Digital Image Processing 2016

Automatic Segmentation and Alignment of QR Codes

Team 8

Abdullah Al Zoabi

David Atef

Farah Bakkari

Fatma Abd El Hamid Habib

Mohamed Samer Kador

Contents

Introduction

QR code (or **Quick Response Code**) is the trademark for a type of matrix barcode (or two-dimensional barcode). [1]

Barcode is a machine-readable optical label that contains information about the item to which it is attached. A QR code uses four standardized encoding modes (numeric, alphanumeric, byte/binary) to efficiently store data; extensions may also be used. [1]

A QR code consists of black squares arranged in a square grid on a white background, which can be read by an imaging device such as a camera, and processed using Reed–Solomon error correction until the image can be appropriately interpreted. The required datum is then extracted from patterns that are present in both horizontal and vertical components of the image. [1]

In addition to data pattern, there are some pattern that use as Meta data such positioning pattern and alignment pattern.

[1][Online] https://en.wikipedia.org/wiki/QR_code [Accessed DEC 2016]

Objective

- The objective of this project is to detect, segment and align of QR Codes.
- Many condition are considered:
  - ❖ Different distances between camera and QR.
  - ❖ Picture taken from different camera perspectives.
  - ❖ Images with different backgrounds.
  - ❖ QR with different in-plane rotation angles.
  - ❖ QR on different sources (i.e. different lighting conditions).
  - ❖ Picture of more than one QR.
- Video and stream camera video are supported.

Implementation

The implementation steps are described in the following sections by order, with an examples for each step input/output.

Thresholding

Assuming the input image is a colored image with non-uniform lighting and the area of interest is only the QR Cod(s).

The output of this stage is a binary image with clear QR(s) without care abou the reaming image content.

❖ Steps:

1. Convert the image to gray scale image.
2. Some QRs may effect by lighting, that means the black color have vale higher than zero, so we normalize the image by subtract the MIN color from all pixels.
3. Since the area of interest is White/Black area, we apply, median filter to avoid possible noise.
4. The global thresholding is impractical in non-uniform lighting so the variable thresholding is required.
5. Divide the image to 20x20 samples.
6. If the sample have the same color (max color ≃ min color) exclude it.
7. Select the local threshold as TH = min color + (max color – min color)/2.
8. Apply threshold to each sample.
9. Binaries the enter image so the excluded sample are turned to white or black.

