applied sciences

Article

Recognition of Perspective Distorted QR Codes with a Partially Damaged Finder Pattern in Real Scene Images

Ladislav Karrach [1], Elena Pivarčiová [1,*] and Pavol Bozek [2]

[1] Department of Manufacturing and Automation Technology, Faculty of Technology, Technical University in Zvolen, Masarykova 24, 960 01 Zvolen, Slovakia; karrach@zoznam.sk

[2] Institute of Production Technologies, Faculty of Materials Science and Technology, Slovak University of Technology in Bratislava, Vazovova 5, 811 07 Bratislava, Slovakia; pavol.bozek@stuba.sk

* Correspondence: pivarciova@tuzvo.sk

Received: 21 September 2020; Accepted: 30 October 2020; Published: 4 November 2020

Featured Application: Mobile robot navigation, automatic object identification and tracking.

Abstract: QR (Quick Response) codes are one of the most famous types of two-dimensional (2D) matrix barcodes, which are the descendants of well-known 1D barcodes. The mobile robots which move in certain operational space can use information and landmarks from environment for navigation and such information may be provided by QR Codes. We have proposed algorithm, which localizes a QR Code in an image in a few sequential steps. We start with image binarization, then we continue with QR Code localization, where we utilize characteristic Finder Patterns, which are located in three corners of a QR Code, and finally we identify perspective distortion. The presented algorithm is able to deal with a damaged Finder Pattern, works well for low-resolution images and is computationally efficient.

Keywords: QR code detection; adaptive thresholding; finder pattern; perspective transformation

1. Introduction

Some of the requirements that are placed on autonomous mobile robotic systems include real-world environments navigation and recognition and identification of objects of interest with which the robotic system have to interact. Computer vision allows machines to obtain a large amount of information from the environment that has a major impact on their behavior. In the surrounding environment there are often numerous different static objects (walls, columns, doors, and production machines) but also moving objects (people, cars, and handling trucks).

Objects that are used to refine navigation, landmarks, can be artificial (usually added by a human) and natural (which naturally occur in the environment) [1,2].

Robotic systems have to work in a real environment and must be able to recognize, for example, people [3], cars [4], product parameters for the purpose of quality control, or objects which are to be handled.

The use of QR (Quick Response) codes (two-dimensional matrix codes) in interaction with robots can be seen in the following areas:

- in the field of navigation as artificial landmarks-analogies of traffic signs that control the movement of the robot in a given area (no entry, driving direction, alternate route, and permitted hours of operation) or as an information board providing context specific information or instructions (such as identification of floor, room, pallets, and working place)

- in the area of object identification, 2D codes are often used to mark products and goods and thus their recognition will provide information about the type of goods (warehouses), the destination of the shipment (sorting lines) or control and tracking during track-and-trace.

QR Codes

QR Codes are classified among 2D matrix codes (similar to Data Matrix codes). QR Codes (Model 1 and Model 2) are squared-shaped 2D matrices of dark and light squares—so called modules. Each module represents the binary 1 or 0. Each QR Code has fixed parts (such as Finder Patterns and Timing Patterns) that are common to each QR Code and variable parts that differ according to the data that is encoded by a QR Code. Finder Patterns, which are located in the three corners of a QR Code, are important for determining the position and rotation of the QR Code. The size of the QR code is determined by the number of modules and can vary from 21 × 21 modules (Version 1) to 177 × 177 (Version 40). Each higher version number comprises four additional modules per side. In the Figure 1 we can see a sample of version 1 QR Code.

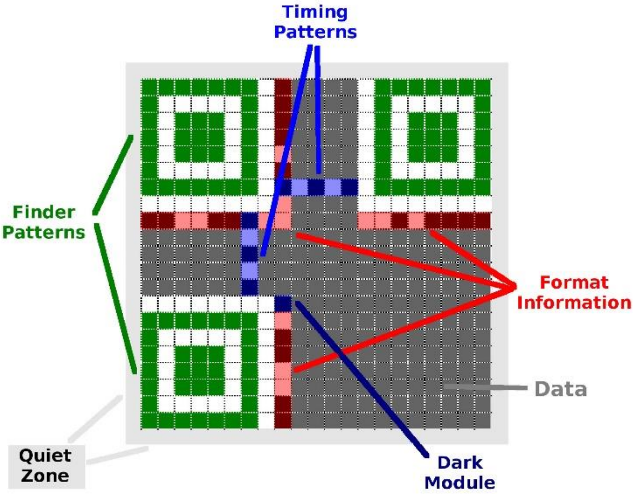

Figure 1. Version 1: 21 × 21 QR Code.

QR Code has error correction capability to restore data if the code is partially damaged. Four error correction levels are available (L–Low, M–Medium, Q–Quartile, and H–High). The error correction level determines how much of the QR Code can be corrupted to keep the data still recoverable (L–7%, M–15%, Q–25%, and H–30%) [5]. The QR Code error correction feature is implemented by adding a Reed–Solomon Code to the original data. The higher the error correction level is the less storage capacity is available for data.

Each QR Code symbol version has the maximum data capacity according to the amount of data, character type and the error correction level. The data capacity ranges from 10 alphanumeric (or 17 numeric) characters for the smallest QR Code up to 1852 alphanumeric (or 3057 numeric) characters for the largest QR Code at highest error correction level [5]. QR Codes support four encoding modes—numeric, alphanumeric, Kanji and binary—to store data efficiently.

QR Code was designed in 1994, in Japan for automotive industry, but currently has a much wider use. QR codes are used to mark a variety of objects (goods, posters, monuments, locations, and business cards) and allow to attach additional information to them, often in the form of a URL to a web page. QR Code is an ISO standard (ISO/IEC 18004:2015) and is freely available without license fees.

In addition to a traditional QR Code Model 1 and 2, there are also variants such as a Micro QR Code (a smaller version of the QR Code standard for applications where a symbol size is limited) or an iQR Code (which can hold a greater amount of information than a traditional QR Code and it supports also rectangular shapes) (Figure 2).

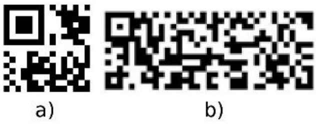

Figure 2. Other types of a QR Code: (**a**) Micro QR code, (**b**) iQR code.

2. Related Work

Prior published approaches for recognizing QR Codes in images can be divided into Finder Pattern based location methods [6–12] and QR Code region based location methods [13–18]. The first group locates a QR Code based on the location of its typical Finder Patterns that are present in its three corners. The second group locates the area of a QR code in the image based on its irregular checkerboard-like structure (a QR Code consists of many small light and dark squares which alternate irregularly and are relatively close to each other).

