QR Code-based Efficient Entry Method for Intelligent Files of Internet of Things

Wang Genwang[1,2(✉)]

[1] School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China
18120130@bjtu.edu.cn
[2] Key Laboratory of Communication and Information Systems, Beijing Municipal Commission of Education, Beijing, China

Abstract. In recent years, with the development and application of technologies such as the Internet of Things, cloud computing, and big data, file management, the originally complicated task, has become more intelligent and flexible. However, when recording files, the existing technology still uses the method of scanning one file at a time. Therefore, when massive file information needs to be recorded at the same time, additional time cost or equipment cost will be incurred. In order to shorten the time of file entry as much as possible and reduce unnecessary equipment costs, we propose to mark the key information of files with the QR code technology in the Internet of Things, and then design a special image segmentation method according to the characteristics of the file bag or file box. Thus, the segmentation and extraction of different archive information within the visual range of the input device can be realized, and the effect of simultaneously inputting multiple archive information can be achieved. The experimental results show that in an ideal situation, our method can input the information of 20 files at one time, which greatly improves the efficiency of file entry. It is expected that if this method is applied to the current file management system, it can solve the problem of time consuming and energy consuming, and realize a more intelligent file management system at the same time.

Keywords: QR code · File management · Internet of Things

1 Introduction

In recent years, with the development of information technology, archive management has gradually shifted from traditional manual management to intelligence. However, due to the limitations of archives, some file archives cannot be completely replaced by electronic archives. Therefore, there is an urgent need for a solution to achieve an efficient and intelligent file entry program while retaining complete file information. The Internet of Things [1], as a network of intelligent identification, monitoring, and management, has been widely applied in the fields of medical treatment [2], transportation [3] and logistics [4]. Therefore, if the Internet of Things technology can be applied to file management, a smart and efficient file management system can be constructed. QR code [5], as one

Y.-B. Lin and D.-J. Deng (Eds.): SGIoT 2020, LNICST 354, pp. 195–207, 2021.
https://doi.org/10.1007/978-3-030-69514-9_16

of the key technologies of Internet of Things, has been applied in smart library [6], smart locker [7], electronic payment and other fields in early years because of its fast decoding, high data storage, strong error correction ability and other characteristics, and has achieved very good results. Therefore, applying it to an intelligent file management system also has a certain application basis. The main work of the current file management system is mostly aimed at efficient retrieval and safe storage after electronic files have been obtained, while ignoring the necessity of traditional paper files in the process of converting paper files into electronic files. The time cost and equipment cost required, so there is still some room for optimization.

Aiming at the disadvantages of low input efficiency and high cost of the existing archive management system, this article makes full use of the advantages of the QR code and saves the key information of the file in the QR code. After that, multiple files were scanned at once, and images containing multiple files were obtained. In the process of extracting image information, referring to the common methods in face recognition in the CV field, the RGB image is converted into YCrCb image, and the Cr channel and Cb channel are used cleverly to eliminate the influence of uneven illumination when some images are collected. Thereby improving the accuracy of extracting information from multiple archives. In addition, when extracting the QR code containing the archive information, the binary image of multiple channels is merged, which eliminates the tiny interference areas and improves the efficiency of multiple archive information recognition. Finally, the transmission transformation is used to correct the distorted QR code containing the file information, and the information of multiple files is decoded. The experimental results show that our method can simultaneously enter the information of multiple files at one time, which greatly reduces the time for file entry. It is expected that this method can be applied to the existing file management system to make the existing file management system more efficient and intelligent.

The remainder of this paper is divided into four sections. Section 2 describes the related work. In Sect. 3 of this paper, we introduce the specific program implementation. The fourth section is the experiment and analysis of the program. Finally, we have summarized the work of this paper and made further prospects and improvement ideas for our method.

