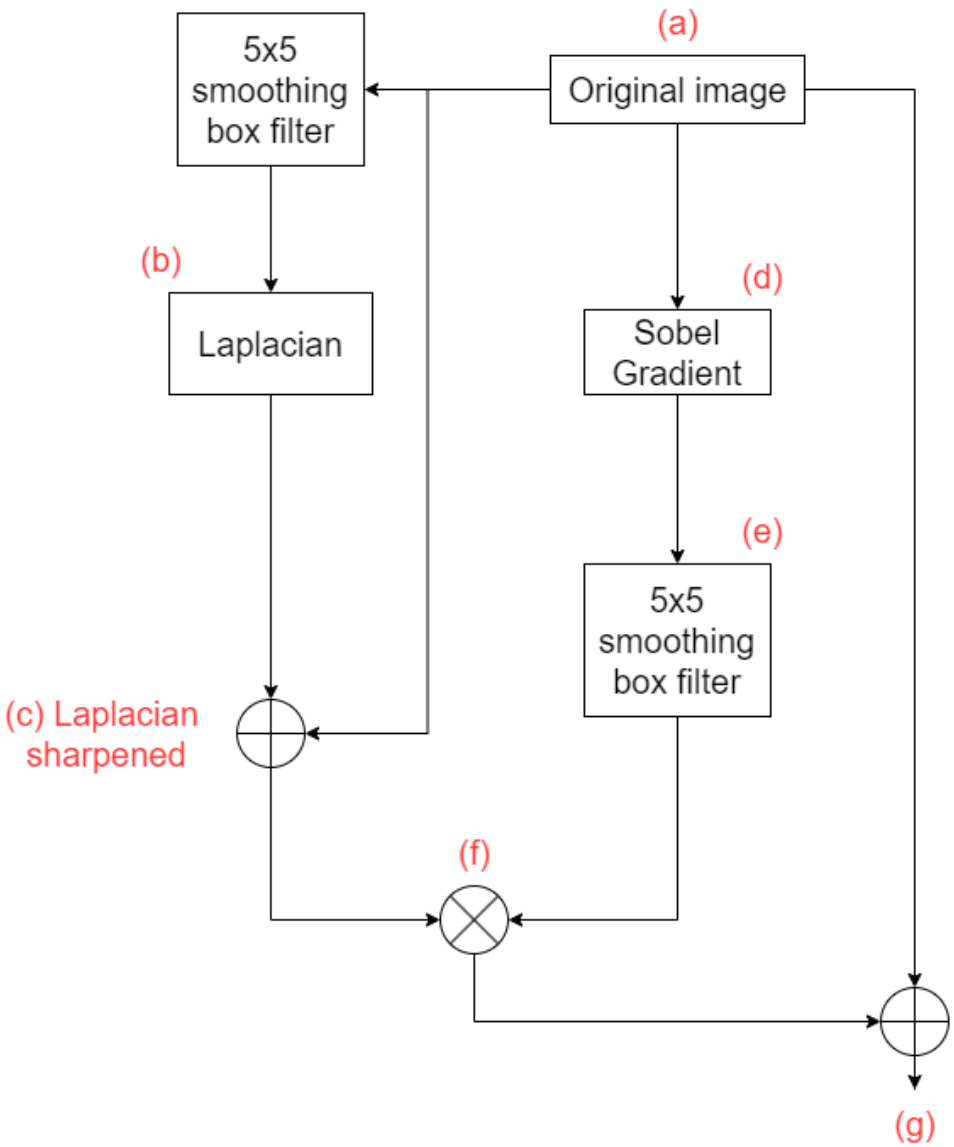
DIP Project 1

彭晨益 學號 310512054

Introduction:

In this project we would implement lots of filtering techniques and show the results.
The images we used are "*kid blurred-noisy.tif* " and *"fruit blurred-noisy.tif"*

Flow chart:

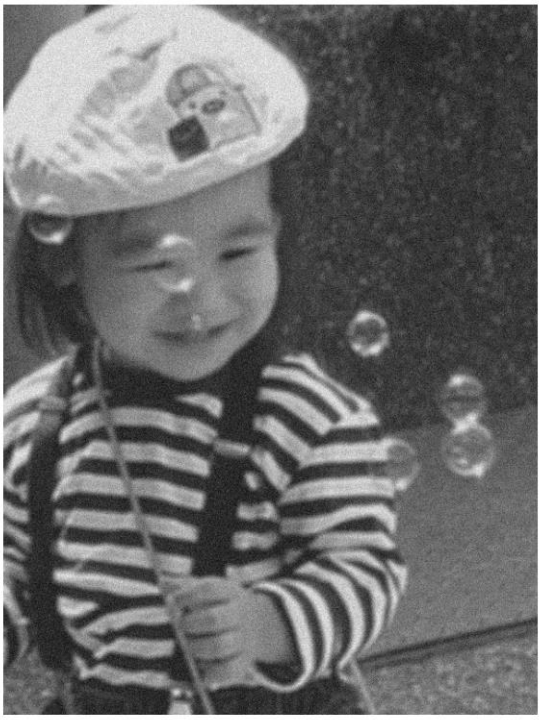


Here I use a 5x5 smoothing box filter before doing Laplacian filter due to the noisy original image, and I get the better result!

