

Bayesian Particle Filter Tracking with CUDA

Geoffrey Ulman

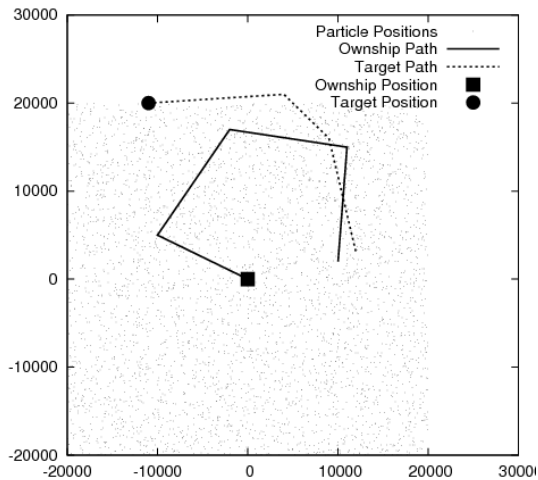
May 6, 2010

Motivating Example

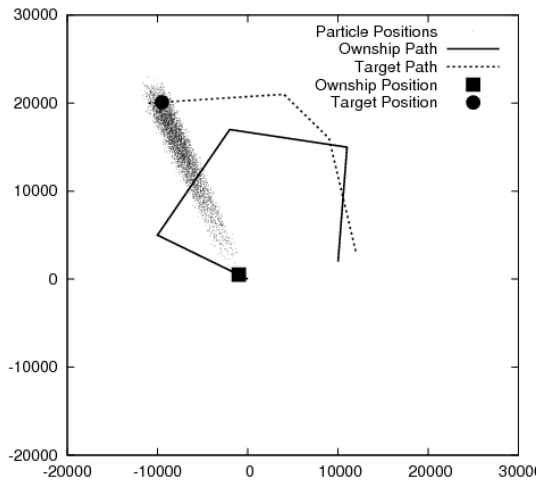
A submarine with a hydrophone (passive sonar) is following another ship using the direction of the sound from the ship's engine.

How can the series of bearing observations from the hydrophone be used to estimate the second ship's position and velocity?

