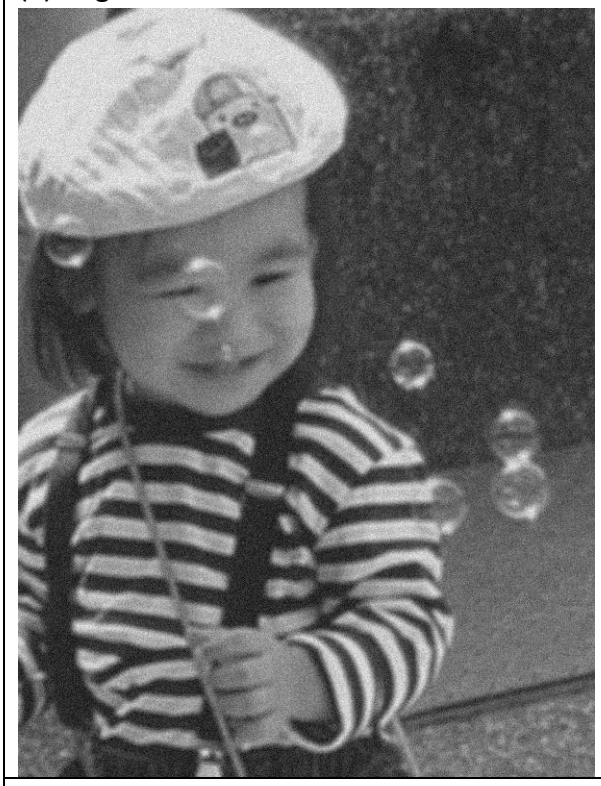
Project1

311512064　鄧書桓

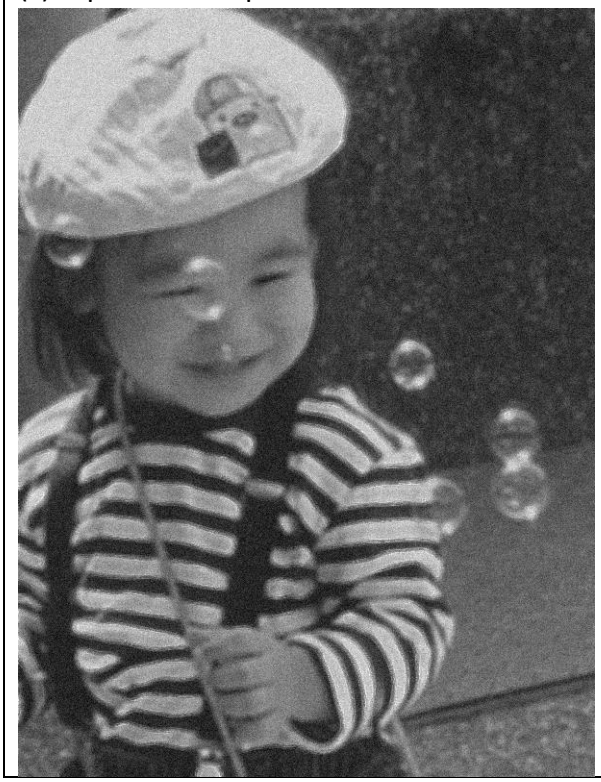
1. Kid blurred-noisy.tiff

(a) original



(b) Laplacian

(c) Laplacian-sharpened

