Midterm Project of Machine Vision

EAN-13 Barcode Reader

Adviser: Prof. Chi Cheng Cheng

Student: Chia Sheng Kuo

Student ID: M063020037

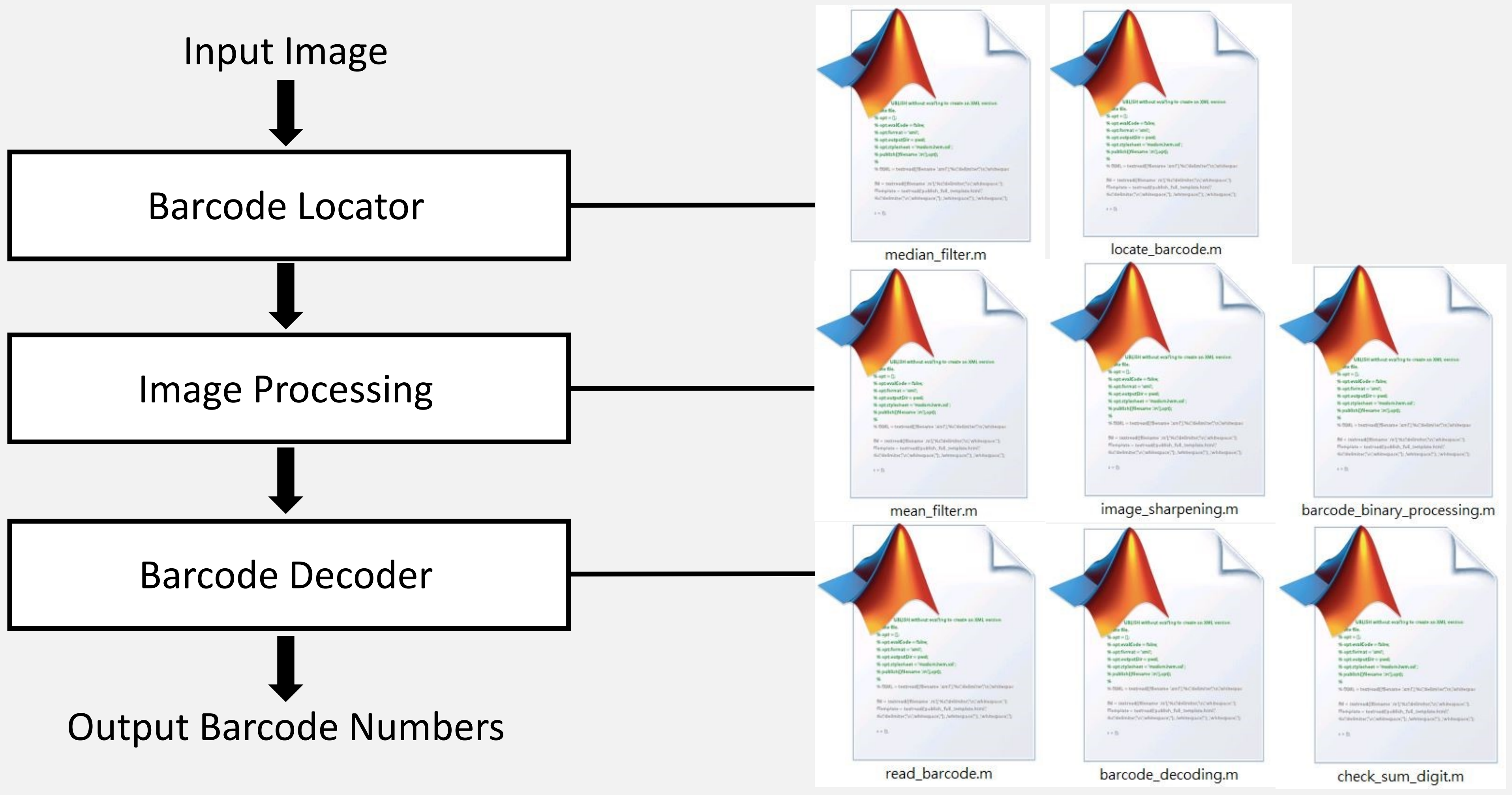
Dept. of Mechanical and Electro-Mechanical Engineering National Sun Yat-Sen University

Outline

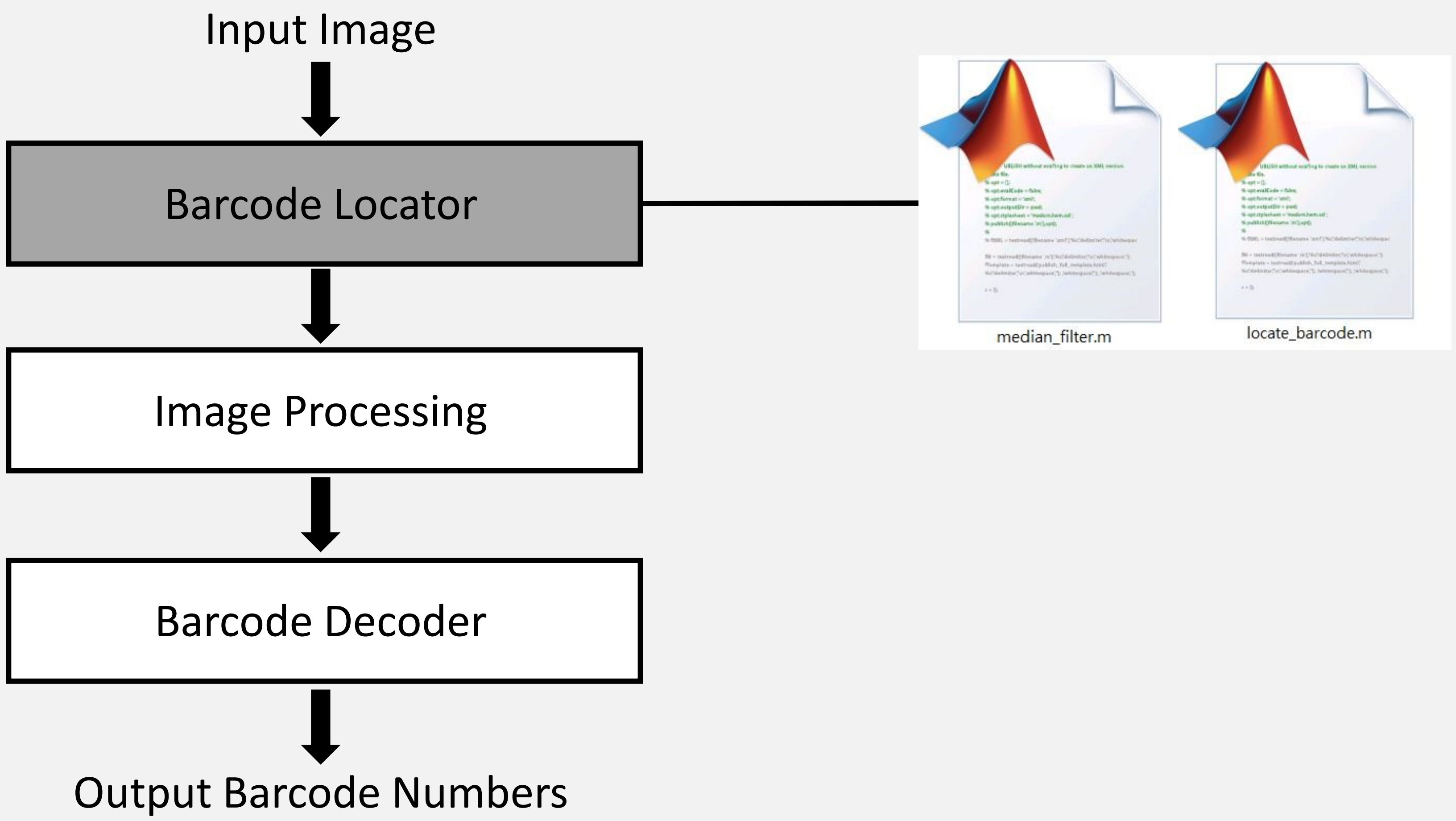
- Strategy
- Function.m
- Result
- Prospective
- Project demo

Strategy

Block diagram of the proposed procedure



Function.m

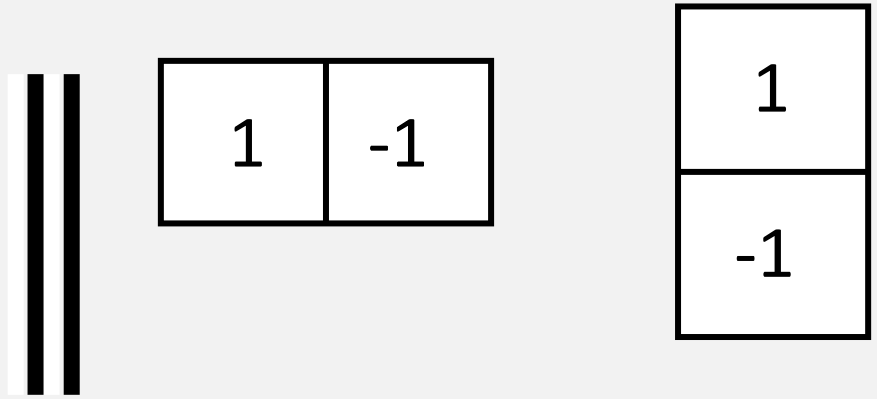

Input Image
Barcode Locator
Image Processing
Barcode Decoder
Output Barcode Numbers
median_filter.m
locate_barcode.m

locate_barcode.m & meaidan_filter.m

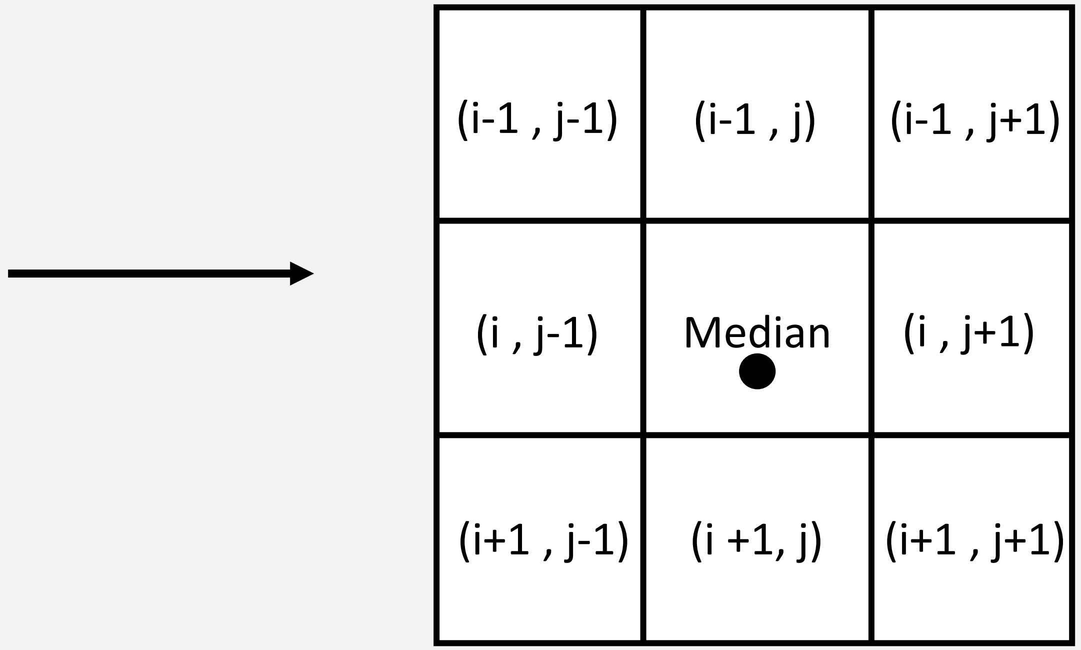
High horizontal gradient

Low vertical gradient

$$\left|\frac{\partial}{\partial x}B(x,y)\right| - \left|\frac{\partial}{\partial y}B(x,y)\right|$$