Results:

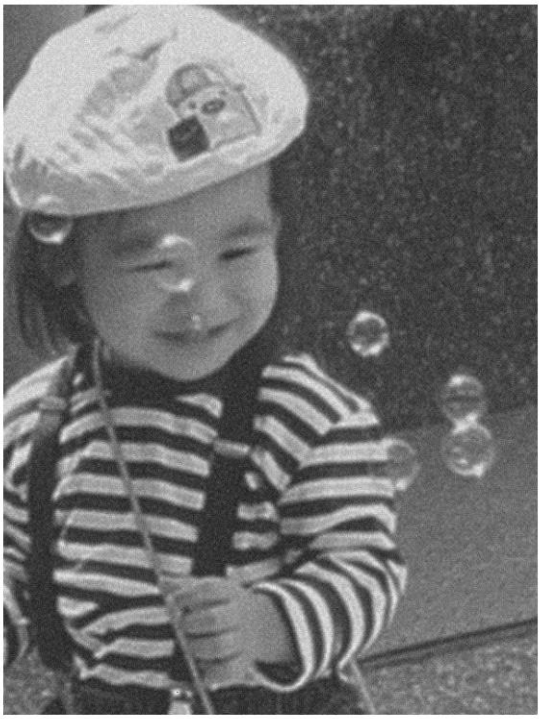
For kid blurred-noisy.tif:

(a) Original-image



(b) Laplacian

(c) Laplacian-sharpened



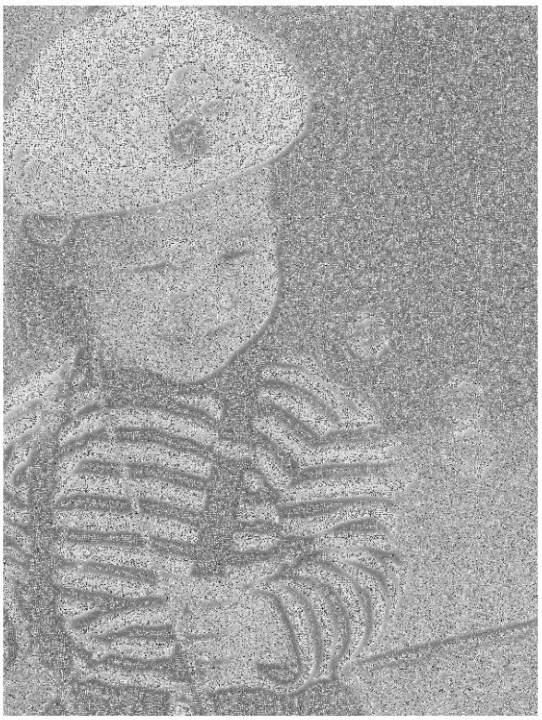
(d) Sobel-gradient

(e) smoothed-gradient

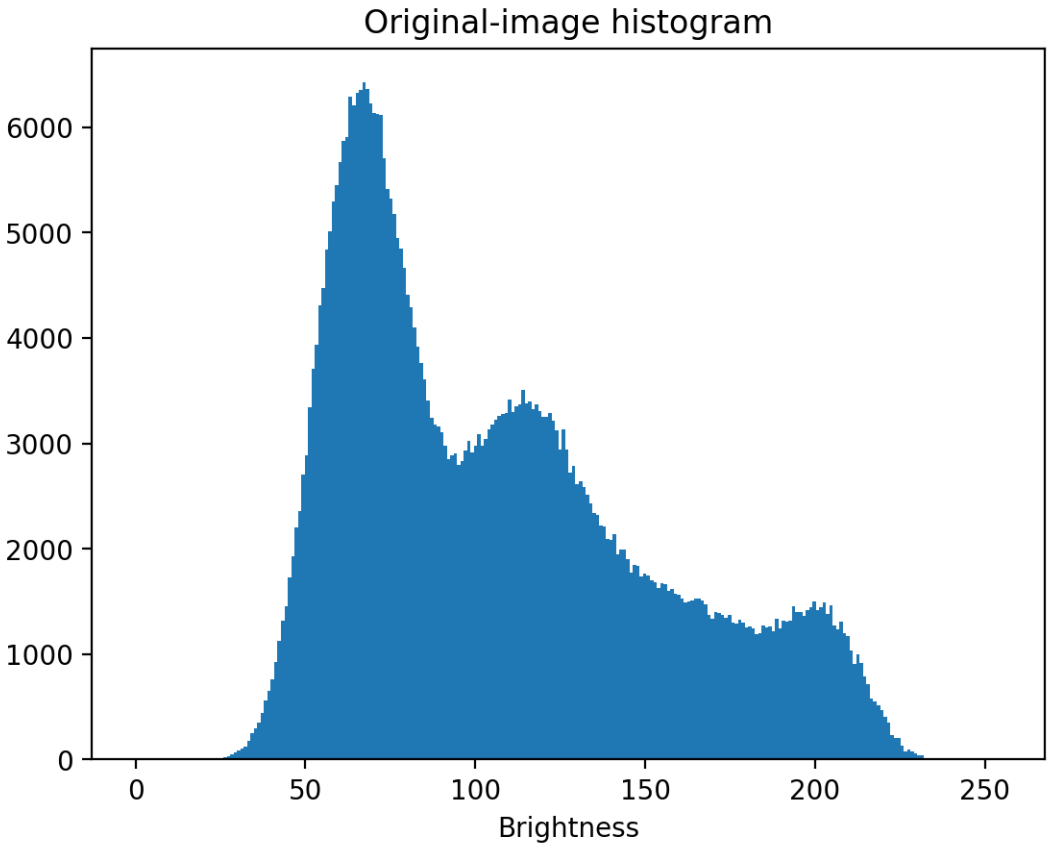
(f) extracted feature

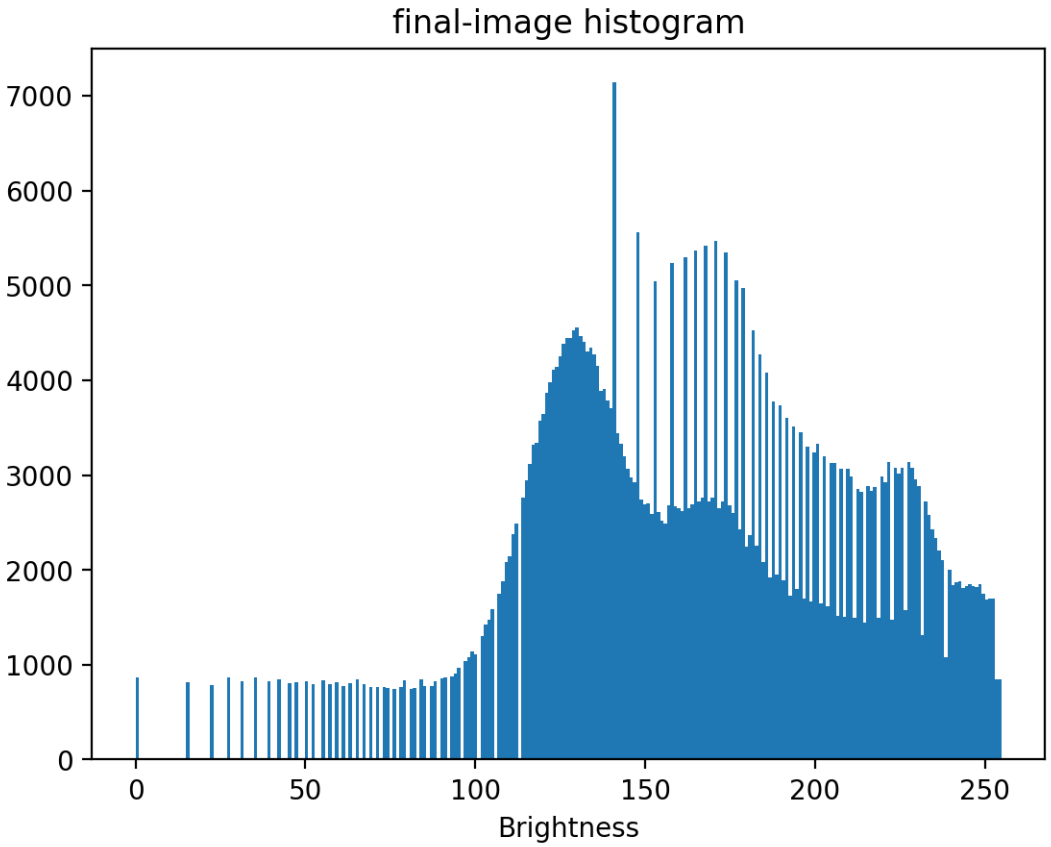
(g)

(h) final-image



Histogram of Original image and Final result:

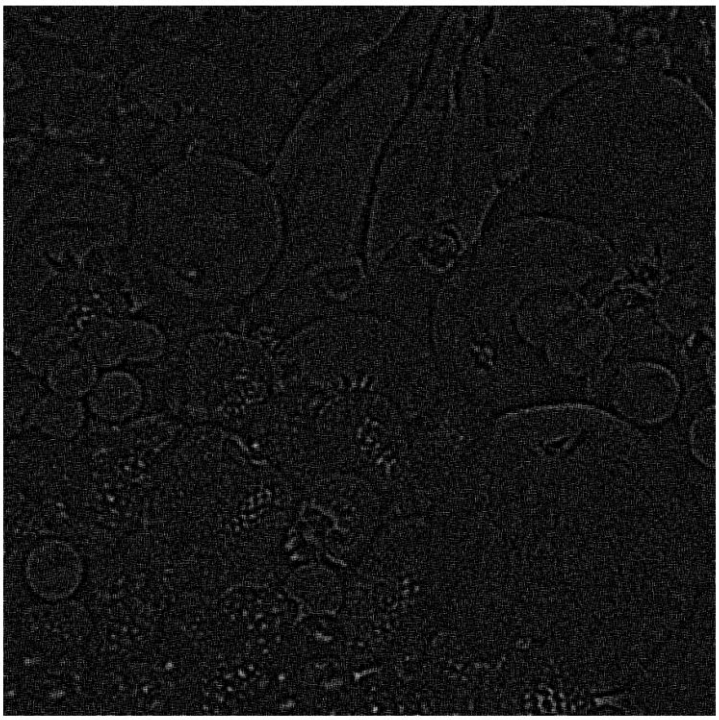




For fruit blurred-noisy.tif:

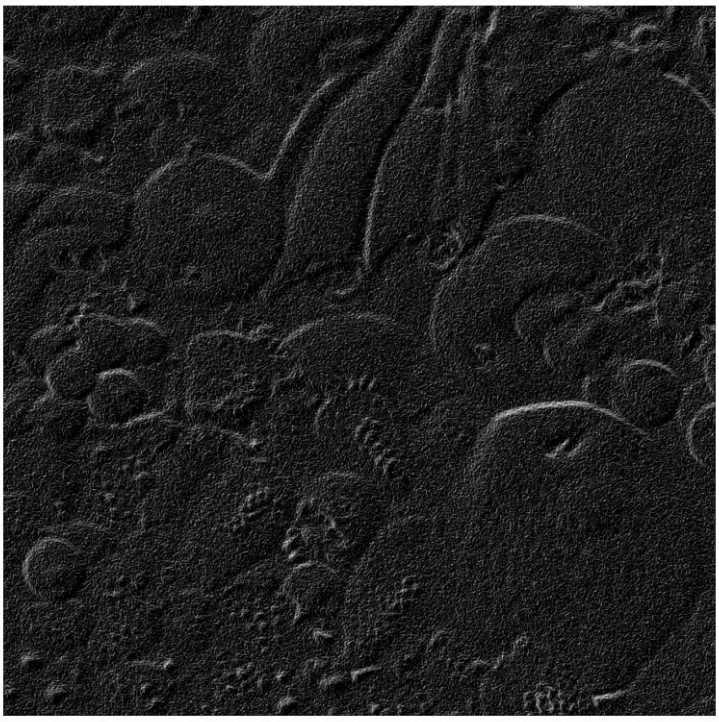
(a) Original-image

(b) Laplacian

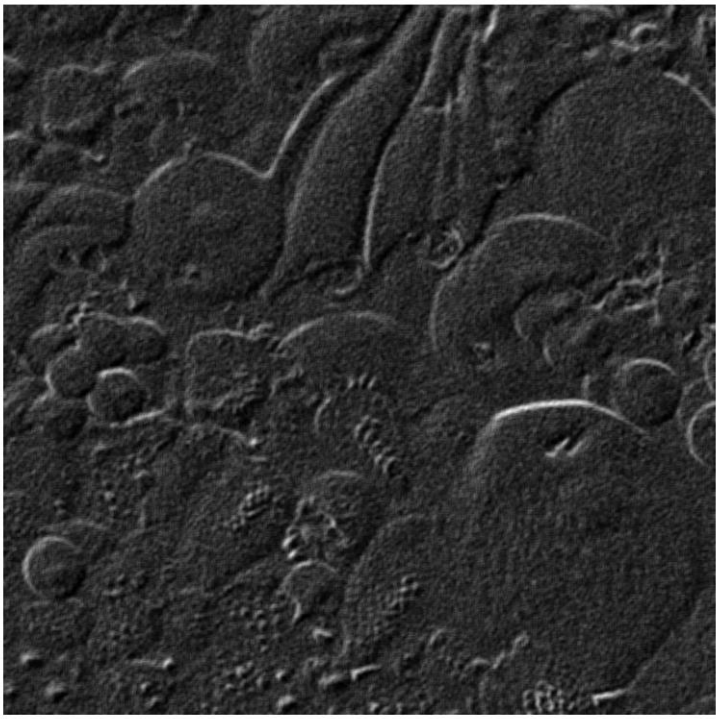


(c) Laplacian-sharpened

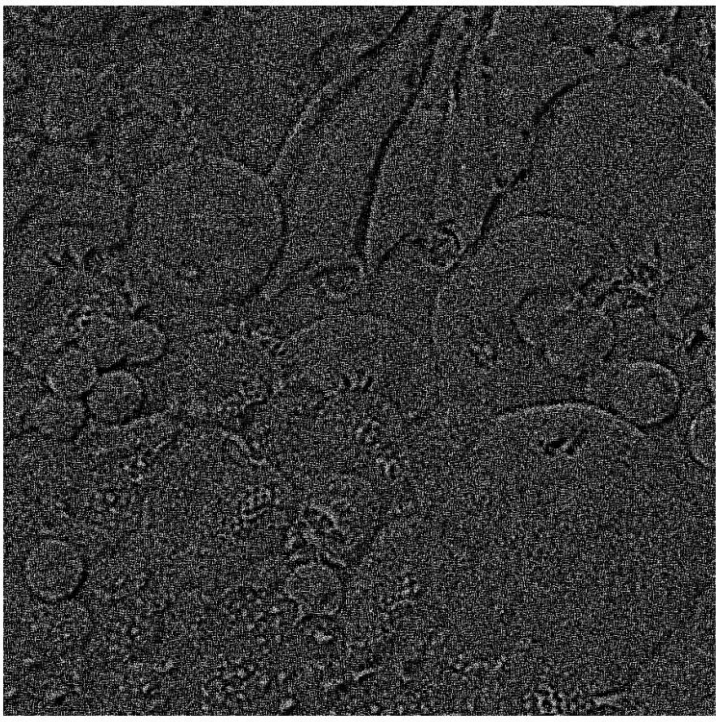
