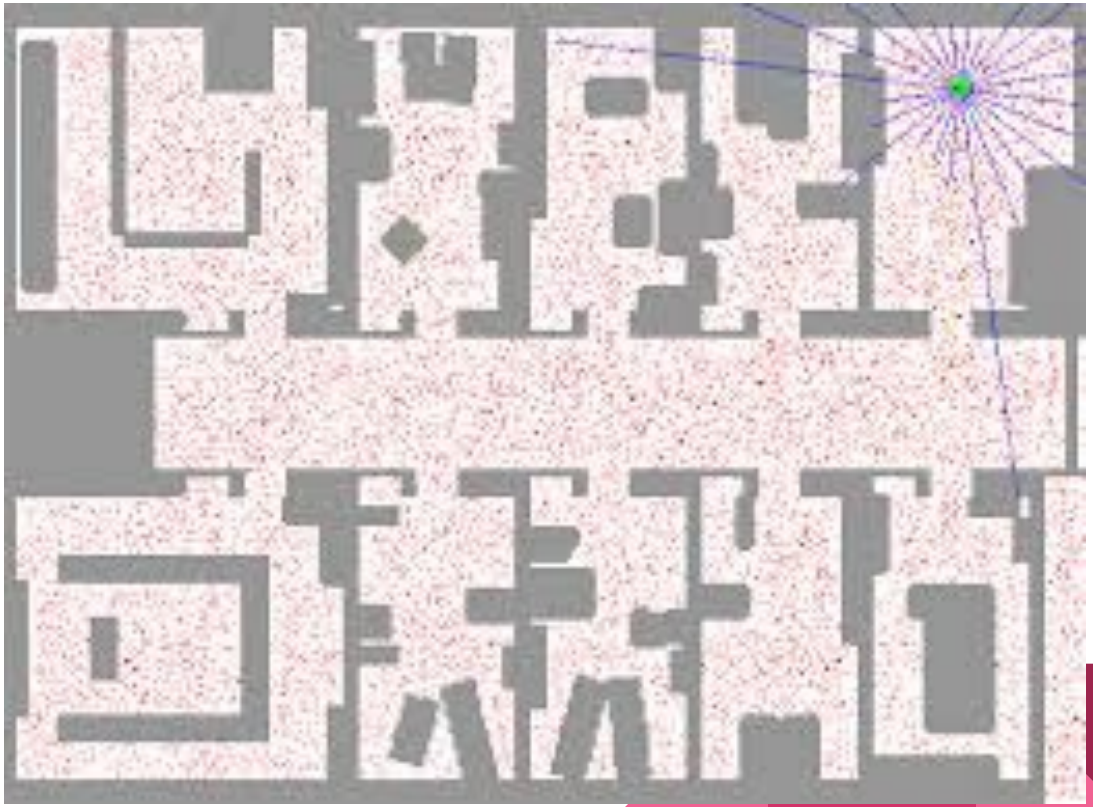
Parallel Particle Filter

Kelvin Silva

Introduction

Particle Filter:

Monte Carlo Localization



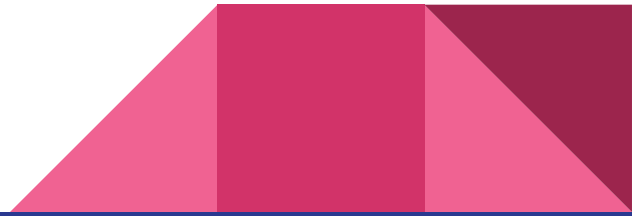
Serial = Slow

1 Motion Update

2 Measurement Update

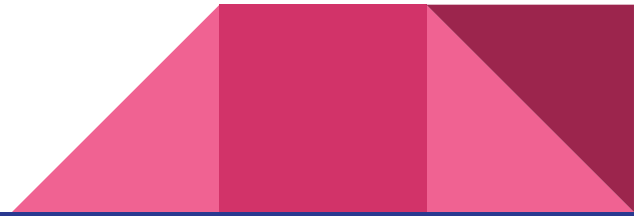
3 Resampling step

For N -> big number doing these 3 steps iteratively is slow



Parallel = Faster

Can use GPU to do Motion, Measurement updates on every particle at the same time



Resampling is Harder

Resampling step is most commonly done with iterative approach based on a cumulative sum of weights.

