area, and the proposed method could decode distorted 2D codes. The performance of the proposed method against distortion level and occlusion amount was also clarified.**

Keywords: Two-dimensional barcode, Image rectification, Convolutional neural network, Multi-agent system, Genetic algorithm

I. Introduction

Two-dimensional (2D) barcodes [**?**] are widely used to quickly identify objects without physical contact because their capacity is greater than that of single-dimensional barcodes. The rapid spread of 2D code decoder software on mobile phones increased the application of 2D codes in fields such as advertisement and authentication. 2D code is made to be printed on a flat object. Although most 2D codes have some functional module patterns to correct distortion, only uniform distortion such as projection transformation and barrel-type distortion can be corrected. In other words, it is difficult to correct irregular distortion of 2D codes, such as that of a wrinkled cloth. A 2D code decoding technology with a non-uniform distortion correction function could be further applied in fields such as commercial distribution and agricultural and medical industries [**?**, **?**].

The authors have proposed a 2D code with colored auxiliary lines and its decoding method [**?**, **?**]. This 2D code can be decoded even if distorted by compensating distortion. The distortion is recognized by tracing the auxiliary lines; then, the blocks formed by the recognized lines are rectified to square-shaped. In addition, occluded areas are estimated using the lines whose color pattern constitutes a de Bruijn sequence [**?**]. Conversely, the 2D codes require color printers to print the colored lines and color cameras to decode them. Furthermore, the decoding method is affected from changes in the lighting condition; thus monochrome lines are preferred to colored ones.

However, it is difficult to apply the color auxiliary line-based method [**?**, **?**] to 2D codes with monochrome auxiliary lines. This is because monochrome line identifications (IDs) cannot be recognized locally, whereas a de-Bruijn-based color combination of only three contiguous parallel lines allows recognizing line IDs. Therefore, belief propagation (BP) [**?**, **?**], the graphical optimization method used in [**?**], must be performed based on the smoothness cost, not data cost. In addition, monochrome lines make it difficult to distinguish close parallel lines when they are crossing or touching.

This paper proposes a 2D code with monochrome auxiliary lines and its decoding method. 2D code with monochrome lines are improved from previous When using monochrome auxiliary lines, line tracking is necessary from regions where line IDs can be recognized correctly such as the boundaries of the 2D code and the areas close to toe finder patterns. Therefore, the proposed method mainly performs multi-agent-based line tracking to recognize module location with eliminating influence of distortion and occlusion. In this method, many agents placed on the 2D code boundaries start tracing auxiliary lines on the code. The agents determine their routes based on the reliability of their movement. Recognized auxiliary lines enable to locating modules and reading the binary codes from the 2D code image; thus, an embedded message can be read.

The proposed decode method involves offline training processes requiring different soft computing techniques Convolutional Neural Network (CNN) and Genetic Algorithm (GA) [**?**]. CNN is adopted to generate a detector that searches for finder pattern, and GA is adopted to optimize feature weights used to reliability calculation.

The experimental results indicate that the proposed CNN-

based detector locates finder patterns even from distorted 2D code images to determine agent initial positions. In addition, according to reliability designed by GA, agents successfully traced distorted auxiliary lines right up to occluded regions. Finally, the proposed method reconstructed 2D codes involving sufficiently few errors that could be complemented by error correction function of 2D code, indicating that the proposed method could successfully decode the distorted 2D codes.
The main contributions of this paper are as follows:

- *A 2D code with monochrome auxiliary lines*, which is robust against non-uniform, non-smooth, and non-periodical distortion and occlusion while maintaining the compatibility with the basic 2D code. The lines also make the code detectable by a Center Surround Extremas (CenSurE) detector [**?**]. Finder patterns are not necessarily available when 2D codes are highly distorted.

- *CNN-based finder pattern detection*, which mainly aims to recognize the 2D code's pose and contour for agent position initialization rather than detecting a 2D code from a captured image.

- *Metaheuristics-based feature weighting for line tracking agent design*, which allows agents to track monochrome auxiliary lines, to keep appropriate distance between neighboring agents, and to stop when they cannot trace the line due to hard distortion or self-occlusion of the 2D code.

II. Related work

A. Line tracking and multi-agent system

Auxiliary lines in a 2D code proposed in this paper are used for recognizing distortion and occlusion that will be discussed in Sec. III. Line tracking is a widely used technique in various fields such as autonomous robots [**?**, **?**, **?**], biometrics [**?**], and augmented reality [**?**] and other computer vision tasks [**?**]. Miura *et al.* proposed a method to extract a finger-vein pattern by an infrared camera, in which a tracking point goes along avalley of intensity values in cross-sectional view of grayscale image [**?**]. Wuest *et al.* proposed a method for real-time tracking of 3D objects, in which 2D anisotropic Gaussian mask is applied before tracking lines [**?**]. There are other line trace methods such as eigenvalue calculation [**?**] and a combination of RANSAC and Kalman filter [**?**].
There are also many line tracking methods by agents such as [**?**, **?**, **?**]. Multi-agent approach has also been attempted for multiple line tracking [**?**, **?**]. For different types of tasks, specific agent learning and cooperation methods have been proposed; for instance, one method proposed in [**?**] changes intensity value on agent trails to avoid repetitive line tracking by other agents, resulting in indirect communication between agents, the other proposed in [**?**] adopts a two-layer multi-agent model where agents communicate directly. A general multi-agent model that simultaneously learns knowledge for line tracking task and cooperative behavior between agents has not been established.

B. CNN-based feature learning

Recently, deep learning (DL), a kind of Neural Network (NN) with many layers, is attracting interest in various research fields. After Hinton's success in deep layer NN using recursive unsupervised learning [**?**], various DL methods have been proposed. In ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 [**?**], DL won the victory by more than 10% accuracy over the second.
The essential advantage of DL is *representation learning*; it can directly be applied to a target problem raw data rather than its analyzed features, whereas conventional methods give image processing to input image to extract features, and convert them to form in accordance with learning machine's input. In particular, DL is appropriate to various tasks in image processing field.
Convolutional Neural Network (CNN) is a type of DL specialized for image recognition, and originally has been developed for character recognition. CNN is classified as feed forward NN that does not involve loop structure, and has an advantage that pre-training is not necessary when big training data is available. CNN involves convolution and pooling layers that alternate with each other. Mainly five techniques are applied to these layers: local receptive field, tied weight, feature maps, convolution, and pooling. Ciresan demonstrated that using plural CNNs in parallel successfully made more outstanding performance than conventional methods in many benchmarks such as MNIST [**?**] and medical image analysis [**?**]. Now, CNN has been a de facto standard for representative learning in image processing field.

