Telecom Saint-Etienne

Image Processing Project
Object recognition for coins calculation
Report

by

Yutong YAN

Heng ZHANG

in the

FISE2

IMAGE2

January 2017

Contents

Chapter 1

Introduction

In this image project, we will realize a coin counter with MATLAB code. To finish this project, we separate it into 4 parts:

$Image\ acquisition\&Calibration \rightarrow Preprocessing \rightarrow Segmentation \rightarrow Recognition \rightarrow Graphic\ Interface$

FIGURE 1.1: an example of dataset

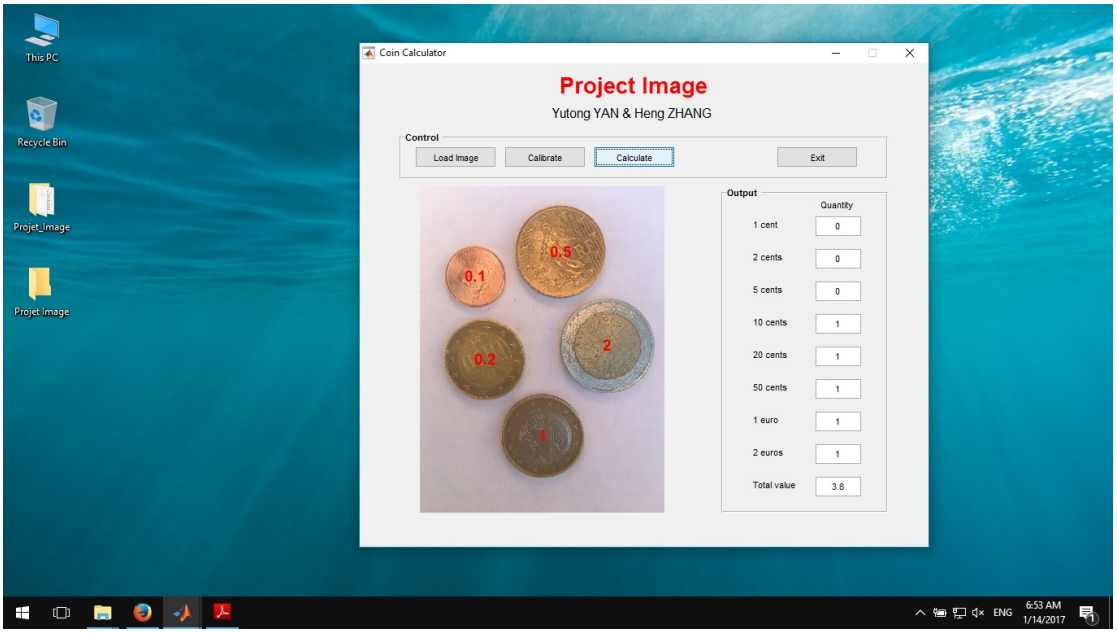

FIGURE 1.2: graphic interface

Chapter 2

Methodology

2.1 Image acquisition & Calibration

For this part, we need to take at least 52 photos of one or many coins with our cell phones. The height must be the same for each photo. We compress these photos to make sure the size is not too big.

After taking these photos, we need to calculate the Scaling factor as well as the RGB value of Gold, Silver and Copper. To make sure our data is precise, we set up a log to record all the data. Here is the screenshot of our log:

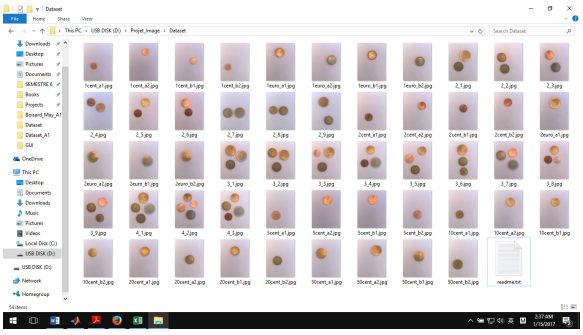

FIGURE 2.1: dataset

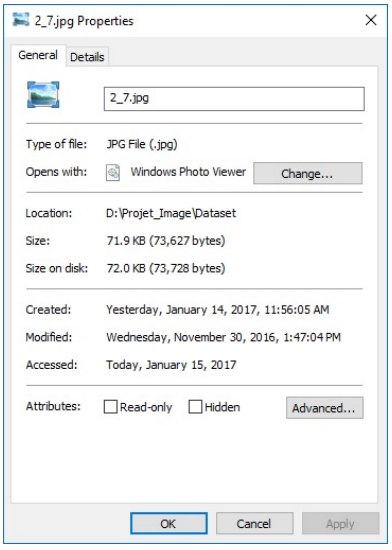

FIGURE 2.2: photo size

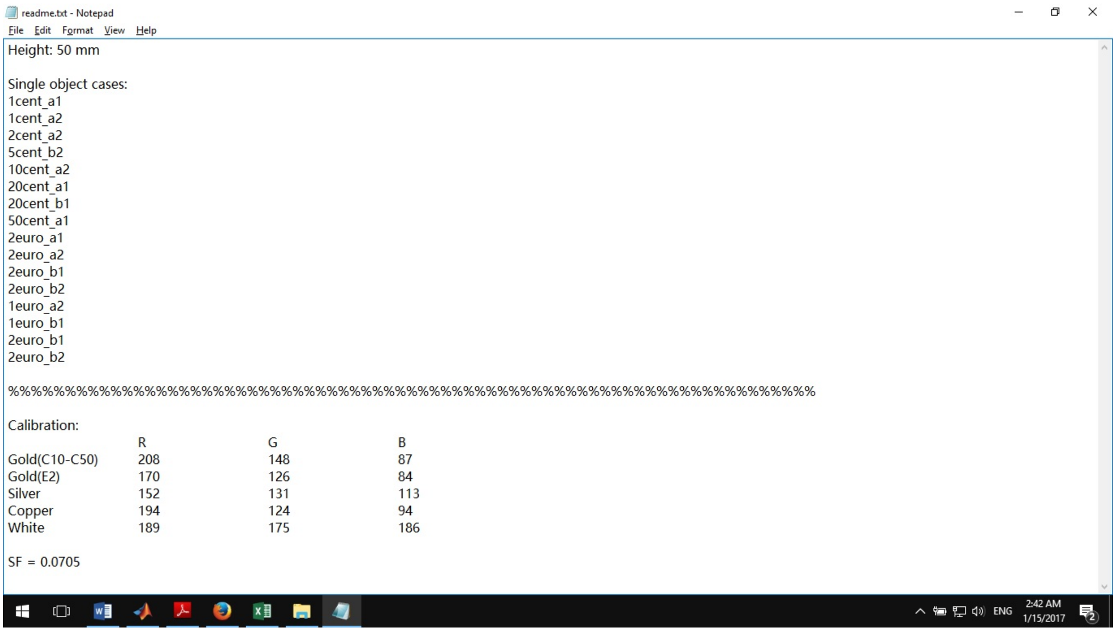

FIGURE 2.3: calibration

	Gold1 (C10-C50)			Gold2 (E2)			Silver			Copper			White			Scaling
Image	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	factor
1cent_a1										164	107	83	200	185	205	0.0720
1cent_a2										240	150	108	169	155	167	0.0712
2cent_a2										198	119	85	188	173	184	0.0702
5cent_b2										176	120	98	206	192	206	0.0725
10cent_a2	216	152	89										185	173	184	0.0705
20cent_a1	222	157	91										185	171	180	0.0706
20cent_b1	212	150	88										189	174	183	0.0706
50cent_a1	181	132	81										186	176	189	0.0703
2euro_a1				196	137	85							176	163	173	0.0694
2euro_a2				154	115	76							199	188	199	0.0696
2euro_b1				144	119	92							212	199	212	0.0709
2euro_b2				186	133	82							177	164	173	0.070937
1euro_a2							171	144	116				190	174	181	0.069611
1euro_b1							140	133	116				181	168	177	0.06982
2euro_b1							170	140	110				189	174	180	0.0702
2euro_b2							128	109	112				185	174	181	0.0694
mean	208	148	87	170	126	84	152	131	113	194	124	94	189	175	186	0.0705

FIGURE 2.4: log

2.2 Preprocessing

For the part of preprocessing, we used a Gaussian filter and an image sharpening to reduce the noise. Here is the result of preprocessing:

FIGURE 2.5: Preprocessing

2.3 Segmentation

For the part of segmentation, we do:

- detect the entire coin
- dilate the image
- fill interior gaps
- remove connected objects on border
- smoothen the coin

Here is the result of segmentation:

FIGURE 2.6: Segmentation

2.4 Recognition

For the part of recognition, we do:

- define static data (like standard size of coins)
- get the centroids with method regionprops
- calculate the contour diameter with method regionprops
- get the median of coins color by calculating the median of the coin
- carry out a decision tree to recognize coins (considering its size and its color)

Here is the result of recognition:

FIGURE 2.7: Recognition

2.5 Graphic Interface

We used guide to create our graphic interface:

Here is the running state of our application

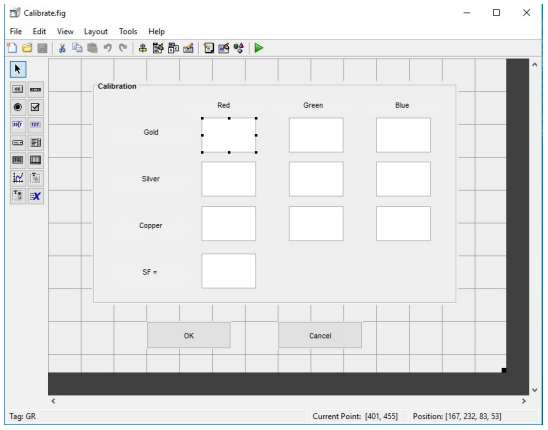

FIGURE 2.8: graphic interface 1

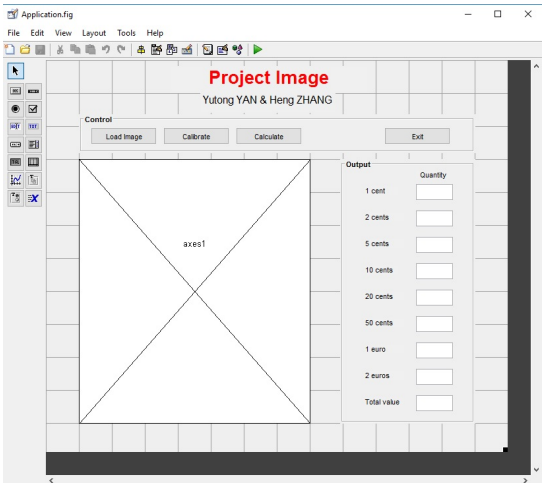

FIGURE 2.9: graphic interface 2

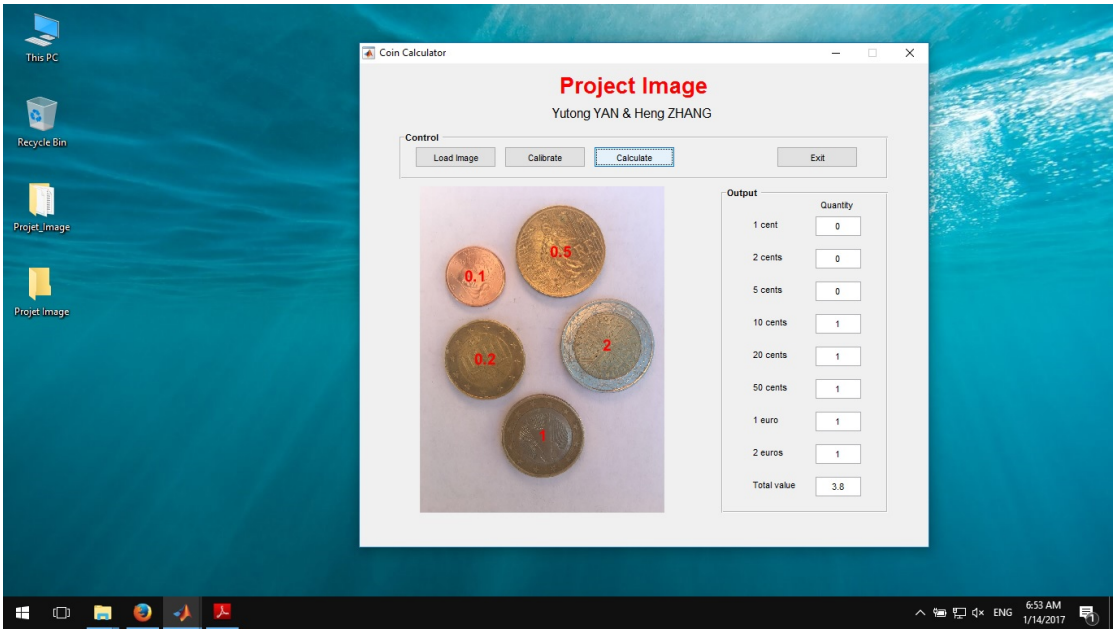

FIGURE 2.10: graphic interface 3

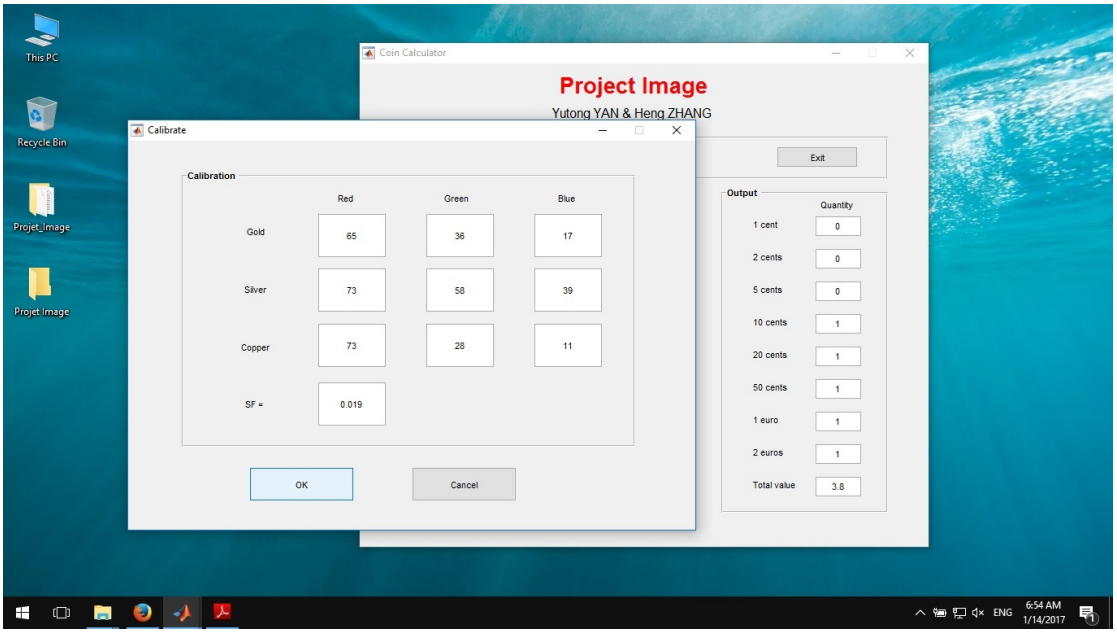

FIGURE 2.11: graphic interface 4

Chapter 3

Project Analysis

3.1 Qualitative

3.1.1 contour diameter

Here is a table to record all the detected contour diameters and standard contour diameters from the photo "4_1".

FIGURE 3.1: contour diameter

From the table we can see that the contour diameter is very close to the standard size. So our segmentation is very effective.

coins	standard	segmentation result	Accuracy
10 cents	19.75	20.55	4.05%
20 cents	22.25	22.53	1.26%
2 euros	25.75	25.63	0.47%
2 cents	18.75	18.76	0.05%

FIGURE 3.2: RGB value

FIGURE 3.3: color recognition (1 means Gold, 2 means Silver and 3 means Copper)

3.1.2 color recognition

From the pictures we can see that our algorithm thinks that 2 euros is silver. It proves that we can always distinguish copper from gold and silver, but sometimes we cant tell between gold and silver. So we should note it and pay attention when setting up the decision tree.