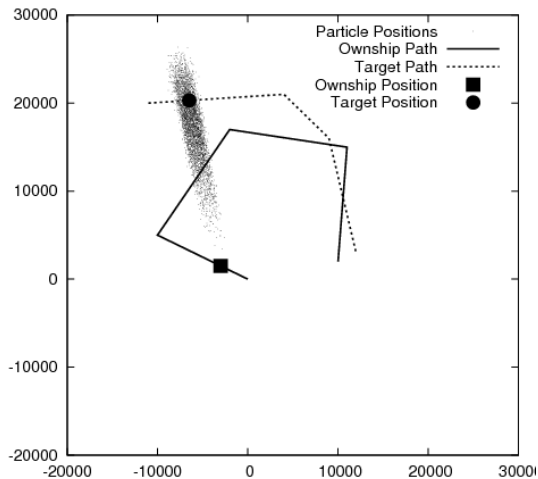
Motivating Example



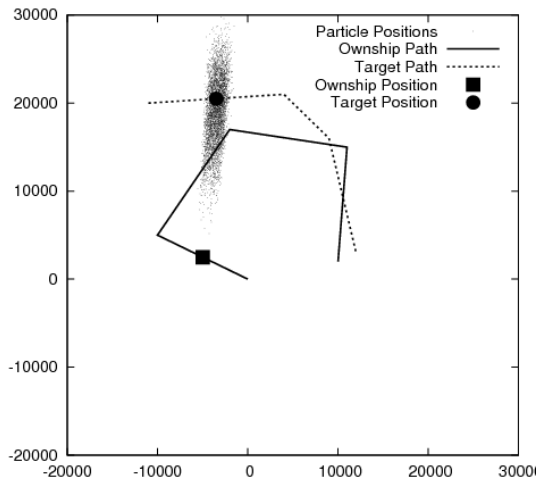
Motivating Example



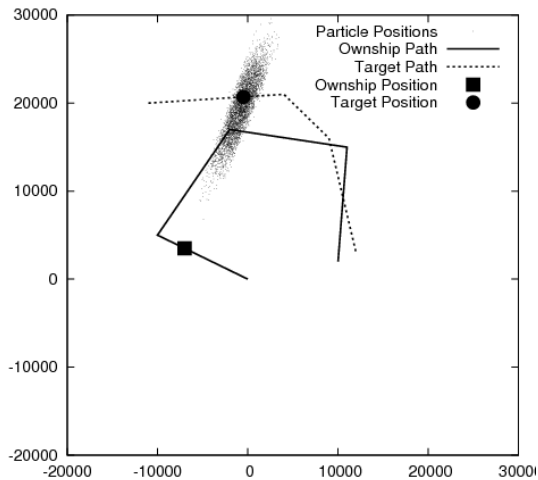
Motivating Example



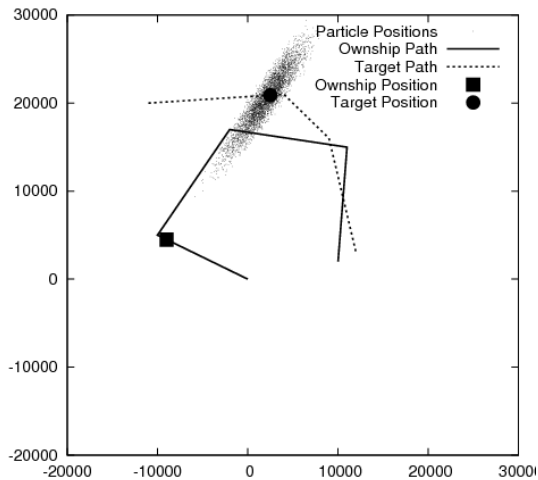
Motivating Example



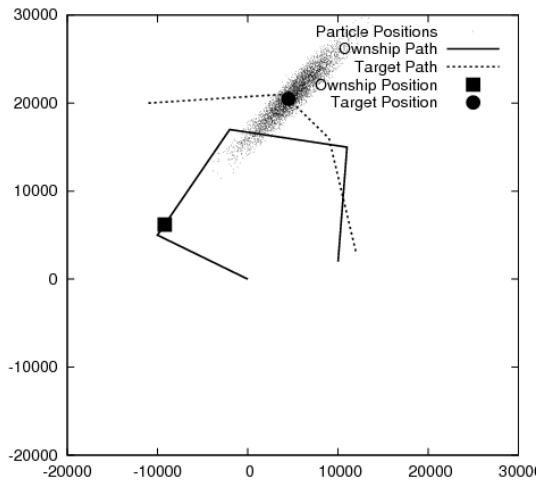
Motivating Example



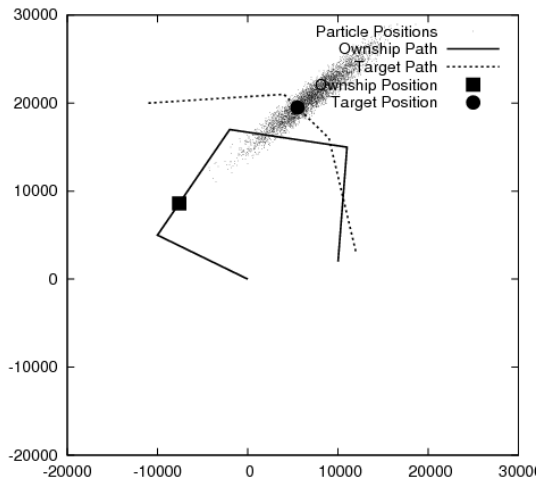
Motivating Example



