CARTOGRAPHIC ANALYSIS

MODES OR VARIATION OF SPATIAL DATA

- Three modes:
  1. spatial—variation from place to place
  2. temporal—variation from time to time
  3. thematic—variation from characteristic to characteristic
- Observing real world phenomena usually involves:
  1. holding one mode “fixed”
  2. varying a second mode in a “controlled” manner
  3. measuring the third

  Ex: measure population density in 1980 over the US by using Census data
    - time is fixed—i.e., 1980
    - vary location in a controlled manner by scanning over the entire US
    - theme is population density
- Fixing time and varying spatial yields *cross-sectional data*
- Fixing spatial and varying time yields *longitudinal data*

NATURE OF GEOGRAPHIC DATA

1. Geographic position
   - location on the earth’s surface—i.e., where is it?
   - actually need elevation but location on a plane or a sphere are usually enough
   - should use a common coordinate system—e.g., UTM
2. Attribute
   - non-spatial—i.e., what is it?
   - often qualitative and not very accurate—e.g., a pine forest does not consist of just pine trees
3. Spatial relationship
   - need to know more than just location
     a. relationship to other features
     b. e.g., how close is a paper mill to water, roads, forests
   - can’t store all possible proximity information—some is derived
4. Time
   - when did the condition or feature exist?
   - historical information—e.g., prior uses

CHARACTERISTICS OF GEOGRAPHIC DATA

1. High volume
2. Dimensionality—points, lines, areas
3. Level of measurement
   - nominal
   - ordinal
   - interval
   - ratio
4. Continuity
   - data such as contours for elevation data
   - discontinuous data as in choropleth maps (i.e., areas of equal value separated by boundaries)

     Ex: tax rate is different across a state border

PROPERTIES OF GEOGRAPHIC DATA

1. Size and its characterization in measurement
   - point—measure location, adjacency, and elevation
   - line—measure length, direction, connectivity, and "wigglyness"
   - polygon—measure topology (e.g., holes, outliers), area, shape, boundary length, location, orientation
   - volume—measure topology, continuity, surface slope, surface aspect, surface trend, surface structure, location, elevation
   - some can be measured easily if the data is geocoded
   - others are very difficult to measure in the real world and thus can only be analyzed using abstractions of mapping
2. Distribution
   - density is a measure of the distribution of a feature across space
   - usually computed by counting cartographic objects or attributes

3. Pattern
   - a characteristic of distributions and is a description of their structure
   - repetition of an attribute over space
   - lack of randomness
4. Neighborhood property
   - mathematical formulation of relationship between a geographic property and distance
   - small separation means similarity
   - large separation means dissimilarity
   - distance function
     a. autocorrelation function
     b. spatial interaction models
     c. distance decay models (e.g., gravitation)
     d. variogram
5. Contiguity
   - property of being related by juxtaposition
   - sharing a common boundary
     a. four-adjacency (just edges)
     b. eight-adjacency (includes corners)
   - connectivity in a network

6. Shape
   - hard to measure and quantify
   - Ex: shape number yielding a comparison with a circle

$$1 - (A \cap B) / (A \cup B)$$

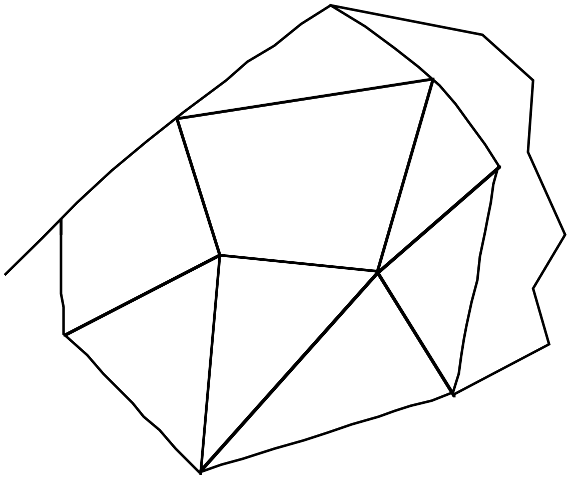
     a. make sure center of circle and center of mass of object coincide
     b. use circle as it has the minimum perimeter for a given area

