Linear Barcodes Verification Project Report

Filippo Samorì Enrico Battistini Matteo Bonucci

Summary

INTRODUCTION **2**

BARCODE FINDER **3**

BARCODE ROTATION **6**

BARCODE CROPPING **7**

LOCATING THE HIGHEST AND LOWEST EXTREMES OF THE SMALLEST BAR 7
LOCATING THE COORDINATES OF THE INITIAL AND FINAL BAR 7
CALCULATING THE PARAMETER X (MINIMUM THICKNESS) 8

PARAMETERS EVALUATION **9**

SYMBOL CONTRAST 10
MIN REFLECTANCE 10
MIN EDGE CONTRAST 10
MODULATION 10
DEFECTS 10

Introduction

The aim of this project is to develop software that, given a set of images, each containing a barcode, is capable of:

- Localizing the ROI (Region Of Interest) containing the barcode and extracting features therein.
- Compute several parameters related to the print quality of the barcode, as defined by the ISO/IEC 15416 specifications. In particular, the following print quality parameters should be computed:
    1. Symbol Contrast
    2. Min reflectance
    3. Min edge contrast
    4. Modulation
    5. Defects
    6. Overall Symbol Grade

The calculation of the Decodability and Decode parameters is not necessary for the project. The calculation of these parameters is used to evaluate the quality of the barcode. Quality refers to how easily it can be read by specialized devices. The parameters are evaluated by considering the variation in pixel intensity along a horizontal line perpendicular to the barcode bars. This line is called the scan reflectance profile.

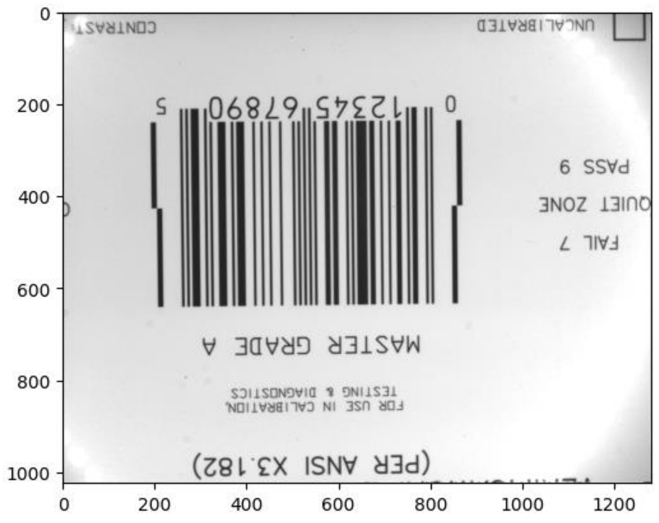


Figure 1 - UPC#01

Barcode Finder

The first step in processing the barcodes is to find the ROI (Region Of Interest) for the given image. It can be observed that the barcode consists of many thin vertical bars, which within an image correspond to areas with high intensity variation. Therefore, it has been thought to derive the image in a way that highlights these variations, at the expense of regions with uniform intensity. By doing so, areas of the image that definitely do not contain the barcode are excluded. First, the function

cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)

is used to calculate the smooth derivative along x and y, which preserves edges by filtering the image to reduce noise. In this case, the algorithm used is Sobel, which aims to preserve isotropy, meaning the independence of the operator with respect to the direction of application.

Subsequently, the gradient magnitude is calculated to obtain a matrix with the same dimensions as the image, containing the magnitude values of the derivative.

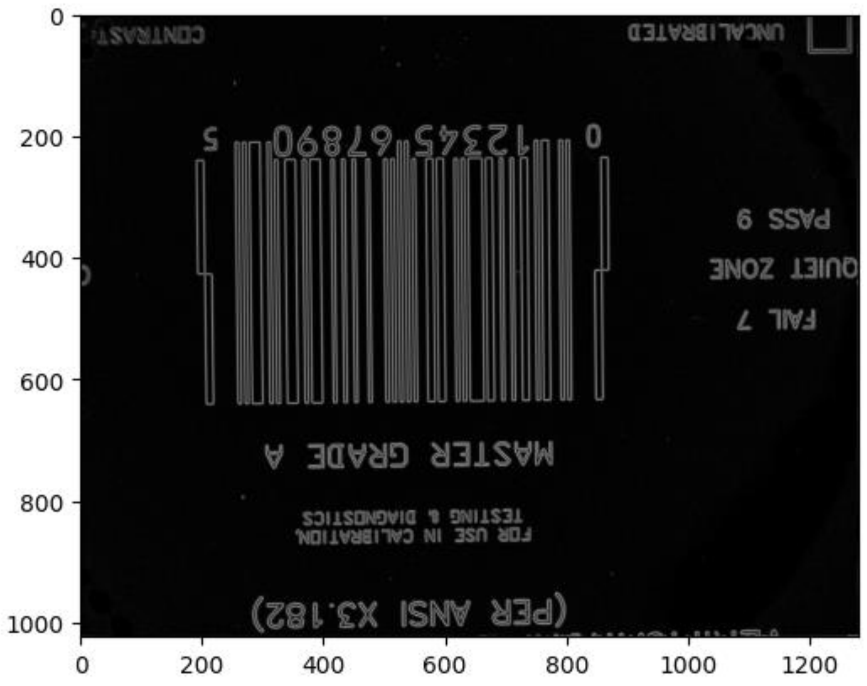


Figure 2 - Sobel Magnitude

The resulting image contains high numerical values in matrix positions corresponding to the most intense variations in the original image. To reduce noise and attempt to smooth out the area with a higher concentration of intense variation values, we use a Gaussian filter. This filter eliminates noise and tends to create a blur effect that better highlights the area where the barcode should appear. The underlying idea is similar to using the scale-space approach, the filter better highlights large features, such as the rectangular shape of the barcode (Sigma = 3, kernel = 19). Once the image is filtered, it is binarized using the Otsu algorithm to identify the most suitable threshold.

The calculation is handled by the function

th,img_bin = cv2.threshold(img_gaussian.astype("uint8"),0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

which calculates the threshold and returns it along with the already binarized image.