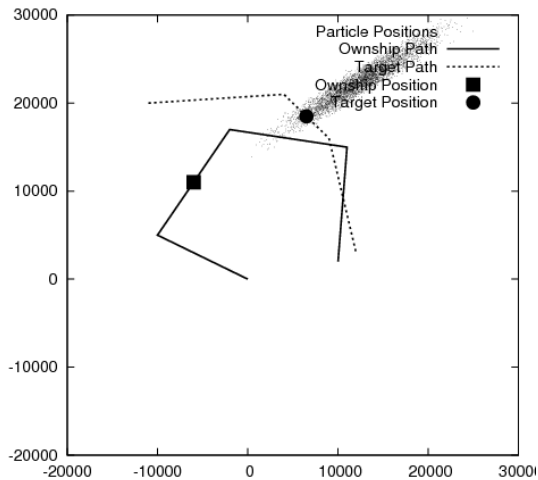
Motivating Example



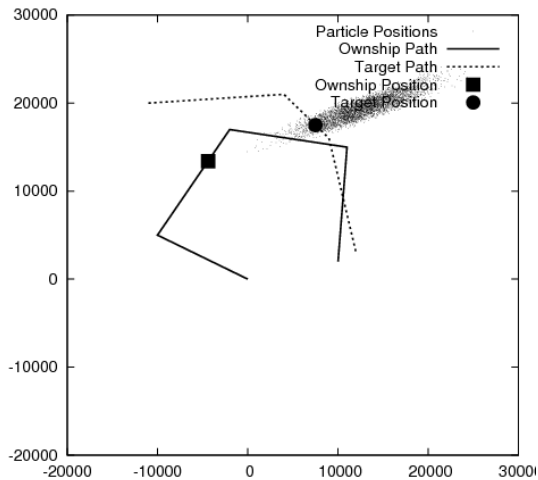
Motivating Example



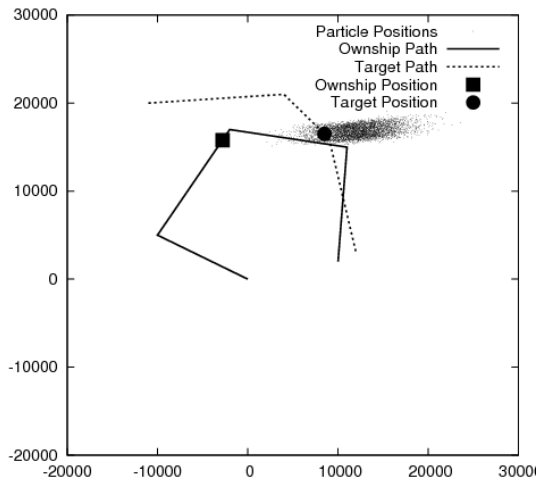
Motivating Example



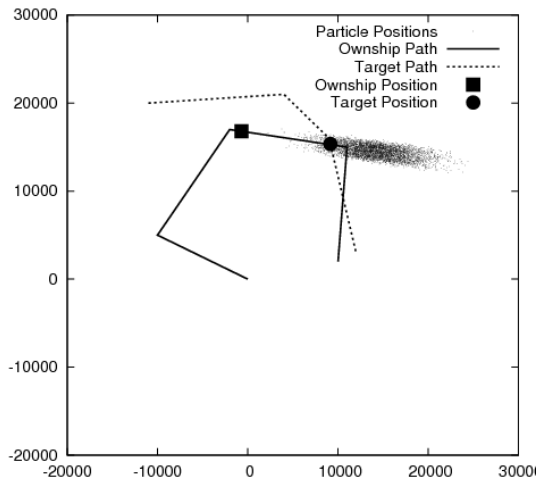
Motivating Example



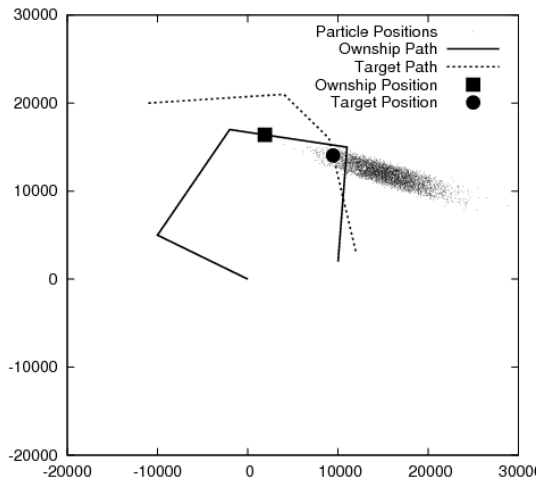
Motivating Example



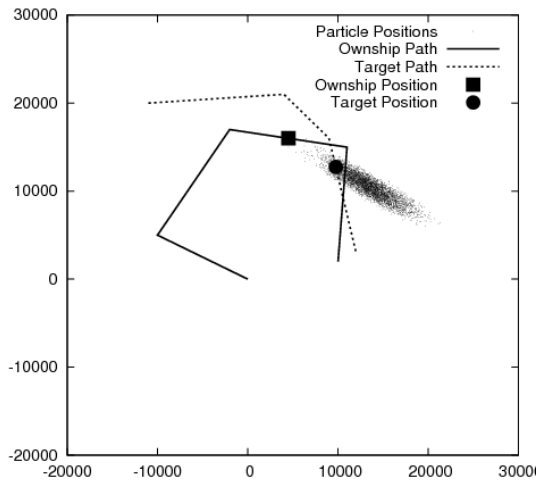
Motivating Example