7. Scale
   - ratio of distances on a map to the same distance on the part of the world shown on the map
   - scale dependence
     a. objects become clearer at certain scales
     b. objects disappear completely at certain scales
   - scale independence
     a. never change
     b. can use fractal geometry
        - lines have dimension between 1.0 and 2.0
        - areas and surfaces have dimension between 2.0 and 3.0
        - volumes have dimension ?

GEOCODING

1. Process of converting spatial information into computer-readable form

2. Possible errors:

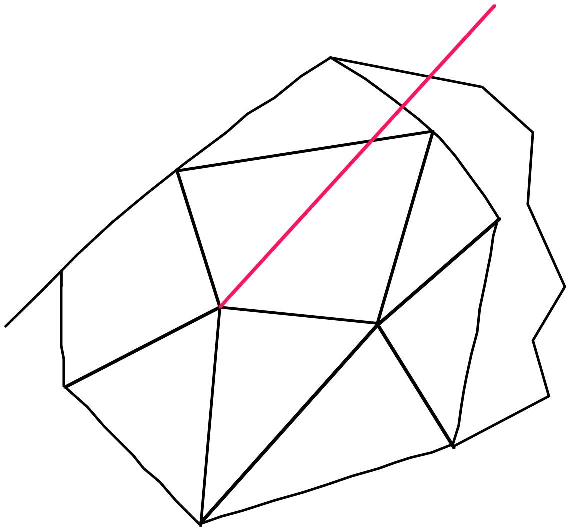


GEOCODING

1. Process of converting spatial information into computer-readable form

2. Possible errors:

   - spike error



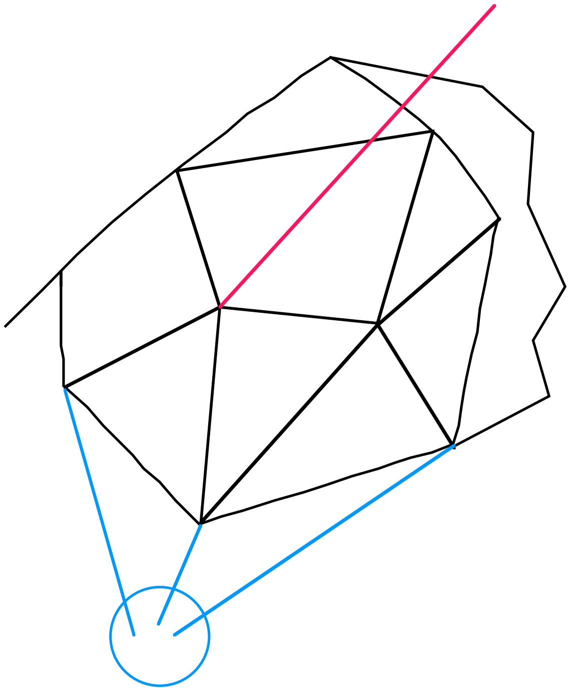
GEOCODING

1. Process of converting spatial information into computer-readable form

2. Possible errors:

    - spike error

    - unsnapped node



GEOCODING

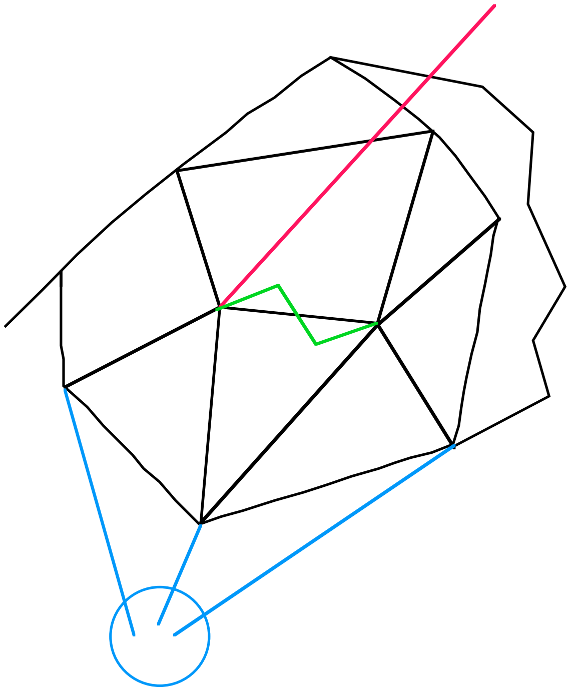
1. Process of converting spatial information into computer-readable form