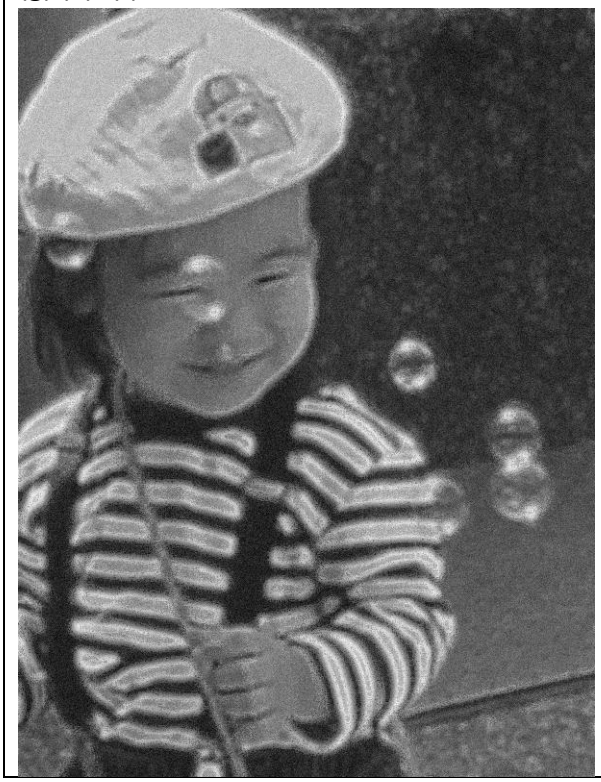


(d) Sobel-gradient

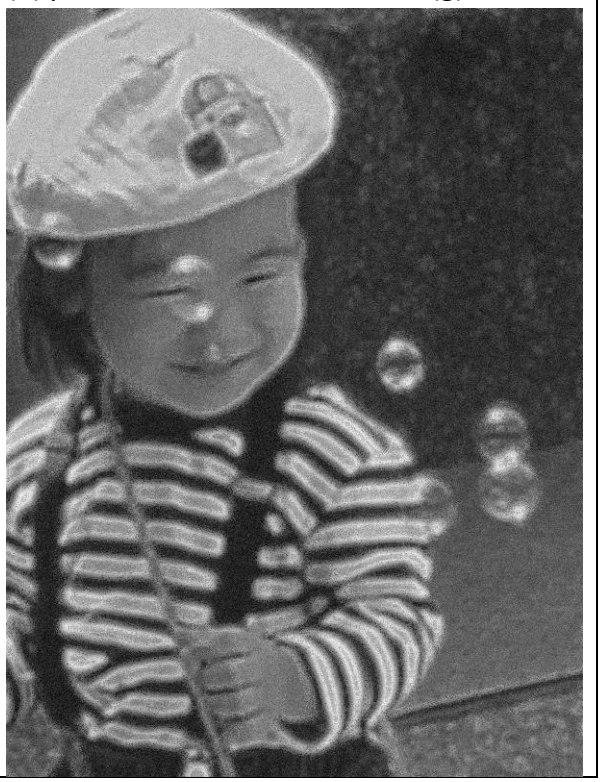
(e) smoothed gradient

(f) (e)・(b)

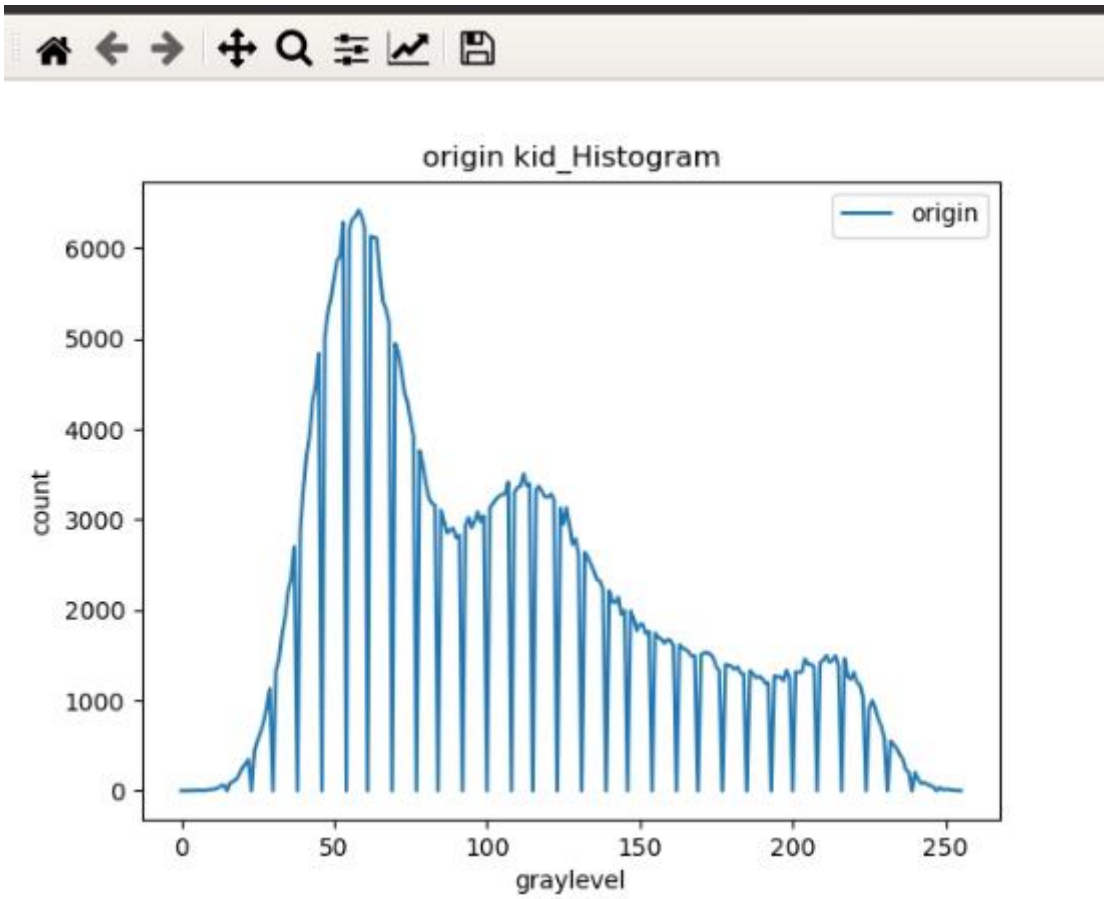
(g) (a)+(f)