A shape of the Finder Pattern (Figure 3) was deliberately chosen by the authors of the QR Code, because "it was the pattern least likely to appear on various business forms and the like" [6]. They found out that black and white areas that alternate in a 1:1:3:1:1 ratio are the least common on printed materials.

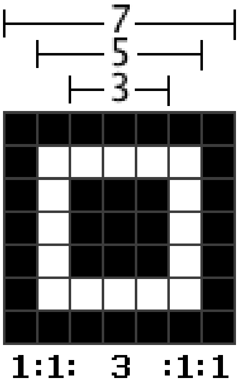

Figure 3. Finder Pattern.

In [7] (Lin and Fuh) all points matching the 1:1:3:1:1 ratio, horizontally and vertically, are collected. Collected points belonging to one Finder Pattern are merged. Inappropriate points are filtered out according to the angle between three of them.

In [8] (Li et al.) minimal containing region is established analyzing five runs in labeled connected components, which are compacted using run-length coding. Second, coordinates of central Finder Pattern in a QR Code are calculated by using run-length coding utilizing modified Knuth–Morris–Pratt algorithm.

In [9] (Belussi and Hirata) two-stage detection approach is proposed. In the first stage Finder Pattern (located at three corners of a QR Code) is detected using a cascaded classifier trained according to the rapid object detection method (Viola–Jones framework). In the second stage geometrical restrictions among detected components are verified to decide whether subsets of three of them correspond to a QR Code or not.

In [10] (Bodnár and Nyúl) Finder Pattern candidate localization is based on the cascade of boosted weak classifiers using Haar-like features, while the decision on a Finder Pattern candidate to be kept or dropped is decided by a geometrical constraint on distances and angles with respect to other probable Finder Patterns. In addition to Haar-like features, local binary patterns (LBP), and histogram of oriented gradients (HOG) based classifiers are used and trained to Finder Patterns and whole code areas as well.

In [11] (Tribak and Zaz) successive horizontal and vertical scans are launched to obtain segments whose structure complies with the ratio 1:1:3:1:1. The intersection between horizontal and vertical segments presents the central pixel of the extracted pattern. All the extracted patterns are transmitted to a filtering process based on principal components analysis, which is used as a pattern feature.

In [12] (Tribak and Zaz) seven Hu invariant moments are applied to the Finder Pattern candidates obtained by initial scanning of an image and using Euclidean metrics they are compared with Hu moments of the samples. If the similarity is less than experimentally determined threshold, then the candidate is accepted.

In [13] (Sun et al.) authors introduce algorithm, which aims to locate a QR Code area by four corners detection of 2D barcode. They combine the Canny edge detector with external contours finding algorithm.

In [14] (Ciążyński and Fabijańska) they use histogram correlation between a reference image of a QR code and an input image divided into a block of size 30 × 30. Then candidate blocks are joined into regions and morphological erosion and dilation is applied to remove small regions.

In [15] (Gaur and Tiwari) they propose approach which uses Canny edge detection followed by morphological dilation and erosion to connect broken edges in a QR Code into a bigger connected component. They expect that the QR Code is the biggest connected component in the image.

In [16,17] (Szentandrási et al.) they exploit the property of 2D barcodes of having regular distribution of edge gradients. They split a high-resolution image into tiles and for each tile they construct HOG (histogram of oriented gradients) from the orientations of edge points. Then they select two dominant peeks in the histogram, which are apart roughly 90°. For each tile, a feature vector is computed, which contains a normalized histogram, angles of two main gradient directions, a number of edge pixels and an estimation of probability score of a chessboard-like structure.

In [18] (Sörös and Flörkemeier) areas with high concentration of edge structures as well as areas with high concentration of corner structures are combined to get QR Code regions.

3. The Proposed Method

Our method is primarily based on searching for Finder Patterns and utilizes their characteristic feature—1:1:3:1:1 black and white point ratio in any scanning direction. The basic steps are indicated in the flowchart in Figure 4.

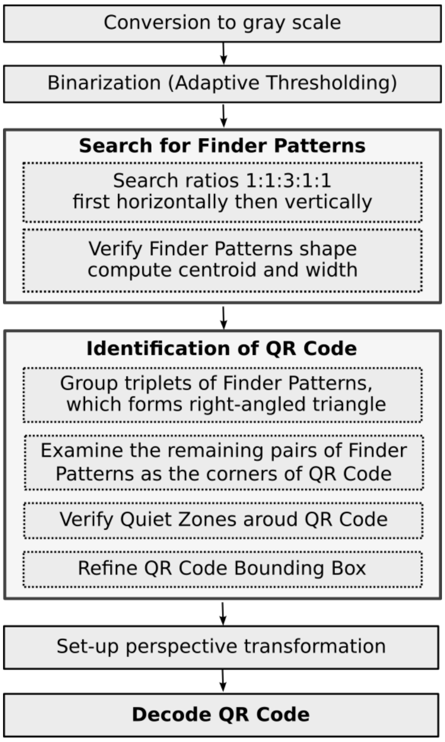
Conversion to gray scale
Binarization (Adaptive Thresholding)
Search for Finder Patterns
Search ratios 1:1:3:1:1 first horizontally then vertically
Verify Finder Patterns shape compute centroid and width
Identification of QR Code
Group triplets of Finder Patterns, which forms right-angled triangle
Examine the remaining pairs of Finder Patterns as the corners of QR Code
Verify Quiet Zones aroud QR Code
Refine QR Code Bounding Box
Set-up perspective transformation
Decode QR Code

Figure 4. The flow chart of proposed algorithm.

Before searching for a QR Code, the original image (maybe colored) is converted to a gray scale image using Equation (1), because the color information does not bear any significant additional information that might help in QR Code recognition.

$$I = \frac{77R + 151G + 28B}{256} \tag{1}$$

where I stands for gray level and R, G, B for red, green, and blue color intensities of individual pixels in the RGB model, respetively. This RGB to gray scale conversion is integer approximation of widely used luminance calculation as defined in recommendation ITU-R BT.601-7:

$$I = 0.299R + 0.587G + 0.114B \tag{2}$$

Next, the gray scaled image is converted to a binary image using modified adaptive thresholding (with the size of window 35—the window size we choose to be at least five times the size of expected size of QRC module) [19]. We expect that black points, which belong to QRC, will become foreground points.