3.2 Quantitative

one coin			*several coins*		
Images	*Success*	*Fail*	*Images*	*Success*	*Fail*
1*cent_a*1	*X*		2_1	*X*	
1*cent_a*2	*X*		2_2	*X*	
1*cent_b*1	*X*		2_3	*X*	
1*cent_b*2	*X*		2_5	*X*	
2*cent_a*1	*X*		2_6	*X*	
2*cent_a*2	*X*		2_7	*X*	
2*cent_b*1	*X*		2_8	*X*	
2*cent_b*2	*X*		2_9	*X*	
5*cent_a*1	*X*		3_1	*X*	
5*cent_a*2	*X*		3_2	*X*	
5*cent_b*1	*X*		3_3	*X*	
5*cent_b*2	*X*		3_4	*X*	
10*cent_a*1	*X*		3_5	*X*	
10*cent_a*2	*X*		3_6	*X*	
10*cent_b*1	*X*		3_7	*X*	
10*cent_b*2	*X*		3_8	*X*	
20*cent_a*1	*X*		3_9	*X*	
20*cent_a*2	*X*		4_1	*X*	
20*cent_b*1	*X*		4_2	*X*	
20*cent_b*2	*X*				
50*cent_a*1	*X*				
50*cent_a*2	*X*				
50*cent_b*1	*X*				
50*cent_b*2	*X*				
1*euro_a*1		*X*			
1*euro_a*2	*X*				
1*euro_b*1		*X*			
1*euro_b*2	*X*				
2*euro_a*1	*X*				
2*euro_a*2	*X*				
2*euro_b*1	*X*				
2*euro_b*2	*X*				
Accuracy	96.23%				

FIGURE 3.4: "1euro_a1" and "1euro_b2"

We tested all of our photos as well as the dataset "Boisard_May_A1", and here is the result table:

Dataset name	*Number of success*	*Number of fail*	*Accuracy*
Ourdataset	51	2	96.23%
*Boisard_May_A*1	39	15	72.22%

Error analysis

- According to the table, we have two errors for the photo "1euro_a1" and "1euro_b2". After we see these two photos, we think its a problem of dataset, because the color of silver is very close to gold.
- From the table we can also see that our application isn't always effective for all of the dataset.

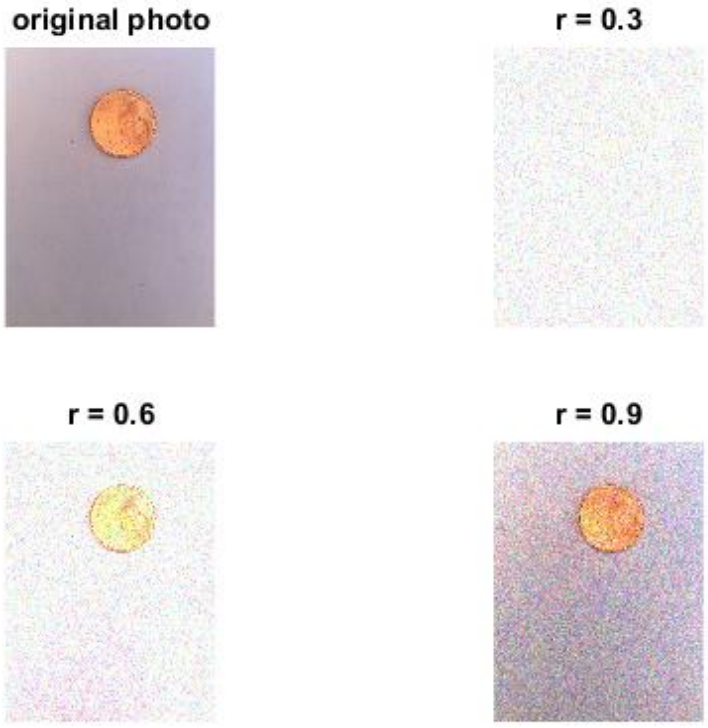

FIGURE 3.5: photo with Gaussian white noise

3.3 Noise Study

We add a Gaussian white noise to our photos and we look at its influence on our recognition (See the code in file "NoiseStudy.m").

After testing all the photos, we have the result:

r	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Accuracy	0	0	0	0	31.25%	78.13%	93.75%	84.38%	71.88%	18.75%

Conclusion: From the plot, we can see that:

- If r is too big, then the noise is relatively strong, so its difficult to recognize the coin.
- If r is too small, then the photo is too bright and difficult to recognize the coin too.