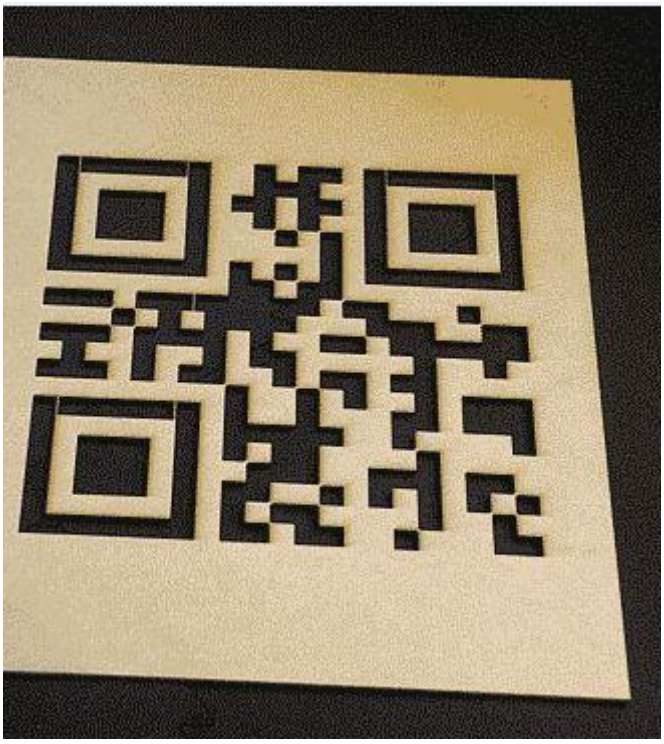


Figure 1 : original image

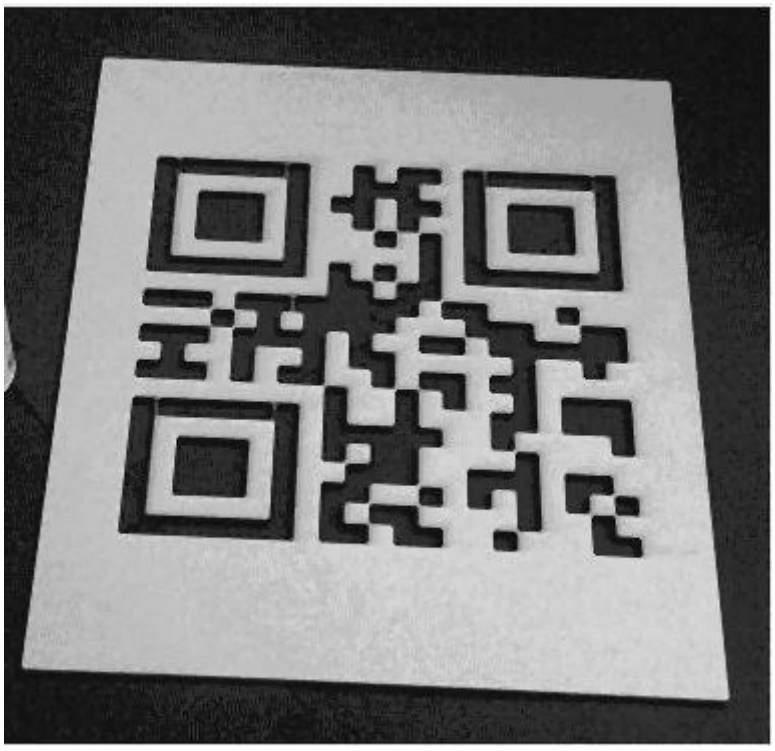


Figure 2 : after normalize and filter

Figure 3 : thresholded image

Figure 4 :non-uniform lighting image

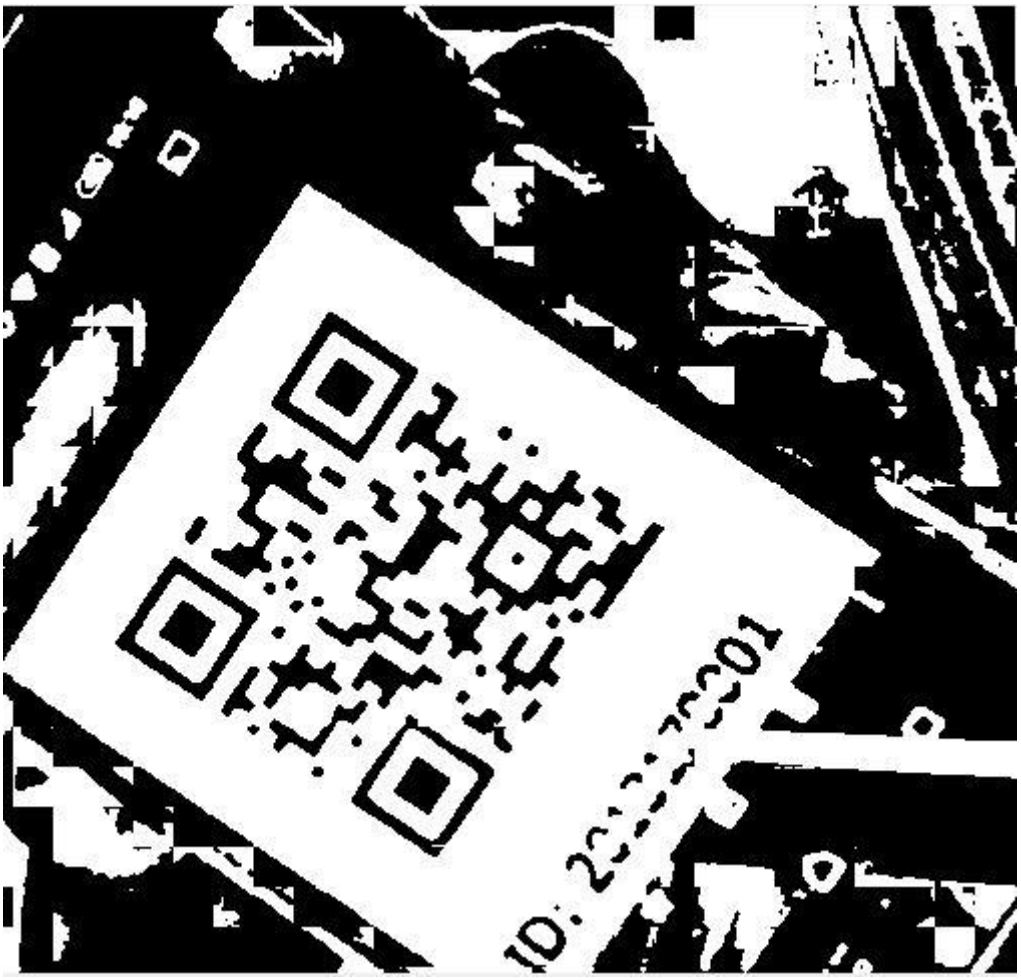


Figure 5 :non-uniform lighting image after threshold

Detect corners

Each QR is characterized by three positioning pattern distributed by three of its corners.

The positioning pattern can described by:

1. White square, contain
2. Black square with the same center,
3. Have a black border around it have the same center.
4. The black border dose not connected with any other black component (have white border).

Figure 6 : positioning pattern

Steps:

1. Get all connected component from the thresholded image.
2. Exclude very large/small component.
3. to Filter the component get only the square component (width ≃ High),
4. Then filter again to get component that have black square inside it with the same center and square done by :
    a. Crop the component,
    b. Check for black color in the center, if yes
    c. Invert the cropped image,
    d. Get the white pixels in the center by connected component,
    e. Check if (Width ≃ High), if yes,
    f. Check if the same center, if yes accept it.
5. Then Check the black square border around the white square don by :
    a. Convert area around the white square to white by hall filling,
    b. Subtract old image from the filed image, then we have single object image (the border), if the object square and have the same center with the white area if yes then accept.
6. All remaining component are the corners.

