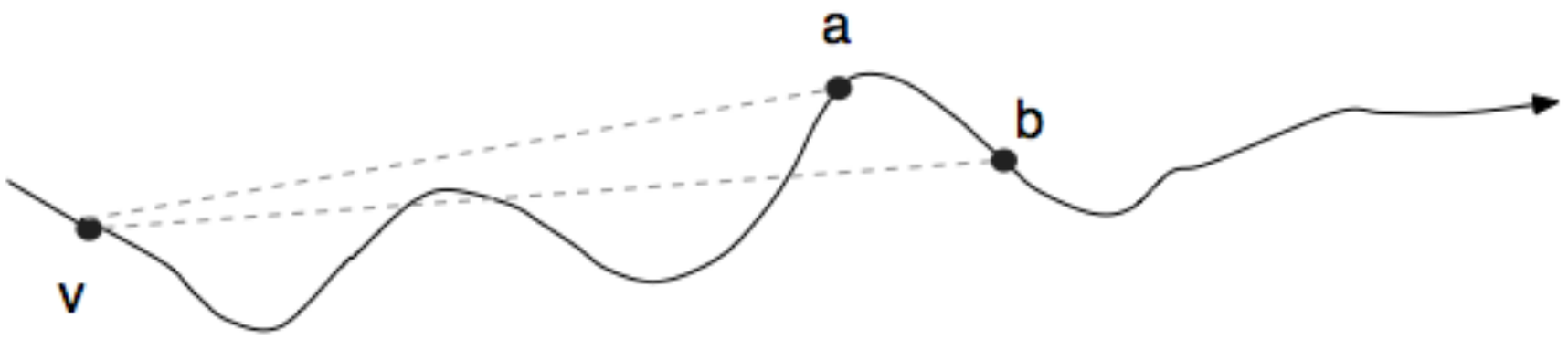
Algorithms for GIS

Visibility on terrains

Laura Toma

Bowdoin College



Points u,v are visible if segment uv does not intersect the terrain

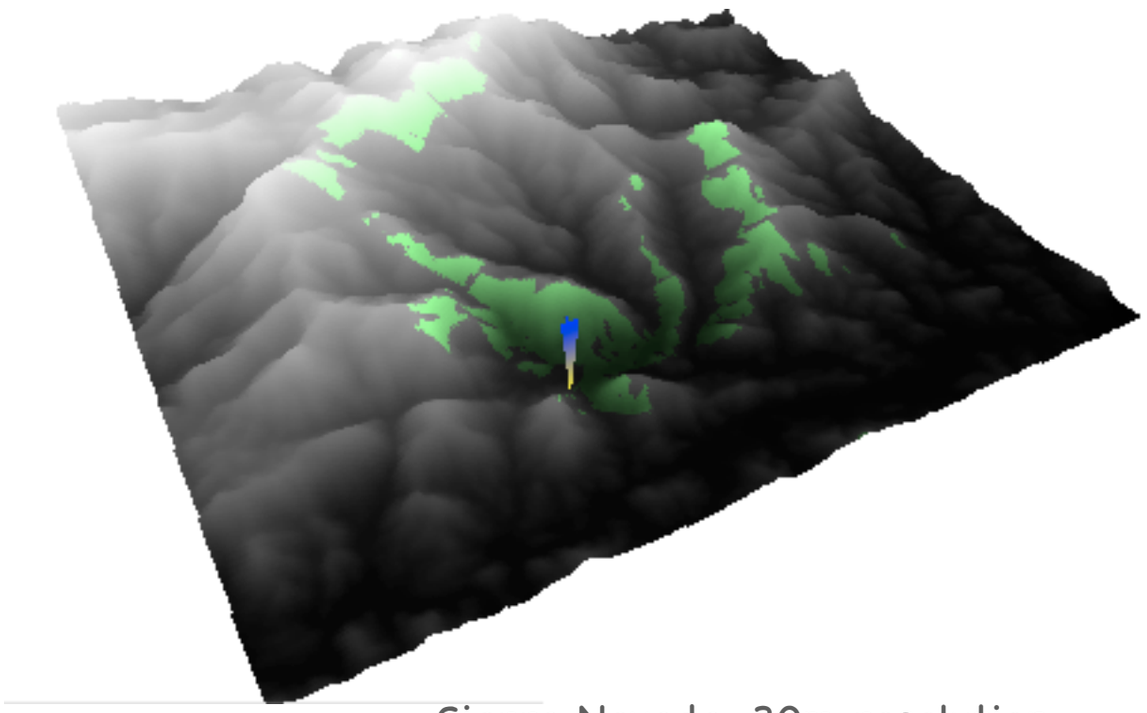
uv is called line-of-sight (LOS)

Visibility on terrains

- What can one see from a given point (on a terrain)?
- What is the point with largest/smallest visibility?
- How to place an ugly pipe in a scenic area?
- How to place a scenic highway?
- What is the cumulative visible area from a set of viewpoints?
- Find a set of tower locations that “covers” the terrain
- …

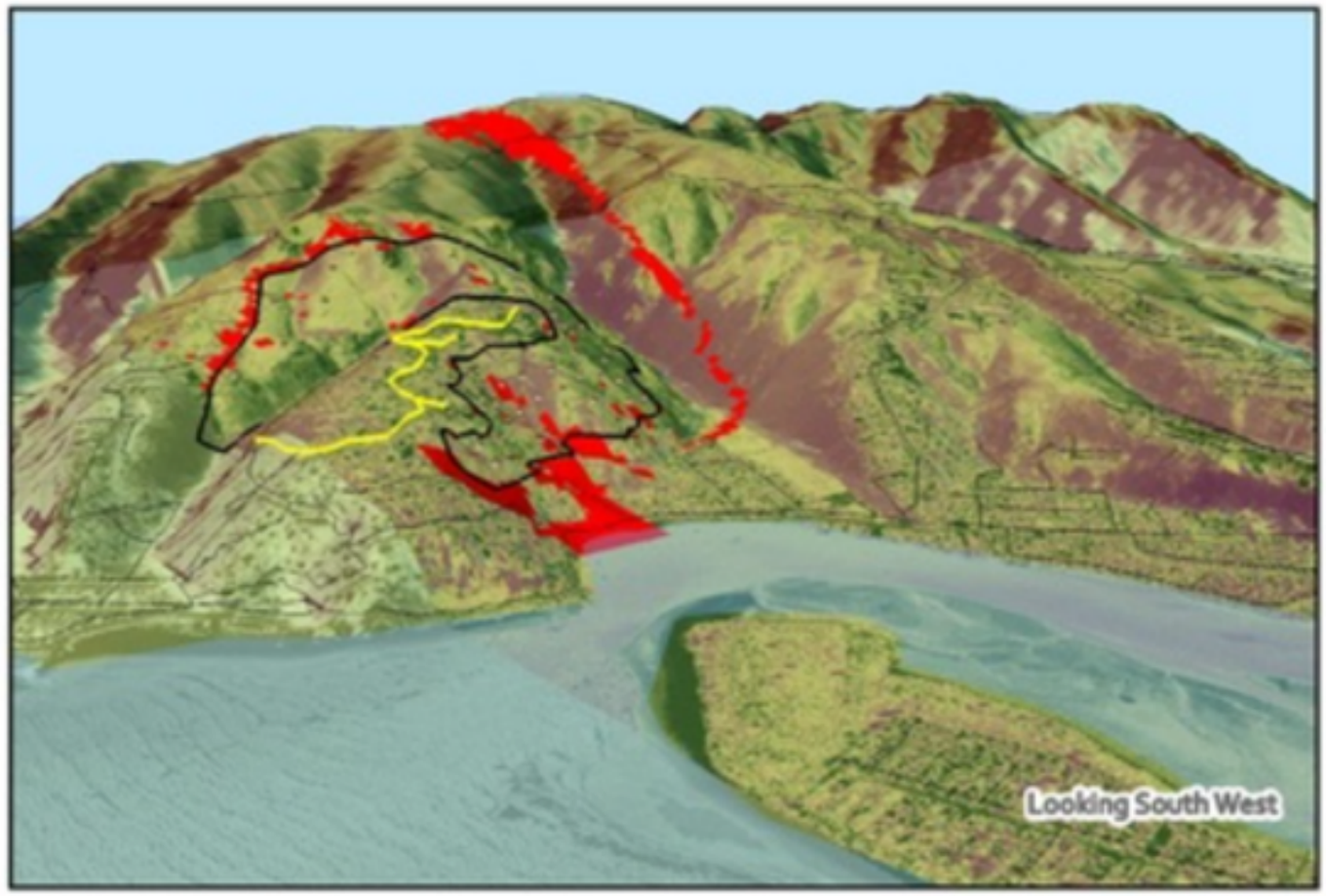
What can one see from a given point on a terrain?

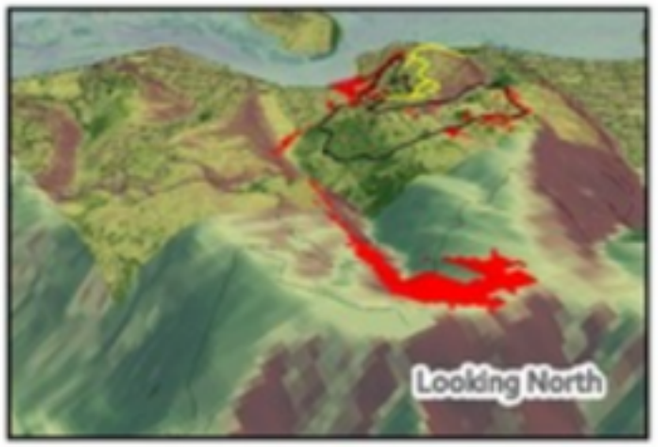
- Terrain model T + viewpoint v
- Compute the **viewshed** of v: the set of points in T visible from v



Sierra Nevada, 30m resolution

Visual Impact of Proposed Subdivision Port Hills, Christchurch

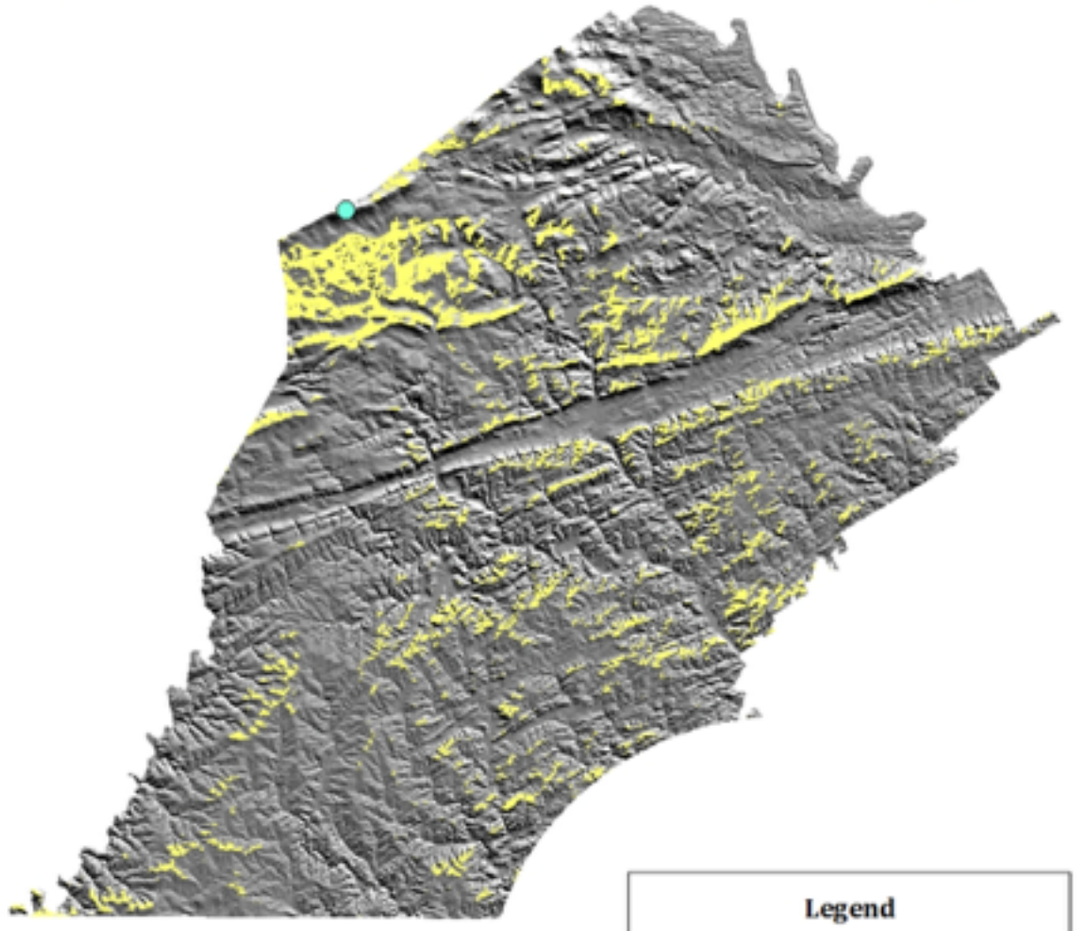




The above images present the proposed new subdivision (outlined black) on the Port Hills. The red area shows potential viewshed loss to residence on Hurst Seager Lane, Panorama Road, Revelation Drive and Starwood Lane (hightlighted yellow) if the proposed subdivision was to go ahead. The impact is based on potential bulidings being up to 10m in height.

GEOG205 Lab 7 - Viewshed Analysis and Visualisation - 28996639 - Bianca de Jong - Lab Stream 4 - 18 September 2012

Areas Visible From Welsh Mountain, Highest Point in Chester County, PA



What Could Joshua Chamberlain See at Gettysburg?

What Could Joshua Chamberlain See at Gettysburg? is a project that re-examines the decisions made during the Battle of Little Round Top in 1863 using digital humanities methods. Chamberlain is well known for ordering a bayonet charge that helped the Union line secure Little Round Top. Until now, Chamberlain's line of sight during the battle has not been investigated using digital tools. Digital tools create data that supplements the pre-existing historical narrative of that fateful day. One of the main objectives is to create data visualizations that show us not only what Chamberlain could actually see from his vantage point, but also uncover the external influences and forgone decisions that affected the outcome at Little Round Top. The visualization software Gephi was used to create digital networks of Chamberlain's overall correspondence based on over 500 original letters. Gephi enabled us to repurpose the original letters to gain insight into the humanistic side of Chamberlain's decisions during the battle. GIS, a mapping software, was then used for georeferencing and geoprocessing. Georeferencing compares the accuracy of the historical maps Chamberlain and his line used to modern topo maps. Using geoprocessing tools, it was possible to create an image that shows Chamberlain's visibility at Little Round Top, which suggests that the southern region of Little Round Top was the least visible from Chamberlain's position. This limited visibility challenges Chamberlain's strategy and could determine if his success in the environment of Little Round Top can be attributed to good

6th Annual Meeting of the European Association of Archaeologists, Lisbon, Portugal - 10th-17th September 2000.

The Application of Viewshed Analysis in Greek Archaeological Landscape.

S. Topouzi, S. Soetens, A. Gkiourou & A. Sarris

Laboratory of Geophysical-Satellite Remote Sensing and Archaeo-environment,
Institute for Mediterranean Studies - Foundation of Research & Technology, Hellas (F.O.R.T.H.),
Melissinou & Nik. Foka 130, PO. Box 119, Rethymnon 74100, Crete, Greece,
tel. ++30-831-25146, 56627, fax. ++30-831-25810

Abstract

Viewshed analysis has been extensively used in spatial modeling or landscape reconstruction of the ancient environmental settings and has been established as one of the most useful modules in the archaeological applications of Geographical Information Systems. The application of the technique has been proven especially effective in modeling spatial patterns within an uneven terrain.

Viewshed analysis was applied in the exploration of the defense system of the Mantineian plain in central Peloponnese, the study of the communication network among the historical towers of the island of Amorgos and the modeling of the location of the Minoan peak sanctuaries of the Lasithi district in NE Crete.

In the first case, the purpose of the analysis was to determine the visibility coverage of the plain surrounding the ancient city of Mantineia from watchtowers situated in the nearby hills and mountains and compare it with the corresponding visibility coverage resulting from each one of the gates of the city walls. The data were correlated to the results of previous surface surveys that had located ancient roads and paths. In this way, a thematic map of the potential defensive region has been created, suggesting either possible gaps or candidate archaeological sites that may have been used to complete the defense system of the region.

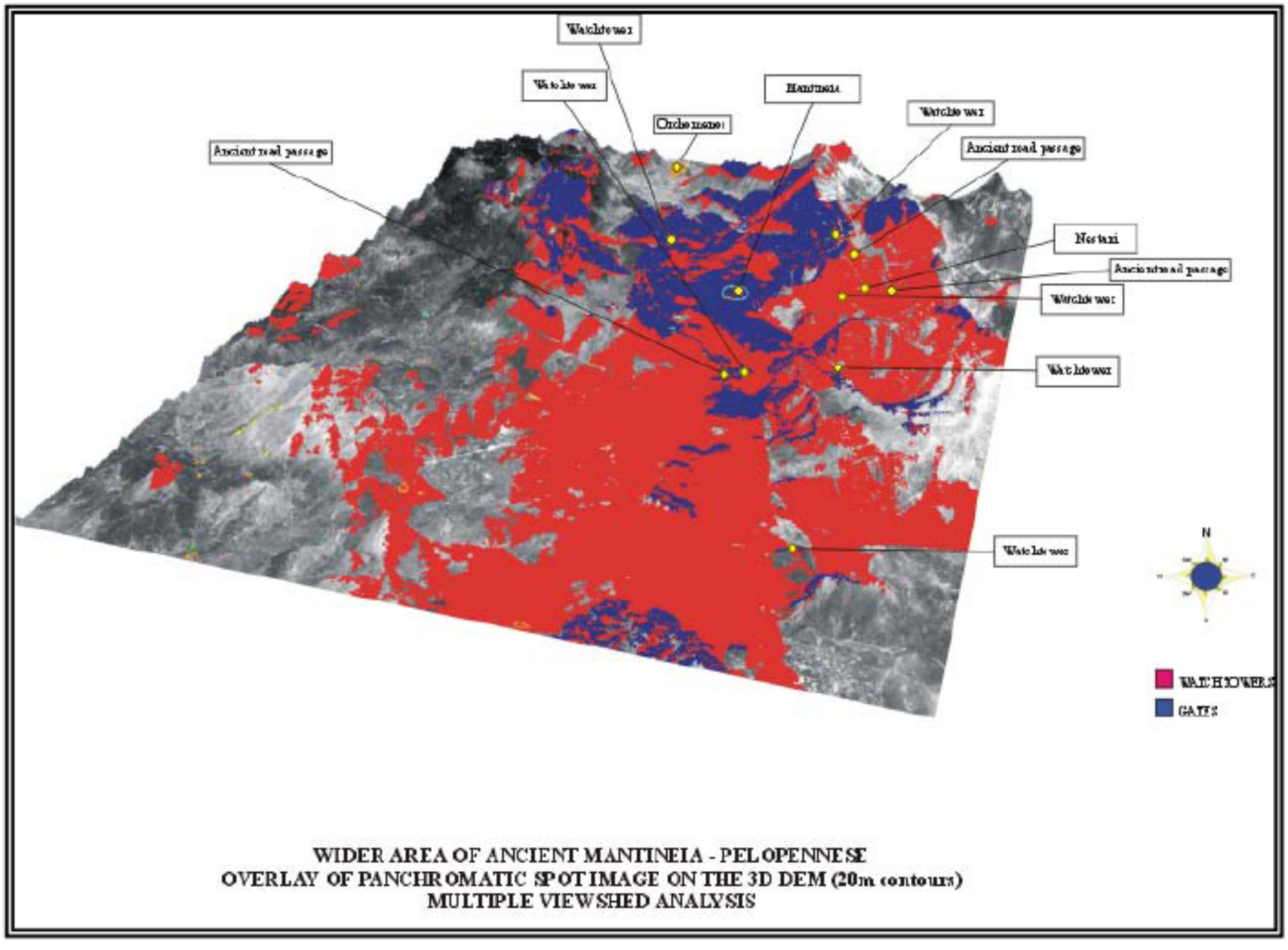


Figure 1-3. Overlay of the multiple viewshed analysis of the defensive network of the Mantineia plain, the outline of archaeological features and the SPOT PAN imagery on the 3D DEM.

Improving Methods for Viewshed Studies in Archaeology: The Vertical Angle

Mar ZAMORA

Universidad Autónoma de Madrid, Spain
mar.zamora@uam.es

Abstract

As many authors have observed, binary viewsheds are too simplistic a way to represent visibility around a particular viewpoint. Several deficiencies, very well summarized in Wheatley and Gillings (2000), must be corrected in order to make computer-generated viewsheds more realistic and geared to archaeological purposes. One of those required improvements relates to the vertical angle of vision. Viewing from a low angle gives less perception of detail than viewing from a high angle. Standard viewsheds do not allow the identification of this kind of perceptual issue. In the real world, visible areas at eye level are seen as a narrow strip; however, on the ground they can extend for many kilometres. The map thus gives a false representation of visibility. Dividing the viewshed calculation into several vertical angles helps to analyze the result in a more realistic way than is customary, especially in warlike contexts where dominant visibility could have been important for military purposes.

Keywords

Viewshed, total viewshed, isocrones, ArcGIS, path distance tool.

1. The importance of visual control during the Late Iron Age in Spain

During the Roman conquest period in Spain (II-I centuries BC), in several areas of Andalusia, some new settlements were located on hilltops, with extensive visibility.

This fact has been interpreted in two ways:

- as a wish to visually supervise indigenous settlements;
- as a way to show Rome's presence in the area, and to reinforce its power.

In particular, in the Guadalquivir River Valley, this fact has been observed in two adjoining areas. Romo *et al.* detected two new settlements on hilltops during the Roman republican period in Gilena. The authors think that these sites were established in order to (visually) control indigenous settlements, in the unstable context of the beginning of the roman conquest of Hispania (Romo *et*

similar phenomenon has been observed east of these zones, in an area shared by the current provinces of Seville and Cordoba, in the Genil River Valley (Zamora in press).

1.1. The Priego-Alcaudete basin

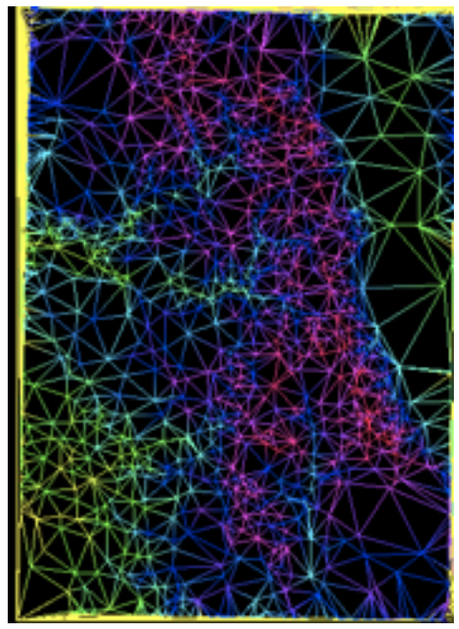
The Priego-Alcaudete basin is located east of the Sierras Subbéticas, in the southern part of the provinces of Cordoba and Jaen (Andalusia, Spain). The area is adjacent to the Genil River Valley (*Fig. 1*).

During the Middle Iberian period (prior to the beginning of the Roman conquest) there was no

What can one see from a given point on a terrain?

