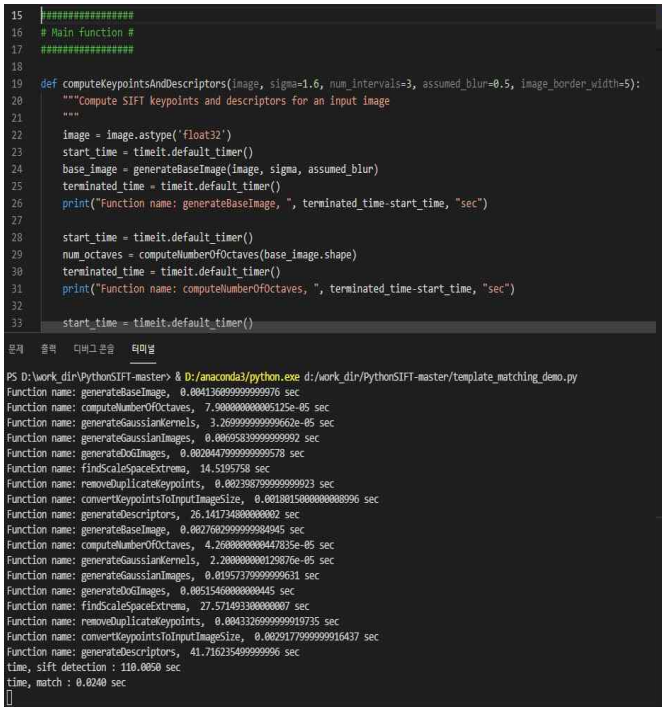
computeKeypointsAndDescriptors 내에서 실행되는 함수들의 실행 시간 분석을 위해, template_matching_demo.py에서 time, sift detection에 걸리는 시간을 측정한 것과 같이 time module을 import하여 시간을 측정한다. 또는 아래처럼 timeit module을 import해도 좋다(Fig. 1). 함수 앞·뒤로 현재 시점을 기록하고 둘의 차이를 구하면 함수 실행 시간을 알 수 있다

가장 느린 함수는 **generateDescriptors**이고 time, sift detection에 무려 110초가 소요됐다. generateDescriptors는 1) keypoints로부터 keypoint를 가져와서 각 keypoint마다 octave, layer, scale을 추출, 2) gaussian_images에서 해당 gaussian_image를 가져와서, 3) 각 row, col에 접근하여 gradient_magnitude, gradient_orientation, weight를 얻은 후 histogram_tensor를 업데이트한다. 4) histogram_tensor는 flatten되어 0~255로 정규화를 거친 뒤 return된다.