(d) Sobel-gradient



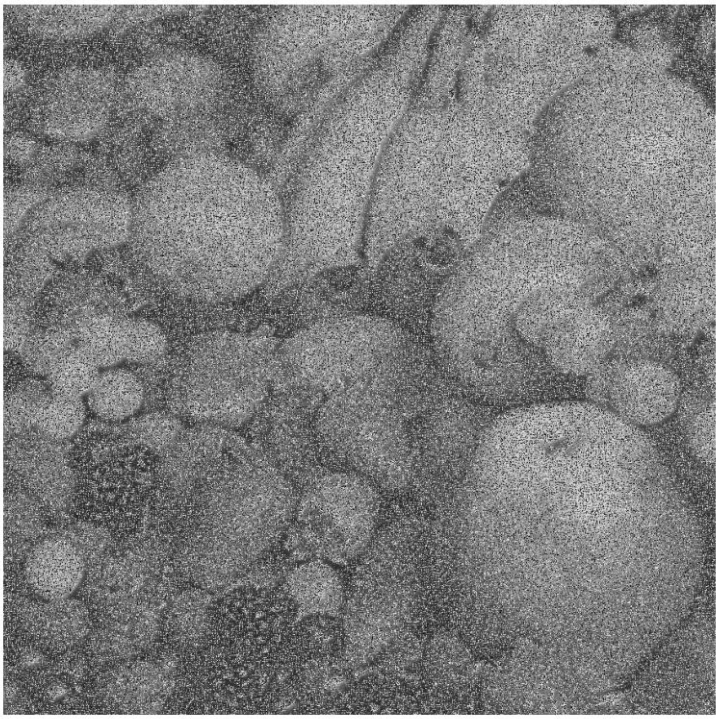
(e) smoothed-gradient



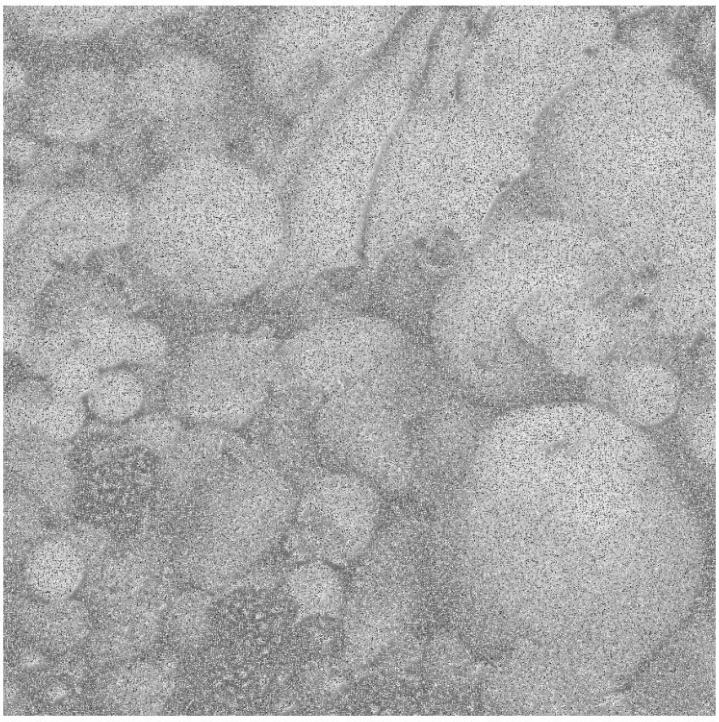
(f) extracted feature



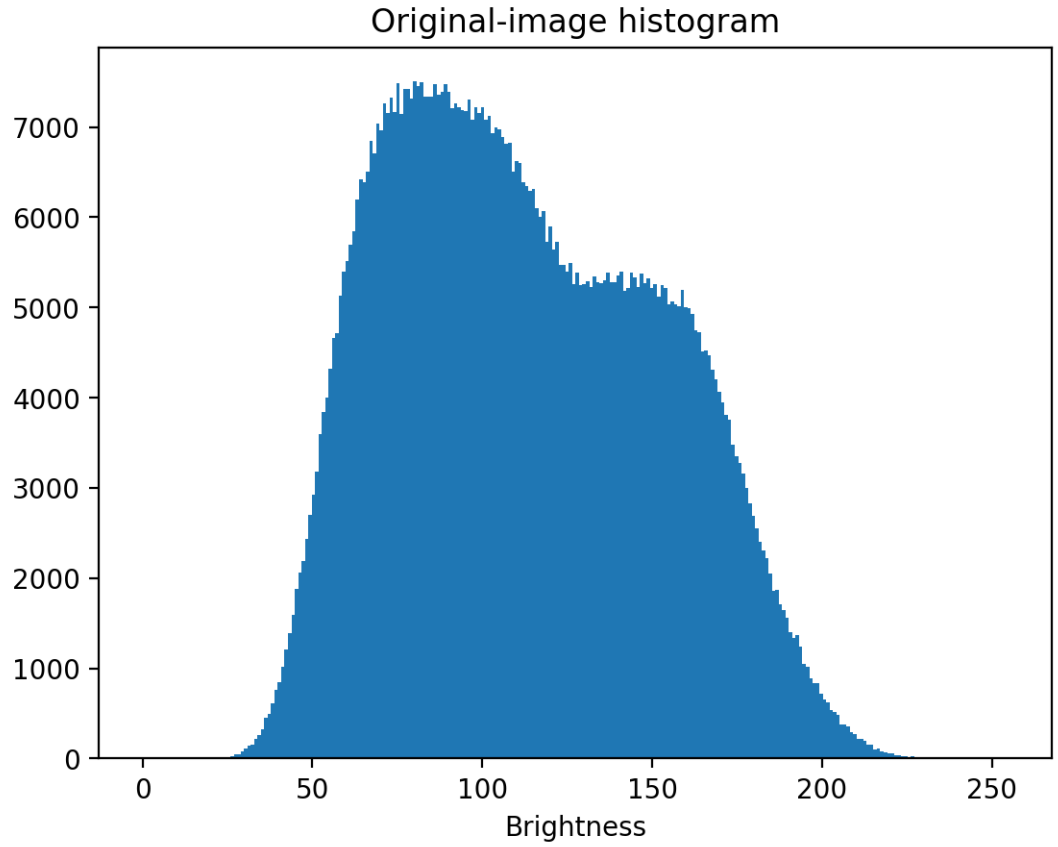
(g)