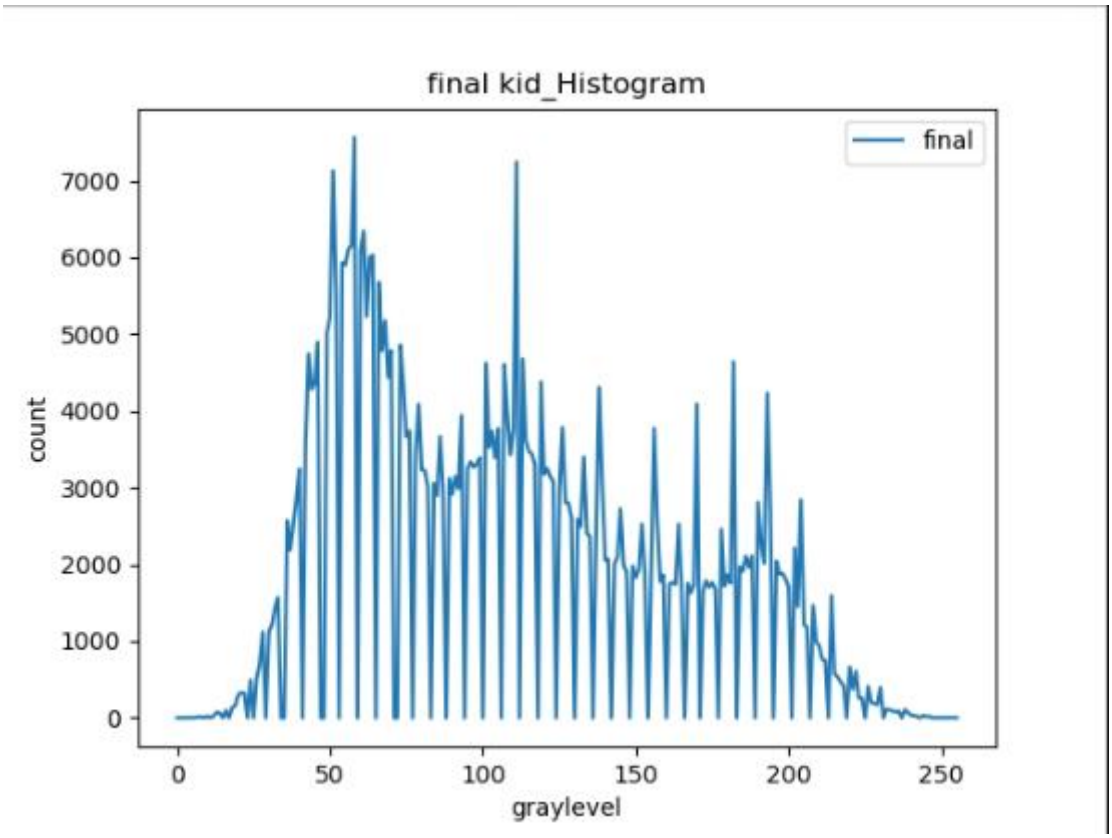
(h) power-law transformation of (g)