2. Possible errors:

   - spike error

   - unsnapped node

   - sliver

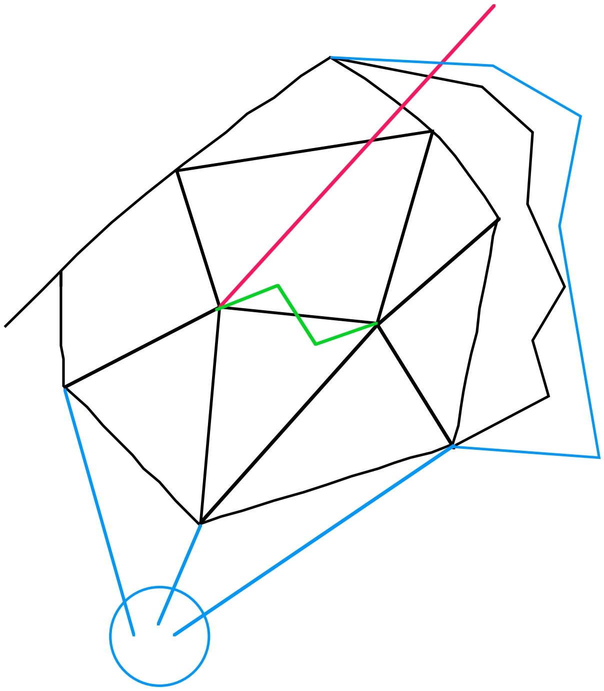


GEOCODING

1. Process of converting spatial information into computer-readable form

2. Possible errors:

   - spike error
   - unsnapped node
   - sliver
   - duplicated line

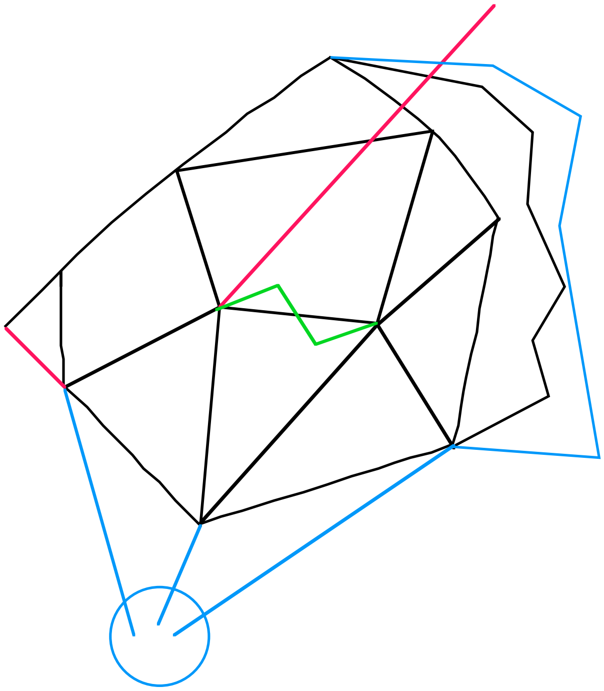


GEOCODING

1. Process of converting spatial information into computer-readable form

2. Possible errors:

- spike error
- unsnapped node
- sliver
- duplicated line
- missing link

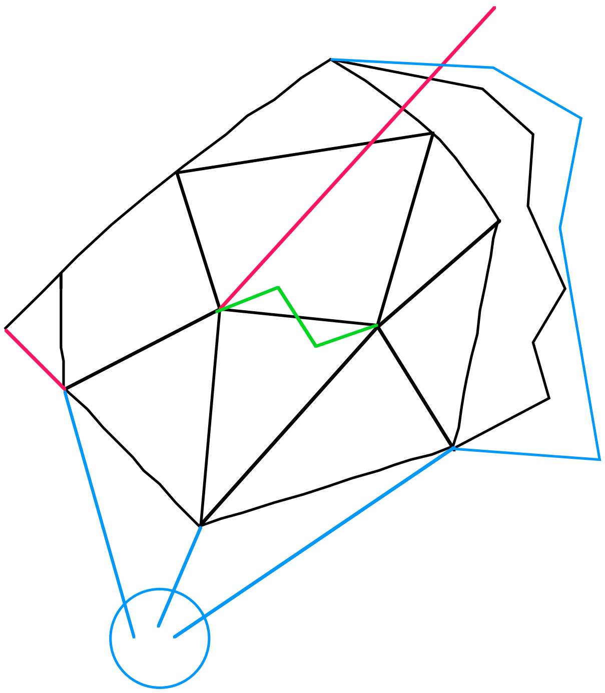


GEOCODING

1. Process of converting spatial information into computer-readable form

2. Possible errors:

   - spike error
   - unsnapped node
   - sliver
   - duplicated line
   - missing link



3. Error correction processes:

   - conflation—reconciles differences between position of the same feature on different map layers so slivers will be avoided
   - edge matching—adjusts positions of features that extend across map sheet boundaries
   - line snapping—connects lines to a node if they end within a specified distance
   - line coordinate thinning—reduces volume of data by purging some points
   - visual inspection

COORDINATES

1. meridian—line connecting the north and south poles through a point
2. latitude—angle formed by lines from center of the earth to a point and the point at which its meridian crosses the equator
   - –90 south to 90 north