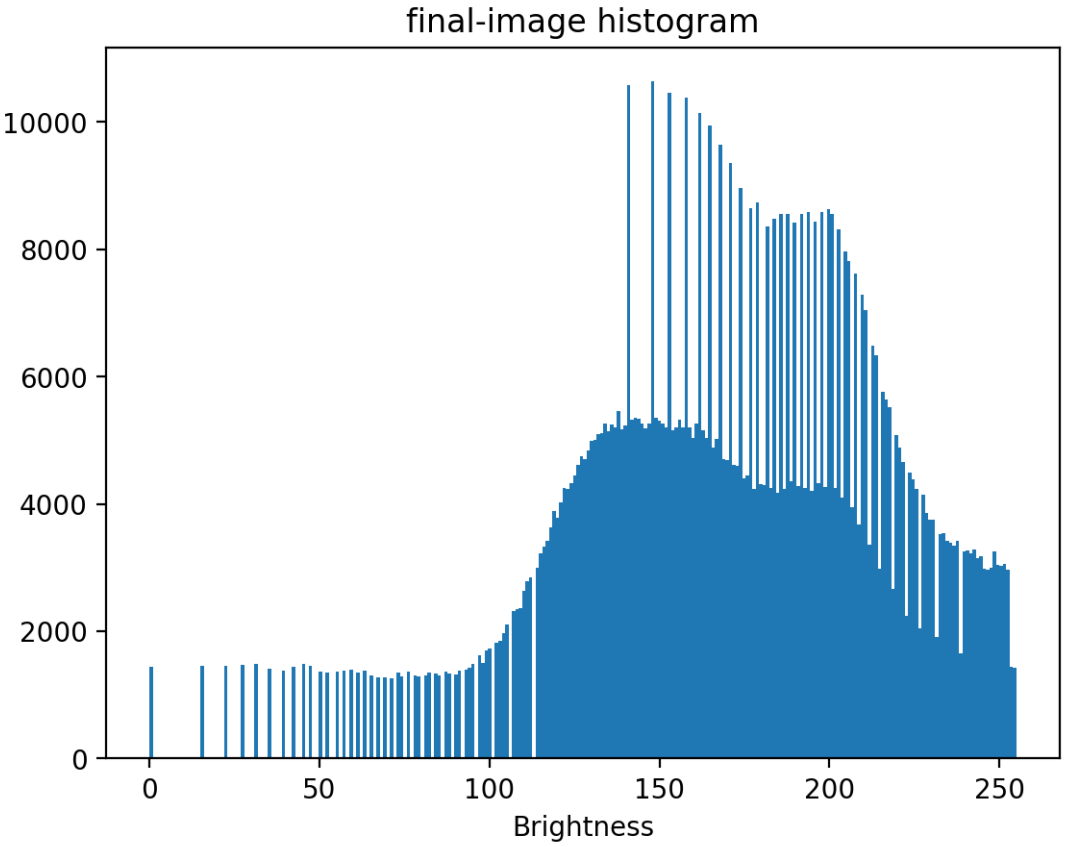


(h) final-image



Histogram of Original image and Final result:





Histograms

r	p(r) (kid original)	p(r) (kid output)	p(r) (fruit original)	p(r) (fruit output)
0	0	864	0	1443
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0
12	0	0	0	0
13	0	0	0	0
14	0	0	0	0
15	0	814	0	1460
16	1	0	0	0
17	0	0	2	0
18	0	0	3	0
19	1	0	1	0
20	1	0	2	0
21	4	0	6	0
22	6	787	8	1459
23	5	0	12	0
24	13	0	12	0
25	14	0	11	0
26	19	0	30	0
27	32	869	45	1469
28	54	0	46	0
29	65	0	81	0
30	85	0	112	0
31	107	830	138	1486
32	124	0	156	0
33	179	0	218	0
34	252	0	265	0
35	295	866	328	1404
36	349	0	452	0

37	444	0	500	0
38	557	0	610	0
39	652	824	758	1374
40	759	0	850	0
41	922	0	1021	0
42	1130	845	1208	1441
43	1320	0	1387	0
44	1458	0	1594	0
45	1731	805	1879	1493
46	1926	0	2061	0
47	2205	811	2186	1457
48	2354	0	2437	0
49	2700	0	2703	0
50	2886	826	2923	1371
51	3344	0	3180	0
52	3706	791	3600	1356
53	3933	0	3843	0
54	4310	0	4002	0
55	4478	834	4326	1363
56	4838	0	4664	0
57	5013	800	4711	1385
58	5295	0	5134	0
59	5447	813	5399	1394
60	5670	0	5510	0
61	5867	771	5692	1347
62	5910	0	5840	0
63	6286	803	6200	1380
64	6205	0	6416	0
65	6322	849	6391	1310
66	6351	0	6501	0
67	6422	796	6850	1283
68	6359	0	6705	0
69	6225	769	7037	1281
70	6133	0	6965	0
71	6123	769	7267	1261
72	6112	0	7151	0
73	5709	766	7330	1351
74	5411	758	7170	1290
75	5325	0	7484	0
76	5176	741	7147	1361
77	4948	0	7424	0