Original histograms



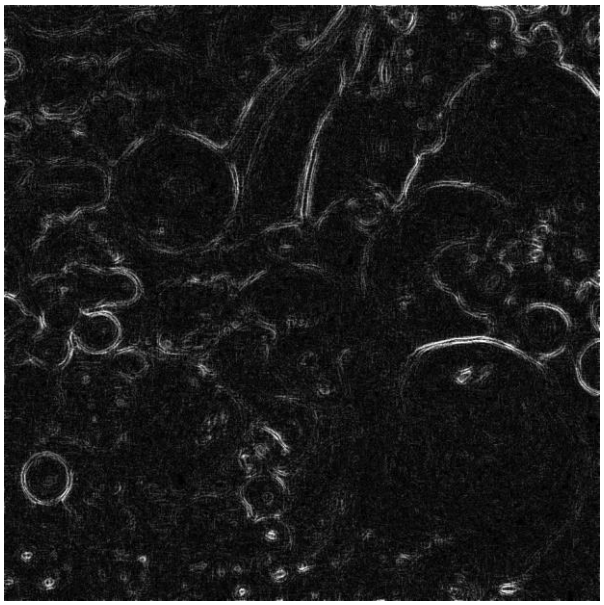
Output histograms



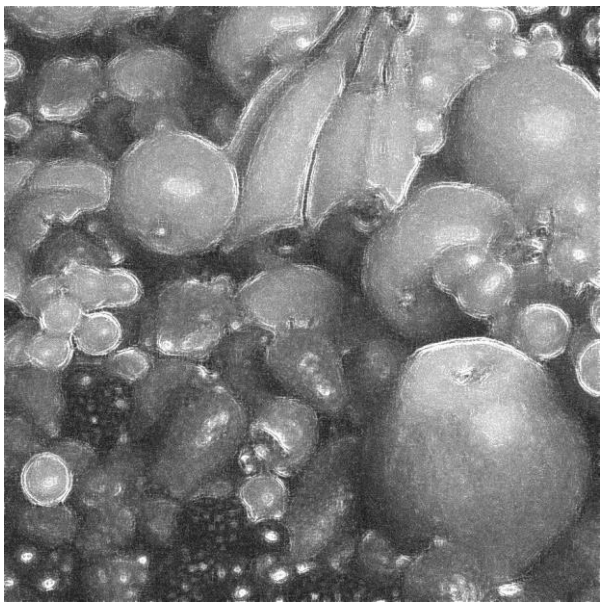
2. Fruit blurred-noisy.tiff

(a) original

(b) Laplacian



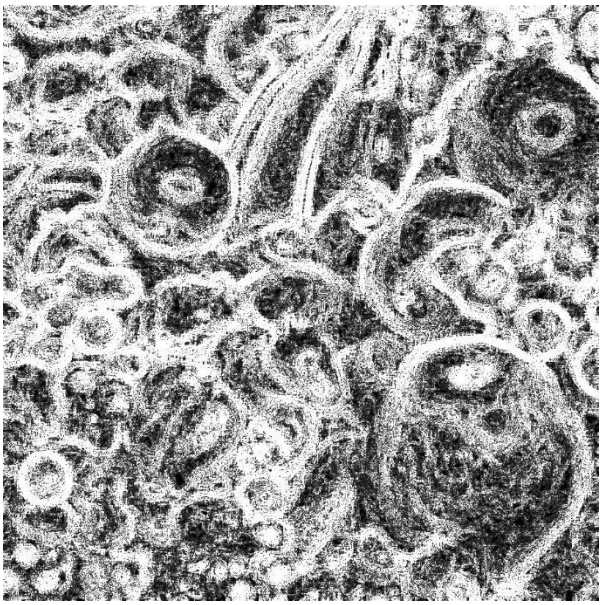
(c) Laplacian-sharpened



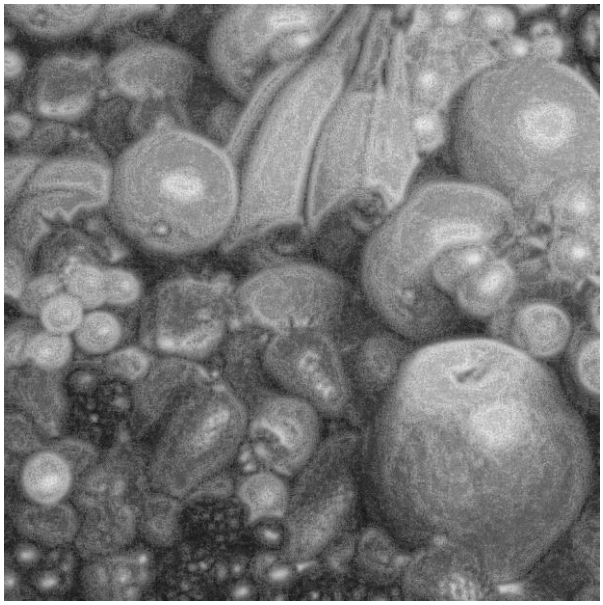
(d) Sobel-gradient

(e) smoothed gradient

(f) (e) • (b)

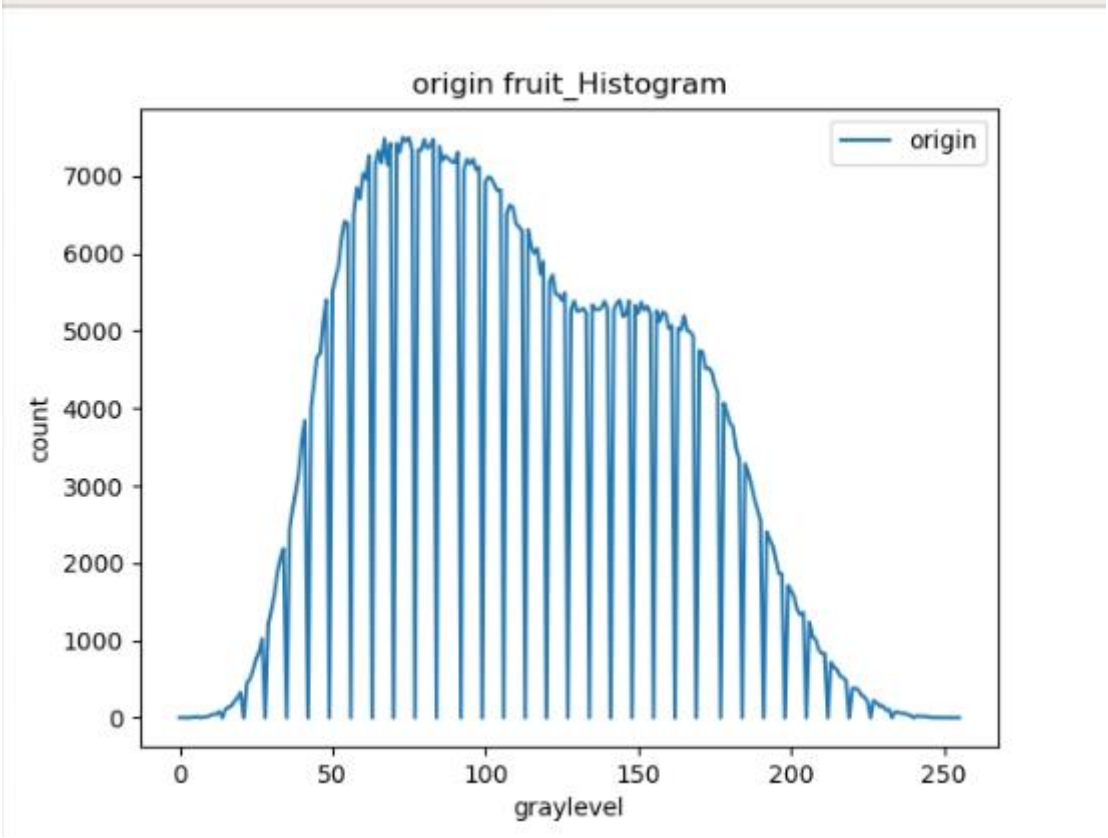


(g) (a)+(f)