Problem: Given terrain model T + viewpoint v, compute the **viewshed of v** (the set of points in T visible from v)

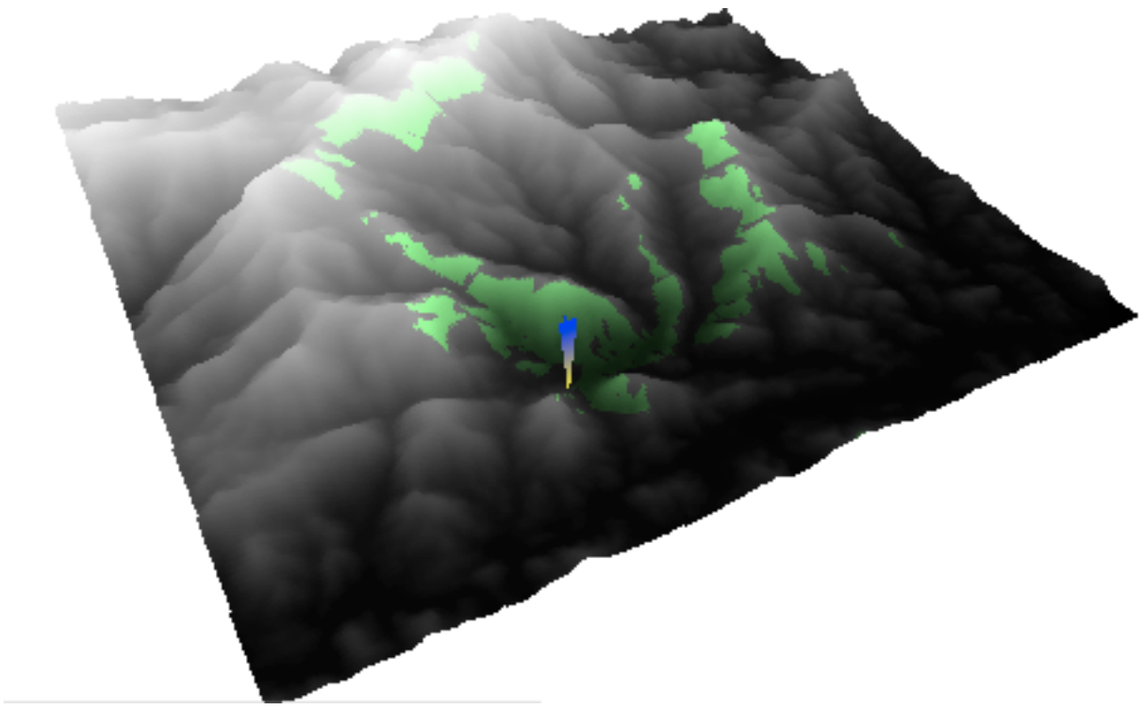
- Grid terrains
  - discrete view shed
  - compute which grid points are visible/invisible
  - viewshed is a grid



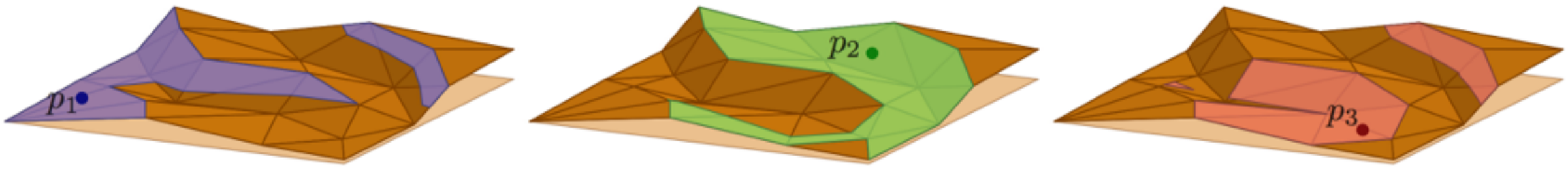
- TIN terrains
  - continuous viewshed
  - compute the exact shape of the viewshed

Viewsheds on grid terrains

20	23	25	26	32	46
21	20	24	28	41	46
24	21	23	31	36	36
23	22	24	27	33	34
32	22	29	30	35	34
29	30	33	34	36	37



Sierra Nevada, 30m resolution



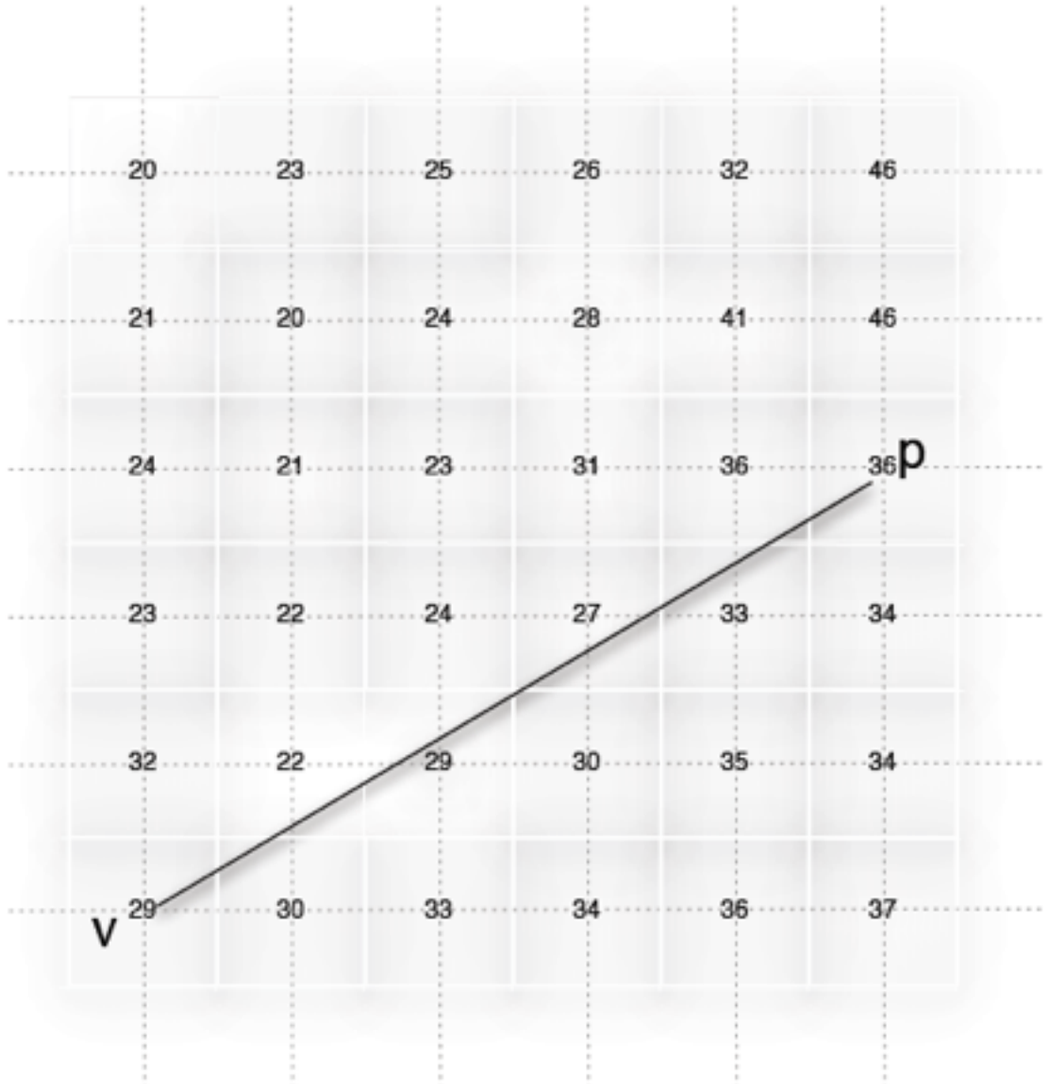
from: http://arxiv.org/pdf/1309.4323.pdf

Basic viewshed algorithm

Basic viewshed algorithm

Input: elevation grid
Output: visibility grid, each point marked visible/invisible

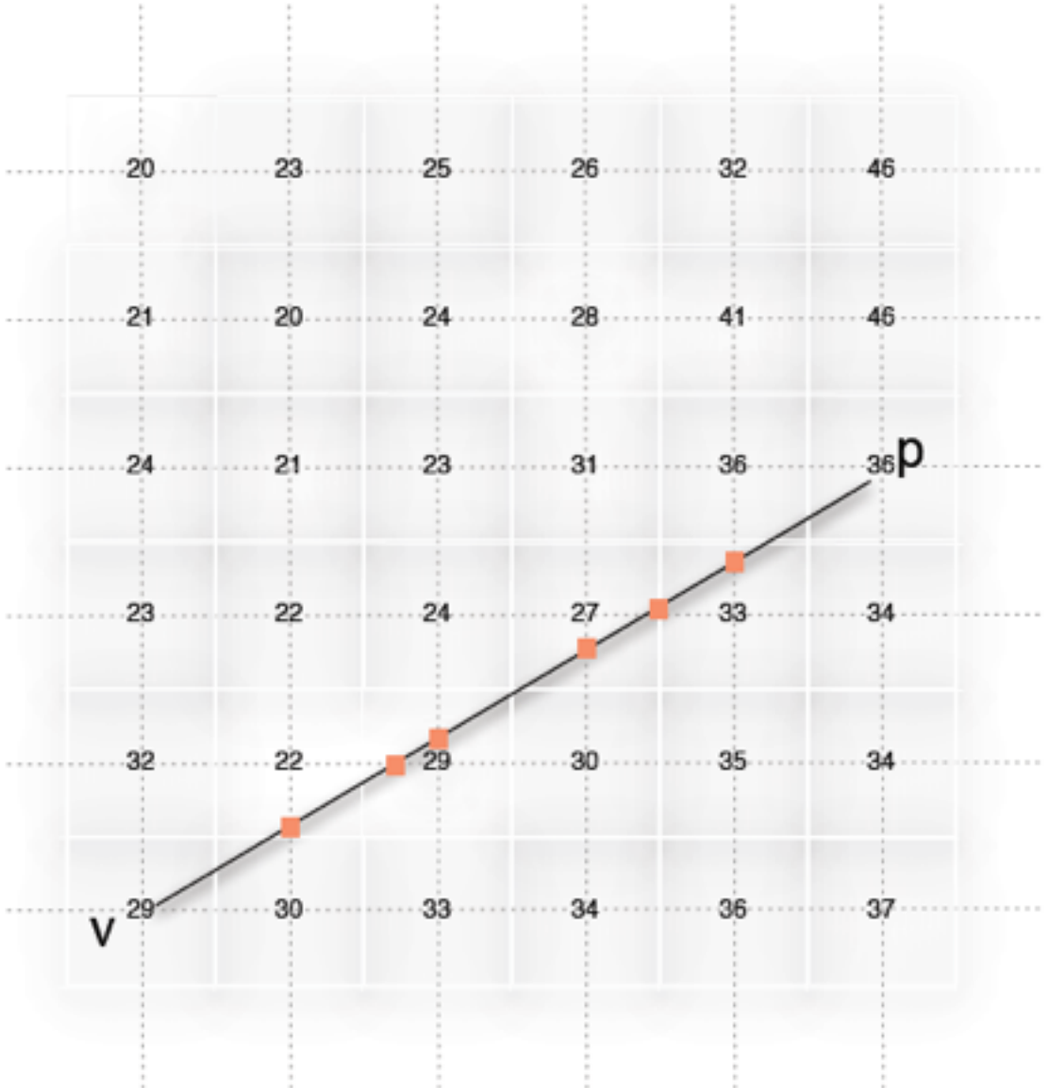
- For each p in grid
  - compute intersections between *vp* and grid lines
  - if all these points are below *vp* then *p* is visible



Basic viewshed algorithm

Input: elevation grid
Output: visibility grid, each point marked visible/invisible

- For each p in grid
  - compute intersections between *vp* and grid lines
  - if all these points are below *vp* then *p* is visible

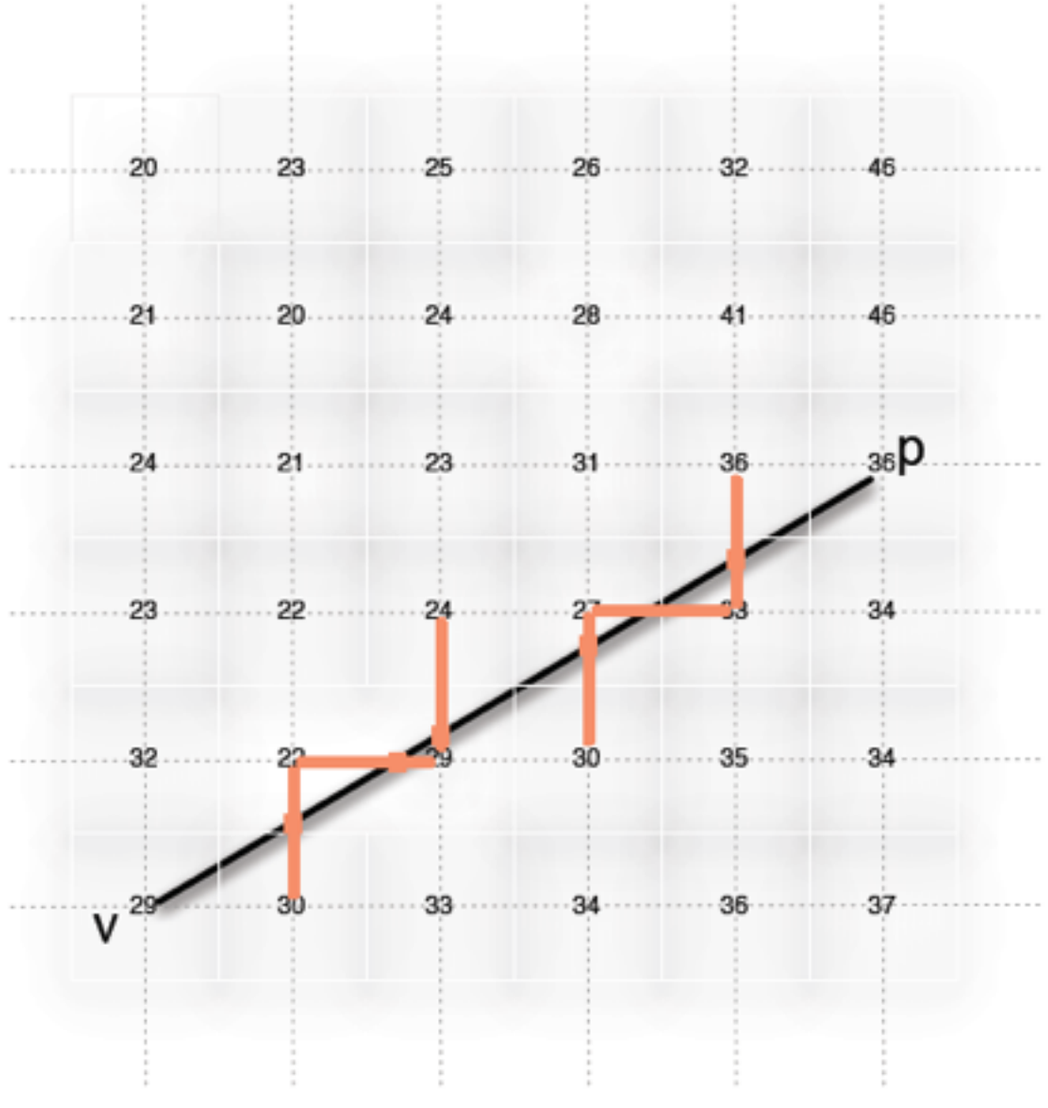


Basic viewshed algorithm

Input: elevation grid
Output: visibility grid, each point marked visible/invisible

- For each p in grid
  - compute intersections between *vp* and grid lines
  - if all these points are below *vp* then *p* is visible

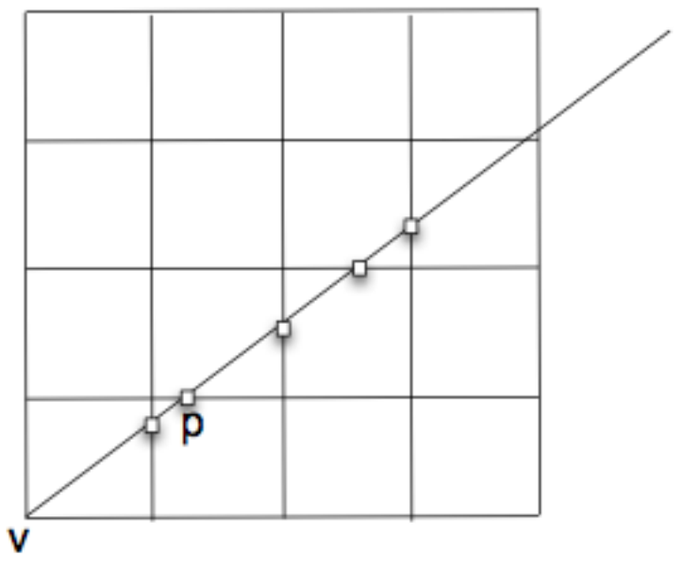
Assume grid of n points
($\sqrt{n}$ x $\sqrt{n}$)
Running time: $O(n\sqrt{n})$

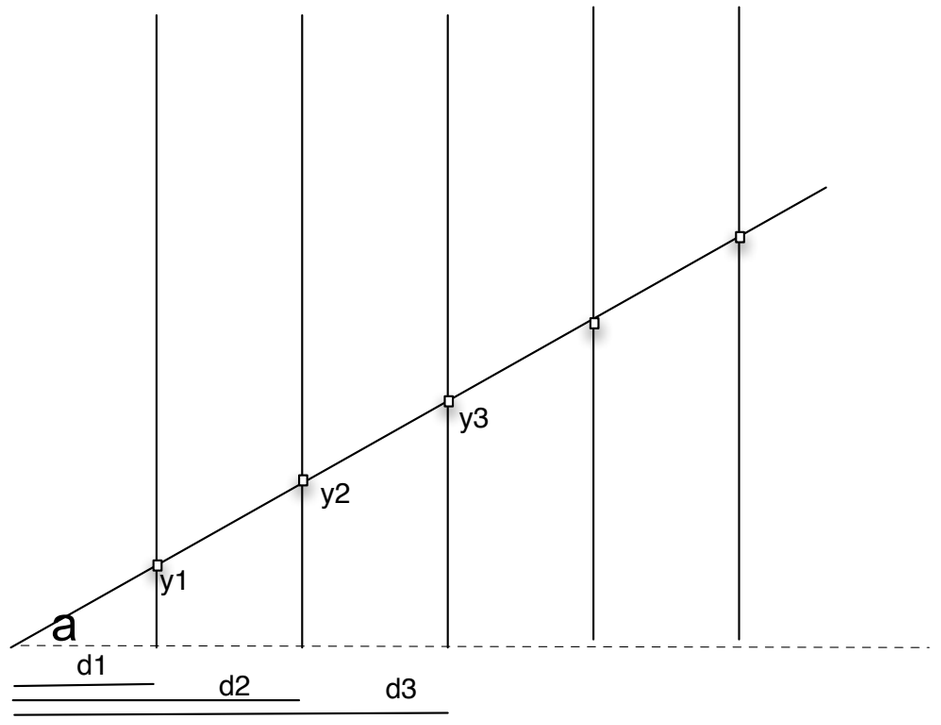


Basic viewshed algorithm

To determine if a point is visible from v:

1. Find the 2D intersections between LOS and the grid lines
2. Lift to 3D: find the height of p by linear interpolation
3. Check if verticalAngle(vp') is below verticalAngle (LOS)





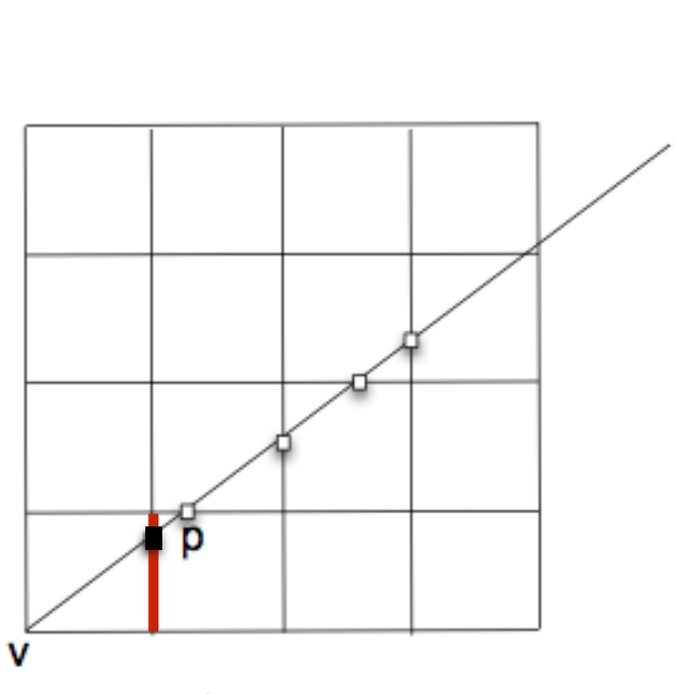
$y1 = d1 \tan a$

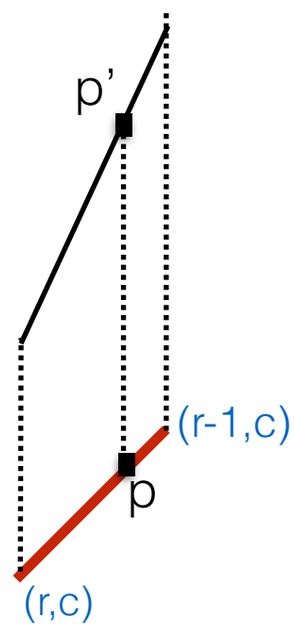
$y2 = d2 \tan a$

Basic viewshed algorithm

To determine if a point is visible from v:

1. Find the 2D intersections between LOS and the grid lines
2. Lift to 3D: find the height of p by linear interpolation
3. Check if verticalAngle(vp') is below verticalAngle (LOS)

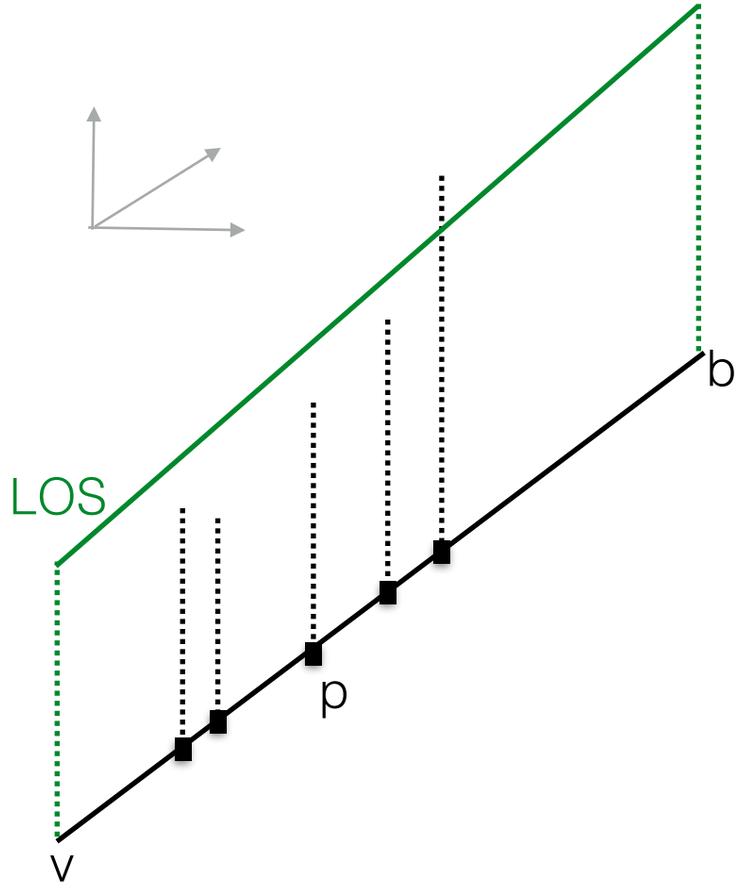


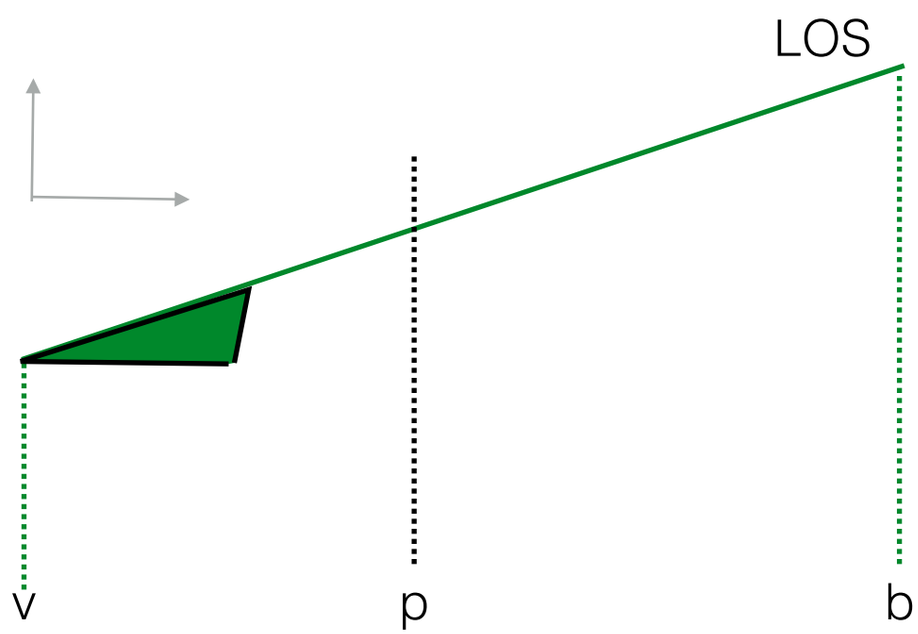


Basic viewshed algorithm

To determine if a point is visible from v:

1. Find the 2D intersections between LOS and the grid lines
2. Lift to 3D: find the height of p by linear interpolation
3. Check if $\text{verticalAngle}_v(p)$ is below $\text{verticalAngle}_v(b)$





$$\tan(\text{verticalAngle}_v(b)) = (h_b - h_v) / d(v,b)$$

Viewshed on grids

Grid of n points:
$\sqrt{n} \times \sqrt{n}$

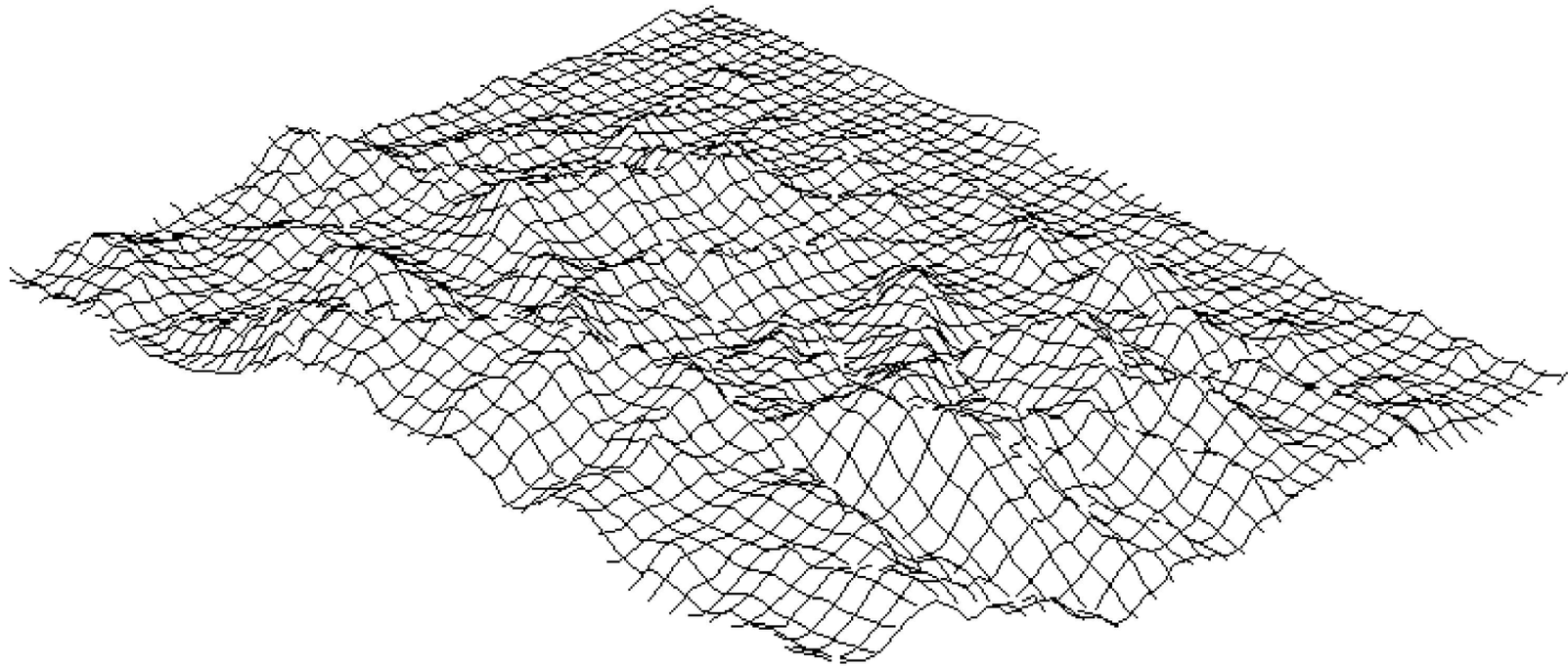
- The straightforward algorithm
  - O($n\sqrt{n}$)
  - uses linear interpolation

