|  |  |
| --- | --- |
| **Running time** | **Function** |
| ~0.3 For One camera (frame) | **Surf.detectandcompute** (open source func.) |
| 0.602 = 0.3\*2+0.02(saving desc and kp) | Mainclass.SurfAllcams |
| ~0.05 | 1-**Flann.knnmatch**(between old and new frame from same cam) |
| ~0.14=0.05+0.09(choosing good 3d points and 2d points for solvepnp) | Mainclass.matchingOLDframessfor2d |
| 0.029 | Mainclass.solvepnpransacm |
| ~0.038=0.029+0.09 | Solvepnpframe2 |
| 1.35~ | 2-**Flann.knnmatch**(between frames taken at the sames time from two different cams) |
| 1.36=1.35+0.01~ | Mainclass.matchingandtriangulation |
| ~2.1517=(2\*Surf.detectandcompute )+( 1-Flann.knnmatch)+(2\*2-flann.knnmatch) + 0.1517(the rest of my code) | Total Running Time of one iteration  (one a single frame from each camera) |

Tables showing different running times using different methods

Using the **Flann.knnmatch** to mach between descriptors

Here im using the **brute force matcher (bf.knnmatch)** which is clearly a much faster method but less accurate than the Flann matcher .

Using the brute force matcher I can cut the running time in half

|  |  |
| --- | --- |
| **Running time** | **Function** |
| ~0.3 For One camera (frame) | **Surf.detectandcompute** (open source func.) |
| 0.602 = 0.3\*2+0.02(saving desc and kp) | Mainclass.SurfAllcams |
| ~0.06 | 1-**bf.knnmatch**(between old and new frame from same cam) |
| ~0.15=0.06+0.09(choosing good 3d points and 2d points for solvepnp) | Mainclass.matchingOLDframessfor2d |
| 0.029 | Mainclass.solvepnpransacm |
| ~0.038=0.029+0.09 | Solvepnpframe2 |
| 0.35~ | 2**-bf.knnmatch**(between frames taken at the sames time from two different cams) |
| 0.36=0.35+0.01~ | Mainclass.matchingandtriangulation |
| ~1.172=(2\*Surf.detectandcompute )+( 1-bf.knnmatch)+(2\*2-bf.knnmatch) + 0.172(the rest of my code) | Total Running Time of one iteration  (one a single frame from each camera) |

Conclusion:

Doing this project I realized that the bottleneck was without a doubt the SURF and MATCHING (either bf or Flann) because no matter how efficiently one can code whenever you use either one of these methods your running time will be dominated by it, and the more cameras you have the longer it will get .