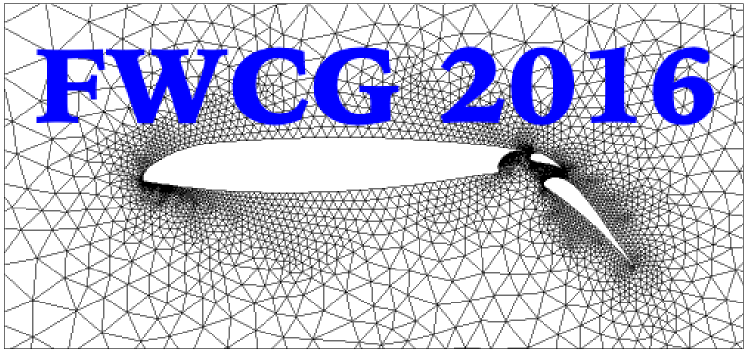


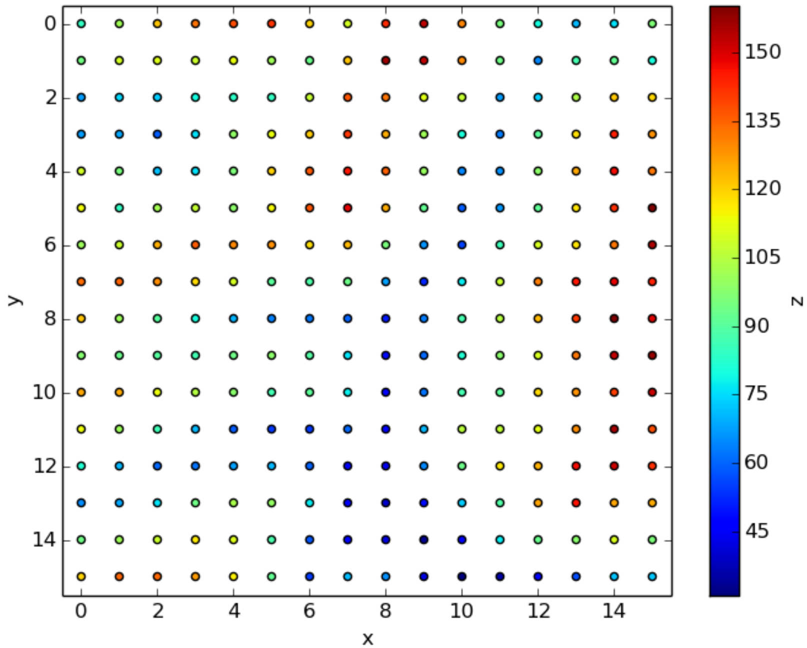
October 27-28, 2016
CUNY Graduate Center
New York, NY

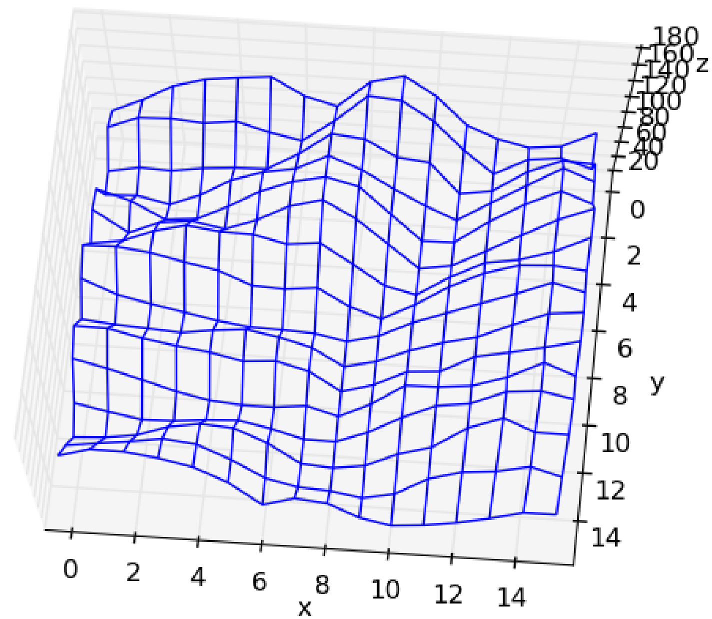
Computing approximate horizons on a GPU

Wenli Li, W. Randolph Franklin, Salles V. G. Magalhães
Rensselaer Polytechnic Institute, Troy, NY, USA
liw9@rpi.edu, mail@wrfranklin.org, vianas2@rpi.edu