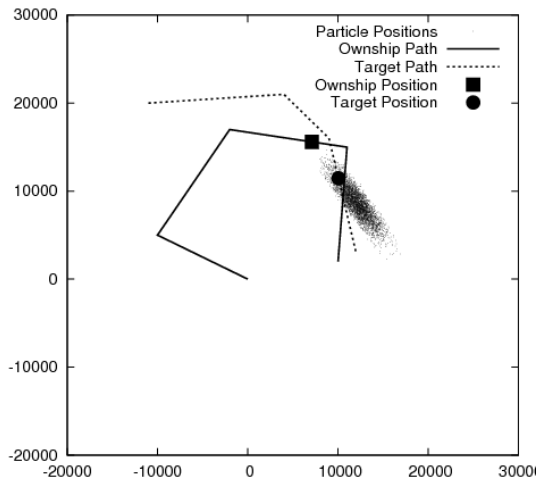
Motivating Example



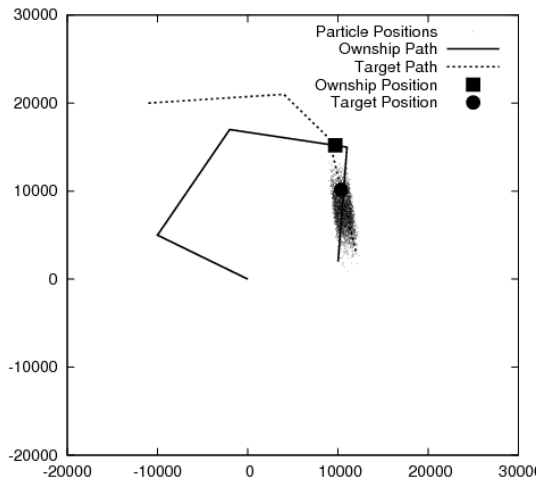
Motivating Example



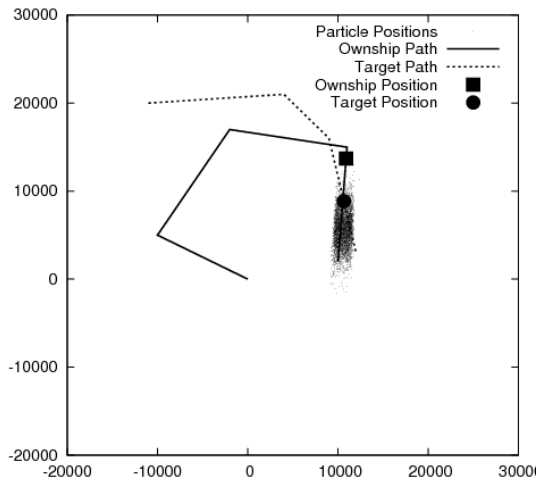
Motivating Example



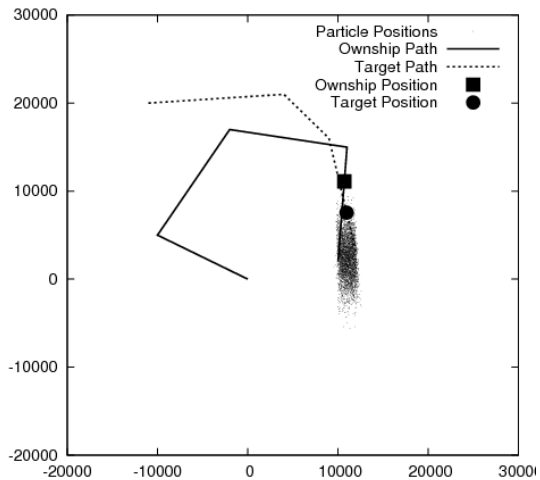
Motivating Example



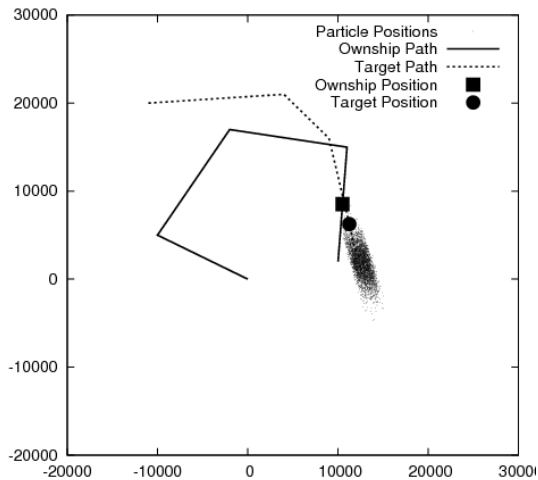
Motivating Example



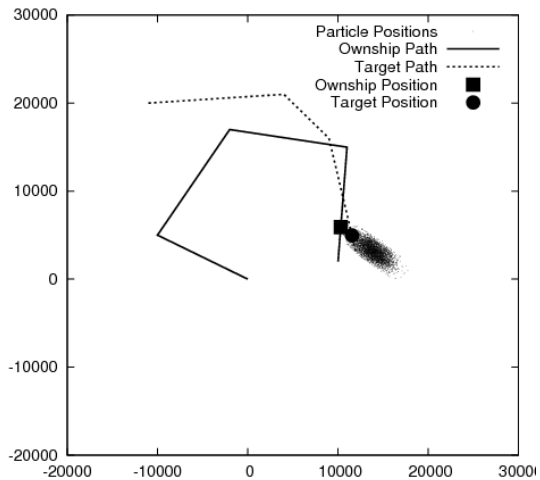
Motivating Example



