Eurographics 2013

9.5.2013

LOW-COMPLEXITY INTERVISIBILITY IN HEIGHT FIELDS

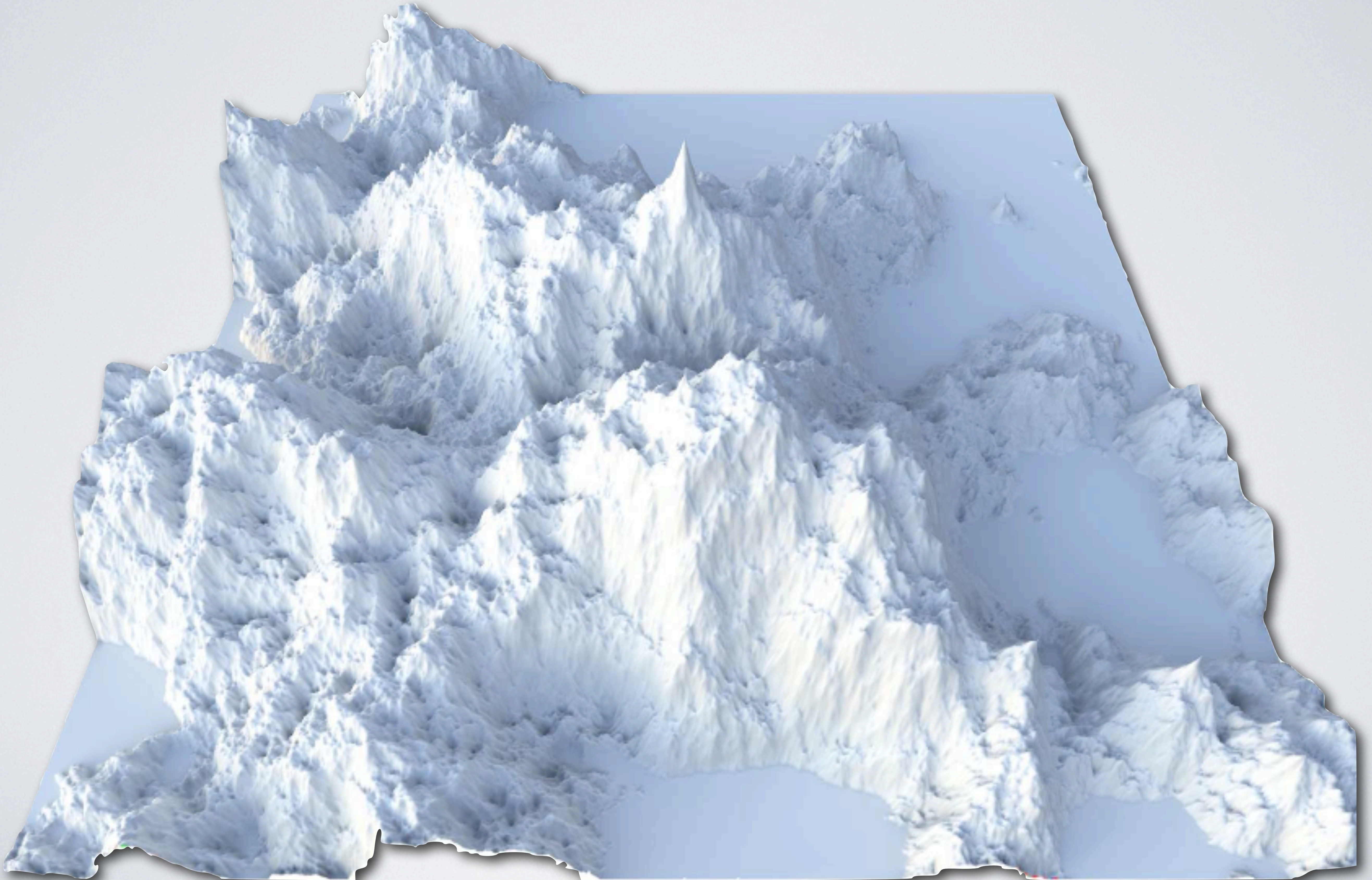
Ville Timonen

Turku Centre for Computer Science
Åbo Akademi University

CONTENTS

1 PROBLEM DESCRIPTION

Here's a height field...



1 PROBLEM DESCRIPTION

Here's another...

1 PROBLEM DESCRIPTION

Even the depth buffer can be interpreted as one...

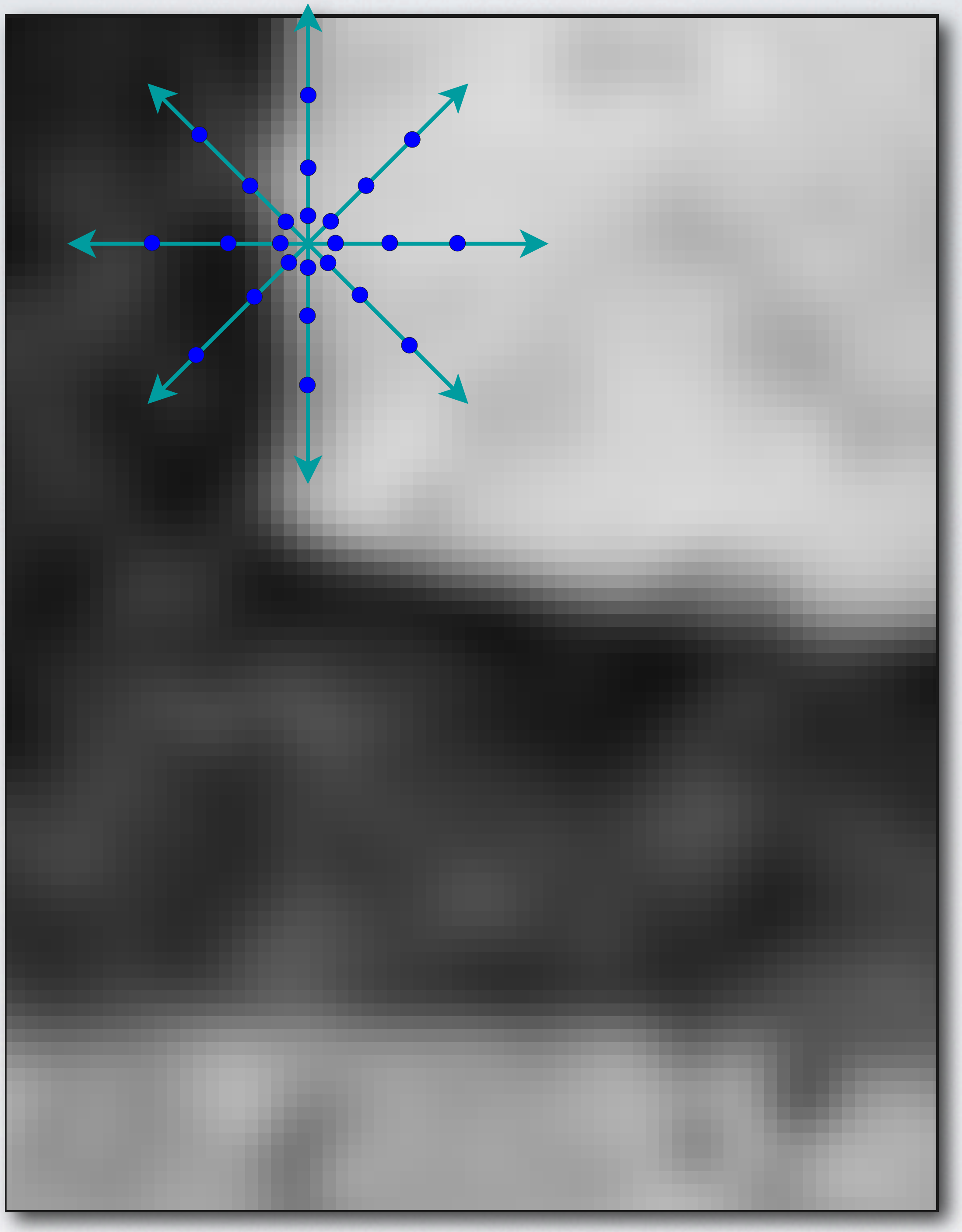
1 PROBLEM DESCRIPTION

Intervisibility is..

- Determining which points in the height field are visible to each height field point
- Can be used to:
  - Find good coverage points for radio towers
  - Plan Mars rover paths that have high camera visibility
  - etc...
- In graphics rendering: geometry culling, lighting
- Lighting: indirect illumination and self-illuminating surfaces

1 PROBLEM DESCRIPTION

Decompose the 2.5D problem into K 1.5D problems



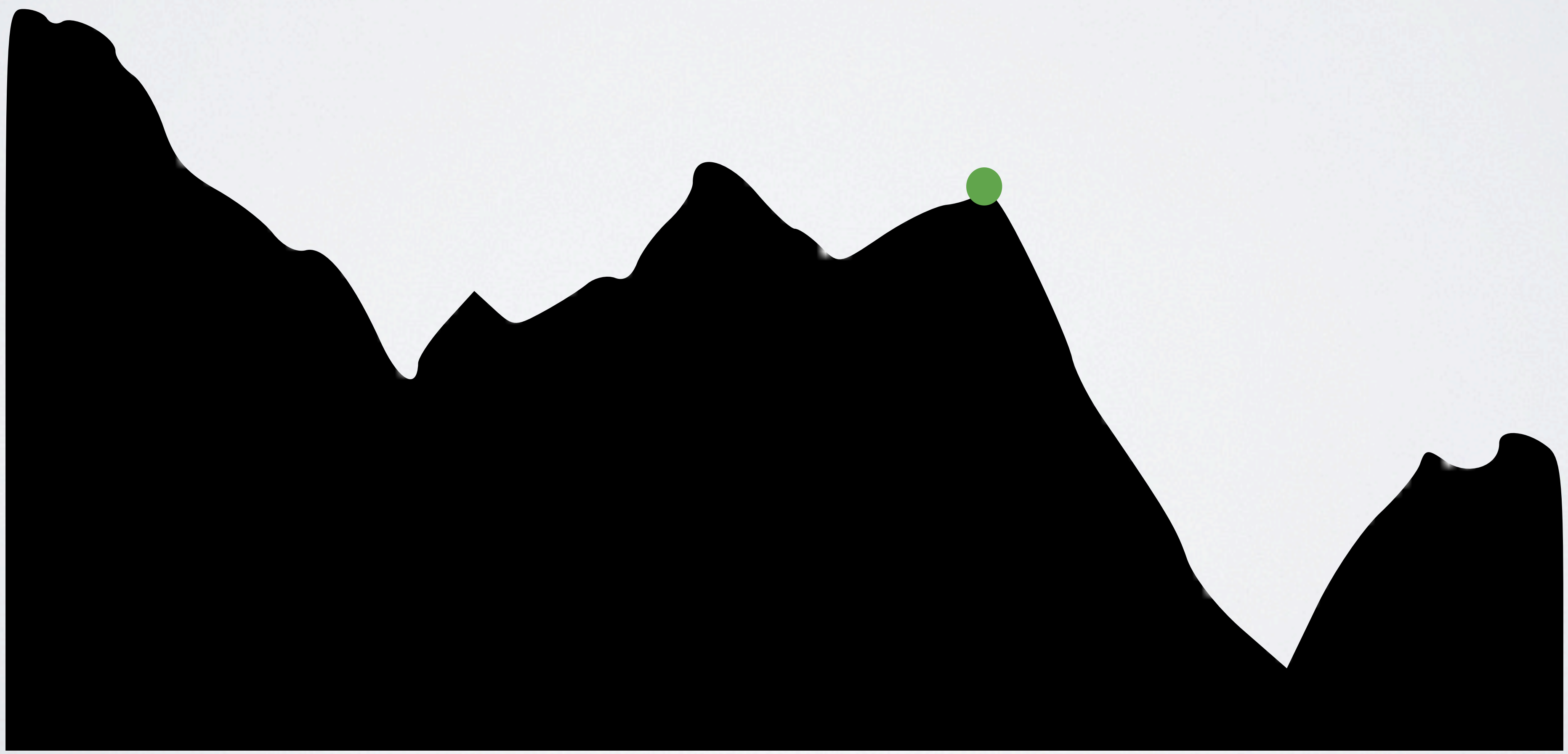
Here K = 8

1 PROBLEM DESCRIPTION

One 1.5D slice from the fractal terrain:

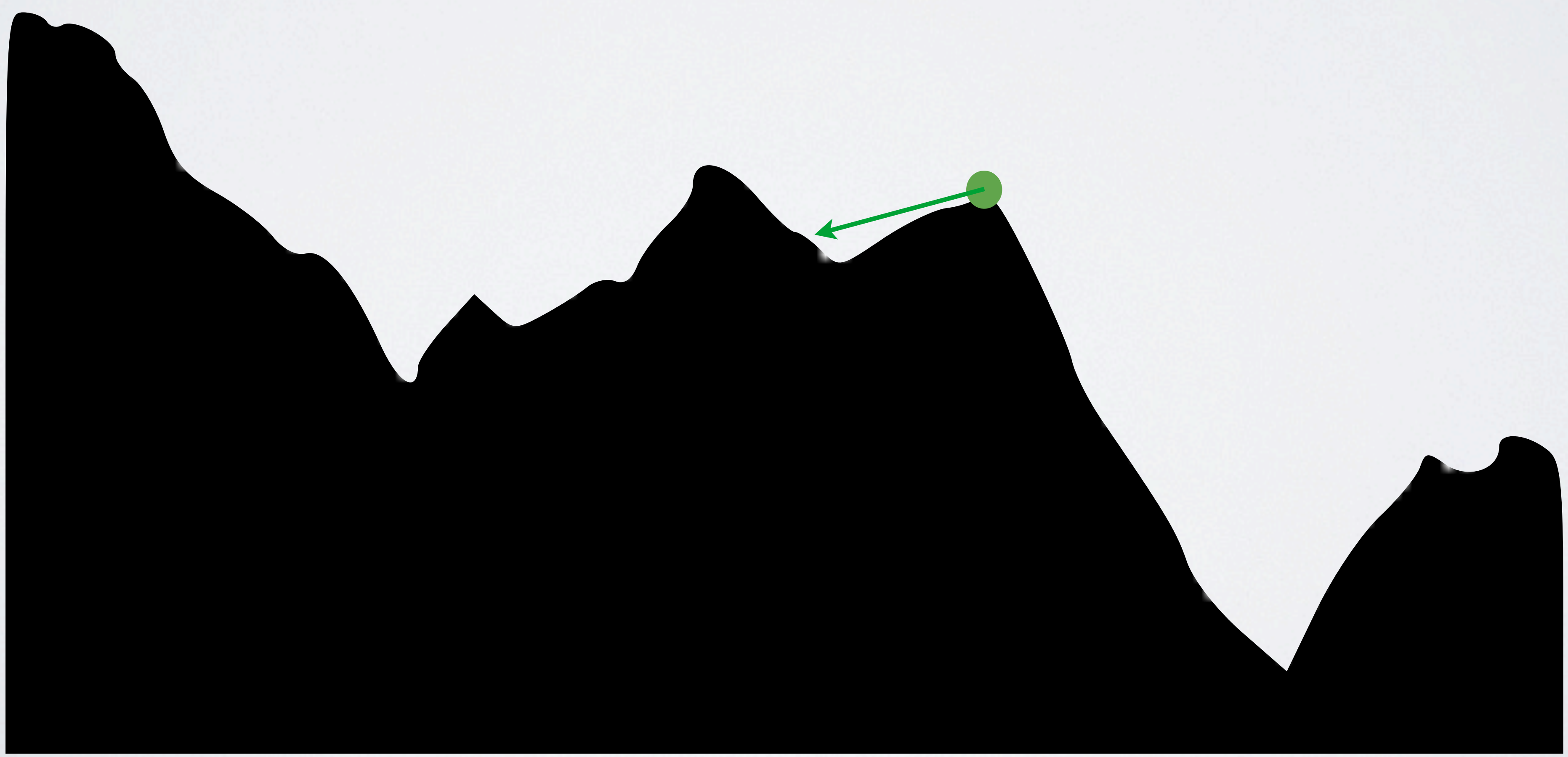
I PREVIOUS SOLUTION

From each point, step through the slice



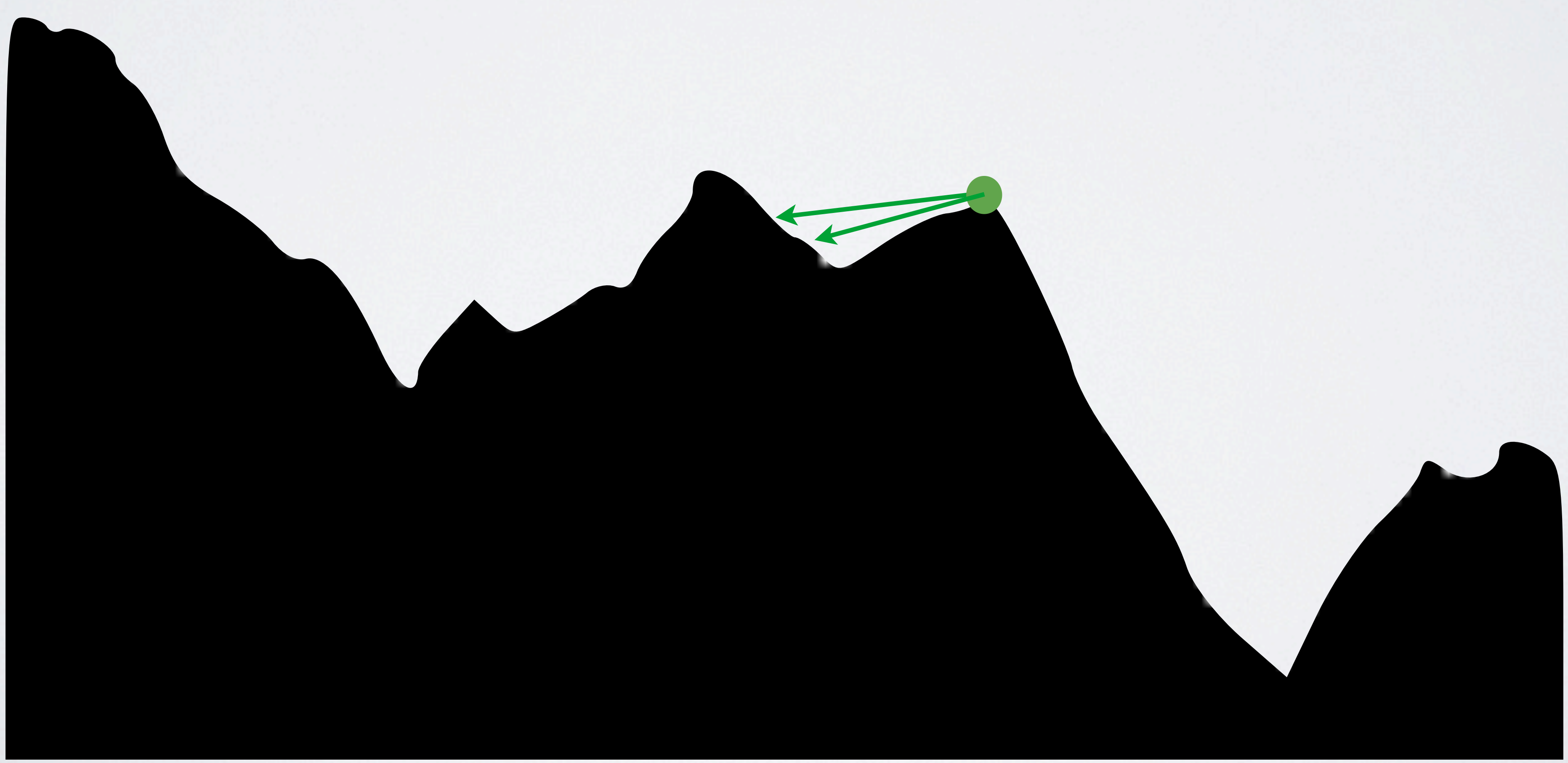
I PREVIOUS SOLUTION

From each point, step through the slice



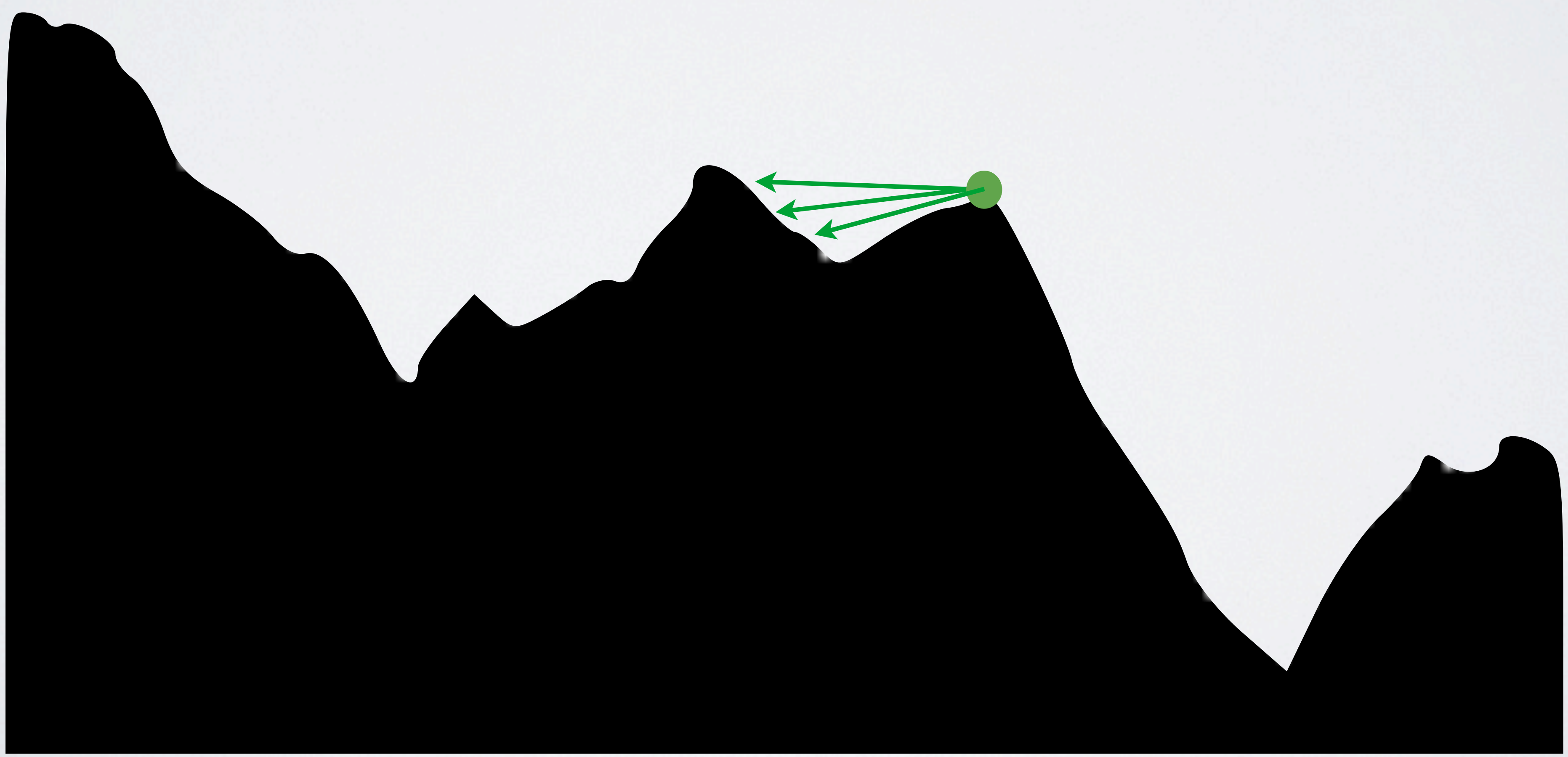
I PREVIOUS SOLUTION

From each point, step through the slice



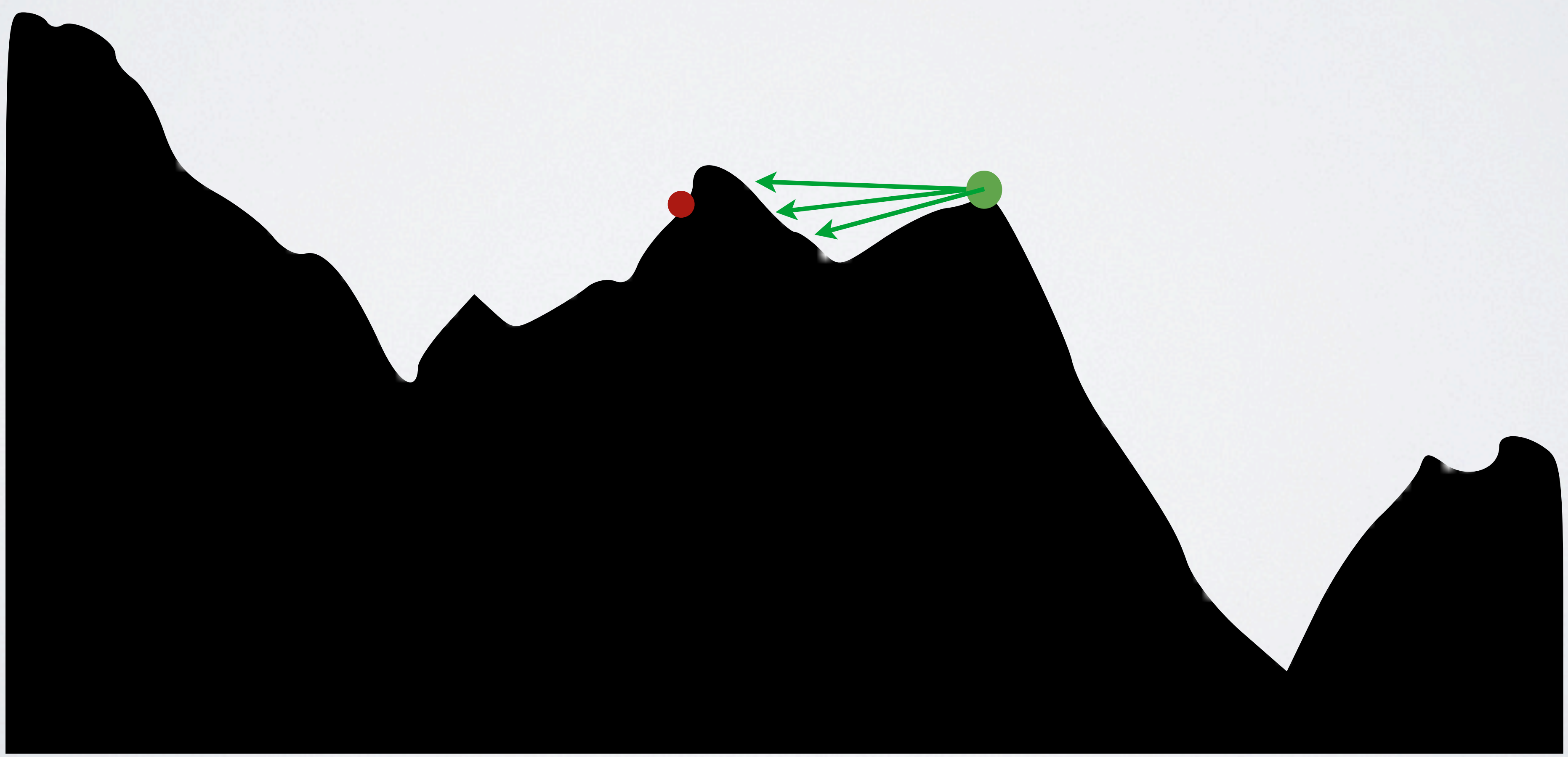
1 PREVIOUS SOLUTION

From each point, step through the slice



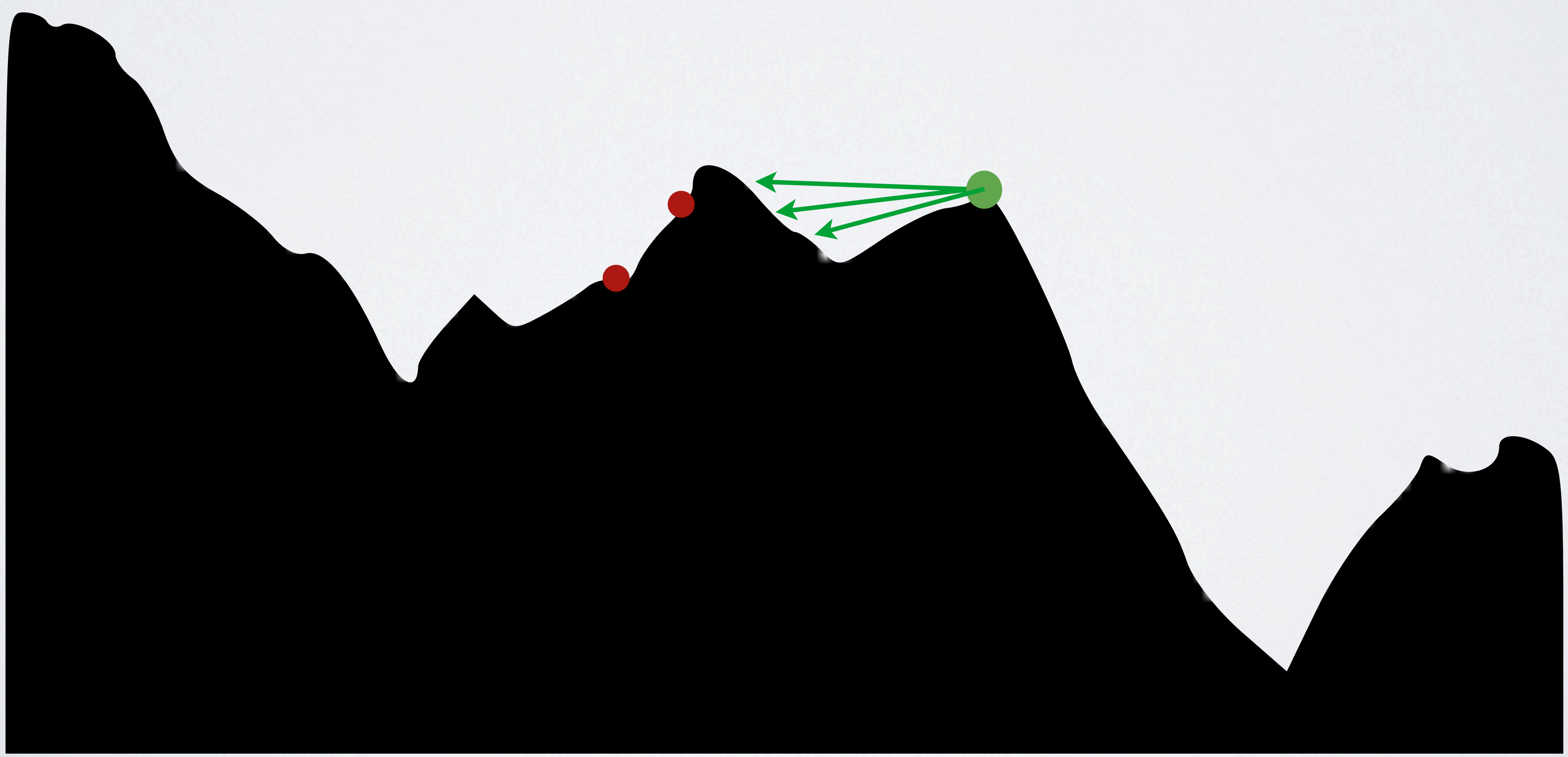
I PREVIOUS SOLUTION

From each point, step through the slice



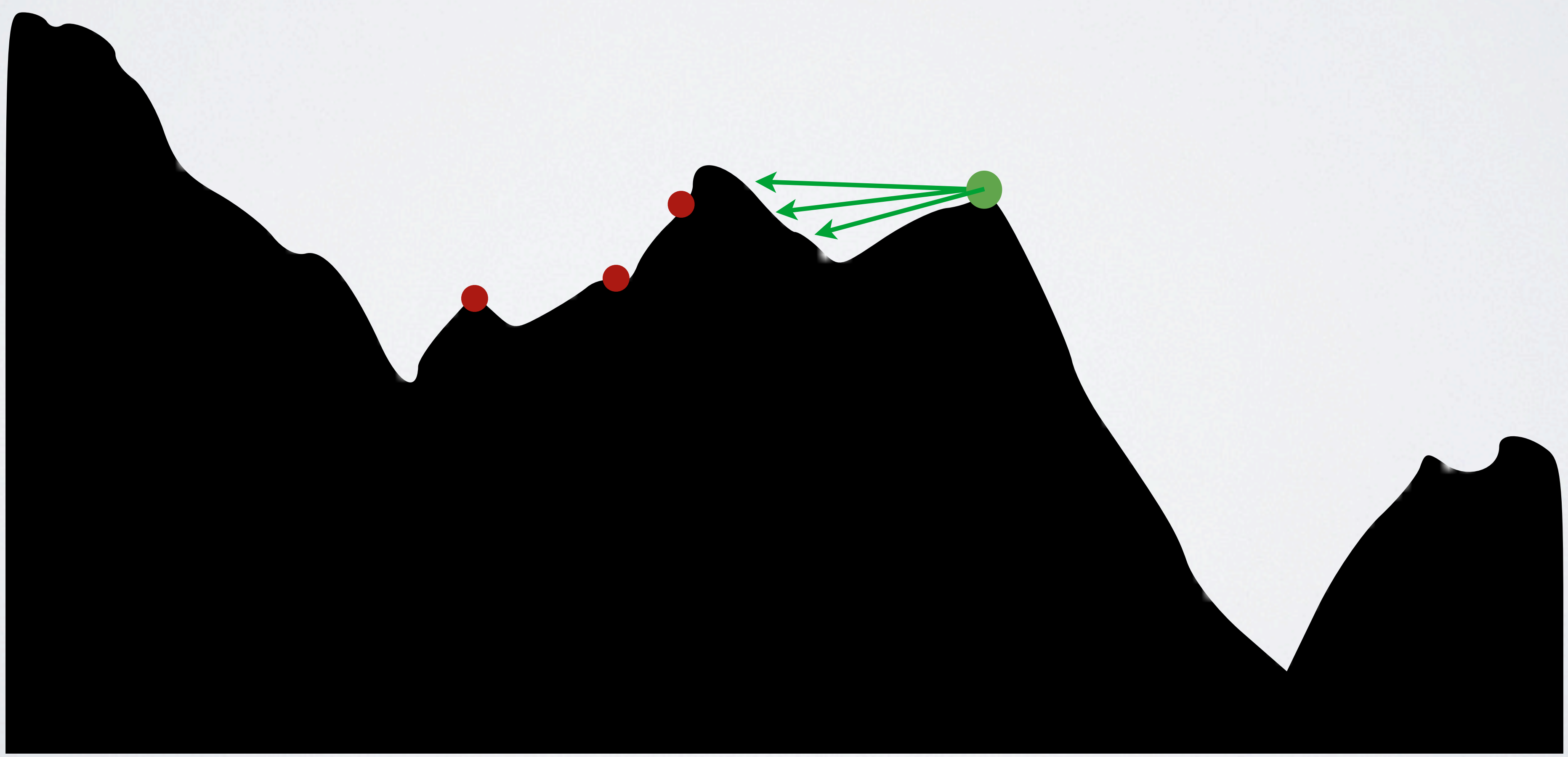
I PREVIOUS SOLUTION

From each point, step through the slice



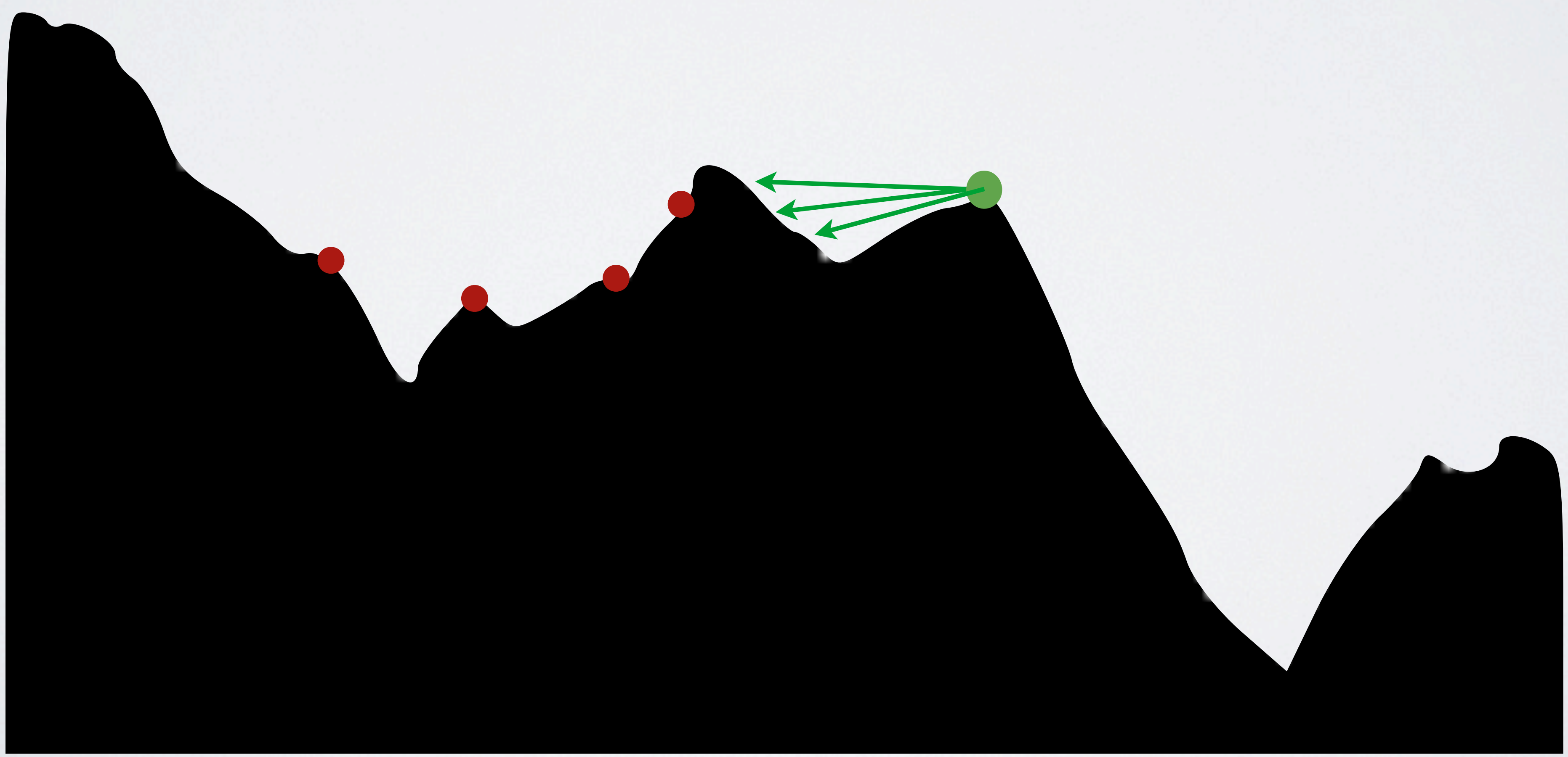