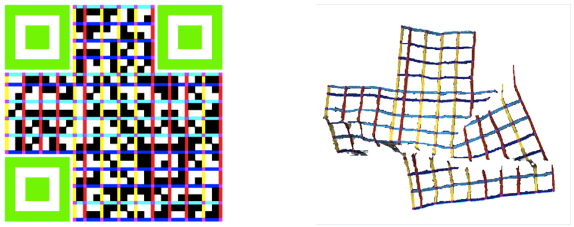
C. Previous work for distorted 2D code decoding

1) Methods for uniform distortion

Some studies have tried to develop a decoding method tolerant to perspective transformation [**?**, **?**, **?**, **?**, **?**, **?**]. Chen *et al.* has also proposed a preprocessing algorithm involving geometric rectification by affine transformation based on finder and alignment pattern positions to cope with unsuitable shooting angle [**?**]. Ohbuchi *et al.* proposed a method for decoding skewed or curved 1D and 2D codes by recursive approximation the code boundaries [**?**]. Hohberger *et al.* proposed a color 2D code tolerant to wave, shear, scaling and perspective distortions [**?**].

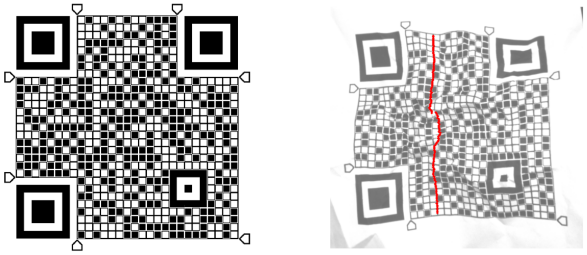
2) 2D code with auxiliary lines

The authors also proposed decoding methods of distorted 2D code with colored auxiliary lines and colored finder patterns as shown in Figure 1 [**?**, **?**]. These methods recognize auxiliary lines using dynamic programming or BP, allowing knowing module locations with eliminating the influence of distortion and occlusion. De Bruijn color sequence makes possible to recognize line IDs only from three contiguous parallel lines. However, coloring requires calibration and good lighting condition, and there are some materials to be printed that cannot be colored. In some applications, discoloring and fading due to aging or chalking by sunlight are not negligible.
The authors attempted to make the auxiliary lines monochromatic and proposed an agent-based line tracing method [**?**]. This method uses 2D code with monochrome lines as shown



(a) 2D code with colored lines (b) Line recognition example.

Figure. 1: 2D code with colored auxiliary lines in previous work [**?**]



(a) 2D code with monochrome lines (b) Line recognition error example.

Figure. 2: 2D code with monochrome auxiliary lines in previous work [**?**].

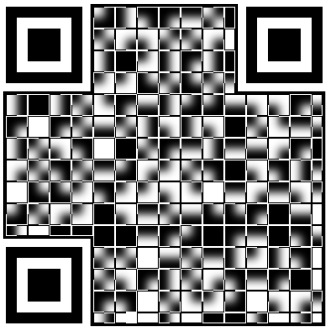


Figure. 3: 2D code with monochrome auxiliary lines and smoothed module edges proposed in this paper, which is compatible with QR code.

in Figure 2 (a). Agents trace the lines and their trajectories are adopted as recognized auxiliary lines telling module locations. However, the dense auxiliary lines frequently causes agent tracking failure as shown in Figure 2 (b). In addition, this 2D code requires specific auxiliary patterns in a quiet zone around the 2D code that helps to determine agent initial position.

III. The Proposed method

A. *Overview*

In this paper, we propose a 2D code with monochrome auxiliary lines and a decoding method for the 2D code. The proposed decoding method adopts multi-agent based approach to recognize the auxiliary lines on the distorted 2D code. An agent is assigned to each horizontal or vertical line from side to side, and traces the line with calculating reliability for each pixel.

The reason why we adopt agent-based approach is that an efficient line tracing algorithm is unknown to trace lines which are highly distorted and orthogonally crossing with many other lines. Furthermore, high distortion makes lines touch each other and causes self-occlusion that hides parts of the lines. Learning appropriate reliability of agent movement with training images allows us to automatically design agent behavior for line tracing. The reliability is calculated from some features such as image features and other agents' trajectories, and its weight parameters are learned by Genetic Algorithm (GA) [**?**]. The reason why GA is adopted is that the objective function of feature weight optimization must be multimodal because similar but different in detail features are included.

Determining initial positions of agents is important in addition to designing agent behavior. The proposed method puts agents on contour of 2D codes region except finder patterns. Therefore, finder patterns must be detected from the estimated 2D code region for determining agent initial position rather than detecting 2D code. In general, the finder patterns can be found by line scan; dark, bright, dark, bright and dark patterns having a ratio of $1:1:3:1:1$ is the key to detect the patterns. However, when decoding distorted 2D codes, the above ratio is frequently deflected. In addition, unlike the previous work [**?**], finder patterns are not colored. Therefore, the proposed method generates a detector for finder patterns using CNN.

In this section, the proposed 2D code with auxiliary lines and the process flow of the proposed decoding method is shown in Sec. III-B and Sec. III-C, respectively. Then, the details of the proposed decoding method are explained by dividing it into two parts: agent initial position determination (Sec. III-D) and line tracking (Sec. III-F). The main contributions of this paper, CNN-based finder pattern detection and GA-based feature weighting for reliability calculation, are discussed in Sec.III-E and Sec.III-G.

B. *2D code with monochrome auxiliary lines*

Figure 3 shows an example 2D code proposed in this paper. Monochrome auxiliary lines are placed on a QR code basically every two modules horizontally and vertically, constructing lattice form, whereas the previous work [**?**] places lines on every module interval as shown in Figure 2(a). This is because dense auxiliary lines easily get together and collide with each other when a 2D code is distorted. On the other hand, too sparse auxiliary lines degrade the performance of distortion correction. To balance this trade-off, adequate auxiliary line interval is every two or three modules. Reducing the lines also increases processing speed.

To recognize monochrome lines, an edge detection process is necessary but edges on module intervals disturb agents' line tracking. Therefore, in the proposed 2D code, the boundaries between modules are smoothed by Gaussian filter.

The above modifications to base 2D code do not destroy compatibility between the proposed and base 2D codes because general 2D code decoders sample a center pixel of each module rather than the entire module region.