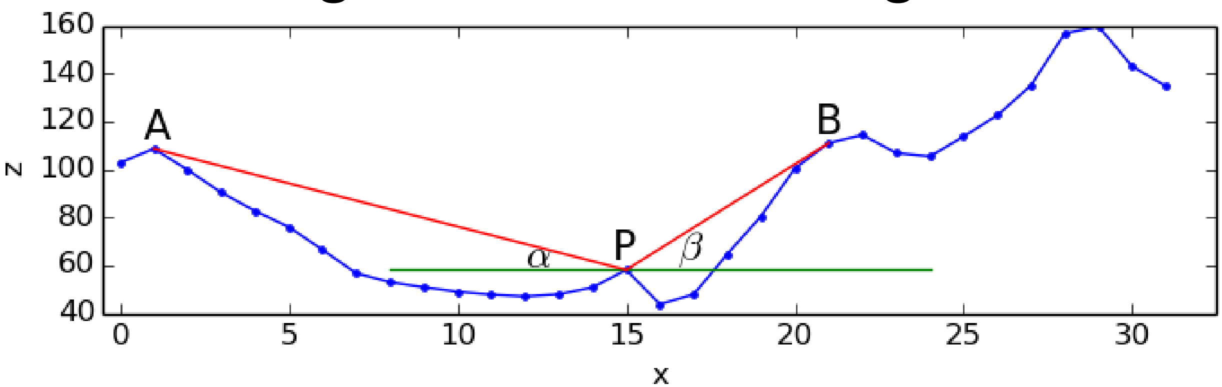
Introduction

- Digital Elevation Model (DEM)





- Horizon: largest elevation angle

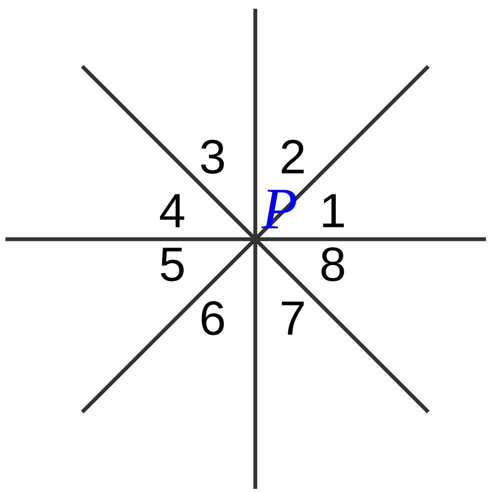


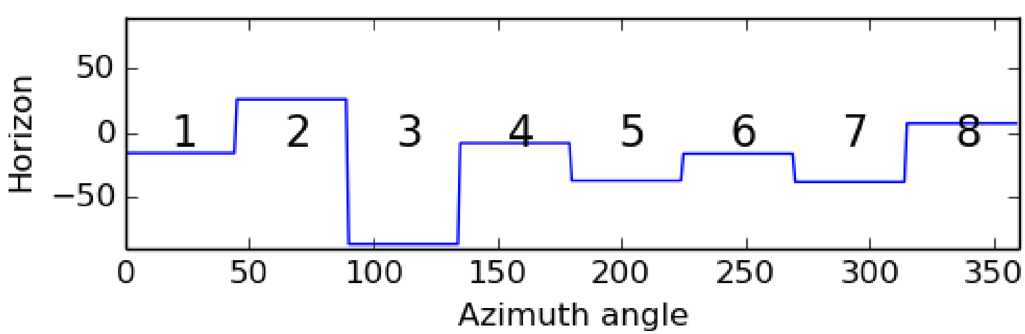
References

- [Ste98] A. J. Stewart. Fast horizon computation at all points of a terrain with visibility and shading applications. IEEE Transactions on Visualization and Computer Graphics, 4(1):82–93, Mar. 1998.
- [TRZ11] S. Tabik, L. F. Romero, and E. L. Zapata. High-performance three-horizon composition algorithm for large-scale terrains. International Journal of Geographical Information Science, 25(4):541–555, Apr. 2011.
- [BH86] J. Barnes and P. Hut. A hierarchical O(N log N) force-calculation algorithm. Nature, 324:446–449, Dec. 1986.