3. longitude—angle on the equatorial plane between the point’s meridian and the Greenwich meridian
   - –180 west to 180 east
4. meridian—a line of constant longitude
5. parallel—a line of constant latitude
6. great circle—imaginary circle on earth’s surface made by a plane that passes through the center of the earth
   - one nautical mile ≈ (miles/degree) on a great circle divided by 60

TOPOLOGICAL DATA MODEL

1. Encodes spatial relationships
   - usually based on arcs (i.e., edges)
   - contrast with “spaghetti” model which represents all data (e.g., arcs, polygons, etc.) as sequences of points
2. Application to spatial data
   - polygon table—lists constituent arcs for each polygon
   - node table—lists incident arcs for each node
   - arc table
     a. start and end nodes
     b. adjacent polygons
   - arc coordinate table
     a. coordinate values for start and end points of each arc
     b. coordinate values of intermediate points
        - assumes straight lines between points
        - known as “shape points” in TIGER
3. Useful for performing spatial analysis
   - avoids rederiving spatial relationships from coordinate values
     a. connectivity
     b. contiguity
   - requires updating

APPLICATIONS OF TOPOLOGICAL CODING

1. GBF/DIME files
   - objects through which the data can be accessed
     a. edges
     b. faces and vertices must be searched for sequentially in the edge table
   - no easy way to access the topology given the location of a point in space