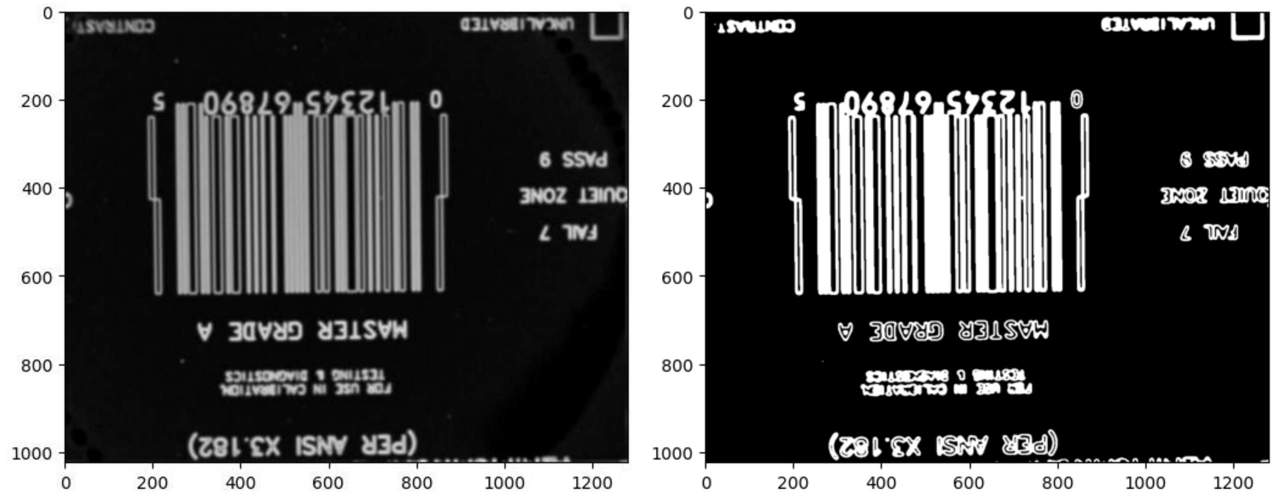


Figure 3 - Gaussian filtering and Binarization

Subsequently, we perform two openings, starting from the same binarized image. The first opening with a kernel composed of a vertical bar (2x60), the second opening with a kernel composed of a horizontal bar (60x2), obtaining two resulting images that are independent of each other.

This way, all patterns in the image that do not correspond to straight bars are eliminated. The opening is performed in both directions because barcodes can be oriented both vertically and horizontally.

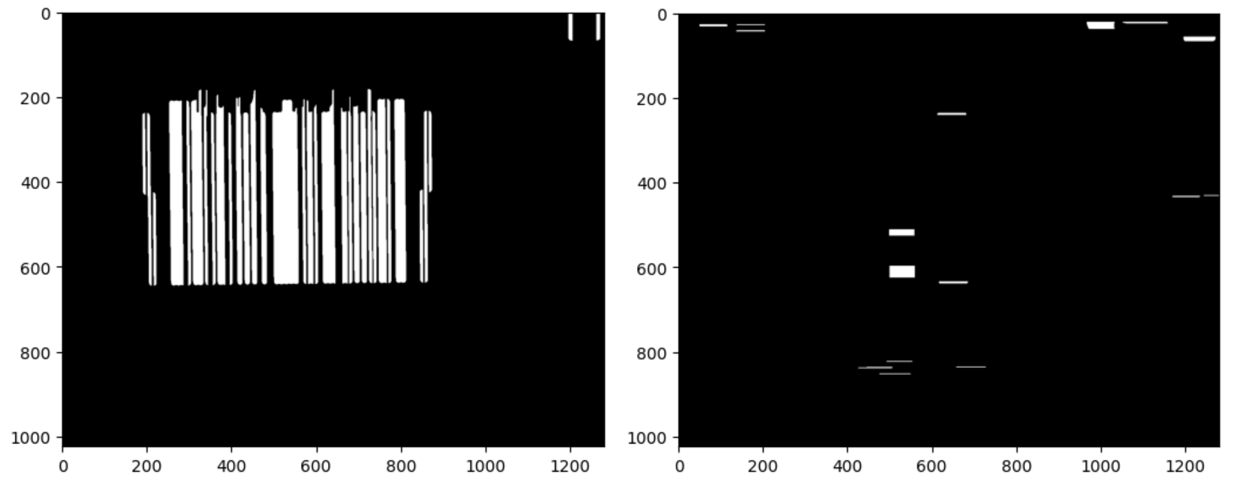


Figure 4 - Vertical and horizontal opening

After partially cleaning the image (referring to the image containing the gradient magnitude), we perform a closing operation with a square kernel to join the barcode bars. Any remaining spurious areas will also be highlighted, but only the connected component with the larger area between the two images is considered.

The areas of the connected components are obtained with the function

cv2.connectedComponentsWithStats(img, connectivity = 8)

which returns a list of connected components along with various parameters, such as area.

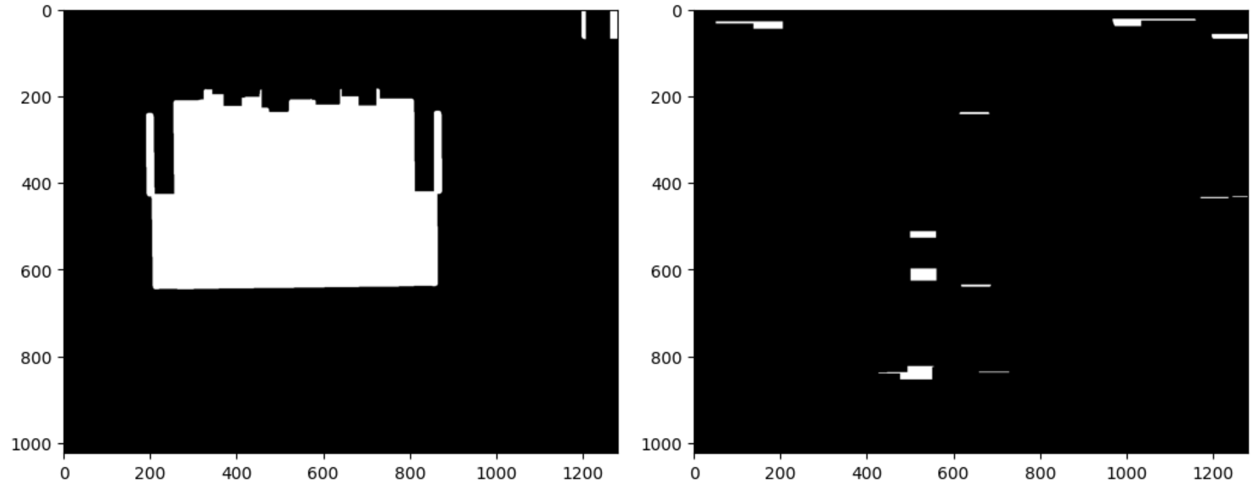


Figure 5 - Closing with square kernel

Once the largest area is found, the corresponding connected component is selected, and the coordinates of its position and its dimensions are extracted. With this information, the ROI can be calculated, and the original image can be cropped.