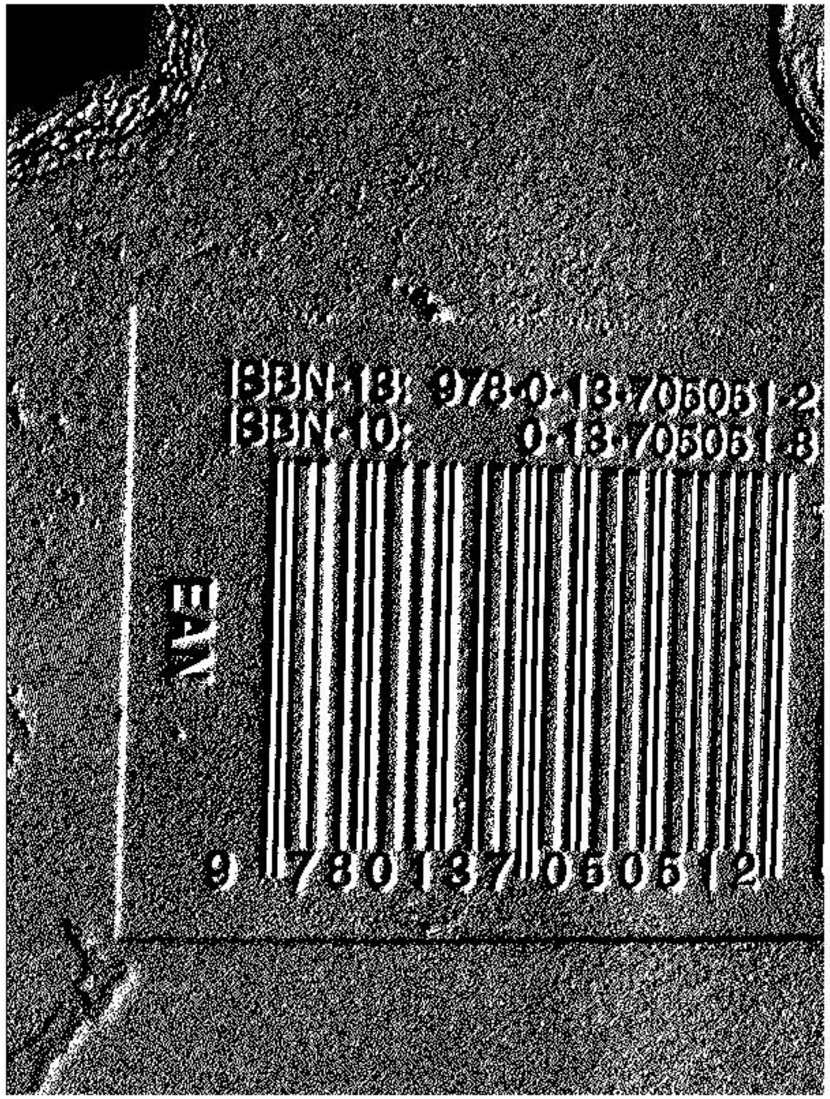
Median filter



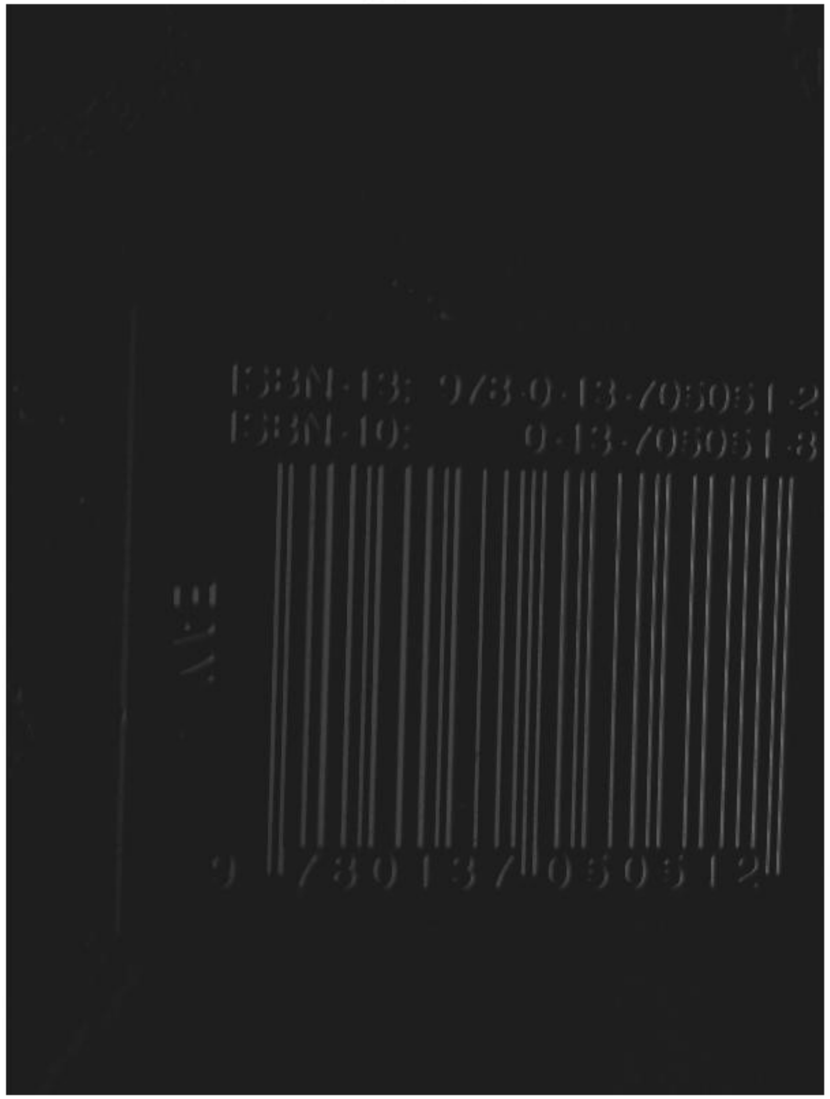
Pixels with relatively high value assumed to be where the barcode locates.

