3.3.1. Utilice el ambiente virtual configurado en la sección 3.2 para ejecutar el código motion_detector.py con la Raspberry Pi Camera Module v2.

- Se utilizo el siguiente comando para activar el ambiente virtual configurado en la seccion anterior

$ workon SEAD_p2

3.3.2. Utilice cProfile para obtener métricas de perfilado de la aplicación. Ejecute la aplicación con cProfile por al menos 10 segundos y máximo 15 segundos

- Se utlizo el siguiente commando para obetener la metricas.

$ DISPLAY=:0 python3 -m cProfile -o prof_rpi_cam.out motion_detector.py

3.3.3. Utilice pstats para reordenar los resultados obtenidos en el paso anterior y visualizar las 10 funciones con mayor tiempo interno (no tiempo acumulado). Asegúrese de utilizar la función strip para hacer más legible los resultados. Puede basarse en el link:
https://www.stefaanlippens.net/python_profiling_with_pstats_interactive_mode
Escriba el resultado obtenido con “stats 10” en el reporte final. (5 pts)

- Se utilizaron los siguientes comandos para obtener las 10 funciones con mayor tiempo interno

$ python3 -m pstats prof_rpi_cam.out

% strip
% sort time
% stats 10

```
Thu Jul 11 05:21:20 2019    prof_rpi_cam.out

         82741 function calls (79969 primitive calls) in 13.548 seconds

   Ordered by: internal time
   List reduced from 1289 to 10 due to restriction <10>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      152    3.363    0.022    3.363    0.022 {GaussianBlur}
      152    3.032    0.020    3.032    0.020 {resize}
        1    2.002    2.002    2.002    2.002 {built-in method time.sleep}
      150    1.391    0.009    1.391    0.009 {waitKey}
       13    0.716    0.055    0.729    0.056 {built-in method _imp.create_dynamic}
      450    0.625    0.001    0.625    0.001 {imshow}
        1    0.460    0.460    0.460    0.460 {method 'read' of 'cv2.VideoCapture' objects}
      150    0.358    0.002    0.358    0.002 {findContours}
      300    0.219    0.001    0.219    0.001 {putText}
      150    0.159    0.001    0.159    0.001 {dilate}
```

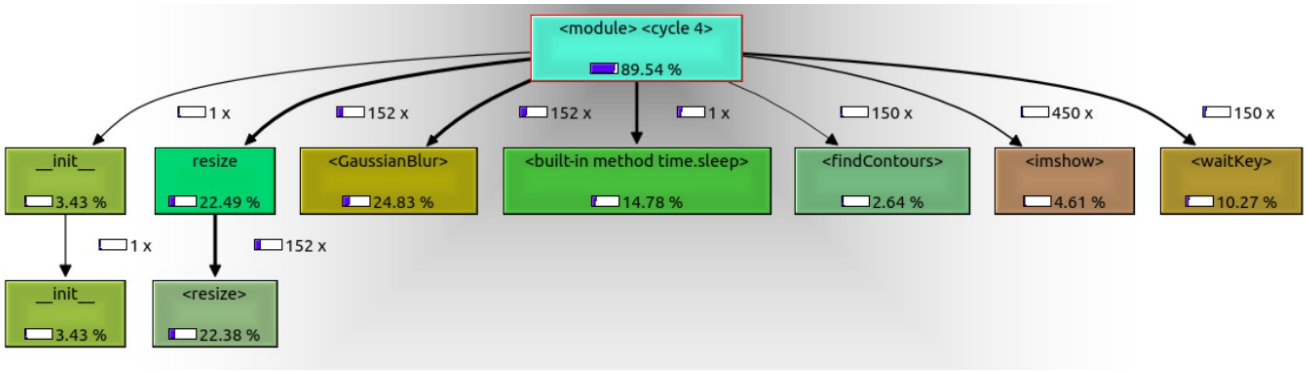
3.3.4. Utilice KCacheGrind como herramienta de visualización de perfilado para obtener el Call Graph de la aplicación. Para utilizar KCacheGrind con Python, se recomienda utilizar el script pyprof2calltree.py.
Incluya el Call Graph en el reporte final (exportando el call graph como imagen). Nombre el archivo call_graph_rpi.png (10 pts)

- Para convertir el archivo de metricas *prof_rpi_cam.out* en un formato soportado por KcacheGrind se utilizo la herramienta pyprof2calltree. Los siguientes comandos fueron necesarios para la instalacion y uso de la misma.