From the identified largest area, the orientation of the image can be deduced. If the largest area corresponds to the image on which an opening with vertical bars was performed, then we can deduce that the orientation of the barcode bars is vertical. Conversely, if the largest area corresponds to the image on which an opening with horizontal bars was performed, then the orientation of the barcode bars is horizontal.

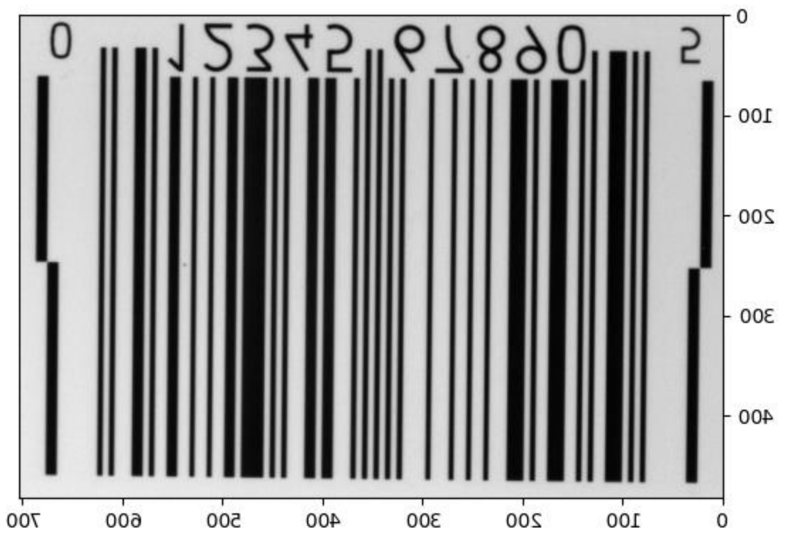


Figure 6 – ROI

Barcode rotation

To orient the barcode and align it perfectly with the horizon, we initially binarize the previously cropped image, still using the Otsu algorithm.

Then, we perform an opening (implemented as a closing since the background is white) as an intermediate operation to remove any numbers that might interfere with rotating the barcode correctly. The image is then treated as a point cloud with coordinates stored in a vector.

The vector of points is used as an argument for the function

_ , size , theta = cv2.minAreaRect(points)

Which returns the size and orientation of the rectangle with the smallest area that encloses the barcode (Minimum Enclosing Rectangle).

We format the angle to be considered relative to the horizontal axis.

The rotation matrix is calculated relative to the center of the image using the function

cv2.getRotationMatrix2D([height/2,width/2],theta, 1.0)

When the barcode is vertical, the shape of the rotated image is adjusted to avoid cutting off any part of it.

The image is rotated using the function

cv2.warpAffine(barcode_bin, rotation_matrix, (width,height), borderValue=(255,255,255))

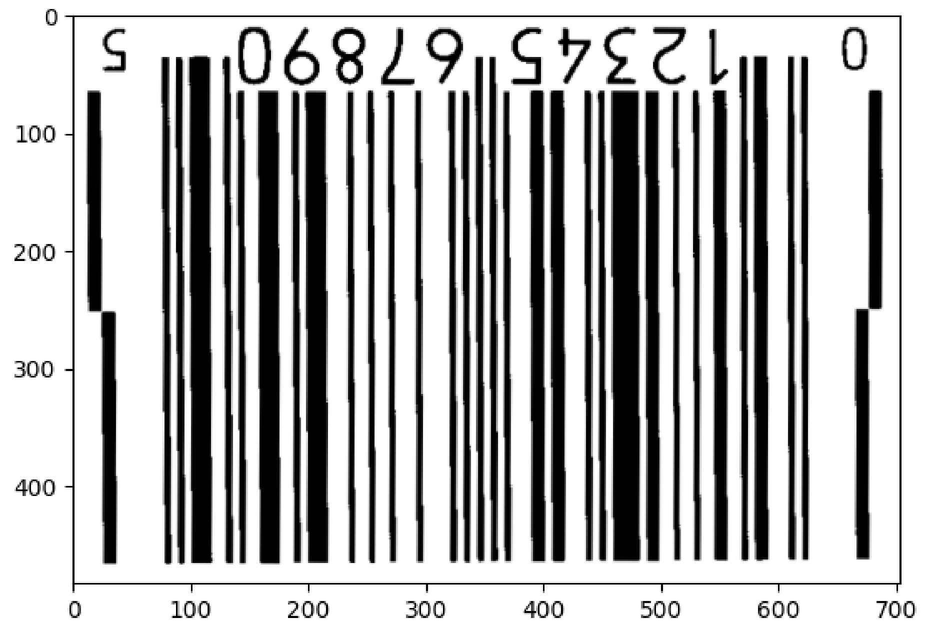


Figure 7 - Rotated binarized image

Barcode Cropping

First, an opening operation is performed (implemented as a closing since the background is white) with vertical bars to remove unwanted elements (such as numbers). It may seem like this operation is being done twice, but since the first opening is performed on an image that is not correctly oriented, it risks damaging the image. Therefore, it was preferred to repeat the operation on the correctly oriented image.