Test 1

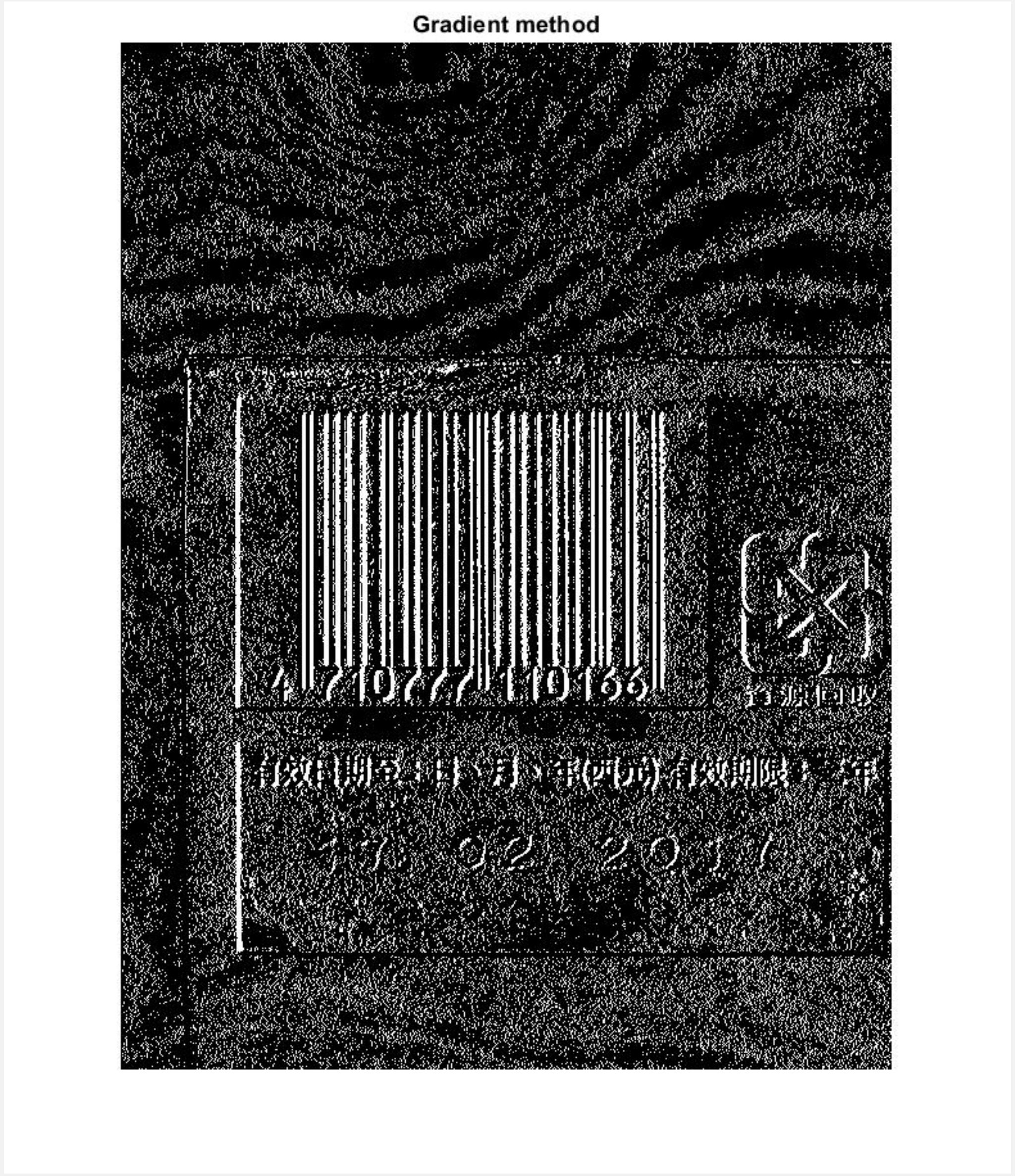
Gradient method

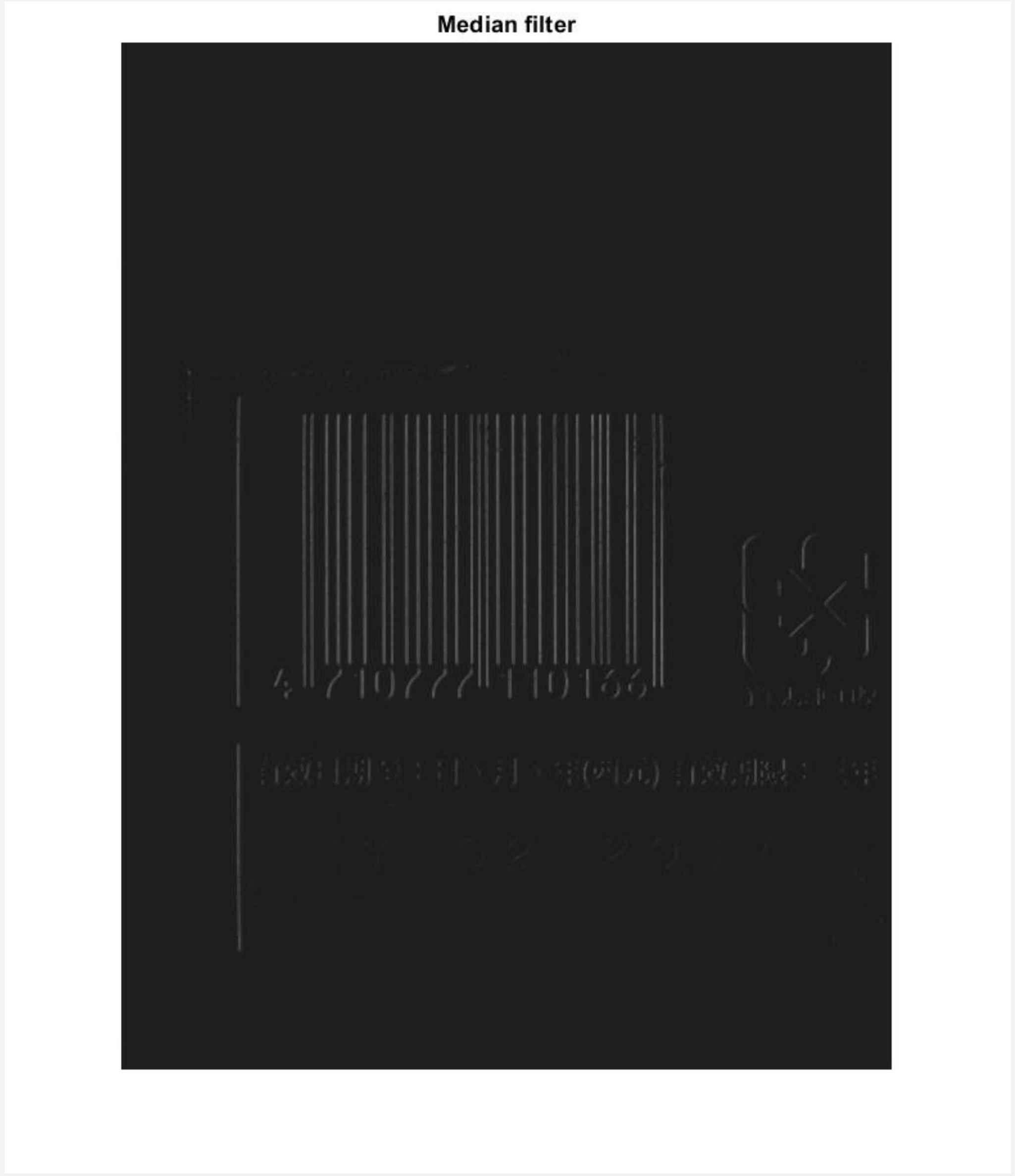


Median filter



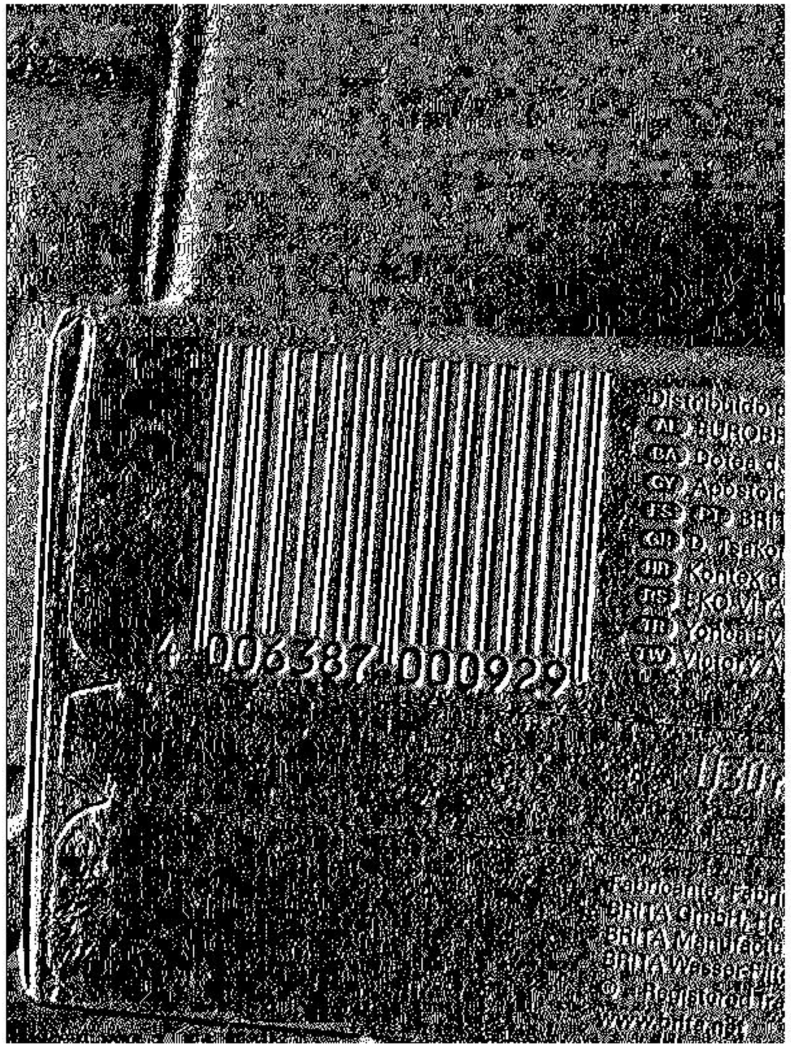
Test 2



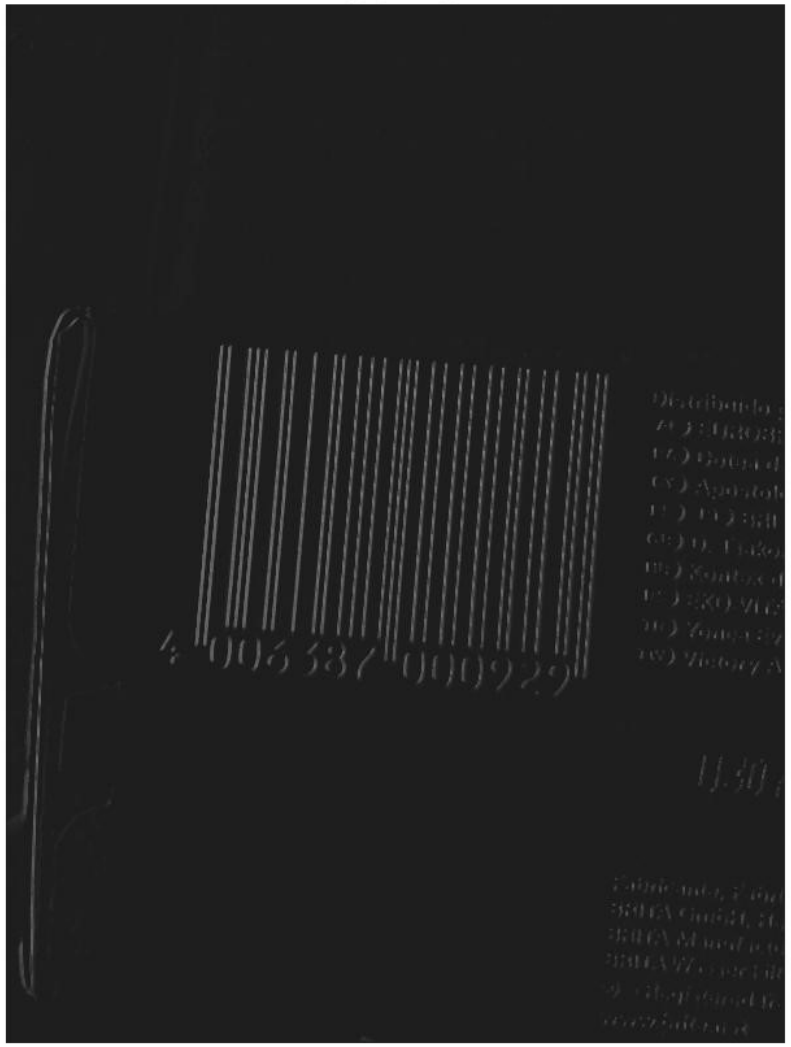


Test 3

Gradient method

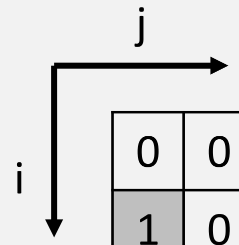


Median filter



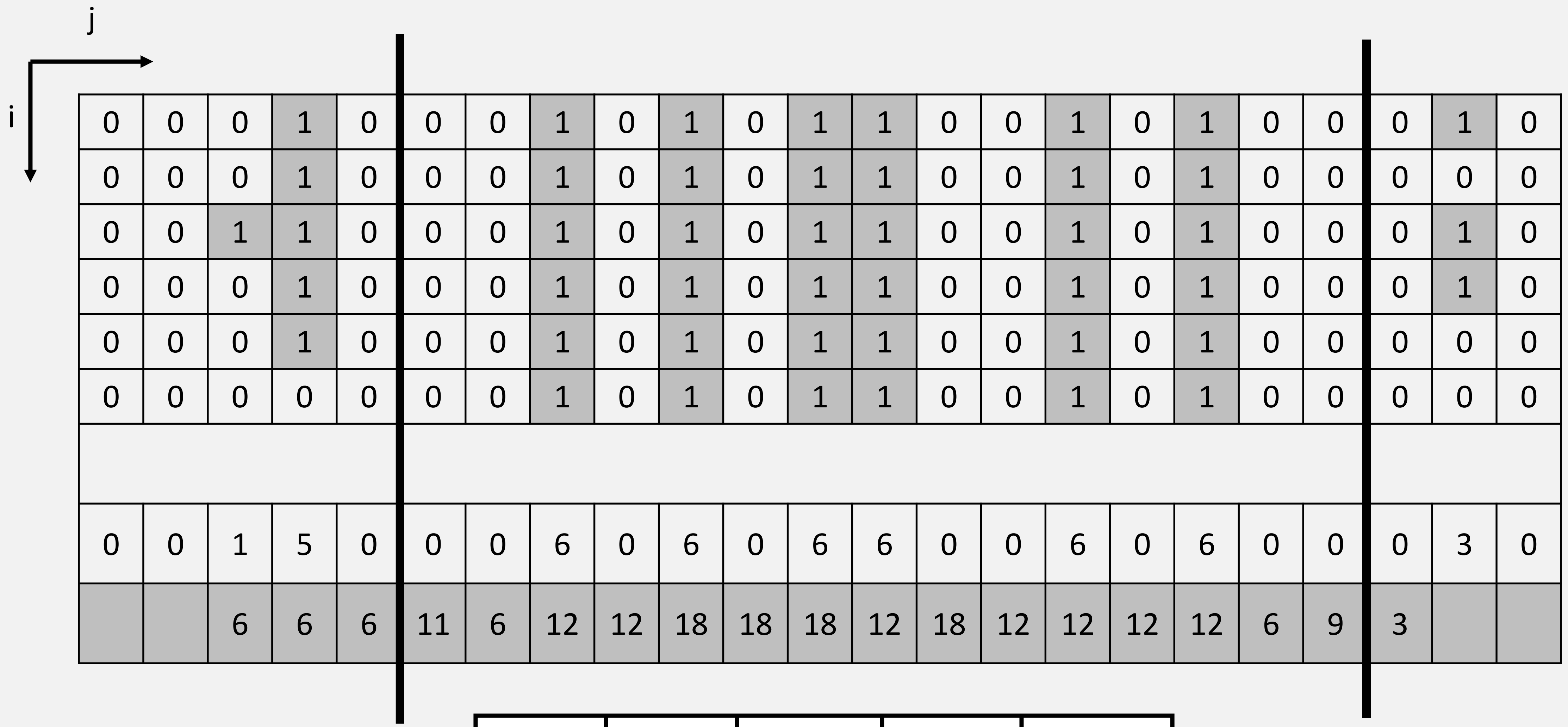
Two algorithms to find i & j directional boundaries

For i direction, rows in the barcode region appear to have high frequency of changing 1 to 0 and 0 to 1.



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2
1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	6
0	0	0	1	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0	0	1	0	14
0	0	0	1	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0	12
0	0	1	1	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0	0	1	0	14
0	0	0	1	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0	0	1	0	14
0	0	0	1	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0	12
0	0	0	0	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0	10
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2
1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	3

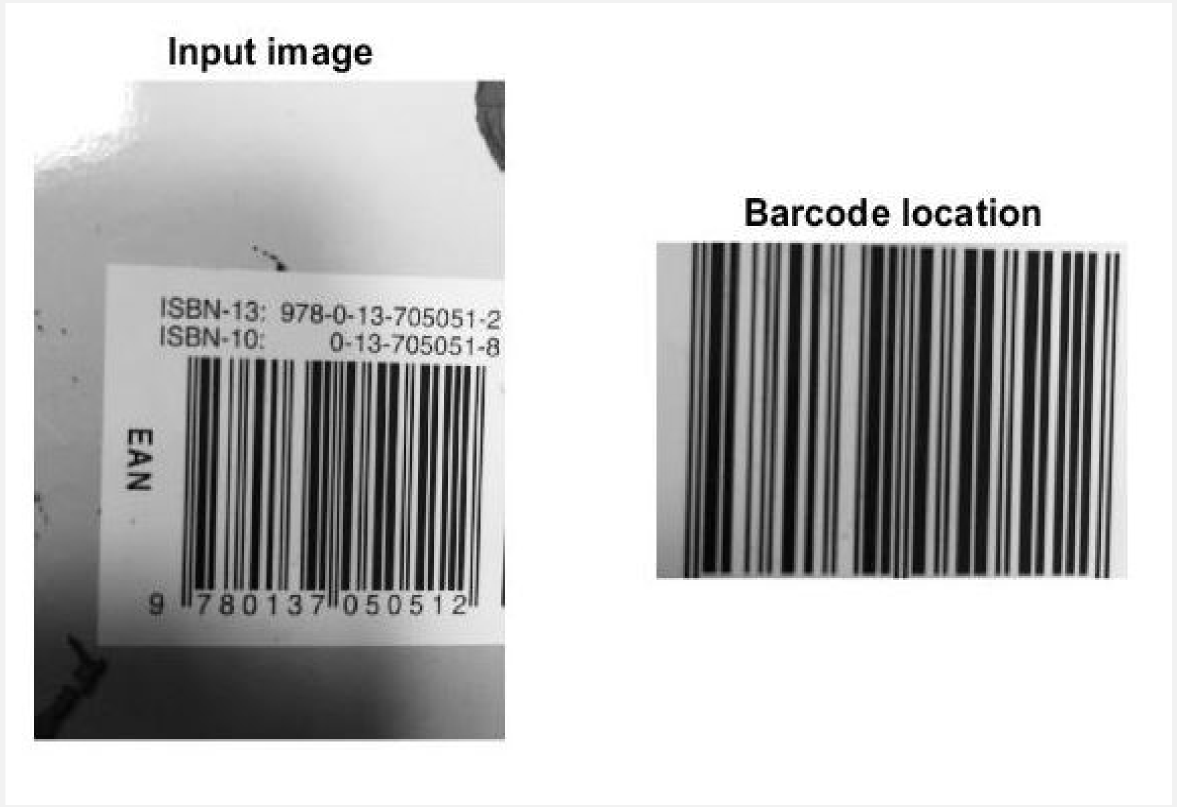
After applying convolution with a one row mask, the boundary in j direction will be obvious.

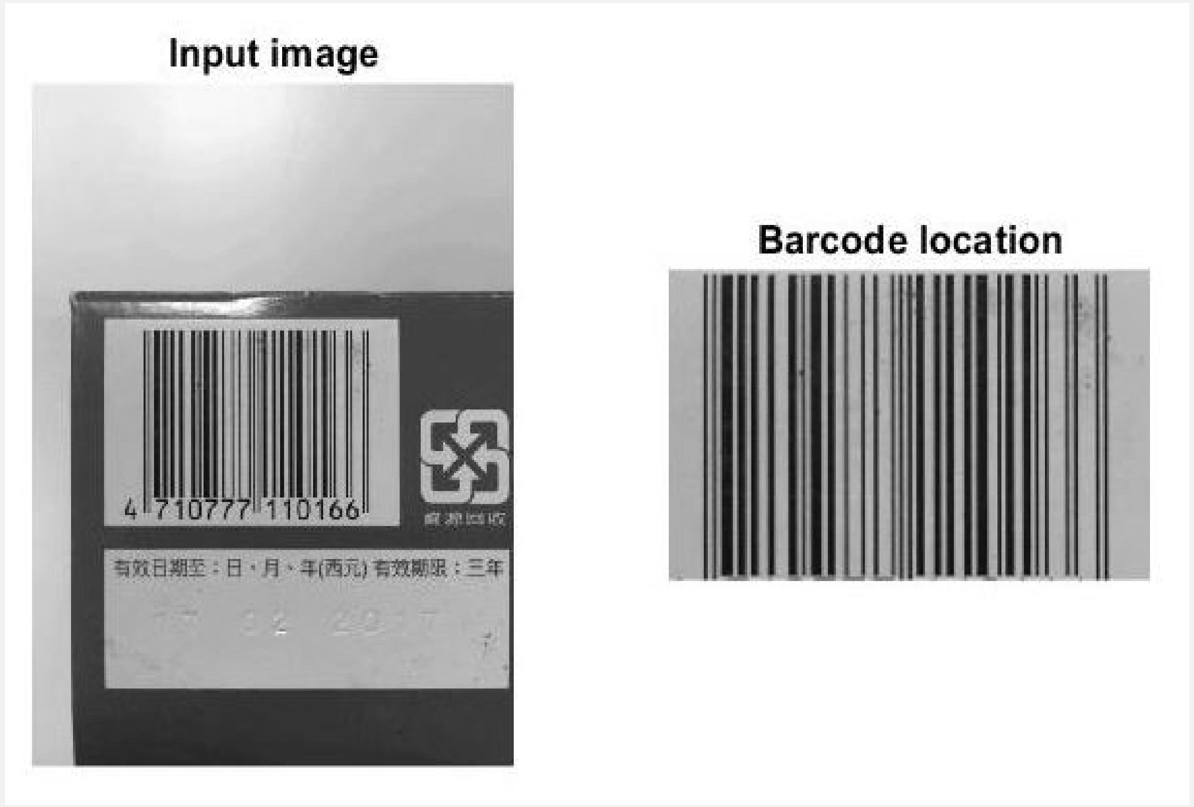


0	0	0	1	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0	0	1	0
0	0	0	1	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0
0	0	1	1	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0	0	1	0
0	0	0	1	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0	0	1	0
0	0	0	1	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0
0	0	1	5	0	0	0	6	0	6	0	6	6	0	0	6	0	6	0	0	0	3	0
		6	6	6	11	6	12	12	18	18	18	12	18	12	12	12	12	6	9	3		