2 Related Work

File management system, as an intelligent and efficient solution for file management, has always been one of the focuses of research. In the context of the era of big data and cloud computing, Sui et al. [8] analyzed the problems existing in the archives of university endowment funds, and put forward suggestions for funding to improve the efficiency of archive utilization. Considering the prospect of cloud computing in the aspects of electronic archives management infrastructure construction, archives business platform construction and archives utilization, an electronic archives cloud platform based on Hadoop [9] was built to realize data information exchange and business collaboration. On the other hand, various new frameworks and technologies have also been applied to the construction of archive management systems. A device file management system [10] was constructed based on Struts2 + Hibernate + SiteMesh. This system uses a bridge-based

page display method to reduce the coupling between programs, and has the advantages of good scalability and easy maintenance. Wang et al. [11] combined with the advantages of Ajax and ASP.NET technology, so as to make the file management system more reliable. In addition, some customized file management systems have been developed to take into account the functional differences between different agencies and units in file management. In view of the fact that the existing student file management system cannot solve high concurrent access, a hybrid database and message queue mechanism [12] is introduced into the student electronic file management system, which reduces the cost of system deployment and maintenance. Taking into account the need for confidentiality of file management in public security agencies, some encryption mechanisms [13] have been added to file management to ensure the security of files. In rural areas, it takes a lot of time to back up the electronic archives of land contracts. Therefore, Xu et al. [14] proposes a solution that can use multithreading and distributed storage to improve the efficiency of data sealing. Our work is aimed at the low efficiency of converting paper files into electronic files, and we have designed an appropriate method to improve the efficiency of file entry.

3 Our Approach

In this section, we will describe the design details of an efficient file entry method.

In the process of program design, we used the C++ programming language. In order to simplify our program, we used the open source image processing library OpenCV3.2 to locate the key areas in the captured image, and the open source barcode recognition library ZBar to extract the file information. According to different functions, we divide the designed program into four modules. Figure 1 shows the basic architecture of the program we designed.

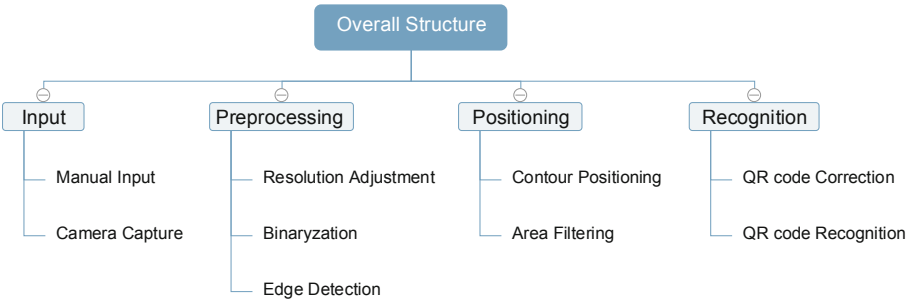

Fig. 1. Overall structure of the program

Considering that the image input and the QR code recognition can directly call the camera or manually input the image and call the QR code recognition library, so only the other two modules will be described in detail here.

3.1 Input

First, we extract the key information of the archive and encode it with a QR code (if security is required, a suitable encryption method can be selected). Then, attach the obtained QR code to the surface of the original file. In order to improve the efficiency of collection, multiple files are overlapped and the QR code information of each file is not blocked. After that, multiple files were scanned at the same time to complete the collection of information. We provide two input methods, automatic scanning and manual input, to facilitate subsequent function expansion.

Figure 2 shows the basic effect of archival information collection.

Fig. 2. An example of image acquisition

3.2 Preprocessing

Resolution Adjustment. Due to the high resolution of the input image, if it is processed directly, more CPU resources and time will be consumed. In order to reduce the time consumed by subsequent operations and improve the efficiency of the program, the resolution of the original image is adjusted. In this way, on the one hand, the amount of image data is reduced, and on the other hand, some tiny noises in the image are eliminated

Among the commonly used image scaling algorithms, there are three types: nearest interpolation, bilinear interpolation, and bicubic interpolation. Here we have tested them separately. The results show that bilinear interpolation achieves a good balance between time consumption and interpolation effect, so bilinear interpolation is used to reduce noise in the original image.

