一、 实验目的和要求：

任选两张同一场景的不同照片（要体现拍照角度，亮度，尺度变化），运行 SIFT 和 SURF 算法，比较两个算法的优缺点

二、 实验环境：

设备：MacBook Pro (Retina, 13-inch, Early 2015)
操作系统：macOS High Sierra 10.13.4
IDE: Clion
编程语言：C 语言
外部库：OpenCV

三、 实验内容以及实验结果：

SIFT，也即尺度不变特征变换（Scale-invariant feature transform），该算法用于在图像中检测关键点，是一种局部特征描述子。SIFT 特征基于物体上的一些局部外观兴趣点，与图片实际的大小以及旋转无关，对于光线以及噪声等的容忍度也相对较高。

利用 SIFT 算法进行图片匹配主要分为特征检测以及特征匹配两个部分。

特征检测部分，首先在图像中搜索所有尺度上的图像位置，通过高斯微分函数来识别潜在的尺度和旋转不变的兴趣点，获得尺度不变性。在候选的位置上，确定位置和尺度，根据尺度间的稳定程度进行筛选。然后基于图像局部的梯度方向，为关键点位置分配方向，从而生成变换的不变性因素，生成特征描述子。这些因素在选定的图像尺度上形成一种图像的局部描述，在形变以及光照变换下也可以保持稳定。

特征匹配部分，根据特征检测获得的对于尺度以及旋转等因素无关的特征向量间的匹配，从而获取图片间的对应关系。具体的匹配方式为，利用关键点特征向量间的欧式距离作为相似度判定标准，对于两幅图片间的特征点进行关联。

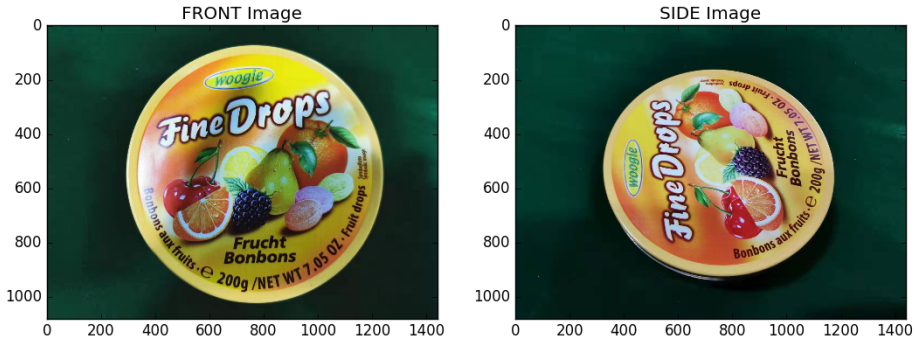
具体代码实现如下

```
1. import numpy as np
2. import cv2
3. from matplotlib import pyplot as plt
4.
5. # 读取图片
6. image1 = cv2.imread('/home/cyx/桌面
   /cv_homework/vision_sift_surf/INPUT/front.jpeg')
7. front_image = cv2.cvtColor(image1, cv2.COLOR_BGR2RGB)
8. front_gray = cv2.cvtColor(front_image, cv2.COLOR_RGB2GRAY)
9.
```