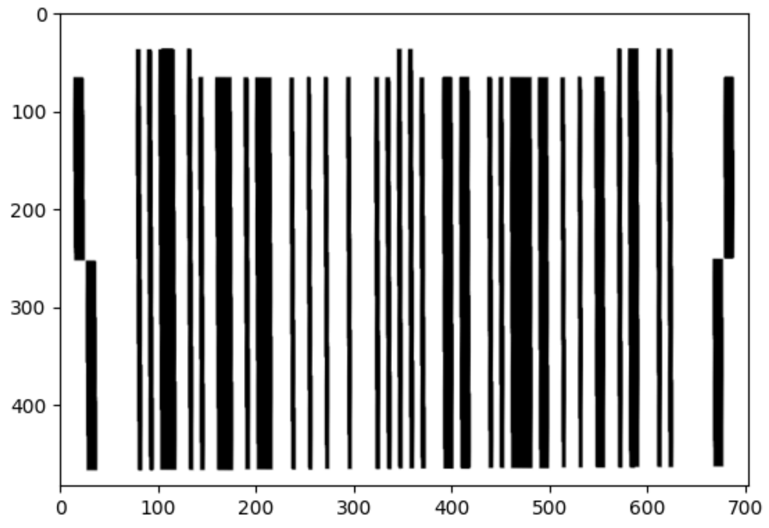


Figure 8 - Number elimination

Locating the highest and lowest extremes of the smallest bar

For identifying the top and bottom extremes of the smallest bar, loops are used to iterate through the image. The y-coordinates of the lower and upper pixels of the barcode are saved in two respective lists. As each y-coordinate is iterated, whenever a black pixel is encountered, it is saved in its respective list, and then the iteration moves to the next x-coordinate. Since the bars may not be perfectly straight, the lists may contain coordinates that do not correspond to the actual upper/lower edges of the bars, requiring the elimination of these incorrect values. Each saved coordinate is compared with its adjacent right and left coordinates. If the y-coordinate falls within a certain range relative to both its left and right neighbors, it is saved in a new list. This process ensures that incorrect values outside of this range are not considered. Finally, for the upper pixels, the coordinate with the maximum value is identified, while for the lower pixels, the coordinate with the minimum value is found. These values should correspond to the extremes of the smallest bar.

Portion of the list of upper y-coordinate with incorrect values (in red):

```
[67, 67, 67, 67, 67, 67, 67, 67, 67, 68, 105, 38, 38, 38, 38, 96, 40, 39, 39, 39, 39,
132, 39, 39, 39, 39, 38, 38, 38, 38, 38, 38, 38, 38, 38, ...]
```

For example, the value 105 is not saved in the filtered list because, having chosen the range equal to 2, the following inequalities are not true:

$$68 - 2 < 105 < 68 + 2 \quad \text{and} \quad 38 - 2 < 105 < 38 + 2$$

Portion of the filtered list of upper y-coordinate, without incorrect values:

```
[67, 67, 67, 67, 67, 67, 67, 67, 38, 38, 39, 39, 39, 39, 39, 39, 38, 38, 38, 38, 38, 38,
38, 38, 38, 38, 38, 38, ...]
```

Locating the coordinates of the initial and final bar

A similar procedure is applied to find the x-coordinate of the first and last bars of the barcode. The x-coordinates of the pixels corresponding to the leftmost and rightmost bars of the barcode are saved in two respective lists. As each x-coordinate is iterated, whenever a black pixel is encountered, it is saved in its respective list, and then the iteration moves to the next y-coordinate.

Since undesired elements might be present to the left of the first bar and to the right of the last bar, it was necessary to define the vertical edge of a bar as the set of pixels (belonging to an edge) whose x-coordinates have a variance less than or equal to 1. If the variance of the list containing the x-coordinates of the left or right edge exceeds 1, the maximum thickness of the identified element is calculated. Then, the search for the bar is restarted, skipping this element, i.e., the for loop starts again from the previously obtained x-coordinate, adding the found thickness. For example, if an undesired element is identified on the left, the next for loop begins by adding its maximum thickness to its leftmost x-coordinate. This search continues until a correct bar is found.

Finally, the maximum value of the list of x-coordinates of the left edge and the minimum value of the list of x-coordinates of the right edge are saved.

Calculating the parameter X (Minimum Thickness)

The method we used to calculate the minimum thickness (also known as X value) of the bars involved iterating through the x-coordinates within the range of values between the left and right edges of the barcode. When transitioning from a white pixel to a black pixel, we begin counting the sequence of pixels. Once a white pixel is encountered again, we save the thickness value if it is the minimum value for the current row being considered. This search is performed on all rows between the previously found upper and lower edges of the barcode.

We save in a list the minimun thickness of each pixel row of the barcode image. We compute the overall minimum thickness as the mean of the values in this list.

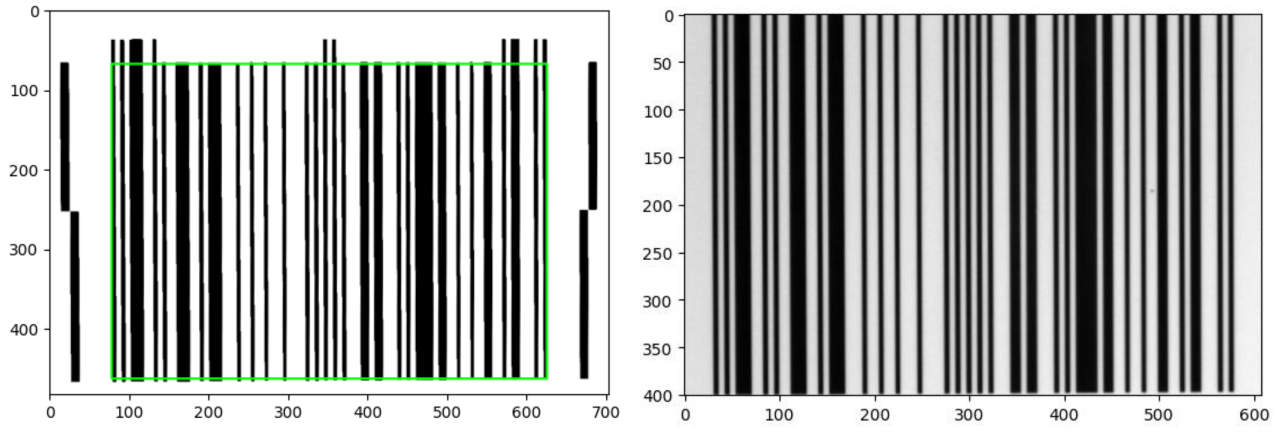


Figure 9 - Cropping of the image

Parameters evaluation

The image should be cropped to have a thickness of X above and below the barcode and a quiet zone of 10X on the left and right sides. Barcodes require a clear area to the left and right of the bar/space pattern, known as the quiet zone. This area must be free of any text or other graphics to ensure successful barcode reading by scanning equipment.