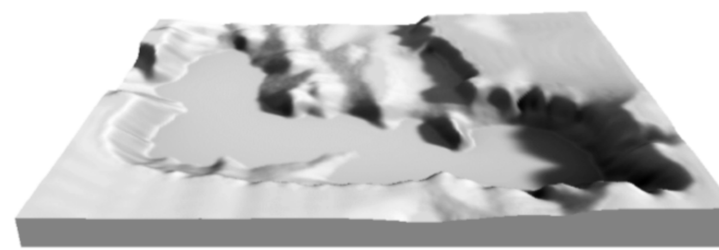
Motivation

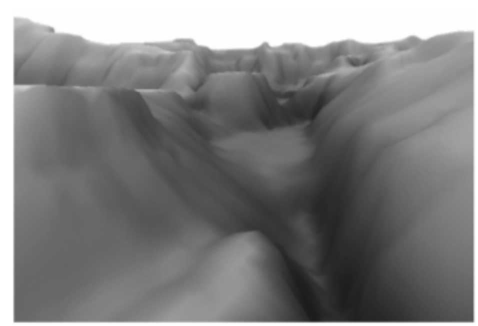
- Approximate horizon: constant in each sector

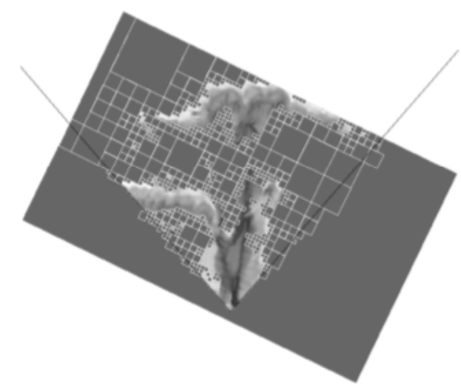




- Applications: shading, visibility [Ste98], solar irradiance [TRZ11]

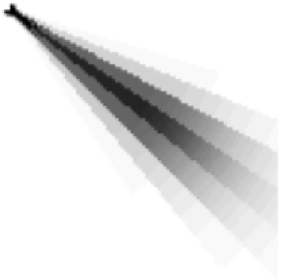






Motivation

- Stewart's algorithm: O(*sn*log^2(*n*))
  - *s* sectors and *n* points
  - Approximate the horizon by the largest elevation angle in each sector
  - Parallel for the sectors and sequential for each sector
- Tabik et al.'s algorithm: dividing a terrain into blocks and using Stewart's algorithm
  - Compute “near” horizons for each block
  - Compute “far” horizons on a lower-resolution terrain
  - Parallel for the blocks and the sectors of a block



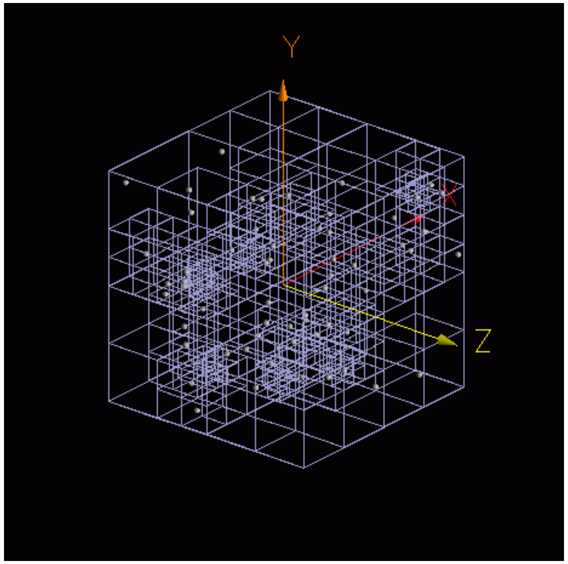
(a) New (64 sectors)

(d) CMS (64 sectors)