Figure 7 : samples after condition 3

Figure 8: samples after condition 4

Figure 8 : after condition 5, only corners

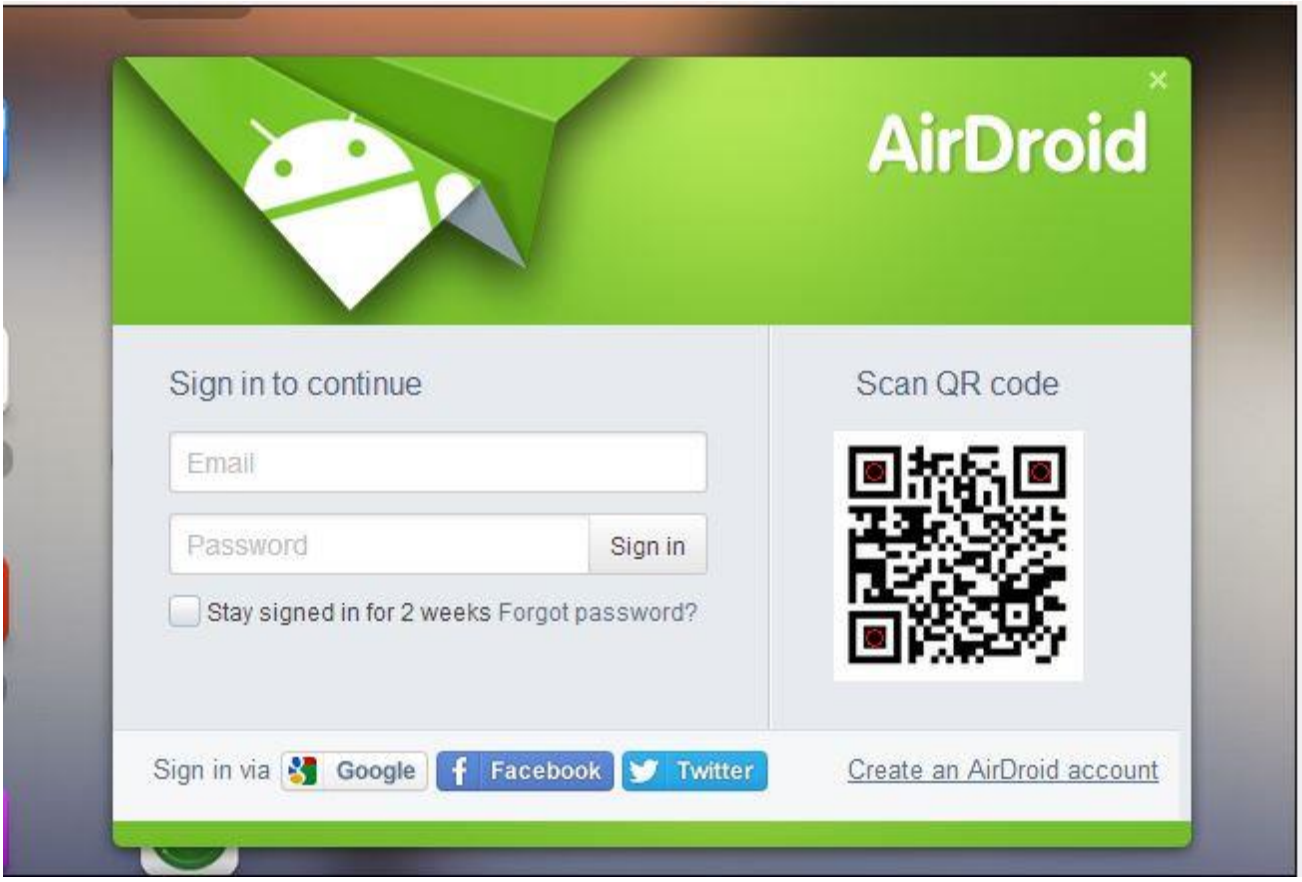


Figure 9 : detected corners

Corners grouping

The output of corner detection stage is a list of point each one present a corner, so in this stage we need to group the points, each 3 together which represent a single QR code.

The 3 corners for the same QR can considered as a Right & isosceles Tringle with the middle one as the right angle.

Also we consider the Image may hold more than one QR.

Steps:

1. Select the firs point a in corner list,
2. Search for any two points b and c which $a \neq b$, $a \neq c$ and $b \neq c$,
3. And $ab \simeq ac$ and $ab^2 + ac^2 = cb^2$
4. If exist chose the two points where ab and ac is the smallest and
5. Label a , b and c together and remove them from the original list,
6. If don't exist select the next point and go to 2.

Repeat the previous steps (number of corners/3) times.