Once the image is cropped, ten scan reflectance profiles are drawn, all equally spaced apart.

The scan profiles used to calculate the parameters consist of ten for each image. The averaging of ten scans provides vertical redundancy, reducing the impact of a single scan passing through a part of the symbol that is unusually good or bad.

The scan lines are handled as vectors, with their elements representing pixel intensities relative to the x-coordinates of the image. These various vectors are then collected into another vector to form a matrix of size 10 x width, where width is the width of the image.

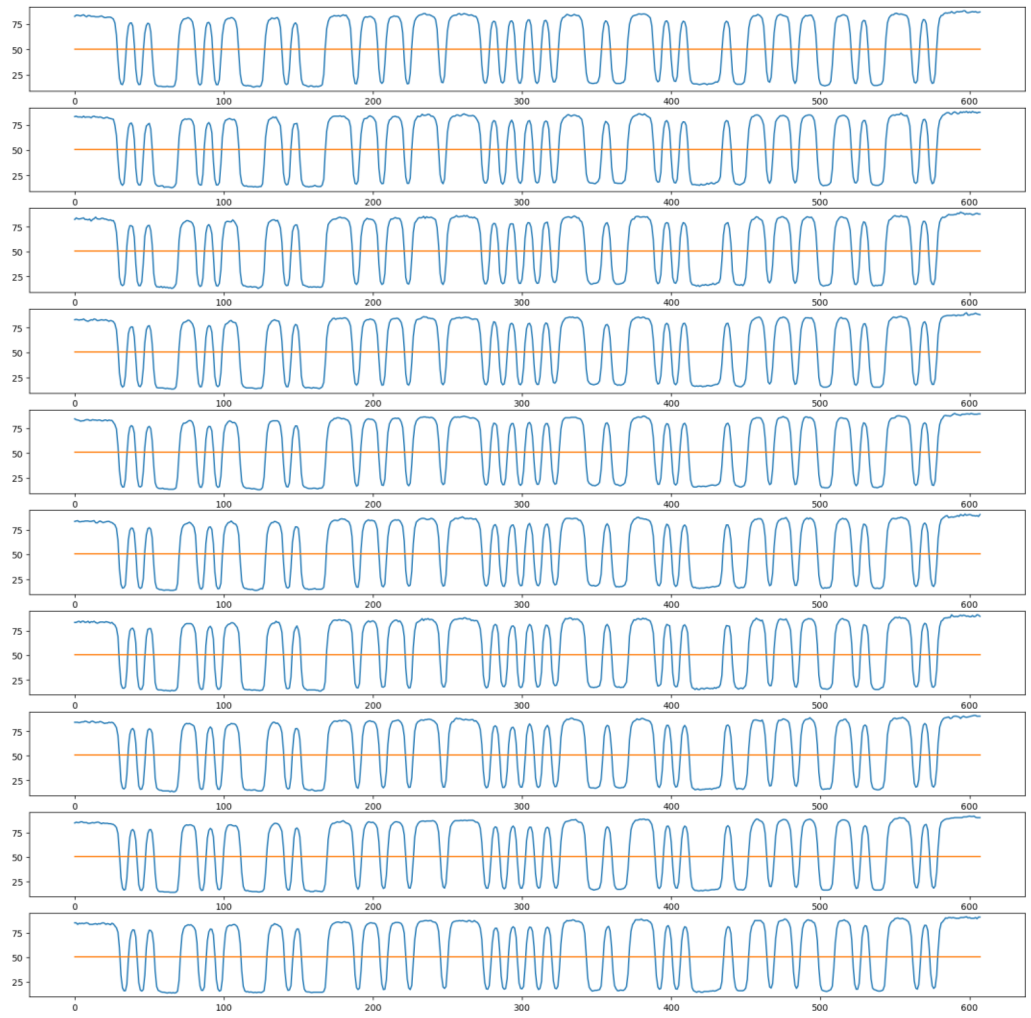


Figure 10 - Scan reflectance profiles

Next, the edges are searched for, which correspond to the x-coordinates that divide black bars and white spaces.

To discern bars and spaces, a Global Threshold is established on the scan reflectance profile by drawing a horizontal line halfway between the highest reflectance value and the lowest reflectance value observed in the profile. Edge determination can then be performed by counting the number of crossings at the Global Threshold.

The edges are stored in a list of elements for each scan line. Subsequently, these lists are stored in a single list of ten elements.

To store the numerical values, a dictionary is defined with entries corresponding to the parameter names. Each entry consists of a list of ten elements representing the parameter's value for each scan line.

The following algorithms are used for calculating the various parameters:

Symbol Contrast

Symbol Contrast is the difference (in percentage) between the maximum and minimum reflectance (i.e. Rmax – and Rmin, respectively). Rmax, Rmin are computed considering both the barcode as well as the Quiet Zone (bright area before and after the code).

Min Reflectance

Minimum Reflectance (i.e. intensity) value along the considered scan reflectance profile.

Min Edge Contrast

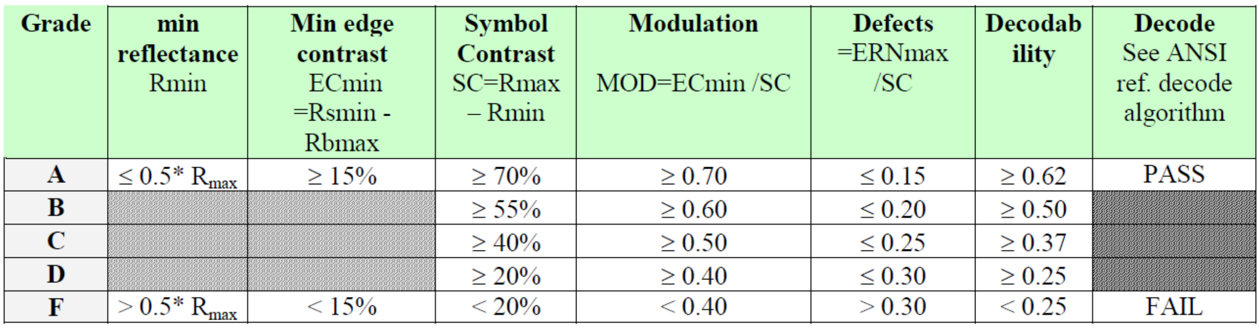
Min Edge Contrast (ECmin) is defined as the minimum difference (in percentage) between the minimum reflectance in a space (bright area) and the maximum reflectance in a bar (dark area) over pairs of adjacent spaces and bars. Each time we encounter an edge, we calculate the difference between the maximum peak and minimum peak between which the previous edge lies. Subsequently, we reset the minimum if we transition from a light area to a dark one, while we reset the maximum in the opposite case.