Here is a brief introduction to bilinear interpolation. For an image with a resolution of $w \cdot h$, if we want to change its resolution to $w\prime \cdot h\prime$, we use the nearest four points in

the coordinates $P(x, y)$ in the original image to calculate the target point. $P\prime(x', y')$ pixel value. The specific practices are as follows:

Let Q_{11}, Q_{12}, Q_{21} and Q_{22} be the nearest four integer points from the distance$P(x, y)$. First, we interpolate in the x direction to get the pixels at R_1, as shown in Eq. (1).

$$f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \tag{1}$$

Similarly, we can get the pixels at R_2, as shown in Eq. (2).

$$f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \tag{2}$$

After that, we use the coordinates of R_1 and R_2 to perform interpolation processing in the y direction, and then the pixel value of the target point $P\prime$ can be obtained, as shown in Eq. (3).

$$f(x, y) = \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} (x, y_2) \tag{3}$$

Resolution adjustment can reduce redundant pixels, but it can also make the reduced image appear jagged. To solve this problem, we use mean filter, median filter and Gaussian filter to convolve the compressed image. Among them, compared with other methods, Gaussian filtering can preserve the edges of the image well and has lower distortion.

Binarization. Binarization is an important step of preprocessing. The quality of the processing result determines whether the QR code area and interference area can be accurately located. Since the foreground and background of the RGB image involved in this scene have good discrimination (black and white and other colors), the OTSU algorithm is suitable. The binary image we hope to obtain satisfies the following conditions: the separated QR code area has a regular shape and does not adhere to the background; the color of the QR code area is consistent with the surrounding area; and the interference area is as few as possible.

The commonly used binarization methods mainly include two categories. The first type is the global thresholding method. This method uses the same threshold for the entire image and has low complexity such as the OTSU method. Consequently, the foreground and background of the preprocessed image are also need to be obviously different; the second type of the methods is the adaptive thresholding method, which is also called a local thresholding method. When an image is processed, the thresholds of different coordinates are dissimilar, and thus it is suitable for processing complex backgrounds. Correspondingly, the complexity of the algorithm will be higher.

Here we briefly introduce the principle of commonly used local binarization.

First, for any point P in the image, its pixel is I. We take a neighborhood of the point, calculate the weighted average m of all point pixels in the domain, and take the difference as n. Then compare the size of I and $C = m - n$, so as to determine whether the pixel after the point P binarization takes 0 or 255.

What matters is that we can calculate the weights of the points in the neighborhood in two ways: one is to give the same weight to each point, and the other is to use the Gaussian convolution kernel to give different weight to each point according to the distance from each point to the center point.

Since both types of algorithms can only process single-channel pictures, we need to convert the original RGB image into a grayscale image. In this process, there are maximum value method, average value method, weighted average method, and so on.

Here we compare the traditional method with our method to show why our method works better.

Traditional Processing Methods. First, the image is converted to a grayscale image. Here, we choose a weighted average method and it works like Eq. (4):

$$Gray = 0.3R + 0.59G + 0.11B \tag{4}$$

After that, the OTSU algorithm and two adaptive threshold algorithms are used to obtain the binary image, as shown in Fig. 3, Fig. 4, Fig. 5:

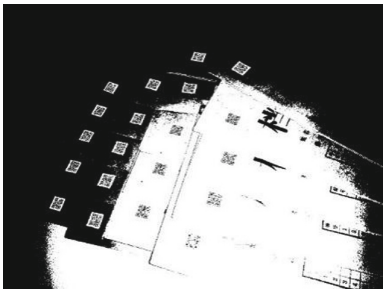

Fig. 3. OTSU method

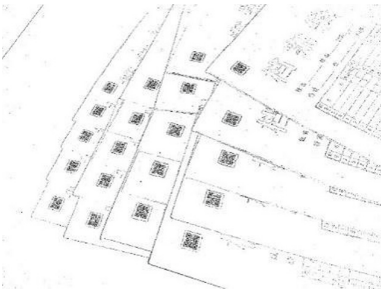

Fig. 4. Adaptive binarization method (gauss)