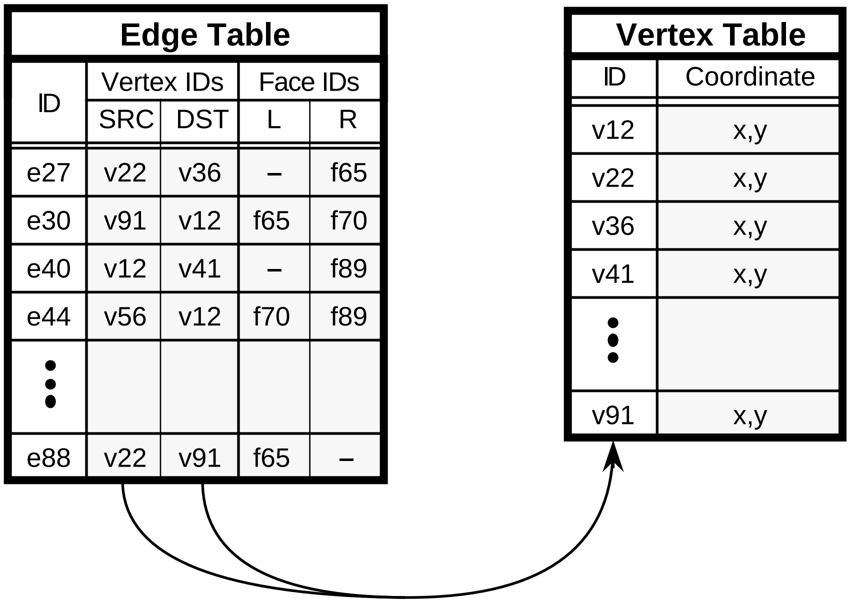
2. TIGER files
   - objects through which the data can be accessed
     a. 0-cells: points (nodes)
     b. 1-cells: lines (segments)
     c. 2-cells: areas (blocks, census tracts, enumeration districts)
   - uses Peano keys to access the topology given the location of a point in space

3. DLG

4. Unified PMR quadtree

GBF/DIME TOPOLOGICAL CODING

- Basic entity is a street segment which includes:
  1. the endpoints
  2. the block faces on its two sides
  3. the range of addresses on its two sides

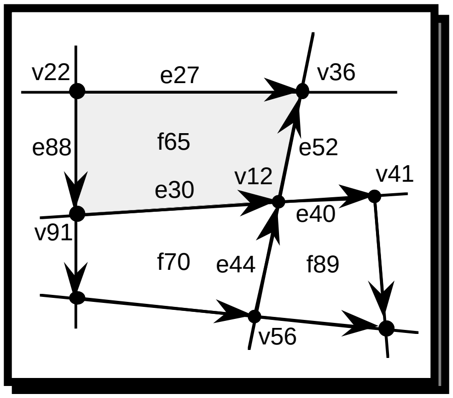


Edge Table

ID	Vertex IDs		Face IDs	
	SRC	DST	L	R
e27	v22	v36	–	f65
e30	v91	v12	f65	f70
e40	v12	v41	–	f89
e44	v56	v12	f70	f89
⋮				
e88	v22	v91	f65	–

