Wounded QR codes

QR codes (short for *Quick Response Codes*) were invented in 1994, by the DENSO Corporation. I'm sure you've seen lots of them. They are the natural progression of bar codes.

Bar codes are one dimensional, encoding data in a series of alternating high contrast lines.

QR codes store data in two dimensions in the form of an array of contrasting regions. The information density of a QR code is much higher than a vanilla barcode; depending on the format used and the resolution of reader, over a thousand bytes can be encoded in a region the size of a postage stamp.

QR codes use a Reed–Solomon error correction based technology to help recover from errors in reading (for instance, caused by a smudge, badly printed code or other deformity).

There are four selectable levels of redundancy that can be used in generating QR codes (more on this later). I was curious as to what this damage tolerance translated to in real life.

That's right, I'm going to maim, damage and disfigure some QR codes (all in the interests of research) …

How do QR codes work?

Before we break things it's probably best to have a quick look at how QR codes work. It don't want to duplicate the excellent resources already out there, so here is just a brief explanation. (For full details, check out the specifications of the standard **ISO/IEC18004**).

On three corners of a QR code are square blocks that the reader uses to coarsely identify and then align the code. A correctly rendered code also has a *"Quiet Region"* (Ideally, at least four 'pixel' elements wide) all around the edge of the code to help the reader disambiguate between the code and any other background image.

A spacing gap appears around the edge of the orientation elements, and around this are Format and Versioning information.

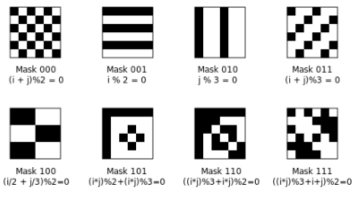
In addition to different storage formats and redundancy definitions, a variety of masks can be applied to the encoding, and the specification of the mask used is stored in the format information.

The masks alter the appearance of the code (toggling elements to make canonical versions that represent the same data, but are rendered in different ways).

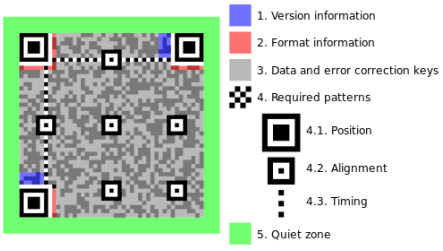
Ideally a mask is chosen that minimizes the appearance of large blocks of the same color that may cause issues in reproduction or reading in a 'noisy' environment).

The mask is applied via an XOR function and can be removed be applying the same Boolean operation a second time. There are multiple masks available in the standard that can be applied.

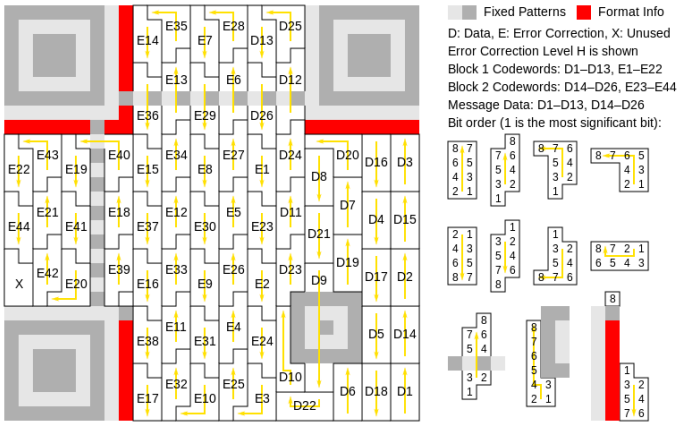
An additional alignment block appears in the fourth corner on small codes (in larger codes, multiple alignment blocks appear in the grid). These alignment blocks help deal with the skew. Finally, there are two timing strips in perpendicular directions composed of alternating elements.



A larger format QR code is shown below showing multiple alignment blocks. All other elements remain the same.



The payload for the code (data and checksum data) is then zig-zagged through the grid around the control elements.



Error Correction

Data is stored in a QR in words that are 8 bits long and use the Reed–Solomon error correction algorithm with four configurable error correction levels. The higher the error correction level, the less storage capacity. The following table lists the approximate error correction capability at each of the four levels:

L	*(Low)*	7% of codewords can be restored.
M	*(Medium)*	15% of codewords can be restored.
Q	*(Quartile)*	25% of codewords can be restored.
H	*(High)*	30% of codewords can be restored.

Here are renderings of a QR code for https://DataGenetics.com. Note, when we have low damage tolerance, the payload is short and so a smaller size QR code is generated. As more redundancy is encoded into the payload the length increases, and the resulting code is larger.

Low

Medium

Quartile

High

Let the destruction begin …

Using the **H** version of the above QR code, I'm going to attempt a few distortions and deformations to see how much damage it is possible to do to the code and still be able to read it.

I'm not being incredibly rigorous or controlled in these tests; all I'm doing is distorting the QR code in some way, then pointing my smart phone (which has a free QR reader app installed) at the screen to see if it could be read. (far away from a standard lab environment!).

There could be variability depending on the brightness of the screen, background lighting … etc. but at least it will give some idea of what is possible. If you try scanning these codes, you might get different results.

To make it a little more fun and interactive, I've turned it into a little game. Below, on the left, you'll see a series of QR codes displayed in some modified ways. In the middle will be a short description of what changed and a button marked Show Result

Your goal is to look at the source image and guess if you think my smartphone reader was able to scan the code. Click on the button to reveal the answer. It's just for fun, no points will be recorded!

Let the destruction begin …

Quiz

Control test. This is the basic QR code. Does it scan?