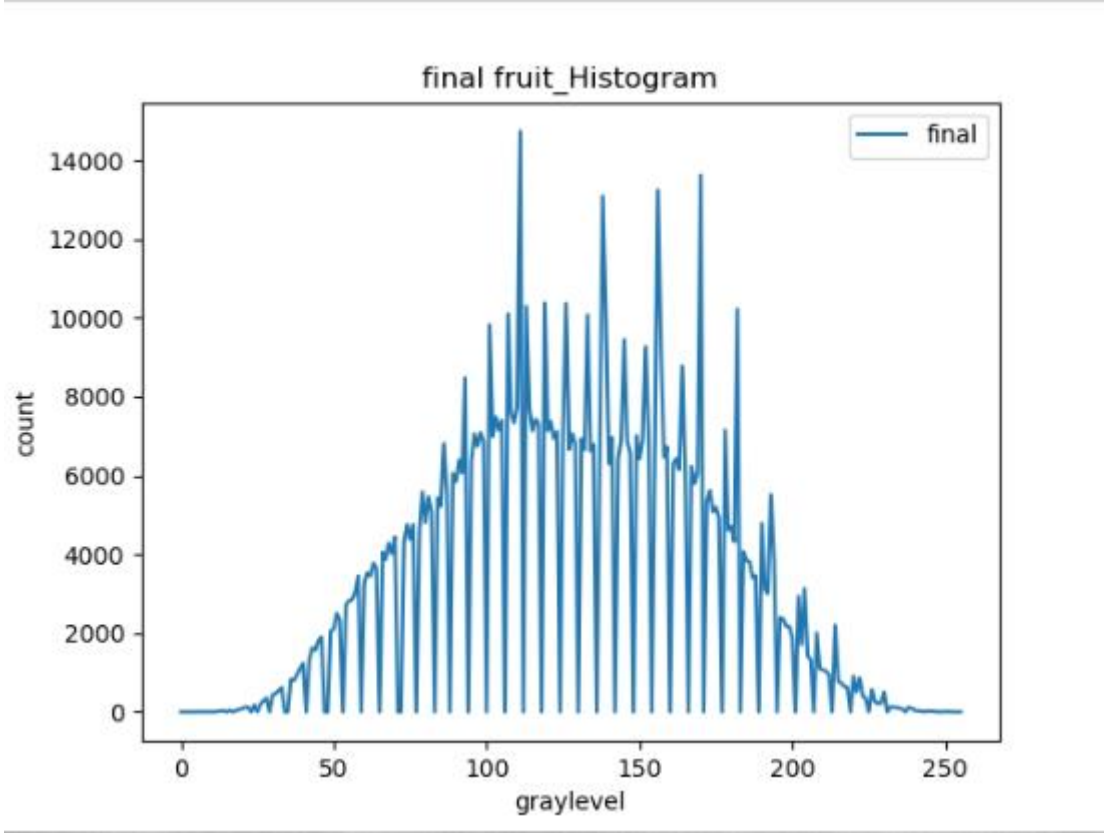


(h) power-law transformation of (g)

Original histograms



Output histograms



討論:

由(b)圖可知，邊緣已經由 Laplacian 擷取出來，並經過與原圖的相加，可得到(c)圖，並能從觀察得知原圖的邊緣已經過強化(較明顯)，其中在做完 Laplacian 需要經過 cv2.converScaleabs()函式將所有的直變成正的(經過 Laplacian 的 kernel 後會將部分的值變成負的)。另外，由(d)圖可知，垂直與水平的邊緣經過 sobel 擷取出來，並經過 smoothing box 將雜訊降低(e 圖)，接著再與(b)圖相乘得到想要的 mask，用來強化邊緣與降低雜訊。最終，將原圖與 mask 進行 high-boost filtering(權重不同)，可得到(f)圖，並藉由 power law 將圖片變亮以便觀察。由最後的圖可知，邊緣經過明顯的強化，且由於有用 smoothing box，(f)圖較原圖模糊。

r	p(r) (kid original)	p(r) (kid output)	p(r) (fruit original)	p(r) (fruit output)
0	1	1	2	1
1	0	0	3	0
2	0	0	1	0
3	1	0	2	2
4	1	5	6	1
5	4	0	8	0
6	6	4	12	2
7	0	10	0	1
8	5	15	12	3
9	13	0	11	0
10	14	22	30	5
11	19	0	45	0
12	32	29	46	11
13	54	73	81	27
14	65	60	0	26

15	0	0	112	0
16	85	98	138	42
17	107	0	156	0
18	124	134	218	38
19	179	160	265	61
20	252	301	328	81
21	295	331	0	128
22	349	322	452	120
23	0	0	500	0
24	444	492	610	175
25	557	0	758	0
26	652	514	850	192
27	759	707	1021	282
28	922	1116	0	347
29	1130	0	1208	0
30	0	1122	1387	435
31	1320	1213	1594	463
32	1458	1437	1879	544
33	1731	1566	2061	617
34	1926	0	2186	0
35	2205	0	0	0
36	2354	2570	2437	831
37	2700	2189	2703	802
38	0	2510	2923	953
39	2886	2855	3180	1101
40	3344	3241	3600	1229
41	3706	0	3843	0
42	3933	3561	0	1317
43	4310	4744	4002	1626
44	4478	4282	4326	1570
45	4838	4362	4664	1809
46	0	4896	4711	1899
47	5013	0	5134	0
48	5295	0	5399	0
49	5447	5015	0	2055
50	5670	5217	5510	2107
51	5867	7125	5692	2502
52	5910	5533	5840	2343
53	6286	0	6200	0
54	0	5934	6416	2717
55	6205	5903	6391	2817