Modulation

Modulation is the ratio between Min Edge Contrast and Symbol Contrast.

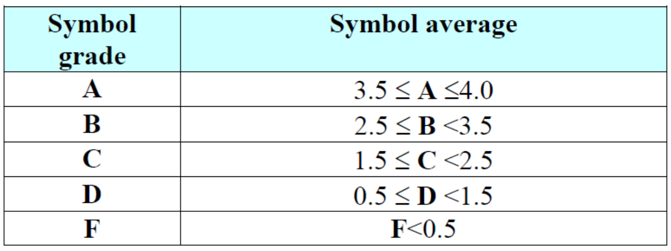
Defects

ERN (Edge Response Noise) is defined as the difference between the highest peak and the lowest valley within an element, which may be either a bar, a space, or a quiet zone. A valley is considered when a local minimum is found among three neighboring pixels (three levels neighborhood), while a peak is considered as a local maximum among three neighboring pixels. Each time an edge is encountered, ERN is calculated and saved if it is the maximum found in the scan line. Then, Defects is defined as the ratio between MAX ERN and Symbol Contrast.



Grade	**min reflectance** Rmin	**Min edge contrast** ECmin =Rsmin - Rbmax	**Symbol Contrast** SC=Rmax – Rmin	**Modulation** MOD=ECmin /SC	**Defects** =ERNmax /SC	**Decodability**	**Decode** See ANSI ref. decode algorithm
A	$\leq 0.5^* R_{max}$	$\geq 15\%$	$\geq 70\%$	≥ 0.70	≤ 0.15	≥ 0.62	PASS
B	▨	▨	$\geq 55\%$	≥ 0.60	≤ 0.20	≥ 0.50	▨
C	▨	▨	$\geq 40\%$	≥ 0.50	≤ 0.25	≥ 0.37	▨
D	▨	▨	$\geq 20\%$	≥ 0.40	≤ 0.30	≥ 0.25	▨
F	$> 0.5^* R_{max}$	$< 15\%$	$< 20\%$	< 0.40	> 0.30	< 0.25	FAIL

Once the parameters are calculated, the overall score for each parameter is taken as the worst among all scan lines (in according to the table above). A numerical value is assigned to it, and then the average of all scores found is calculated. Based on the numerical value of this average (in according to the table below), a global score is assigned to the image (Overall barcode grade).

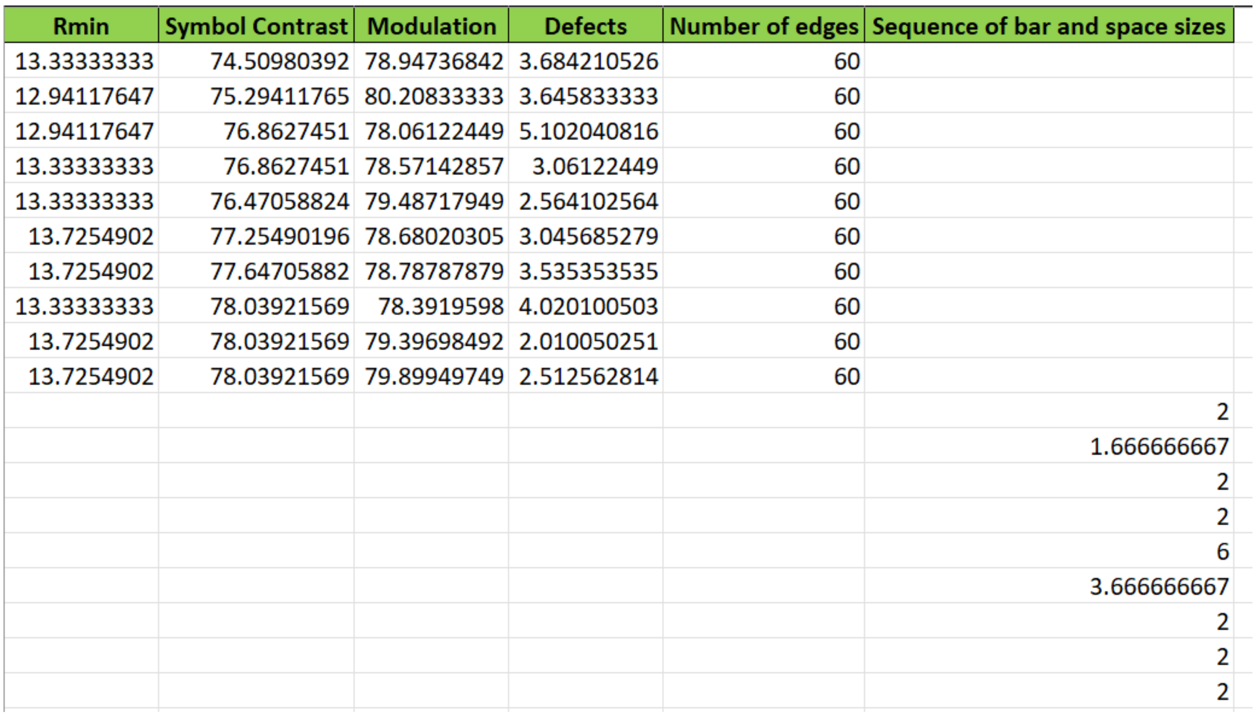


Symbol grade	Symbol average
A	3.5 ≤ **A** ≤4.0
B	2.5 ≤ **B** <3.5
C	1.5 ≤ **C** <2.5
D	0.5 ≤ **D** <1.5
F	**F**<0.5

After evaluating an image, the parameters to be included in the Excel file are saved in a dictionary. The dictionary entries are indicated by the name of the data that will be reported in the Excel table, while the data is stored in lists to which new values are added every time a new image is processed.