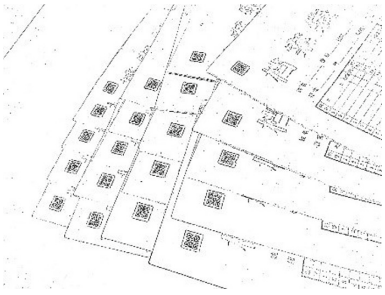

Fig. 5. Adaptive binarization method (mean)

It can be seen that the binarized images obtained in the above three cases do not meet our requirements, which is reflected in two aspects:

First, there are too many point-like interference regions in the image, and some of the interference regions are too close to the QR code, which is not conducive to the extraction of the QR code.

Second, the color of the QR code area and the background area in the image can't be divided clearly.

Our Method. In order to solve the above two problems, and considering the time complexity, we consider the global thresholding method, that is, to expand the difference between the foreground and the background in the grayscale image before using the OTSU algorithm.

Since the color of the portfolio is very close to the skin color in this scene, we use the method in skin color detection. First, the original RGB image color space is converted into YCrCb space, and then the image is separated into three channels of Y, Cr and Cb. Since the channel Y represents brightness, we abandon it to avoid the effects of uneven illumination. The QR code regions and the interference regions in the channel Cr and the channel Cb are clearly colored, so the two channels are used to replace the original grayscale image, and the OTSU algorithm is used for processing. The results show that both channels can separate the QR code area from the background area. As shown in Fig. 6, Fig. 7:

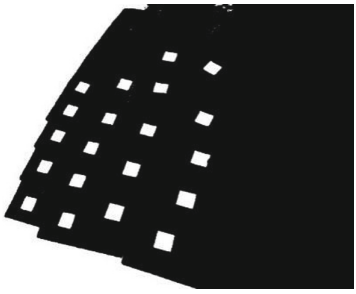

Fig. 6. Channel Cr

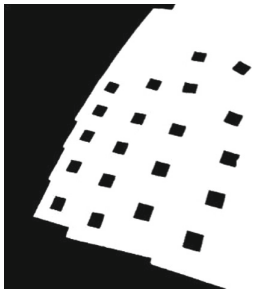

Fig. 7. Channel Cb

In the binary image obtained by the channel Cr, the boundary of the corresponding region of each QR code is smooth, but there are slightly more interference regions in the background.

In the binary image obtained by the channel Cb, there are fewer interference regions in the background, but there are certain burrs in the corresponding boundary of each QR code region.

In order to obtain a binary image with less interference area and smooth QR code region, we need to combine the advantages of the two channels, so we reverse the binary image obtained by the channel Cr, and then XOR the processed image with channel Cb, which combines the advantages of both images.

Since the obtained QR code image is black in the binary image and the background is white, and we hope that the background area can be connected as much as possible to eliminate the black holes in the background, the morphological closed operation is adopted.

Through the above operations, we can visually see the target area and the background area. Because there are obvious color mutations at the boundary, we can perform contour

detection based on this feature, and obtain a binary image containing only the outline of each area. Here we test the commonly used edge detection operator Sobel operator, Laplace operator and Canny operator respectively. The results show that the Canny operator detects the edge of the contour and the pseudo edge.

3.3 Positioning

In the part of QR code positioning, because some complicated algorithms are involved, and these algorithms have been implemented in OpenCV, in the following part we only introduce the call of the corresponding function of each algorithm, if you want to understand the specific implementation of the relevant algorithm, please see the official OpenCV source code.