We use the modification of the well-known adaptive thresholding technique (Equation (3)), which calculates individual threshold for every point in the image. This threshold is calculated using average intensity of points under a sliding window. To speed up the thresholding we pre-calculate the integral sum image and we also use the global threshold value (points with intensity above 180 we always consider as background points). Adaptive thresholding can successfully threshold also uneven illuminated images.

$$B(x,y) = \begin{cases} 0 & \leftarrow \quad I(x,y) > 180 \\ 0 & \leftarrow \quad I(x,y) >= T(x,y) \\ 1 & \leftarrow \quad I(x,y) < T(x,y) \end{cases} \tag{3}$$

$$T(x,y) = m(x,y) - \frac{I(x,y)}{10} - 10$$

$$m(x,y) = \frac{1}{35 \times 35} \sum_{i=-17}^{17} \sum_{j=-17}^{17} I(x+i, y+j)$$

where I is gray scale (input) image, B is binary (output) image, T is threshold value (individual for each pixel at coordinates x, y), and m is average of pixel intensities under sliding window of the size 35×35 pixels.

In order to improve adaptive thresholding results, some of the image pre-processing techniques, such as histogram equalization, contrast stretching or deblurring, are worth to consider.

3.1. Searching for Finder Patterns

First, the binary image is scanned from top to bottom and from left to right, and we look for successive sequences of black and white points in a row matching the ratios 1:1:3:1:1 (W_1:W_2:W_3:W_4:W_5 where W_1, W_3, W_5 indicates the number of consecutive black points which are alternated by W_2, W_4 white points) with small tolerance (tolerance is necessary due to imperfect thresholding and noise in the Finder Pattern area, black and white points in a line do not alternate in ideal ratios 1:1:3:1:1):

$$\begin{aligned} & W_1, W_2, W_4, W_5 \in \langle w - 1.5, w + 2.0 \rangle \\ & W_3 \in \langle 3w - 2, 3w + 2 \rangle \qquad\qquad , \text{ where} \\ & W_3 \geq \max(W_1 + W_2, W_4 + W_5) \end{aligned} \tag{4}$$

$$w = \frac{W_1 + W_2 + W_3 + W_4 + W_5}{7} = \frac{W}{7}$$

For each match in a row coordinates of Centroid (C) and Width ($W = W_1 + W_2 + W_3 + W_4 + W_5$) of the sequence (of black and white points) are stored in a list of Finder Pattern candidates (Figure 5).

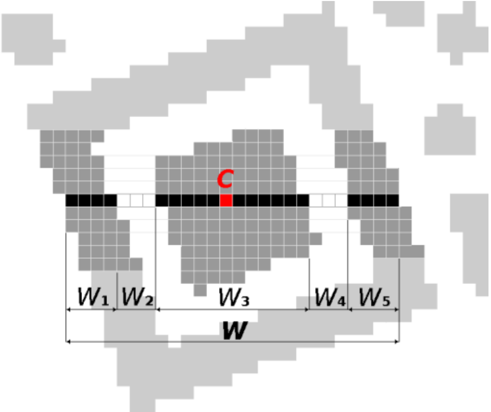

Figure 5. A Finder Pattern candidate matching 1:1:3:1:1 horizontally.

Then, Finder Pattern candidates (from the list of candidates) that satisfy the following criteria are grouped:

- their centroids C are at most $3/7W$ points vertically and at most 3 points horizontally away from each other,
- their widths W does not differ by more than 2 points.

We expect, that the Finder Pattern candidates in one group belong to the same Finder Pattern and therefore we set the new centroid C and width W of the group as average of x, y coordinates and widths of the nearby Finder Pattern candidates (Figure 6a).

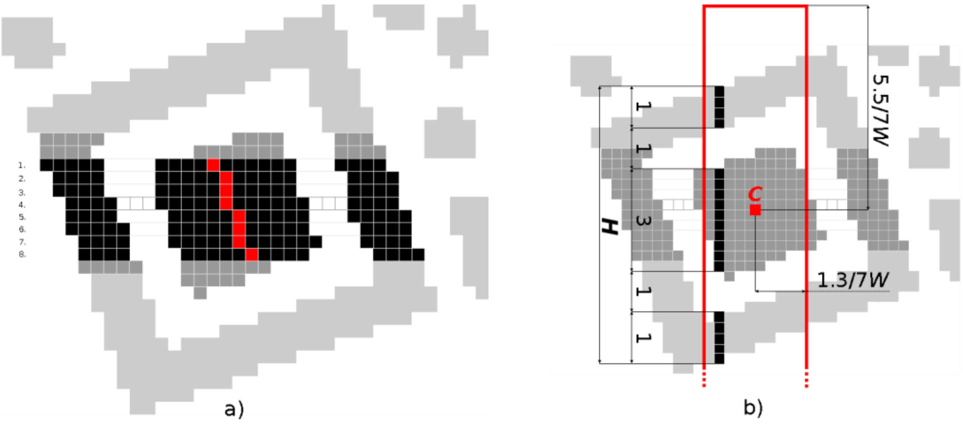

Figure 6. (**a**) Group of 8 Finder Pattern candidates matching 1:1:3:1:1 in rows, (**b**) Finder Pattern candidate matching 1:1:3:1:1 vertically.

After grouping the Finder Patterns, it must be verified whether there are sequences of black and white points also in the vertical direction, which alternate in the ratio 1:1:3:1:1 (Figure 6b). A bounding box around the Finder Pattern candidate, in which vertical sequences are looked for, is defined as

$$x \in \langle C_x \pm 1.3/7W \rangle,\ y \in \left\langle C_y \pm 5.5/7W \right\rangle \tag{5}$$

where C (C_x, C_y) is a centroid and W is width of the Finder Pattern candidate. We work with a slightly larger bounding box in case the Finder Pattern is stretched vertically. Candidates, where no vertical match is found or where the ratio $H/W < 0.7$, are rejected. For candidates, where a vertical match is found, the y coordinate of centroid C (C_y) is updated as an average of y coordinates of centers of the vertical sequences.