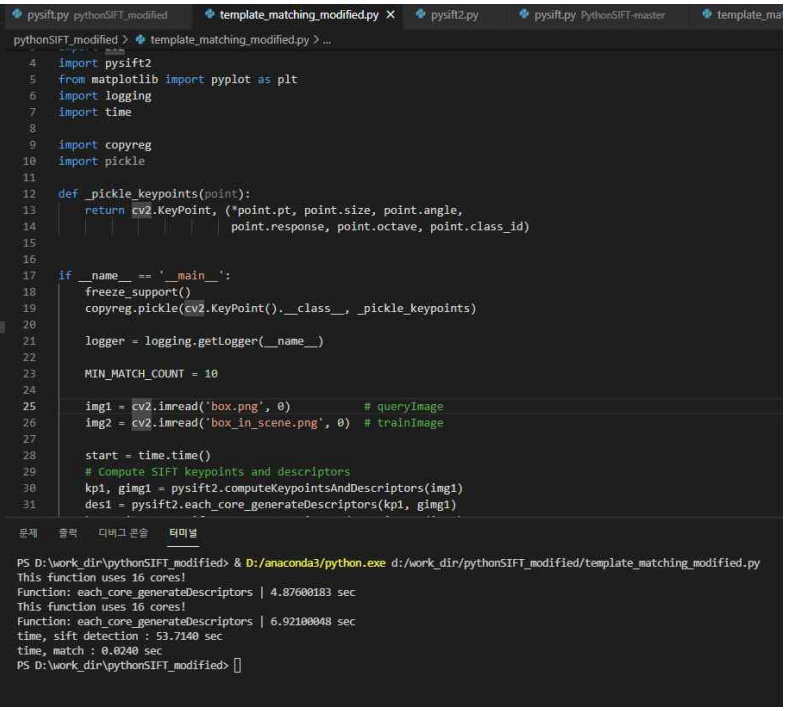
이때 모든 keypoint로부터 histogram_tensor를 업데이트하는 과정은 똑같은데, keypoints와 gaussian_images가 한 함수로 전달되어 하나의 프로세스 내에서만 순차적으로 실행된다. 이 과정을 조금 변형해주어 여러 core를 이용하여 여러 프로세스에서 처리되게 하면 시간을 많이 단축할 수 있다. multiprocessing module을 import하여 zip(keypoint, gaussian_image, scale)을 인자로 넘겨줘서 처리하게 하고, cv2.Keypoint 형식은 pickle되지 않으므로 main에서 간단히 전처리를 해주었다. 측정 결과, time, sift detection에 53.7초가 소요되었다(Fig. 2). 실행 환경은 16 cores를 사용하는 환경이다.



```
##################
# Main function #
##################

def computeKeypointsAndDescriptors(image, sigma=1.6, num_intervals=3, assumed_blur=0.5, image_border_width=5):
    """Compute SIFT keypoints and descriptors for an input image
    """
    image = image.astype('float32')
    start_time = timeit.default_timer()
    base_image = generateBaseImage(image, sigma, assumed_blur)
    terminated_time = timeit.default_timer()
    print("Function name: generateBaseImage, ", terminated_time-start_time, "sec")

    start_time = timeit.default_timer()
    num_octaves = computeNumberOfOctaves(base_image.shape)
    terminated_time = timeit.default_timer()
    print("Function name: computeNumberOfOctaves, ", terminated_time-start_time, "sec")

    start_time = timeit.default_timer()
```

```
PS D:\work_dir\PythonSIFT-master> & D:/anaconda3/python.exe d:/work_dir/PythonSIFT-master/template_matching_demo.py
Function name: generateBaseImage,  0.004136099999999976 sec
Function name: computeNumberOfOctaves,  7.900000000005125e-05 sec
Function name: generateGaussianKernels,  3.269999999999662e-05 sec
Function name: generateGaussianImages,  0.006958399999999992 sec
Function name: generateDoGImages,  0.0020447999999999578 sec
Function name: findScaleSpaceExtrema,  14.5195758 sec
Function name: removeDuplicateKeypoints,  0.002398799999999923 sec
Function name: convertKeypointsToInputImageSize,  0.0018015000000008996 sec
Function name: generateDescriptors,  26.141734000000002 sec
Function name: generateBaseImage,  0.0027682999999984945 sec
Function name: computeNumberOfOctaves,  4.260000000447835e-05 sec
Function name: generateGaussianKernels,  2.200000000129876e-05 sec
Function name: generateGaussianImages,  0.01957379999999631 sec
Function name: generateDoGImages,  0.0051546000000445 sec
Function name: findScaleSpaceExtrema,  27.571493300000007 sec
Function name: removeDuplicateKeypoints,  0.004332699999919735 sec
Function name: convertKeypointsToInputImageSize,  0.0029177999999916437 sec
Function name: generateDescriptors,  41.716235499999996 sec
time, sift detection : 110.0050 sec
time, match : 0.0240 sec
```

Fig. 1. 각 함수 실행 시간 측정; 함수가 두 번 실행됨.



```
import pysift2
from matplotlib import pyplot as plt
import logging
import time

import copyreg
import pickle

def _pickle_keypoints(point):
    return cv2.KeyPoint, (*point.pt, point.size, point.angle,
                          point.response, point.octave, point.class_id)


if __name__ == '__main__':
    freeze_support()
    copyreg.pickle(cv2.KeyPoint().__class__, _pickle_keypoints)

    logger = logging.getLogger(__name__)

    MIN_MATCH_COUNT = 10

    img1 = cv2.imread('box.png', 0)          # queryImage
    img2 = cv2.imread('box_in_scene.png', 0)  # trainImage

    start = time.time()
    # Compute SIFT keypoints and descriptors
    kp1, gimg1 = pysift2.computeKeypointsAndDescriptors(img1)
    des1 = pysift2.each_core_generateDescriptors(kp1, gimg1)
```

```
PS D:\work_dir\pythonSIFT_modified> & D:/anaconda3/python.exe d:/work_dir/pythonSIFT_modified/template_matching_modified.py
This function uses 16 cores!
Function: each_core_generateDescriptors | 4.87600183 sec
This function uses 16 cores!
Function: each_core_generateDescriptors | 6.92100048 sec
time, sift detection : 53.7140 sec
time, match : 0.0240 sec
PS D:\work_dir\pythonSIFT_modified>
```

Fig. 2. 가장 느린 함수 실행 시간 측정; 1/8 정도로 감소했다.