Contour Positioning. Since the image used may not be incomplete during the shooting process, the result of the contour detection may be discontinuous, so we need to do further processing. Our goal here is to close the contours so that each contour becomes a closed curve. The common method is a polygonal curve approximation, and the *approxPolyDP* function in OpenCV can help us. This has the advantage of reducing the number of vertices per contour, simplifies the image outline to a certain extent, and also makes the contour continuous.

First, we use the *findContours* function in OpenCV for contour detection. Since the outline of the hierarchy will include many layers, there is a nesting relationship between the layers. In this task, we only need to locate each target area, so we only need to detect an outline for each target area. In order to avoid missing defects in the corresponding position of the original image area, here we choose the outer contour. After using the *findContours* function, the coordinates of the points contained in each outline will be stored separately.

Since the image used may not be incomplete during the shooting process, the result of the contour detection may be discontinuous, so we need to do further processing. Our goal here is to close the contours so that each contour becomes a closed curve. The common method is a polygonal curve approximation, and the *approxPolyDP* function in OpenCV can help us. This has the advantage of reducing the number of vertices per contour, simplifies the image outline to a certain extent, and also makes the contour continuous.

Area Filtering. Next we filter the resulting contour and remove the contours that are not part of the QR code area.

Since our target area has an approximately quadrilateral shape, and the shape of other interference areas is mostly irregular, we can filter them according to this feature.

First, we obtain the minimum enclosing rectangle according to the vertices of each contour. Then we calculate the perimeter of each contour, the enclosed area and the perimeter and area of its enclosing rectangle. Table 1 and Table 2 show some measured data.

Table 1. The perimeters of the contours and the enclosing rectangles

Contour/pix			Rectangular/pix		
130.2	132.3	128.0	178.4	181.0	177.5
126.4	103.5	120.0	170.6	139.3	165.9
122.6	102.7	97.2	171.5	139.6	132.3
111.8	101.9	96.0	152.7	136.5	129.9
111.3	0.0	0.0	160.2	0.0	0.0
98.8	106.2	7.2	136.2	149.1	14.4
44.8	68.0	88.9	89.6	116.8	124.8
18.1	0.0	19.8	36.2	0.0	39.7
283.2	100.6	102.1	286.9	140.0	146.2
91.9	92.4	2057.0	125.1	128.7	2323.3

Table 2. The areas of the contours and the enclosing rectangles

Contour/pixˆ2			Rectangular/pixˆ2		
1893.0	1981.0	1904.5	1987.9	2043.5	1962.0
1754.0	1181.0	1604.0	1816.0	1211.8	1720.0
1734.0	1169.5	1019.5	1823.7	1217.5	1089.0
1402.0	1116.0	992.5	1453.1	1164.0	1050.5
1474.5	0.0	0.0	1594.8	0.0	0.0
1088.0	1257.5	0.0	1158.8	1386.7	0.0
0.0	149.5	903.0	0.0	492.0	973.0
0.0	0.0	0.0	0.0	0.0	0.0
31.0	1144.5	1190.0	4756.2	1225.0	1318.2
922.0	950.5	103.5	975.9	1034.6	289068

By Analyzing the Data in Tables 1 and 2, We Find that:

For a normal target area, the perimeter of the contour is not much different from the perimeter of the corresponding enclosing rectangle. However, for the interference area, a part of the contour perimeter is not much different from the perimeter of the enclosing rectangle, and the other part is half of the perimeter of the enclosing rectangle.

For a normal target area, the contour area is approximately equal to the area of the corresponding enclosing rectangle, but for the interference area, the areas of the two are different or both are 0.

Therefore, based on these results, we first filter out the area of the enclosing rectangle with a small length and width. Then, calculate the ratio of the area of the enclosing rectangle to the area of the contour. If the ratio is approximately 1, it is the target area. If the ratio is too large, then it is the interference area.