3.2. Verification of Finder Patterns

Each Finder Pattern consists of a central black square with the side of 3 units (R_1), surrounded by a white frame with the width of 1 unit (R_2), surrounded by a black frame with the width of 1 unit (R_3). In Figure 7 there are colored regions R_1, R_2 and R_3 in red, blue, and green, respectively. For each Finder Pattern candidate Flood Fill algorithm is applied, starting from the centroid C (which lies in the region R_1) and continuing through white frame (region R_2) to black frame (region R_3). As continuous black and white regions are filled, following region descriptors are incrementally computed:

- area ($A = M_{00}$)
- centroid ($C_x = M_{10}/M_{00}$, $C_y = M_{01}/M_{00}$), where M_{00}, M_{10}, M_{00} are raw image moments
- bounding box (Top, Left, Right, Bottom)

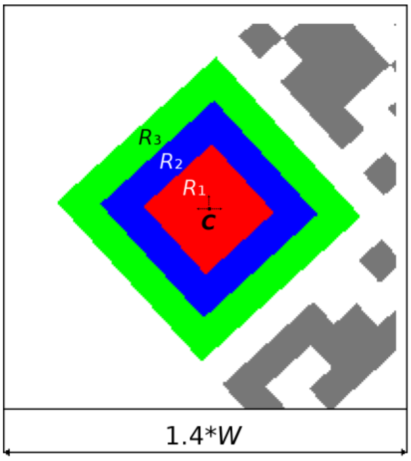

Figure 7. Regions of the Finder Pattern candidate.

The Finder Pattern candidate, which does not meet all of the following conditions, is rejected.

- Area(R_1) < Area(R_2) < Area(R_3) and
  1.1 < Area(R_2)/Area(R$_1$) < 3.4 and
  1.8 < Area(R_3)/Area(R_1) < 3.9
- 0.7 < AspectRatio(R_2) < 1.5 and
  0.7< AspectRatio(R_3) < 1.5
- |Centroid(R_1), Centroid(R_2)| < 3.7 and
  |Centroid(R_1), Centroid(R_3)| < 4.2

Note: the criteria were set to be invariant to the rotation of the Finder Pattern, and the acceptance ranges were determined experimentally. In an ideal undistorted Finder Pattern, the criteria are met as follows:

- Area(R_2)/Area(R_1) = 16/9 = 1.8 and Area(R_3)/Area(R_1) = 24/9 = 2.7
- AspectRatio(R_1) = 1 and AspectRatio(R_2) = 1 and AspectRatio(R_3) = 1
- Centroid(R_1) = Centroid(R_2) = Centroid(R_3)

In real environments there can be damaged Finder Patterns. The inner black region R_1 can be joined with the outer black region R_3 (Figure 8a) or the outer black region can be interrupted or incomplete (Figure 8b). In the first case the bounding box of the region R_2 is completely contained by bounding box of the region R_1 and in the second case is bounding box of the region R_3 contained in bounding box of the region R_2. These cases are handled individually. If the first case is detected, then the region R_1 is proportionally divided into R_1 and R_3 and if the second case is detected then the region R_2 is instantiated using the region R_1 and R_3.

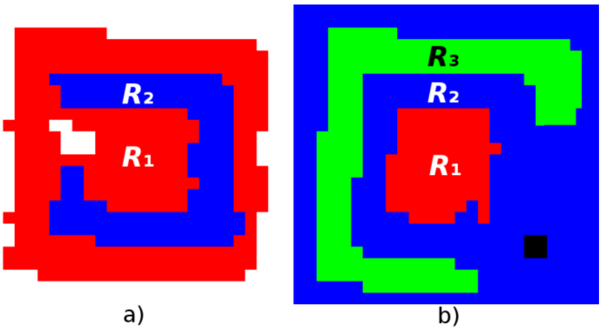

Figure 8. Various damages of Finder Patterns: (**a**) merged inner and outer black regions; (**b**) interrupted outer black region.

The Centroid (C) and Module Width (*MW*) of the Finder Pattern candidate are updated using the region descriptors as follows:

$$C = \left(\frac{M_{10}(R_1) + M_{10}(R_2) + M_{10}(R_3)}{M_{00}(R_1) + M_{00}(R_2) + M_{00}(R_3)}, \frac{M_{01}(R_1) + M_{01}(R_2) + M_{01}(R_3)}{M_{00}(R_1) + M_{00}(R_2) + M_{00}(R_3)}\right) \tag{6}$$

$$MW = \sqrt{M_{00}(R_1) + M_{00}(R_2) + M_{00}(R_3)}/7$$

3.3. Grouping of Finder Patterns

3.3.1. Grouping Triplets of Finder Patterns

In the previous steps, Finder Patterns in the image were identified and now such triplets (from the list of all Finder Patterns) must be selected, which can represent 3 corners of a QR Code. Matrix of the distances between the centroid of all Finder Patterns is build and all 3-element combinations of all Finder Patterns are examined. For each triplet it is checked whether it is possible to construct a right-angled triangle from it so that the following conditions are met:

- the size of each triangle side must be in predefined range
- the difference in sizes of two legs must be less than 21
- the difference in size of the real and theoretical hypotenuse must be less than 12

In this way, a list of QR Code candidates (defined by a triplet FP_1, FP_2, FP_3) is built. However, such a candidate for a QR Code is selected only on the basis of the mutual position of the 3 FP candidates. As is shown in Figure 9 not all QR Code candidates are valid (dotted red $FP_{3'}$-$FP_{3''}$-$FP_{2''}$ is false positive). These false positive QR Code candidates will be eliminated in the next steps.

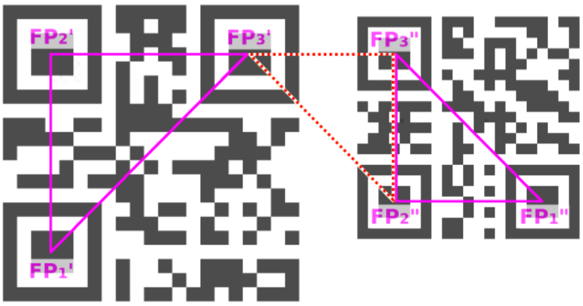

Figure 9. QR Code candidates.

Finally, Bottom-Left and Top-Right Finder Pattern from the triplet (FP_1, FP_2, FP_3) is determined by using formula:

If (FP_3.x − FP_2.x)(FP_1.y − FP_2.y) − (FP_3.y − FP_2.y)(FP_1.x − FP_2.x) < 0 then Bottom-Left is FP_3 and Top-Right is FP_1 else vice versa.

3.3.2. Grouping Pairs of Finder Patterns

If any QR Code has one of the 3 Finder Patterns significantly damaged, then this Finder Pattern might not be identified and there remains two Finder Patterns in the Finder Patterns list, that were not selected (as the vertices of a right-angled triangle) in the previous step (Figure 10a). The goal of this step is to identify these pairs and determine the position of the third missing Finder Pattern. A square shape of the QR Code is assumed.