- Can we do better (faster)?
  - without introducing any approximation

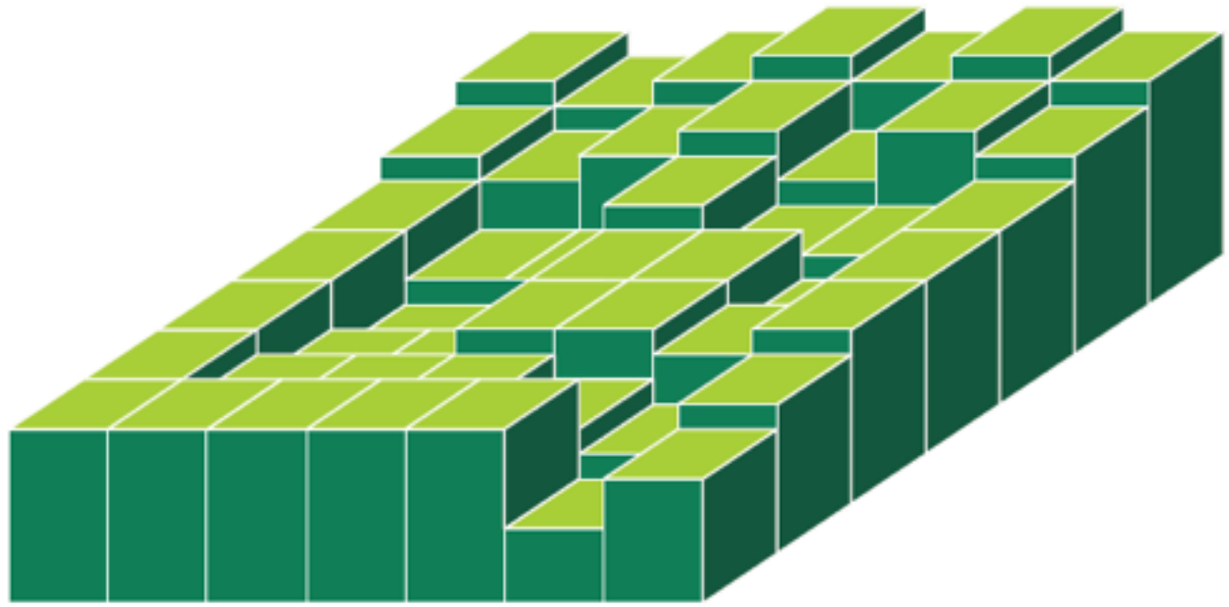
- Van Kreveld[vK'96] algorithm
  - O (n lg n)
  - uses nearest neighbor interpolation

Linear vs. nearest neighbor interpolation

20	23	25	26	32	46
21	20	24	28	41	46
24	21	23	31	36	36
23	22	24	27	33	34
32	22	29	30	35	34
29	30	33	34	36	37

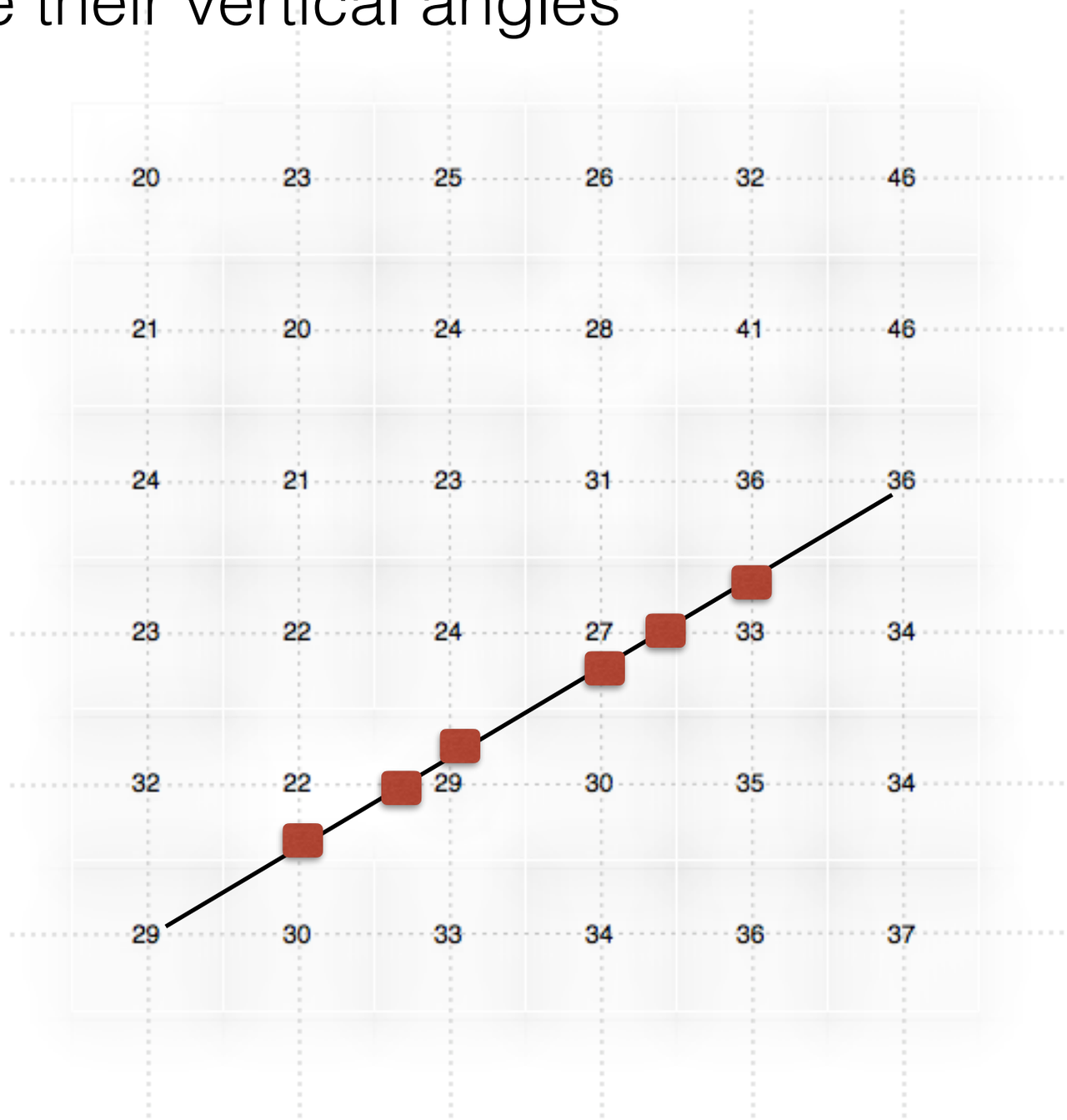


20	23	25	26	32	46
21	20	24	28	41	46
24	21	23	31	36	36
23	22	24	27	33	34
32	22	29	30	35	34
29	30	33	34	36	37

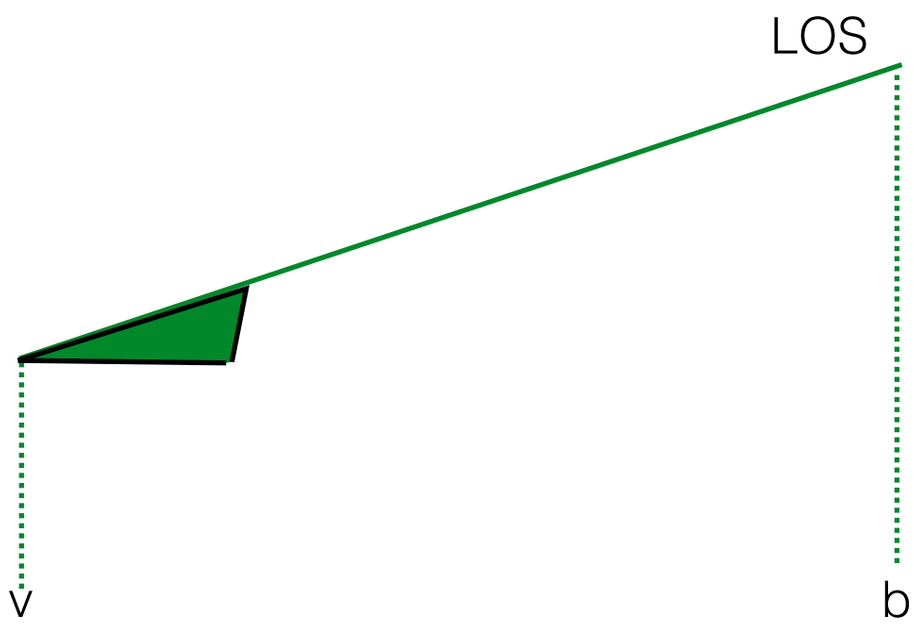


Van Kreveld's radial sweep viewshed algorithm [vK'96]

1. Compute the intersection of los with the grid
2. Compute their vertical angles

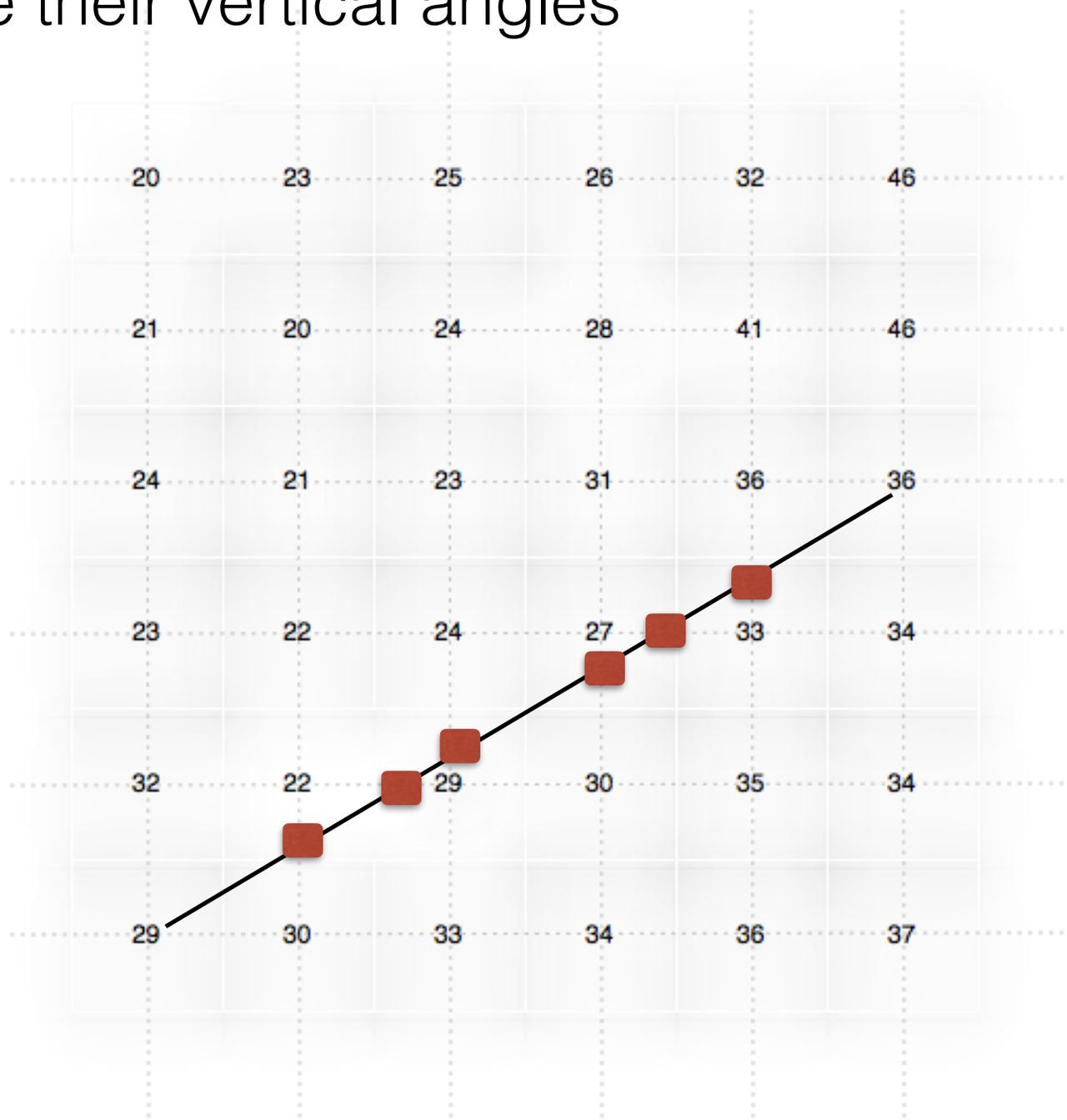


Vertical angle of b with respect to v



How b appears from v: atan (h_b - h_v) / d(v,b)

1. Compute the intersection of los with the grid
2. Compute their vertical angles



But.. a straightforward implementation leads to O(√n) per point

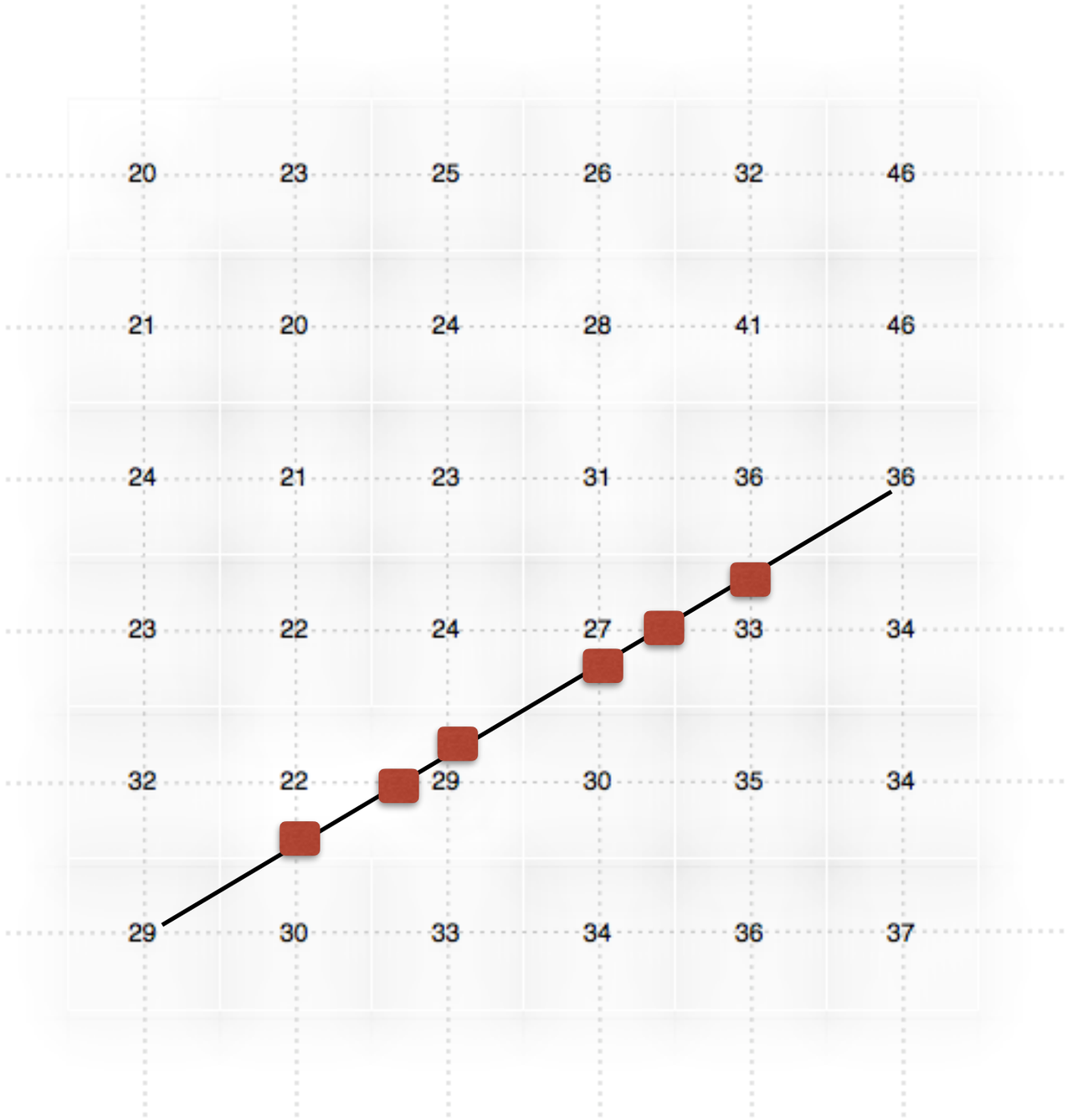
Getting below O(√n) per point

Do we need to compute the O(√n) intersection points and their vertical angles **from scratch** for every point?

Van Kreveld's O(n lg n) viewshed algorithm

- **Idea 1**: The line-of-sight to two nearby points share a lot of the same information. Re-use.
    - instead of computing visibility of points in row-column order, compute in radial order

- **Idea 2**: Assume that the vertical angle for a cell is the same throughout the cell
    - i.e. nearest neighbor interpolation instead of linear

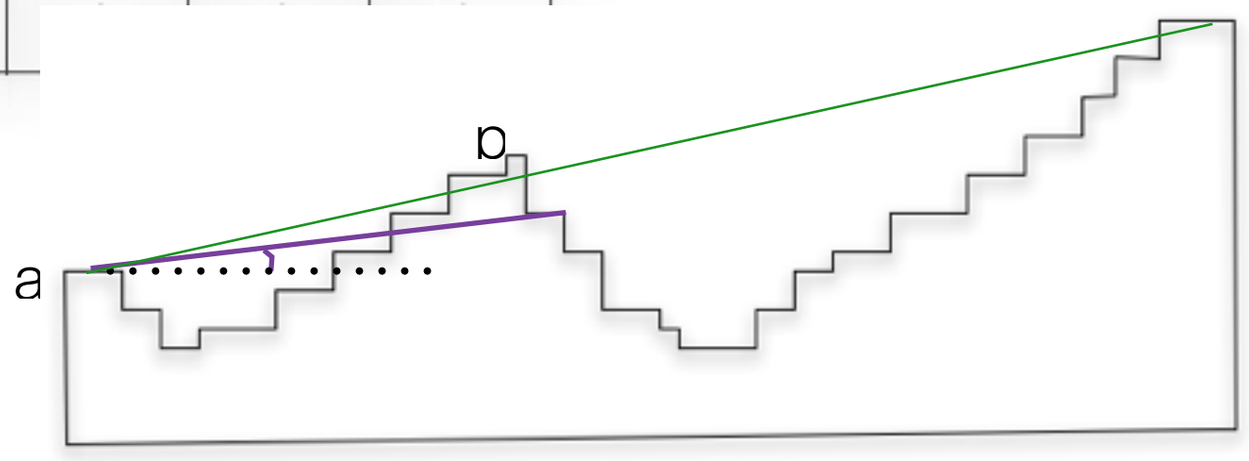

20 23 25 26 32 46
21 20 24 28 41 46
24 21 23 31 36 36
23 22 24 27 33 34
32 22 29 30 35 34
29 30 33 34 36 37

20	23	25	26	32	46
21	20	24	28	41	46
24	21	23	31	36	36
23	22	24	27	33	34
32	22	29	30	35	34
29	30	33	34	36	37

Assume that the vertical angle for a cell is the same throughout the cell

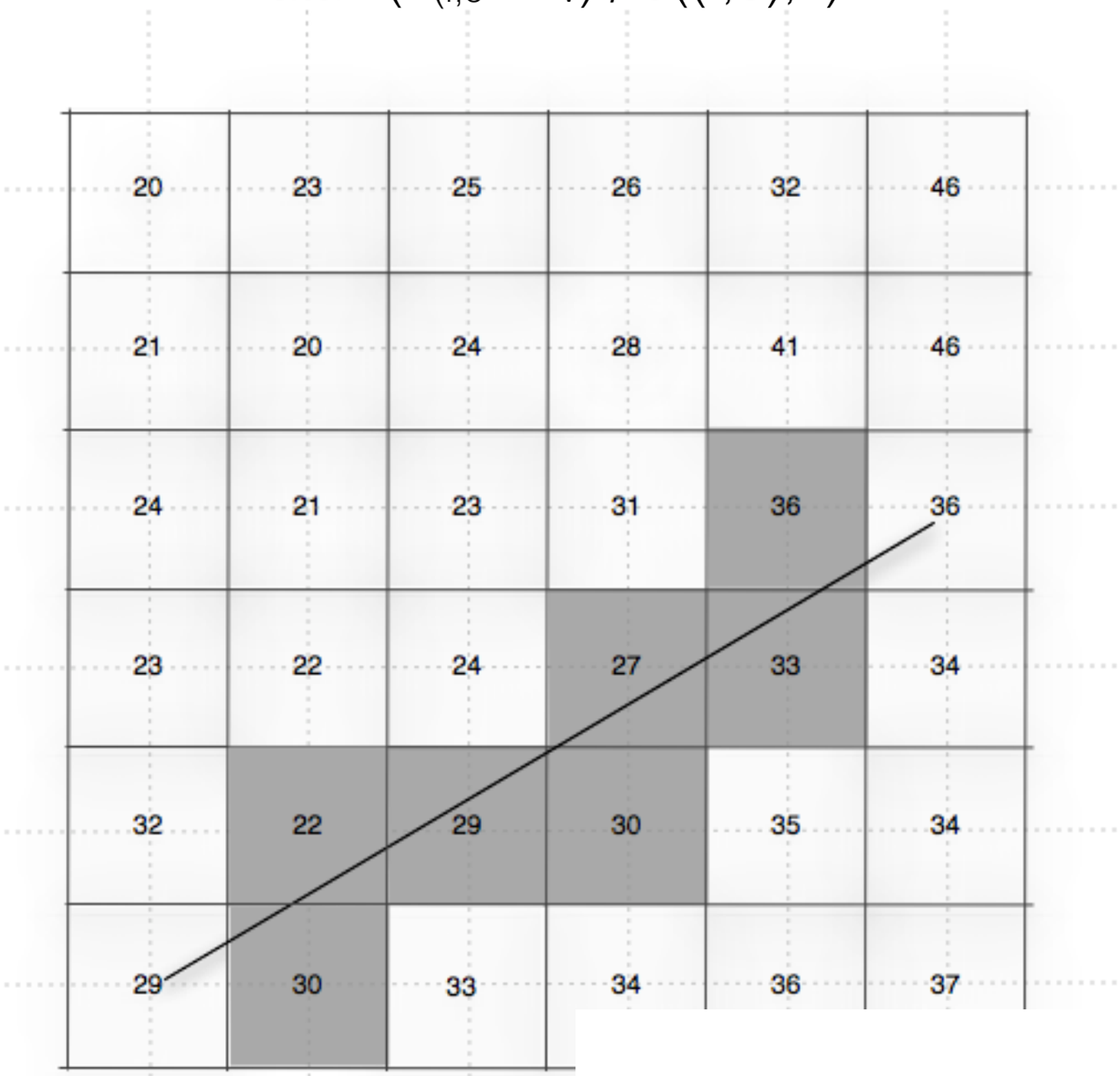
atan $(h_{(r,c} - h_v) / d((r,c),v)$

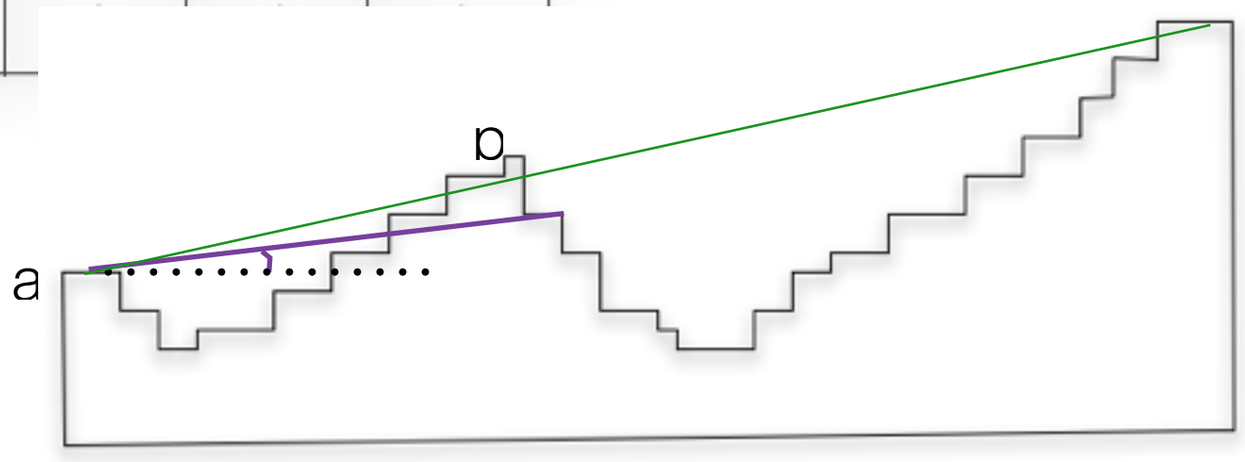
20	23	25	26	32	46
21	20	24	28	41	46
24	21	23	31	36	36
23	22	24	27	33	34
32	22	29	30	35	34
29	30	33	34	36	37