Motivation

- Barnes-Hut algorithm: O(*n*log(*n*))
  - N-body simulation
  - Divide the space in an octree and store the center of mass and total mass in each internal node
  - Recursively traverse the tree to approximate the gravitational force on a body
  - Treat an internal node as a single body if $w/d < \theta$
    - *w*: width of the node
    - *d*: distance between the body and the node's center of mass

http://insidehpc.com/2015/05/direct-n-body-simulation/



http://15418.courses.cs.cmu.edu/spring2013/article/18

GPU-friendly algorithms

- Brute-force algorithm: $O(n^2)$

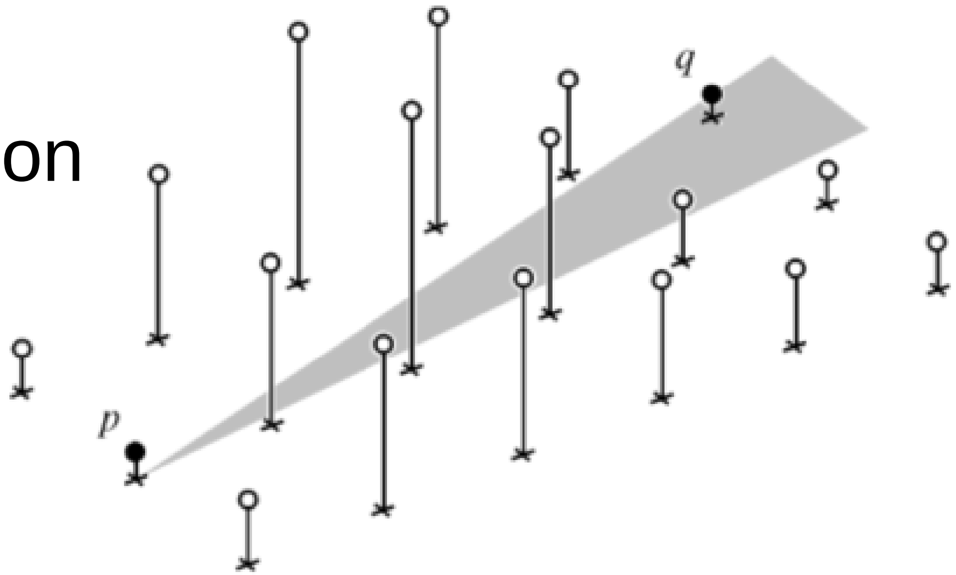
  foreach point *p* **do**

      foreach point *q* **do**

          find the sector *s* of *p* containing *q*;

          update the horizon of *p* in *s* using *q*;

- Narrow sectors [Ste98]
  - Underestimate the horizon



GPU-friendly algorithms

- Narrow sectors
  - Stewart's solution: checking about $s/2\pi$ bordering points on each side
  - Our solution: checking $s/2\pi$ points along the bisector
- Brute-force algorithm

foreach point *p* **do**

  foreach sector *s* **do**

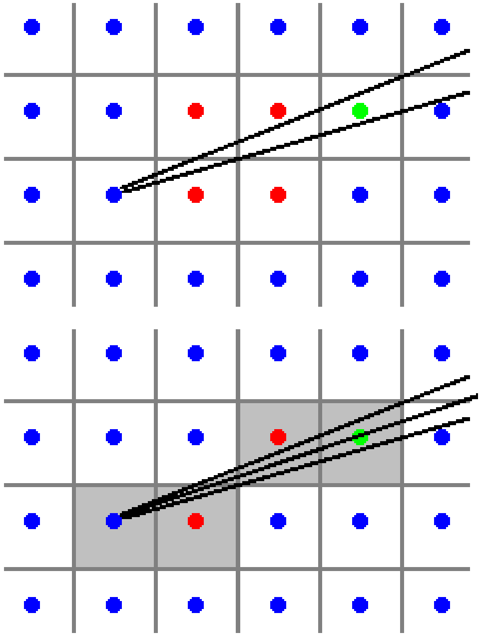
    foreach point *q* of a few points along the bisector **do**

      update the horizon of *p* in *s* using *q*;

  foreach point *q* **do**

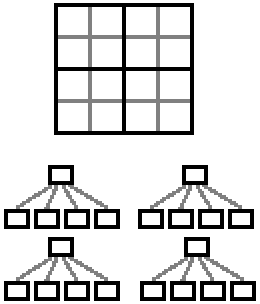
    find the sector *s* of *p* containing *q*;

    update the horizon of *p* in *s* using *q*;