Although the auxiliary lines are added to mainly recognize distortion and occlusion, the lines are also useful to locate 2D

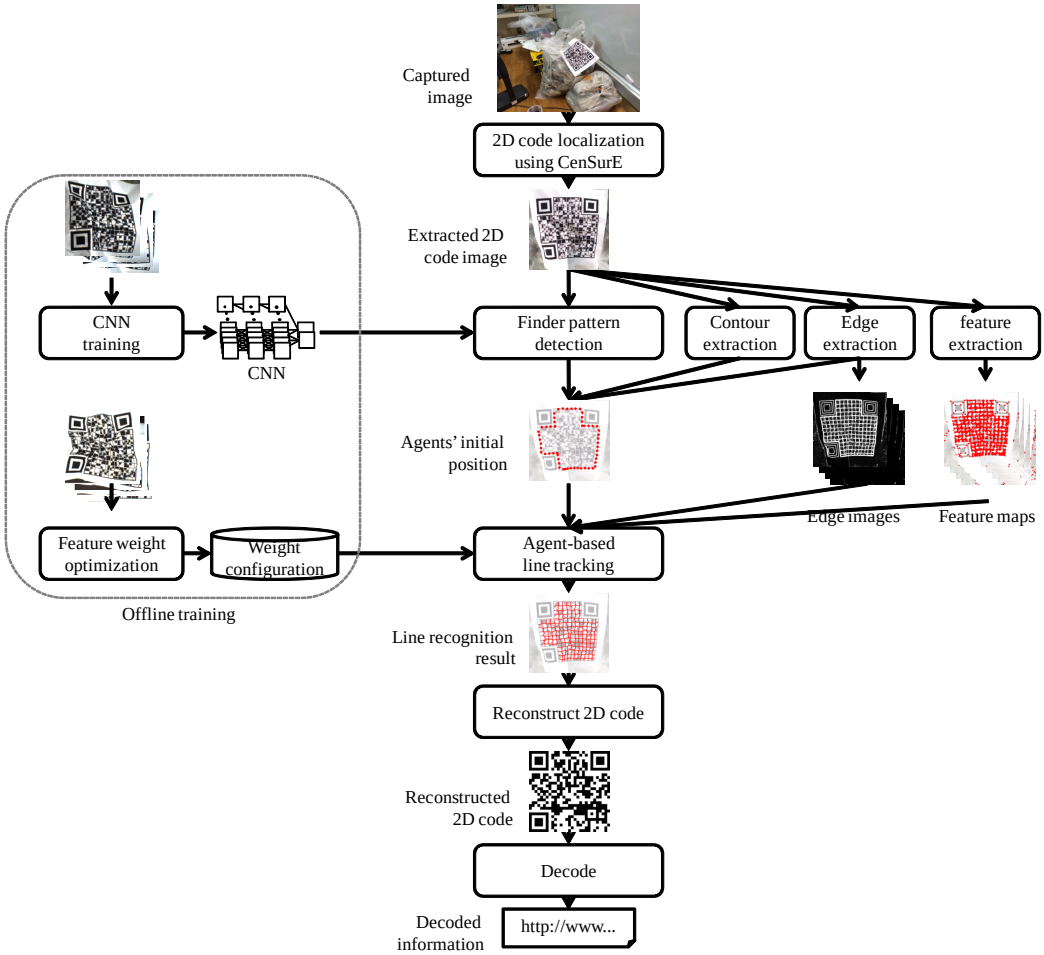


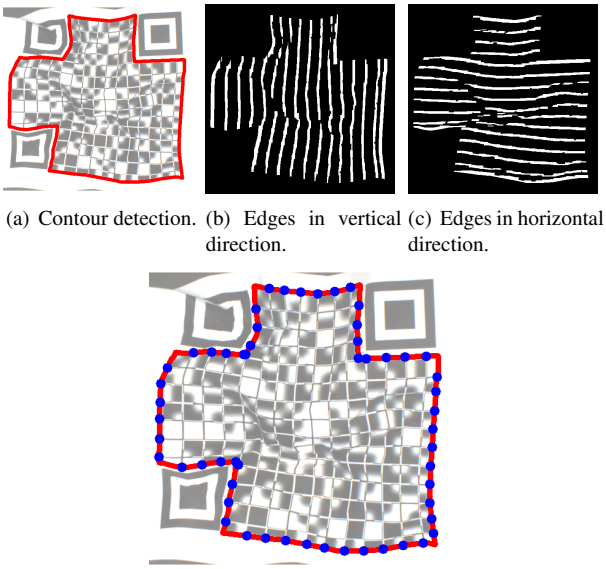
Figure. 4: Process flow of the proposed method.

codes from the entire captured images; many same-size rectangles compose a characteristic pattern, which is easily distinguished from background. The proposed method locates 2D codes by adopting a CenSurE detector [**?**], which uses a simplified bi-level kernel as a center-surround filter. This filter works considerably well for locating the 2D code because of the code's latticed lines [**?**]. Finder patterns are not necessarily available when 2D codes are highly distorted because the bright and dark module with ratio of $1:1:3:1:1$ gets easily deviated.

In addition, by simplifying CNN-based finder pattern detector from ternary to binary classification, the 2D code proposed in this paper successfully eliminates pentagon-like auxiliary patterns on quiet zone that was not negligible for agent initialization in the previous work [**?**].

C. Process flow of the proposed decode method

Figure 4 shows the process flow of the proposed method. At first, the proposed method finds 2D code using CenSurE detector, and detects finder patterns using another detector automatically designed by CNN, as described in Sec. III-E. Then, the proposed method restores the distorted 2D code by recognizing these auxiliary lines, and assigning IDs to detected lines as described in Sec. III-F. Finally, the proposed method reconstructs 2D codes, which is performed by and sampling



(a) Contour detection. (b) Edges in vertical direction. (c) Edges in horizontal direction.

(d) Estimated start positions.

Figure. 5: Estimation of agents' start positions.

four center pixels of four modules in each block formed by recognized auxiliary lines, and decodes the 2D code.

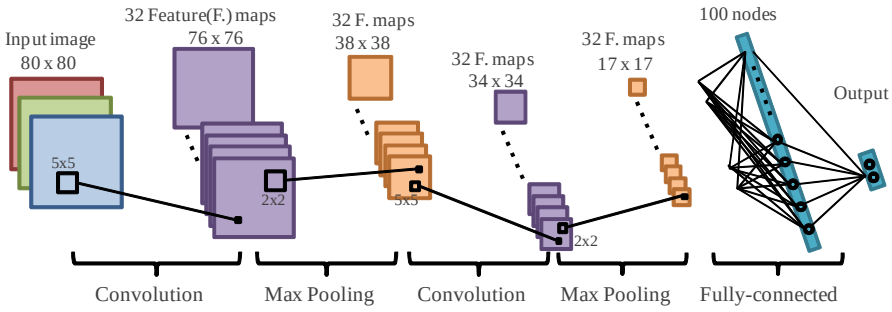


Figure. 6: CNN structure for finder pattern detection.