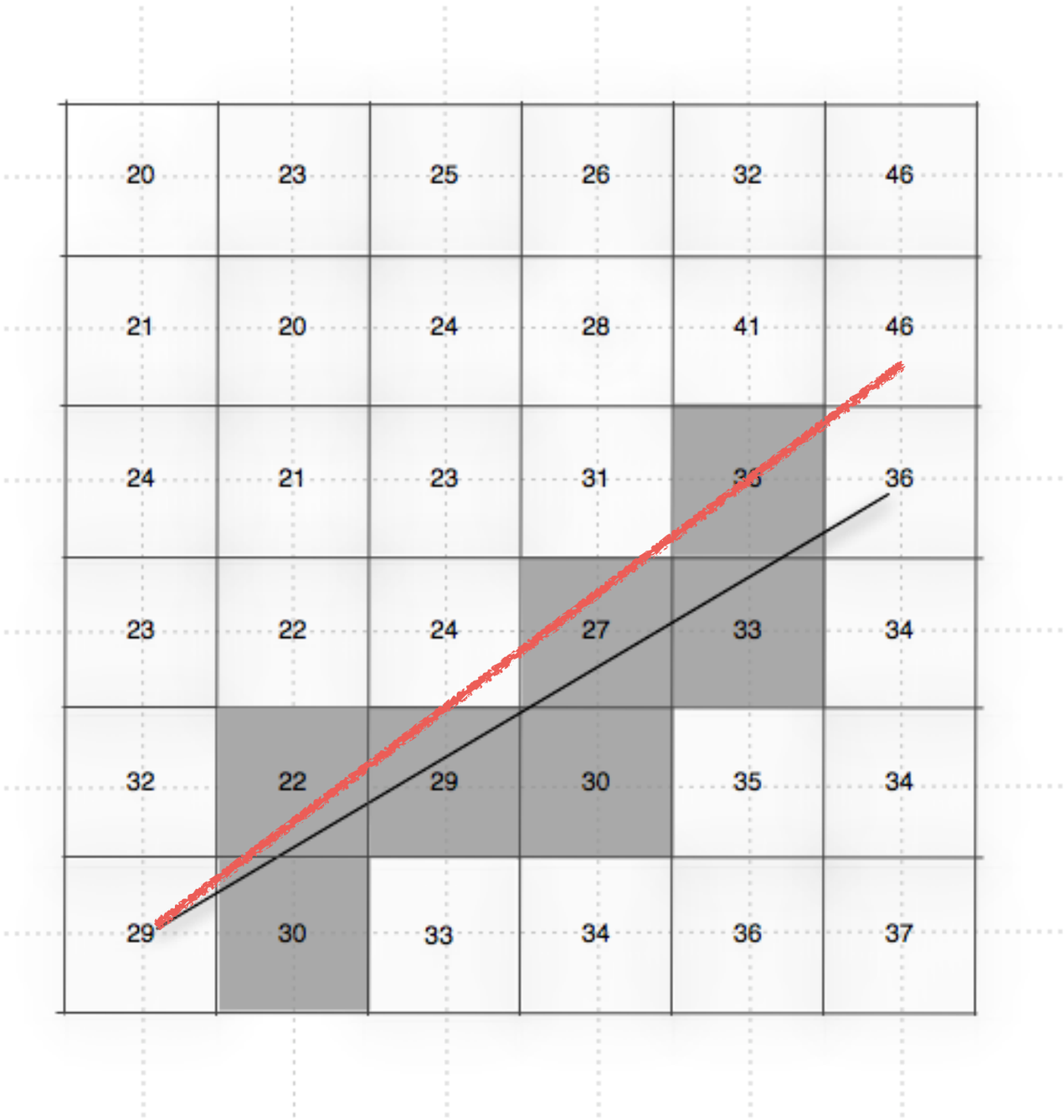
Assume that the vertical angle for a cell is the same throughout the cell

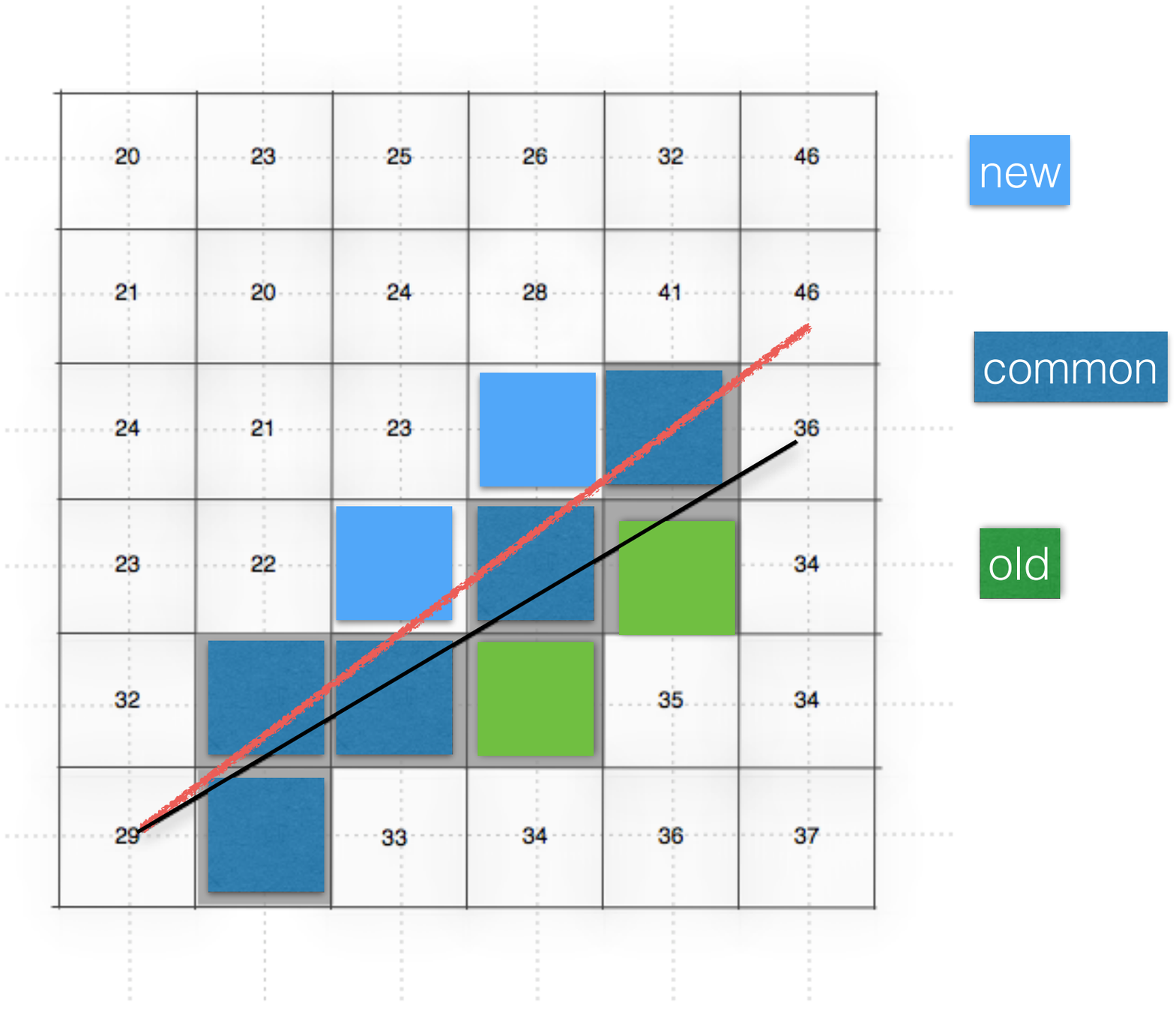
atan $(h_{(r,c} - h_v) / d((r,c),v)$

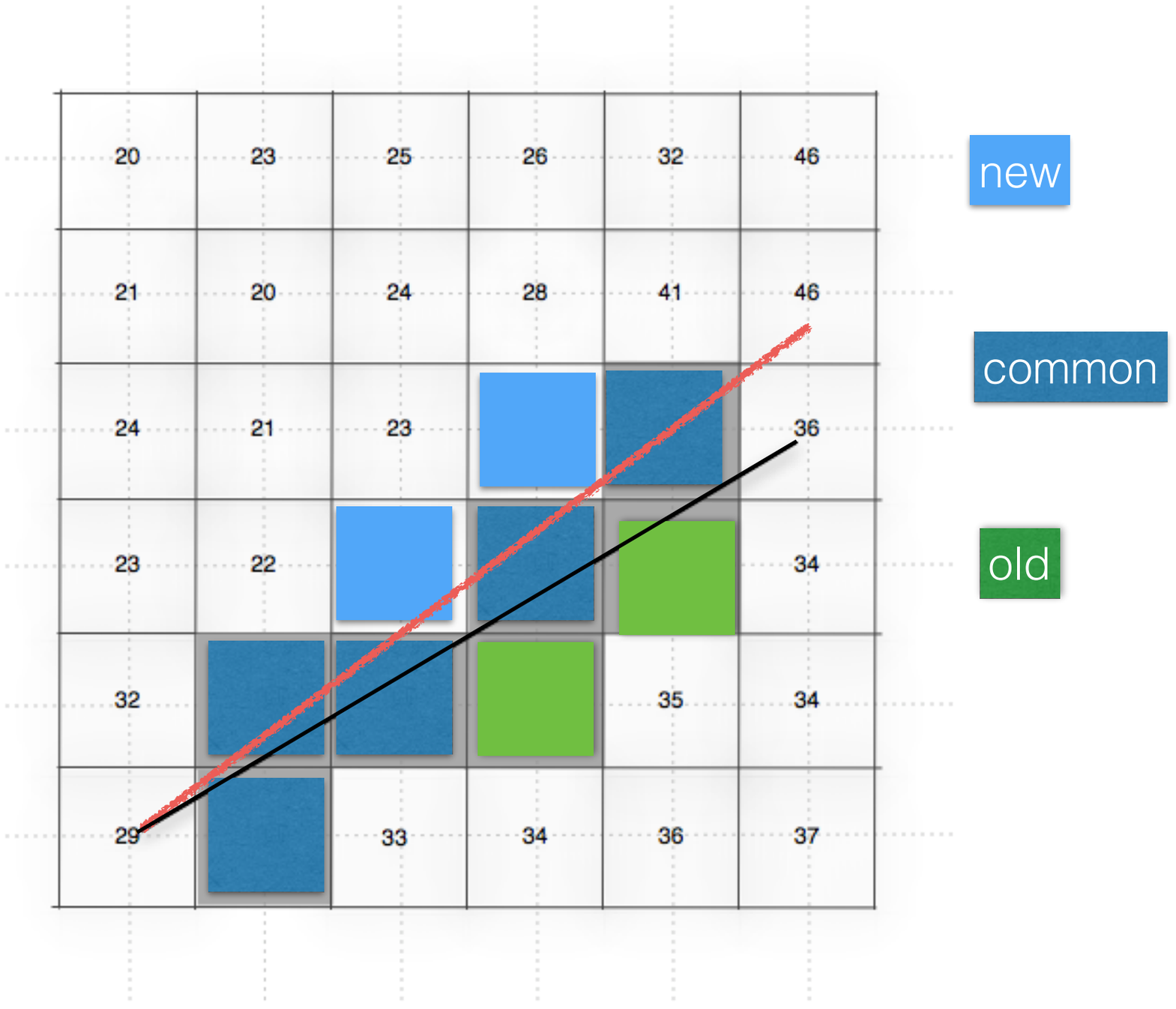




Idea: Two nearby points intersect a lot of the same cells




20 23 25 26 32 46
21 20 24 28 41 46
24 21 23 36
23 22 34
32 35 34
29 33 34 36 37
new
common
old



We are going to compute visibility of points in radial order around v

Viewshed in row-column order

- for i = 0; i< rows; i++
  - for j=0; j< cols; j++
    - find if (i,j) is visible from v

Viewshed in radial order

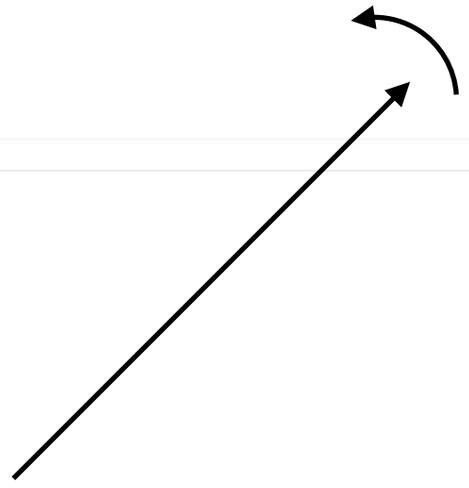
- sort points (i,j) by radial angle of (i,j)
- for each point (i,j) in order:
  - find if (i,j) is visible from v

Viewshed in radial order

- sort points (i,j) by radial angle of (i,j)
- for each point (i,j) in order:
  - find if (i,j) is visible from v

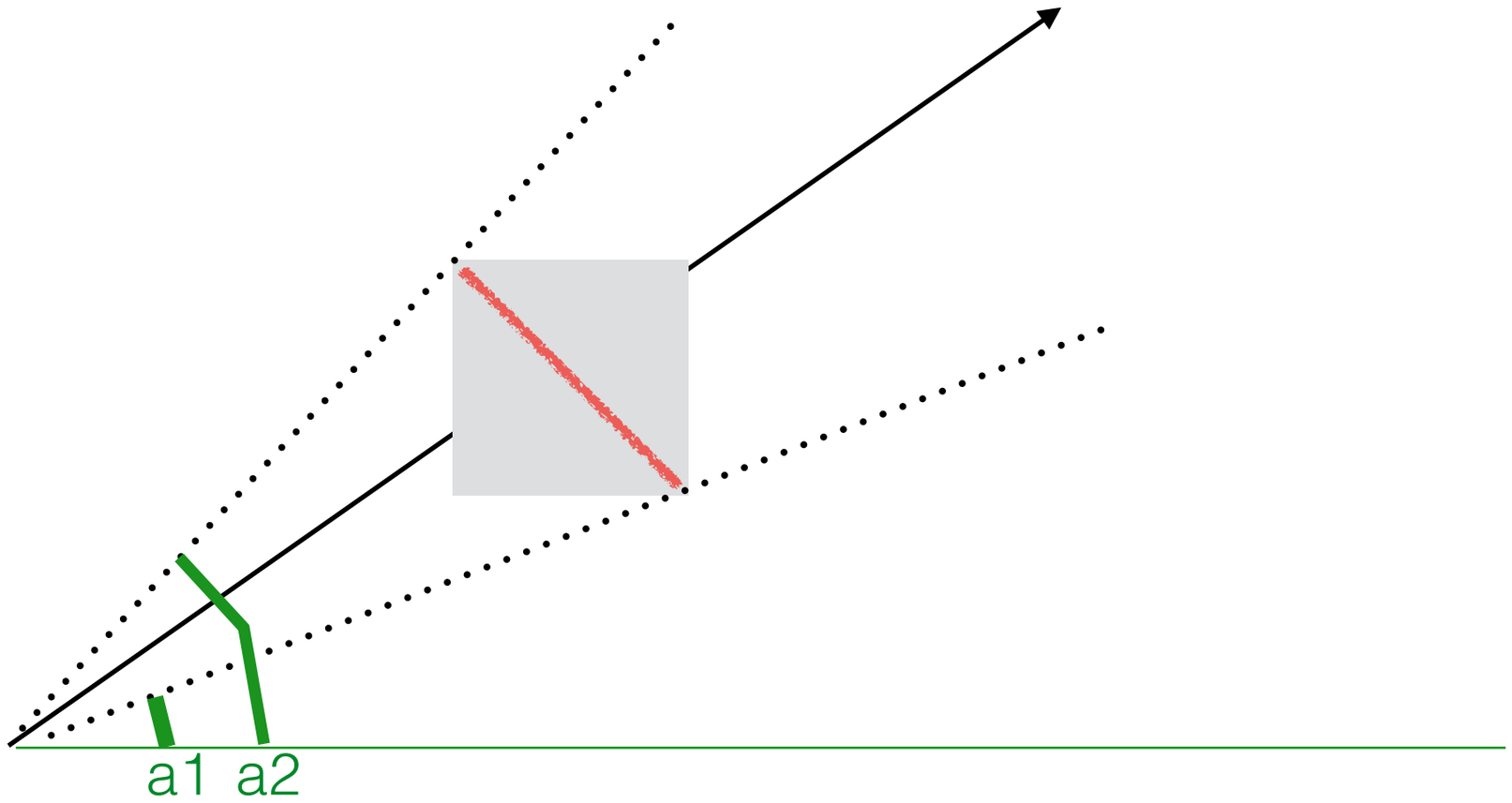
Some of the cells that intersected the previous point will also intersect the current point; re-use them HOW???

- **Radial sweep**
  - rotate a ray radially around v
  - respond to “interesting” events

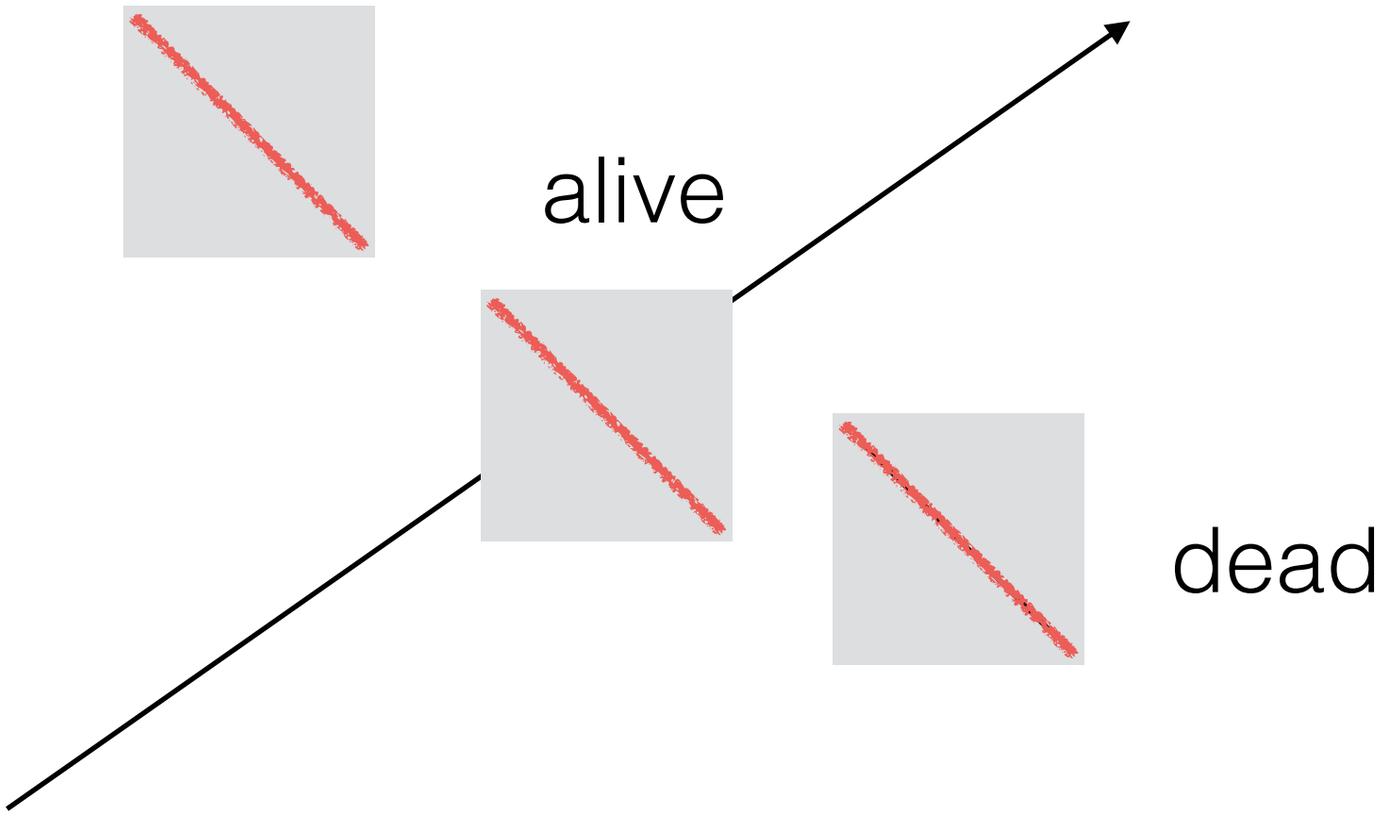


- What are the events?
  - when ray hits a grid point (i,j): determine if (i,j) is visible

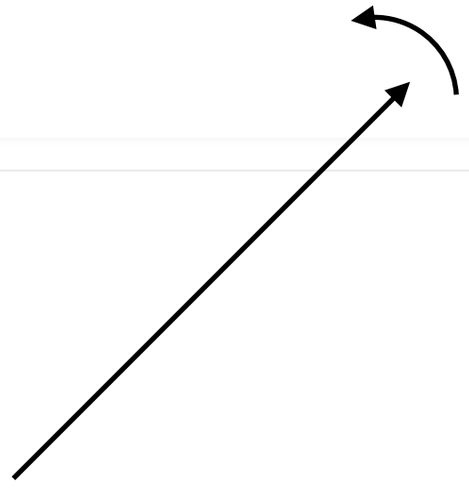
- Framing as moving the ray around v allows to express the cells that intersect the ray in terms of their radial angle



- Ray hits a1: cell alive
- Ray hits a2: cell dies
- Ray between a1 and a2: cell intersects ray


alive
dead

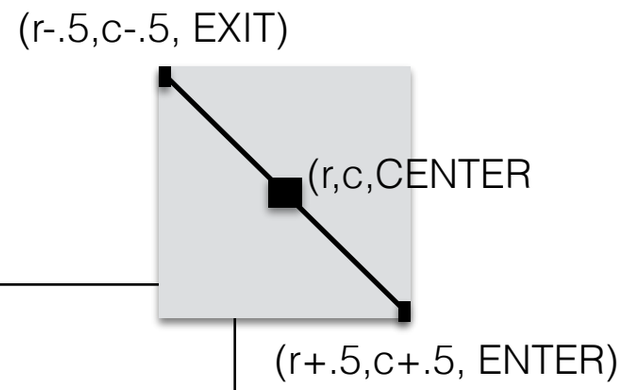
- **Radial sweep**
  - rotate a ray radially around v
  - respond to “interesting” events



- What are the events?
  - when ray hits a grid point (i,j): determine if (i,j) is visible
  - when ray hits ENTER(i,j): cell (i,j) becomes active
  - when ray hits EXIT(i,j): cell (i,j) becomes inactive

everything that happens in between ENTER(i,j) and EXIT(i,j): cell (i,j) will be active

Van Kreveld's radial sweep algorithm

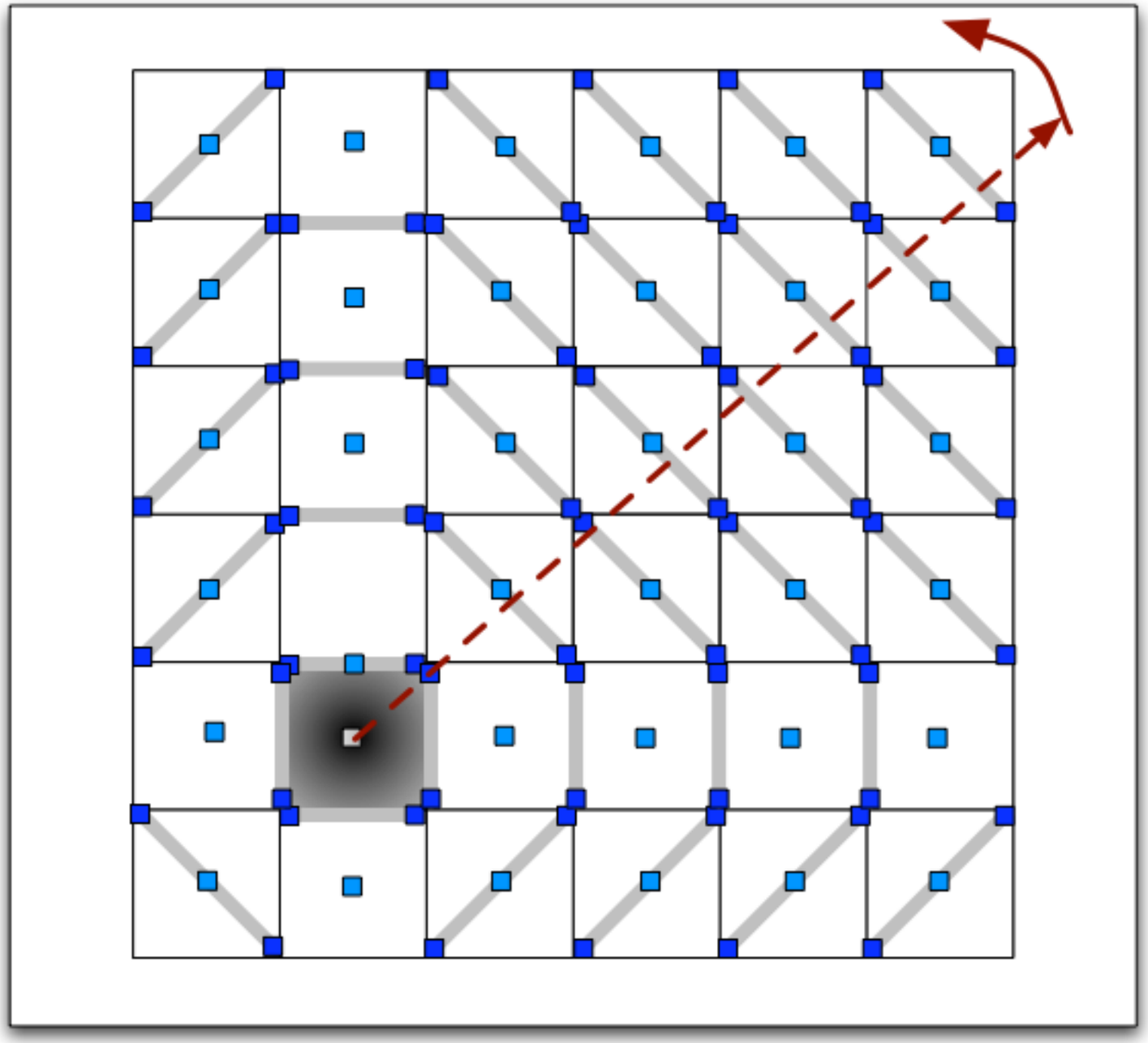


- For each point (i,j): compute its ENTER, CENTER, EXIT events
- Sort all events by radial angle wrt v
- initialize AS to contain all cells that are active at angle=0
- For next event (r,c, type) in radial order
  - if type is ENTER: //cell becomes active
    - insert cell(r,c) in AS
  - if event is EXIT: //cell stops being active
    - delete cell(r,c) from AS
  - if event is CENTER:

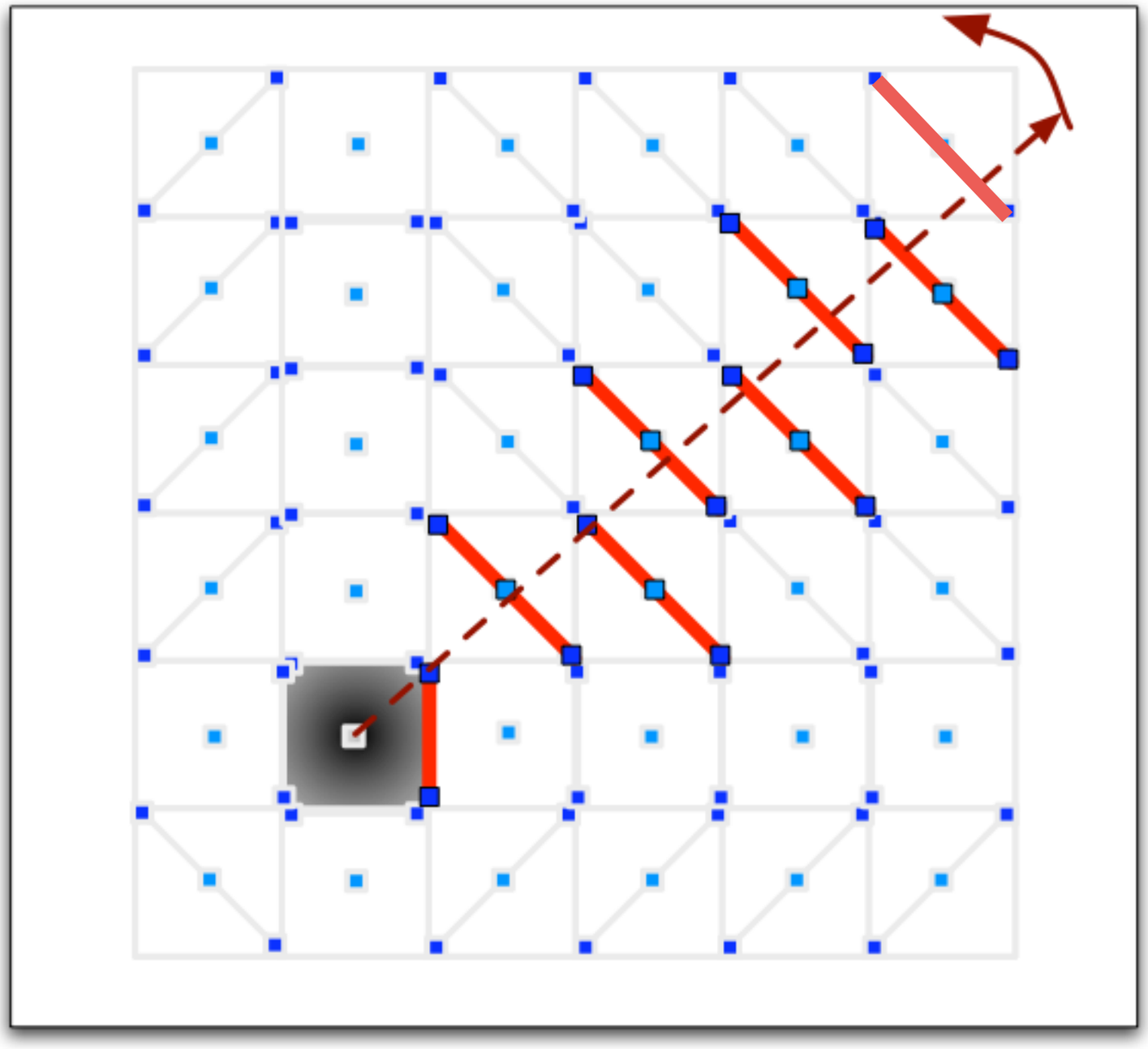
    //CLAIM: all cells that intersect the los from v to (r,c) must be in the AS

    - use AS to find maximum verticalAngle of all cells between v and cell(r,c)
    - if this angle is below verticalAngle(r,c) then (r,c) is visible; otherwise (r,c) is invisible

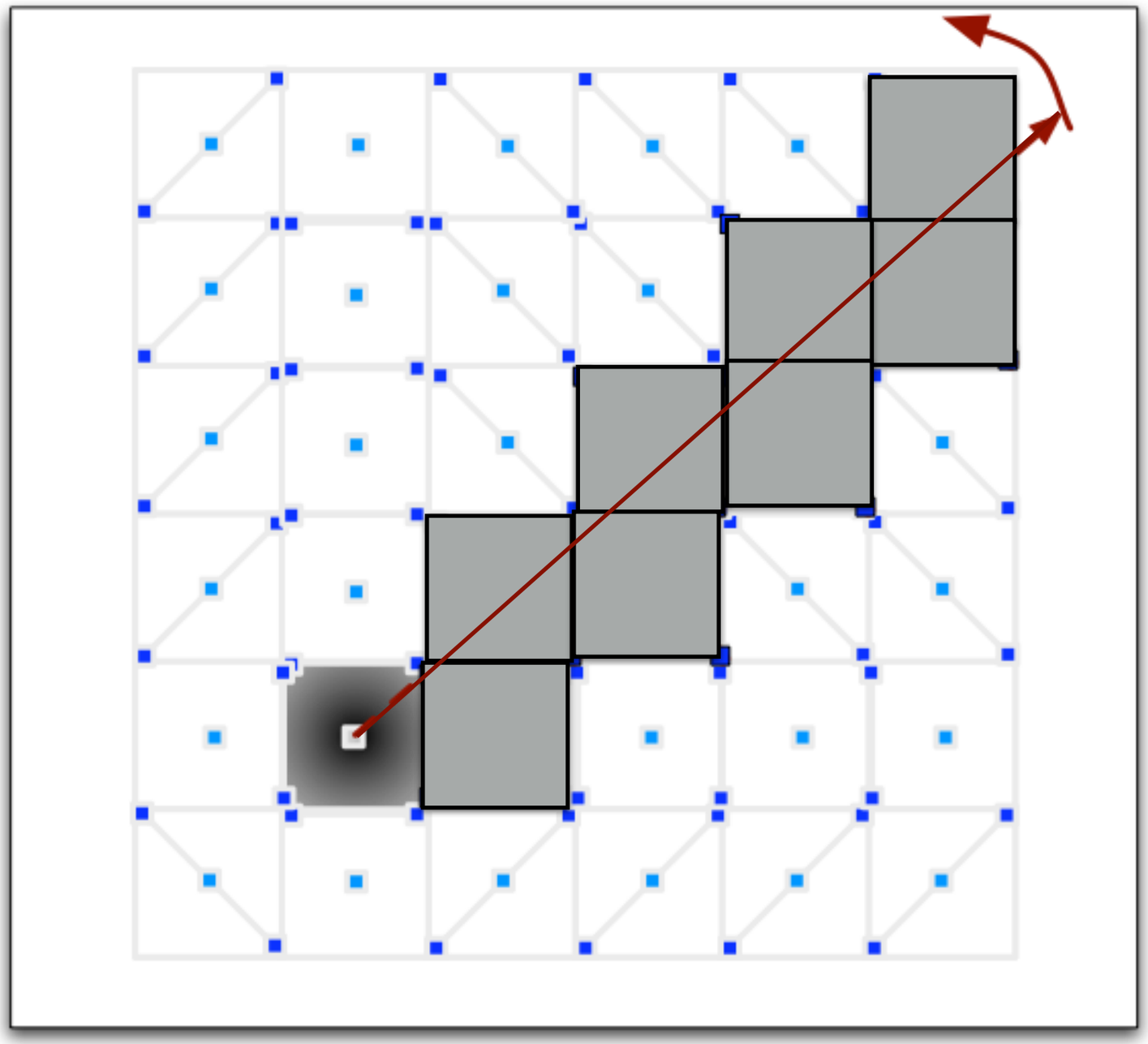
The 3 events corresponding to each cell



For an arbitrary position of the ray, all cells that it intersects will be in AS



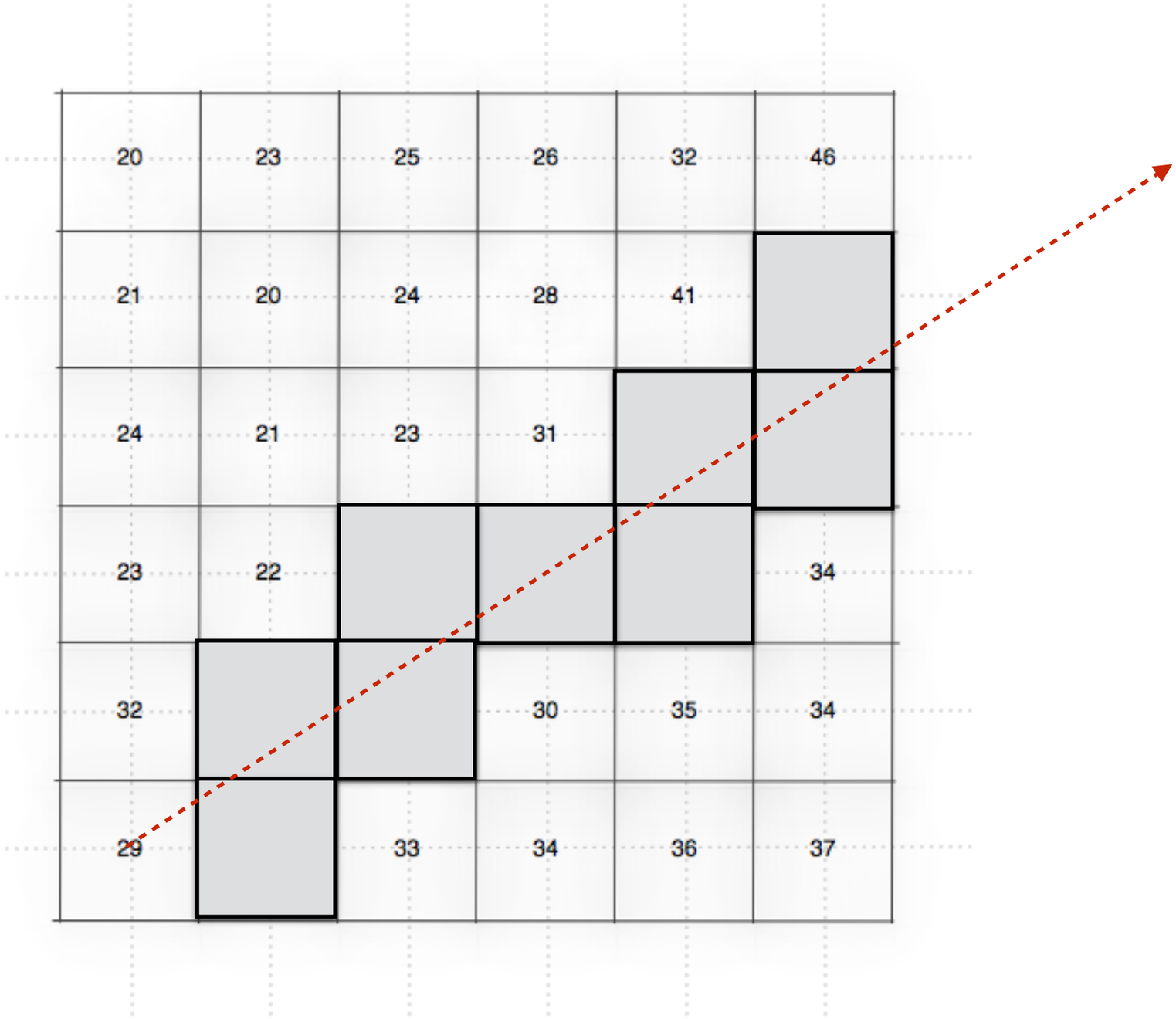
For an arbitrary position of the ray, all cells that it intersects will be in AS



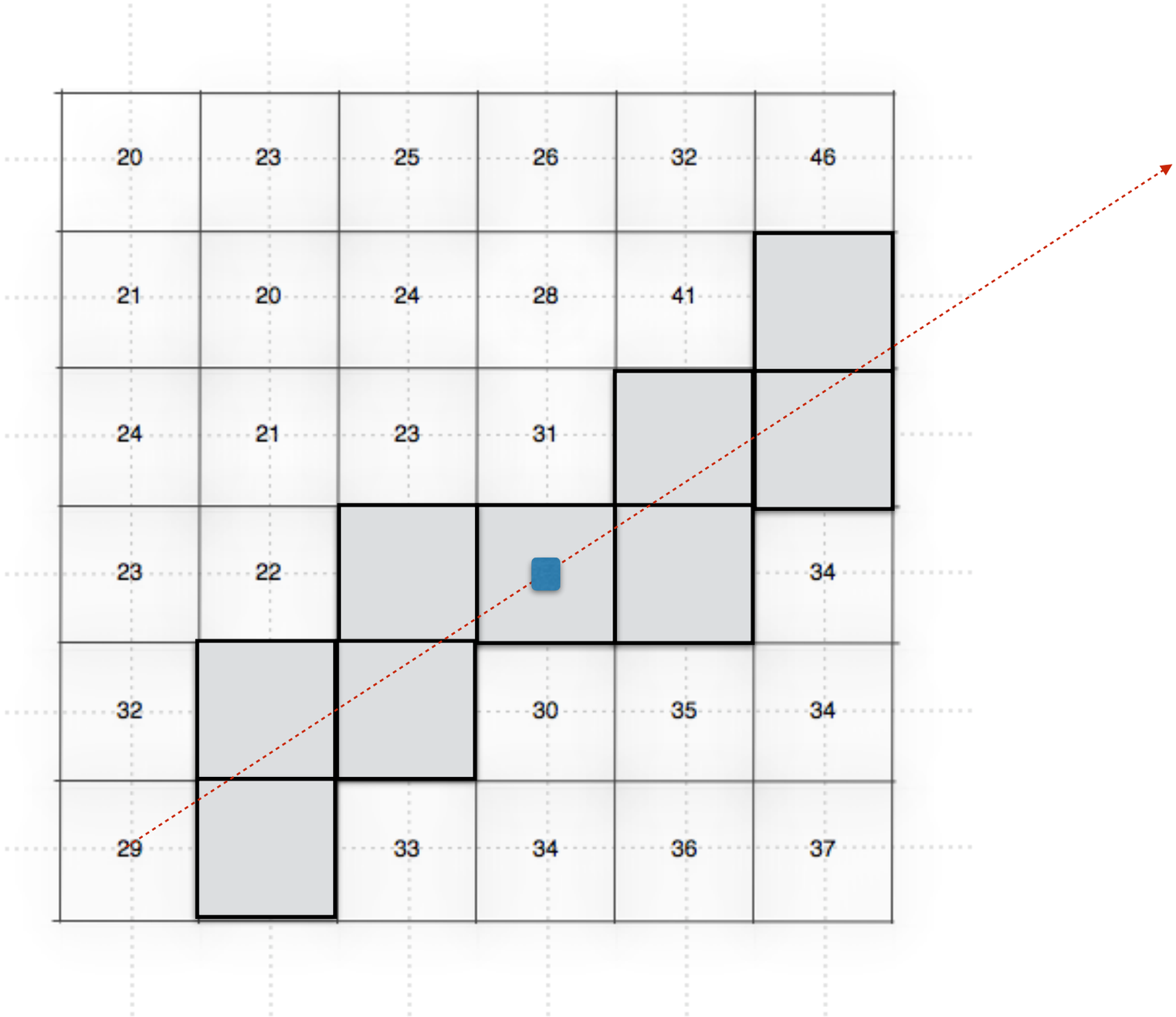
Analysis

- What's a good data structure for the AS?
  - Needs to be able to insert and delete cell
  - Find vertical angles of all active cells between v and given (r,c)

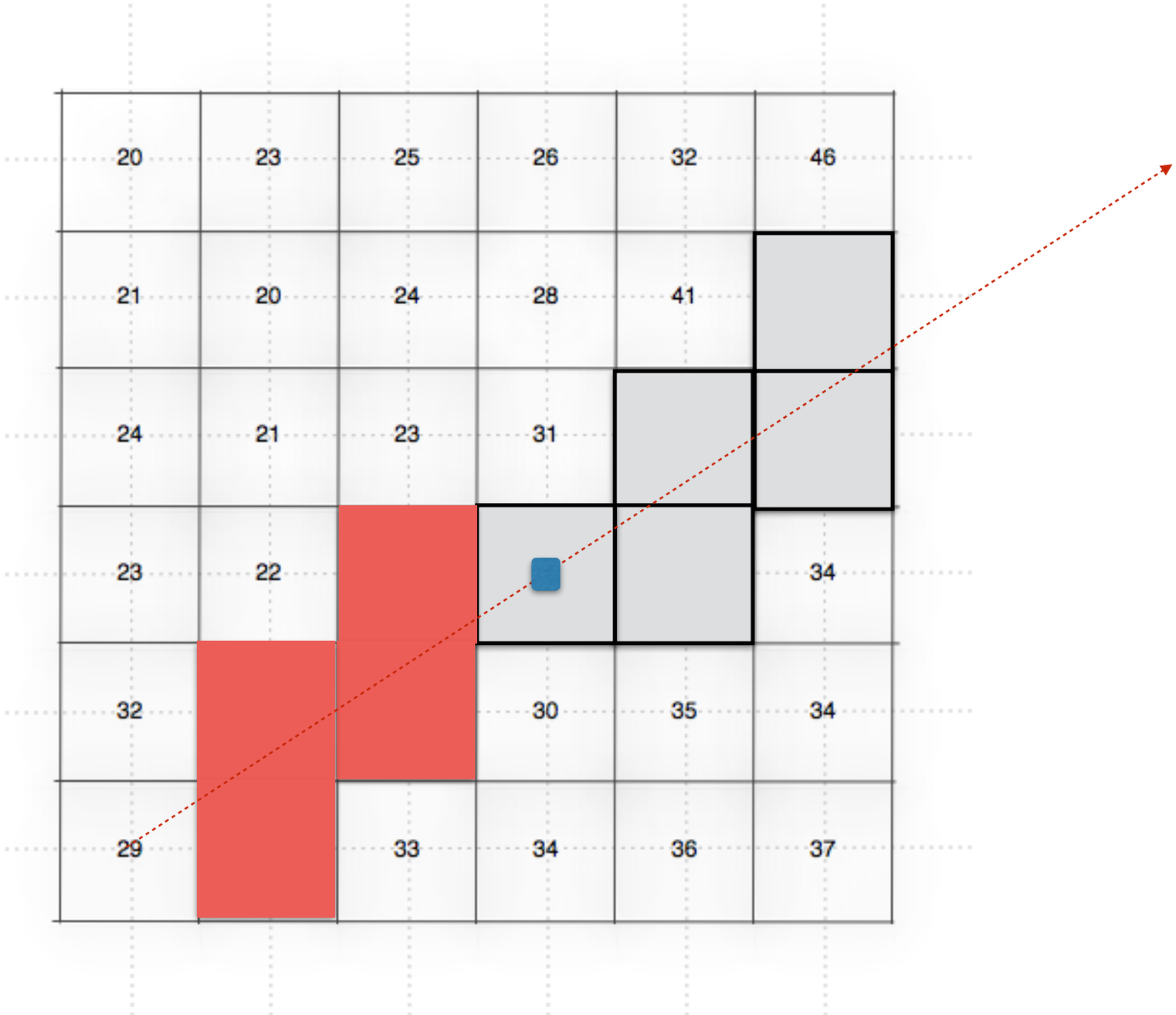
For an arbitrary position of the ray, all cells that it intersects will be in AS



When process CENTER(r,c): , all cells that intersect ray will be in AS



When process CENTER(r,c): , all cells that intersect ray will be in AS



Want only the cells that are in between v and (r,c)

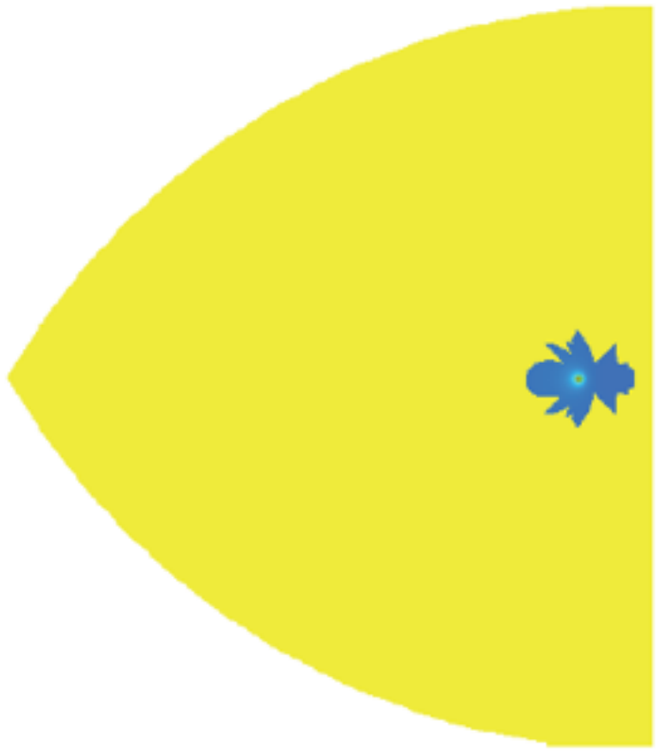
Computing viewsheds

Grid of n points:
$\sqrt{n} \times \sqrt{n}$

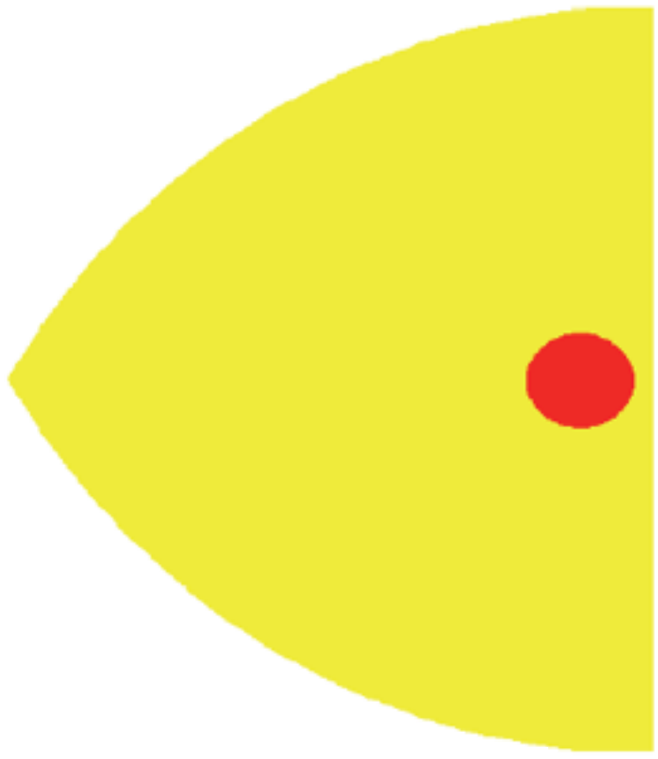
- Straightforward algorithm
  - O($n\sqrt{n}$)
  - Uses linear interpolation
  - Can be adapted to other interpolations
- Radial sweep approach
  - O(n lg n)
  - Uses nearest neighbor interpolation
  - Not easy to adapt: crucially exploits that cells are “flat”
  - Nearest neighbor produces some artifacts
- NEXT: Concentric sweep and horizons

Accuracy!!