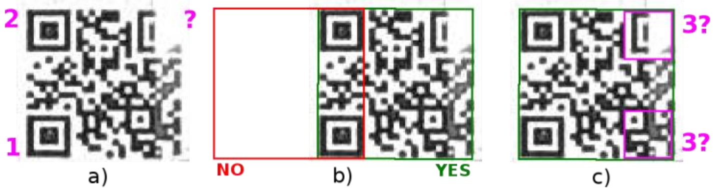

Figure 10. QR Code with one damaged Finder Pattern: (**a**) two known Finder Patterns; (**b**) two possible regions of QR Code; (**c**) two possible positions of 3rd Finder Pattern.

All two element combinations of remaining Finder Patterns, whose distance is in a predefined interval, are evaluated. A pair of Finder Patterns can represent Finder Patterns that are adjacent corners of a QR Code square (Figure 10a) or are in the opposite corners. If they are adjacent corners, then there are two possible positions of the QR Code (in Figure 10b depicted as a red square and green square). If they are opposite corners, then there are other two possible positions of the QR Code.

All four possible positions of the potential QR Code are evaluated against the following criteria:

- Is there a Quiet Zone around the bounding square at least 1 *MW* wide?
- Is there a density of white points inside bounding square in interval (0.4; 0.65)?
- Is there a density of edge points inside bounding square in interval (0.4; 0.6)?

Density of edge points is computed as the ratio of the number of edge points to area*2/*MW*.

Region of a QR Code is expected to have relative balanced density of white and black points and relative balanced ratio of edges to area.

A square region that meets all the above conditions is considered a candidate for a QR Code. There are two possible corners in the QR Code bounding square where 3rd Finder Pattern can be located (Figure 10c). For both possible corners Finder Pattern match score is computed and one with better score is selected (in other words question, "In which corner is the structure that more closely resembles the ideal Finder Pattern?" must be answered). Match score is computed as

$$MS = \operatorname{argmin}(OS + \min(BS, WS)) \tag{7}$$

where *MS* is match score (lower is better), *OS* is overall pattern match score, *BS* is black module match score and *WS* is white module match score. *BS* stores matches only for expected black points and *WS* stores matches only for expected white points between the mask and the image. *BS* and *WS* was introduced to handle situations when over the area of Finder Pattern is placed black or white spot, which would cause a low match score if only a simple pattern matching technique were used.

The match score is computed for several Finder Pattern mask positions by moving the mask in a spiral from its initial position up to a radius of *MW* with a step of *MW*/2 (for the case of small geometric deformations of the QR Code).

3.4. Verification of Quiet Zone

According to ISO standard a Quiet Zone is defined as "a region 4X wide which shall be free of all other markings, surrounding the symbol on all four sides". So, it must be checked if there are only white points in the image in the rectangular areas wide 1*MW* which is parallel to line segments defined by FP_1–FP_2 and FP_2–FP_3 (Figure 11). For fast scanning of the rectangle points Bresenham's line algorithm is utilized [20].

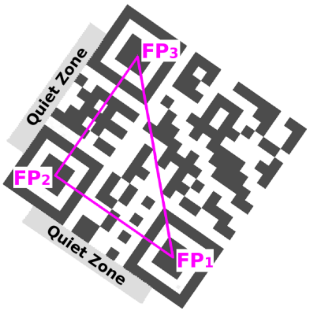

Figure 11. Quiet Zones.

QR Code candidates which do not have quiet zones around are rejected. Rejected are also QR Code candidates whose outer bounding box (larger) contains outer bounding box (smaller) of another QR Code candidate.

3.5. QR Code Bounding Box

Centroids of the 3 Finder Patterns (which represent the QR Code candidate) are the vertices of the triangle FP_1–FP_2–FP_3. This inner triangle must be expanded to outer triangle P_1–P_2–P_3, where the arms of the triangle pass through boundary modules of the QR Code (Figure 12). For instance, the shift of FP_3 to P_3 may be expressed as

$$P_3 = FP_3 + \frac{FP_3 - FP_1}{|FP_3, FP_1|} MW\sqrt{18} \tag{8}$$

where *MW* is module width (Equation (6))

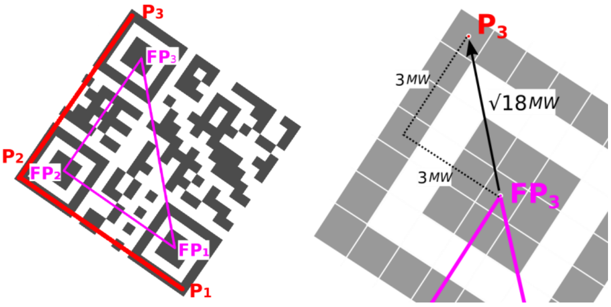

Figure 12. Bounding Box.

3.6. Perspective Distortion

For perspective undistorted QR Codes (only shifted, scaled, rotated, or sheared) it is sufficient to have only 3 points to set-up the affine transformation from a square to destination parallelogram. However, for perspective (projective) distorted QR Codes 4 points are required to set-up perspective transformation from a square to destination quadrilateral [21].

Some authors (for example [7,22]) search for Alignment Pattern to obtain 4th point. However, version 1 QR Codes does not have Alignment Pattern, so we have decided not to rely on Alignment Patterns.

Instead of that we use an iterative approach to find the opposite sides to P_2–P_1 and P_2–P_3, which aligns to QR Code borders.

1. We start from initial estimate of P_4 as an intersection of the line L_1 and L_3, where L_1 is parallel to P_2–P_3 and L_3 is parallel to P_2–P_1 (Figure 13a).
2. We count the number of pixels which are common to the line L_3 and the QR Code for each third of the line L_3 (Figure 13a).
3. We shift the line L_3 by width of one module away from the QR Code and again we count the number of pixels which are common to the shifted line $L_{3'}$ and the QR Code. The module width is estimated as *MW* (from Equation (6)) (Figure 13b).
4. We compare overlaps of the line L_3 from the step 2 and 3, and

   a. If L_3 was whole in the QR Code and the shifted $L_{3'}$ is out of the QR Code, then initial estimation of P_4 is good and we end.
   b. If L3 was whole in the QR Code and 3rd third of the shifted $L_{3'}$ is again in the QR Code, then we continue by step 5.
   c. If 3rd third of L_3 was in quiet zone and 2nd and 3rd third of the shifted $L_{3'}$ is in the quiet zone or if 2nd third of L_3 was in the quiet zone and 1st and 2nd third of the shifted $L_{3'}$ is in the quiet zone then we continue by step 6.