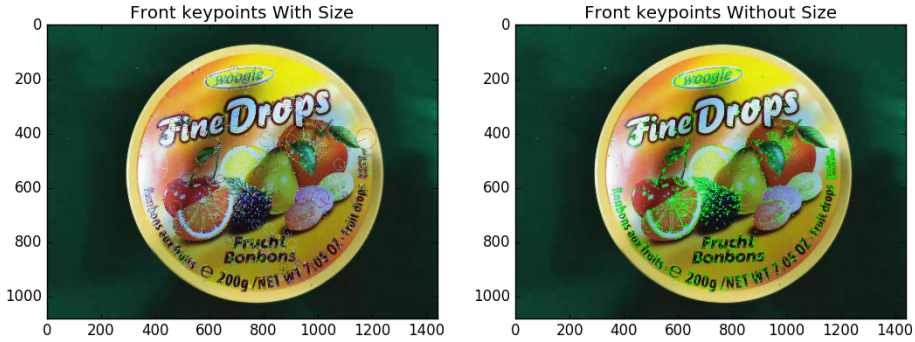
```
10.
11. side_image = cv2.imread('/home/cyx/桌面
    /cv_homework/vision_sift_surf/INPUT/side.jpeg')
12. side_image = cv2.cvtColor(side_image, cv2.COLOR_BGR2RGB)
13. side_gray = cv2.cvtColor(side_image, cv2.COLOR_RGB2GRAY)
14.
15.
16. # 显示图片
17. fig, plots = plt.subplots(1, 2, figsize=(20,10))
18.
19. plots[0].set_title("FRONT Image")
20. plots[0].imshow(front_image)
21.
22. plots[1].set_title("SIDE Image")
23. plots[1].imshow(side_image)
24. plt.show()
25. cv2.waitKey()
26.
27. sift = cv2.xfeatures2d.SIFT_create()
28.
29. front_keypoints, front_descriptor = sift.detectAndCompute(front_gray, None)

30. side_keypoints, side_descriptor = sift.detectAndCompute(side_gray, None)
31.
32. keypoints_without_size = np.copy(front_image)
33. keypoints_with_size = np.copy(front_image)
34.
35. cv2.drawKeypoints(front_image, front_keypoints, keypoints_without_size, colo
    r = (0, 255, 0))
36.
37. cv2.drawKeypoints(front_image, front_keypoints, keypoints_with_size, flags =
     cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
38.
39. # 显示提取结果
40. fx, plots = plt.subplots(1, 2, figsize=(20,10))
41.
42. plots[0].set_title("Front keypoints With Size")
43. plots[0].imshow(keypoints_with_size, cmap='gray')
44.
45. plots[1].set_title("Front keypoints Without Size")
46. plots[1].imshow(keypoints_without_size, cmap='gray')
47.
48. plt.show()
49. cv2.waitKey()
```

```
50.
51. # 匹配
52. bf = cv2.BFMatcher(cv2.NORM_L1, crossCheck = False)
53. matches = bf.match(front_descriptor, side_descriptor)
54. # 按照距离排序，距离越小匹配效果好
55. matches = sorted(matches, key = lambda x : x.distance)
56.
57. result = cv2.drawMatches(front_image, front_keypoints, side_gray, side_keypo
    ints, matches[:50], side_gray, flags = 2)
58.
59. # 显示最佳匹配结果
60. plt.rcParams['figure.figsize'] = [14.0, 7.0]
61. plt.title('Best Matching Points')
62. plt.imshow(result)
63. plt.show()
64.
65.
66. cv2.waitKey()
```