test grid: hemisphere



viewshed with NN interpolation



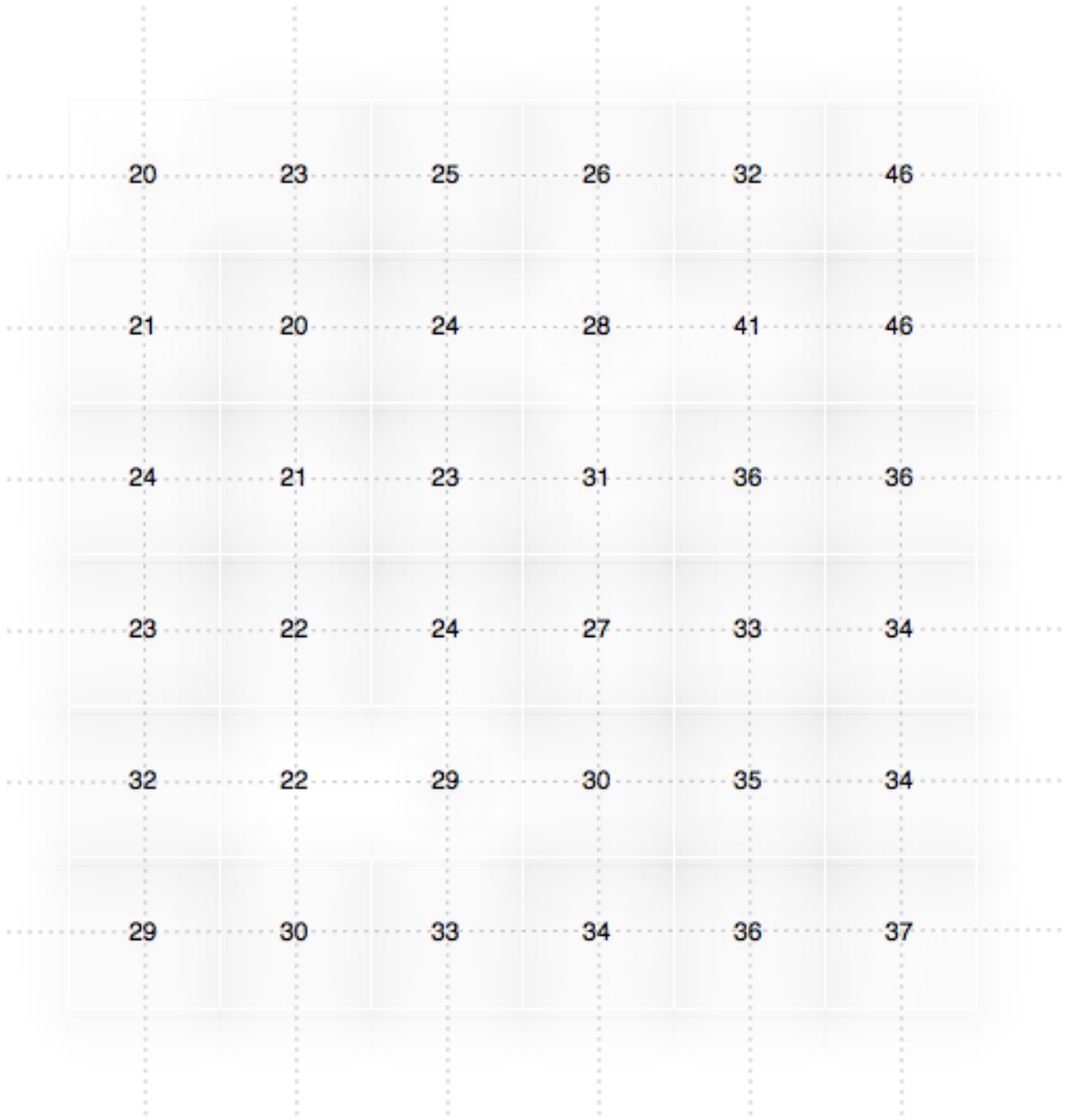
viewshed with linear interpolation

Viewsheds and horizons

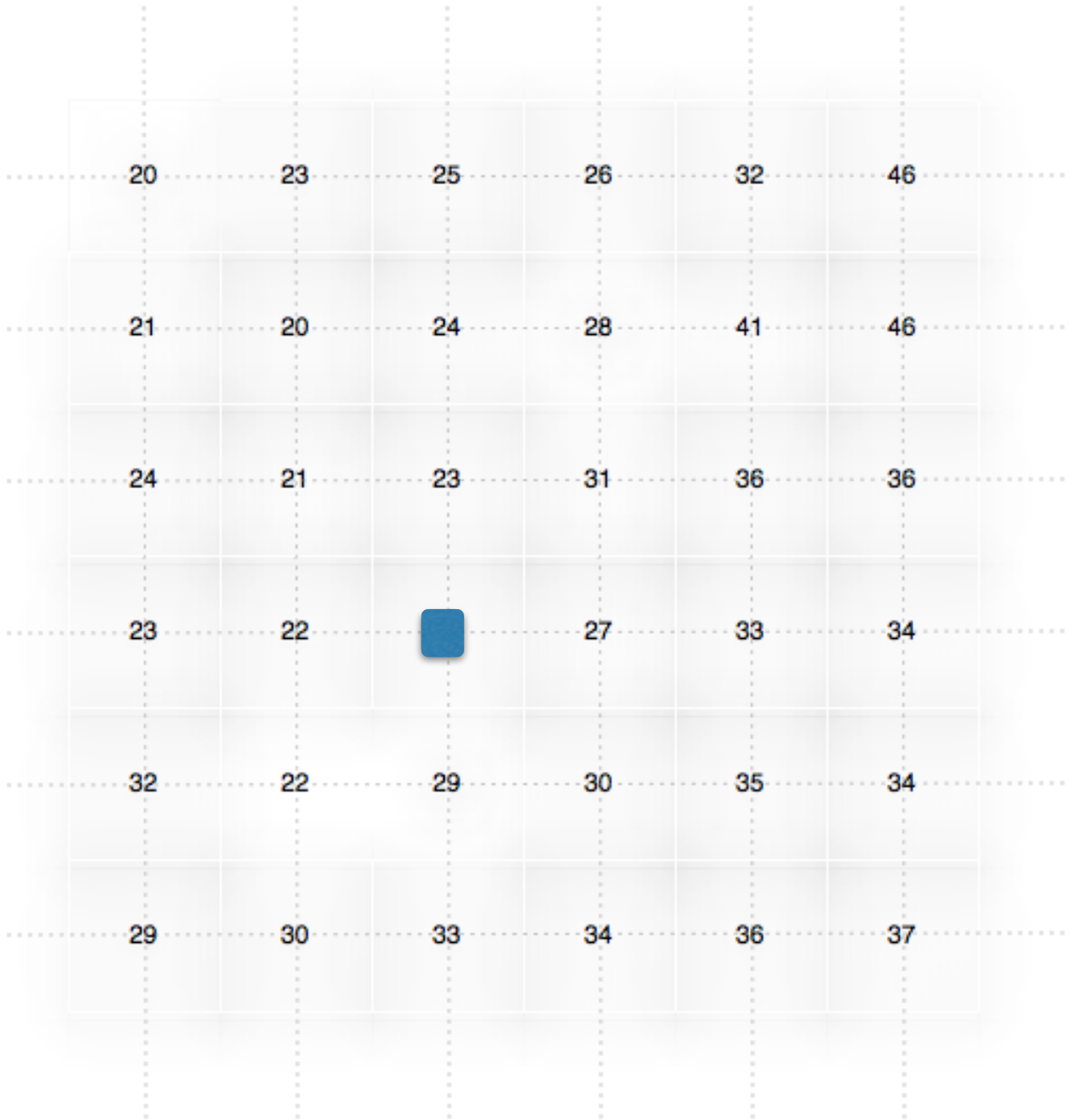
Ingredient 1: concentric sweep

Ingredient 2: horizons

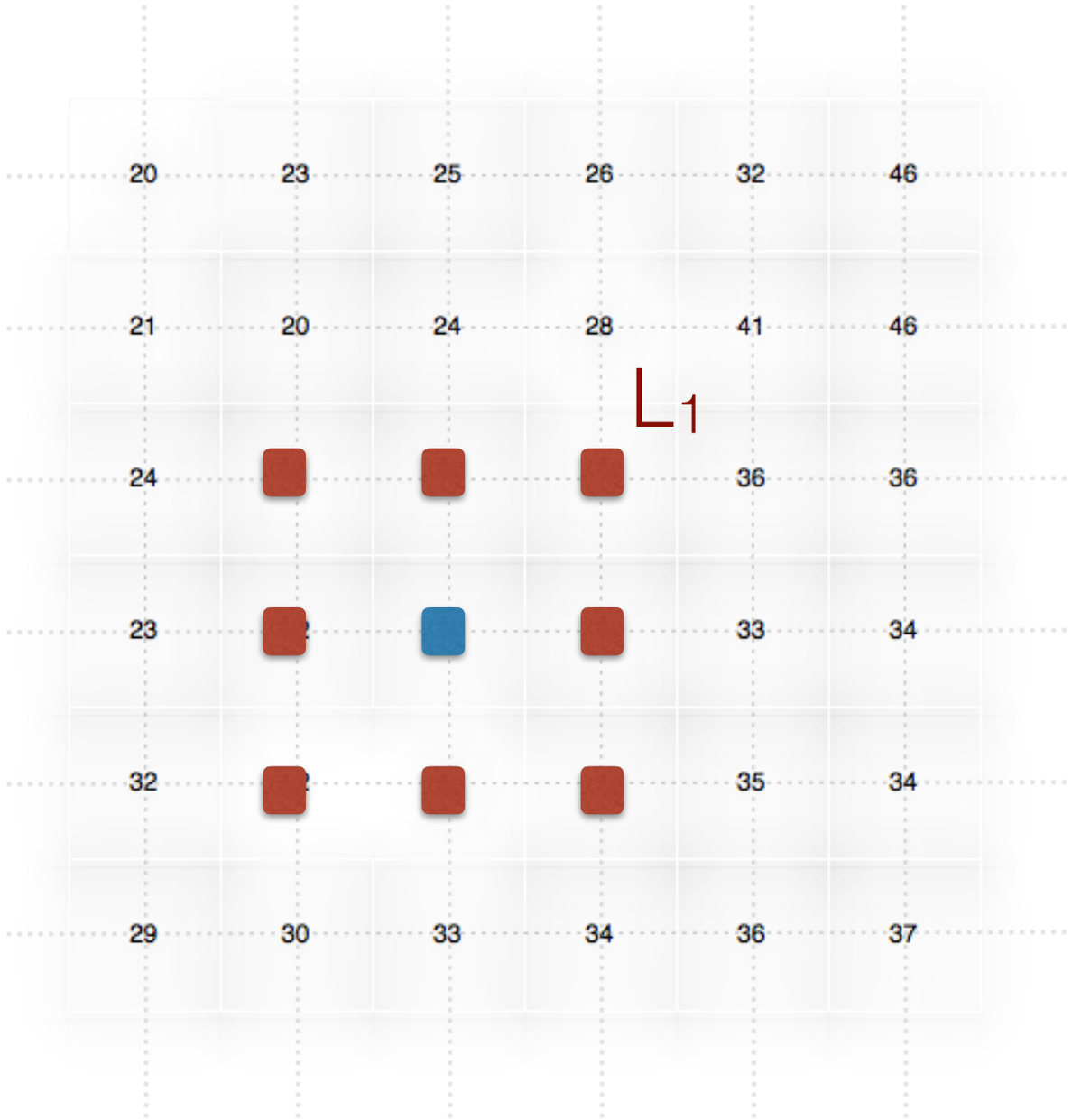
Concentric sweep



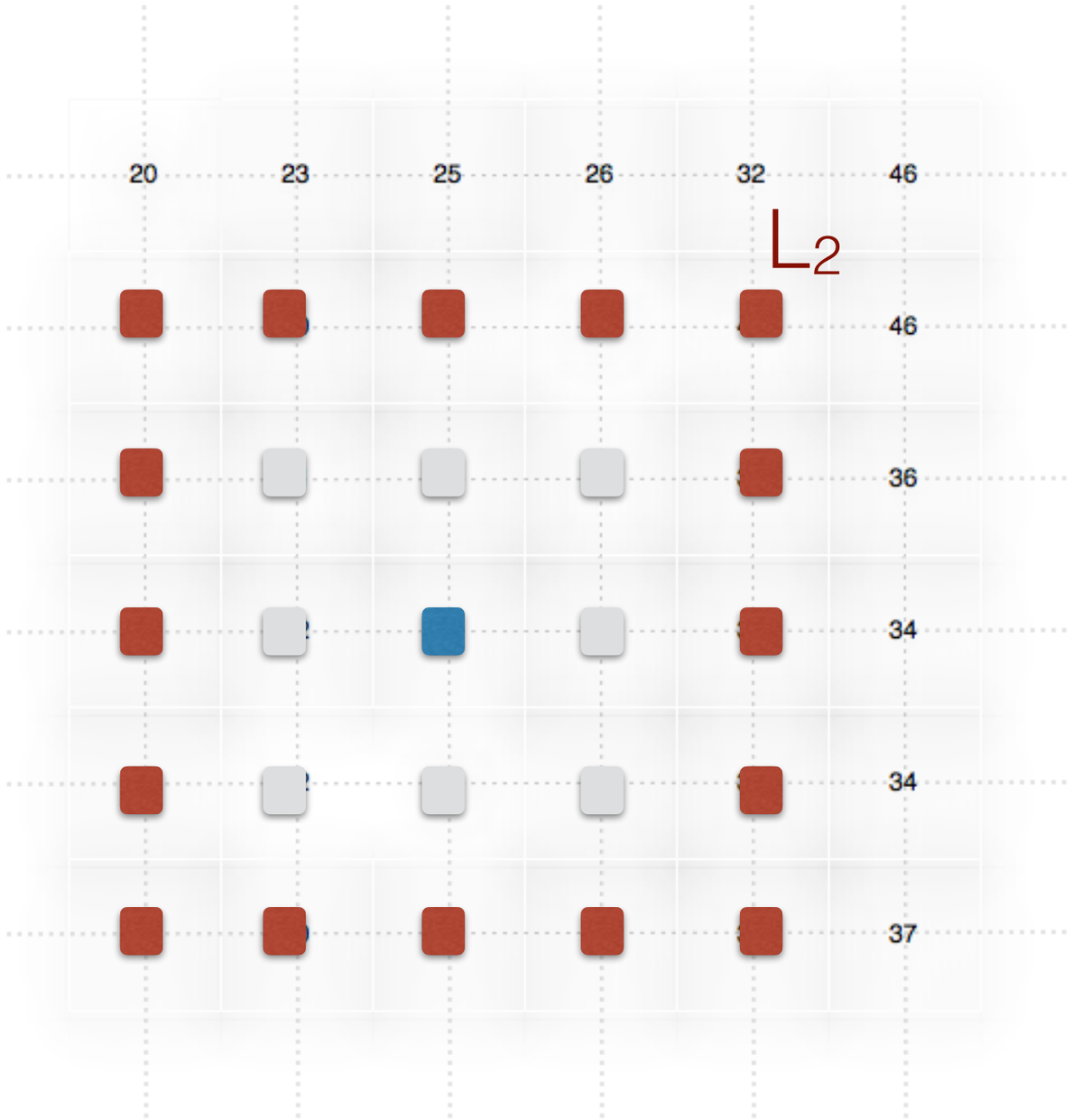
Concentric sweep



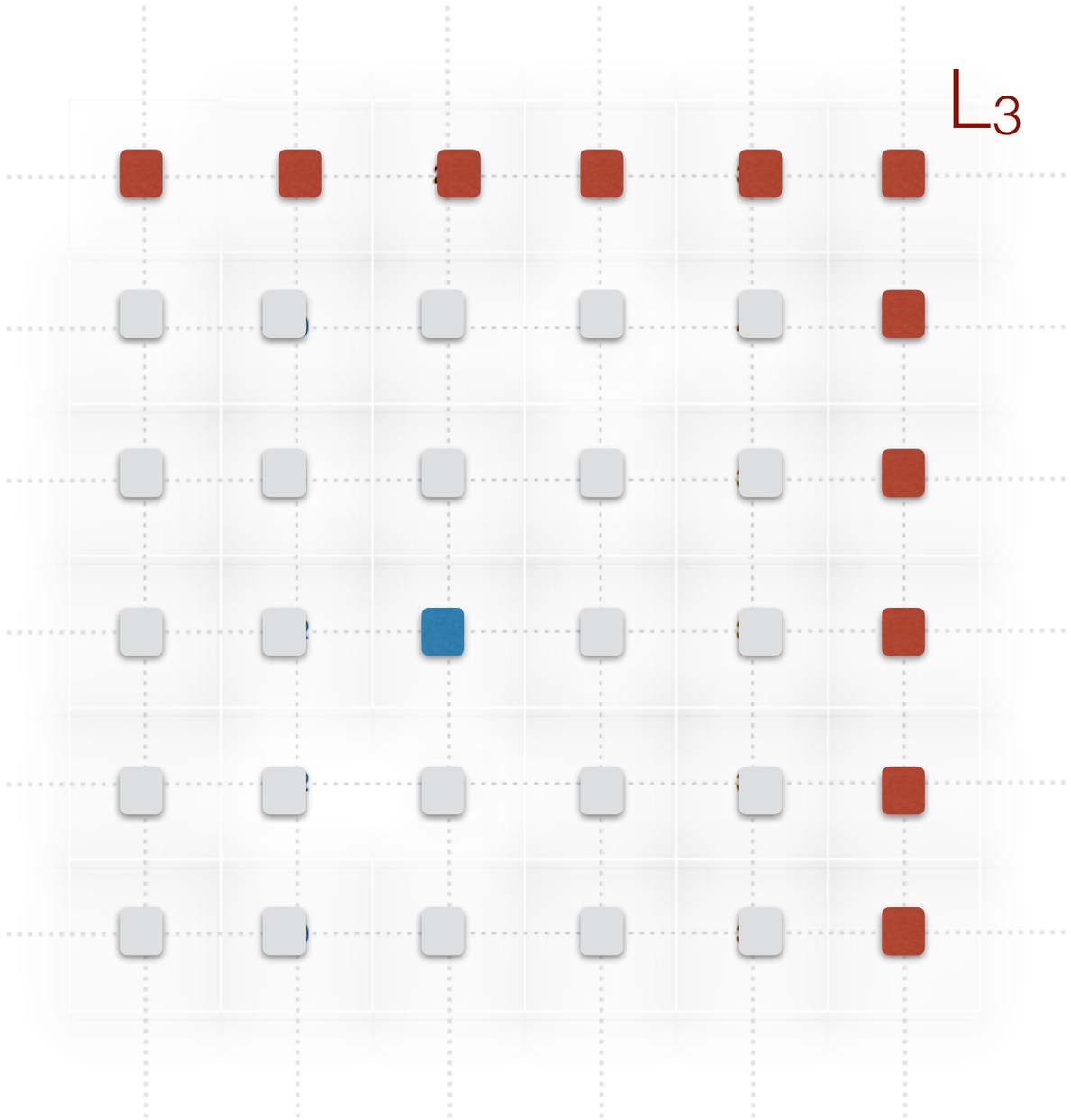
Concentric sweep



Concentric sweep



Concentric sweep



Horizons

- Merriam Webster:
  - the line where the terrain and the sky seem to meet

Horizons

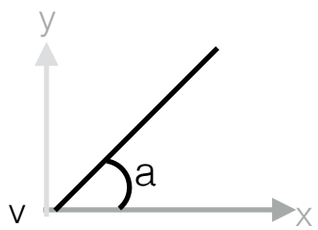
- Merriam Webster:
  - the line where the terrain and the sky seem to meet

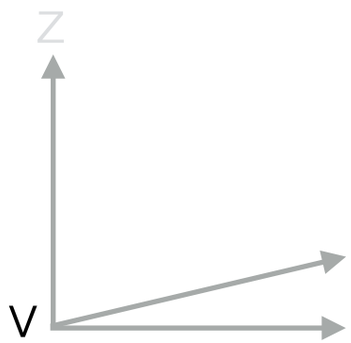
Horizon

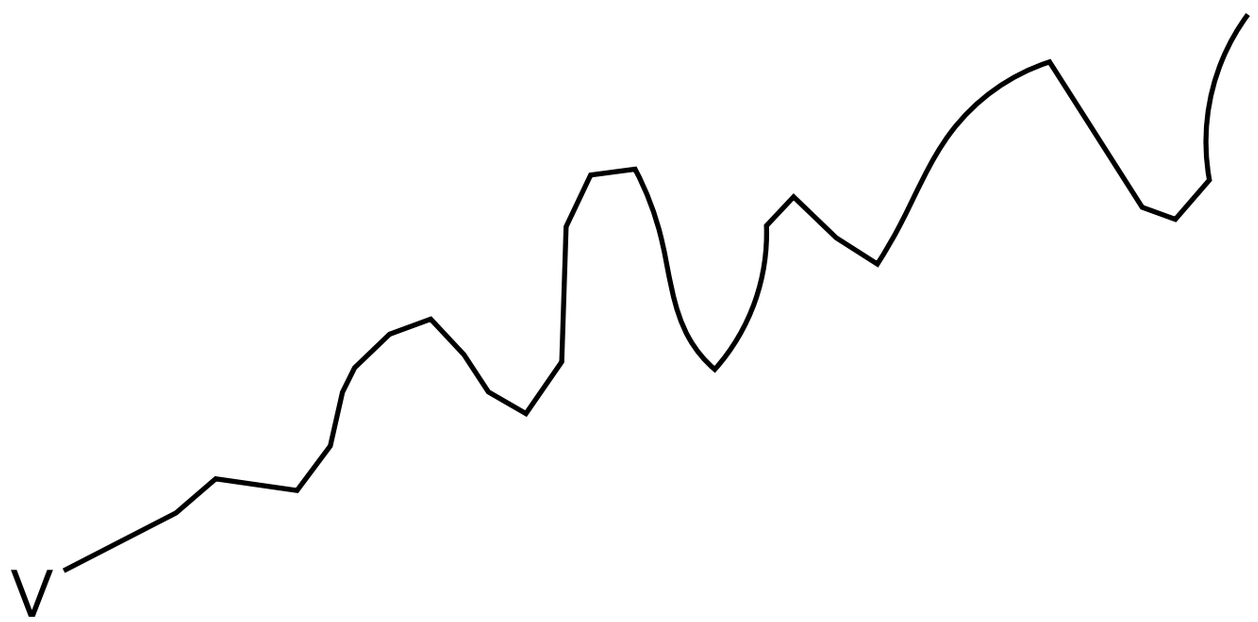
H_v : [0, 2PI) ——> R

$H_v(a)$: horizon (with respect to v) in direction a

- cut the terrain with a vertical plane through ray from v of azimuth a
- $H_v(a)$ is the maximum vertical angle (zenith) of all points intersected by this plane (all the points on T whose projection on the xy-plane has azimuth a)





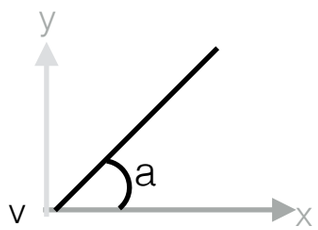


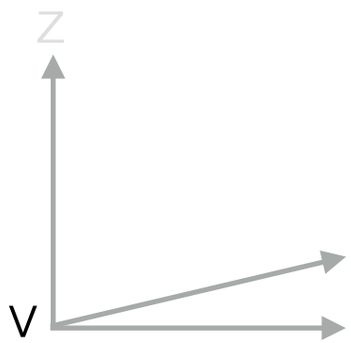
Horizon

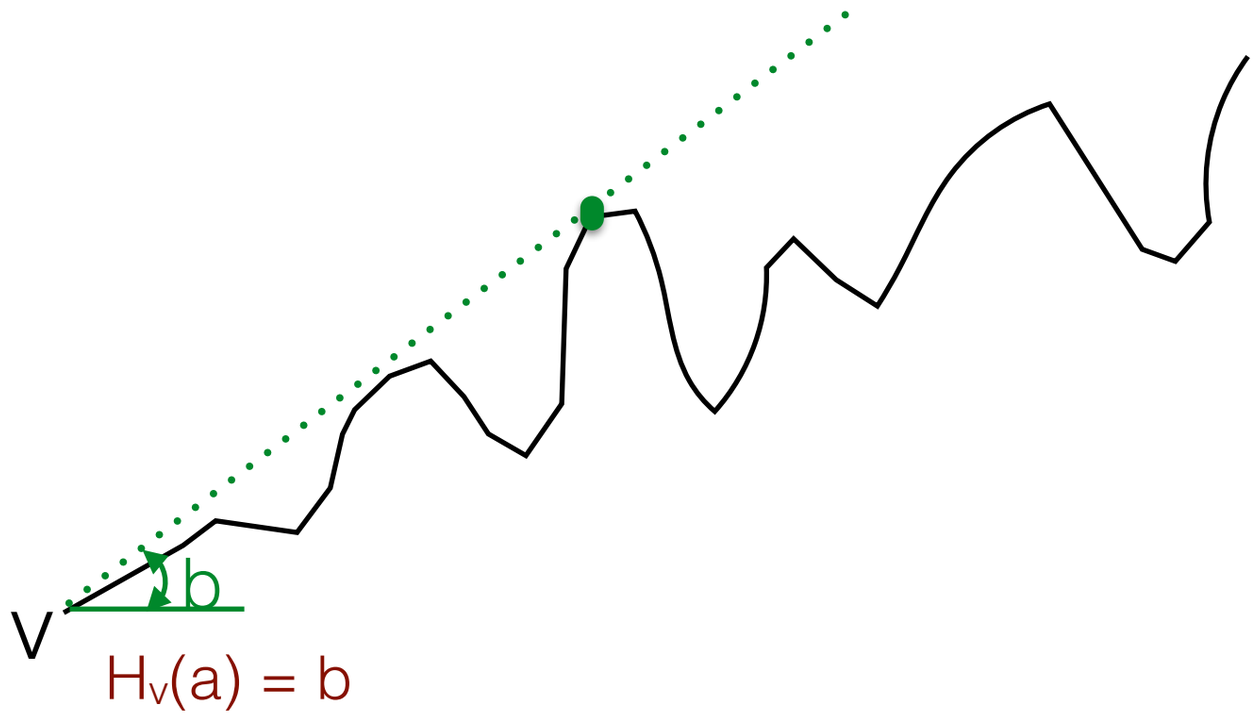
H_v : [0, 2PI) ——> R

$H_v(a)$: horizon (with respect to v) in direction a

- cut the terrain with a vertical plane through ray from v of azimuth a
- $H_v(a)$ is the maximum vertical angle (zenith) of all points intersected by this plane (all the points on T whose projection on the xy-plane has azimuth a)





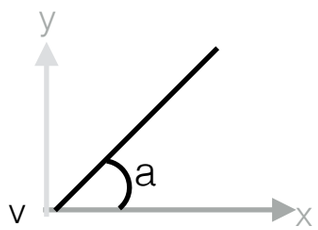


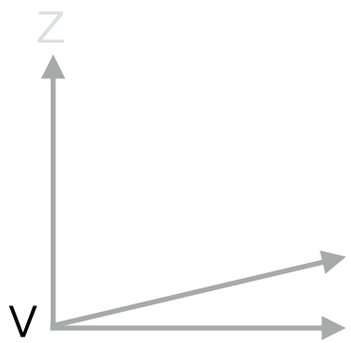
Horizon

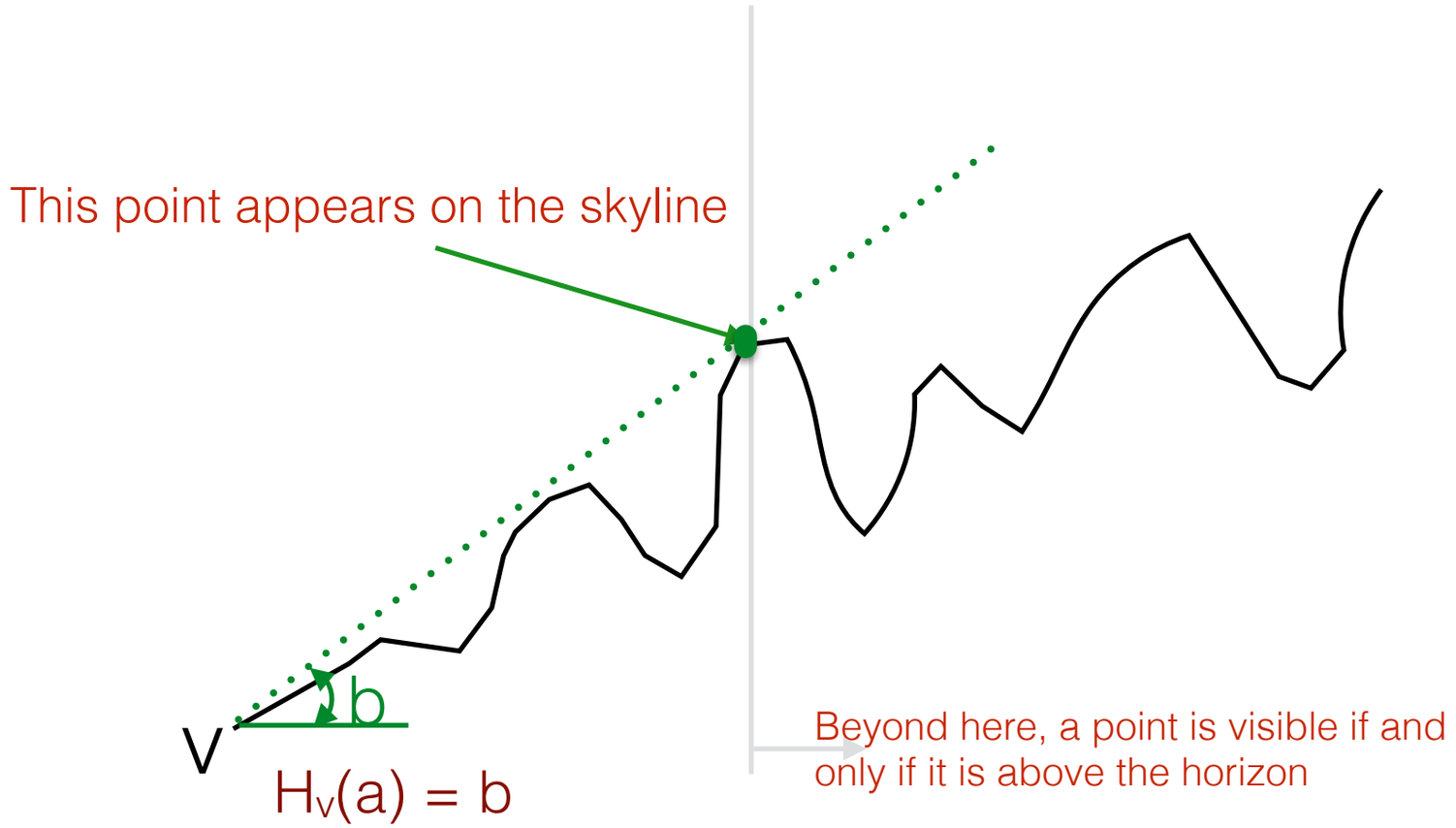
H_v : [0, 2PI) ——> R

$H_v(a)$: horizon (with respect to v) in direction a

- cut the terrain with a vertical plane through a ray from v of azimuth a
- $H_v(a)$ is the maximum vertical angle (zenith) of all points intersected by this plane (all the points on T whose projection on the xy-plane has azimuth a)