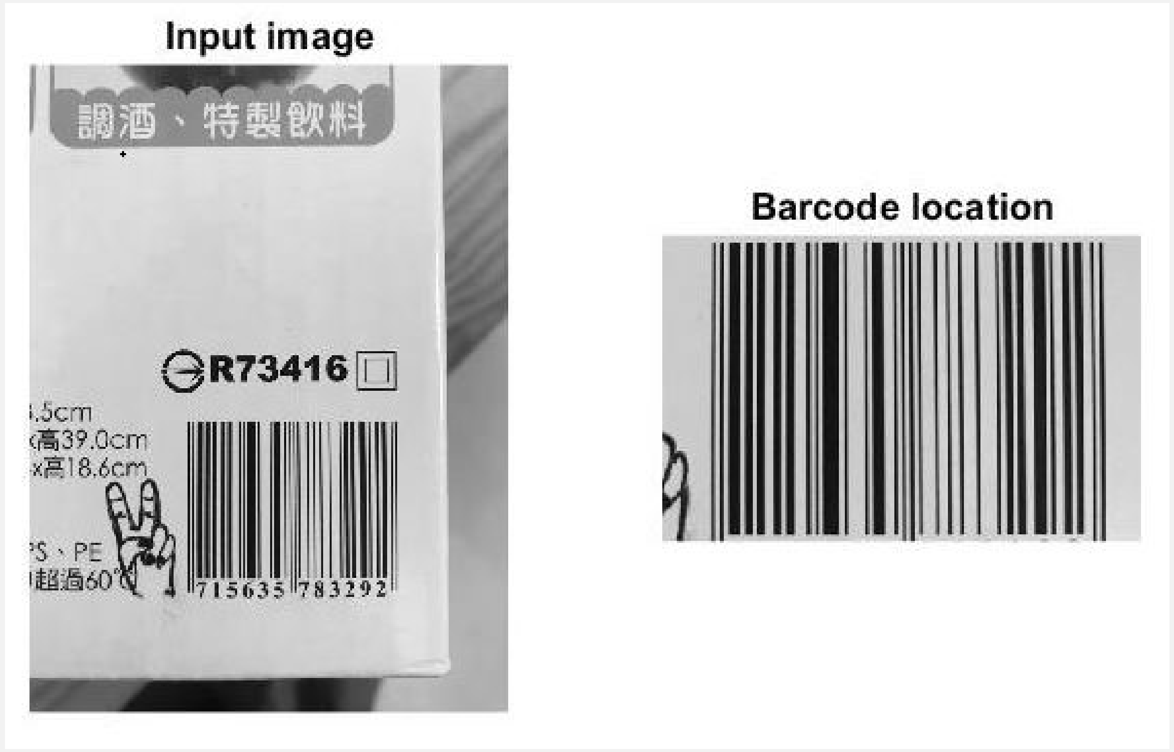
(i , j-2)	(i , j-1)	(i , j) ●	(i , j+1)	(i , j+2)

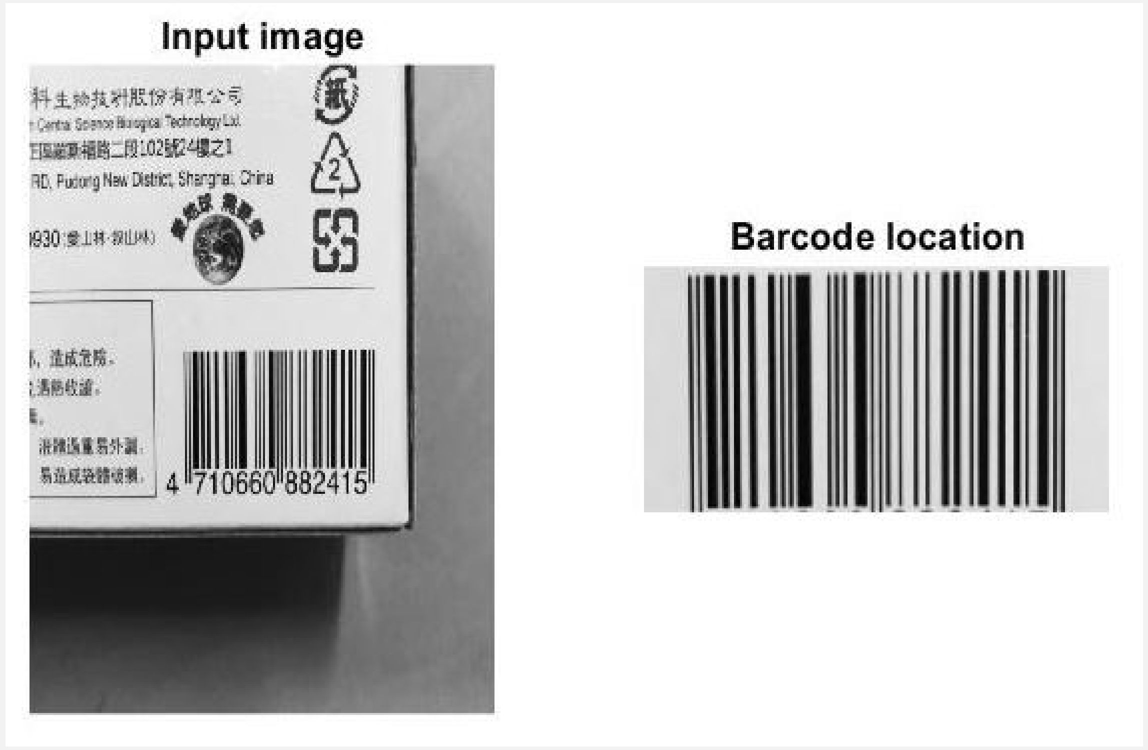
Test 1 & 2


Input image
ISBN-13: 978-0-13-705051-2
ISBN-10: 0-13-705051-8
EAN
9 780137 050512
Barcode location

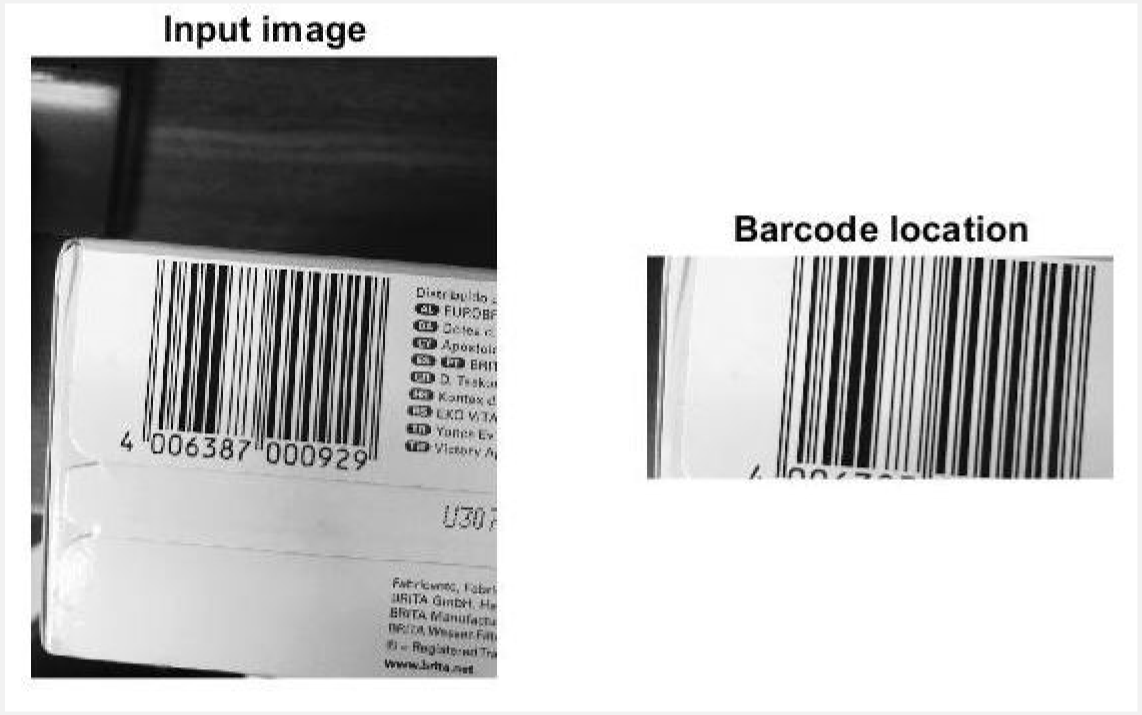

Input image
4 710777 110166
有效日期至：日・月・年(西元) 有效期限：三年
Barcode location

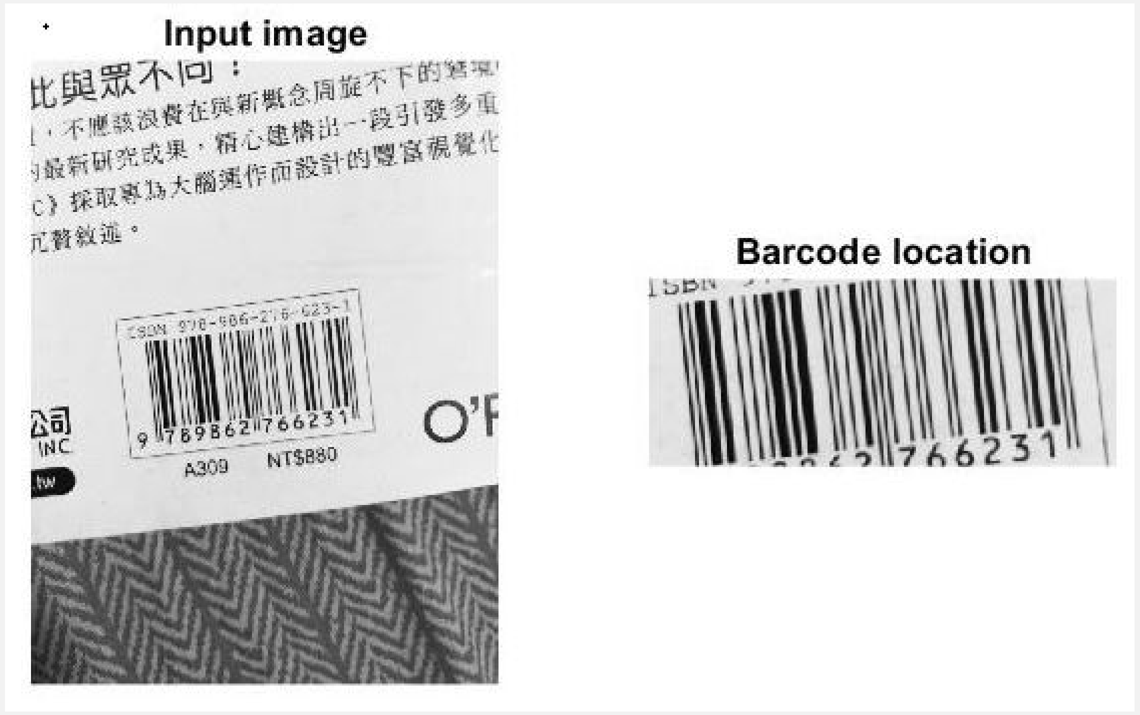
Test 3 & 4

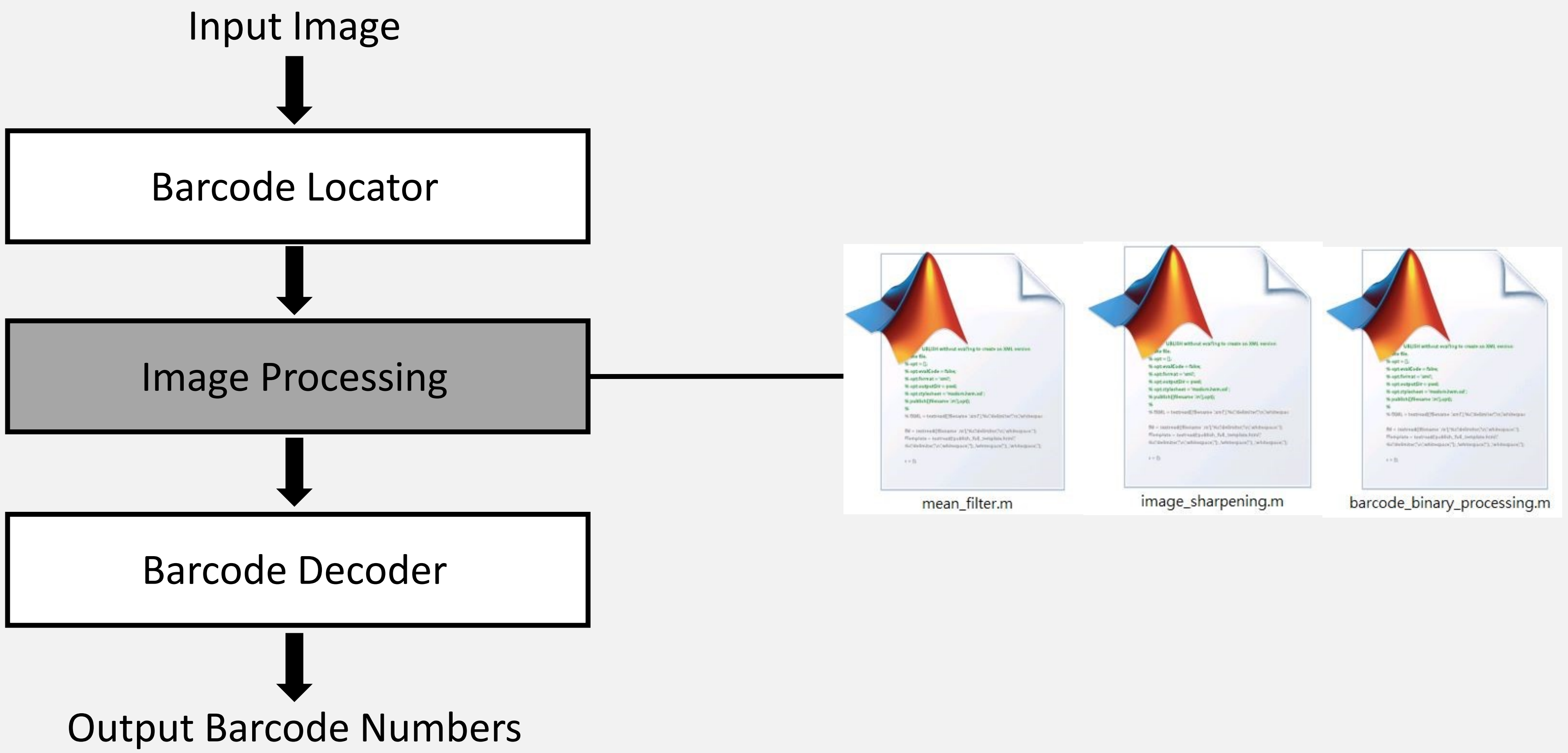

Input image
Barcode location


Input image
Barcode location

Test 5 & 6


Input image
Barcode location


Input image
Barcode location


Input Image
Barcode Locator
Image Processing
Barcode Decoder
Output Barcode Numbers
mean_filter.m
image_sharpening.m
barcode_binary_processing.m

mean_filter.m

Mean filter

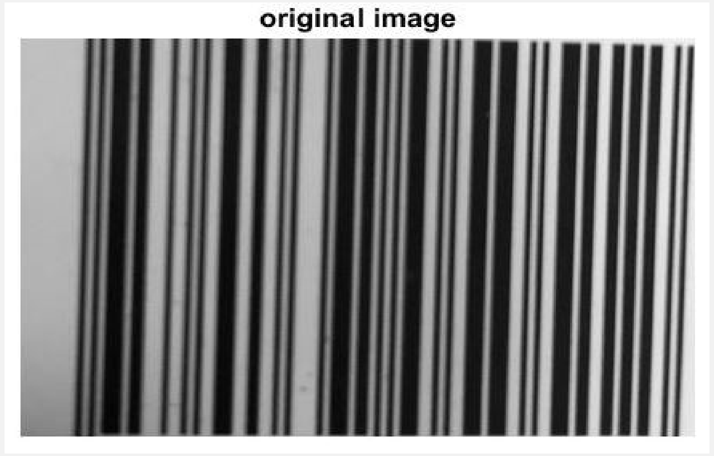
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$ ●	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