56	6322	6108	0	2851
57	6351	6157	6501	3022
58	6422	7567	6850	3446
59	6359	0	6705	0
60	6225	6104	7037	3256
61	0	6345	6965	3532
62	6133	5230	7267	3451
63	6123	6002	0	3775
64	6112	6028	7151	3646
65	5709	0	7330	0
66	5411	5676	7170	4056
67	5325	4785	7484	3867
68	5176	5173	7147	4284
69	0	4439	7424	4021
70	4948	4789	0	4440
71	4844	0	7423	0
72	4663	0	7313	0
73	4414	4858	7503	4317
74	4296	4328	7450	4752
75	4102	3666	7496	4380
76	3914	3736	7336	4751
77	0	0	0	0
78	3761	3440	7337	4593
79	3608	4086	7338	5584
80	3411	3232	7474	4803
81	3242	3225	7359	5453
82	3182	3005	7391	5051
83	3159	0	7473	0
84	0	3053	0	5433
85	3104	2896	7386	5220
86	2980	3666	7205	6809
87	2851	2951	7261	5525
88	2885	0	7224	0
89	2906	3119	7184	6049
90	2798	2914	7176	5845
91	2828	3147	7306	6389
92	0	2989	0	6064
93	2935	3940	7084	8479
94	3022	0	7219	0
95	2912	3247	7160	6358
96	2982	3336	7218	7058

97	3086	3269	7079	6742
98	2976	3310	7121	7082
99	3041	3389	0	6856
100	0	0	6930	0
101	3134	4622	6992	9822
102	3178	3525	6971	6993
103	3224	3737	6885	7488
104	3262	3389	6812	7151
105	3278	3769	6823	7394
106	3285	0	0	0
107	3415	4606	6511	10103
108	0	3946	6627	7586
109	3299	3430	6599	7338
110	3355	3768	6387	7740
111	3371	7247	6347	14729
112	3506	0	6298	0
113	3375	4675	0	10287
114	3398	3618	6311	7685
115	0	3475	6097	7136
116	3323	3444	6004	7424
117	3369	3303	6065	7321
118	3310	0	5732	0
119	3252	4378	5895	10376
120	3248	3176	0	7141
121	3284	3249	5639	7368
122	3214	3144	5725	6945
123	0	3087	5473	7111
124	3126	0	5468	0
125	2945	2936	5396	7084
126	3134	3784	5495	10366
127	2940	2805	0	6661
128	2726	2801	5258	7058
129	2786	2606	5387	6828
130	2617	0	5252	0
131	0	2588	5260	6926
132	2636	2493	5294	6658
133	2583	3399	5232	10075
134	2509	2406	0	6621
135	2430	2362	5339	6788
136	2336	0	5281	0
137	2321	2291	5274	6630

138	2219	4307	5300	13082
139	0	3059	5384	9978
140	2211	2052	5282	6299
141	2092	2067	0	6962
142	2081	0	5275	0
143	2143	2009	5358	6415
144	1951	2088	5394	6907
145	1996	2723	5189	9446
146	0	1990	5216	6879
147	1989	1898	5389	6558
148	1905	0	0	0
149	1773	1978	5332	7006
150	1843	1831	5227	6421
151	1840	1941	5371	6863
152	1739	2525	5270	9260
153	1764	1877	5319	6880
154	0	0	5212	0
155	1745	1852	0	6507
156	1698	3774	5260	13238
157	1679	2603	5124	9223
158	1632	1777	5245	6470
159	1671	1860	5217	6723
160	1664	0	5033	0
161	1604	1741	5068	6296
162	0	1765	0	6429
163	1617	1738	5032	6145
164	1576	2524	5014	8776
165	1565	1699	5196	5820
166	1531	0	5005	0
167	1490	1750	4991	6232
168	1496	1631	4931	5796
169	0	1734	0	6084
170	1506	4089	4743	13620
171	1525	0	4729	0
172	1530	1656	4514	5321
173	1511	1788	4524	5623
174	1475	1705	4475	5078
175	1372	1762	4307	5196
176	1333	1677	4205	4933
177	0	0	0	0
178	1397	2460	4065	7150

179	1388	1719	3947	4617
180	1376	1866	3808	4703
181	1347	1754	3757	4326
182	1369	4640	3479	10216
183	1298	0	3352	0
184	1291	1969	0	4070
185	0	1909	3280	3840
186	1329	2107	3161	3807
187	1297	1961	3004	3404
188	1252	2109	2826	3457
189	1264	0	2694	0
190	1243	2813	2551	4783
191	1188	2271	0	3117
192	1199	2016	2407	3000
193	0	4232	2307	5512
194	1271	2749	2221	3839
195	1258	0	2046	0
196	1261	2041	1862	2400
197	1218	1873	1865	2352
198	1334	1883	0	2175
199	1247	1811	1712	2167
200	0	1704	1642	1887
201	1319	0	1562	0
202	1310	2213	1399	2937
203	1315	1450	1332	1709
204	1459	2841	1364	3131
205	1401	1214	0	1425
206	1399	1182	1238	1328
207	1366	0	1048	0
208	0	1463	1013	2008
209	1418	985	885	1096
210	1445	939	831	1068
211	1497	760	831	1034
212	1420	747	0	940
213	1441	0	718	0
214	1494	1593	658	2193
215	1382	574	621	774
216	0	533	541	736
217	1465	470	518	646
218	1268	414	481	618
219	1234	0	0	0