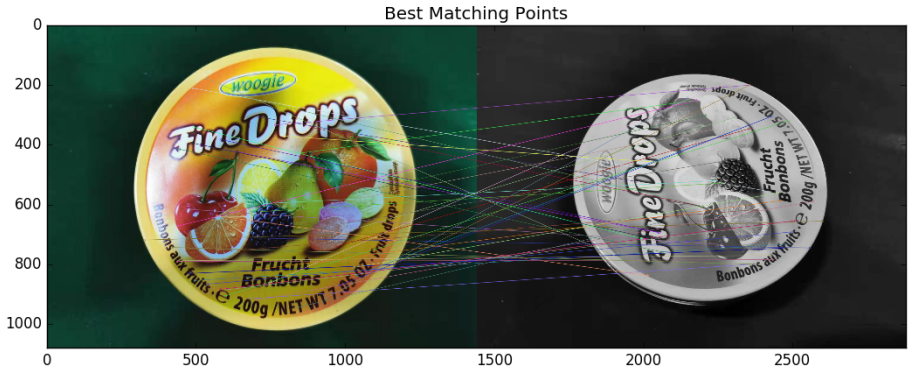
实验采用的图片为正面近距离以及侧面远距离得到的糖果盒图像，原图如下



提取 SIFT 特征点结果如下



根据特征点间的欧氏距离进行匹配得到的匹配结果如下



可以看到，连线对应的端点为两幅图片间的对应部分，可以根据距离阈值来对匹配进行筛选，这里认为 SIFT 匹配效果较好。

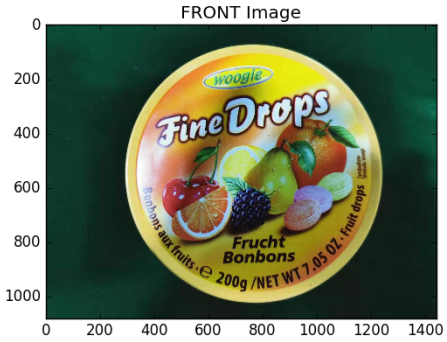
SURF，也即 Speeded Up Robust Features，是对于 SIFT 算法的一种改进。SURF 选用 Hessian 矩阵变换图像的方式替代 SIFT 采用的构建高斯差分金字塔的方式检测图像中尺度空间上的极值点。另外，SURF 利用盒装模糊滤波求高斯模糊的近似值，通过对盒子滤波器的尺寸变化来代替 SIFT 中的下采样，更快捷地构建尺度空间。最后 SURF 在计算关键点的主方向时，采用 Harr 小波转换代替直方图统计，从而简化运算，加快速度。在构建特征描述子时，SURF 也采用了更小的数据结构，降低了运算的维数。

具体代码实现如下

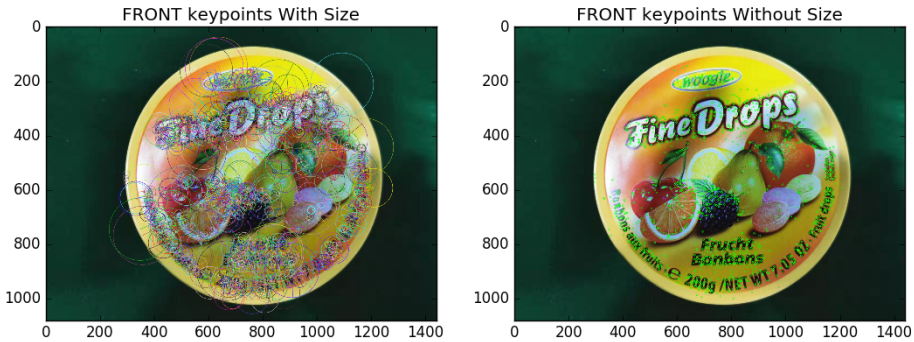
```
import numpy as np
import cv2
from matplotlib import pyplot as plt

# 读取图片
image1 = cv2.imread('/home/cyx/桌面/cv_homework/vision_sift_surf/INPUT/front.jpeg')
front_image = cv2.cvtColor(image1, cv2.COLOR_BGR2RGB)
front_gray = cv2.cvtColor(front_image, cv2.COLOR_RGB2GRAY)


side_image = cv2.imread('/home/cyx/桌面/cv_homework/vision_sift_surf/INPUT/side.jpeg')
side_image = cv2.cvtColor(side_image, cv2.COLOR_BGR2RGB)
side_gray = cv2.cvtColor(side_image, cv2.COLOR_RGB2GRAY)


# 显示图片
fig, plots = plt.subplots(1, 2, figsize=(20,10))

plots[0].set_title("FRONT Image")
plots[0].imshow(front_image)

plots[1].set_title("SIDE Image")
plots[1].imshow(side_image)
plt.show()
cv2.waitKey()

surf = cv2.xfeatures2d.SURF_create(800)

front_keypoints, front_descriptor = surf.detectAndCompute(front_gray, None)

side_keypoints, side_descriptor = surf.detectAndCompute(side_gray, None)

keypoints_without_size = np.copy(front_image)
keypoints_with_size = np.copy(front_image)
```

```
34.
35. cv2.drawKeypoints(front_image, front_keypoints, keypoints_without_size, colo
    r = (0, 255, 0))
36.
37. cv2.drawKeypoints(front_image, front_keypoints, keypoints_with_size, flags =
     cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
38.
39. # 显示提取结果
40. fx, plots = plt.subplots(1, 2, figsize=(20,10))
41.
42. plots[0].set_title("FRONT keypoints With Size")
43. plots[0].imshow(keypoints_with_size, cmap='gray')
44.
45. plots[1].set_title("FRONT keypoints Without Size")
46. plots[1].imshow(keypoints_without_size, cmap='gray')
47.
48. plt.show()
49. cv2.waitKey()
50.
51. # 匹配
52. bf = cv2.BFMatcher(cv2.NORM_L1, crossCheck = False)
53. matches = bf.match(front_descriptor, side_descriptor)
54. # 按照距离排序，距离越小匹配效果好
55. matches = sorted(matches, key = lambda x : x.distance)
56.
57. result = cv2.drawMatches(front_image, front_keypoints, side_gray, side_keypo
    ints, matches[:50], side_gray, flags = 2)
58.
59. # 显示最佳匹配结果
60. plt.rcParams['figure.figsize'] = [14.0, 14.0]
61. plt.title('Best Matching Points')
62. plt.imshow(result)
63. plt.show()
64.
65.
66. cv2.waitKey()
```