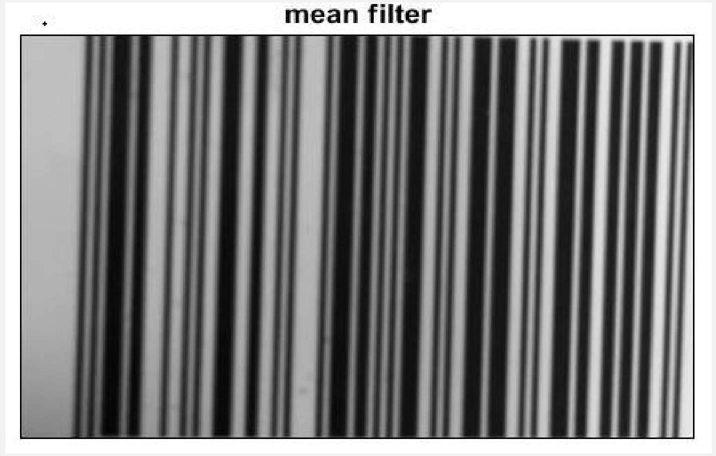
$\longrightarrow$

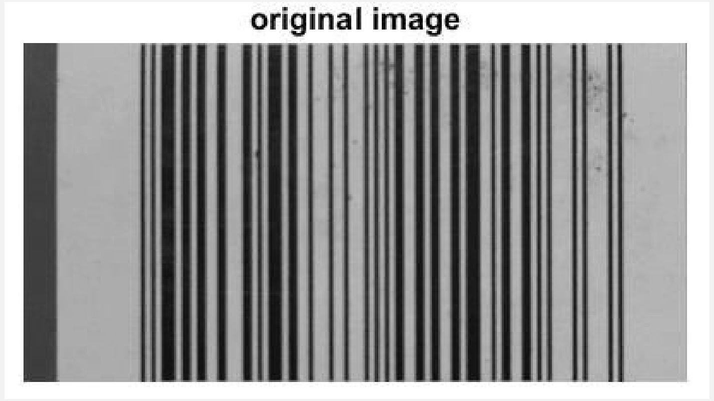
Any feature with a sharp discontinuity, like noise, will be enhanced by a Laplacian operator.

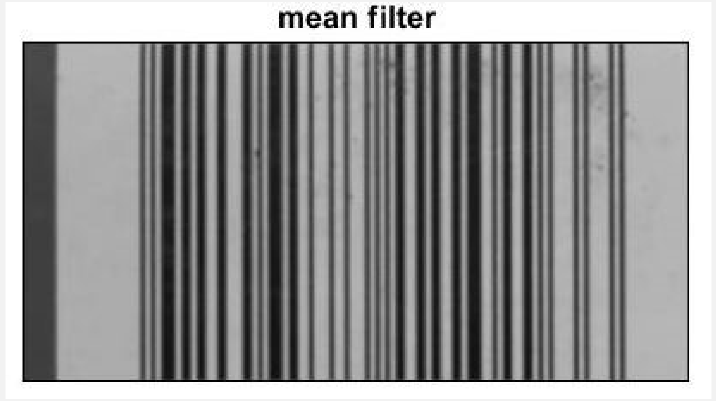
Noises in an image have to be smoothed in advance.

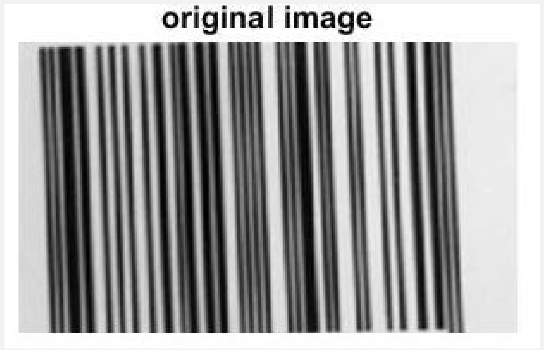
Test 1 & 2 & 3

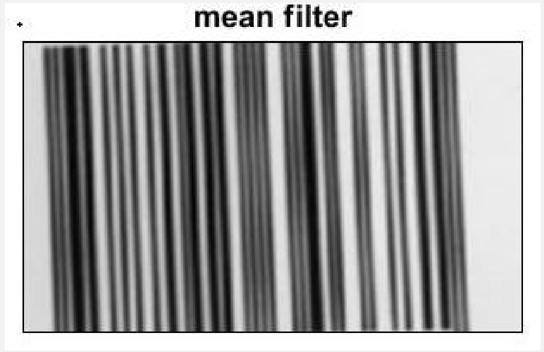

original image


mean filter


original image


mean filter


original image


mean filter

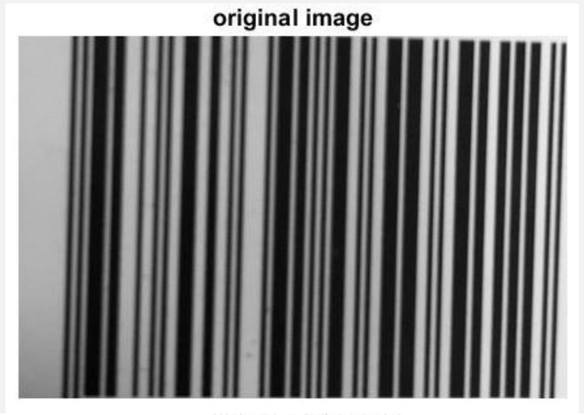
image_sharpening.m

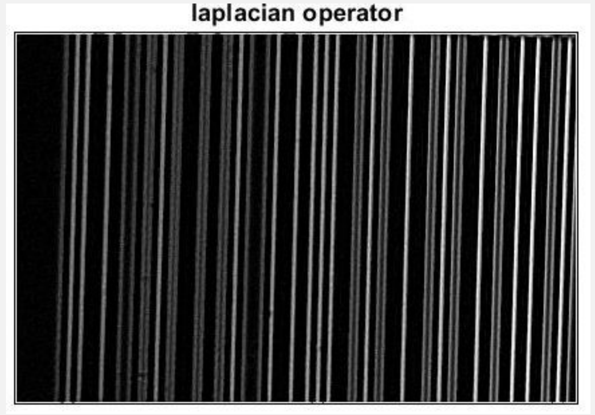
Laplacian operator

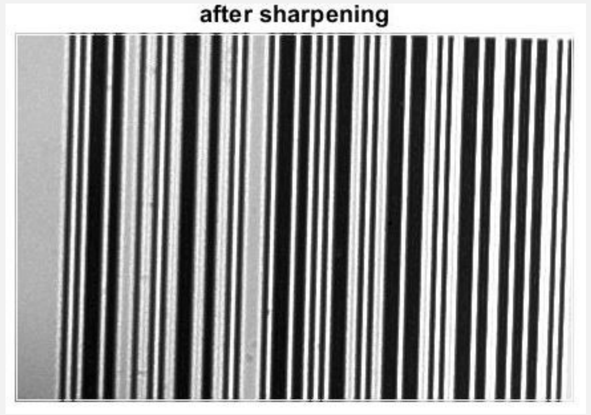
-1	-4	-1
-4	20 ●	-4
-1	-4	-1

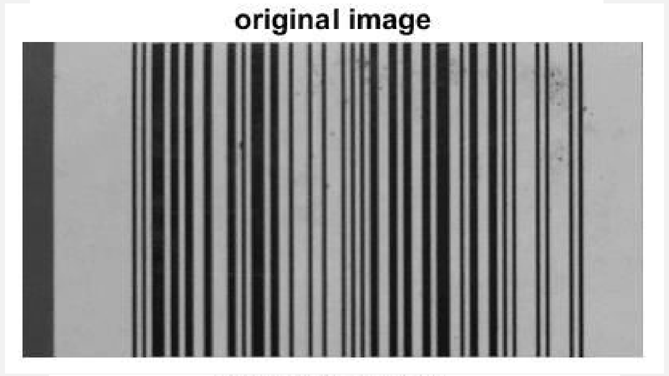
Add the result to the original image in order to sharpen the image.