So need to use Metropolis resampling

Parallel:

Pseudocode for Metropolis resampling.

METROPOLIS-ANCESTORS($\mathbf{w} \in [0,\infty)^N, B \in \{1,2,\ldots\}) \to \{1,\ldots,N\}^N$

```
for each i ∈ {1,...,N}
    k ← i
    for n = 1,...,B
        u ~ U[0,1]
        j ~ U{1,...,N}
        if u ≤ w^j / w^k
            k ← j
    a^i ← k
return a
```

Serial:

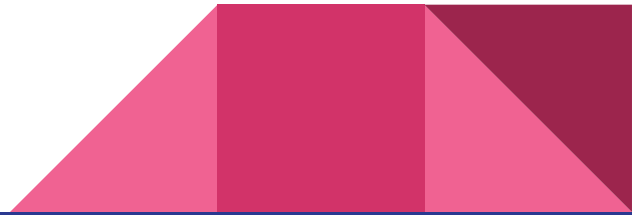
```
# particle filter resampling
p3 = []
index = int(random.random() * N)
beta = 0.0
mw = max(w)
for i in range(N):
    beta += random.random() * 2.0 * mw
    while beta > w[index]:
        beta -= w[index]
        index = (index + 1) % N
    p3.append(p[index])
p = p3
```

Code Review

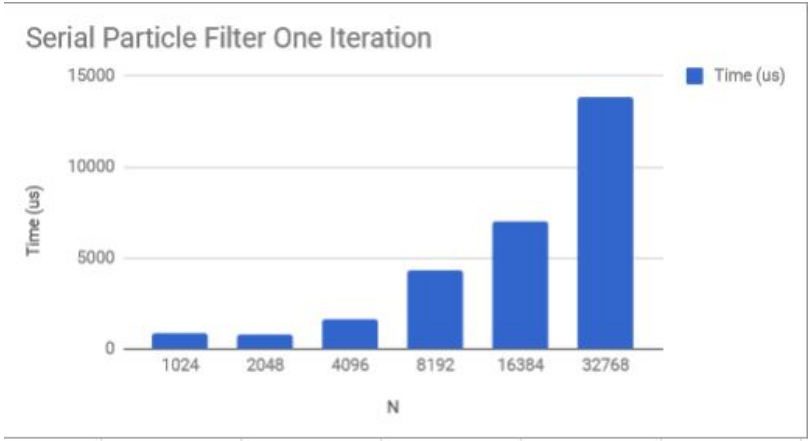
https://github.com/kelvinsilva/ams148-gpu-programming-cuda/tree/master/particle-filter-parallel

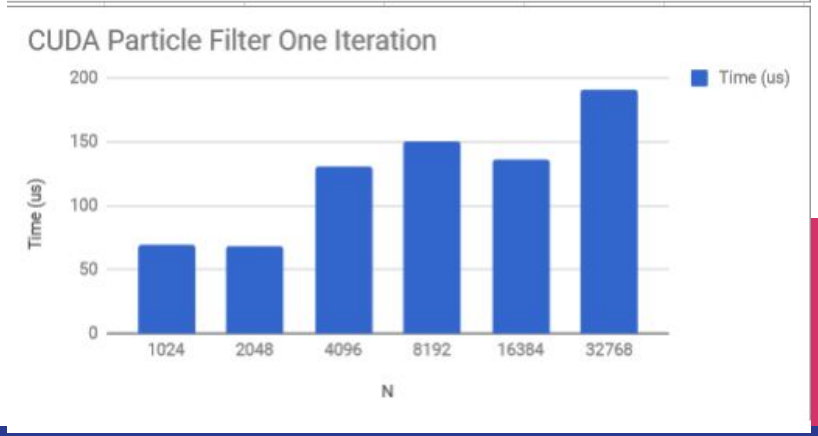
Thrun Udacity:

https://gist.github.com/kelvinsilva/6c19c12c54c5d873aad01f65017cb7d9



Results Speedup





Conclusion

For large amount of particles better to parallelize

Due to algorithm runtime complexity, even an unoptimized parallel particle filter will run faster than a serial one for LARGE N

For small amounts (less than 1024) iterative is better

GPU Memory limiting factor