5. We start to move P_4 end of line segment P_3–P_4 away from the QR Code until 3rd third of L_3 touches the quiet zone (Figure 13c).
6. We start to move P_4 end of line segment P_3–P_4 towards the QR Code until 3rd third of L_3 touches the QR Code.
7. We apply the same procedure also to the line L_1 like for the line $L_{3.}$
8. The intersection of the shifted lines L_1 and L_3 is a new P_4 position.

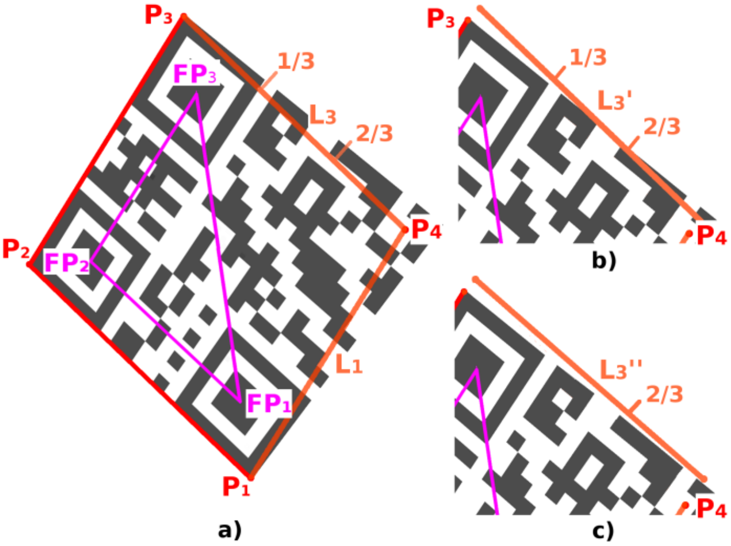

Figure 13. Perspective distortion: (**a**) initial estimate of the point P_4 and lines L_1, L_3; (**b**) first shift of the line L_3; (**c**) second shift of the line L_3.

Once the position of 4th point, P_4, is obtained, perspective transformation from the source square representing the ideal QR Code to destination quadrilateral representing the real QR Code in the image can be set-up (Figure 14).

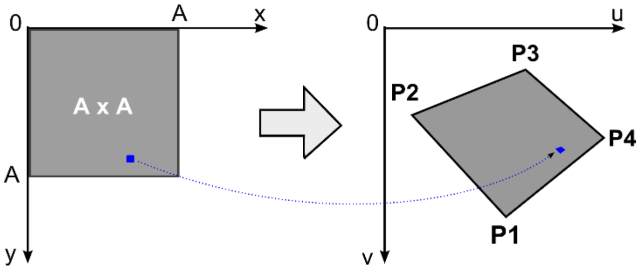

Figure 14. Perspective transformation.

Using equations [21]:

$u = \frac{ax+by+c}{gx+hy+1}$, $v = \frac{dx+ey+f}{gx+hy+1}$, where the transformation coefficients can be calculated from the coordinates of the points $\mathrm{P}_1(u_1, v_1)$, $\mathrm{P}_2(u_2, v_2)$, $\mathrm{P}_3(u_3, v_3)$, $\mathrm{P}_4(u_4, v_4)$ as

$$\begin{aligned}
&a = (u_3 - u_2)/A + gu_3,\ b = (u_1 - u_2)/A + hu_1,\ c = u_2 \\
&d = (v_3 - v_2)/A + gv_3,\ e = (v_1 - v_2)/A + hv_1,\ f = v_2 \\
&g = \frac{\begin{vmatrix} du_3 & du_2 \\ dv_3 & dv_2 \end{vmatrix}}{\begin{vmatrix} du_1 & du_2 \\ dv_1 & dv_2 \end{vmatrix}},\ h = \frac{\begin{vmatrix} du_1 & du_3 \\ dv_1 & dv_3 \end{vmatrix}}{\begin{vmatrix} du_1 & du_2 \\ dv_1 & dv_2 \end{vmatrix}} \\
&du_1 = (u_3 - u_2)A,\ du_2 = (u_1 - u_4)A,\ du_3 = u_2 - u_3 + u_4 - u_1 \\
&dv_1 = (v_3 - v_2)A,\ dv_2 = (v_1 - v_4)A,\ dv_3 = v_2 - v_3 + v_4 - v_1
\end{aligned}$$

It sometimes happens, that the estimate of P_4 position is not quite accurate so we move P_4 in spiral from its initial position (obtained in previous step) and we calculate match score of bottom-right Alignment Pattern (Alignment Pattern exists only in QR codes version 2 and above). For each shift we recalculate coefficients for the perspective transformation, and we recalculate also match score. For the given version of the QR Code we know the expected position and size of bottom-right Alignment Pattern so we can calculate match between expected and real state. The final position P_4 is the one with the highest match score.

Another alternative method how to handle perspective distorted QR Codes and how to determine position of the P_4 point is based on edge directions and edge projections analysis [23].

3.7. Decoding of a QR Code

A QR Code is 2D square matrix in which the dark and light squares (modules) represent bits 1 and 0. In fact, each such a module on the pixel level is usually made up of cluster of adjacent pixels. In a QR Code decoding process, we have to build a 2D matrix that has elements with a value of 1 or 0 (Figure 15).

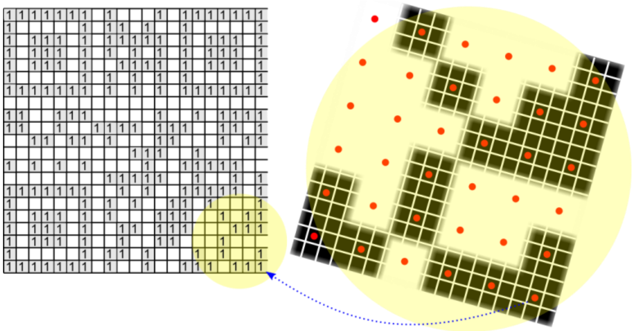

Figure 15. Transformation of the QR Code into the binary grid.

The pixels, where the brightness is lower than the threshold, are declared as 1 and the others are declared as 0. When analyzing such a cluster of pixels (module), the central pixel of the module plays a decisive role. If the calculated position of the central pixel does not align to integer coordinates in the image, the brightness is determined by bilinear interpolation.