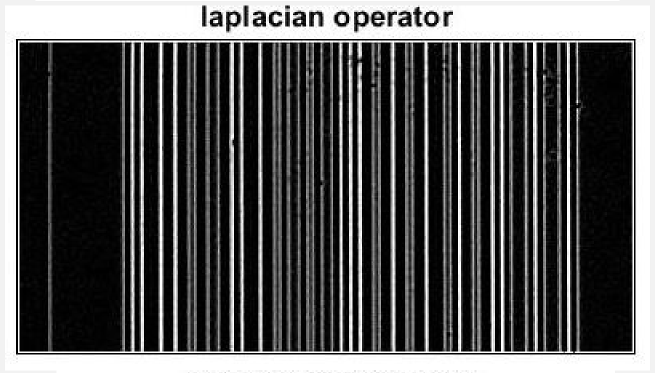
Test 1 & 2 & 3

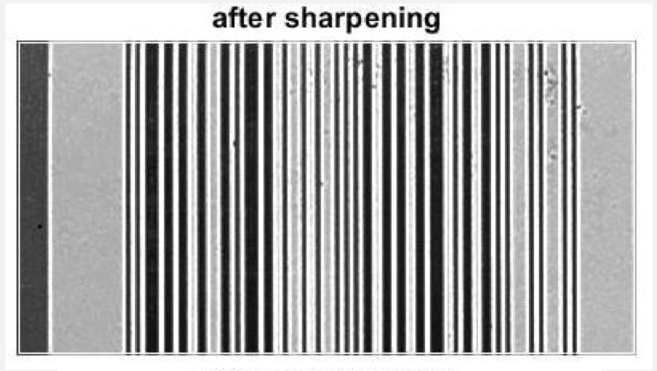

original image

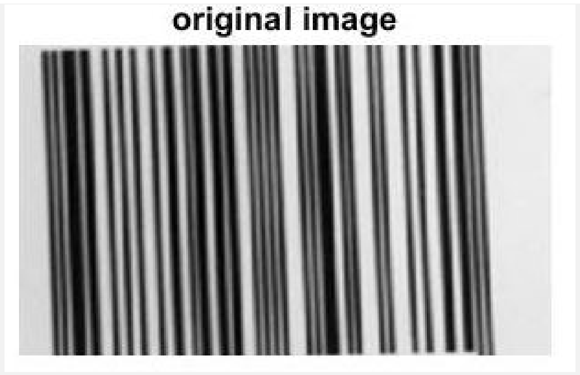

laplacian operator

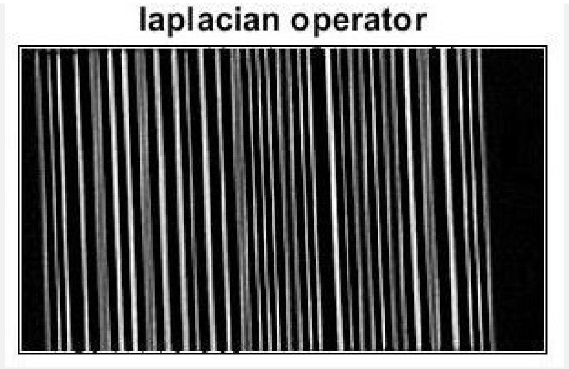

after sharpening

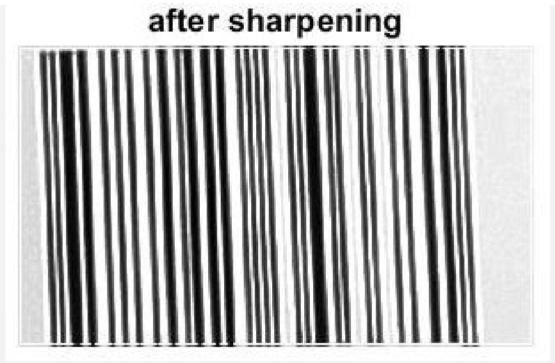

original image


laplacian operator


after sharpening

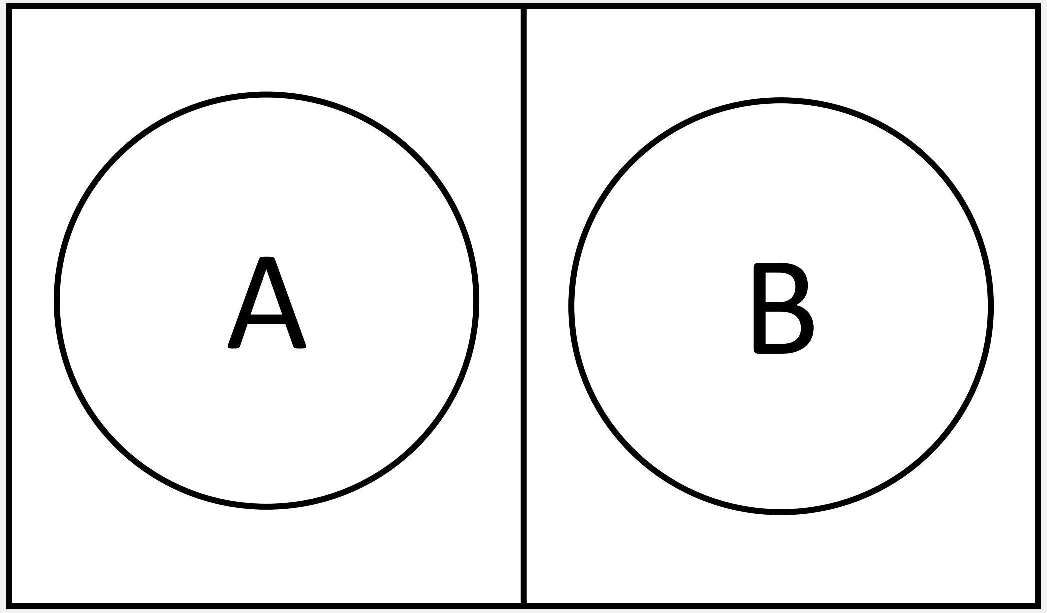

original image


laplacian operator


after sharpening

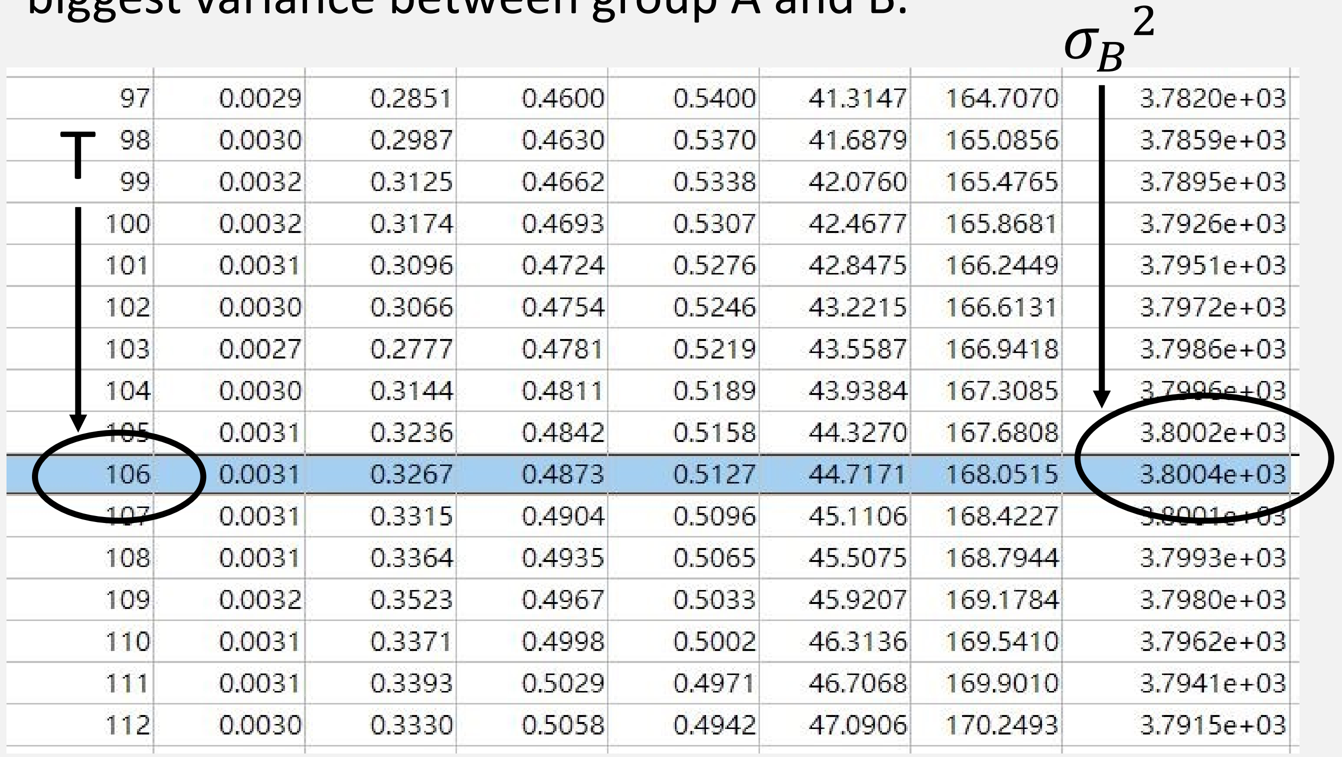
barcode_binary_processing.m

Ostu's thresholding



1. The technique is to test each gray value which exists in the image to serve as a threshold and use it to separate the image in to two groups A and B.

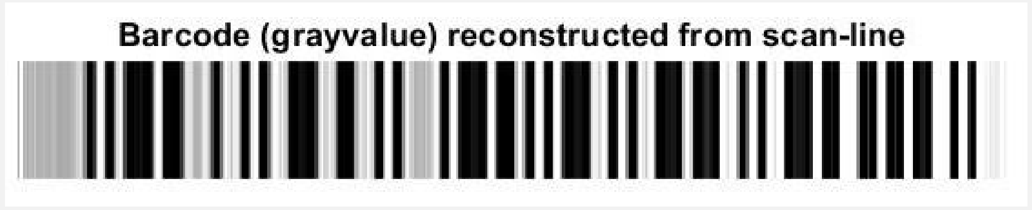
2. There is only one gray value that corresponds to the biggest variance between group A and B.

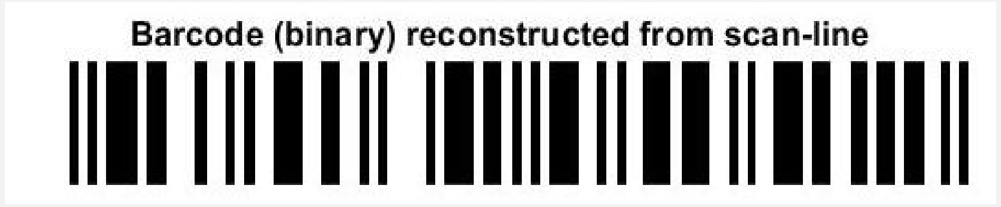


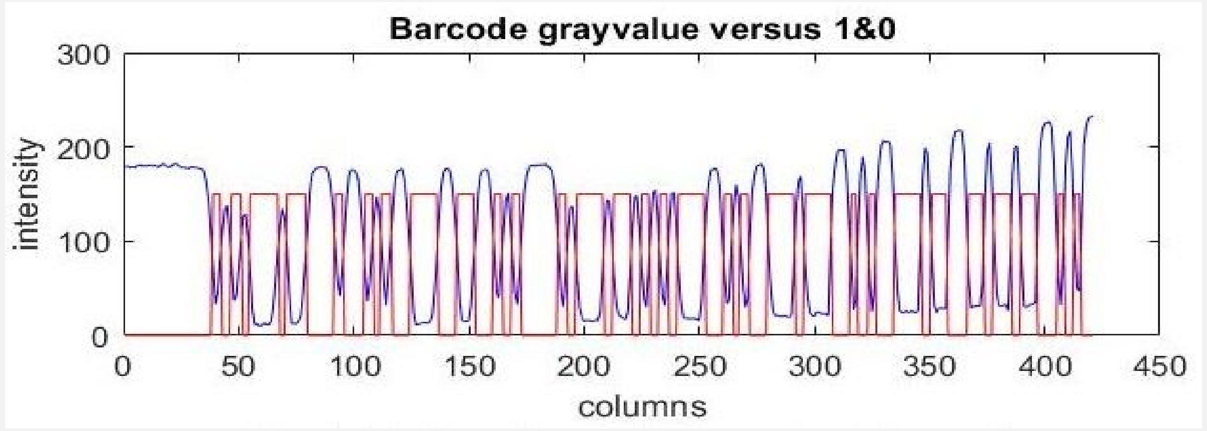
97	0.0029	0.2851	0.4600	0.5400	41.3147	164.7070	3.7820e+03
98	0.0030	0.2987	0.4630	0.5370	41.6879	165.0856	3.7859e+03
99	0.0032	0.3125	0.4662	0.5338	42.0760	165.4765	3.7895e+03
100	0.0032	0.3174	0.4693	0.5307	42.4677	165.8681	3.7926e+03
101	0.0031	0.3096	0.4724	0.5276	42.8475	166.2449	3.7951e+03
102	0.0030	0.3066	0.4754	0.5246	43.2215	166.6131	3.7972e+03
103	0.0027	0.2777	0.4781	0.5219	43.5587	166.9418	3.7986e+03
104	0.0030	0.3144	0.4811	0.5189	43.9384	167.3085	3.7996e+03
105	0.0031	0.3236	0.4842	0.5158	44.3270	167.6808	3.8002e+03
106	0.0031	0.3267	0.4873	0.5127	44.7171	168.0515	3.8004e+03
107	0.0031	0.3315	0.4904	0.5096	45.1106	168.4227	3.8001e+03
108	0.0031	0.3364	0.4935	0.5065	45.5075	168.7944	3.7993e+03
109	0.0032	0.3523	0.4967	0.5033	45.9207	169.1784	3.7980e+03
110	0.0031	0.3371	0.4998	0.5002	46.3136	169.5410	3.7962e+03
111	0.0031	0.3393	0.5029	0.4971	46.7068	169.9010	3.7941e+03
112	0.0030	0.3330	0.5058	0.4942	47.0906	170.2493	3.7915e+03

After thresholding, the middle row is determined to be a scan line and this information will be sent to the read_barcode.m.

