Profiling

Code profiling is the process of identifying where in a program a CPU is spending most of its time. If you need to speed up your program, profiling your code will help identify where you should optimize!

The following instructions are based on this website.

C++

Run valgrind on your program:

```
valgrind --tool=callgrind <executable name>
```

This will start up your node. Let it run for a while, then use `rosnode kill` to kill your node, and valgrind will dump its output to a file called `callgrind.out.<pid>`.

Next, load up this output file with kcachegrind:

```
kcachegrind callgrind.out.<pid>
```

Python

Run your program with profiling turned on:

```
python -m cProfile -o <profile file name> <script>.py
```

Same as the C++ instructions, let it run for a while and then use rosnode kill to kill your node. The output will be dumped to `<profile file name>`.

View the output in kcachegrind:

```
pyprof2calltree -i <profile file name> -k
```

Notes on Optimizing Code

Profiling does not tell you whether or not you should optimize your code, it only tells you **where** optimizing would do the most help. Only profile your code if you notice your program is running with a much higher CPU usage than what you would expect (or just for fun if you're curious). There's no need to profile every program you write.

The control system is a good example here. Profiling the program at at its current state (1/2/2017) reveals that 87% of the time the control system is running its main update function which calculates what values to send to the thrusters. This is unsurprising. Within this function, 41% of the time is spent reloading the PID parameters from the parameter server, 30% of the time is spent in the r3D() function, while the remaining 29% is doing the rest of the control calculations.

Seeing this, one might think *“OMG, reloading the PID parameters every cycle is so wasteful! We should cut that out now!”*. However, it's important to keep this in perspective, the control system uses a grand total of 2.3% of my machine's computing resources. Sure, a large chunk of that 2.3% is spent reloading PID parameters, but who cares? If someday we are really strapped for CPU resources we can modify the program to reload parameters in a timer callback (so we can have it occur less often), and then look how we can optimize the r3D() function.

Just because you can optimize doesn't mean it's worth the effort. Remember, the control system is just a drop in the bucket compared to the vision processing system. Look at the overall CPU usage of a node relative to all others to determine if optimization is warranted.

Profiling Caveats

Here is a great article on how profilers can lie to us.