1 PREVIOUS SOLUTION

From each point, step through the slice



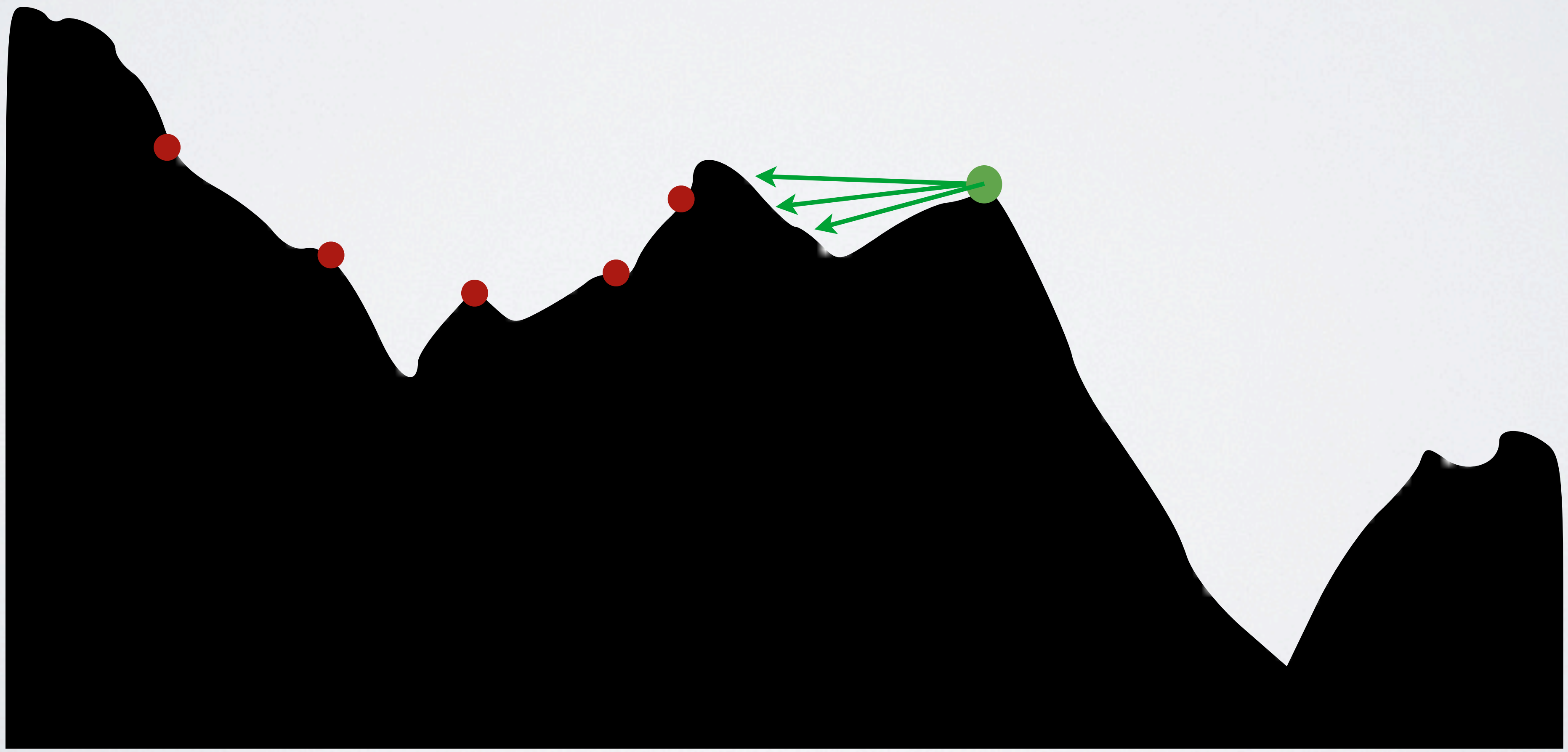
I PREVIOUS SOLUTION

From each point, step through the slice



1 PREVIOUS SOLUTION

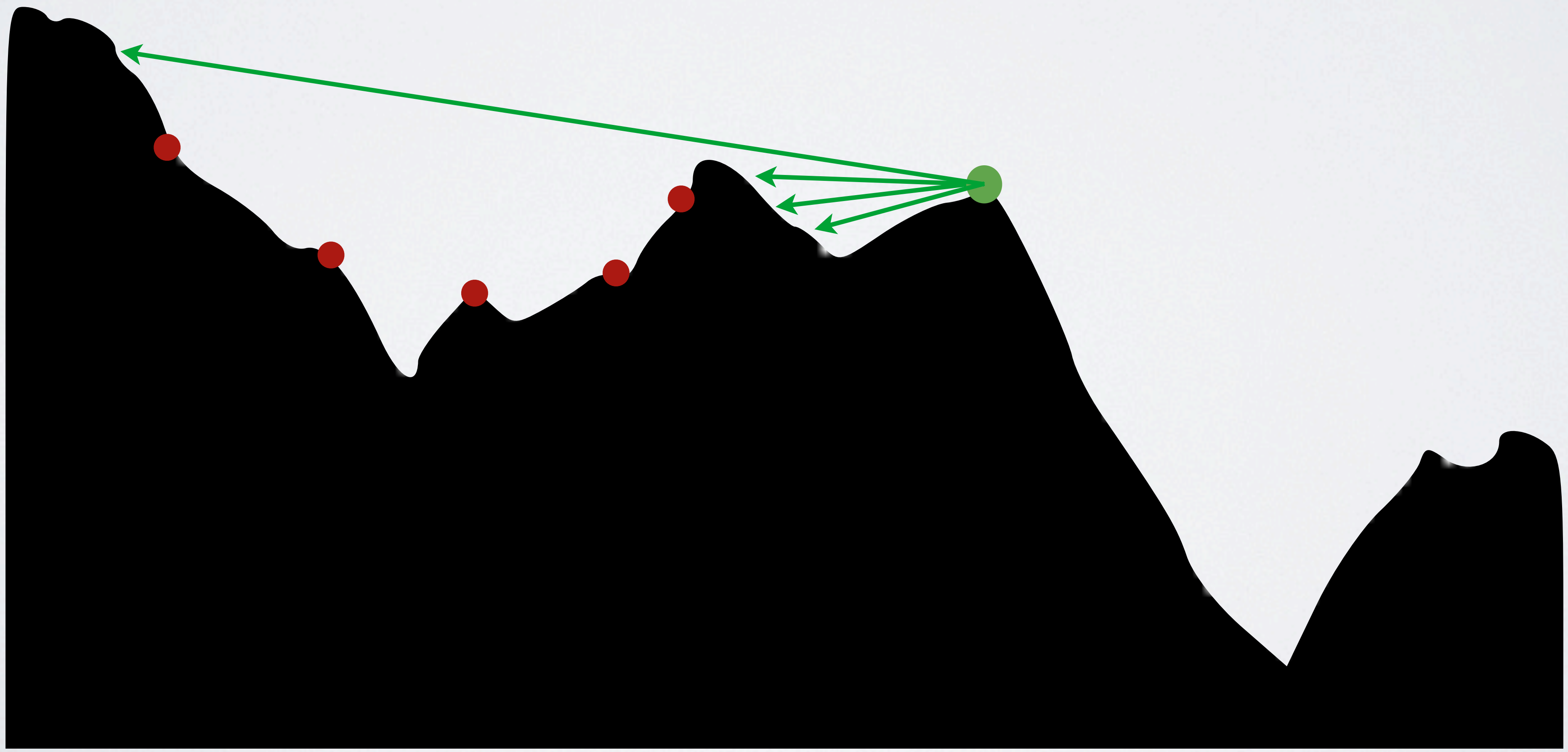
From each point, step through the slice



1 PREVIOUS SOLUTION

From each point, step through the slice

Given 1000 texels in a slice, ~500 steps taken per receiver on average to get accurate visibility

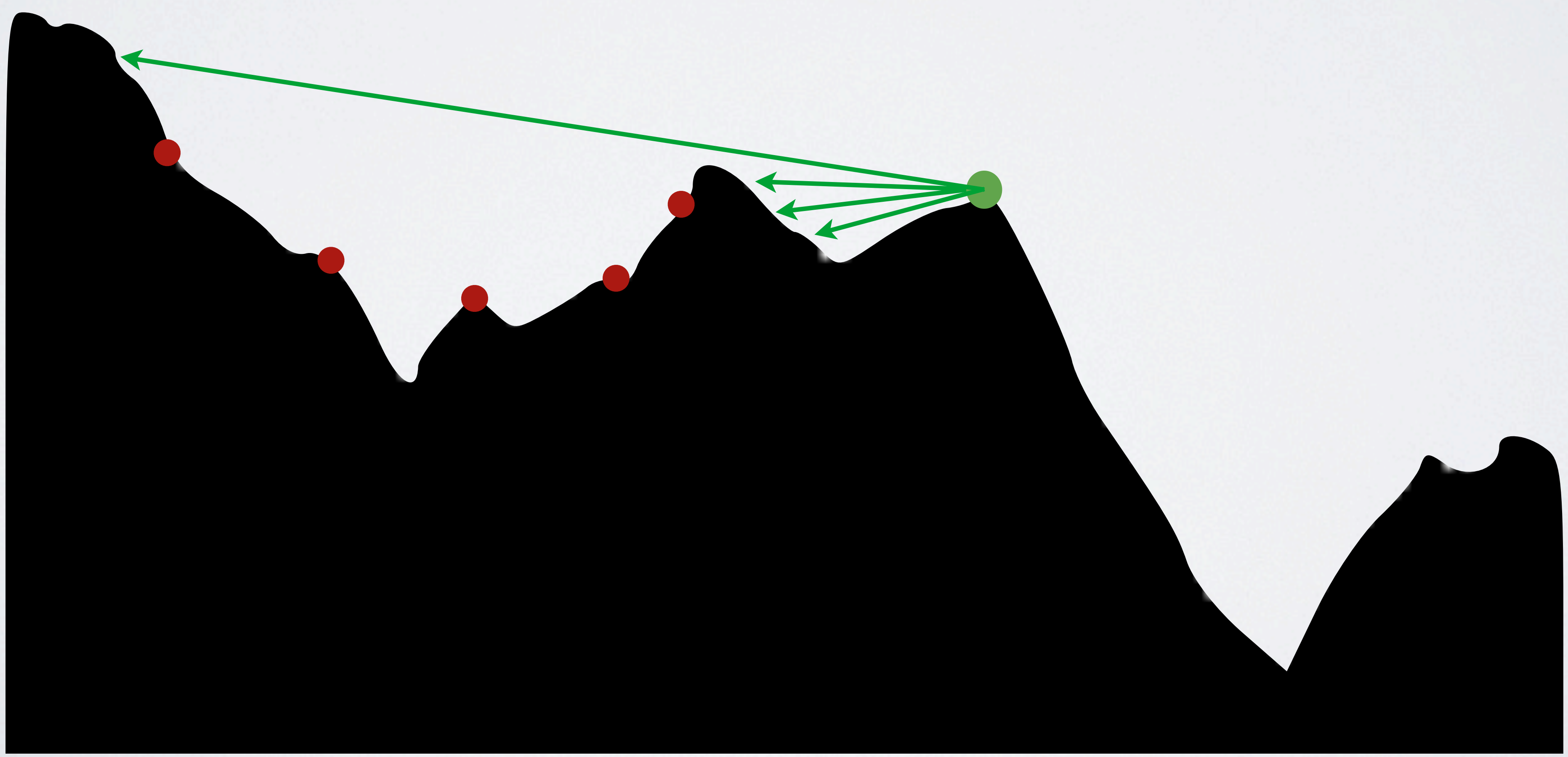


1 PREVIOUS SOLUTION

From each point, step through the slice

Given 1000 texels in a slice, ~500 steps taken per receiver on average to get accurate visibility