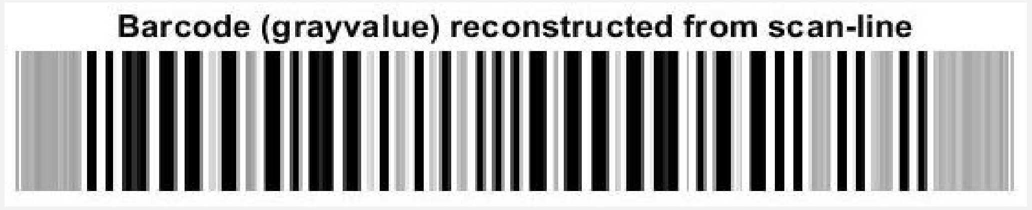
Test 1

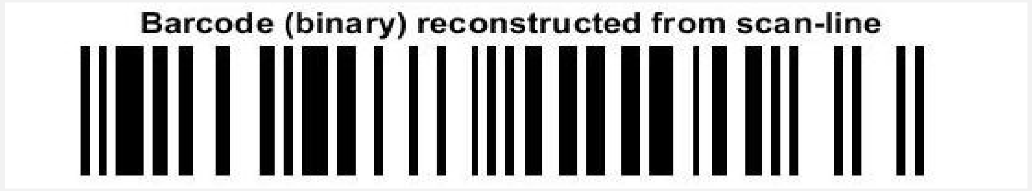


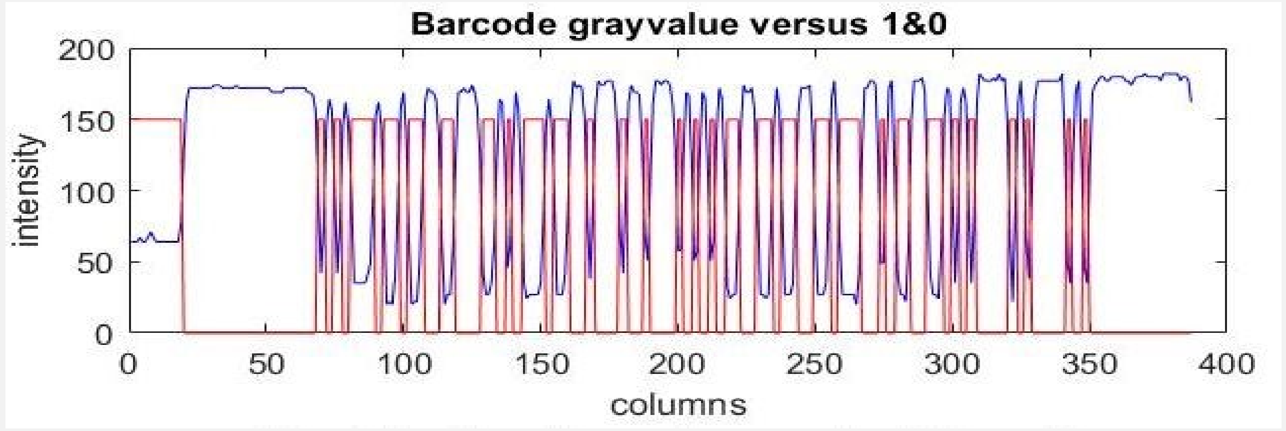




Test 2



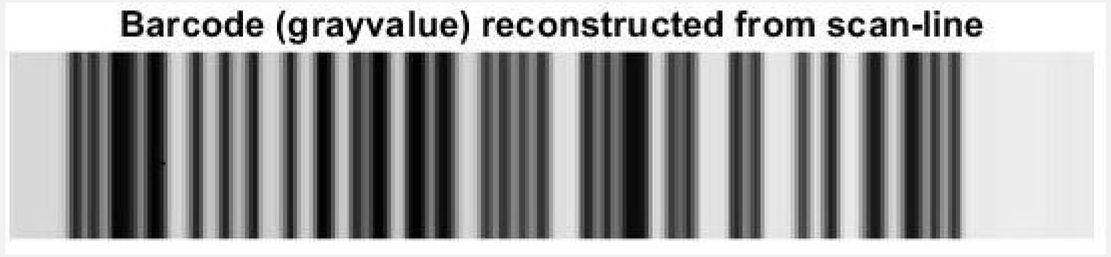


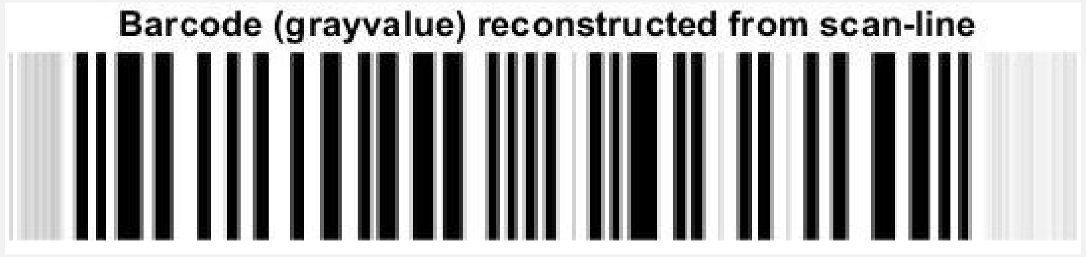


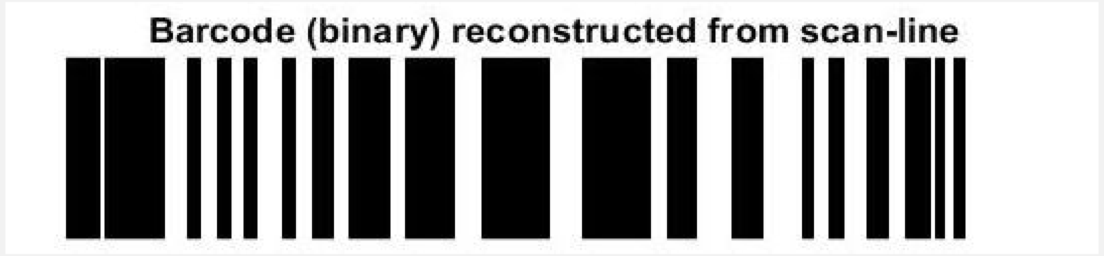
Test 3

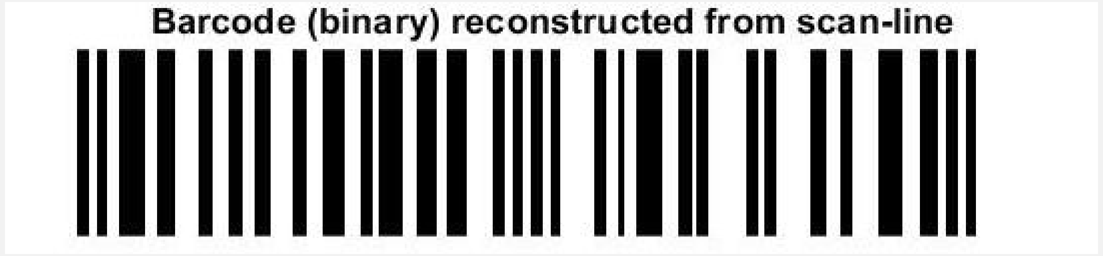
Without sharpening

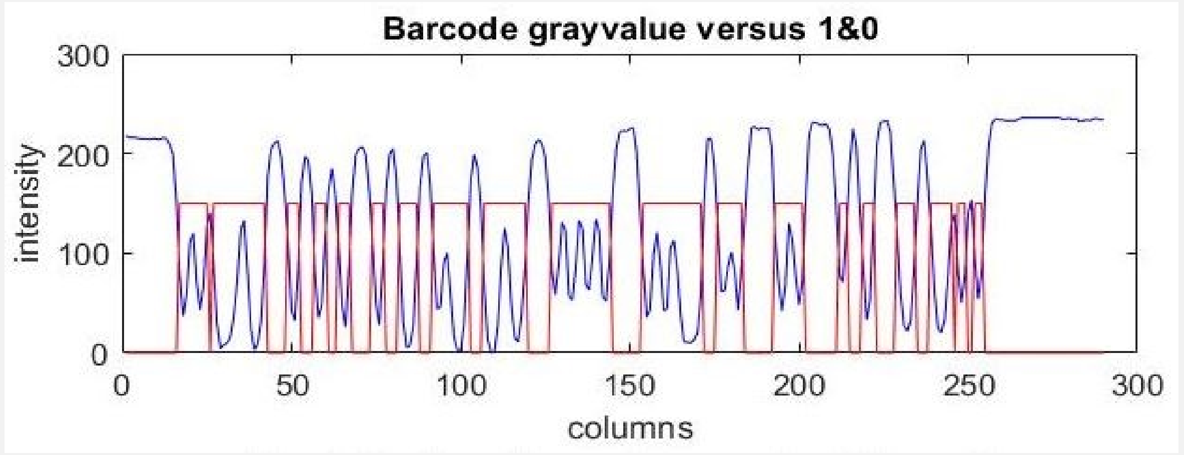
After sharpening

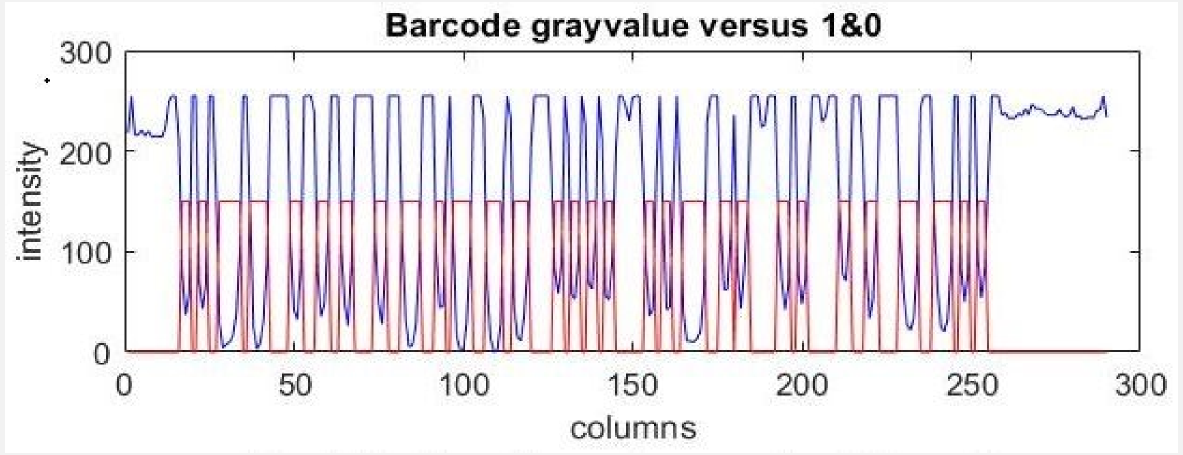


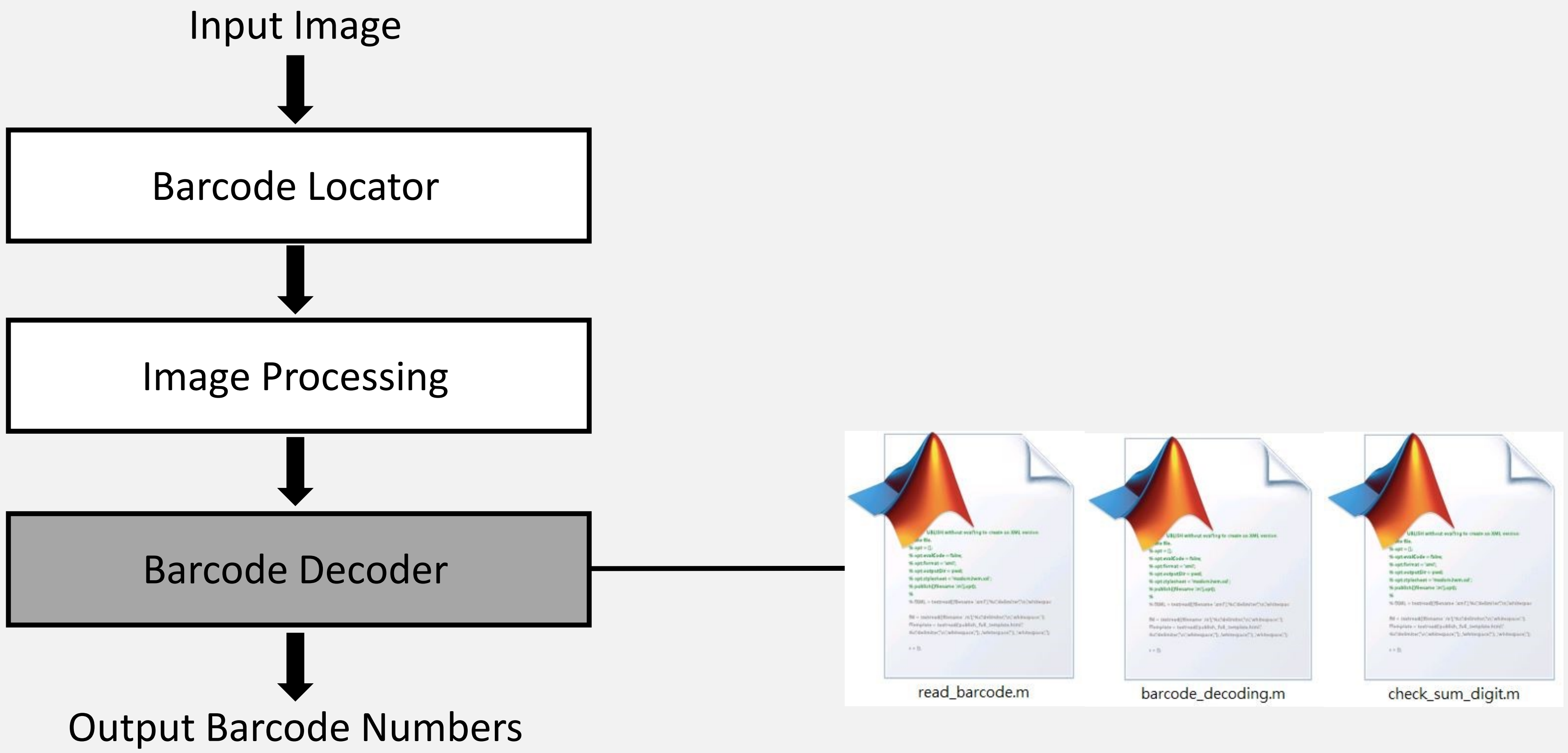






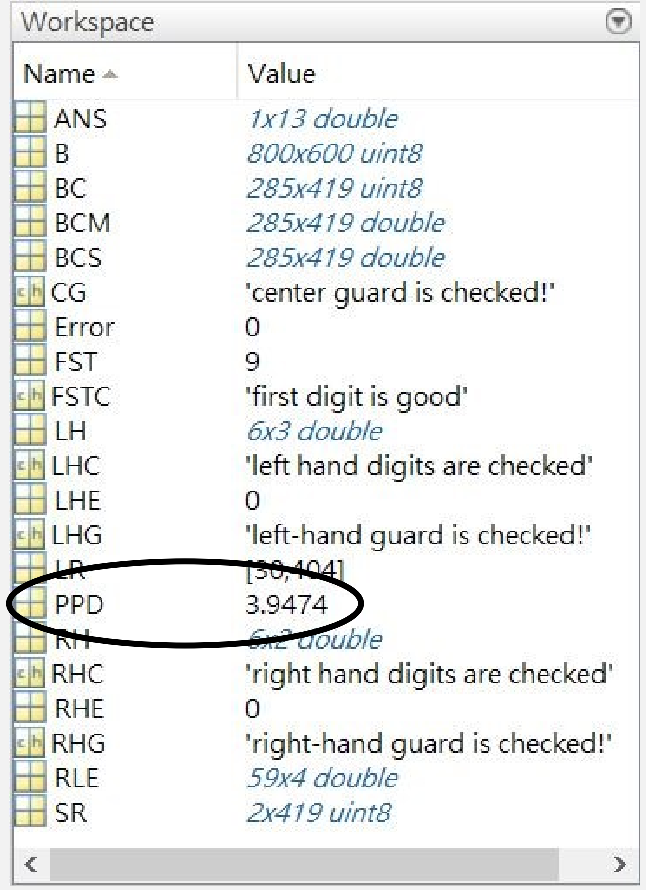





Input Image
Barcode Locator
Image Processing
Barcode Decoder
Output Barcode Numbers
read_barcode.m
barcode_decoding.m
check_sum_digit.m

read_barcode.m

Total pixels from left hand guard to right hand guard divided by 95 = pixel per digit.