78	4844	761	7423	1309
79	4663	833	7313	1292
80	4414	0	7503	0
81	4296	740	7450	1305
82	4102	759	7496	1356
83	3914	0	7336	0
84	3761	848	7337	1336
85	3608	775	7338	1299
86	3411	0	7474	0
87	3242	775	7359	1366
88	3182	821	7391	1330
89	3159	0	7473	0
90	3104	854	7386	1324
91	2980	867	7205	1388
92	2851	0	7261	0
93	2885	873	7224	1391
94	2906	904	7184	1420
95	2798	969	7176	1487
96	2828	0	7306	0
97	2935	1038	7084	1616
98	3022	1076	7219	1503
99	2912	1140	7160	1693
100	2982	1108	7218	1735
101	3086	0	7079	0
102	2976	1300	7121	1812
103	3041	1424	6930	1855
104	3134	1472	6992	1969
105	3178	1591	6971	2113
106	3224	0	6885	0
107	3262	1746	6812	2324
108	3278	1879	6823	2353
109	3285	2085	6511	2356
110	3415	2143	6627	2633
111	3299	2374	6599	2786
112	3355	2492	6387	2843
113	3371	0	6347	0
114	3506	2758	6298	2990
115	3375	2948	6311	3219
116	3398	3117	6097	3329
117	3323	3324	6004	3418
118	3369	3339	6065	3628

119	3310	3578	5732	3883
120	3252	3639	5895	3778
121	3248	3870	5639	4019
122	3284	3977	5725	4244
123	3214	4107	5473	4229
124	3126	4139	5468	4321
125	2945	4256	5396	4441
126	3134	4386	5495	4613
127	2940	4443	5258	4756
128	2726	4442	5387	4705
129	2786	4526	5252	4835
130	2617	4560	5260	4987
131	2636	4463	5294	5011
132	2583	4407	5232	5098
133	2509	4305	5339	5111
134	2430	4341	5281	5257
135	2336	4271	5274	5141
136	2321	4149	5300	5249
137	2219	3886	5384	5200
138	2211	3904	5282	5459
139	2092	3790	5275	5179
140	2081	3703	5358	5232
141	2143	7137	5394	10587
142	1951	3439	5189	5320
143	1996	3327	5216	5360
144	1989	3195	5389	5343
145	1905	3067	5332	5260
146	1773	2974	5227	5186
147	1843	2920	5371	5268
148	1840	5563	5270	10634
149	1739	2742	5319	5358
150	1764	2690	5212	5308
151	1745	2700	5260	5267
152	1698	2587	5124	5202
153	1679	5042	5245	10463
154	1632	2613	5217	5155
155	1671	2519	5033	5208
156	1664	2489	5068	5321
157	1604	2677	5032	5200
158	1617	5238	5014	10389
159	1576	2670	5196	5207

160	1565	2651	5005	5030
161	1531	2625	4991	5263
162	1490	5297	4931	10138
163	1496	2653	4743	5165
164	1506	2686	4729	5044
165	1525	5368	4514	9951
166	1530	2721	4524	4889
167	1511	2759	4475	5024
168	1475	5415	4307	9639
169	1372	2725	4205	4707
170	1333	2765	4065	4683
171	1397	5473	3947	9363
172	1388	2655	3808	4607
173	1376	2725	3757	4602
174	1347	5350	3479	8966
175	1369	2678	3352	4397
176	1298	2601	3280	4455
177	1291	5052	3161	8651
178	1329	2432	3004	4237
179	1297	4972	2826	8734
180	1252	2243	2694	4306
181	1264	2371	2551	4294
182	1243	4529	2407	8354
183	1188	2257	2307	4247
184	1199	4273	2221	8487
185	1271	2086	2046	4170
186	1258	4081	1862	8563
187	1261	1924	1865	4242
188	1218	3777	1712	8560
189	1334	1949	1642	4351
190	1247	3736	1562	8417
191	1319	1888	1399	4283
192	1310	3599	1332	8559
193	1315	1732	1364	4254
194	1459	3515	1238	8580
195	1401	1794	1048	4201
196	1399	3454	1013	8440
197	1366	1696	885	4332
198	1418	3297	831	8586
199	1445	1664	831	4264
200	1497	3236	718	8630

201	1420	3325	658	8560
202	1441	1645	621	4246
203	1494	3202	541	8312
204	1382	1620	518	4097
205	1465	3123	481	7968
206	1268	3128	383	7814
207	1234	1512	383	3946
208	1313	3065	360	7621
209	1201	1507	296	3670
210	1173	3071	266	7281
211	1038	2985	214	7049
212	908	1490	222	3359
213	996	2854	193	6494
214	912	2825	149	6338
215	791	1440	156	2977
216	713	2885	101	5759
217	583	2835	108	5643
218	551	2870	76	5516
219	515	1492	73	2664
220	469	2989	57	5083
221	403	2928	61	4880
222	351	3135	42	4659
223	232	1471	33	2240
224	209	3073	23	4488
225	205	3016	22	4384
226	131	3072	14	4232
227	80	1577	21	2053
228	92	3142	7	4149
229	75	3075	8	3857
230	55	2952	5	3760
231	45	2887	2	3754
232	36	1308	2	1916
233	14	2720	1	3521
234	13	2577	4	3535
235	17	2432	1	3417
236	7	2334	1	3390
237	5	2209	0	3346
238	2	2107	0	3419
239	0	1075	0	1656
240	0	1999	0	3249
241	0	1836	0	3266