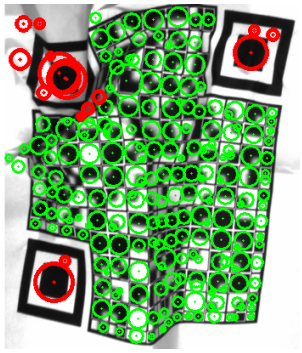


Figure. 7: Example of blob detection

Table 1: Details of CNN network structure.

layer	type	kernel size	image resolution	#nodes
input	grayscale	–	80×80	1
1st	convolution	5×5	76×76	32
2nd	max pooling	2×2	38×38	32
3rd	convolution	5×5	34×34	32
4th	max pooling	2×2	17×17	32
5th	fully connected	–	–	100
output	label	2	–	–

(a) Finder pattern (b) Other modules

Figure. 8: Examples of training images. Red marks denote the target pixels, which are actually not included in the training data images.

D. Agents' initial position determination

In the proposed method, estimating appropriate agents' start positions is important because it affects the entire line recognition result. The proposed method estimates the initial positions on the contour by recognizing finder patterns and modules located on the boundary of 2D code. To begin, a contour of 2D code except the finder patterns is extracted, as shown in Figure 5(a). The method to find finder patters is discussed in Sec. III-E. Then, the proposed method applies opening (a morphological image process) filter with kernels having a strong directivity to obtain vertical and horizontal edge components as shown in Figs 5(b) and 5(c). Finally, starting positions are determined according to intersection points between the contour and edge components as shown in Figure 5(d).

E. CNN-based finder pattern detection

In the proposed method, finder patterns of 2D code are recognized agent initialization. When determining the start points of agent-based tracking, the boundary of 2D code is necessary and the detected finder patterns are also used to extract the boundary. To detect finder patterns, a CNN-Based detector is proposed in this paper.

We regard this problem as a binary classification problem, and this CNN-based detector assigns one of two kinds of labels, "a finder pattern", and "others", to a target pixel. Figure 6 shows the structure of the CNN-based detector. This involves five layers: input, convolution, max pooling, convolution, max pooling and fully-connected layers. Table 1 shows the details of the network. In convolution layers, Rectified Linear Units function is used, which allows CNN to be trained faster. In the fully-connected layer, Dropout method is used to avoid overfitting. This selects some edges randomly and sets their weight to be zero; only remaining edges are trained.

Basically, the proposed method applies the CNN-based detector to keypoints that are centers of blobs extracted from a photographed 2D code. The target of the above blob detection is ellipse-shaped blobs involving partially defected ones. This is because the patterns and modules might have a deficit in a part of their boundary or their vertical and horizontal aspect ratios might change due to distortion. The above blob detection allows us to obtain many rectangles as shown in Figure 7. The extracted blob set involves modules, finder patterns, and other rectangle-like shapes caused by distortion. To apply the CNN-based detector, a rectangle image window of $S_W \times S_W$ pixels is trimmed so that a target keypoint corresponds to the center of the region and the region involves about 7×7 modules, as shown in Figure 8.

Training image dataset is generated from photographed 2D code images. Finder patterns in the training images are colored only for preparing ground truth, allowing us to easily annotate the training data; the images are translated into grayscale image after annotation. The training data is multiplied by rotating and laterally reversing the captured images.

When applying the trained CNN-detector, the proposed method applies Laplacian filter to obtain an edge image of input, and then extracts keypoints corresponding to centers of extracted blobs. Then, the CNN-based detector is applied for each keypoint. After assigning labels to the all keypoints, blobs are clustered into two kinds of areas, allowing the proposed method to obtain finder patterns.

At the same time, the proposed method can obtain average size of blobs whose center is assigned a label corresponding to "others", which is used as an estimated block size $\overline{S}_B$ in the following processes including line trace by agents.

F. Agent-based auxiliary line detection

In the proposed method, many agents simultaneously track all auxiliary lines on a photographed 2D code image. The

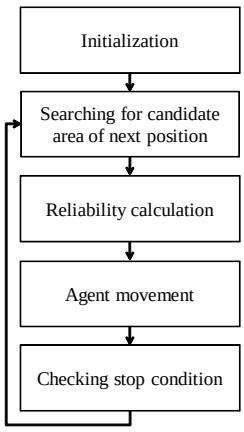
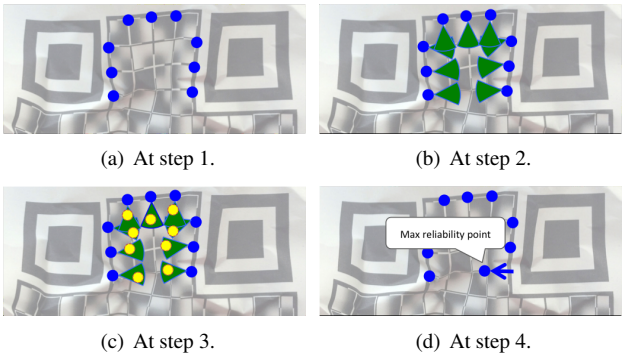


Figure. 9: Agent's line tracking algorithm.



(a) At step 1. (b) At step 2. (c) At step 3. (d) At step 4.

Figure. 10: Agent movement.

procedure how to detect auxiliary lines is as follows.

[Step 1] Initialization:
Extracting 2D code contour and set agents at starting points which are intersections of auxiliary lines on the contour as shown in Figure 10(a). Each agent has its own ID corresponding to the target auxiliary line ID. For each auxiliary line, two agents are assigned; they start tracking from either side of the 2D code contour. The starting position of one agent corresponds to the others destination position. Using two agents for each line enables avoiding bad influence from occluded region; it is expected that one occluded region on a line can be estimated correctly by being tracked by two agents.
Details of estimating starting points are described in III-D.

[Step 2] Searching for candidate area of next position:
For each agent at each iteration, candidate area of next position is determined and reliability is calculated for each pixel in the area. The candidate area is a fan-shaped region whose radius is $\frac{1}{3} \times \overline{S}_B$ and whose central angle is 90 degree as shown in Figure 10(b). Considering the distance to its contiguous agents, areas whose distance to the trajectory of the contiguous agent is $1.5 \times \overline{S}_B$ are removed from the candidate area. This is because subsequent agents who move to the same direction should not leave each other more than $\overline{S}_B$ in the orthogonal direction to their trajectory. In addition, areas in which angle between agent inertia direction and a vector from its start to destination positions exceeds 90 degrees are prohibited to move.

[Step 3] Reliability calculation:
Reliability is calculated for each pixel in the moving destination candidate region for each agent. The reliability represents how likely a target pixel is on an auxiliary line, as described in III-G. The pixel with the highest value of reliability in the candidate region is the moving destination candidate pixel as shown in Figure 10(c).

[Step 4] Agent movement:
At each iteration, only the agent that has the moving destination candidate pixel with the highest reliability moves to its candidate pixel as shown in Figure 10(d). This allows agents with lower reliability to refer trajectories of other agents' with higher reliability.

