**3.2.2. Responda las siguientes preguntas relacionadas con la instalación de Python 3 y**

**OpenCV 4:**

**¿Cuáles son las ventajas de usar un ambiente virtual de Python? (2 pts)**

**¿Qué es NumPy? (2 pts)**

**Describa ejemplos de 2 aplicaciones de numpy. (2 pts)**

**¿Para qué sirve CMake? (2 pts)**

**¿Qué ventajas/desventajas tiene CMake ante otras herramientas de configuración**

**de Makefiles (Autotools, qmake, Scons, etc) en sistemas embebidos? (2 pts)**

**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**](https://www.stefaanlippens.net/python_profiling_with_pstats_interactive_mode)

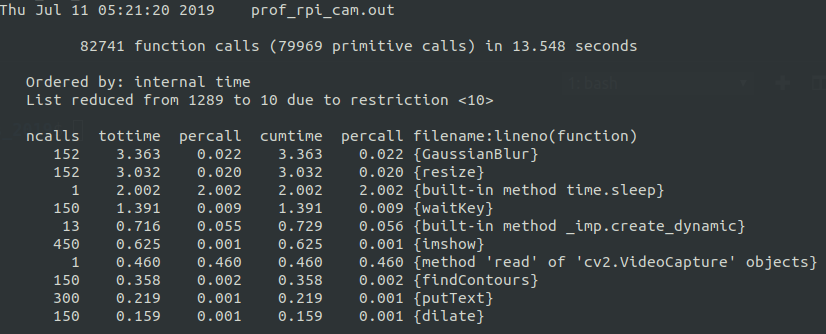
**Escriba el resultado obtenido con “stats 10” en el reporte final. (5 pts)**

- 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*

**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.**

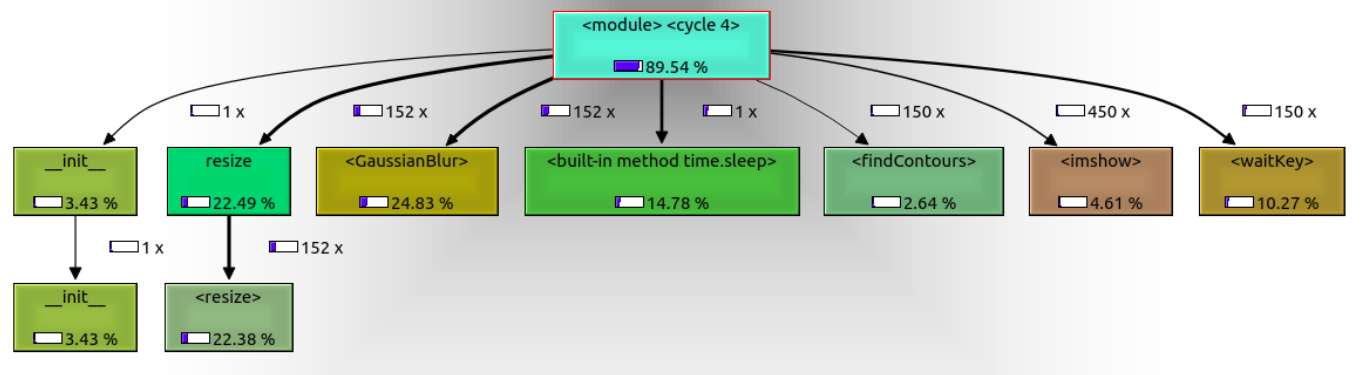
- Para convertir el archivo de metricas *prof\_rpi\_cam.out* en un formato soportado por KcacheGrind se utilizo la herramienta pyprof2calltree. Los siguientes comandos fueron necesarios para la instalacion y uso de la misma.

*$ 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*

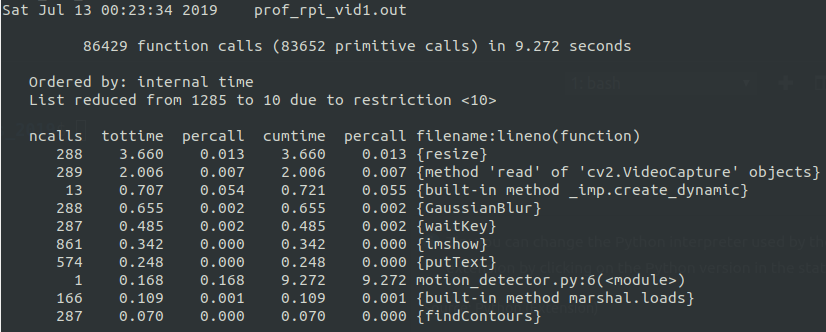
- Se utilizaron los siguientes comandos para obtener las 10 funciones con mayor tiempo interno

*$ python3 -m pstats prof\_rpi\_vid1.out*

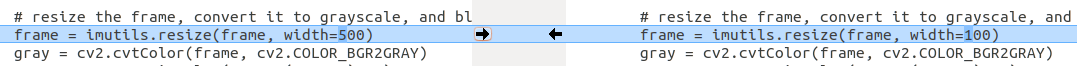
*% strip*

*% sort time*

*% stats 10*

**

**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:**

**

*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*

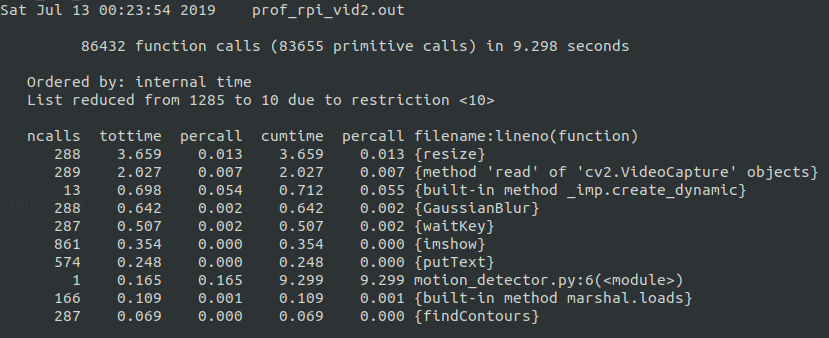
- Se utilizaron los siguientes comandos para obtener las 10 funciones con mayor tiempo interno

*$ python3 -m pstats prof\_rpi\_vid2.out*

*% strip*

*% sort time*

*% stats 10*

**

**3.3.9. (Opcional)**

**Optimice la función “resize” para obtener al menos un 20% de mejora en el tiempo**

**de ejecución total de la aplicación. La única restricción es que no puede modificar**

**el tamaño de resizing. Es decir, no puede cambiar el argumento “width=500” ni**

**modificar de ninguna otra forma el tamaño actual de las ventanas**