Speed Improvements for Image Recognition

**Aufnahmemodus:**

Aufgenommen wird im “capture\_continuos» modus, direkt ab Raspberry Pi Cam.



Es wird jedes Frame in einer For-Schleife verarbeitet.

Am Ende jedes Durchganges wird via Truncate das ‘PiRGBArray geleert. Somit ist die wahre FPS (Frames Per Seconds) Prozessierrate abhängig von der Verarbeitungsgeschwindigkeit des Codes.

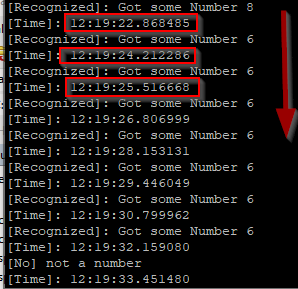


**Messergebnis Singlecore**

Mit der ursprünglichen Lösung wir alles in einem Prozess verarbeitet, das beinhaltet:

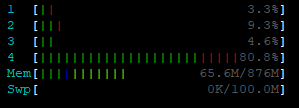
* Auswertung des Frames
* Konturen finden (potentielle Nummer)
* Konturen auswerten mit Tesseract (ist es eine Nummer)

Dieser Prozess dauert sehr lange und belegt beinah 100% CPU eines Cores:



Die Verarbeitungszeit beträgt im Schnitt 1.3 Sekunden (pro Frame)

CPU Auslastung:



**Multicore Lösung:**

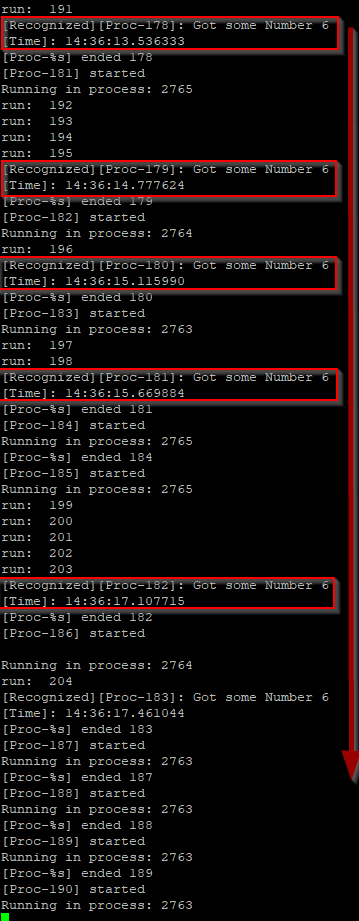
**Anpassungen Verarbeitung**

Folgende Anpassungen wurden vorgenommen:

* Verwendung der multiprocessor.Pool Funktion
* 3-4 Worker Pool bereitstellen: pool = Pool(processes=4)
* Jedes Img (aus frame generiert) an externen Worker Prozess übergeben: pool.apply\_async(process\_frame\_worker, (img,counter,))

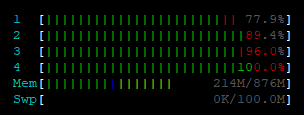
**Messergebnisse Multicore**

Die Prozesse laufen zwar asynchron ab, jedoch spielt dies keine grosse Rolle in der Praxis:



Durch die Multiprocessor Lösung dauert das erkennen einer Nummer im Schnitt noch 0.5s mit einem Pool von 3 Workern. Somit können 2-3 mal mehr Frames in der gleichen Zeit verarbeitet werden.

CPU Auslatung:



Die Multiprocess Variante lastet den RPI fast zu 100% aus auf jedem Kern.

**Weitere Bemerkungen Multiprocessing Variante**

Durch das Multiprocessing wird das RAM effektiv schneller belegt.  
Es empfiehlt sich unbedingt eine Erhöhung des SWAPS von 100Mb auf 512MB:

sudo sed -i 's/CONF\_SWAPSIZE=100/CONF\_SWAPSIZE=512/g' /etc/dphys-swapfile

sudo /etc/init.d/dphys-swapfile stop

sudo /etc/init.d/dphys-swapfile start

**Weitere Verwendung des Multiprocessing Verfahrens**

Durch Multiprocessing lassen sich nun 3-4 nacheinanderfolgende Frames verarbeiten. Das ermöglicht ein fast eindeutiges erkennen der Nummer. Jedoch werden die Jobs «queued», wenn also viele Nummern nacheinander erkannt werden (z.B. 100) wird die Queue sehr gross und das Memory füllt sich schnell.

Deshalb sollte noch ein Break Out programmiert werden, sobald dieselber Nummer x-mal erkannt wurde, soll die For-Schleife beendet und alle Workers terminiert werden.