220	1313	663	383	906
221	1201	375	383	511
222	1173	606	360	863
223	1038	257	296	405
224	0	256	266	340
225	908	0	214	0
226	996	409	0	566
227	912	200	222	257
228	791	190	193	216
229	713	173	149	225
230	583	394	156	503
231	0	0	101	0
232	551	112	108	131
233	515	104	0	121
234	469	87	76	116
235	403	79	73	92
236	351	86	57	79
237	232	0	61	0
238	209	106	42	116
239	0	70	33	96
240	205	34	0	39
241	131	29	23	27
242	80	20	22	24
243	92	0	14	0
244	75	31	21	27
245	55	17	7	11
246	45	19	8	20
247	0	3	0	5
248	36	2	5	3
249	14	0	2	0
250	13	4	2	9
251	17	0	1	3
252	7	3	4	1
253	5	2	1	1
254	0	0	0	0
255	2	1	1	1

3.code

```python
import cv2
import numpy as np
from PIL import Image
import os
import matplotlib.pyplot as plt
import openpyxl

if not (os.path.exists("./images")):
    os.makedirs('images')

kid_img, fruit_img = cv2.imread(
    "kid blurred-noisy.tif"), cv2.imread("fruit blurred-noisy.tif")

# (a)origin
cv2.imshow('(a)kid_original', kid_img)
kid_img_norm = cv2.normalize(kid_img, None, alpha=0,
                             beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
kid_img_norm = (255*kid_img_norm).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(kid_img_norm, cv2.COLOR_BGR2RGB))
image.save('./images/(a)kid_original.jpg', dpi=(200.0, 200.0))
# blur
blur = cv2.medianBlur(kid_img, 15)
# blur = cv2.GaussianBlur(kid_img, (9, 9), 0)

# (b)Laplician
Laplician = cv2.Laplacian(blur, cv2.CV_64F, ksize=3)
# take absolute value
abs_Laplacian = cv2.convertScaleAbs(Laplician)
cv2.imshow('(b)kid_Laplacian', abs_Laplacian)
norm_img = cv2.normalize(abs_Laplacian, None, alpha=0,
                         beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
norm_img = (255*norm_img).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(norm_img, cv2.COLOR_BGR2RGB))
image.save('./images/(b)kid_Laplacian.jpg', dpi=(200.0, 200.0))

# (c)Laplician_sharpened
Laplacian_sharpened = cv2.add(
    kid_img, abs_Laplacian)
cv2.imshow('(c)kid_Laplacian_sharpened', Laplacian_sharpened)
norm_img = cv2.normalize(Laplacian_sharpened, None, alpha=0,
                         beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
norm_img = (255*norm_img).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(norm_img, cv2.COLOR_BGR2RGB))
image.save('./images/(c)kid_Laplacian_sharpened.jpg', dpi=(200.0, 200.0))

# (d)sobel
sobelx = cv2.Sobel(blur, cv2.CV_64F, 1, 0, ksize=3)
# take absolute value
sobelx = cv2.convertScaleAbs(sobelx)
sobely = cv2.Sobel(blur, cv2.CV_64F, 0, 1, ksize=3)
# take absolute value
sobely = cv2.convertScaleAbs(sobely)
Sobel_gradient = cv2.add(sobelx, sobely)
cv2.imshow('(d)kid_Sobel_gradient', Sobel_gradient)
norm_img = cv2.normalize(Sobel_gradient, None, alpha=0,
                         beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
norm_img = (255*norm_img).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(norm_img, cv2.COLOR_BGR2RGB))
image.save('./images/(d)kid_Sobel_gradient.jpg', dpi=(200.0, 200.0))

# (e)smoothing_box_filter
smoothing_box_filter = cv2.boxFilter(
    Sobel_gradient, -1, (5, 5), normalize=True)
cv2.imshow('(e)kid_smoothing_box_filter)', smoothing_box_filter)
norm_img = cv2.normalize(smoothing_box_filter, None, alpha=0,
                         beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
```

```python
norm_img = (255*norm_img).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(norm_img, cv2.COLOR_BGR2RGB))
image.save('./images/(e)kid_smoothing_box_filter.jpg', dpi=(200.0, 200.0))

f_img = cv2.multiply(smoothing_box_filter, abs_Laplacian)
cv2.imshow('(f)kid_)', f_img)
norm_img = cv2.normalize(f_img, None, alpha=0,
                         beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
norm_img = (255*norm_img).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(norm_img, cv2.COLOR_BGR2RGB))
image.save('./images/(f)kid.jpg', dpi=(200.0, 200.0))

# (g)high_boost filtering
c = 0.15
g_img = cv2.addWeighted(f_img, c, kid_img, 1-c, 0)
cv2.imshow('(g)kid_)', g_img)
norm_img = cv2.normalize(g_img, None, alpha=0,
                         beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
norm_img = (255*norm_img).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(norm_img, cv2.COLOR_BGR2RGB))
image.save('./images/(g)kid.jpg', dpi=(200.0, 200.0))

# (h)Power_law
h_img = np.array(255*(g_img/255)**0.9, dtype='uint8')
cv2.imshow('(h)kid_', h_img)
final_norm_img = cv2.normalize(h_img, None, alpha=0,
                               beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
final_norm_img = (255*final_norm_img).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(norm_img, cv2.COLOR_BGR2RGB))
image.save('./images/(h)kid.jpg', dpi=(200.0, 200.0))

origin_hist = cv2.calcHist([kid_img_norm], [0], None, [256], [0, 256])
final_hist = cv2.calcHist([final_norm_img.astype('uint8')], [
```

```python
                           0], None, [256], [0, 256])
plt.plot(origin_hist)
plt.plot(final_hist)
plt.title('kid_Histogram')