[Step 5] Checking stop condition:
The stop condition of this algorithm is that all agent pairs pass each other on their target line. Each agent stops when it meets another agent tracking the same line from the opposite side of the 2D code. If the condition is not satisfied, then go back to Step 2.

G. Reliability calculation

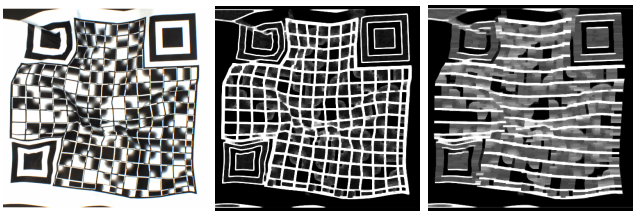
As described in Sec. III-F, agents move toward its goal position according to reliability. A reliability value $R(a, p)$ of agent a at pixel p is defined as follows:

$$R(a,p) = \sum_{i=1}^{K} w_i \mathrm{E}_i \tag{1}$$

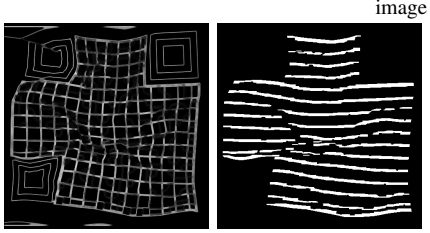
where E_i is subfunction related to i-th feature, and w_i is a weight parameter, and K $(= 26)$ is the total number of kinds of features. Features used in the proposed method are mainly classified into two types: image-based features and agent-related features.
Reliability subfunctions E_1 to E_{18} based on image features refer five types of input images:

- a source image I_S,
- its edge image I_E obtained by applying Laplacian edge detection and closing (a morphological image process) filters,
- directional edge images $I_{ED}^{(v)}$ and $I_{ED}^{(h)}$ obtained by applying Laplacian filter and closing with strong directionality,
- Local Binary Pattern (LBP) feature map I_L for an image obtained by applying Laplacian filter, and
- directional edge images $I_{LD}^{(v)}$ and $I_{LD}^{(h)}$ obtained by applying closing with strong directionality to I_L.

Agents moving horizontally refer $I_{ED}^{(h)}$ and $I_{LD}^{(h)}$ that have the horizontal direction, whereas ones moving vertically refer $I_{ED}^{(v)}$ and $I_{ED}^{(v)}$ that have the vertical direction. Figure 11 shows examples of these images, and Table 2 shows a list of subfunctions using image features.
Subfunctions E_{19} to E_{26} based on agent behavior are classified into three types: based on its own behavior (C_1), neighbor's behavior (C_2), and population's behavior (C_3). Table 3 shows a list of the subfunctions based on agent behavior. In the second types of the subfunctions, the value of the subfunction is averaged over those of two neighbor agents if



(a) Source image I_S (b) Edge image I_E (c) Directional edge image $I_{ED}^{(v)}$



(c) Edge image by LBP I_L (c) Directional edge image by LBP $I_{LD}^{(v)}$

Figure. 11: Example edge images for reliability calculation.

Table 2: Reliability subfunctions based on image features.

No	feature	image
$E_1, E_2, E_7, E_{10}, E_{11}$	Intensity on source or edge image	$I_E, I_S, I_{ED}, I_L, I_{LD}$
E_3, E_4	Corner detection result (Eigen value, Harris)	I_S
E_5	Distance from blob center	I_S
E_6	Distance from edge	I_E
E_8	Distance from center of estimated auxiliary line	I_E
E_9	Number of blobs detected nearby	I_S
$E_{12}, \ldots, E_{18}$	Feature detector (FAST, STAR, SIFT, SURF, MSER, FAST pyramid, FAST dynamic)	I_S

there are two contiguous neighbors that move to the same direction.

H. Weight parameter learning by GA

The weight $\boldsymbol{w} = \{w_1, w_2, \ldots, w_K\}$ is optimized by GA to select and prioritize appropriate features for the auxiliary line tracking. The optimization function to be minimized in this problem is calculated from the actual result of auxiliary line tracking as follows:

$$f(\boldsymbol{w}) = \sum |\boldsymbol{x}_T - \boldsymbol{x}_a| \tag{2}$$

where $\boldsymbol{x}_T$ and $\boldsymbol{x}_a$ are intersection point coordinates of ground truth and estimated by agent a with $\boldsymbol{w}$, respectively.

When calculating a fitness function of an individual (potential solution), leave-one-out cross validation is performed for a training dataset, and a mean value of $f(\boldsymbol{w})$ for test data is adopted as its fitness value.

IV. Experiments

A. Setup

This section demonstrates the effectiveness of the proposed 2D code and its decoding method for distortion. First, two of the main contributions of the proposed method, CNN-based finder pattern detection and agent-based auxiliary line tracking, were demonstrated with three test images shown in Figure 12, whereas 2D code detection from a captured image was not evaluated here because the proposed method uses CenSurE detector as with the previous work [**?**, **?**]. Then, the performance of the proposed method against the degree of distortion and self-occlusion was clarified with 30 test images.

Table 3: Reliability subfunctions based on agents' behavior

No	feature	type
E_{19}	Angle difference to its own trajectory	C_1
E_{20}	Distance from neighbor agent d_{18}	C_2
E_{21}	Angle difference to trajectory of neighbors	C_2
E_{22}	Reliability of neighbors	C_2
E_{23}	Ratio of white pixels at directed edge	C_3
E_{24}	Angle difference to other agents walking in the same direction	C_3
E_{25}	Angle difference to other agents walking in orthogonal direction	C_3
E_{26}	Number of passed agents	C_3

The three test images shown in Figure 12 have following properties: In test image 1, all auxiliary lines were not occluded; however, some modules collapsed by distortion and there are shadowed areas. Test image 2 involves a self-occluded area at the center of the code where auxiliary lines cannot be viewed. There is also an area where modules are highly distorted in the upper middle part of the test image 2. Test image 3 involves two self-occluded area that may cause serious line recognition failure because it is difficult to track lines in visible area caught by occluded areas.

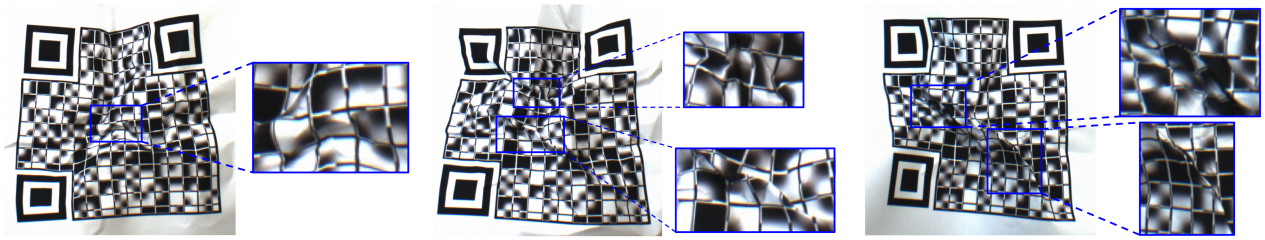
1) Parameter configurations

Configurations of two kinds of offline training, CNN-based detector learning and GA-based feature weight learning, were as follows.