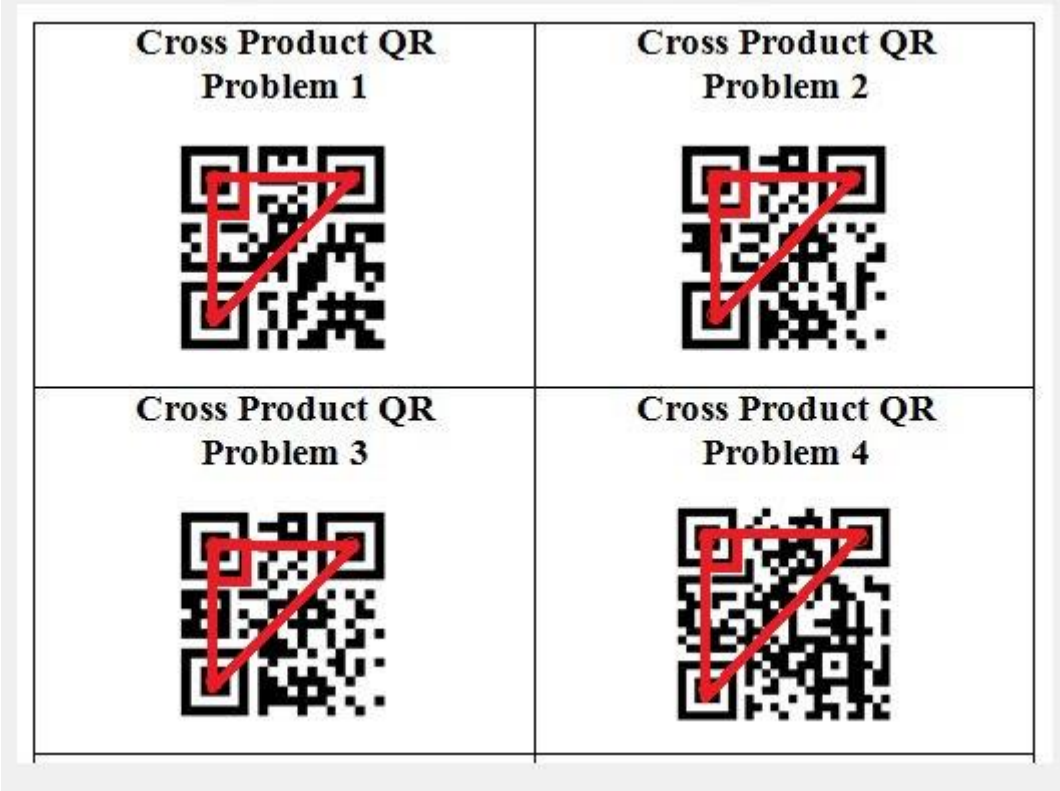


Figure 10 : corner grouping

Alignment and segmentation

The input to this stage is a list of grouped points-group of the three points -with labeled point -the Middle one-the list length is the number of QR cods in the image.

For each QR, may does not the correct alignment so we need to put it in the correct one.

Figure11 : wrong alignment

Figure12 : segmented QR with the correct alignment

As the middle corner is known and its new position is known we can estimate the new positions for the other two corner:

For 3 corners there are 3 value for X and 3 values for Y, and the middle point (x,y) must take one of the following value :

1. X_{max}
2. X_{min}
3. Y_{max}
4. Y_{min}

To explain this step is better to take an example:

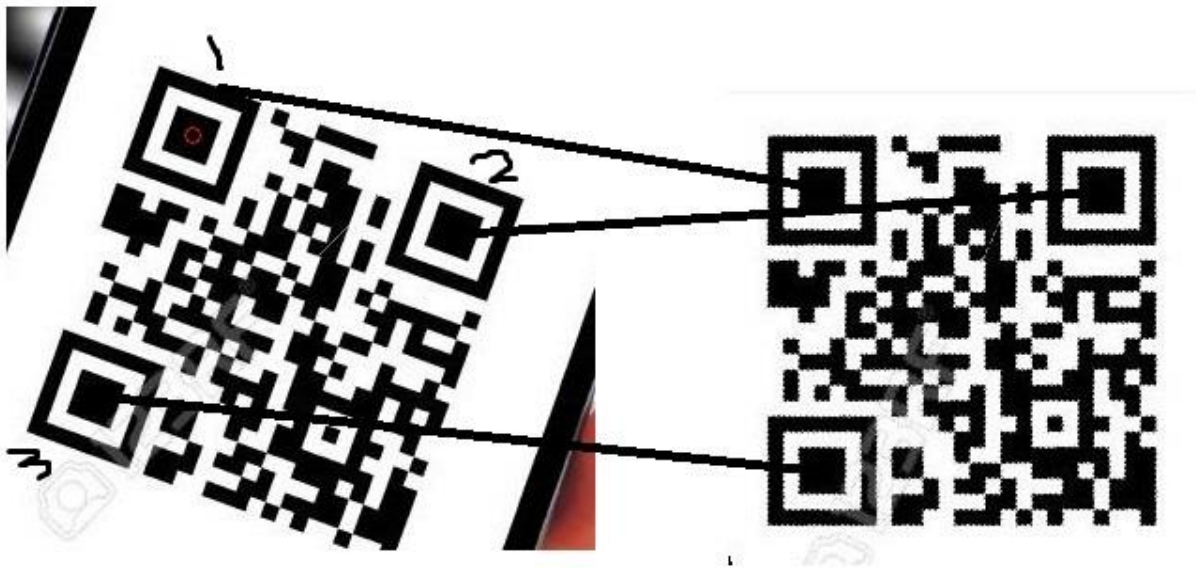


Figure 13 : alignment technique

The above QR have the middle corner (corner 1) in the Y_{min}, and X2>X1 and X1>X2 so we can say that 1 will mapped to the top right, 2 to the top left and 3 to the button right.