(Teaser: our method achieves this in ~10 iters per receiver)



1 PROBLEM DESCRIPTION

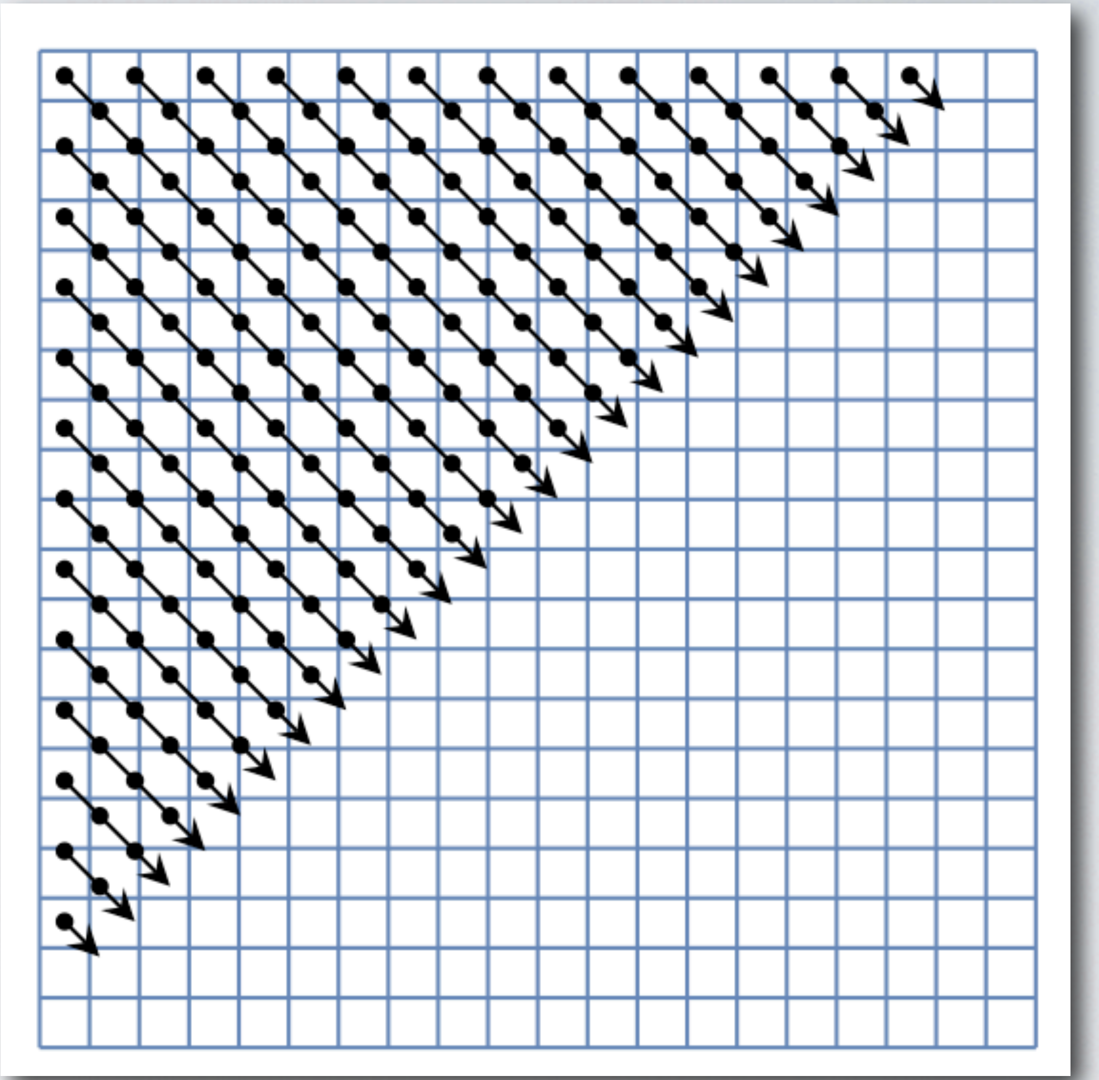
- Computationally very exhaustive
- It is not always necessary to solve accurately..
  - e.g. "Fast Global Illumination on Dynamic Height Fields" (Nowrouzezahrai & Snyder 2009)
- ..But we improve the time complexity of the *accurate* solution
- Useful for: movie-grade GI and (semi)glossy and self-illuminating surfaces

CONTENTS

2 OUR METHOD

Overview

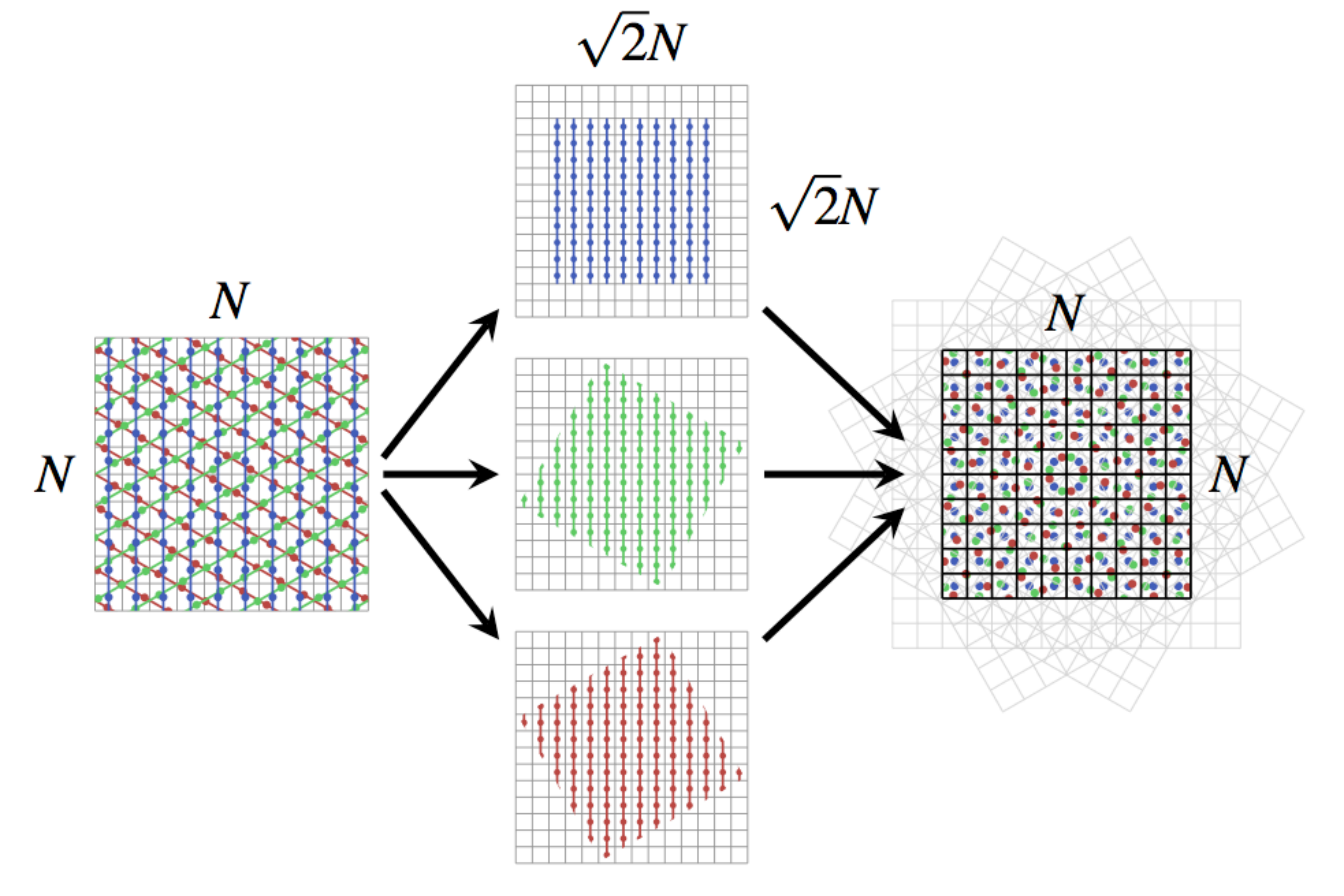
- Instead of computing visibility independently for each point..
- ..we compute visibility in parallel *lines* along the height field.
- Visibility *of* points along a line are computed *from* points along a line.
- Lines traversed incrementally, step by step.
- An internal representation of the HF is maintained along lines.



2 OUR METHOD

Overview

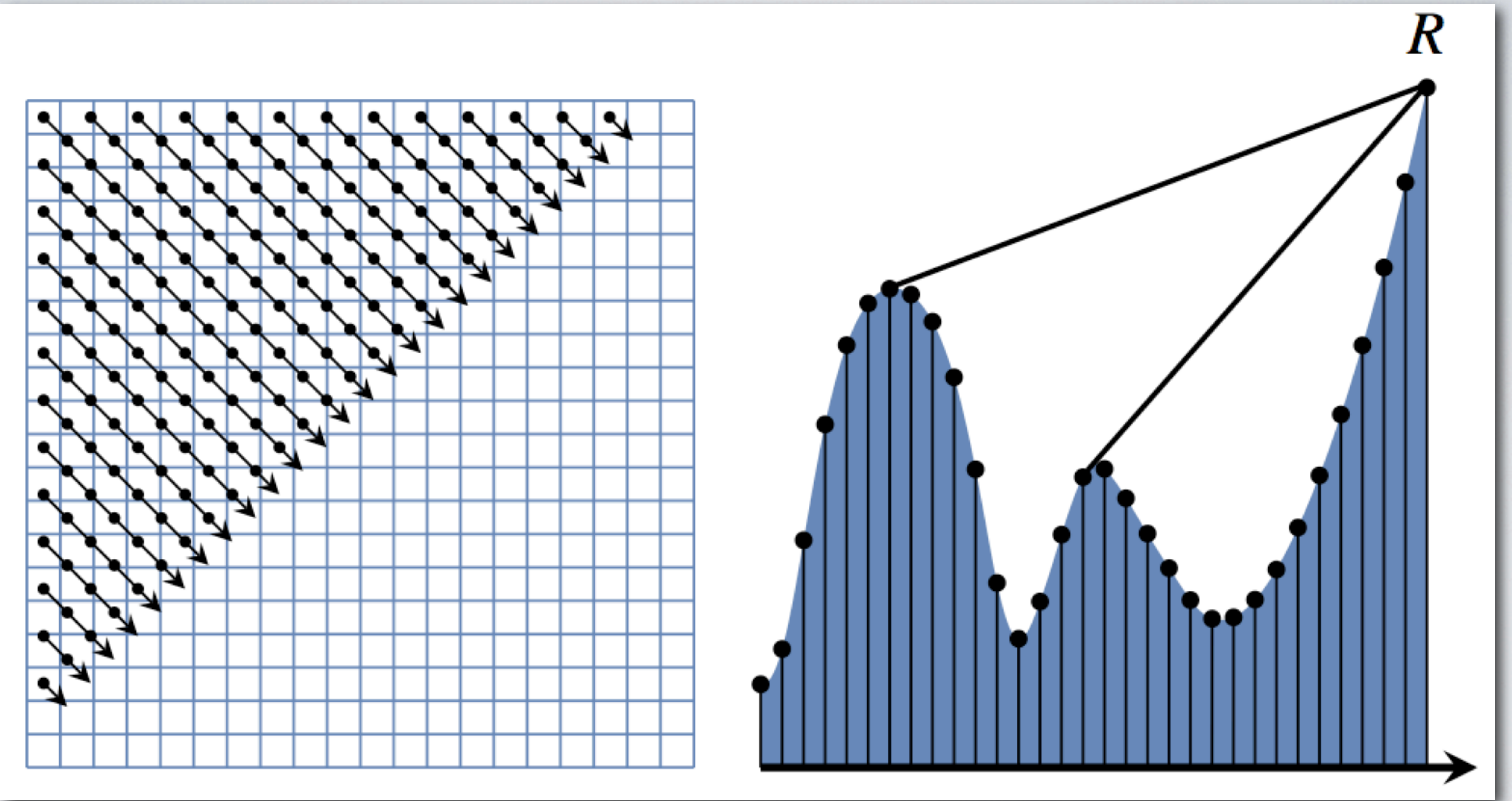
Line sweeps in K=3 directions (color coded)



The intermediate results are application specific (e.g. lighting values)