Training a CNN-based detector for finder patterns was performed with the training data described below; 50 distorted 2D code images were photographed with varying exposure angles, generating 250 distorted 2D code images. Then, rotation and mirroring were applied to the photographed images to multiply training data. Blob detection is performed for each image and $S_W \times S_W$ size image is extracted for each detected blob where $S_W = 80$ pixels. Finally, 130,000 training images were available for training.

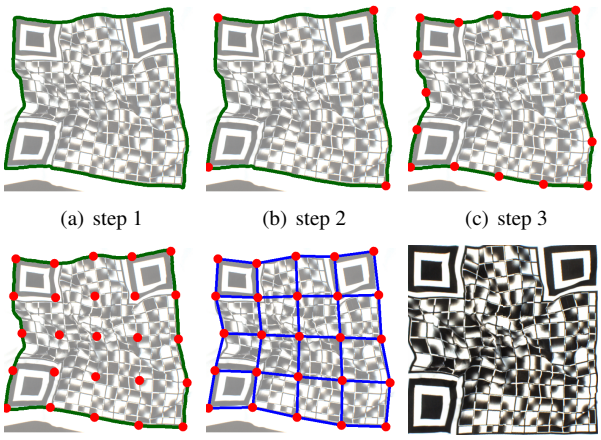
Agents for line tracking are trained by optimizing weight vector $\boldsymbol{w}$. A training dataset involving four images was used in this experiment, and the averaged value of $f(\boldsymbol{w})$ to the test image in four folds of leave-one-out cross validation was adopted as a fitness of an individual. In GA, design variables (feature weights w_i) were represented as real values, and real-coded ensemble crossover (REX), uniform mutation, and Just Generation Gap (JGG) were adopted. Other control parameters in GA were configured as follows. Population size and maximum number of generations were set to be 200 and 10,000, respectively. Crossover and mutation ratio were 90% and 5%, respectively. The number of offsprings, which is a control parameter in REX, was set to 30.

All images in training and test datasets were captured by Lynx acA1300-30gc and Fujinon DF6HA-1B with the resolution of $1,278 \times 958$. 2D codes were printed with the size of 16 [cm], and the photographing distance was set to about 40 [cm].



(a) Tested image 1. (b) Tested image 2. (c) Tested image 3.

Figure. 12: Examples of tested images and enlargements around self-occluded areas.



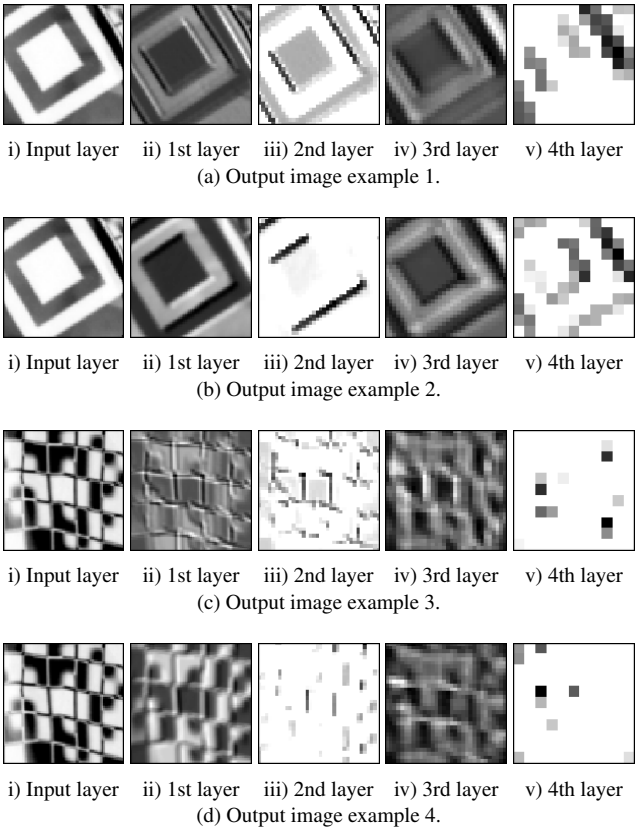
(a) step 1 (b) step 2 (c) step 3

(d) step 4 (e) step 5 (f) step 6

Figure. 13: Contour-based rectification method based on [**?**].

2) Compared method: contour-based rectification method

For comparative study, we implement a contour-based rectification method based on the idea of [**?**], which aims to decode 2D codes printed on cylinder or other uniformly curved surface. To reconstruct a 2D code, this contour-based method first extracts contour of 2D code involving finder patterns (Figure 13(a)), then divides the 2D code into 4×4 grids (Figure 13(e)), and finally rectifies each block to a square (Figure 13(f)). The grid is constructed by finding corners (Figure 13(b)), dividing each side into four segments having equal length (Figure 13(c)), and linearly interpolating inner grid points (Figure 13(d)).



i) Input layer ii) 1st layer iii) 2nd layer iv) 3rd layer v) 4th layer
(a) Output image example 1.

i) Input layer ii) 1st layer iii) 2nd layer iv) 3rd layer v) 4th layer
(b) Output image example 2.

i) Input layer ii) 1st layer iii) 2nd layer iv) 3rd layer v) 4th layer
(c) Output image example 3.

i) Input layer ii) 1st layer iii) 2nd layer iv) 3rd layer v) 4th layer
(d) Output image example 4.

Figure. 14: Example output images in each CNN layers.

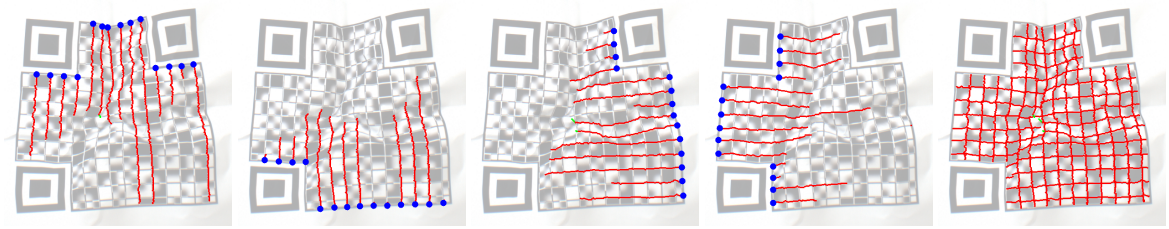
B. Results on agent position initialization involving finder pattern detection

To determine initial agent position, the proposed method extracts a contour of a 2D code except finder patterns using the CNN-based detector. Therefore, we first demonstrate intermediate node output examples of the trained CNN, and then show the recognition result of finder patterns and agent initial positions in the three test images.