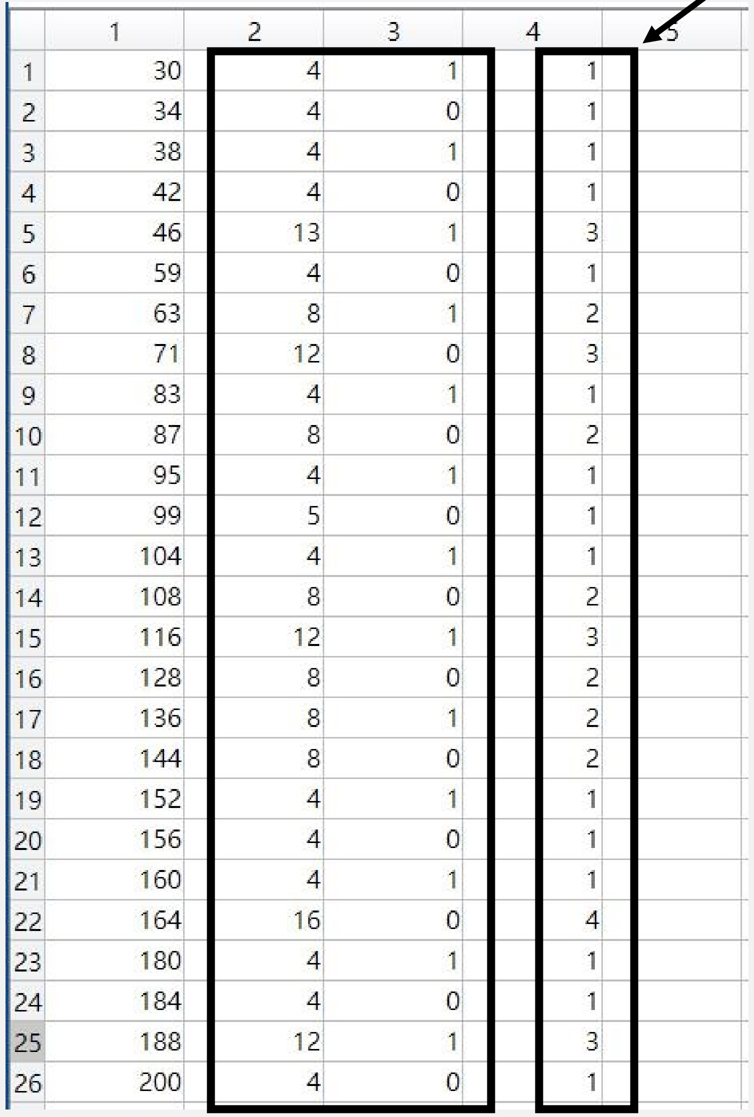


Workspace

Name	Value
ANS	*1x13 double*
B	*800x600 uint8*
BC	*285x419 uint8*
BCM	*285x419 double*
BCS	*285x419 double*
CG	'center guard is checked!'
Error	0
FST	9
FSTC	'first digit is good'
LH	*6x3 double*
LHC	'left hand digits are checked'
LHE	0
LHG	'left-hand guard is checked!'
LR	[30,404]
PPD	3.9474
RH	*6x2 double*
RHC	'right hand digits are checked'
RHE	0
RHG	'right-hand guard is checked!'
RLE	*59x4 double*
SR	*2x419 uint8*

Divided by PPD & read by barcode_decoding.m

Run-length encoding



	1	2	3	4	5
1	30	4	1	1	
2	34	4	0	1	
3	38	4	1	1	
4	42	4	0	1	
5	46	13	1	3	
6	59	4	0	1	
7	63	8	1	2	
8	71	12	0	3	
9	83	4	1	1	
10	87	8	0	2	
11	95	4	1	1	
12	99	5	0	1	
13	104	4	1	1	
14	108	8	0	2	
15	116	12	1	3	
16	128	8	0	2	
17	136	8	1	2	
18	144	8	0	2	
19	152	4	1	1	
20	156	4	0	1	
21	160	4	1	1	
22	164	16	0	4	
23	180	4	1	1	
24	184	4	0	1	
25	188	12	1	3	
26	200	4	0	1	

barcode_decoding.m

Left-hand guard	Center guard	Right-hand guard
111	11111	111

Character set encoding table

Digit	Left-hand	
	Right-hand	
	Odd parity	Even parity
0	3211	1123
1	2221	1222
2	2122	2212
3	1411	1141
4	1132	2311
5	1231	1321
6	1114	4111
7	1312	2131
8	1213	3121
9	3112	2113

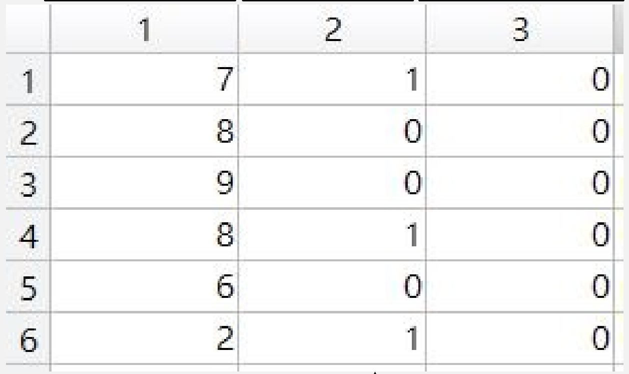
3+6*4+5+6*4+3 = 59 rows

To decode the barcode in a run-length encoding way can avoid the cumulative error caused by the image noises.

check_sum_digit.m

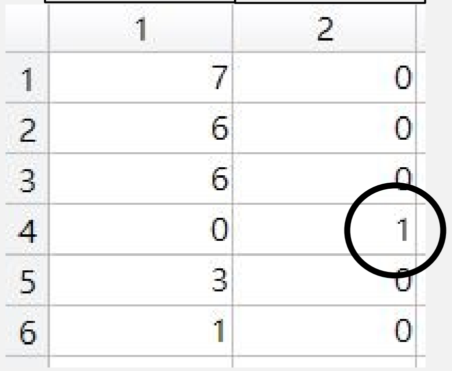
If there is only one error in barcode_decoding.m, the error can be fixed by the checksum digit.

For example:



	Left	1st	errors
	1	2	3
1	7	1	0
2	8	0	0
3	9	0	0
4	8	1	0
5	6	0	0
6	2	1	0

First digits = 9

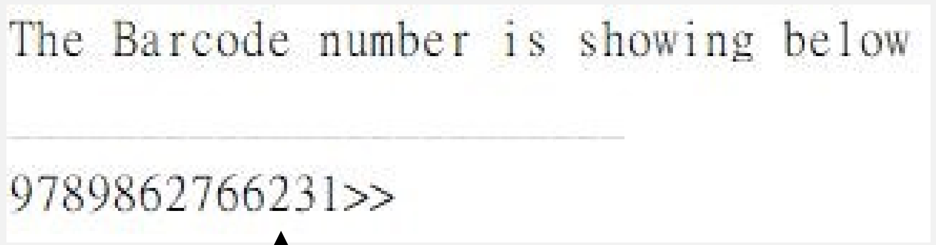


	Right	errors
	1	2
1	7	0
2	6	0
3	6	0
4	0	1
5	3	0
6	1	0

The checksum digit rule

(3+6+7+6+9+7)*3 + (0+6+2+8+8+9) +1 = 148

Take ceiling(148/10) -148 = 2

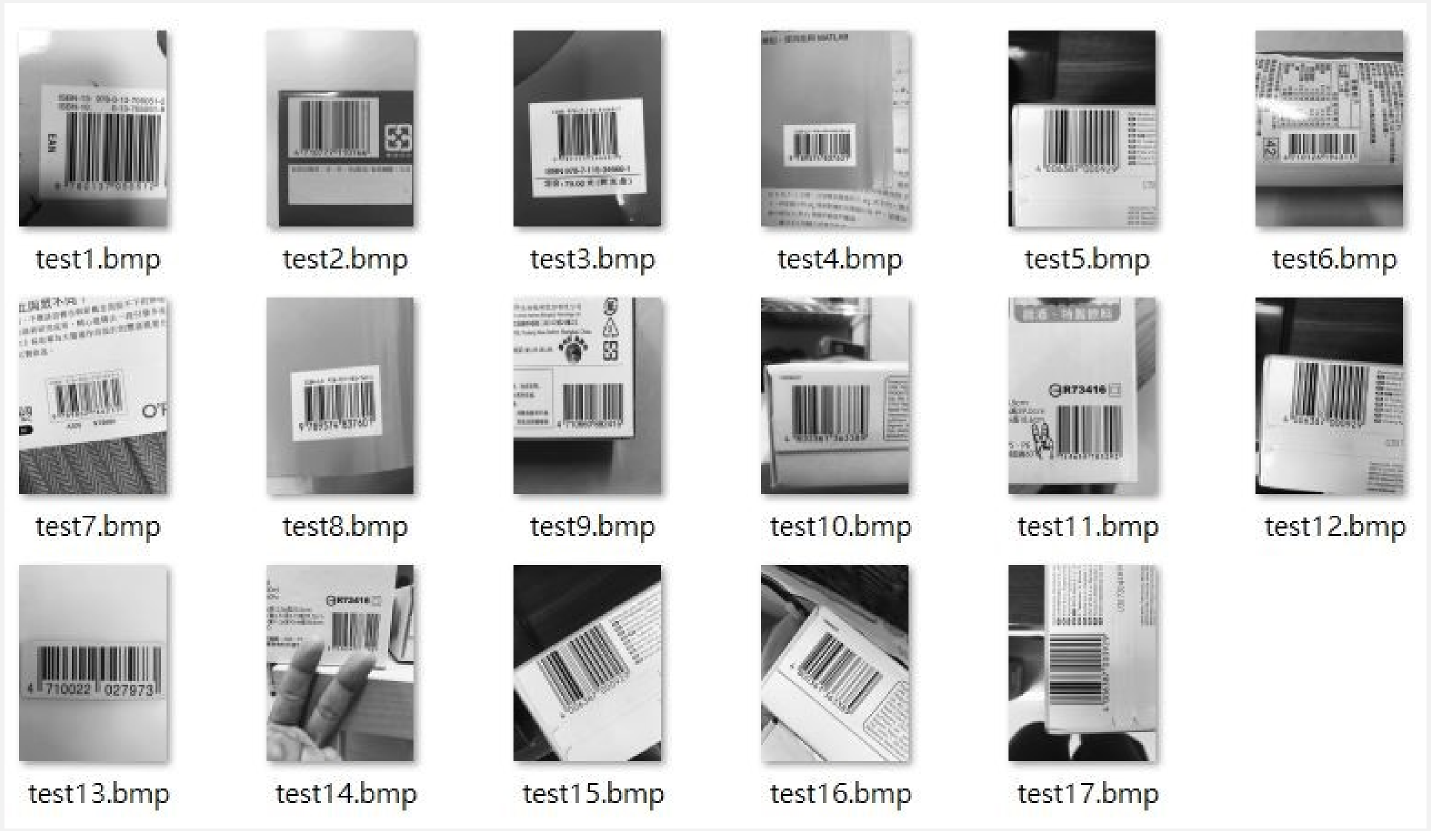


```
The Barcode number is showing below

9789862766231>>
```

Error fixed

Result



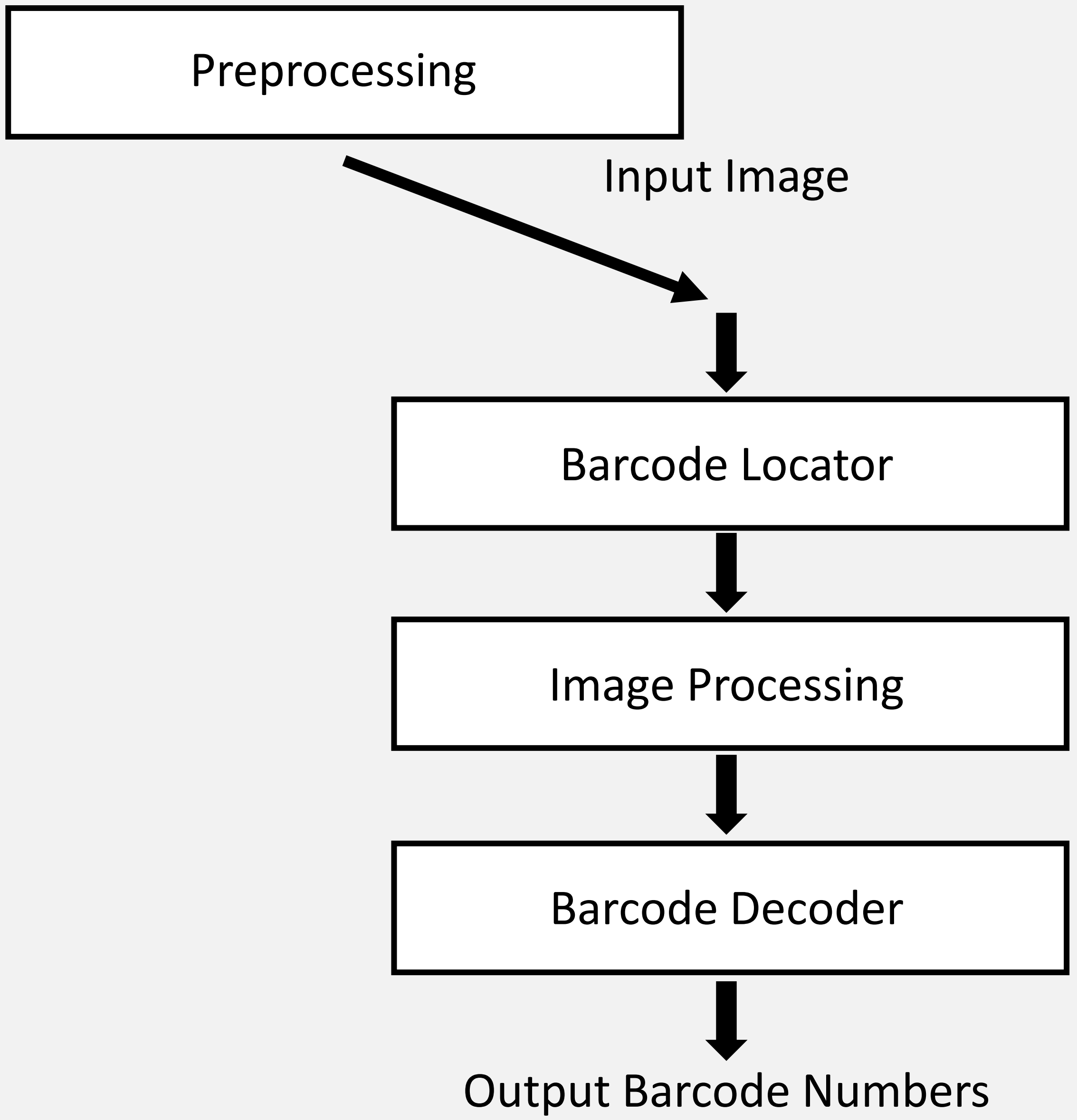
Test 1-12	No problem
Test 13	1 error, solved by checksum digit.
Test 3,14	Too much errors
Test 15-17	The program can’t solve

Prospective

The shortcoming of my barcode reader is that this program can only deal with horizontal or nearly horizontal barcode.

To fix this problem, a preprocessing technique have to be developed.

A Sobel operator and a Hough transform could be adopted.



Project demo