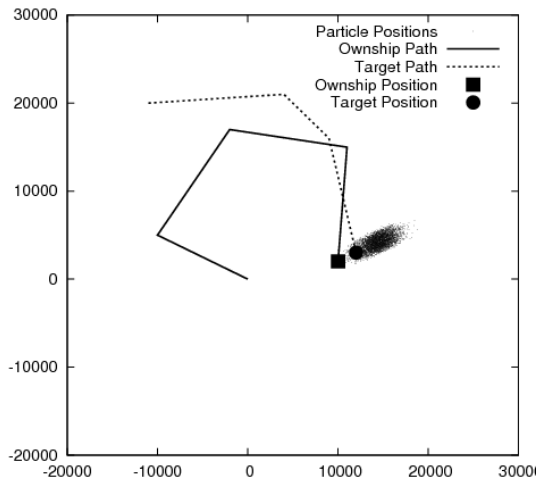
Motivating Example



Motivating Example



Motivating Example



Likelihood Function

$$L(y|x) = P(Y = y|X = x) \quad \text{for } x \in S \quad (1)$$

- ▶ X Random variable on S
- ▶ Y Random variable on measurement space H
- ▶ $P(\cdot|x)$ Probability density function on H
- ▶ $P(y|\cdot)$ Likelihood function relating X and Y