Once the binary matrix of 1 and 0 is created, the open-source ZBar library [24] can be used to start the final decoding of the binary matrix and to receive the original text encoded by the QR Code.

4. Results

We used a test dataset of 595 QR Code samples to verify the method described in this paper. The testing dataset contained 25 artificial QR codes of different sizes and rotations, 90 QR Codes from Internet images, and 480 QR Codes from a specific industrial process. Several examples of the testing samples are in Figure 16.

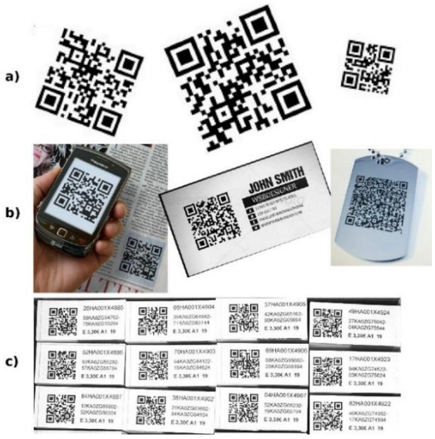

Figure 16. Testing samples: (**a**) artificial, (**b**) Internet, (**c**) industrial.

In Table 1 and Figure 17 there are our results compared to competing QR Code decoding solutions (commercial and also open-source). In the table are the numbers of correctly decoded QR Codes from the total number of 595 QR Codes.

Table 1. Comparison of competing solutions with our proposed method.

Solution	Artificial Samples	Internet Samples	Industrial Samples
ZXing (open-source) [25]	2	72	48
Quirc (open-source) [22]	12	69	45
LEADTOOLS QR Code SDK [26]	9	72	147
ZBar (open-source) [24]	23	84	398
Inlite Barcode Reader SDK [27]	25	80	421
DataSymbol Barcode Reader SDK [28]	25	89	471
Dynamsoft Barcode Reader SDK [29]	25	88	478
Our solution	25	87	480

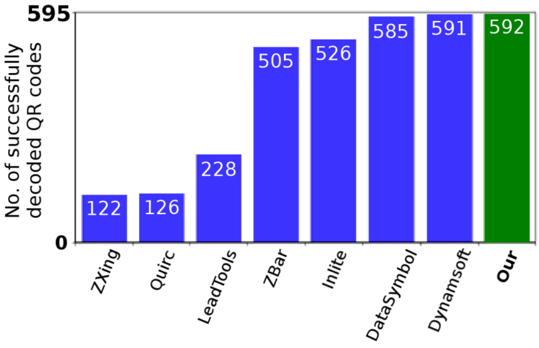

Figure 17. Comparison of the recognition rate of our method against competing solutions.

Our method successfully detected all QR Codes but failed to decode three samples. Two samples were QR Codes placed on a bottle, where perspective and cylindrical distortions were combined. How to deal with this type of combined distortion is a challenge for future research.

As the commercial solutions have closed source code, we performed the testing using the black-box method. We have compiled our own QR Code test dataset (published together with this article under "Supplementary Material") to evaluate and compare our method, as a standardized universal dataset is not publicly available.

In Table 2 the computational complexity of our algorithm is compared to competing open-source solutions (commercial solutions were tested online). Our algorithm was implemented in Free Pascal and tests were run on an i5-4590 3.3GHz CPU (Intel Corporation, Santa Clara, CA, USA).

Table 2. Dependence of computational complexity on image resolution and number of QR codes in an image.

Solution	**1296 × 960**		**2592 × 1920**		
	1 Code	**10 Codes**	**1 Code**	**10 Codes**	**50 Codes**
ZBar (open-source)	85 ms	185 ms	347 ms	372 ms	1744 ms
Quirc (open-source)	13 ms	26 ms	45 ms	130 ms	493 ms
Our solution	18 ms	21 ms	73 ms	77 ms	107 ms

We see the main contribution of our method in the way the broken Finder Pattern is dealt with (Section 3.3.2) and how the QR Code bounding box is determined (Section 3.6), especially for perspective distorted QR Codes. The presented method can still locate a QR Code if one of the three Finder Patterns is significantly damaged. Consecutive tests in the real manufacturing process showed that this situation occurs much more often than a situation where two or three opposite Finder Patterns are damaged. In order to detect a QR Code with multiple damage Finder Patterns, it will be necessary to combine Finder Pattern based localization with the region based localization.

5. Conclusions

We have designed and tested a computationally efficient method for precise location of 2D QR Codes in arbitrary images under various illumination conditions. The proposed method is suitable for low-resolution images as well as for real time processing. The designed Finder Pattern based localization method uses three typical patterns of QR Codes to identify three corners of QR Codes in an image. If one of the three Finder Patterns is so destroyed that it cannot be localized, we have suggested a way to deal with it. The input image is binarized and scanned horizontally to localize the Finder Pattern candidates, which are subsequently verified in order to localize the raw QR Code region. For distorted QR Codes, the perspective transformation is set-up by gradually approaching the boundary of the QR Code.

This method was validated on the testing dataset consisting of a wide variety of samples (synthetic, real world, and specific industrial samples) and it was compared to competing software. The experimental results show that our method has a high detection rate.

The application of QR Codes and their optical recognition has wide use in identification, tracing or monitoring of items in production [30], storing and distribution processes, to aid visually impaired and blind people, to let autonomous robots [31] to acquire context-relevant information, to support authorization during log-in process, to support electronic payments, to increase industrial production surety factor [32], etc.

In cases where is required to place the 2D matrix codes on a very small area it may be preferable to use Data Matrix codes [33].

Supplementary Materials: The following are available online at http://www.mdpi.com/2076-3417/10/21/7814/s1, one ZIP file contains images of QR Codes testing dataset used to evaluate the competing solutions.

Author Contributions: Conceptualization, methodology, software, writing—original draft preparation, L.K.; validation, writing—review and editing, visualization, E.P.; supervision, project administration, funding acquisition, P.B.; All authors have read and agreed to the published version of the manuscript.

Funding: The article is funded of the research project KEGA 013TUKE-4/2019 "Modern educational tools and methods for shaping creativity and increasing the practical skills and habits of graduates of technical departments of universities".