2 OUR METHOD

Processing one line



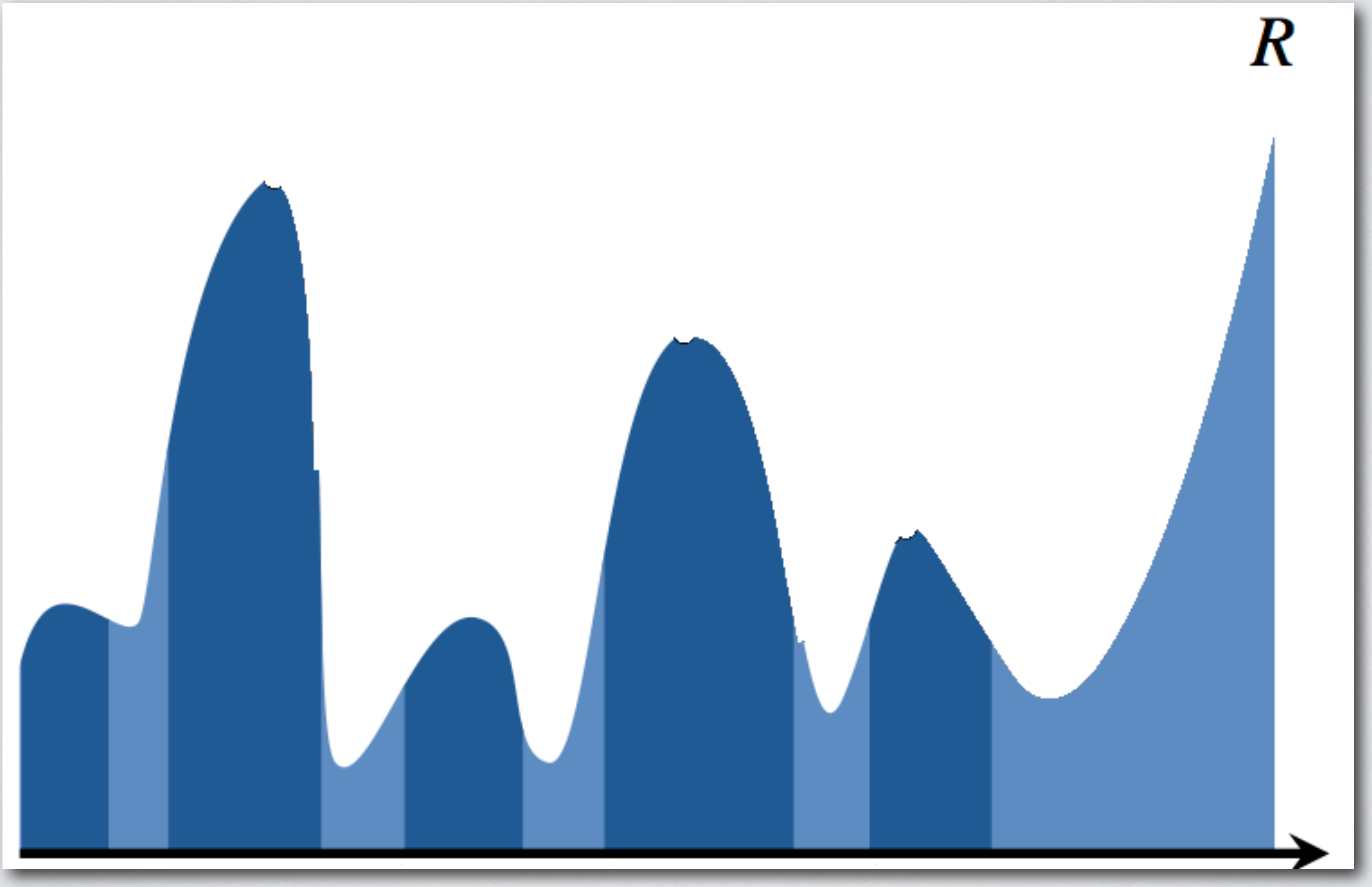
Parallel line sweeps

One line sweep

2 OUR METHOD

Processing one line

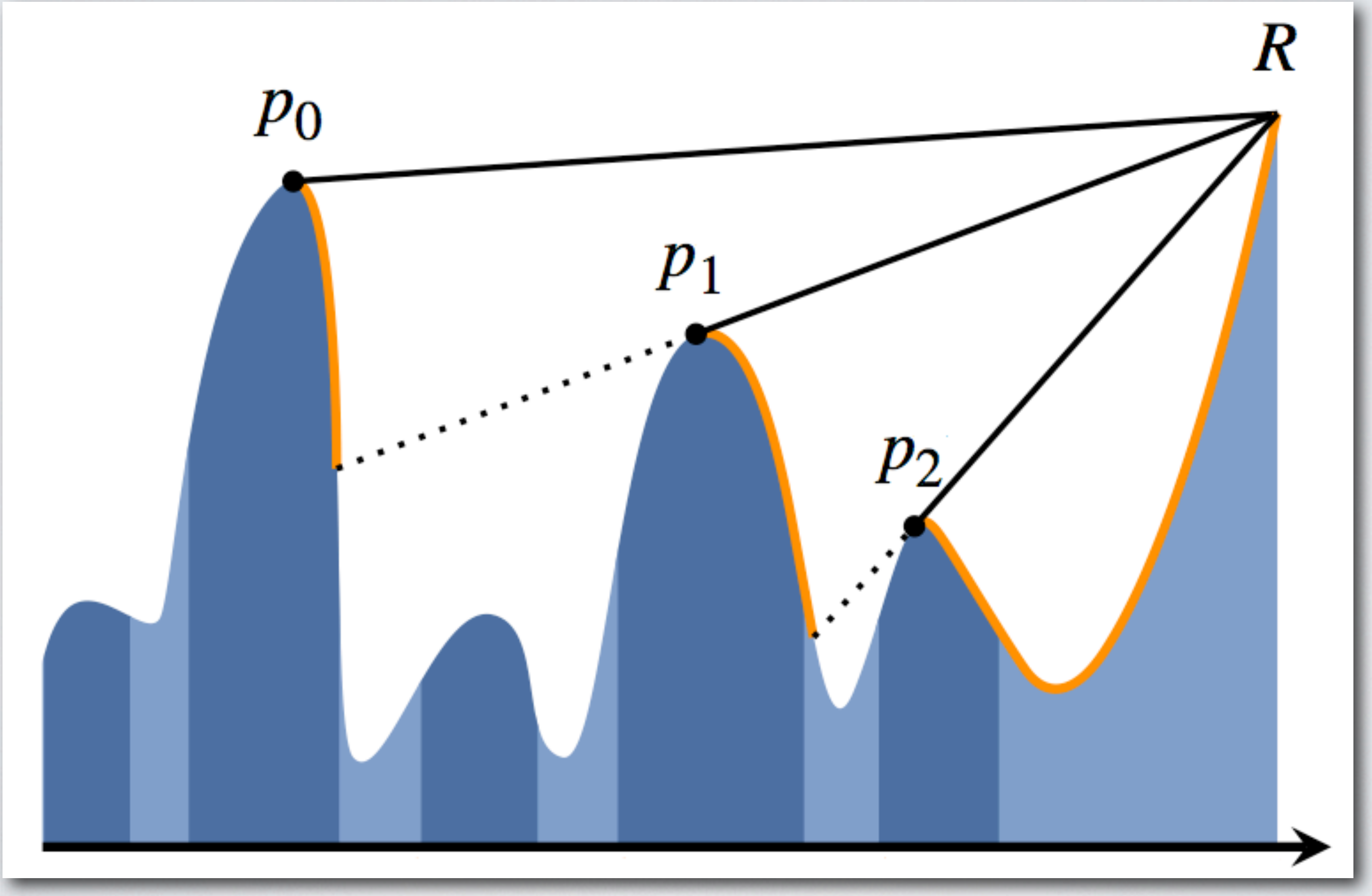
Firstly, the traversed line is split into convex (dark blue) and concave (light blue) segments:



2 OUR METHOD

Processing one line

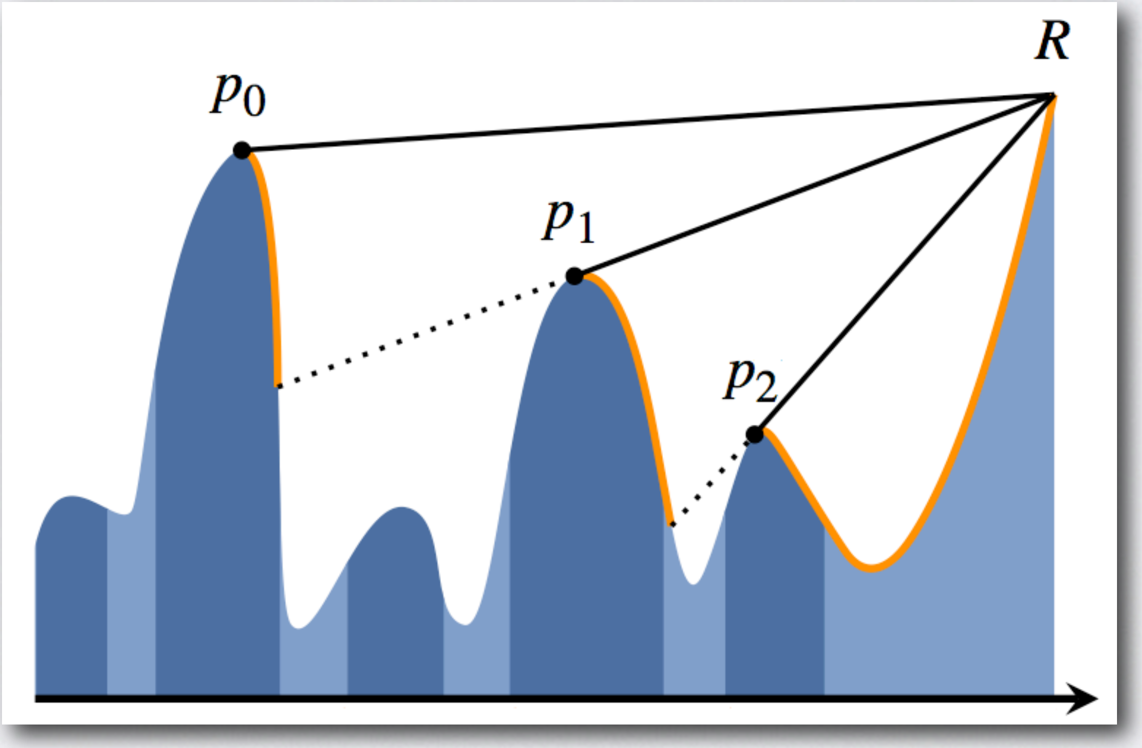
- Points p_i are at local peaks (as seen from R)
- Lines from R to p_i are *visibility horizons*



2 OUR METHOD

Processing one line

- If you want to enumerate visible points: start from p_i and step towards R until below the line from R to p_{i+1}
- Visibility horizons therefore describe the same visibility, but are more compact: a pair encloses all consecutive points

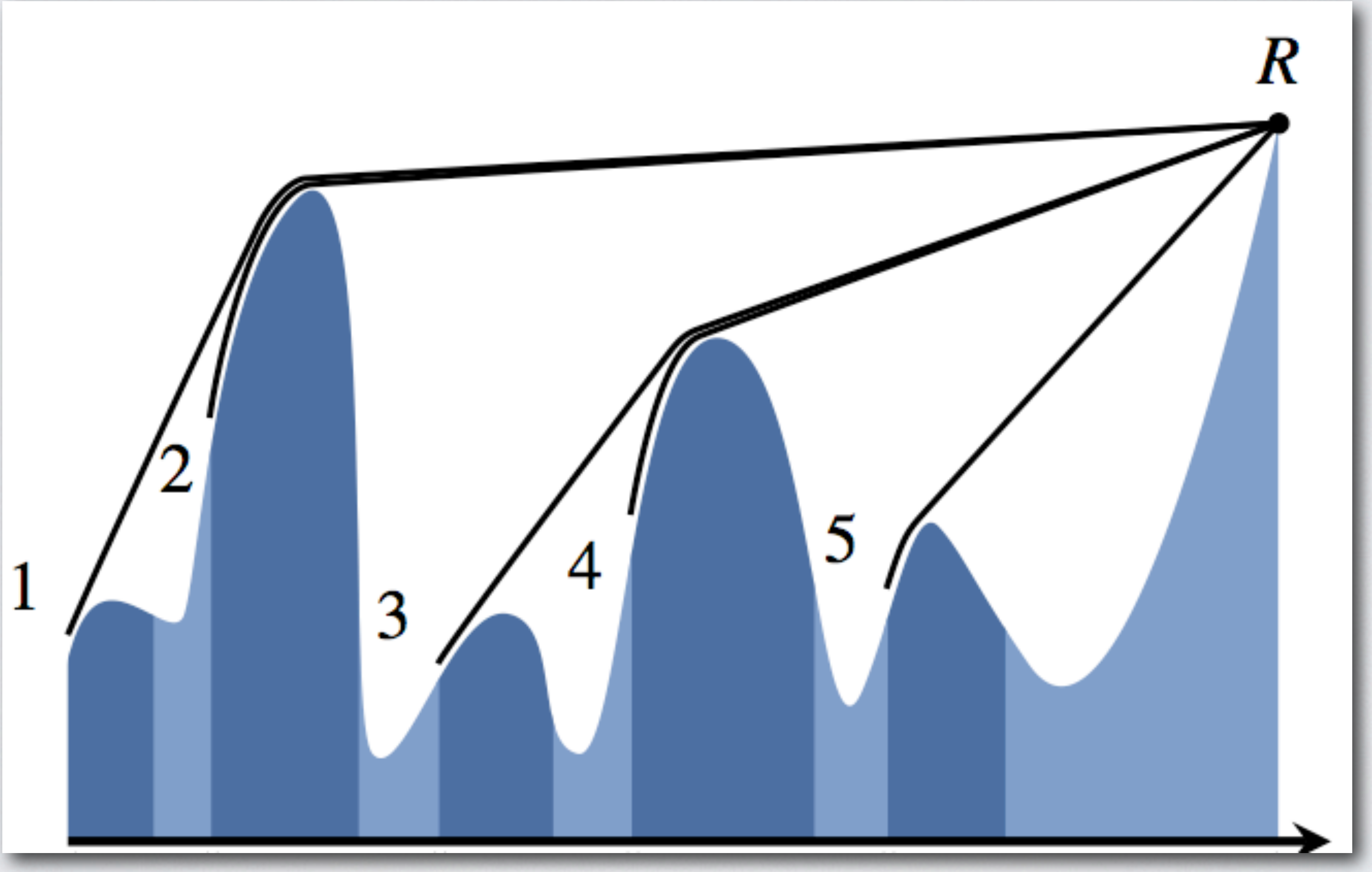


Orange = points visible to R

2 OUR METHOD

In search of p_i

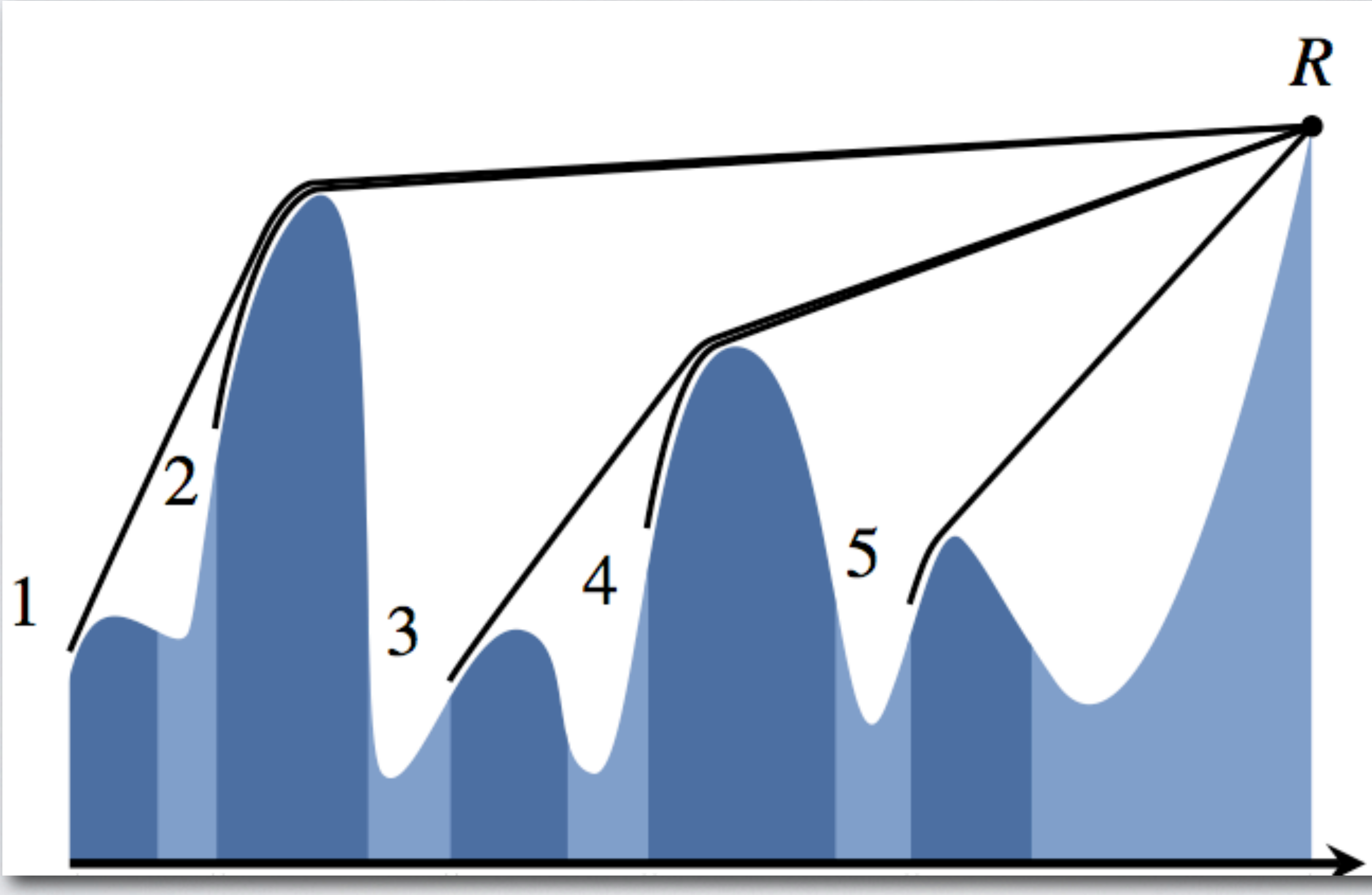
We construct a *convex hull* from the beginning of each convex section to *R*