GPU-friendly algorithms

- Quadtree-forest algorithm: O(*n*log(*n*))
  - Like a 2D Barnes-Hut algorithm
  - Divide a terrain into blocks and build a largest-value quadtree for each block
  - Recursively traverses each quadtree to compute a horizon
  - Use a fixed-sized stack to simulate recursion on the GPU
  - Use a quadtree-forest instead of a quadtree
    - First few levels of a quadtree are not treated as points
    - Higher trees require a larger stack and more stack operations



GPU-friendly algorithms

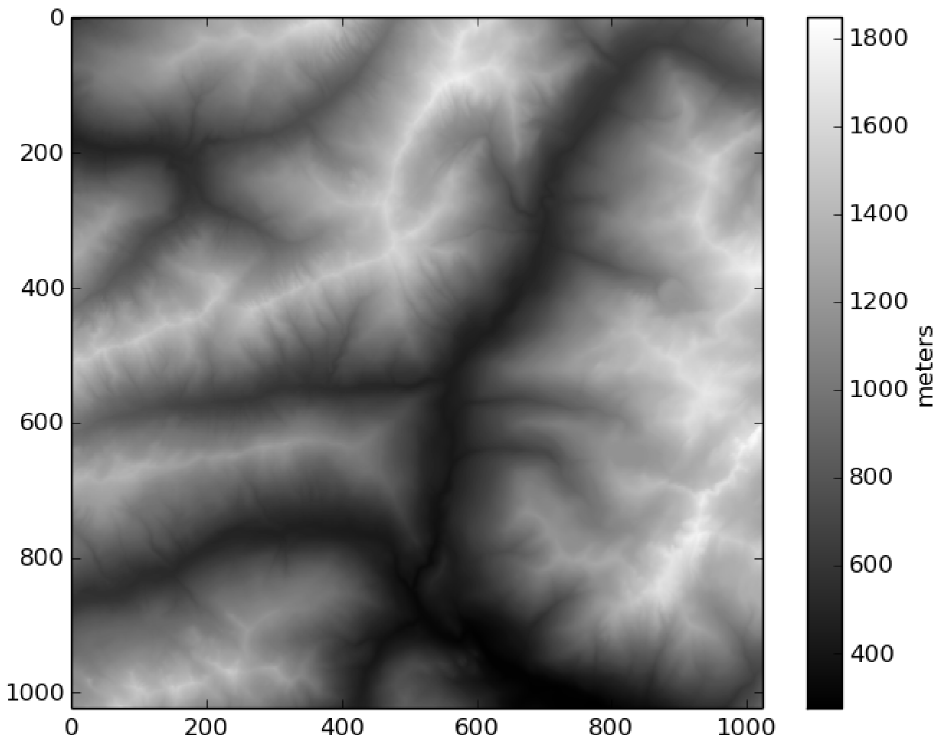
- Quadtree-forest algorithm

```
divide the terrain into blocks and build a quadtree for each block;
foreach point p do
    foreach sector s do
        foreach point q of a few points along the bisector do
            update the horizon of p in s using q;
    foreach quadtree t do
        push the root of t on stack;
        while the stack is not empty do
            pull a node n from stack;
            foreach child c of n do
                if c is not a leaf and w/d > θ then
                    push c on stack;
                else
                    find the sector s of p containing c;
                    update the horizon of p in s using c;
```

```
if n is not a leaf and w/d > θ then
    foreach child c of n do
        push c on stack;
else
    find the sector s of p containing n;
    update the horizon p in s using n;
```