Likelihood Function

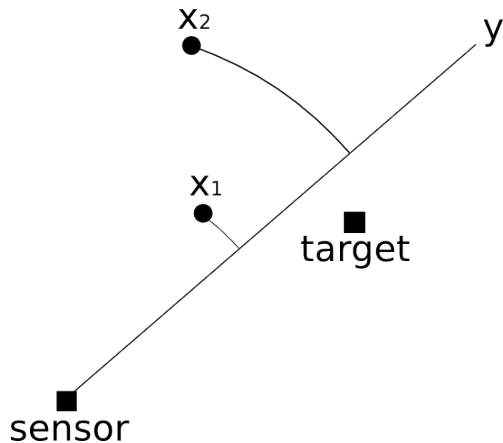
Why is $P(y|\cdot)$ called a "Likelihood Function"?

For $x_1, x_2 \in S$ if $L(y|x_1) > L(y|x_2)$ then:

The observation y is more likely to have come from a target with state x_1 than a target with state x_2

Likelihood Function

$$L(y|x_1) > L(y|x_2)$$



Bayes' Rule

Reverend Thomas Bayes (1702-1761):

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (2)$$

Bayesian Inference

$$P(x|y) = \frac{L(y|x)P(x)}{P(y)} = \frac{L(y|x)P(x)}{\int L(y|x)P(x) dx} \quad (3)$$

- ▶ Observation y is fixed, x represents possible target states
- ▶ $P(x)$ Prior probability density function on true target state
- ▶ $P(x|y)$ Posterior distribution given observation y

"Multiply each particle's weight by the likelihood function evaluated at the particle then normalize by the sum over all particle weights."

Prior Distribution

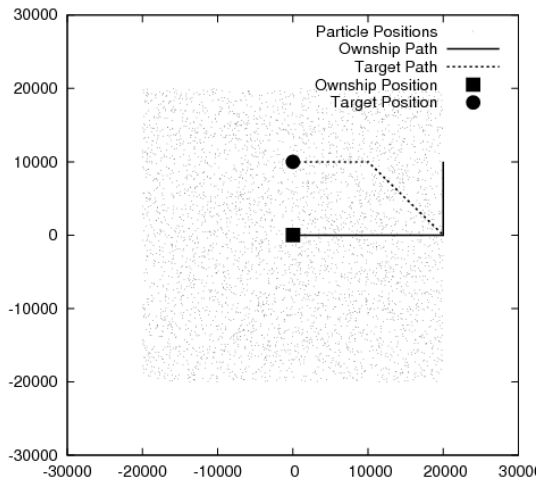


Figure: Prior Particle Position Distribution

Information Update

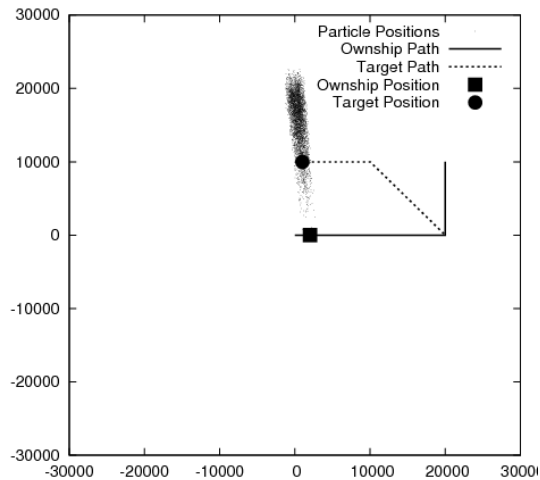


Figure: Posterior Particle Position Distribution after Azimuth Observation

Resampling

$$C = \frac{n}{\sum_{i=0}^n w_i} \quad (4)$$

- ▶ C Normalizing constant
- ▶ w_i Particle weights
- ▶ n Number of particles

Example w_i Array:

0.8	0.4	6.6	0.6	1.6
-----	-----	-----	-----	-----

$$C = \frac{5}{10} = 0.5$$

Resampling

$$\bar{w}_i = Cw_i \quad (5)$$

Example \bar{w}_i Array:

0.4	0.2	3.3	0.3	0.8
-----	-----	-----	-----	-----

Resampling

$$\hat{w}_i = \text{floor}\left(\sum_{j=0}^i \bar{w}_j\right) \quad (6)$$

Example \hat{w}_i Array:

Cumulative Sums:

0.4	0.6	3.9	4.2	5.0
-----	-----	-----	-----	-----

Floor:

0	0	3	4	5
---	---	---	---	---

Parallel Summation

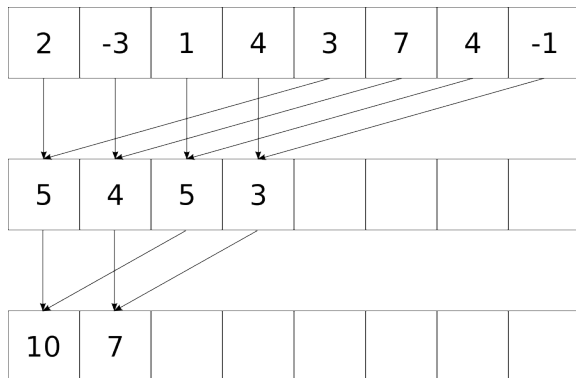


Figure: Prior Particle Position Distribution

Parallel Summation

```
__shared__ float s_shared[ blockDim.x ];
int tid = threadIdx.x;
unsigned int offset = blockDim.x / 2;

while ( offset > 0 )
{
    if ( tid < offset )
    {
        s_shared[ tid ] += s_shared[ tid + offset ];
    }

    offset = offset >> 1; // divide offset by 2

    __syncthreads();
}
```

Parallel Summation Improved

...

```
while ( offset > 32 )  
{  
    ...  
}
```

```
if ( tid < 32 )  
{  
    s_shared[ tid ] += s_shared[ tid + 32 ];  
    s_shared[ tid ] += s_shared[ tid + 16 ];  
    s_shared[ tid ] += s_shared[ tid + 8 ];  
    s_shared[ tid ] += s_shared[ tid + 4 ];  
    s_shared[ tid ] += s_shared[ tid + 2 ];  
    s_shared[ tid ] += s_shared[ tid + 1 ];  
}
```