2 OUR METHOD

In search of p_i

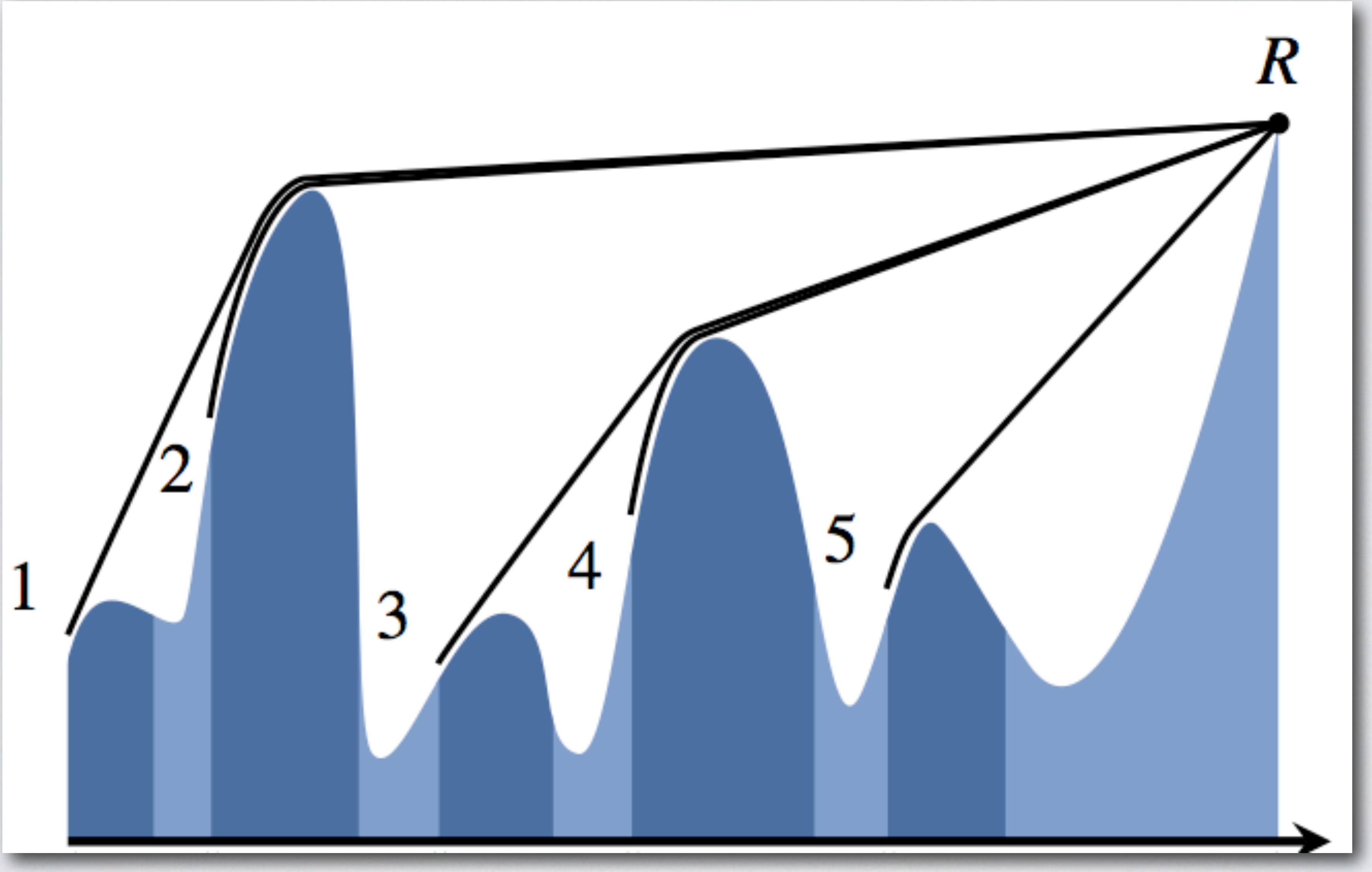
Convex hulls can be incrementally updated via Graham's scan when moving onto next R along the line (remove elements at the end until convex before putting R in)



2 OUR METHOD

In search of p_i

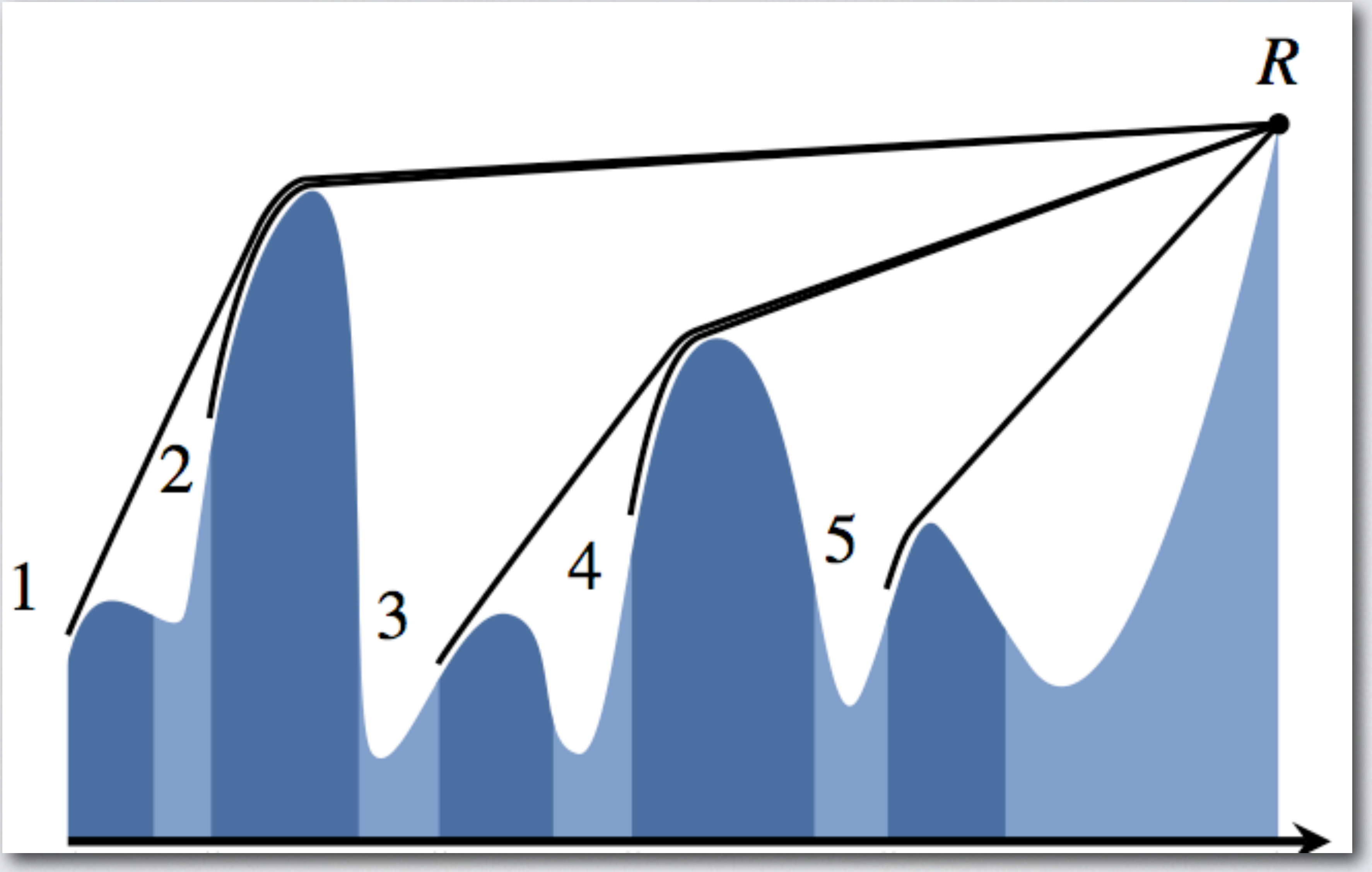
But look! The convex hulls seem to form a tree



2 OUR METHOD

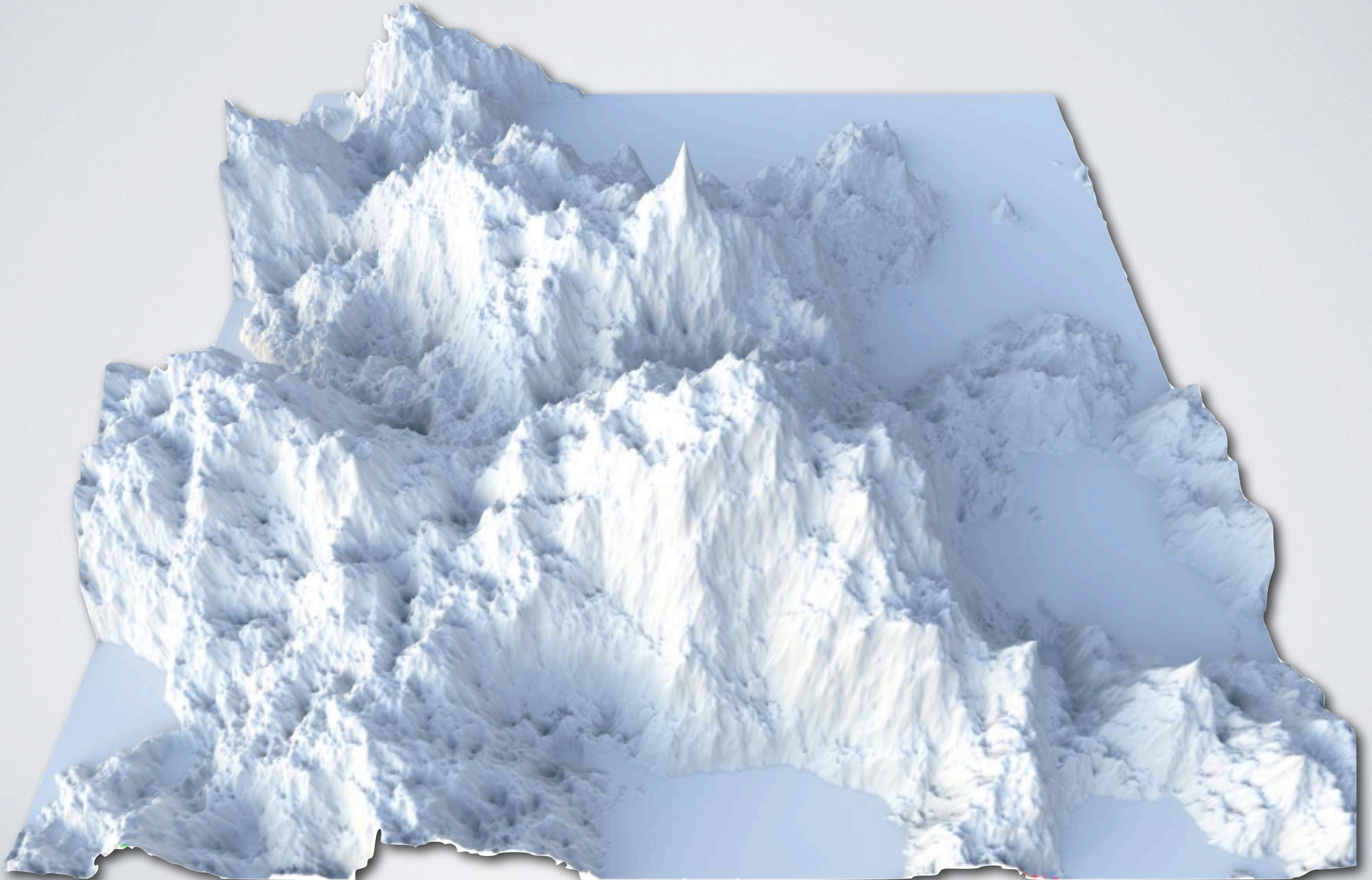
In search of p_i

- When put in a tree, visible segments become direct children of R
- Therefore no need to find which segments are visible
- And later on tree traversal only visits visible parts of the height field