Show Result

Yes

Of course this scans!

1

How about if we shrink it down. Does it scan?

Show Result

Yes

Yes, as long as our scanner has the resolution to differentiate the dots we can scan it.

2

How about if we rotate and shrink it. Does it scan?

Show Result

3

Yes

Rotation of the image does not effect the ability to read. The alignment blocks are used to guide the process.

Reduced contrast. Does it scan?

Show Result

4

Yes

My smartphone app had no problem with reading this image.

Even more reduced contrast. Does it scan?

Show Result

5

No

The contrast of this image was too poor for the camera on my phone to differentiate.

What happens if we lose the corner and edge? Does it scan?

Show Result

6

Yes

As long we we don't lose more than approx 30% of the code we can recover.

What happens if you code catches *yellow spot fever?* Does it scan?

Show Result

7

Yes

As long we we don't lose more than approx 30% of the code we can recover.

How about *yellow and red spot fever?* Does it scan?

Show Result

8

What happens if we lose part of the positioning block? Does it scan?

Show Result

9

No

It appears that the presence of the corner position blocks is essential for the app (at least the version I was using) to determine there was a code there and to orientate it! Even though these corner blocks contain no data, they appear essential to readers to help determine where to read.

How about a blob of orange paint in the middle, rather than lots of little blobs? Does it scan?

Show Result

10

Yes

Again, we're less than 30%, so everything is cool.

Here the code has been ripped and not put together correctly, leaving a gap. Does it scan?

Show Result

11

No

The additional gap caused by pulling the code apart has messed up the timing. This code does not scan.

If, instead of pulling apart, the code is glued back together with an excess of (orange) glue will that make things better? Does it scan?

Show Result

12

Yes

Obscuring the code with excess glue is fine, and better than leaving a gap (see above). If you are repairing a damaged QR code to make it read, try and get it as square as possible.

Missing a chunk in the middle. Does it scan?

Show Result

13

Yes

No problem!

What if we fill the void with *"noise"*? Does it scan?

Show Result

14

Yes

The noise* is ignored by the decoding algorithm. If you search the web you will find examples of companies that have generated *"creative"* codes by inserting the logos into QR codes and still having them read OK.

*Hey, don't call me noise!

How about a skew? Does it scan?

Show Result

15

Yes

Providing we don't take this to ridiculous extremes, skews and distortions can be read. The corner alignment symbols are used by the reader to help interpolate and re-project back to a square grid. This helps when scanning a code in real life from a position not directly perpendicular

Skewing in two planes and some added noise. Does it scan?

Show Result

16

Yes

This scans without a problem.

More extreme skewing/tomb-stoning. Does it scan?

Show Result

17

No

This is just too severe for the timing to recover from.

Need eyeglasses? How about a Gaussian Blur? Does it scan?

Show Result

18

Yes

Same answer!

A little bit more blurring. Does it scan?

Show Result

19

Yes

Yes, I was able to get this code to scan!

Random colored noise. Does it scan?

Show Result

20

Yes

As long as there is sufficient contrast, the code can be read.

Clipping a few corners. Does it scan?

Show Result

21

Yes

Yes, this code (created using one of those artistic filters in a paint package) is still well formed..

Feeling a little dizzy? Does it scan?

Show Result

22

Yes

The slight swirl distortion in the middle is not severe enough to deviate the median points of the pixels enough to stop it being read.

Feeling more dizzy. Does it scan?

Show Result

23

No

These is too much distortion for the reader to deal with.

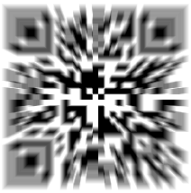
Star-Wars. Starting jump to hyperspace. Does it scan?

Show Result

24

Yes

The force is strong with this one.



Faster hyperspace. Does it scan?

Show Result

25

No

These are not the pixels you are looking for. Too much blurring and merging of pixels at the edge stops this being read.

Blurry edges. Does it scan?

Show Result

26

Yes

This surprised me too. There was still sufficient contrast once the image was quantised to a grid to allow it to be read correctly. Nice.

27

Ants are eating my code. Does it scan?

Show Result

Yes

No problem.

28

Bigger ants. Does it scan?

Show Result

Yes

No problem.

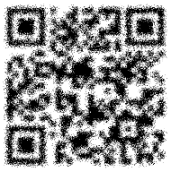
29

Someone spilled water on my code. Does it scan?

Show Result

No

I could not get this image to scan. It was just a little too out of focus.



30

Spray painted code. Does it scan?

Show Result

No

I could not get this image to scan.

31

Sent through a very poor quality fax machine. Does it scan?

Show Result

Yes

Too easy!

32

No *"Quiet Zone"* and lost in the middle of other QR code noise. Does it scan?

Show Result

Yes

No problem. The code was found and read even with all the additional location and alignment blocks.

Test

Does your smart phone have a QR code scanner? See if you can read the following code:

If you are interested in error correction codes, you might also like this article about how Credit Card Numbers Work.

Update

Since writing this article, various people have contacted me to say that they were able to successfully scan all the wounded codes I created. This is great! As I mentioned earlier in the article, there are many variables, and my testing was far removed from laboratory conditions. There are differences in monitor dot pitch, brightness and contrast between yours and mine. There are differences in lighting conditions, scan conditions and scan distance. There are differences in camera quality, resolution and tolerance. And, (probably most importantly), there are differences in the scanning app. Some software is more sophisticated/tolerant, and able to deal with a wider range of discrepancies than the app I used. This is good news for all the wounded QR codes out there!