242	0	1867	0	3222
243	0	1878	0	3278
244	0	1804	0	3144
245	0	1829	0	3180
246	0	1850	0	2986
247	0	1830	0	2972
248	0	1818	0	2996
249	0	1846	0	3253
250	0	1749	0	3039
251	0	1683	0	3029
252	0	1697	0	3051
253	0	1695	0	2968
254	0	847	0	1439
255	0	850	0	1429

Source Code:

```
import os
import sys
import cv2
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import openpyxl

def save_img(src, title, save_path):
    plt.figure()
    plt.imshow(src, cmap="gray")
    plt.title(title)

    if (os.path.exists(save_path + title + ".png")):
        return
    if is_save:
        plt.savefig(save_path + title + ".png", dpi=200, bbox_inches='tight')

def save_hist(src, title, save_path):
    plt.figure()
    plt.hist(src.ravel(), 256, [0, 255])
    plt.title(title.split()[1] + ' histogram')

    if (os.path.exists(save_path + title.split()[1] + "-histogram.png")):
        return
```

```
    if is_save:
        plt.savefig(save_path + title.split()[1] + "-histogram.png", dpi=200,
bbox_inches='tight')



def check_folder(path):
    if not os.path.exists(path):
        os.makedirs(path)
    return path

def main(img, img_save_path):

    global is_save
    img_save_path = check_folder(img_save_path)
    image = cv2.imread(img, 0)



    if image is None:
        print("Could not open or find the image")
        sys.exit()


    kernel = np.array([[-1, -1, -1],
                       [-1,  8, -1],
                       [-1, -1, -1]], dtype="float32")

    g_x = np.array([[-1,  0,  1],
                    [-2 , 0,  2],
                    [-1 , 0,  1]], dtype="float32")

    g_y = np.array([[-1, -2, -1],
                    [0 ,  0,  0],
                    [1 ,  2,  1]], dtype="float32")


    smooth = (1 / 25) * np.ones((5, 5), dtype="float32")

    #============================================================
    # Do the Laplacian with smoothing first
    smoothed_img = cv2.filter2D(image, -1, smooth)
    laplacian = abs(cv2.filter2D(smoothed_img, -1, kernel))
```

```
    # Do the Laplacian without smoothing (uncomment the following line)
    # laplacian = cv2.filter2D(image, -1, kernel)
    #=================================================================

    laplacian_sharpen = image + laplacian

    gx_filtered = cv2.filter2D(image, -1, g_x)
    gy_filtered = cv2.filter2D(image, -1, g_y)

    sobel_gradient = abs(gx_filtered + gy_filtered)

    smoothed_gradient = cv2.filter2D(sobel_gradient, -1, smooth)

    f = smoothed_gradient * laplacian
    gamma = 0.5
    s = np.array(255 * ((f + image) / 255) ** gamma, dtype = 'uint8')


    img_list = [image, laplacian, laplacian_sharpen, sobel_gradient, smoothed_gradient, f,
(f+image), s]
    img_title = ["(a) Original-image", "(b) Laplacian", "(c) Laplacian-sharpened", "(d)
Sobel-gradient", "(e) smoothed-gradient", "(f) extracted feature", "(g)", "(h) final-
image"]

    fig, ax = plt.subplots(2, 4, figsize=(16, 16))
    fig2, ax2 = plt.subplots(2, 4, figsize=(16, 16))
    for i in range(2):
        for j in range(4):
            ax[i, j].imshow(img_list[j + i*4], cmap="gray")
            ax[i, j].set_title(img_title[j + i*4])
            ax[i, j].axis("off")
            ax2[i, j].hist(img_list[j + i*4].ravel(), 256, [0, 255])
            ax2[i, j].set_title(img_title[j + i*4])


    plt.show()
    save_hist(img_list[0], img_title[0], img_save_path)
    save_hist(img_list[-1], img_title[-1], img_save_path)



    origin_img_hist = np.histogram(image.ravel(), 256, [0, 255])
```

```
    final_img_hist = np.histogram(s.ravel(), 256, [0, 255])


    if not os.path.exists('Histograms.xlsx'):
        df_ = pd.read_excel('Histograms.xlsx', index_col=0)

        if (img == image_kid):
            df_["p(r) \n(kid original)"] = origin_img_hist[0].tolist()
            df_["p(r) \n(kid output)"] = final_img_hist[0].tolist()
            writer = pd.ExcelWriter('Histograms.xlsx', engine='openpyxl')
            # Convert the dataframe to an XlsxWriter Excel object.
            df_.to_excel(writer, sheet_name='Sheet1')
            writer.save()
        else:

            df_["p(r) \n(fruit original)"] = origin_img_hist[0].tolist()
            df_["p(r) \n(fruit output)"] = final_img_hist[0].tolist()
            writer = pd.ExcelWriter('Histograms.xlsx', engine='openpyxl')
            # Convert the dataframe to an XlsxWriter Excel object.
            df_.to_excel(writer, sheet_name='Sheet1')
            writer.save()

    # save the images
    for i in range(len(img_list)):
        save_img(img_list[i], img_title[i], img_save_path)
    plt.close('all')



if __name__ == "__main__":

    image_fruit = "fruit_blurred_noisy.tif"
    image_kid = "kid_blurred_noisy.tif"
    is_save = True
    main(image_kid, "image_kid/")
    main(image_fruit, "image_fruit/")
```