Acknowledgments: The contribution is sponsored by the project.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, X.; Zhu, S.; Wang, Z.; Li, Y. Hybrid visual natural landmark–based localization for indoor mobile robots. *Int. J. Adv. Robot. Syst.* **2018**, *15*. [CrossRef]
2. Bozek, P.; Pokorný, P.; Svetlik, J.; Lozhkin, A.; Arkhipov, I. The calculations of Jordan curves trajectory of the robot movement. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 1–7. [CrossRef]
3. Wang, K.; Zhou, W. Pedestrian and cyclist detection based on deep neural network fast R-CNN. *Int. J. Adv. Robot. Syst.* **2018**, *16*. [CrossRef]
4. Zhang, X.; Gao, H.; Xue, C.; Zhao, J.; Liu, Y. Real-time vehicle detection and tracking using improved histogram of gradient features and Kalman filters. *Int. J. Adv. Robot. Syst.* **2018**, *15*. [CrossRef]
5. Denso Wave Incorporated. What is a QR Code? 2018. Available online: http://www.qrcode.com/en/about/ (accessed on 6 September 2018).
6. Denso Wave Incorporated. History of QR Code. 2018. Available online: http://www.qrcode.com/en/history/ (accessed on 6 September 2018).
7. Lin, J.-A.; Fuh, C.-S. 2D Barcode Image Decoding. *Math. Probl. Eng.* **2013**, *2013*, 848276. [CrossRef]
8. Li, S.; Shang, J.; Duan, Z.; Huang, J. Fast detection method of quick response code based on run-length coding. *IET Image Process.* **2018**, *12*, 546–551. [CrossRef]
9. Belussi, L.F.F.; Hirata, N.S. Fast Component-Based QR Code Detection in Arbitrarily Acquired Images. *J. Math. Imaging Vis.* **2012**, *45*, 277–292. [CrossRef]
10. Bodnár, P.; Nyúl, L.G. Improved QR Code Localization Using Boosted Cascade of Weak Classifiers. *Acta Cybern.* **2015**, *22*, 21–33. [CrossRef]
11. Tribak, H.; Zaz, Y. QR Code Recognition based on Principal Components Analysis Method. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 241–248. [CrossRef]
12. Tribak, H.; Zaz, Y. QR Code Patterns Localization based on Hu Invariant Moments. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 162–172. [CrossRef]
13. Sun, A.; Sun, Y.; Liu, C. The QR-code reorganization in illegible snapshots taken by mobile phones. In Proceedings of the 2007 International Conference on Computational Science and its Applications (ICCSA 2007), Kuala Lumpur, Malaysia, 26–29 August 2007; pp. 532–538.
14. Ciążyński, K.; Fabijańska, A. Detection of QR-Codes in Digital Images Based on Histogram Similarity. *Image Process. Commun.* **2015**, *20*, 41–48. [CrossRef]
15. Gaur, P.; Tiwari, S. Recognition of 2D Barcode Images Using Edge Detection and Morphological Operation. *Int. J. Comput. Sci. Mob. Comput. IJCSMC* **2014**, *3*, 1277–1282.
16. Szentandrási, I.; Herout, A.; Dubská, M. Fast detection and recognition of QR codes in high-resolution images. In Proceedings of the 28th Spring Conference on Computer Graphics, Budmerice, Slovakia, 2–4 May 2012; ACM: New York, NY, USA, 2012.
17. Dubská, M.; Herout, A.; Havel, J. Real-time precise detection of regular grids and matrix codes. *J. Real Time Image Process.* **2016**, *11*, 193–200. [CrossRef]
18. Sörös, G.; Flörkemeier, C. Blur-resistant joint 1D and 2D barcode localization for smartphones. In Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia, MUM, Lulea, Sweden, 2–5 December 2013; pp. 1–8.
19. Bradley, D.; Roth, G. Adaptive Thresholding using the Integral Image. *J. Graph. Tools* **2007**, *12*, 13–21. [CrossRef]
20. Bresenham, J.E. Algorithm for computer control of a digital plotter. *IBM Syst. J.* **1965**, *4*, 25–30. [CrossRef]
21. Heckbert, P. Fundamentals of Texture Mapping and Image Warping. Master's Thesis, University of California, Berkeley, CA, USA, June 1989.

22. Beer, D. Quirc—QR Decoder Library. 2018. Available online: https://github.com/dlbeer/quirc (accessed on 22 September 2018).
23. Karrach, L.; Pivarčiová, E.; Božek, P. Identification of QR Code Perspective Distortion Based on Edge Directions and Edge Projections Analysis. *J. Imaging* **2020**, *6*, 67. [CrossRef]
24. Terriberry, T.B. ZBar Barcode Reader. 2018. Available online: http://zbar.sourceforge.net (accessed on 22 September 2018).
25. Google. ZXing (Zebra Crossing) Barcode Scanning Library for Java, Android. 2018. Available online: https://github.com/zxing (accessed on 22 September 2018).
26. Leadtools. QR Code SDK Technology. 2018. Available online: http://demo.leadtools.com/JavaScript/Barcode/index.html (accessed on 22 September 2018).
27. Inlite Research Inc. Barcode Reader SDK. 2018. Available online: https://online-barcode-reader.inliteresearch.com (accessed on 22 September 2018).
28. DataSymbol. Barcode Reader SDK. 2018. Available online: http://www.datasymbol.com/barcode-reader-sdk/barcode-reader-sdk-for-windows/online-barcode-decoder.html (accessed on 22 September 2018).
29. Dynamsoft. Barcode Reader SDK. 2018. Available online: https://www.dynamsoft.com/Products/Dynamic-Barcode-Reader.aspx (accessed on 22 September 2018).
30. Bako, B.; Božek, P. Trends in Simulation and Planning of Manufacturing Companies. *Procedia Eng.* **2016**, *149*, 571–575. [CrossRef]
31. Frankovsky, P.; Pastor, M.; Dominik, L.; Kicko, M.; Trebuna, P.; Hroncova, D.; Kelemen, M. Wheeled mobile robot in structured environment. In Proceedings of the 12th International Conference ELEKTRO 2018, Mikulov, Czech Republic, 21–23 May 2018; pp. 1–5.
32. Pivarčiová, E.; Božek, P. Industrial production surety factor increasing by a system of fingerprint verification. In Proceedings of the 2014 International Conference on Information Science, Electronics and Electrical Engineering (ISEEE 2014), Sapporo, Japan, 26–28 April 2014; pp. 493–497.
33. Karrach, L.; Pivarčiová, E.; Nikitin, Y.R. Comparing the impact of different cameras and image resolution to recognize the data matrix codes. *J. Electr. Eng.* **2018**, *69*, 286–292. [CrossRef]

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.