After the above operation, the interference area in the contour is basically filtered out, and if other interference areas still exist, it can be excluded in the QR code recognition process.

3.4 Recognition

QR Code Correction. Since the QR code recognition library ZBar we use has poor support for the deformation of the QR code, we initially get that the target area is not a standard square, so we need to get the contour area before the QR code recognition. The perspective transformation is performed to restore the shape of the inclined area to further improve the success rate of the QR code recognition.

Perspective transformation is a kind of nonlinear transformation, which can project the original image into a new plane of view, and then correct the image. Equation (5) shows how the transmission transformation is calculated

$$\left[x^{'}, y^{'}, z^{'}\right] = [u, v, w] \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{5}$$

In the above formula, $[x = x'/z', y = y'/z']$ represents the transformed pixel coordinates, $[u, v, w]$ represents the pixel coordinates before the transformation, and a is the matrix of the transmission transformation. Since we deal with the image transformation of a QR plane, w and a33 are always equal to 1.

According to the transmission transformation, we can calculate the corrected QR code area. First, we calculate the coordinates of the corresponding points in the original image according to the coordinates of the vertices in the contour and sort the vertices (because we have reduced the image during preprocessing), then we use the perspective transformation to deform the corresponding regions in the original image. Here we call the *getPerspectiveTransform* function in OpenCV to get the perspective transformation matrix, and then call the *warpPerspective* function to perform affine transformation, and finally get the corrected QR code region.

QR Code Recognition. After that, we call the open source QR code recognition library ZBar to detect the corrected QR code region, and coordinate the region where the QR code exists.

4 Experiments and Analysis

Parameter setting: In the resolution adjustment section, the resolution of the input image is 1440 * 1080, so the resolution is adjusted to 800*600. In the binarization part, the default parameter setting is adopted in the OTSU method, and the filter used by the adaptive binarization (Gauss) is 3 * 3. In the edge detection part, the lower limit of Canny operator is 80 and the upper limit is 240. In the area filtering section, filter out areas where the length and width of the enclosing rectangle are less than 25 pixels or

the width and height ratio of the enclosing rectangle is less than 1.4, and areas where the ratio of the area of the enclosing circle to the area of the enclosing rectangle is less than 0.4.

Experimental setup: We selected 30 images taken from different angles, including different numbers of QR codes, to test the missing detection of the target area and the correct rate of decoding the QR code. Table 3 shows the test results.

Table 3. QR code recognition rate (recognized QR codes / total QR codes)

QR code recognition rate					
20/20	15/15	20/20	12/13	20/20	20/20
20/20	12/13	11/13	19/20	9/9	20/20
8/9	13/13	15/15	9/9	20/20	12/12
15/15	20/20	11/12	11/13	15/15	9/9
11/12	10/12	9/9	15/15	19/20	12/12

It can be seen from Table 3 that our algorithm has a good recognition effect on a variety of QR codes, but in some cases, the QR code cannot be recognized occasionally. After analysis, we found that when the portfolio is not empty and the shooting angle is too oblique, it will cause different degrees and types of distortion in the QR code in the captured image, which may result in the QR code not being decoded. Therefore, our algorithm applies to multiple QR code recognition that is less severely deformed.

Table 4. Regional omissions (missing areas/redundant areas)

Regional omissions					
0/0	0/0	0/0	0/1	0/0	0/0
0/0	0/1	2/0	1/1	0/0	0/2
0/0	0/0	0/1	1/2	0/1	0/0
0/0	0/0	2/0	0/1	0/0	0/0
0/0	0/0	0/1	0/0	0/0	0/1

As Table 4 shows, our algorithm can detect and separate all regions containing the QR code in most cases. For the extra interference area in the detection, we can eliminate it by detecting whether there is a QR code. For the missing QR code area in the detection, we suggest to put the QR code in the center of the image when the image is taken to help reduce the deformation of the QR code area.