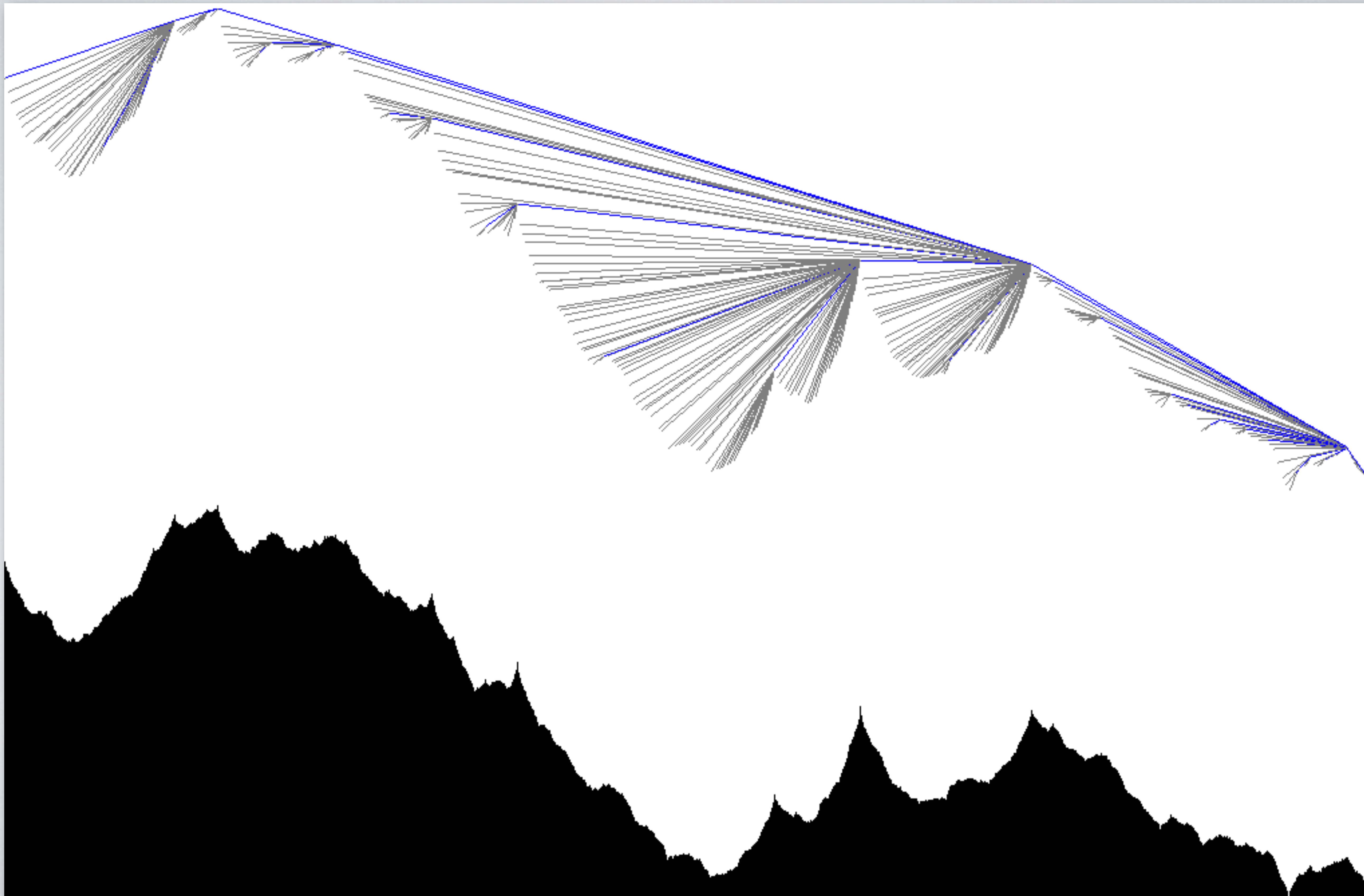
2 OUR METHOD

Visualizing the convex hull tree



2 OUR METHOD

Visualizing the convex hull tree

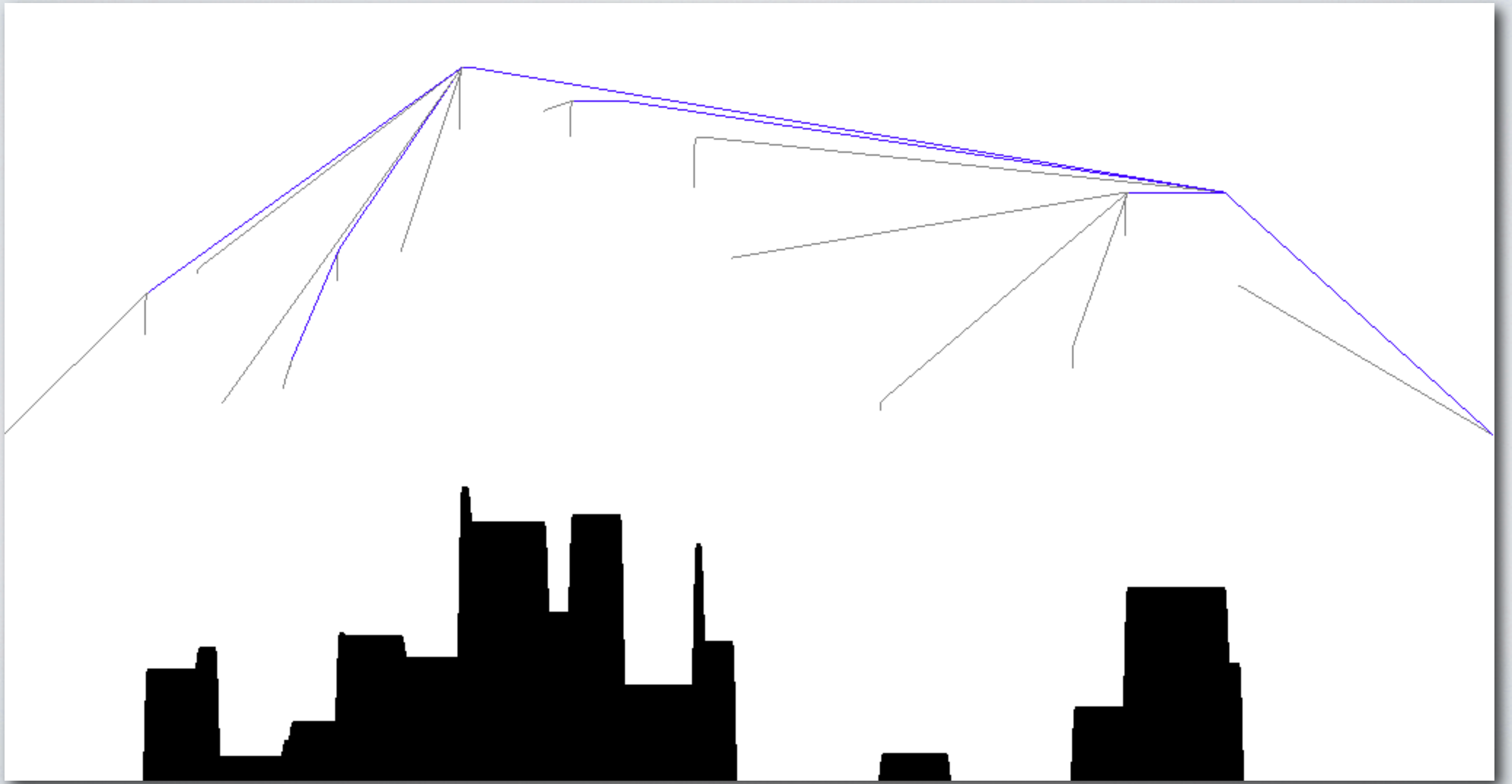


2 OUR METHOD

Visualizing the convex hull tree

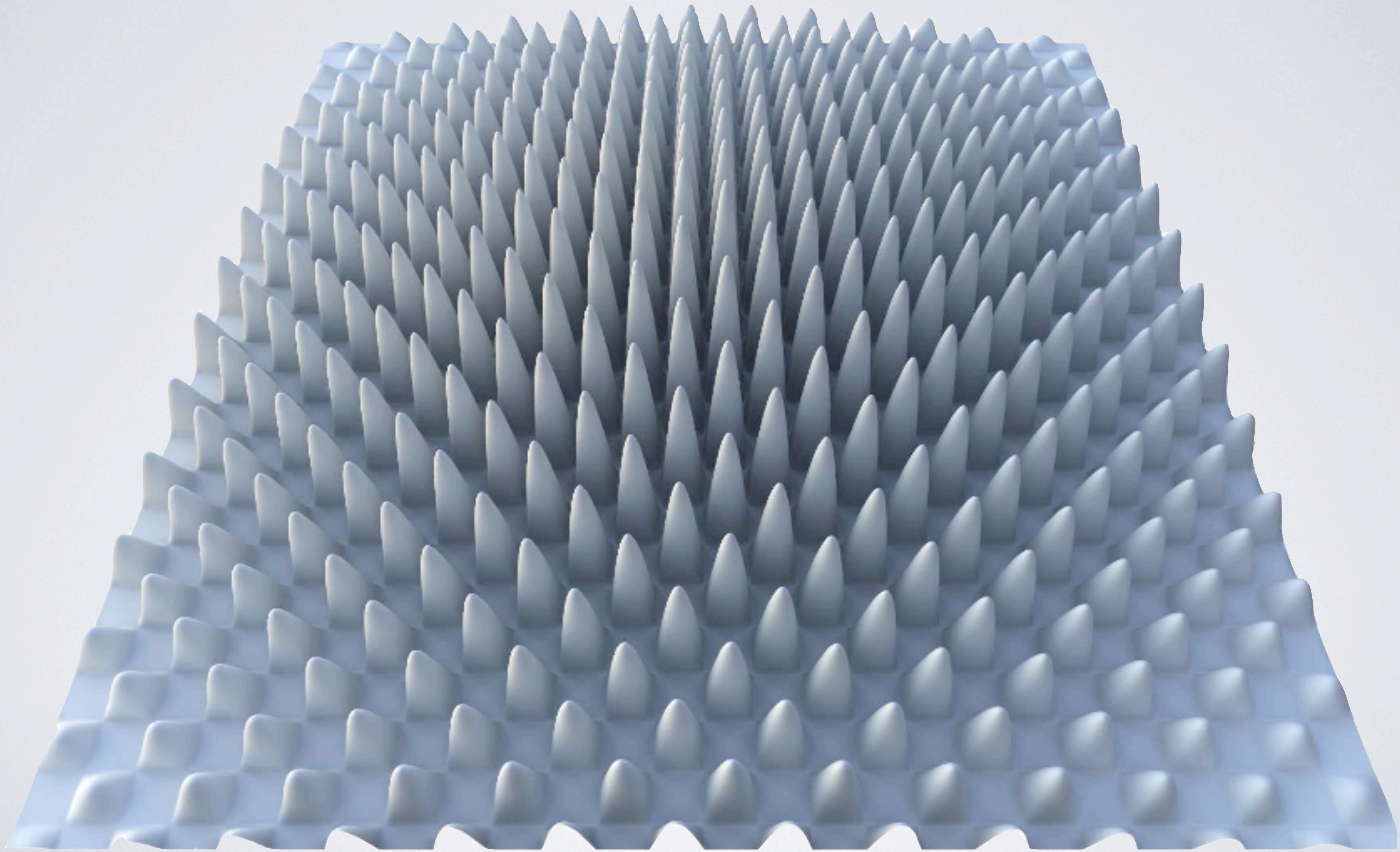
2 OUR METHOD

Visualizing the convex hull tree



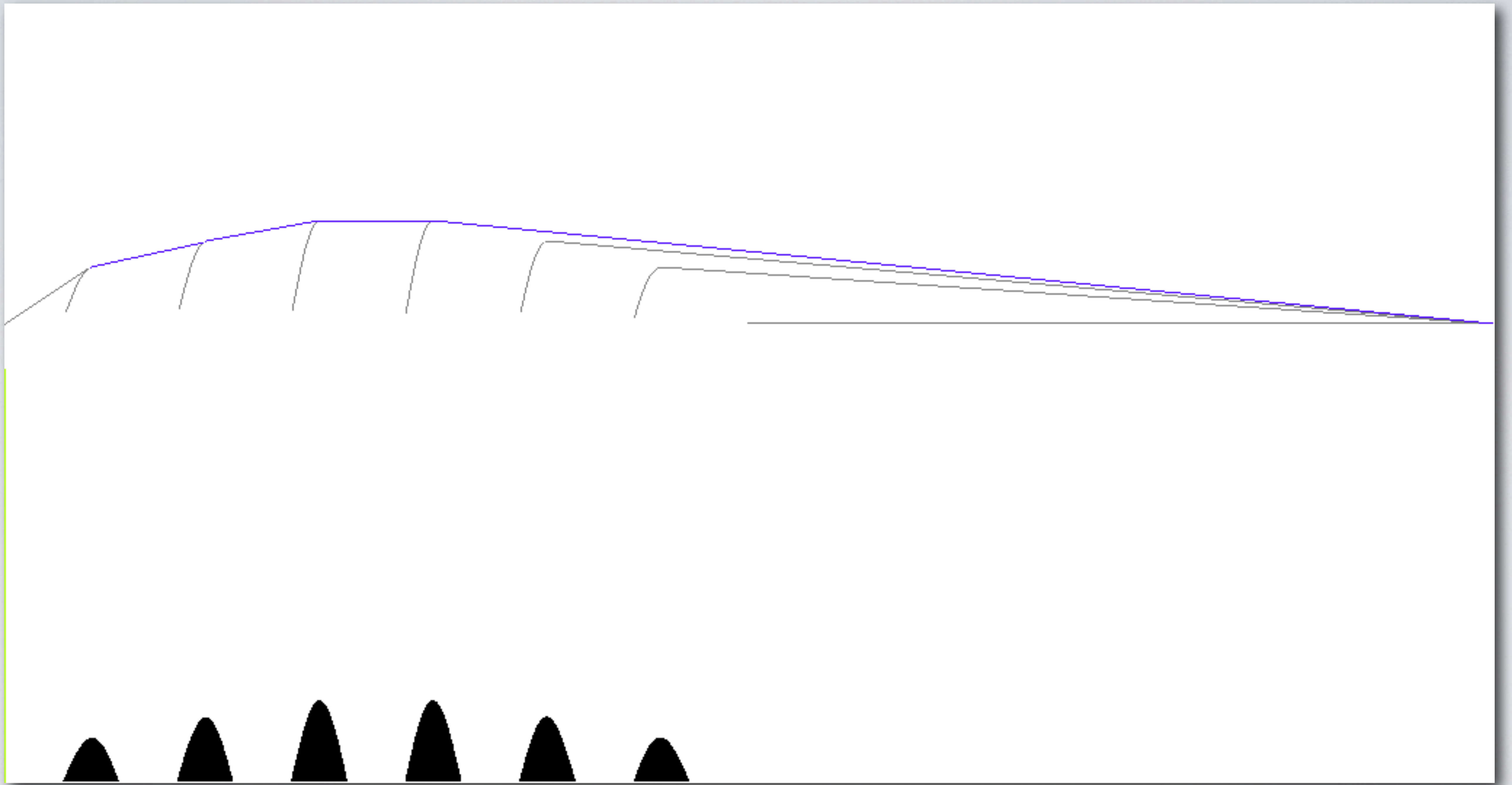
2 OUR METHOD

Visualizing the convex hull tree



2 OUR METHOD

Visualizing the convex hull tree



(I flattened the tail to reveal the structure better)

2 OUR METHOD

Creating the convex hull tree

(See the accompanied video)

2 OUR METHOD

Creating the convex hull tree

The convex hull tree can be incrementally updated at each new *R* in O(*h*) time where *h* is the number of visibility horizons

Algorithm 1 RecConvexity(child$_T$, parent$_T$, root)

if !convex(child$_T$ $\rightarrow$ parent$_T$ $\rightarrow$ root)
  connect child$_T$ to root before parent$_T$
  if child$_T$ has a next sibling
    first child of parent$_T$ $\leftarrow$ next sibling of child$_T$
    // Step wider
    RecConvexity(next sibling of child$_T$, parent$_T$, root)
  else
    delete parent$_T$

  if child$_T$ has a first child
    // Step deeper
    RecConvexity(first child of child$_T$, child$_T$, root)

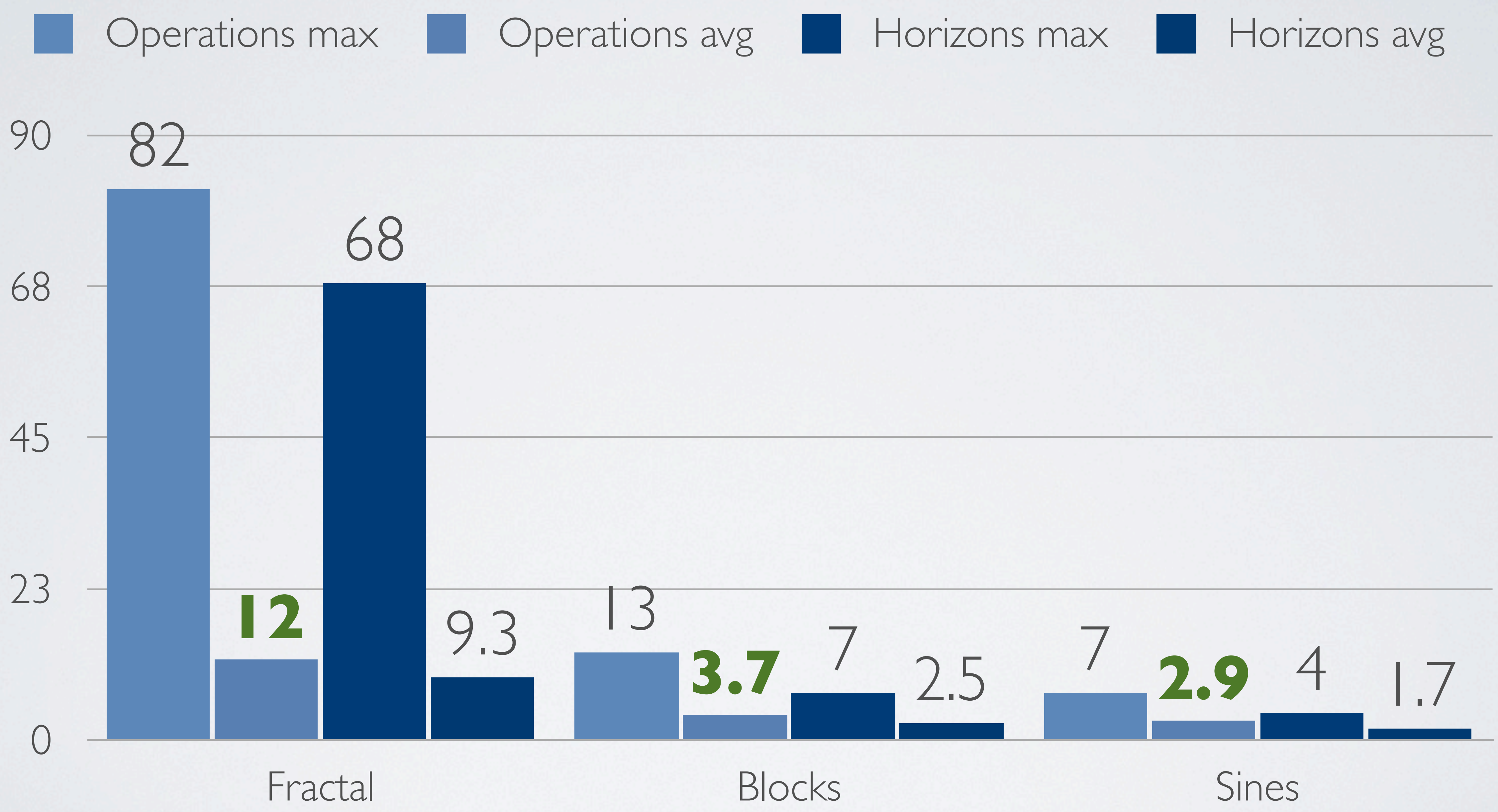
For more info, check out the paper
(especially if interested in an efficient GPU implementation)