plt.xlabel('graylevel')
plt.ylabel('count')
plt.legend(['origin', 'final'])
plt.show()

wb = openpyxl.load_workbook("Histograms.xlsx", data_only=True)
sheet = wb["histograms"]
for i in range(256):
    sheet.cell(i+2, 2).value = int(origin_hist[i])
    sheet.cell(i+2, 3).value = int(final_hist[i])
wb.save("Histograms.xlsx")
# cv2.waitKey()

#########################################################################################################################

# (a)origin
cv2.imshow('(a)fruit_original', fruit_img)
fruit_img_norm = cv2.normalize(fruit_img, None, alpha=0,
                               beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
fruit_img_norm = (255*fruit_img_norm).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(fruit_img_norm, cv2.COLOR_BGR2RGB))
image.save('./images/(a)fruit_original.jpg', dpi=(200.0, 200.0))
# blur
blur = cv2.medianBlur(fruit_img, 15)
# blur = cv2.GaussianBlur(fruit_img, (23, 23), 0)

# (b)Laplician
Laplician = cv2.Laplacian(blur, cv2.CV_64F, ksize=5)
```

```python
#  take absolute value
abs_Laplacian = cv2.convertScaleAbs(Laplician)
cv2.imshow('(b)fruit_Laplacian', abs_Laplacian)
norm_img = cv2.normalize(abs_Laplacian, None, alpha=0,
                         beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
norm_img = (255*norm_img).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(norm_img, cv2.COLOR_BGR2RGB))
image.save('./images/(b)fruit_Laplacian.jpg', dpi=(200.0, 200.0))

# (c)Laplician_sharpened
Laplacian_sharpened = cv2.add(
    fruit_img, abs_Laplacian)
cv2.imshow('(c)fruit_Laplacian_sharpened', Laplacian_sharpened)
norm_img = cv2.normalize(Laplacian_sharpened, None, alpha=0,
                         beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
norm_img = (255*norm_img).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(norm_img, cv2.COLOR_BGR2RGB))
image.save('./images/(c)fruit_Laplacian_sharpened.jpg', dpi=(200.0, 200.0))

# (d)sobel
sobelx = cv2.Sobel(blur, cv2.CV_64F, 1, 0, ksize=3)
# take absolute value
sobelx = cv2.convertScaleAbs(sobelx)
sobely = cv2.Sobel(blur, cv2.CV_64F, 0, 1, ksize=3)
# take absolute value
sobely = cv2.convertScaleAbs(sobely)
Sobel_gradient = cv2.add(sobelx, sobely)
cv2.imshow('(d)fruit_Sobel_gradient', Sobel_gradient)
norm_img = cv2.normalize(Sobel_gradient, None, alpha=0,
                         beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
norm_img = (255*norm_img).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(norm_img, cv2.COLOR_BGR2RGB))
image.save('./images/(d)fruit_Sobel_gradient.jpg', dpi=(200.0, 200.0))
```

```python

# (e)smoothing_box_filter
smoothing_box_filter = cv2.boxFilter(
    Sobel_gradient, -1, (5, 5), normalize=True)
cv2.imshow('(e)fruit_smoothing_box_filter)', smoothing_box_filter)
norm_img = cv2.normalize(smoothing_box_filter, None, alpha=0,
                         beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
norm_img = (255*norm_img).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(norm_img, cv2.COLOR_BGR2RGB))
image.save('./images/(e)fruit_smoothing_box_filter.jpg', dpi=(200.0, 200.0))

f_img = cv2.multiply(smoothing_box_filter, abs_Laplacian)
cv2.imshow('(f)fruit_)', f_img)
norm_img = cv2.normalize(f_img, None, alpha=0,
                         beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
norm_img = (255*norm_img).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(norm_img, cv2.COLOR_BGR2RGB))
image.save('./images/(f)fruit.jpg', dpi=(200.0, 200.0))

# (g)high_boost filtering
c = 0.15
g_img = cv2.addWeighted(f_img, c, fruit_img, 1-c, 0)
cv2.imshow('(g)fruit_)', g_img)
norm_img = cv2.normalize(g_img, None, alpha=0,
                         beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
norm_img = (255*norm_img).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(norm_img, cv2.COLOR_BGR2RGB))
image.save('./images/(g)fruit.jpg', dpi=(200.0, 200.0))

# (h)Power_law
h_img = np.array(255*(g_img/255)**0.9, dtype='uint8')
cv2.imshow('(h)fruit_', h_img)
final_norm_img = cv2.normalize(h_img, None, alpha=0,
```

```python
                               beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
final_norm_img = (255*final_norm_img).astype(np.uint8)
image = Image.fromarray(cv2.cvtColor(norm_img, cv2.COLOR_BGR2RGB))
image.save('./images/(h)fruit.jpg', dpi=(200.0, 200.0))

origin_hist = cv2.calcHist([fruit_img_norm], [0], None, [256], [0, 256])
final_hist = cv2.calcHist([final_norm_img.astype('uint8')], [
                          0], None, [256], [0, 256])
plt.plot(origin_hist)
plt.plot(final_hist)
plt.title('kid_Histogram')
plt.xlabel('graylevel')
plt.ylabel('count')
plt.legend(['origin', 'final'])
plt.show()

wb = openpyxl.load_workbook("Histograms.xlsx", data_only=True)
sheet = wb["histograms"]
for i in range(256):
    sheet.cell(i+2, 4).value = int(origin_hist[i])
    sheet.cell(i+2, 5).value = int(final_hist[i])
wb.save("Histograms.xlsx")

cv2.waitKey()

```