5 Conclusion and Future Work

In this paper, we first propose the use of QR code to store key archive information, and design a method of simultaneously inputting multiple pieces of archive information, to solve the problem of low efficiency of document management information inputting in the Internet of Things era. The results show that, compared with the existing method of entering one file at a time, the method we designed can simultaneously enter the information of multiple files at the same time, and the number of simultaneous entries can also be adjusted, which has a good expansion. So our method not only improves the efficiency of file information entry, but also provides new solutions for data entry in the existing intelligent file management system.

Of course, our method still has some limitations. In our experiment, the QR codes of all file labels are of the same type and the same size, and when entering specific files, you may encounter situations where different types of files are simultaneously entered. In addition, when multiple files are overlapped, the types and degrees of deformation of different QR codes are not consistent, and how to correct the deformed QR codes becomes very challenging. Therefore, we will conduct further research on these issues in future work to build a more intelligent file management system.

Acknowledgment. This work is supported by the National Key R&D Program of China under Grant 2018YFC0831300, China Postdoctoral Science Foundation under grant number: 2018M641172 and the National Natural Science Foundation of China under Grant 61701019.

References

1. Limin, L.: IoT and a sustainable city. In: 5th International Conference on Energy and Environment Research (ICEER), pp. 23–27. Elsevier Ltd, Prague, Czech Republic (2018)
2. Rajithkumar, B.K.: IoT based integrated health monitoring system. Int. J. Eng. Sci. Res., 225–231 (2018)
3. Praveen, M., Harini, V.: NB-IOT based smart car parking system. In: 2019 International Conference on Smart Structures and Systems (ICSSS), pp. 1–5. Chennai, India (2019)
4. Zhi, L., Guo, L., Xinjun, L.: IoT-based tracking and tracing platform for prepackaged food supply chain. In: Industrial Management and Data Systems, pp. 1906–1916 (2017)
5. Tiwari, S.: An introduction to QR code technology. In: 2016 International Conference on Information Technology (ICIT), pp. 39–44. Bhubaneswar (2016)
6. Pandey, J., Kazmi, S.I.A., Hayat, M.S., Ahmed, I.: A study on implementation of smart library systems using IoT. In: 2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS), pp. 193–197. Dubai (2017)
7. Sa-ngiampak, J., Hirankanokkul, C., Sunthornyotin, Y.: LockerSwarm: an IoT-based smart locker system with access sharing. In: 2019 IEEE International Smart Cities Conference (ISC2), pp. 587–592. Casablanca, Morocco (2019)
8. Jie, S., Linqian, D., Jing, S.: Management of university endowment fund archives under the background of big data. In: 2016 8th International Conference on Information Technology in Medicine and Education (ITME), pp. 584–586. Fuzhou, China (2016)
9. Wang, Y.: Design and implementation of electronic archives information management under cloud computing platform. In: 2019 11th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), pp. 154–158. Qiqihar, China (2019)

10. Guigui, Y., Gengguo, C., Kaomin, B.: Design and realization of equipment's archives management system based on Struts2 and Hibernate. In: 2010 Second International Conference on Information Technology and Computer Science, pp. 466–469. Kiev (2010)
11. Xiang, W., Yuhang, W., Kaiwen, H.: The Design of file management system based on website and Qr. In: 2019 International Conference on Smart Grid and Electrical Automation (ICSGEA), pp. 157–160. Xiangtan, China (2019)
12. Cui, Z., Cui, B.: Research on low cost student electronic archives management system in high-concurrency environment. In: 2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC), pp. 248–252. Beijing, China (2020)
13. Zhi, L.: Design and implementation of the comprehensive archives information digital management system. In: 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), pp. 1764–1767. Yichang, China (2012)
14. Xu, J., et al.: Archiving system of rural land contractual management right data using multithreading and distributed storage technology. In: 2019 8th International Conference on Agro-Geoinformatics (Agro-Geoinformatics), pp. 1–5. Istanbul, Turkey (2019)