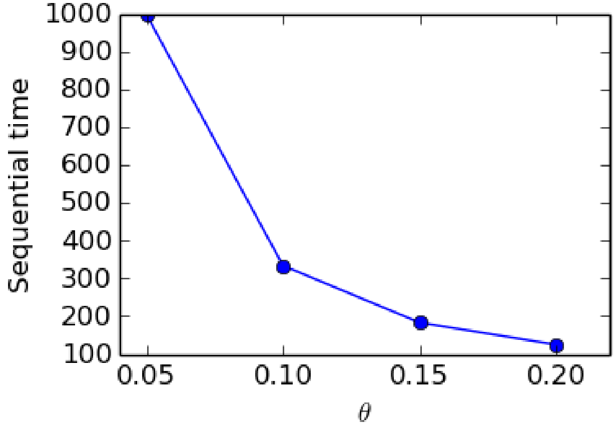
Results

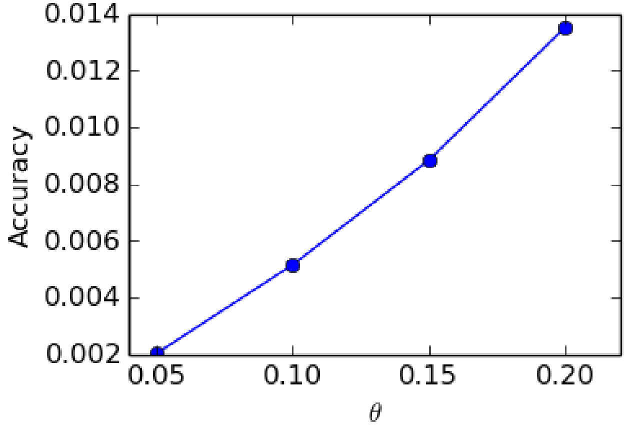
- Implementations
  - Sequential programs on CPU
  - CUDA programs on GPU
- Hardware
  - Intel Xeon E5-2660 v4 CPU
  - NVIDIA GeForce GTX 1080 GPU
- Dataset: 1024x1024 DEM
  - 10-meter resolution
  - [274.7, 1846.8]-meter range
  - 64 sectors



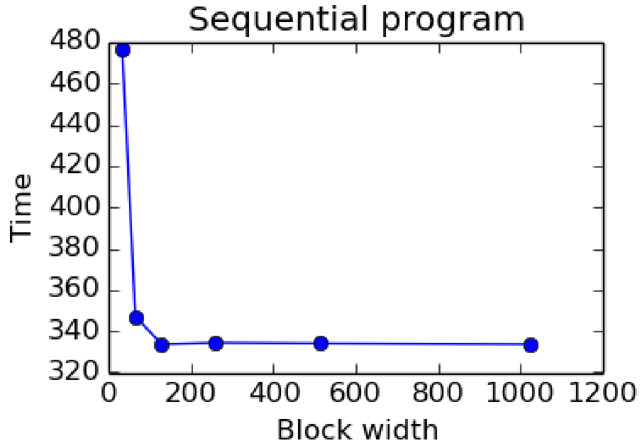
Results

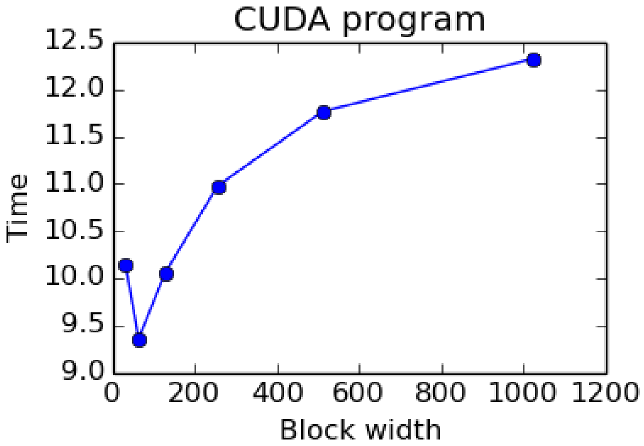
- θ, running time, and accuracy of the quadtree-forest algorithm





- Block width and running time of the quadtree-forest algorithm (θ = 0.1)





Results

- Running time and relative speedup of the programs
  - Quadtree-forest algorithm: $\theta = 0.1$
    - Sequential program: block width = 1024
    - CUDA program: block width = 64