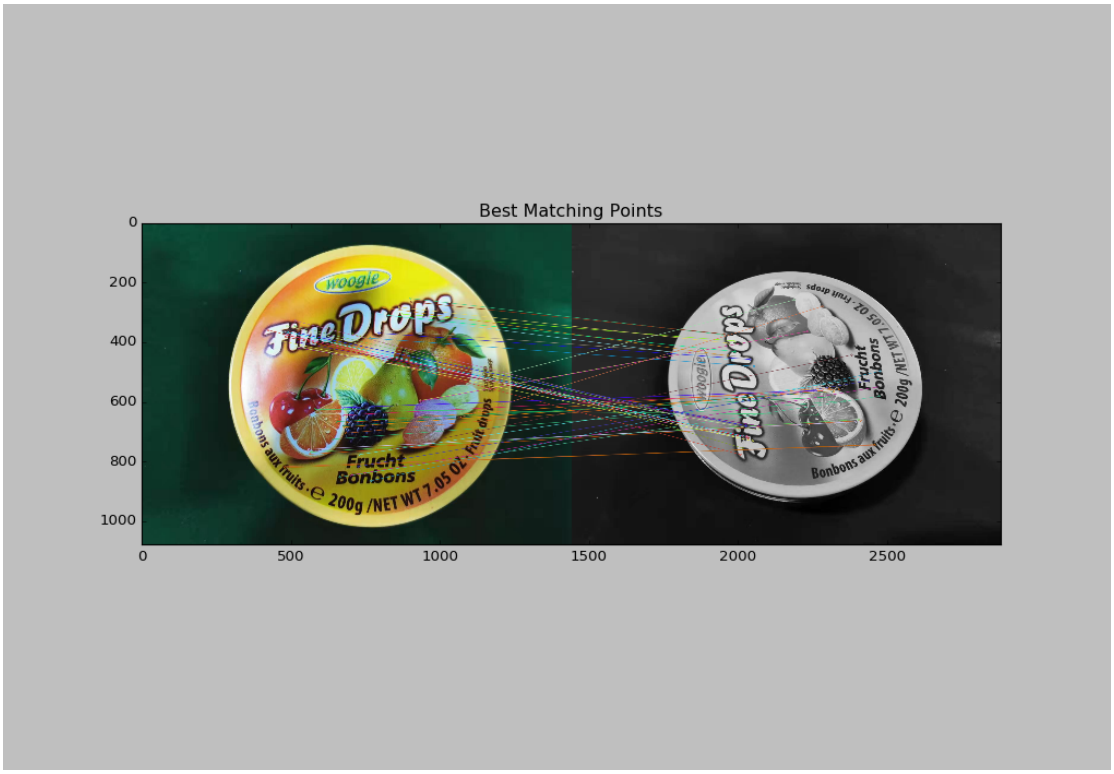
同样的，原图如下



提取 SURF 特征点结果如下



进行特征匹配可以得到



对比两者结果可以发现，SURF 算法相对于 SIFT 算法运行效率更高，特征点的寻找更快，更多。但是匹配中相对于 SIFT 算法会产生更多的不稳定点，存在一定的误差。但是两者都可以实现较为精准的图片匹配，认为满足实验要求。

四、 实验探究：

整体来讲，SIFT 算法匹配准确度相对较高，SURF 算法速度较快，需要根据项目实际需求进行权衡。

五、 代码：

见附件