The principal_table is updated for each image, while the image_table is compiled for a single image and then overwritten the next time.

Using the Python Pandas package, dictionaries are converted into databases, which are then transformed into Excel sheets. For the image_table, a page is created for each image, and the data is exported once the parameter calculation is completed, as the database will be overwritten later. As for the principal_table, it is converted into a single Excel sheet only after processing all the images is completed.



Rmin	Symbol Contrast	Modulation	Defects	Number of edges	Sequence of bar and space sizes
13.33333333	74.50980392	78.94736842	3.684210526	60	
12.94117647	75.29411765	80.20833333	3.645833333	60	
12.94117647	76.8627451	78.06122449	5.102040816	60	
13.33333333	76.8627451	78.57142857	3.06122449	60	
13.33333333	76.47058824	79.48717949	2.564102564	60	
13.7254902	77.25490196	78.68020305	3.045685279	60	
13.7254902	77.64705882	78.78787879	3.535353535	60	
13.33333333	78.03921569	78.3919598	4.020100503	60	
13.7254902	78.03921569	79.39698492	2.010050251	60	
13.7254902	78.03921569	79.89949749	2.512562814	60	
					2
					1.666666667
					2
					2
					6
					3.666666667
					2
					2
					2

Figure 11- Image_table UPC#01 example

Images	Overall barcode grade	X-dimension	Height	Left-up	Left-down	Right-up	Right-down	Center	Orientation
C128_4.4LOW.BMP	F	5	787	[(261, 418)]	[(1048, 418)]	[(261, 1273)]	[(1048, 1273)]	[(638, 830)]	0.286476523
C128_4.4UP.BMP	B	6	792	[(291, 363)]	[(1083, 363)]	[(291, 1224)]	[(1083, 1224)]	[(670, 780)]	0
C128_7.5LOW.BMP	C	5	475	[(244, 218)]	[(719, 218)]	[(244, 1021)]	[(719, 1021)]	[(462, 602)]	0.430787206
C128_7.5UP.BMP	B	5	476	[(261, 145)]	[(737, 145)]	[(261, 949)]	[(737, 949)]	[(483, 533)]	0
C39_4.4LOW.BMP	C	2	510	[(173, 351)]	[(683, 351)]	[(173, 881)]	[(683, 881)]	[(413, 602)]	-0.156547546
C39_4.4UP.BMP	C	3	511	[(252, 356)]	[(763, 356)]	[(252, 891)]	[(763, 891)]	[(493, 610)]	0
C39_7.5LOW.BMP	D	3	510	[(289, 179)]	[(799, 179)]	[(289, 1081)]	[(799, 1081)]	[(528, 616)]	0.154435664
C39_7.5UP.BMP	A	5	510	[(277, 166)]	[(787, 166)]	[(277, 1071)]	[(787, 1071)]	[(517, 604)]	-0.150779724
EAN-UPC-CONTRAST IMGB.BMP	C	4	440	[(339, 344)]	[(779, 344)]	[(339, 951)]	[(779, 951)]	[(544, 633)]	-0.530502319
EAN-UPC-DECODABILITY IMGB.BMP	A	4	436	[(286, 354)]	[(722, 354)]	[(286, 955)]	[(722, 955)]	[(464, 640)]	-0.784820557
EAN-UPC-DEFECTS IMGB.BMP	C	5	436	[(470, 229)]	[(906, 229)]	[(470, 831)]	[(906, 831)]	[(649, 512)]	90
EAN-UPC-EAN-13 MASTER GRADE IMGB.BMP	A	5	436	[(318, 333)]	[(754, 333)]	[(318, 935)]	[(754, 935)]	[(488, 564)]	0.259255141
EAN-UPC-UPC-A MASTER GRADE IMGB.BMP	A	5	437	[(433, 211)]	[(870, 211)]	[(433, 813)]	[(870, 813)]	[(605, 424)]	90
EAN128-CONTRAST IMGB.BMP	C	5	181	[(128, 111)]	[(309, 111)]	[(128, 1274)]	[(309, 1274)]	[(197, 681)]	-0.12374115
EAN128-DEFECTS IMGB.BMP	F	7	183	[(150, 176)]	[(333, 176)]	[(150, 1332)]	[(333, 1332)]	[(135, 741)]	0
EAN128-LOW DECODABILITY IMGB.BMP	A	5	183	[(172, 196)]	[(355, 196)]	[(172, 1369)]	[(355, 1369)]	[(227, 765)]	0.498211592
EAN128-MASTER IMGB.BMP	A	6	183	[(146, 213)]	[(329, 213)]	[(146, 1372)]	[(329, 1372)]	[(219, 690)]	0
I25-CONTRAST IMGB.BMP	C	7	228	[(150, 221)]	[(378, 221)]	[(150, 1340)]	[(378, 1340)]	[(230, 758)]	0.97102195
I25-DEFECTS IMGB.BMP	F	7	232	[(133, 217)]	[(365, 217)]	[(133, 1342)]	[(365, 1342)]	[(150, 762)]	0.502581656
I25-LOW DECODABILITY IMGB.BMP	A	7	229	[(66, 245)]	[(295, 245)]	[(66, 1366)]	[(295, 1366)]	[(149, 791)]	-0.091529846
I25-MASTER GRADE IMGB.BMP	A	7	232	[(145, 269)]	[(377, 269)]	[(145, 1393)]	[(377, 1393)]	[(238, 812)]	0.550904036
UPC#01.BMP	A	3	395	[(242, 259)]	[(637, 259)]	[(242, 807)]	[(637, 807)]	[(371, 455)]	0.36885187
UPC#03.BMP	A	3	395	[(340, 327)]	[(735, 327)]	[(340, 873)]	[(735, 873)]	[(468, 586)]	-0.572944641
UPC#04.BMP	B	3	395	[(338, 359)]	[(733, 359)]	[(338, 903)]	[(733, 903)]	[(465, 615)]	-0.962860107
UPC#05.BMP	C	4	397	[(337, 368)]	[(734, 368)]	[(337, 916)]	[(734, 916)]	[(467, 630)]	0
UPC#06.BMP	D	3	396	[(321, 301)]	[(717, 301)]	[(321, 845)]	[(717, 845)]	[(446, 557)]	-2.121101379
UPC#07.BMP	F	3	395	[(328, 342)]	[(723, 342)]	[(328, 887)]	[(723, 887)]	[(455, 600)]	-0.987762451
UPC#08.BMP	A	3	397	[(331, 338)]	[(728, 338)]	[(331, 884)]	[(728, 884)]	[(459, 596)]	-0.707313538
UPC#09.BMP	B	3	396	[(337, 336)]	[(733, 336)]	[(337, 882)]	[(733, 882)]	[(465, 594)]	-1.101707458
UPC#10.BMP	C	3	397	[(348, 333)]	[(745, 333)]	[(348, 879)]	[(745, 879)]	[(475, 591)]	-1.080917358
UPC#11.BMP	D	3	395	[(340, 339)]	[(735, 339)]	[(340, 883)]	[(735, 883)]	[(464, 595)]	-1.672393799
UPC#12.BMP	F	5	396	[(323, 346)]	[(719, 346)]	[(323, 894)]	[(719, 894)]	[(454, 607)]	0
UPC#13.BMP	A	4	397	[(321, 329)]	[(718, 329)]	[(321, 876)]	[(718, 876)]	[(450, 588)]	-0.561698914
UPC#14.BMP	A	3	395	[(333, 356)]	[(728, 356)]	[(333, 902)]	[(728, 902)]	[(461, 616)]	-0.622756958
UPC#15.BMP	B	4	395	[(324, 355)]	[(719, 355)]	[(324, 900)]	[(719, 900)]	[(452, 613)]	-0.858299255
UPC#16.BMP	D	5	396	[(324, 345)]	[(720, 345)]	[(324, 892)]	[(720, 892)]	[(455, 605)]	0
UPC#17.BMP	D	4	396	[(332, 332)]	[(728, 332)]	[(332, 878)]	[(728, 878)]	[(462, 593)]	-0.339027405
UPC#18.BMP	A	4	396	[(332, 343)]	[(728, 343)]	[(332, 891)]	[(728, 891)]	[(463, 605)]	-0.12537384
UPC#19.BMP	B	3	396	[(327, 346)]	[(723, 346)]	[(327, 892)]	[(723, 892)]	[(456, 605)]	-0.713180542
UPC#20.BMP	B	4	396	[(335, 311)]	[(731, 311)]	[(335, 858)]	[(731, 858)]	[(465, 571)]	-0.392433167
UPC#21.BMP	C	4	395	[(326, 303)]	[(721, 303)]	[(326, 851)]	[(721, 851)]	[(455, 563)]	0.232908383
UPC#22.BMP	F	5	397	[(323, 299)]	[(720, 299)]	[(323, 847)]	[(720, 847)]	[(454, 560)]	0
UPC#23.BMP	A	5	397	[(319, 293)]	[(716, 293)]	[(319, 842)]	[(716, 842)]	[(450, 555)]	0
UPC#24.BMP	A	4	397	[(327, 303)]	[(724, 303)]	[(327, 850)]	[(724, 850)]	[(456, 562)]	-0.341041565
UPC#25.BMP	A	4	395	[(340, 292)]	[(735, 292)]	[(340, 840)]	[(735, 840)]	[(467, 553)]	-0.241752625
UPC#26.BMP	A	4	396	[(331, 275)]	[(727, 275)]	[(331, 822)]	[(727, 822)]	[(458, 533)]	-0.753845215
UPC#27.BMP	A	4	397	[(337, 270)]	[(734, 270)]	[(337, 818)]	[(734, 818)]	[(466, 530)]	-0.502586365
UPC#28.BMP	A	4	396	[(337, 263)]	[(733, 263)]	[(337, 811)]	[(733, 811)]	[(465, 523)]	-0.6902771
UPC#29.BMP	A	3	397	[(337, 267)]	[(734, 267)]	[(337, 815)]	[(734, 815)]	[(465, 527)]	-0.6902771
UPC#30.BMP	A	4	397	[(315, 292)]	[(712, 292)]	[(315, 838)]	[(712, 838)]	[(443, 550)]	-0.753845215
UPC#31.BMP	A	4	397	[(325, 294)]	[(722, 294)]	[(325, 841)]	[(722, 841)]	[(453, 552)]	-0.812652588
UPC#32.BMP	A	5	397	[(325, 309)]	[(722, 309)]	[(325, 858)]	[(722, 858)]	[(454, 570)]	0