Resampling Results

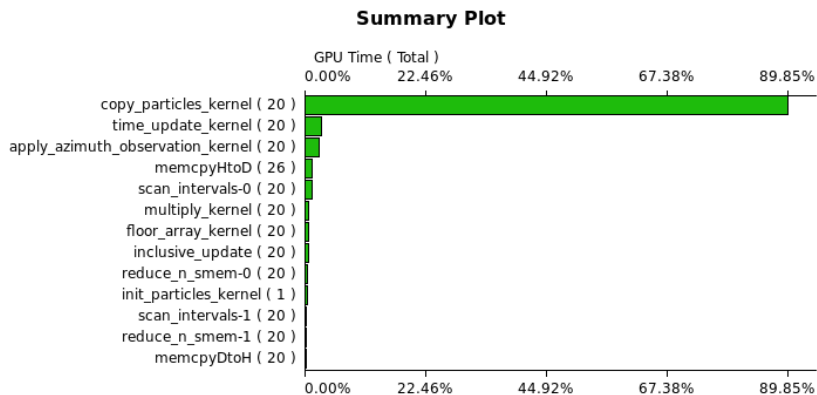


Figure: CUDA Visual Profiler Version 1 Results

Improved Resampling Results

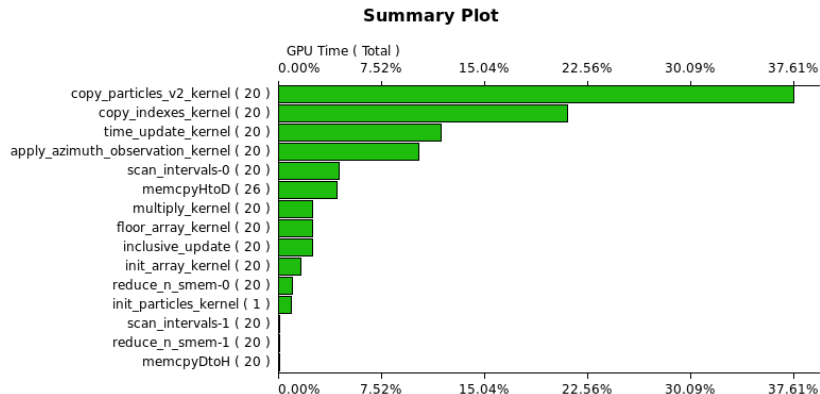
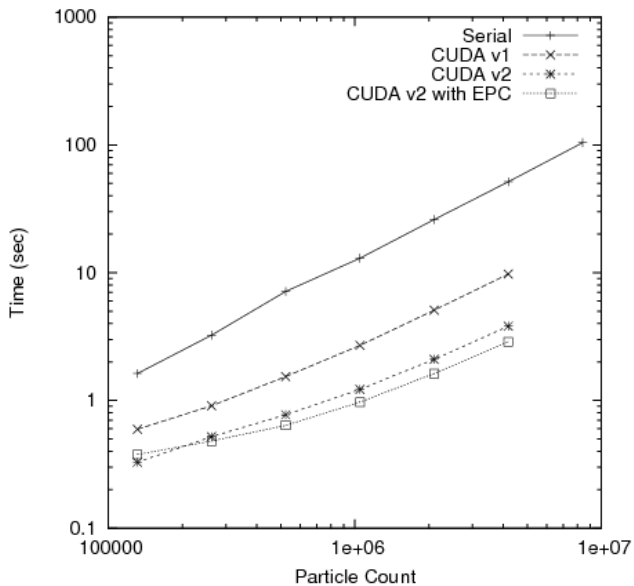
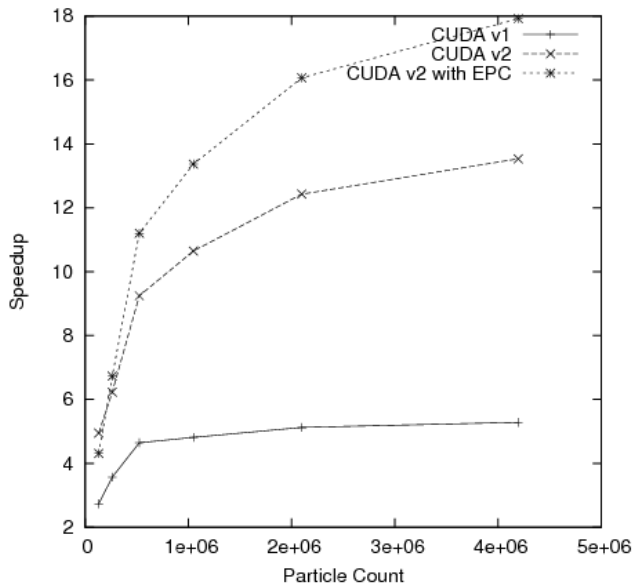


Figure: CUDA Visual Profiler Version 2 Results

Performance



Performance



Lessons Learned

- ▶ CUDA optimization is *hard*
 - ▶ Uncoalesced memory access
 - ▶ Shared memory bank conflicts
 - ▶ Slow operators (like modulus)
- ▶ However, even codes with the above problems can achieve significant speedup
- ▶ CUDA Visual Profiler
 - ▶ useful and user friendly
- ▶ thrust
 - ▶ fast and well documented

Backup Slides

Backup Slides

Effective Particle Count

$$\bar{w}_i = \frac{w_i}{\sum_{i=1}^n w_i} \quad (7)$$

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^n \bar{w}_i^2} \quad (8)$$

For $\bar{w}_i = \frac{1}{n}$ $N_{\text{eff}} = \frac{1}{n \frac{1}{n^2}} = n$

For $\bar{w}_i = \begin{cases} 1 & \text{for } i = 0 \\ 0 & \text{otherwise} \end{cases}$ $N_{\text{eff}} = \frac{1}{1} = 1$