Figure 14 shows output examples of two nodes in each layer. Images shown in Figure 14(a) and (c) are produced by the same node in each layer. Similarly, images shown in Figure 14(b) and (d) are also the output produced by the same nodes. In addition, examples shown in Figure 14(a) and (b) are outputs for the same finder pattern image, and examples shown in Figure 14(c) and (d) are for the same image involving modules and auxiliary lines. These intermediate output images show that the trained CNN successfully produced features to distinguish edges of finder patterns and other modules; edges of finder patterns remained in deeper layer while edges of other modules and auxiliary lines were destroyed.

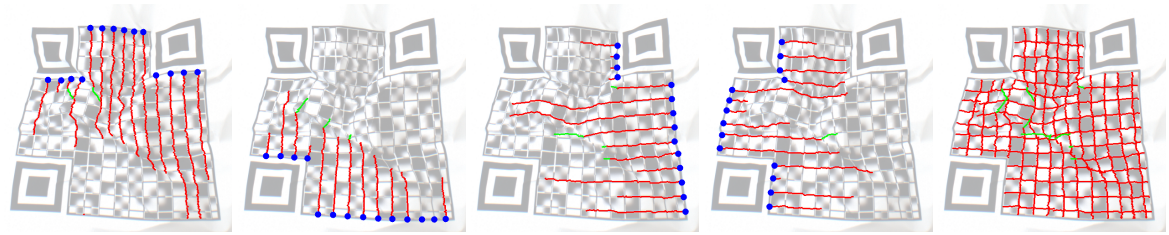
Figure 15 shows the example result of finder pattern detection using trained CNN-based detector; red circles show blobs recognized as finder patterns. Those examples showed that finder patterns were successfully found by the trained CNN.

Figures (a), (b) (c), and (d) in 17, 18, and 19 show agent initial positions and trajectories; blue circles show the agent initial positions. Those figures demonstrate that the proposed method successfully determined initial positions of all agents in the tested three images.



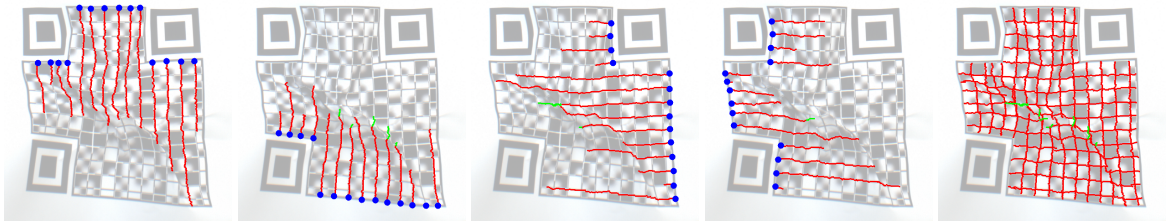
(a) Top to bottom (b) Bottom to top (c) Right to left (d) Left to right (e) Line recognition result

Figure. 17: Agent trajectories and extracted auxiliary lines in tested image 1.



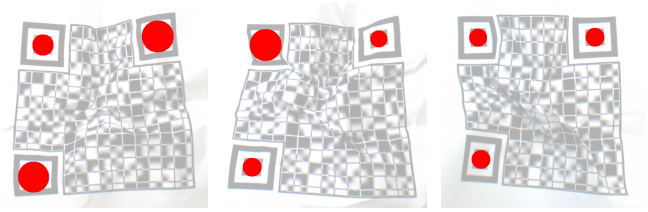
(a) Top to bottom (b) Bottom to top (c) Right to left (d) Left to right (e) Line recognition result

Figure. 18: Agent trajectories and extracted auxiliary lines in tested image 2.



(a) Top to bottom (b) Bottom to top (c) Right to left (d) Left to right (e) Line recognition result

Figure. 19: Agent trajectories and extracted auxiliary lines in tested image 3.



(a) Tested iamge 1 (b) Tested image 2 (c) Tested image 3

Figure. 15: Example results of finder pattern search by CNN.

C. Results on auxiliary line recognition by agents

The proposed method reconstructs distorted 2D codes by recognizing auxiliary lines and locating all modules. As described in Sec. III-F, III-G, and III-H, agents move based on reliability that is calculated with optimized feature weights. Then, this section shows the optimized feature weight by GA and demonstrates agent trajectories.

Figure 16 shows the feature weights optimized by GA for reliability calculation. Basically, there is a tendency that image features were prioritized than agent-related features. In detail, features that directly contribute to find lines such as E_7 and E_8 were viewed as important.

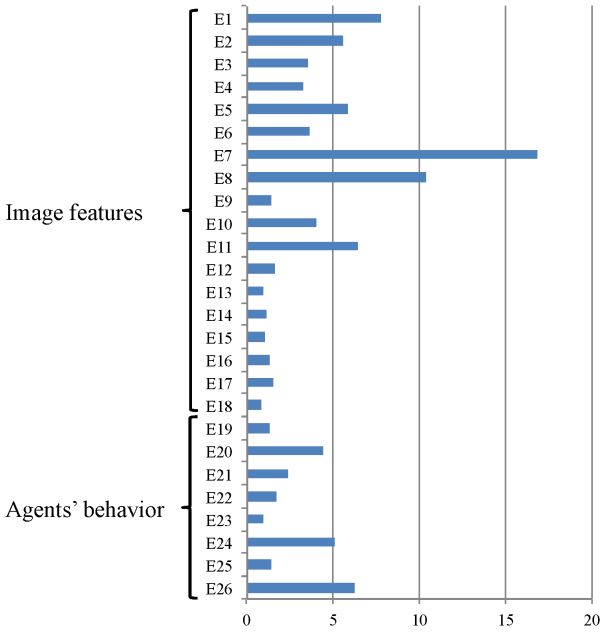


Figure. 16: Optimized feature weights by GA.

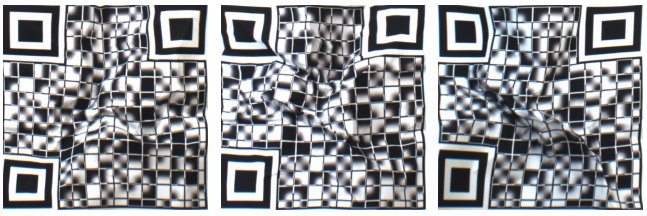
Figures 17, 18, and 19 show the example agent trajectories with the optimized weights and recognized auxiliary lines. In these figures, red lines show agent trajectories that correctly coincided to the actual auxiliary lines, and green lines show the trajectories that were not match the actual lines. These figures shows that agents tracing in flatter area were prioritized, and agents in highly distorted area slowed down or stopped. Figures 18 and 19 demonstrated that the agents

(a) Tested image 1 (b) Tested image 2 (c) Tested image 3

Figure. 20: Reconstructed 2D code by the proposed method.