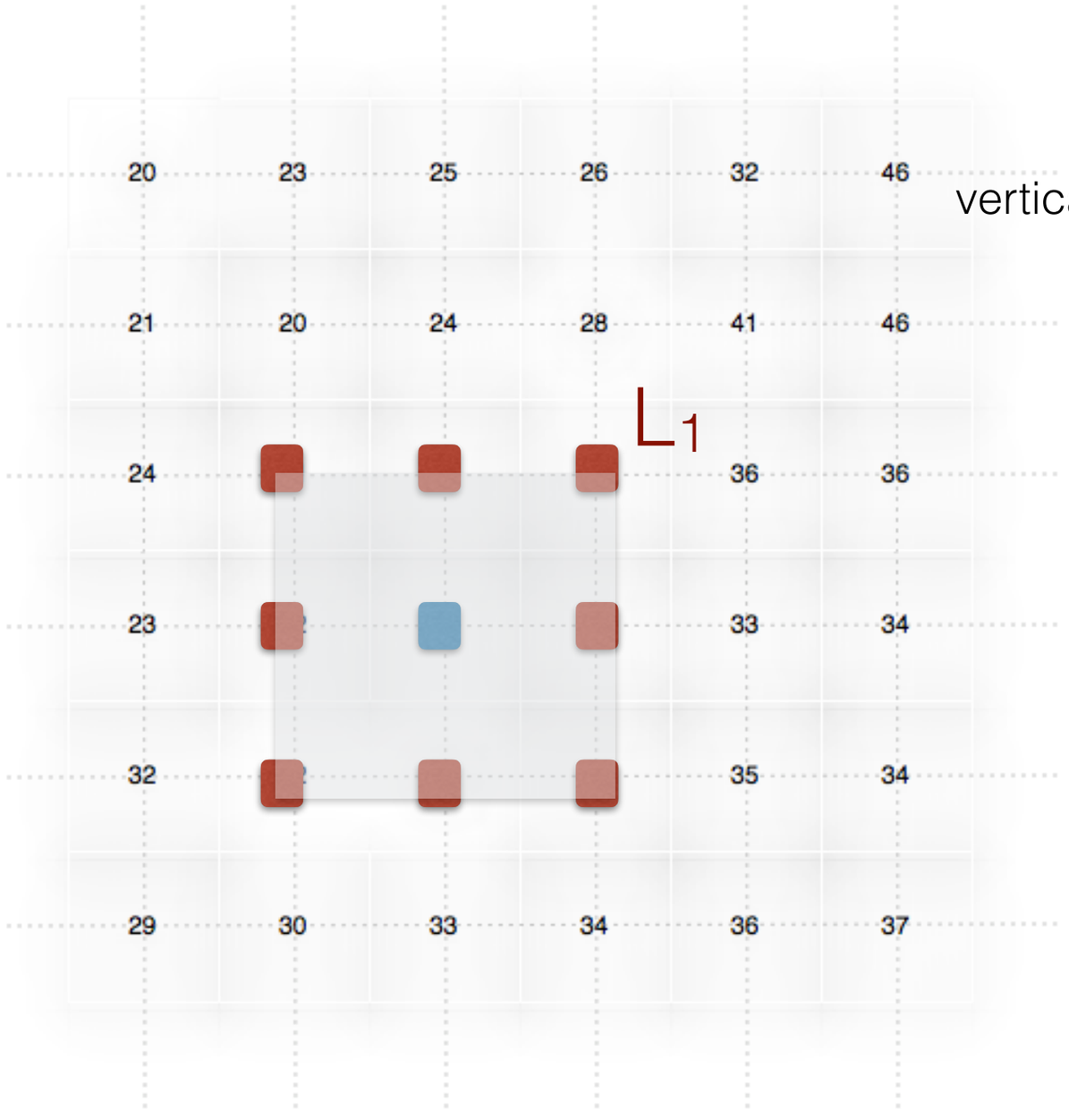


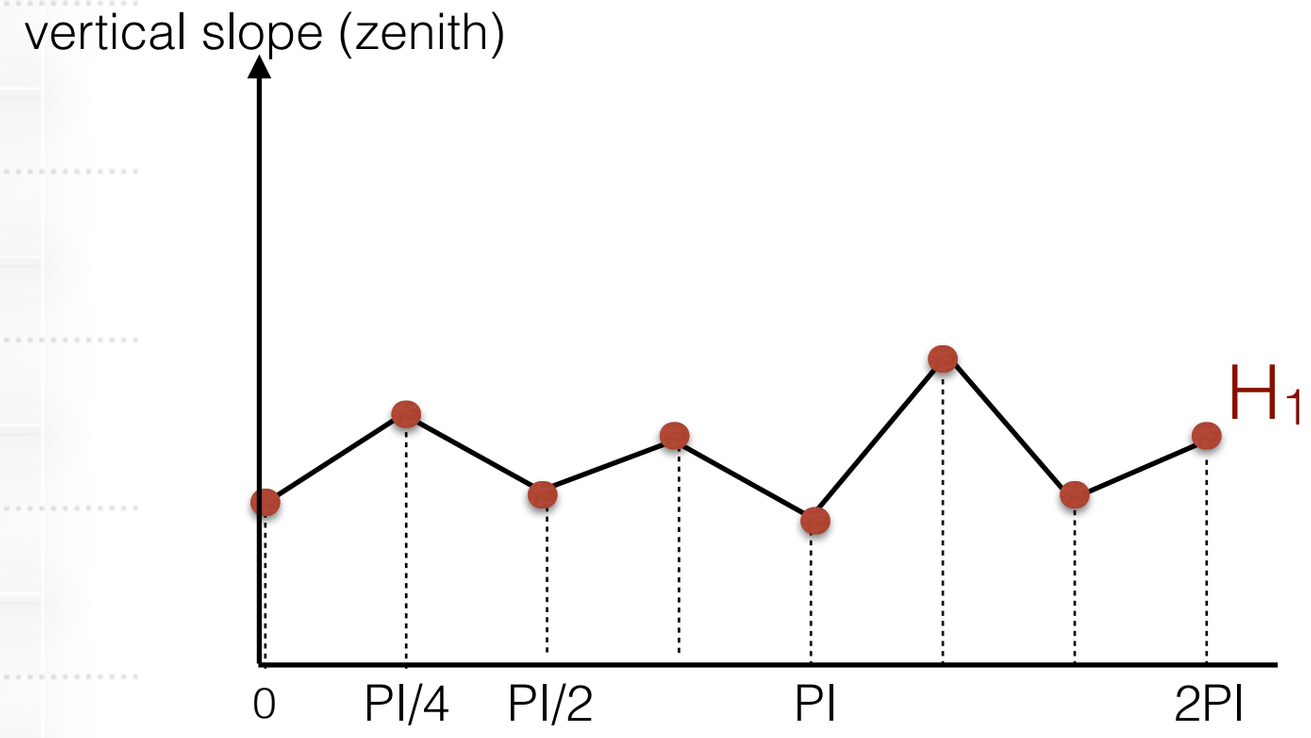


A point is visible if it is above the horizon.

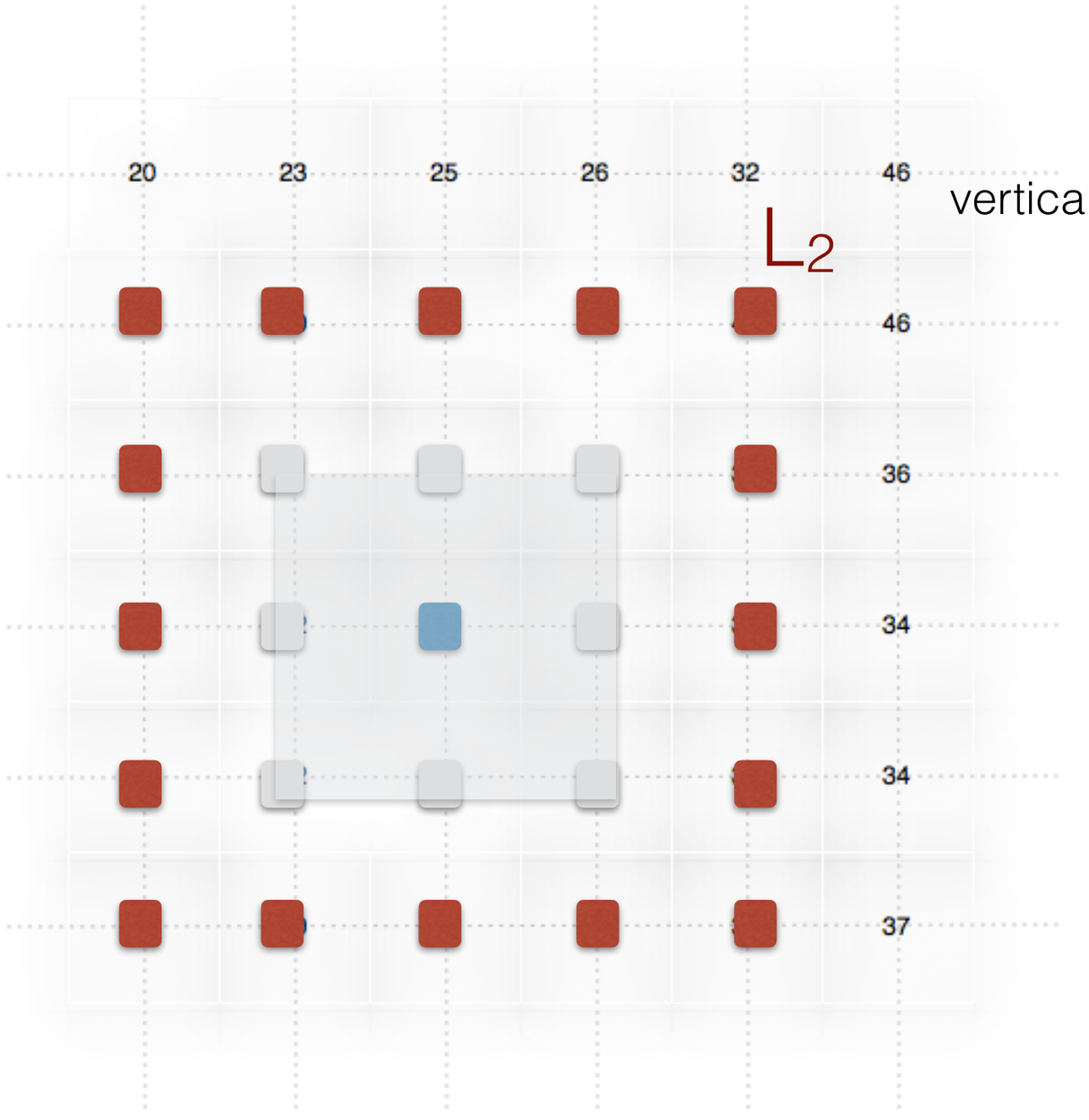
Idea: compute horizons.