The 3 other cases will handled in the same method.

The distance between the new positions calculated from the distance between the old corners.

Now we use the wrap function to map the QR to the correct alignment.

Actually the segmentation done by the warping.

Video and live stream

As known the video is a collection of frames displayed after each other, actually the frames are images.

Thus the app read the video farm by frame and process only one frame of each 60 (frame rate = 60).

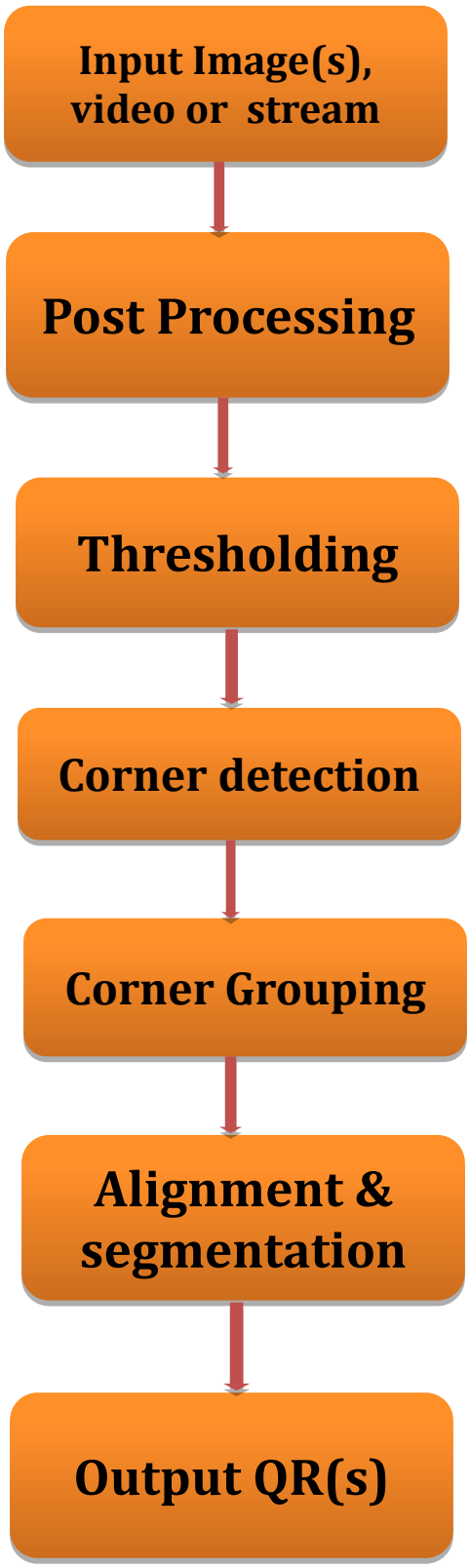
In most videos 60 frame are 2 seconds, and this time is reasonable for the appearance of the QR(s).

Reduce the frame rate mean reduce the performance, more processing time.

The same for live stream frame are taken with frame rate 60, from the PC camera and processed.

The video and live stream also support the multi-QR per image.

The big picture



Test and Performance

For image set with 19 image and 35 it take 13 second including the reading/writing time so it take 0.68 second/image and 0.37 second/QR.

All the result are perfect without any errors.

User guide

1. Run the application.
2. Press “save folder” and select the folder to save the result then,
3. Press “single image ” and select image to process it or,
4. Press “images folder ” and select folder to process entire folder of images or,
5. Press “video folder ” and select folder to process entire folder of video or,
6. Press “stream video “to open the PC camera.