CONTENTS

3 RESULTS

Creating the convex hull tree

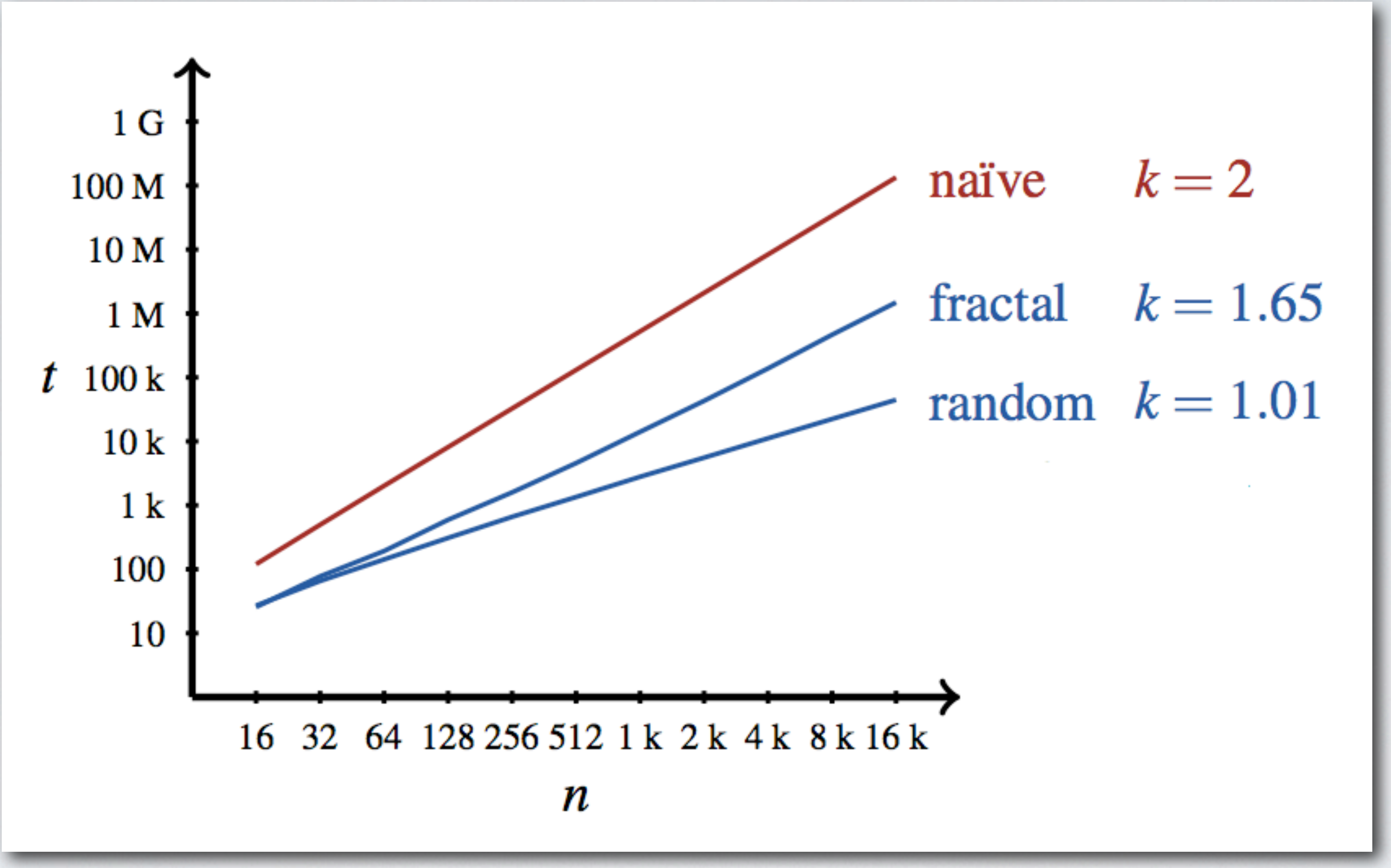
On a 1024^2 height field, per step on a slice:



40-200x improvement over the constant ~512 iters

Creating the convex hull tree

Scaling of t: $O(n^k)$



t = number of iterations, n = length of the line

3 RESULTS

Execution time

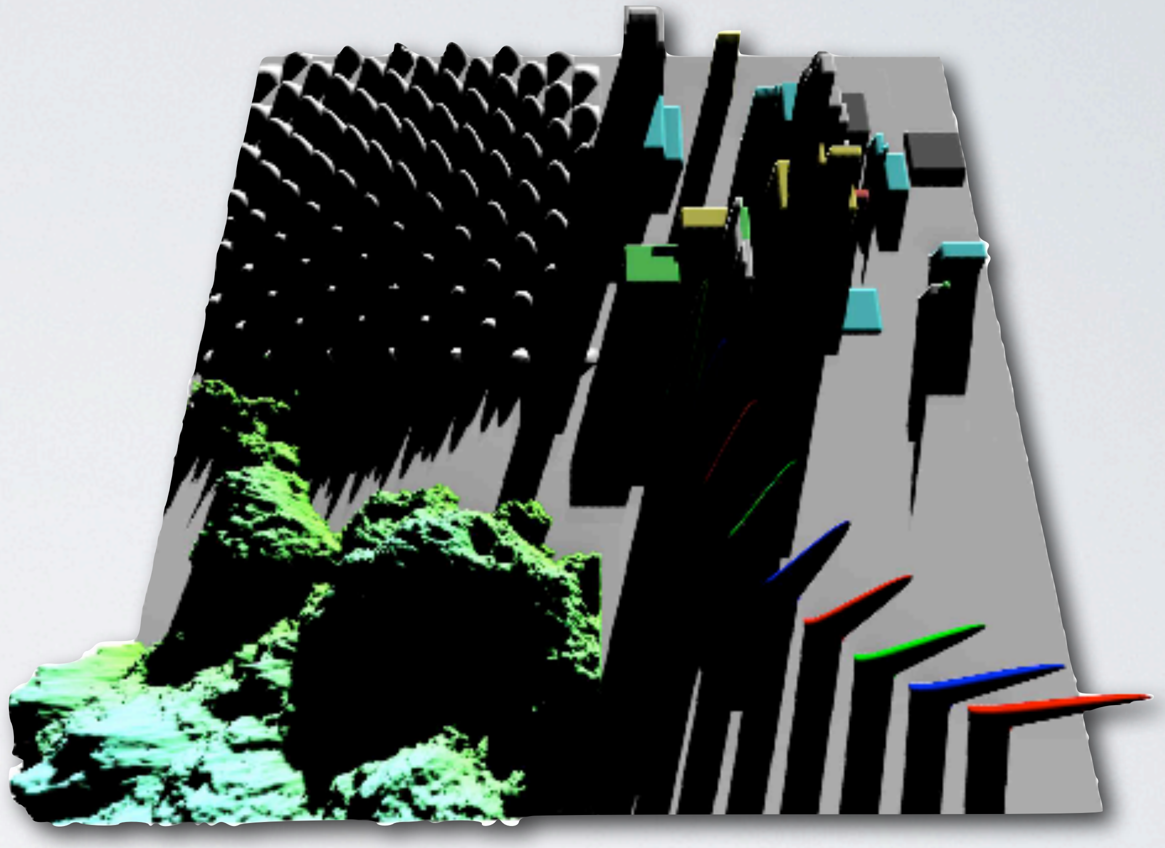
For one azimuthal direction
(multiply by K to get total times)
The height fields are 1024^2

Height field	Time	Speedup
naïve	21.2 ms	
fractal terrain	8.85 ms	*2.4x*
brick surface	5.05 ms	*4.2x*
sine grid	1.61 ms	*13x*
blocks	0.52 ms	*41x*
Figure 11	3.14 ms	*6.8x*
Figure 12	0.59 ms	*36x*

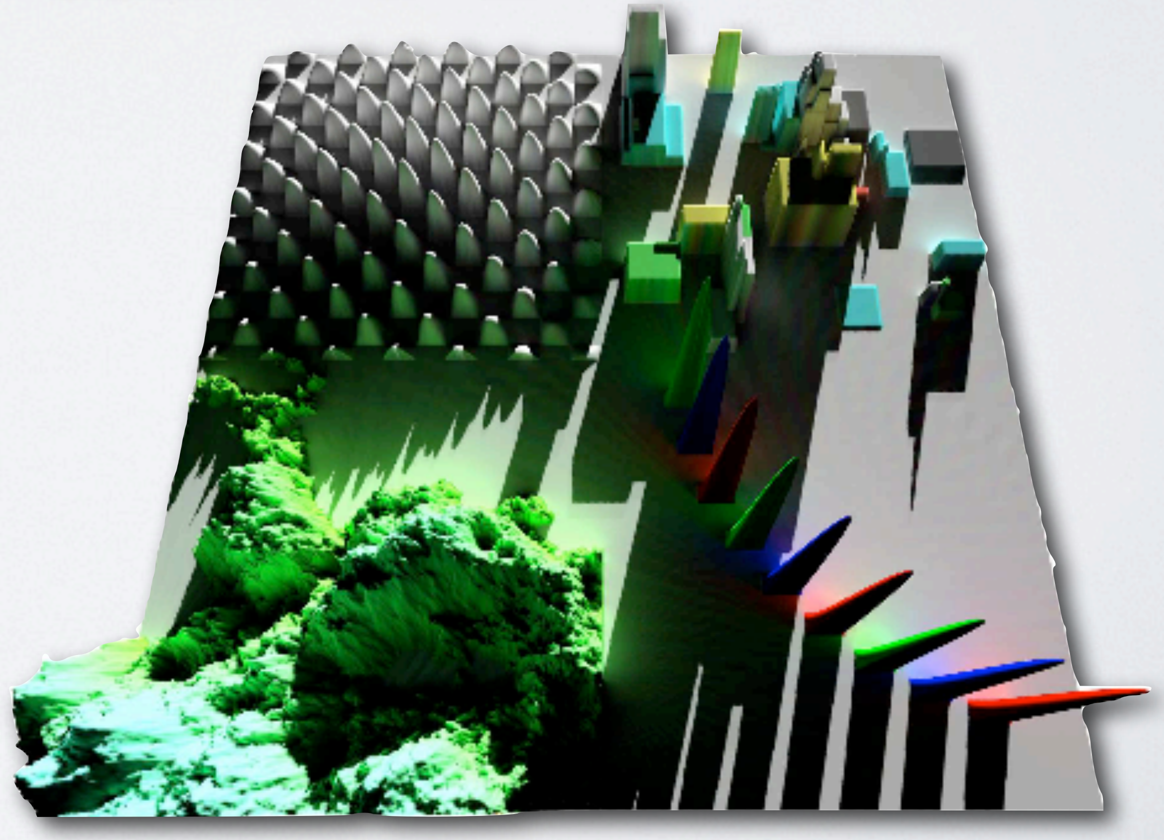
3 RESULTS

Indirect illumination

Direct lighting only

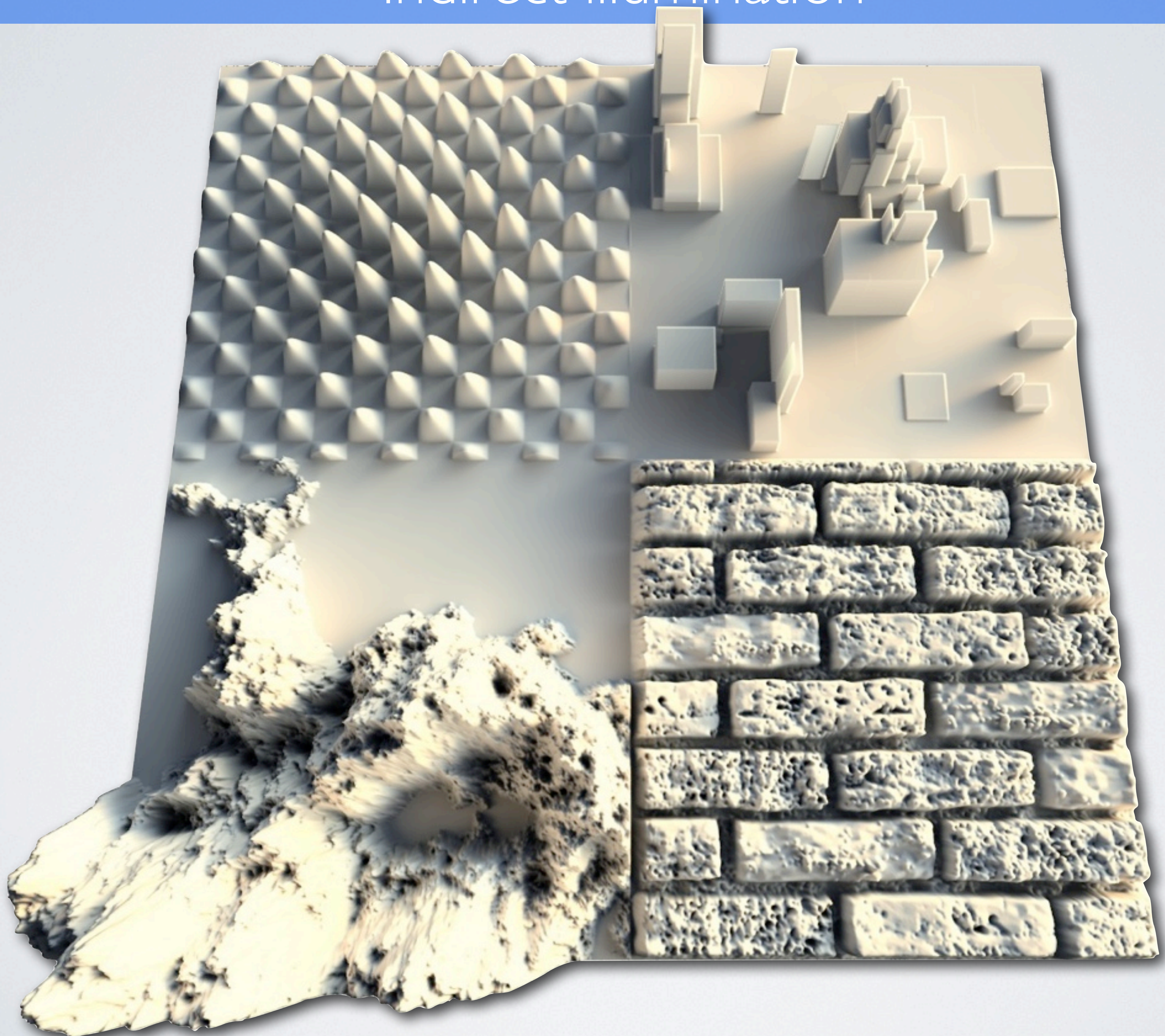


Direct + indirect



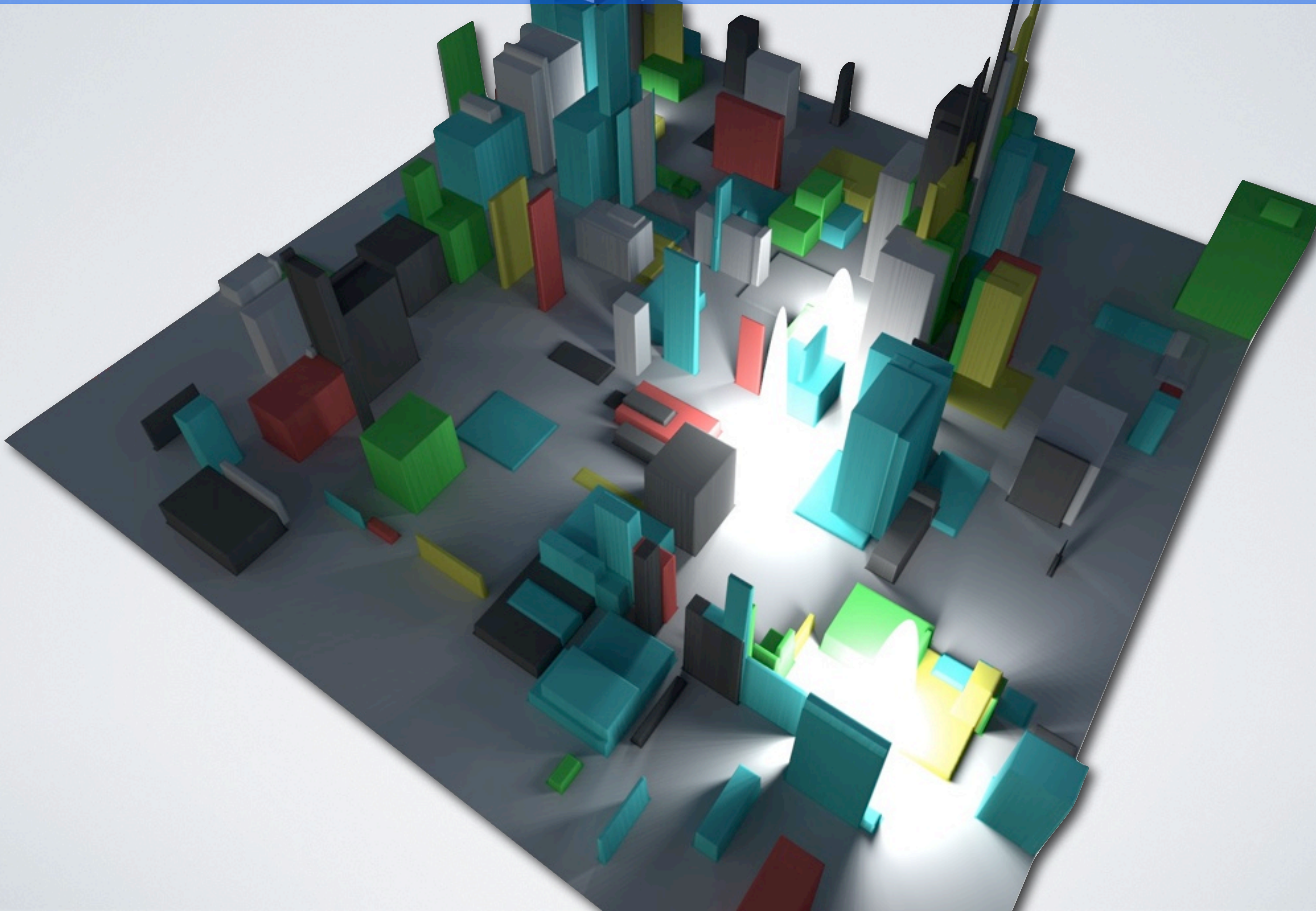
3 RESULTS

Indirect illumination



3 RESULTS

Self illumination



4 QUESTIONS

Or comments...

?