$ pip3 install pyprof2calltree
$ pyprof2calltree -i prof_rpi_cam.out -o prof_rpi_cam_callgrind.out

- Para la visualizacion del perfilado se utilizo KcacheGrind. A continuacion el comando de instalacion y resultado.

$ sudo apt-get install kcachegrind



3.3.5. Utilice la Raspberry Pi Camera Module v2 para grabar un video corto (mínimo 5 segundos). Puede utilizar herramientas incluidas en la distribución de Raspbian para esto.

- Se utilizaron los siguientes comandos para grabar un video de 10 segundos y resolucion 640x480 y su respectiva conversion a formato MP4.

$ raspivid -t 10000 -w 640 -h 480 -o test_video.h264
$ MP4Box -add test_video.h264 test_video.mp4

3.3.7. Similar a los pasos 3.3.2 y 3.3.3, utilice cProfile y pstats para obtener métricas de perfilado con cProfile y visualización de las 10 funciones con mayor tiempo interno sólo que esta vez debe hacerlo con el video corto grabado: python -m cProfile -o out.prof motion_detector.py –video video.mp4

- Se utilizo el siguiente commando para obtener las metricas de la ejecucion de motion_detector utilizando el video grabado.

$ DISPLAY=:0 python3 -m cProfile -o prof_rpi_vid1.out motion_detector.py --video test_video.mp4

- Se utilizaron los siguientes comandos para obtener las 10 funciones con mayor tiempo interno

$ python3 -m pstats prof_rpi_vid1.out

% strip
% sort time
% stats 10

```
Sat Jul 13 00:23:34 2019    prof_rpi_vid1.out

         86429 function calls (83652 primitive calls) in 9.272 seconds

   Ordered by: internal time
   List reduced from 1285 to 10 due to restriction <10>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      288    3.660    0.013    3.660    0.013 {resize}
      289    2.006    0.007    2.006    0.007 {method 'read' of 'cv2.VideoCapture' objects}
       13    0.707    0.054    0.721    0.055 {built-in method _imp.create_dynamic}
      288    0.655    0.002    0.655    0.002 {GaussianBlur}
      287    0.485    0.002    0.485    0.002 {waitKey}
      861    0.342    0.000    0.342    0.000 {imshow}
      574    0.248    0.000    0.248    0.000 {putText}
        1    0.168    0.168    9.272    9.272 motion_detector.py:6(<module>)
      166    0.109    0.001    0.109    0.001 {built-in method marshal.loads}
      287    0.070    0.000    0.070    0.000 {findContours}
```

3.3.8. En el código motion_detector.py, busque la línea de código donde se llama a la función resize, para modificar el tamaño de la ventana:

```
# resize the frame, convert it to grayscale, and bl
frame = imutils.resize(frame, width=500)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
# resize the frame, convert it to grayscale, and
frame = imutils.resize(frame, width=100)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Vuelva a ejecutar los pasos 3.3.2 y 3.3.3.

- Se utilizo el siguiente commando para obtener las metricas de la ejecucion de motion_detector utilizando el video grabado.

$ DISPLAY=:0 python3 -m cProfile -o prof_rpi_vid2.out motion_detector.py --video test_video.mp4

- Se utilizaron los siguientes comandos para obtener las 10 funciones con mayor tiempo interno

$ python3 -m pstats prof_rpi_vid2.out

% strip
% sort time
% stats 10

```
Sat Jul 13 00:23:54 2019    prof_rpi_vid2.out

         86432 function calls (83655 primitive calls) in 9.298 seconds

   Ordered by: internal time
   List reduced from 1285 to 10 due to restriction <10>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      288    3.659    0.013    3.659    0.013 {resize}
      289    2.027    0.007    2.027    0.007 {method 'read' of 'cv2.VideoCapture' objects}
       13    0.698    0.054    0.712    0.055 {built-in method _imp.create_dynamic}
      288    0.642    0.002    0.642    0.002 {GaussianBlur}
      287    0.507    0.002    0.507    0.002 {waitKey}
      861    0.354    0.000    0.354    0.000 {imshow}
      574    0.248    0.000    0.248    0.000 {putText}
        1    0.165    0.165    9.299    9.299 motion_detector.py:6(<module>)
      166    0.109    0.001    0.109    0.001 {built-in method marshal.loads}
      287    0.069    0.000    0.069    0.000 {findContours}
```