(a) Tested image 1 (b) Tested image 2 (c) Tested image 3

Figure. 21: Error modules by the proposed method.



(a) Tested image 1 (b) Tested image 2 (c) Tested image 3

Figure. 22: Rectification result by the previous method.

(a) Tested image 1 (b) Tested image 2 (c) Tested image 3

Figure. 23: Reconstructed 2D code by the previous method.

(a) Tested image 1 (b) Tested image 2 (c) Tested image 3

Figure. 24: Error modules by the previous method.

stopped around the occluded regions; although, on the occluded area, the agents went wrong way, they could successfully trace auxiliary lines right up to the occluded area.

D. 2D code reconstruction results

Finally, we demonstrate the reconstructed 2D codes by the proposed method, and compare the results with contour-based rectification method. In addition, the performance of the proposed method against distortion level and occlusion amount is shown.

Figures 20 and 21 show the reconstruction results of the 2D codes by the proposed method and their error module distribution. In Figure 21, red and pink modules indicate error modules wrongly judged as bright and dark one, respectively. The number of error modules in the tested images 1, 2 and 3 were 7, 37, and 27, respectively. Figures 21(b) and (c) showed that modules on the occluded areas were difficult to sample correctly. However, all the reconstruction results shown in Figure 20 were successfully decoded by general QR code decoders thanks to error correction function of Reed-Solomon code.

Figures 22, 23, and 24 show rectification results, reconstructed 2D codes, and error distributions of the contour-based rectification method. Although the contour-based rectification method rectified the distorted 2D codes to square-shaped, modules were not placed on the right position due to non-uniform and non-local distortion, resulting in failure of sampling grid construction. The all reconstructed 2D codes shown in Figure 23 were not decoded by the general decoders.

Figure 25 shows the distributions of the success and failure images by the proposed method; markers show the decode result in which a circle means success and x-mark means failure. Distortion level is defined by root mean square error (RMSE) of distorted 2D code depth against its approximated plane in 3D space. The depth images were captured by a projector-camera system involving EPSON ELP710 and Point Grey Flea3 with FUJIFILM DF6HA-1B and projecting time-varying structured light [**?**] [1] . Occlusion amount is defined as the number of modules that were counted manually. Figure 25 demonstrates that 20 of the tested 30 images of distorted 2D codes were successfully decoded by the proposed method. The proposed method failed to decode the 2D codes when either distortion or occlusion become hard at a certain level; harder distortion made it more difficult to trace auxiliary lines that were sharply bent or shaded darkly, and large occlusion hided many modules that could not be complemented by error correction function of QR code.

Processing time of the proposed method was 1.66 [s] on average[2], whereas that of previous work [**?**] was 3.16 [s]. This implies that the processing speed of the proposed method was raised by 1.9 times as high as the previous work by reducing the number of auxiliary lines, which determines the number of agents. Because the implemented system was not parallelized at all, the practical speed would be achieved by parallel implementation or conversion into hardware.

V. Conclusion

Proposed in this paper is a decoding method for distorted 2D codes with monochrome auxiliary lines. The proposed method comprises three key ideas:

[1]Note that 3D information is used only for calculating distortion level. The proposed method only uses 2D image information and does not require any 3D scan device.

[2]The implemented system was run on a PC/AT compatible machine with Intel Core i7 870 (2.93GHz) and 16GB RAM. This processing time does not involve the time to find a 2D code from a captured image.

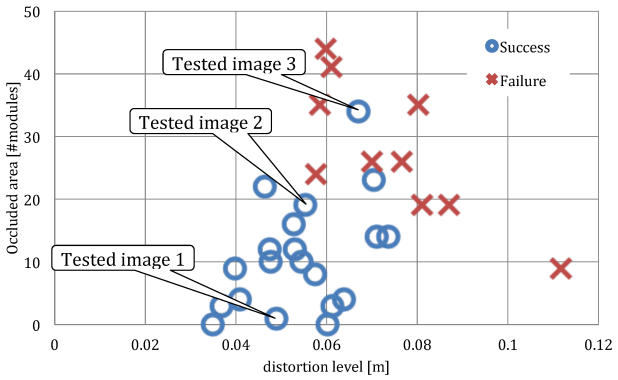


Figure. 25: Distribution of decode results.

- 2D code with relatively sparse lines and smoothed module boundaries,
- finder pattern recognition by CNN, and
- agent-based auxiliary line detection.

Experimental results have shown that the proposed CNN-based pattern detector successfully found finder patterns that guides agent initial position correctly, and the agents optimized by GA successfully tracked the auxiliary lines until they face occluded region. However, it was difficult to trace the lines that were highly distorted or occluded.
In future, we attempt to let agents use several different weight parameter configurations for reliability calculation in accordance with estimated distortion level. In addition, effective use of alignment patterns is also important to improve robustness against distortion.

Acknowledgment

Part of this work was supported by JSPS KAKENHI Grant Number 15H02758. The authors also would like to thank Emeritus Prof. Shinsuke Fujita at Kagoshima University for his help.

Author Biographies

Kazuya Nakamura recieved the B.S. degree in Engineering from Kagoshima University in 2014. His research interests include multi-agent simulation, genetic programming and deep neural network.

Kohei Kamizuru recieved the B.S. degree in Engineering from Kagoshima University in 2015. His research interests include two-dimensional barcode decoding and computer vision.

Hiroshi Kawasaki is a full professor of Department of Information and Biomedical Engineering at Kagoshima University. He received a Master degree in 2000 and a Ph.D. degree in 2003 from University of Tokyo. He started working at Kagoshima University in 2010. Prior to Kagoshima University, he worked at Saitama University. He also researched at Columbia University as visiting researcher in 2012, INRIA Rhone Alpes in 2009 and Microsoft Research Redmond in 2000. He is a member of VRSJ, IPSJ, IEICE, ACM, and IEEE. His current research focus is on a 3D capturing technique of moving objects and its application on VR and AR systems.

Satoshi Ono received his Ph.D. degree in Engineering at University of Tsukuba in 2002. He worked as a Research Fellow of the Japan Society for the Promotion of Science (JSPS) from 2001 to 2003. Subsequently, he joined Department of Information and Computer Science, Graduate School of Science and Engineering, Kagoshima University as a Research Associate. He is currently an Associate Professor in Department of Information Science and Biomedical Engineering in the same university. His research interest includes artificial intelligence and evolutionary computation and their application to real world problems.