C128_4.4LOW.BMP
C39_7.5UP.BMP
EAN128-DEFECTS IMGB.BMP
UPC#01.BMP
UPC#09.BMP
UPC#16.BMP
UPC#23.BMP
UPC#30.BMP
C128_4.4UP.BMP
EAN-UPC-CONTRAST IMGB.BMP
EAN128-LOW DECODABILITY IMG
UPC#03.BMP
UPC#10.BMP
UPC#17.BMP
UPC#24.BMP
UPC#31.BMP
C128_7.5LOW.BMP
EAN-UPC-DECODABILITY IMGB.BMP
EAN128-MASTER IMGB.BMP
UPC#04.BMP
UPC#11.BMP
UPC#18.BMP
UPC#25.BMP
UPC#32.BMP
C128_7.5UP.BMP
EAN-UPC-DEFECTS IMGB.BMP
I25-CONTRAST IMGB.BMP
UPC#05.BMP
UPC#12.BMP
UPC#19.BMP
UPC#26.BMP
C39_4.4LOW.BMP
EAN-UPC-EAN-13 MASTER GRADE IMGB.BMP
I25-DEFECTS IMGB.BMP
UPC#06.BMP
UPC#13.BMP
UPC#20.BMP
UPC#27.BMP
C39_4.4UP.BMP
EAN-UPC-UPC-A MASTER GRADE IMGB.BMP
I25-LOW DECODABILITY IMGB.B
UPC#07.BMP
UPC#14.BMP
UPC#21.BMP
UPC#28.BMP
C39_7.5LOW.BMP
EAN128-CONTRAST IMGB.BMP
I25-MASTER GRADE IMGB.BM
UPC#08.BMP
UPC#15.BMP
UPC#22.BMP
UPC#29.BMP
0.0
0.5
1.0