Noise Study										
Images	r									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1cent_a1	×	×	×	×	√	√	√	√	√	×
1cent_a2	×	×	×	×	×	√	√	√	×	×
1cent_b1	×	×	×	×	×	×	×	×	√	×
1cent_b2	×	×	×	×	×	×	√	√	√	×
2cent_a1	×	×	×	×	√	√	√	√	√	×
2cent_a2	×	×	×	×	×	×	√	√	√	×
2cent_b1	×	×	×	×	√	√	√	√	√	×
2cent_b2	×	×	×	×	×	×	√	√	√	×
5cent_a1	×	×	×	×	×	×	√	√	√	×
5cent_a2	×	×	×	×	×	√	√	√	×	×
5cent_b1	×	×	×	×	×	×	×	×	×	×
5cent_b2	×	×	×	×	×	×	√	√	√	×
10cent_a1	×	×	×	×	√	√	√	√	√	×
10cent_a2	×	×	×	×	√	√	√	√	√	×
10cent_b1	×	×	×	×	×	√	√	√	√	×
10cent_b2	×	×	×	×	√	√	√	√	√	√
20cent_a1	×	×	×	×	×	√	√	√	×	×
20cent_a2	×	×	×	×	√	√	√	√	√	√
20cent_b1	×	×	×	×	√	√	√	√	√	×
20cent_b2	×	×	×	×	×	√	√	√	√	×
50cent_a1	×	×	×	×	×	√	√	√	×	×
50cent_a2	×	×	×	×	×	√	√	×	×	×
50cent_b1	×	×	×	×	×	√	√	√	√	×
50cent_b2	×	×	×	×	√	√	√	√	√	√
1euro_a1	×	×	×	×	√	√	√	×	×	×
1euro_a2	×	×	×	×	×	√	√	×	×	×
1euro_b1	×	×	×	×	×	√	√	√	×	×
1euro_b2	×	×	×	×	×	√	√	√	√	×
2euro_a1	×	×	×	×	×	√	√	√	√	√
2euro_a2	×	×	×	×	×	√	√	√	√	√
2euro_b1	×	×	×	×	×	√	√	√	√	×
2euro_b2	×	×	×	×	×	√	√	√	√	√
Accuracy	0.00%	0.00%	0.00%	0.00%	31.25%	78.13%	93.75%	84.38%	71.88%	18.75%

FIGURE 3.6: log of noise study

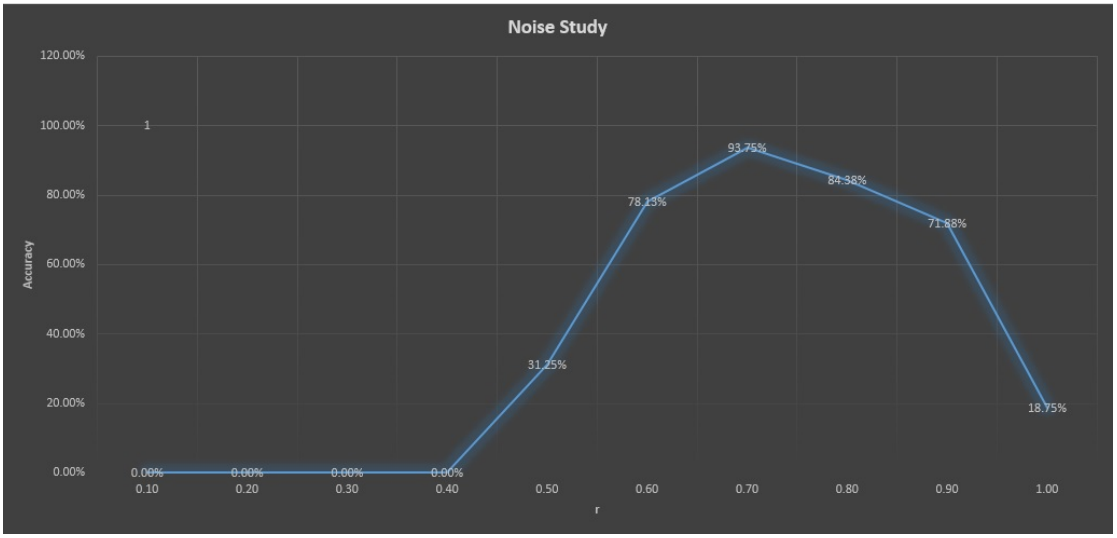

FIGURE 3.7: plot of noise study

Chapter 4

Code explanation

To launch the application, you should run the file Application.m which with call following files:

- Calibrate.m: calibration window
- preprocessing.m: image preprocessing function
- segmentation_3.m: image segmentation function
- recognition_2.m: image recognition function

For more detailed explanation, please check the commentary of the MATLAB code.

Chapter 5

Reference

- MATLAB Help center
- Detecting a Cell Using Image Segmentation - MathWorks
- Color difference - Wikipedia
- Decision tree - Wikipedia