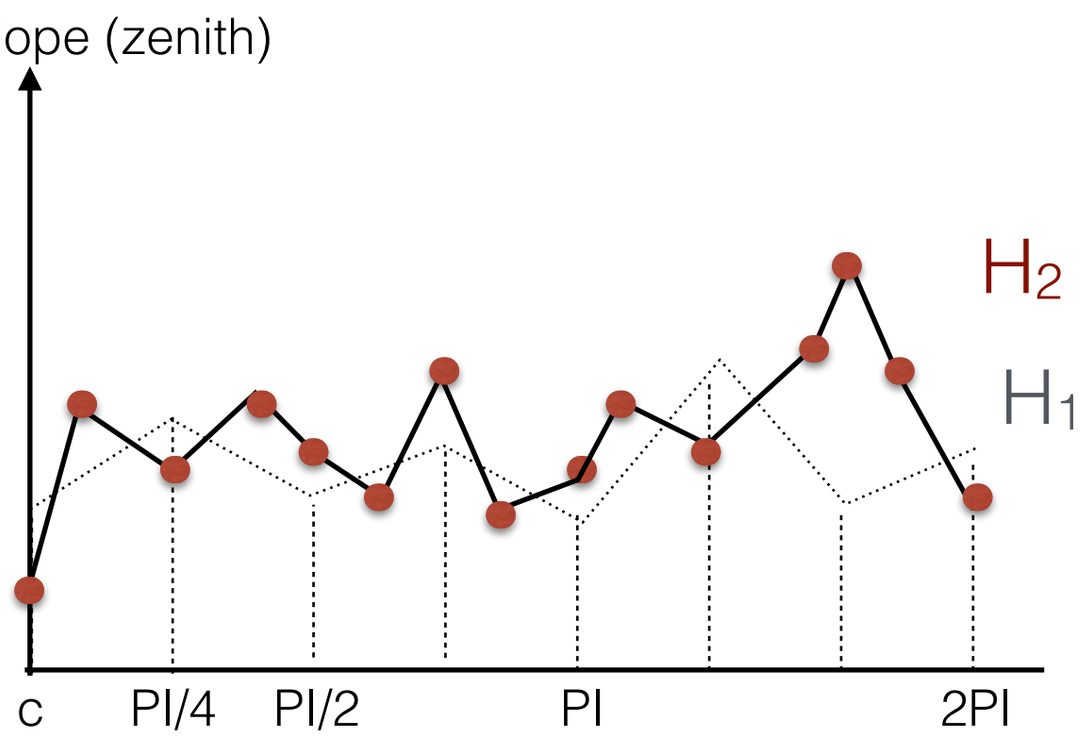
Viewshed and horizons



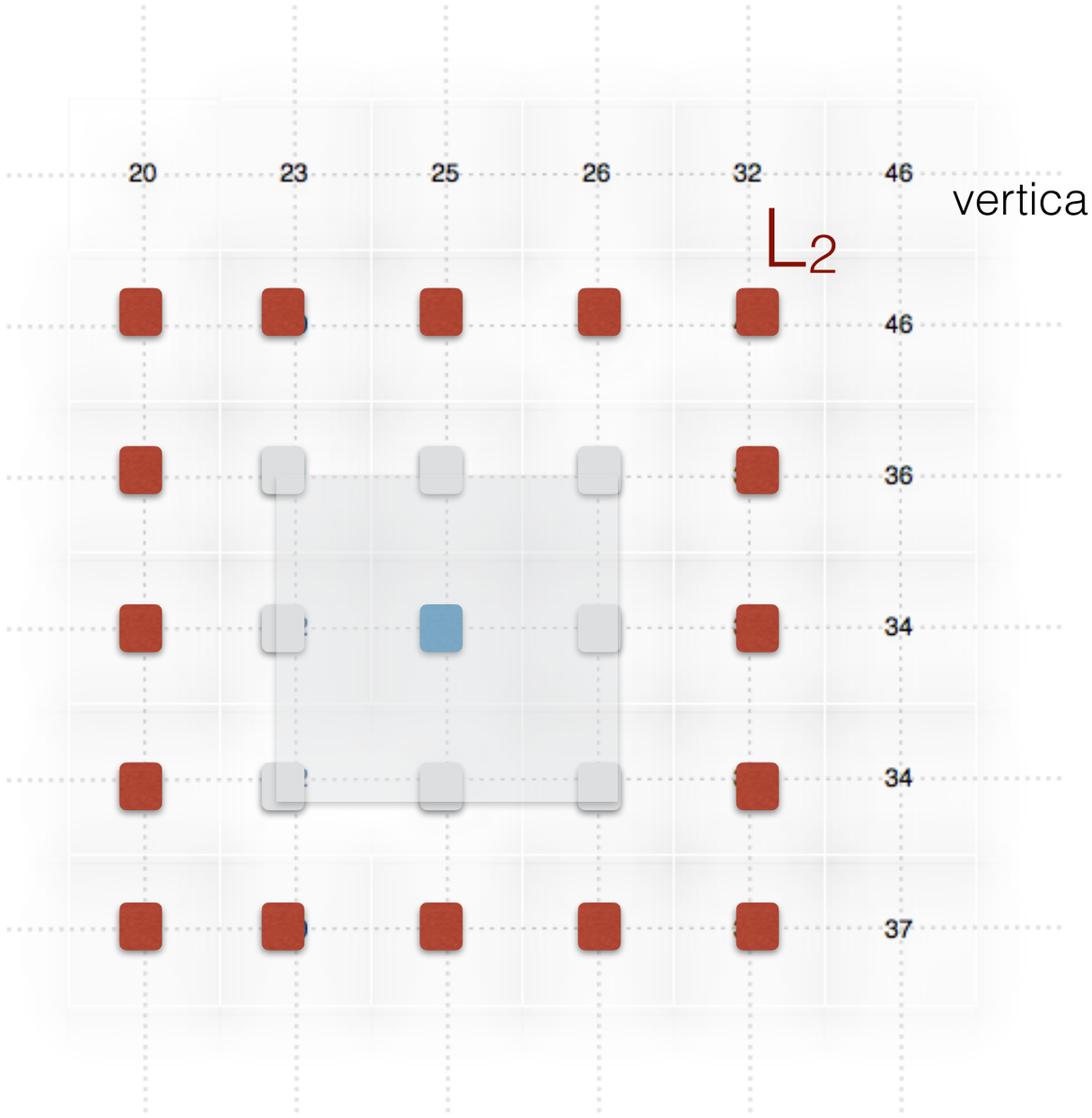


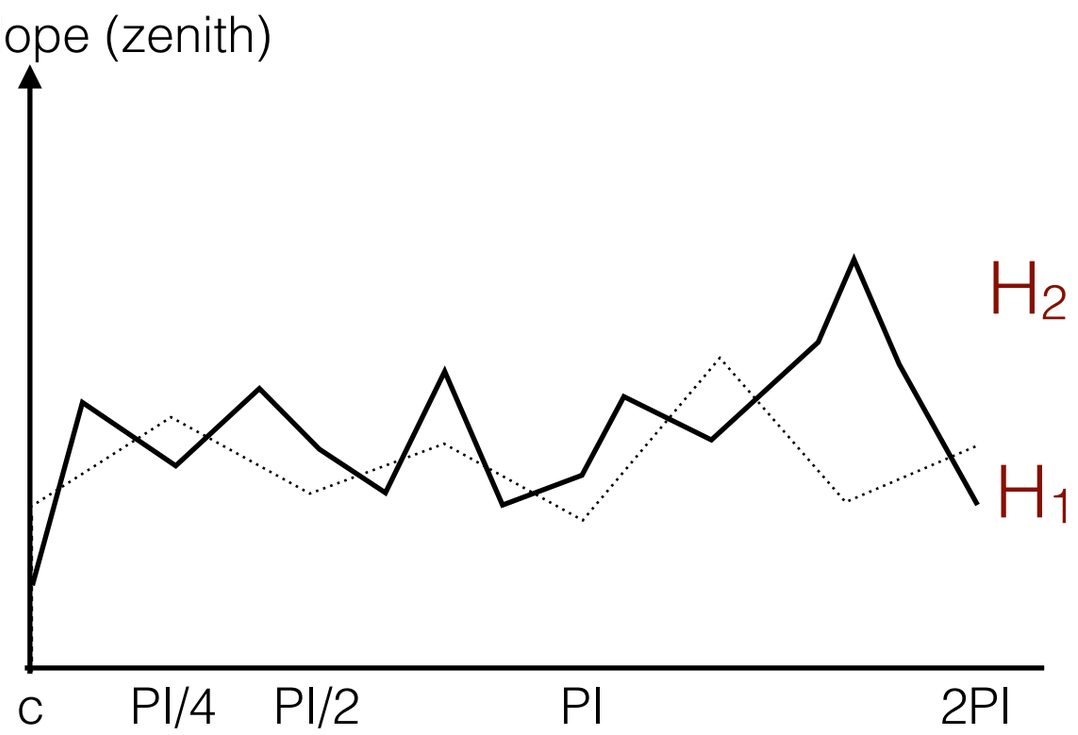
Viewshed and horizons



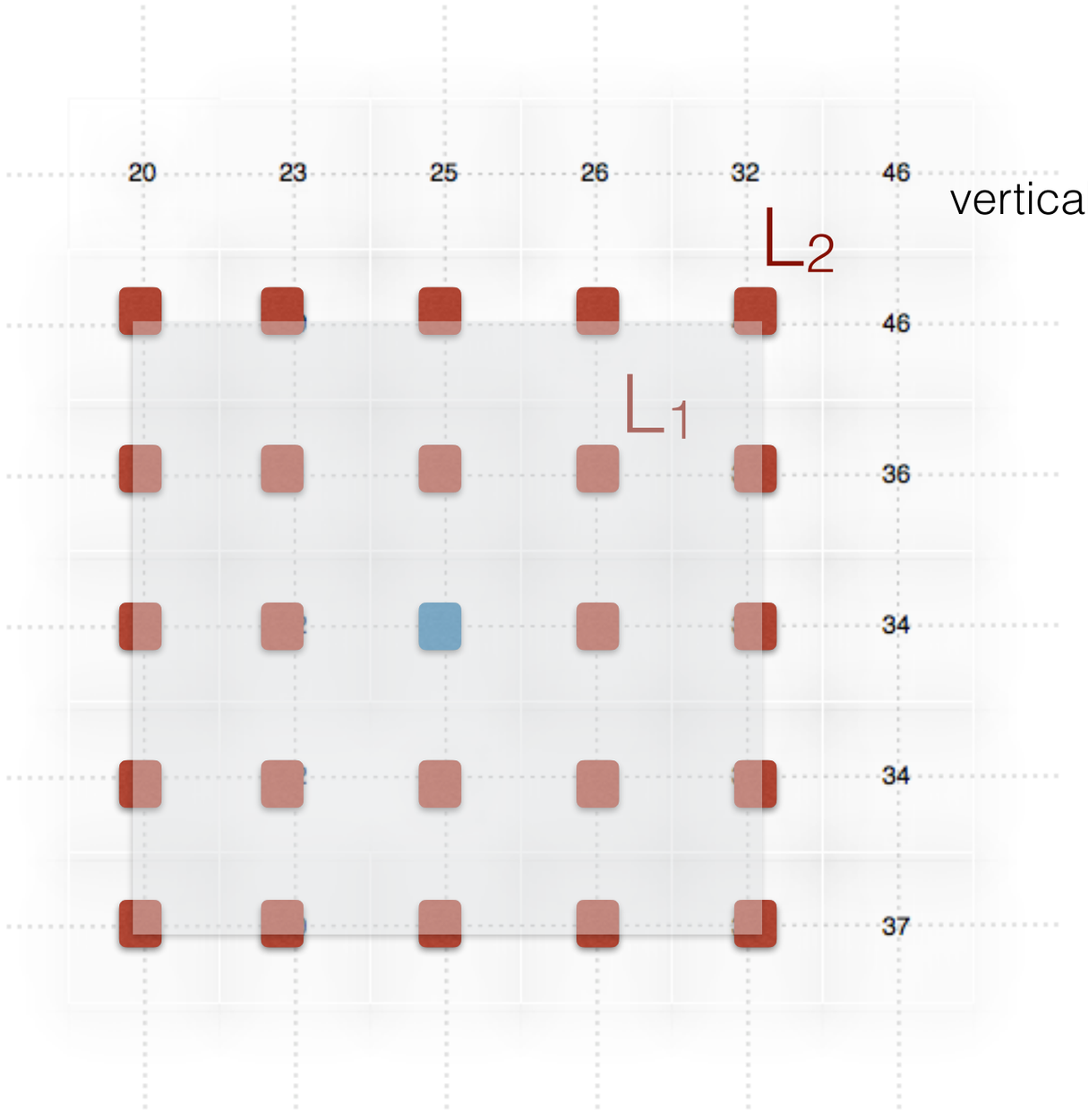


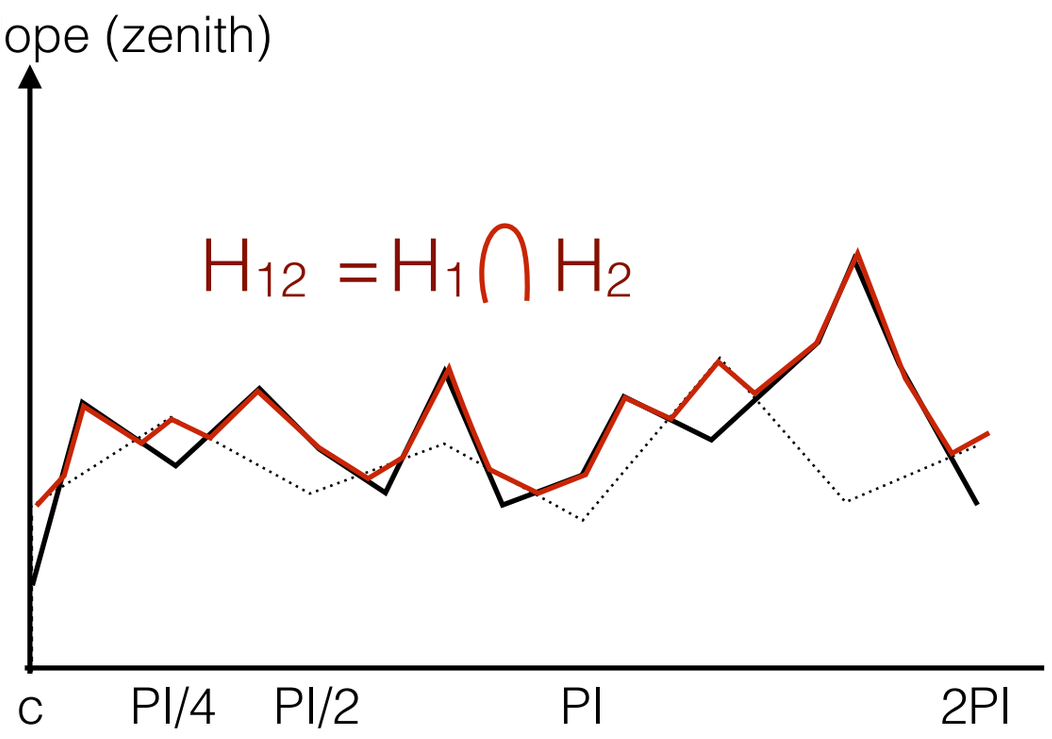
Viewshed and horizons



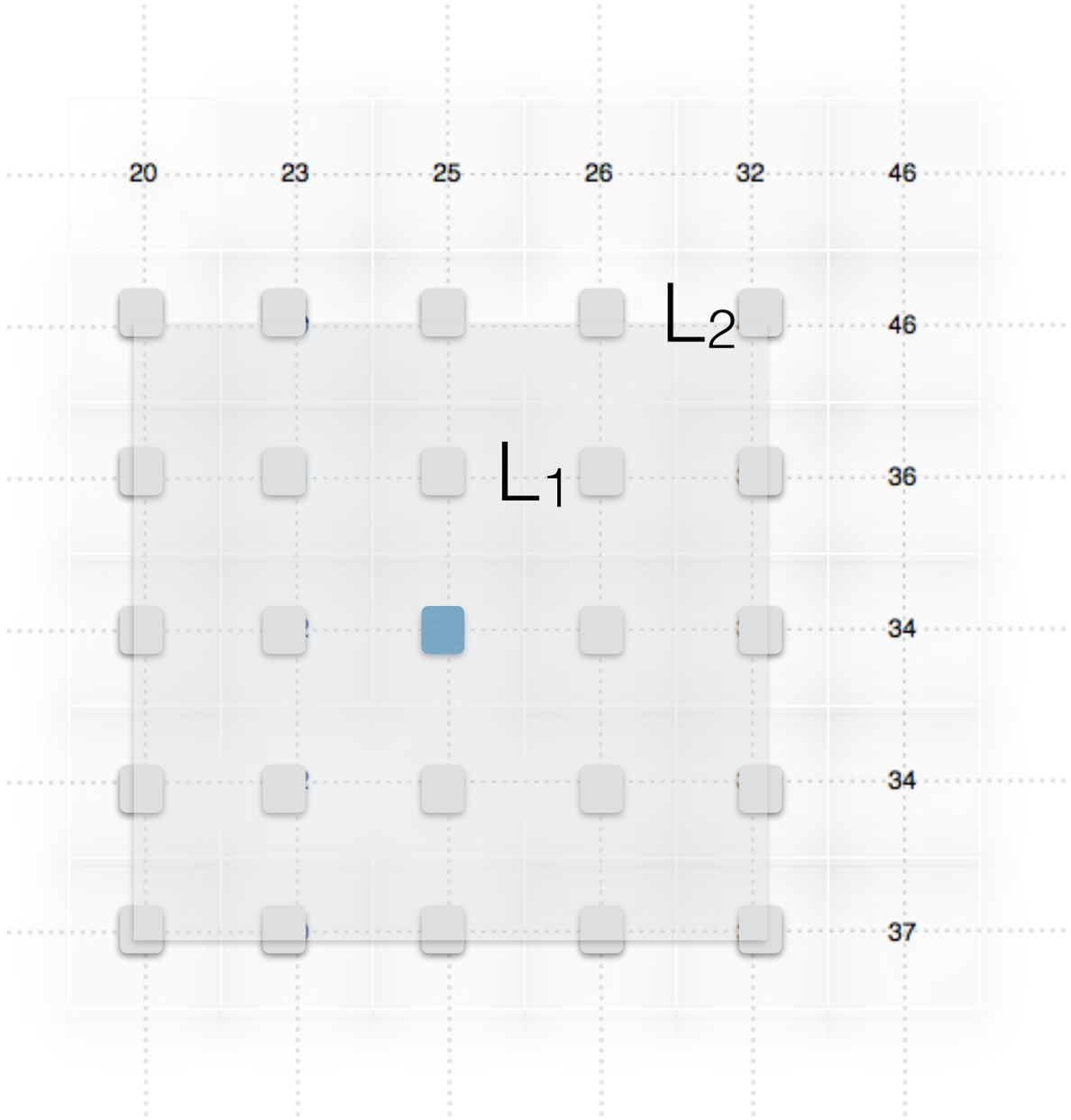


Viewshed and horizons

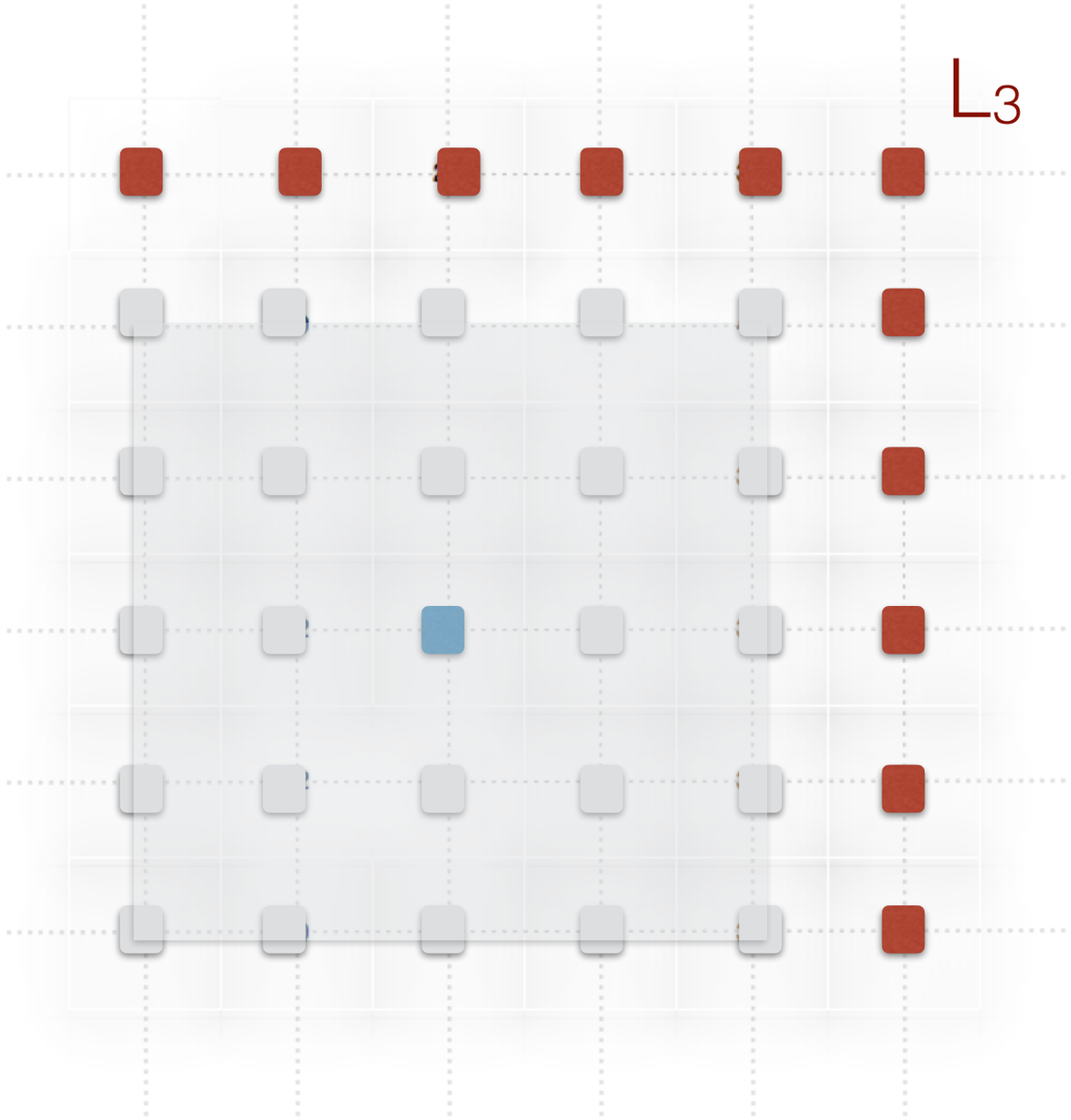




Viewshed and horizons

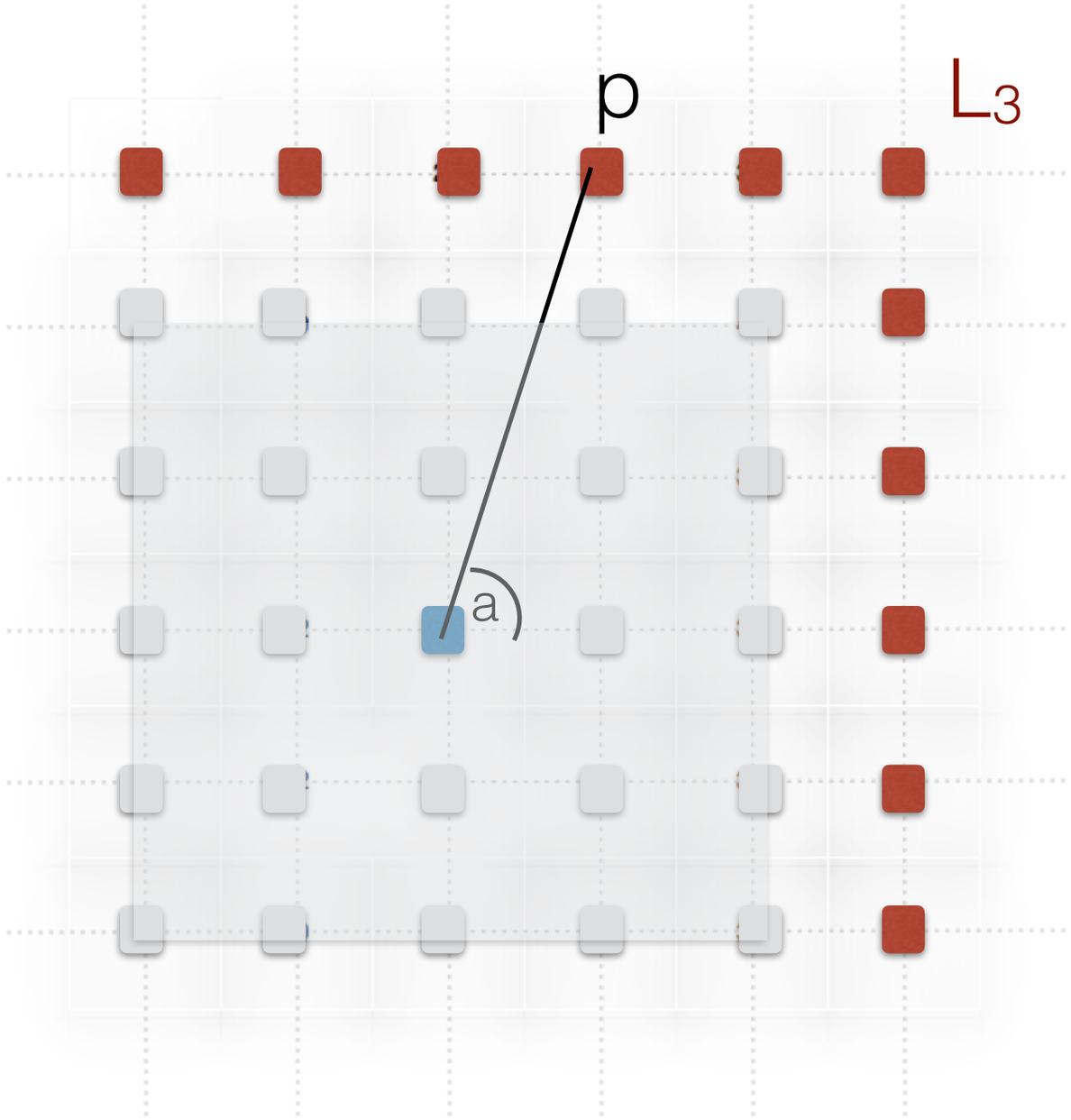


Viewshed and horizons



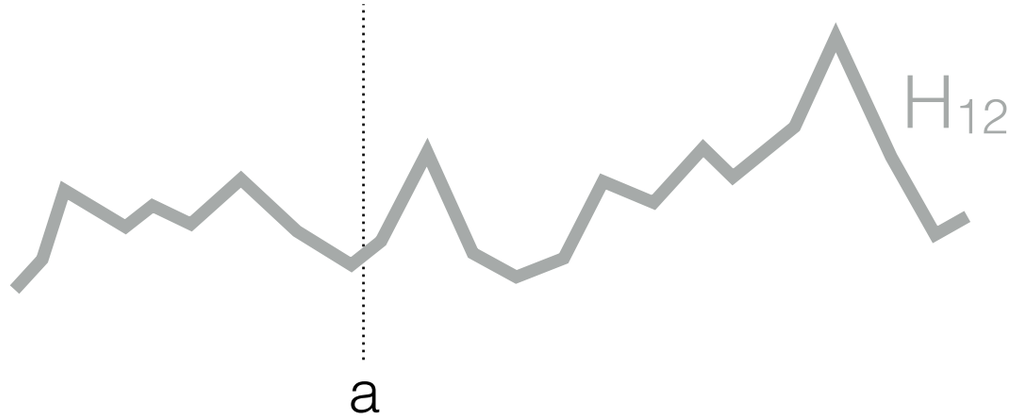
We walk along L3, computing the horizon of L3 and determining if points on L3 are visible or not

Viewshed and horizons



Is point p in L3 visible ?

p is visible if
slope(vp) < $H_{12}(a)$



Viewshed and horizons

- Elegant techniques that can be extended
  - Linear interpolation or nearest neighbor,…
  - Starting point for triangulated terrains

- Worst-case bounds not great
  - fast in practice because horizons stay very small

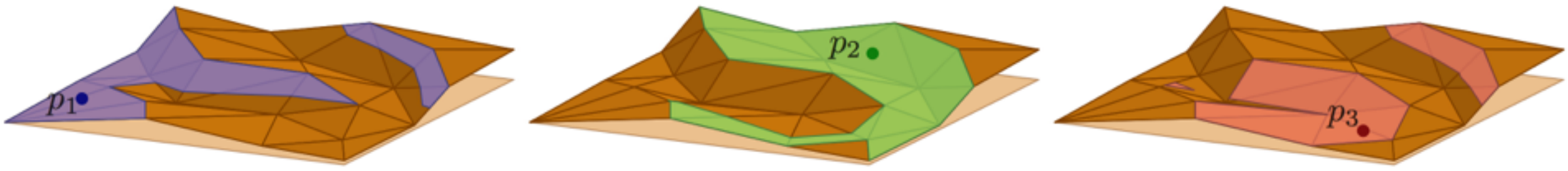
Viewsheds on triangulated terrains

Viewsheds on triangulated terrains

- Several algorithms are known
- Based on horizons
  - Idea: traverse triangles in order of increasing distance form viewpoint, and update horizon.
  - Bootstrap with divide and conquer

Viewsheds on triangulated terrains

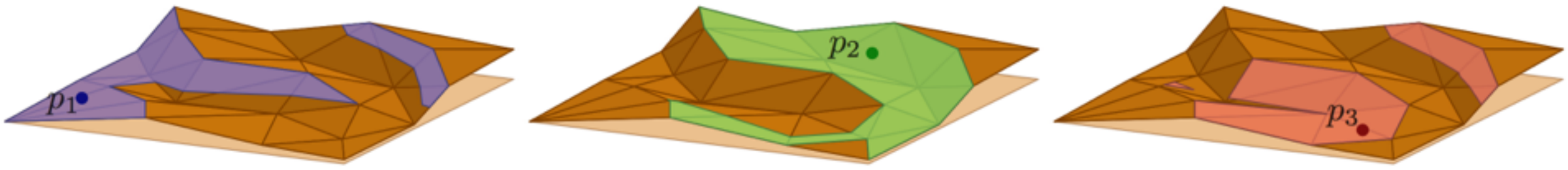
- viewshed(p) contains all points of the terrain that are visible from p



from: http://arxiv.org/pdf/1309.4323.pdf

Viewsheds on triangulated terrains

- viewshed(p) contains all points of the terrain that are visible from p



from: http://arxiv.org/pdf/1309.4323.pdf

- viewshed(p) may intersect a triangle multiple times.
- How big can it be?
  - Space complexity = number of edges on boundary of viewshed(p)
  - It is known that the complexity of a viewshed on a triangulated terrain can be $O(n^2)$. On a triangulated grid, the complexity of a viewshed is $O(n\sqrt{n})$
  - Contrived. In practice view shed are small. Proving realistic upper bounds still open problem.

from: HH, MdB, KT 2009

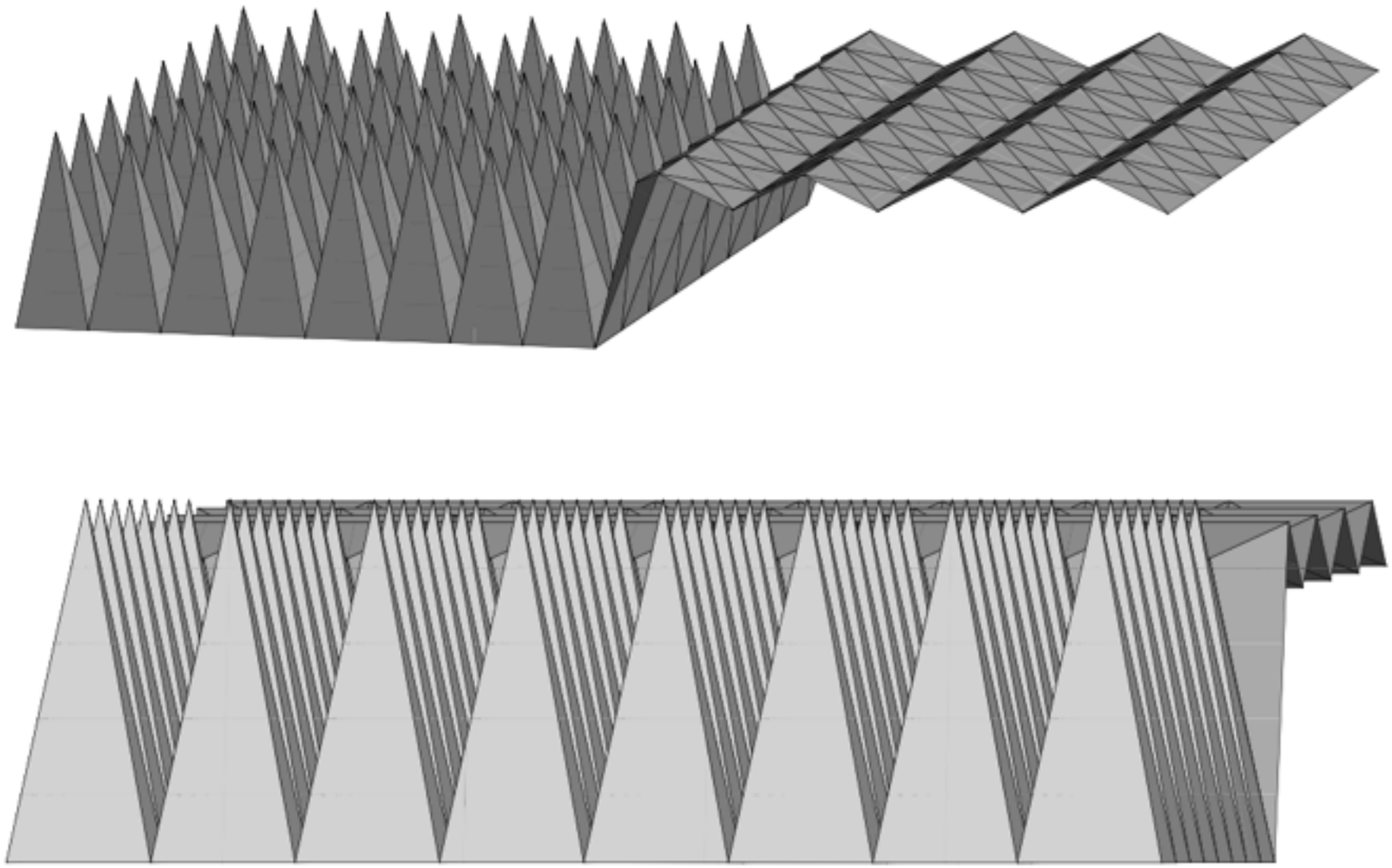


Figure 1: Two views of the same terrain defined by a regular grid. The second view gives a visibility map of complexity $\Theta(n\sqrt{n})$. Note that the terrain can be flattened further without changing the view combinatorially.

- Cumulative viewshed
- Total viewshed
- Find point of maximum/minimum visibility
- Find optimal paths

Extending viewsheds

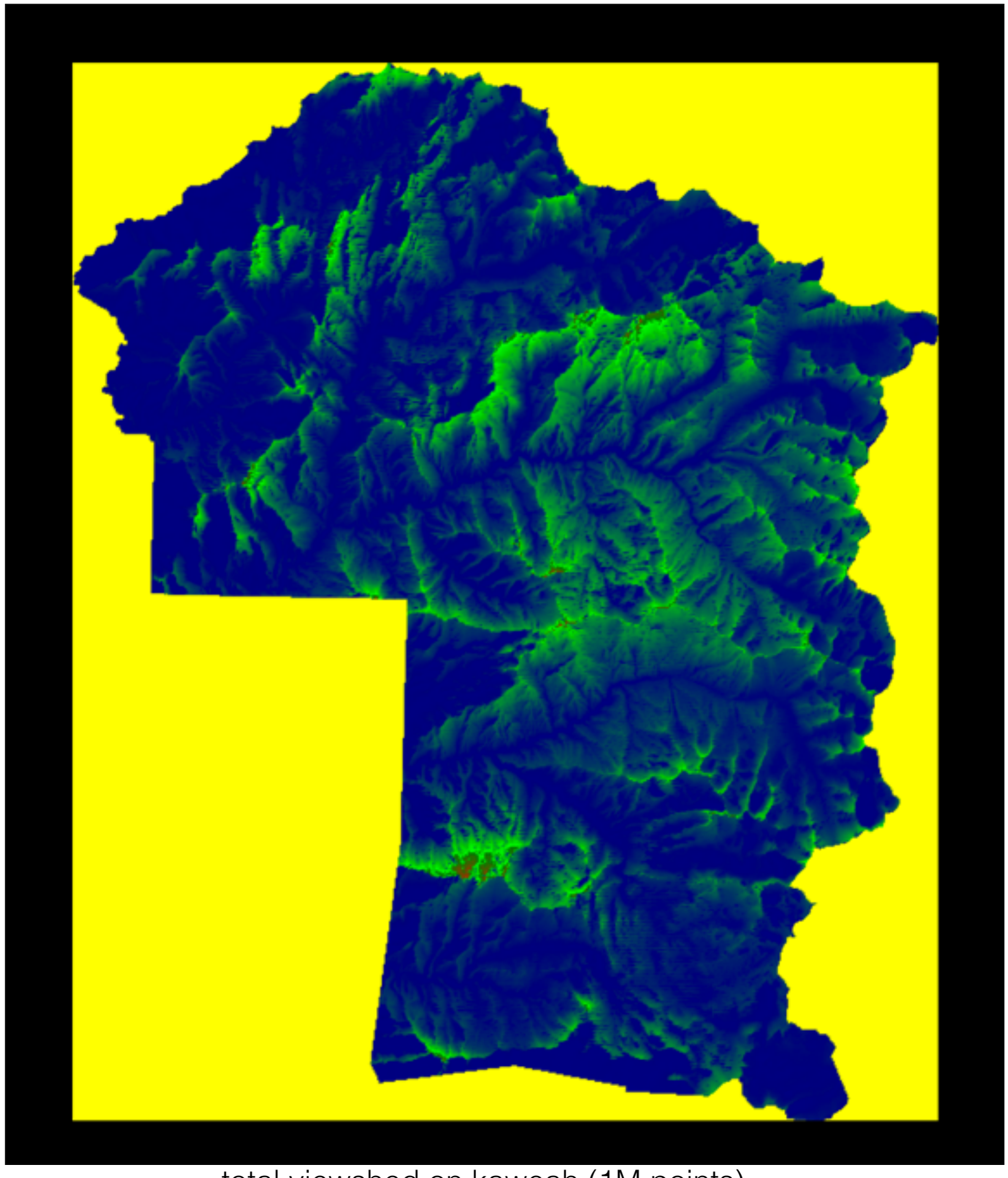
Extending viewsheds

- Cumulative viewshed
  - Given a set of viewpoints, compute their joint visibility

Total viewshed

- Input: elevation grid G
- Output: TV grid
    - TV(i,j) = nb. visible points in viewshed(i,j)

- Algorithm?
- Running time?



total viewshed on kaweah (1M points)
time: 42.6 hrs

Summary

- Viewshed
  - Straightforward solution
    - Reasonably fast even for very large terrains (as long as they fit in memory)
  - Refined solutions expose elegant ideas
    - Radial sweep + augmented RB-trees
    - Horizons
    - Carry on to triangulated terrains
  - Accuracy
    - Interpolation is important
- Total viewshed
    - Straightforward solution: $O(n^2\sqrt{n})$
    - Refined: $O(n^2 \lg n)$
    - Too slow…

Need approximate viewsheds!
Need parallel algorithms!