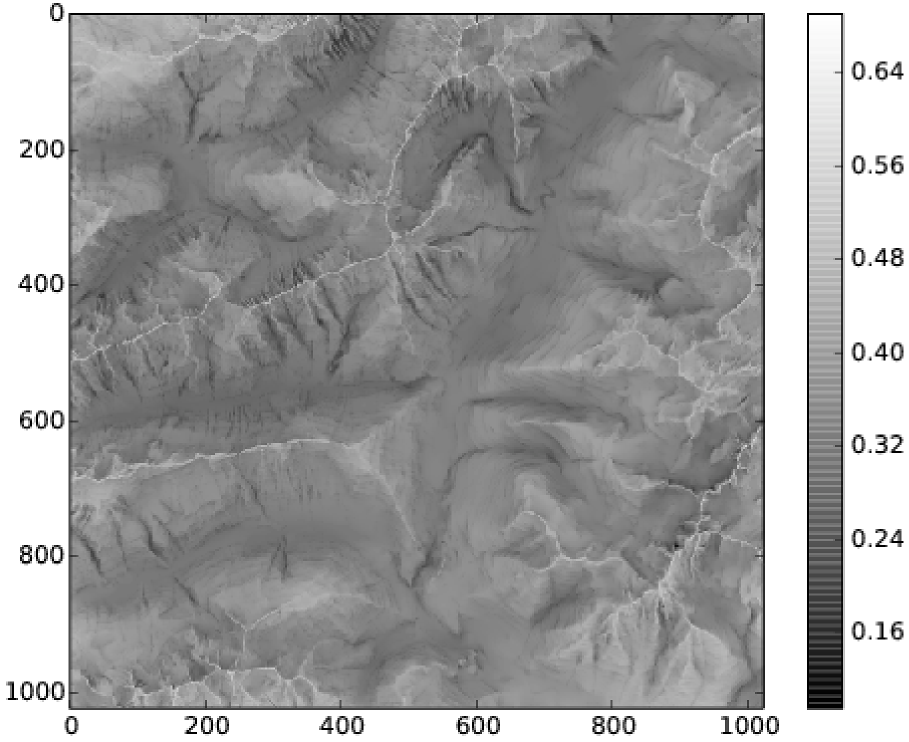
Algorithm	Sequential time	CUDA time	Speedup
Brute force	55278	984	56
Quadtree forest	334	9	36

Conclusions

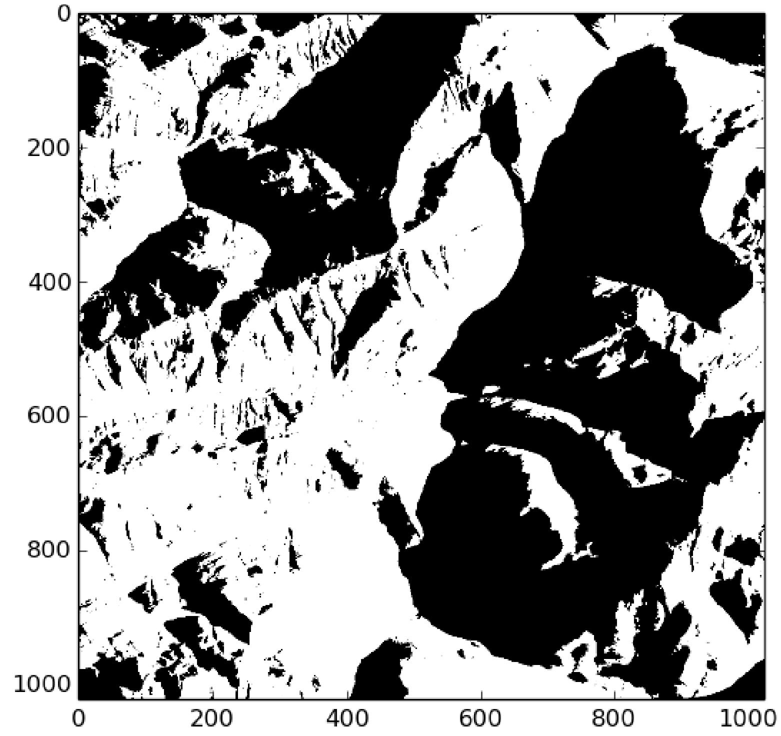
- Conclusions
  - The quadtree-forest algorithm is asymptotically faster and more suitable for the GPU
  - The result of the quadtree-forest algorithm is very close to that of the brute-force algorithm
- Future work
  - O(*n*) algorithm?
    - W. Dehnen. A hierarchical O(N) force-calculation algorithm. *Journal of Computational Physics*, 179(1):27–42, Jun. 2002.
  - Applications of approximate horizons

Thank you

- Visible sky area



- Casting shadows



This research was partially supported by NSF grant IIS-1117277 and CAPES (Ciência sem Fronteiras).