Vertex Table

ID	Coordinate
v12	x,y
v22	x,y
v36	x,y
v41	x,y
⋮	
v91	x,y

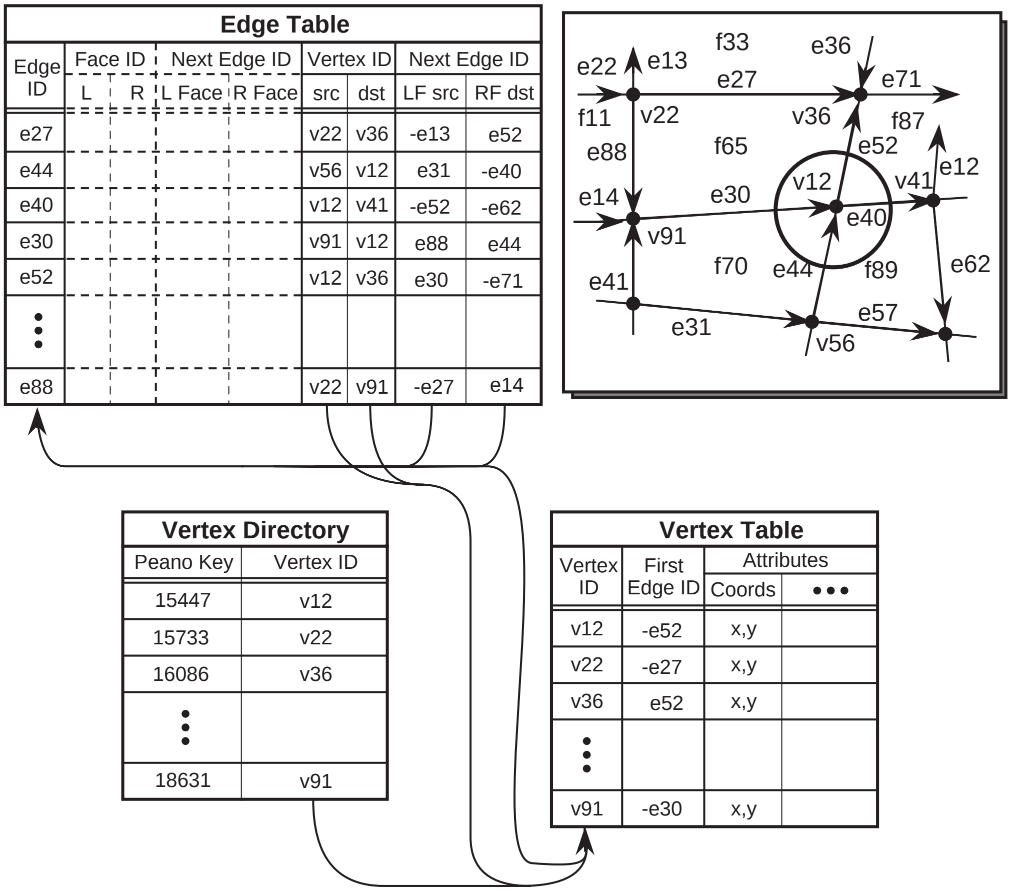


TIGER TOPOLOGICAL CODING

- Objects
  1. 0-cells: vertices or points (nodes)
  2. 1-cells: edges or lines (segments)
  3. 2-cells: faces or areas (blocks, census tracts, enumeration districts)
- Method of access by directories
  1. Vertex Directory: index by Peano key of a vertex
     - points at a Vertex Table entry
  2. Face Directory: index by Peano key of a point in a face
     - points at a Face Table entry
- Relationships
  1. vertex relationships (Vertex Table and Edge Table) enable accessing all edges that meet at a vertex
  2. face relationships (Face Table and Edge Table) enable accessing all edges that form a face
  3. Edge Table is a common link between both relationships
  4. similar to winged-edge data structure of Baumgart

TIGER VERTEX RELATIONSHIPS

- Vertex Directory:
  - access by looking up Peano key of coordinates of the vertex
- Vertex Table: correspondence between a vertex and an edge in a list of edges incident at it stored in Edge Table
- Edge Table: circular list in counter-clockwise order of edges that meet at a vertex
  1. one list for each of the 2 vertices comprising the edge
  2. negative next edge link if the edge is directed away from the vertex and positive otherwise
  3. actual implementation in TIGER uses a linked list with the edges in no particular order
- Ex: all edges meeting at vertex v12



Edge Table

Edge ID	Face ID L	Face ID R	Next Edge ID L Face	Next Edge ID R Face	Vertex ID src	Vertex ID dst	Next Edge ID LF src	Next Edge ID RF dst
e27					v22	v36	-e13	e52
e44					v56	v12	e31	-e40
e40					v12	v41	-e52	-e62
e30					v91	v12	e88	e44
e52					v12	v36	e30	-e71
⋮								
e88					v22	v91	-e27	e14

Vertex Directory

Peano Key	Vertex ID
15447	v12
15733	v22
16086	v36
⋮	
18631	v91

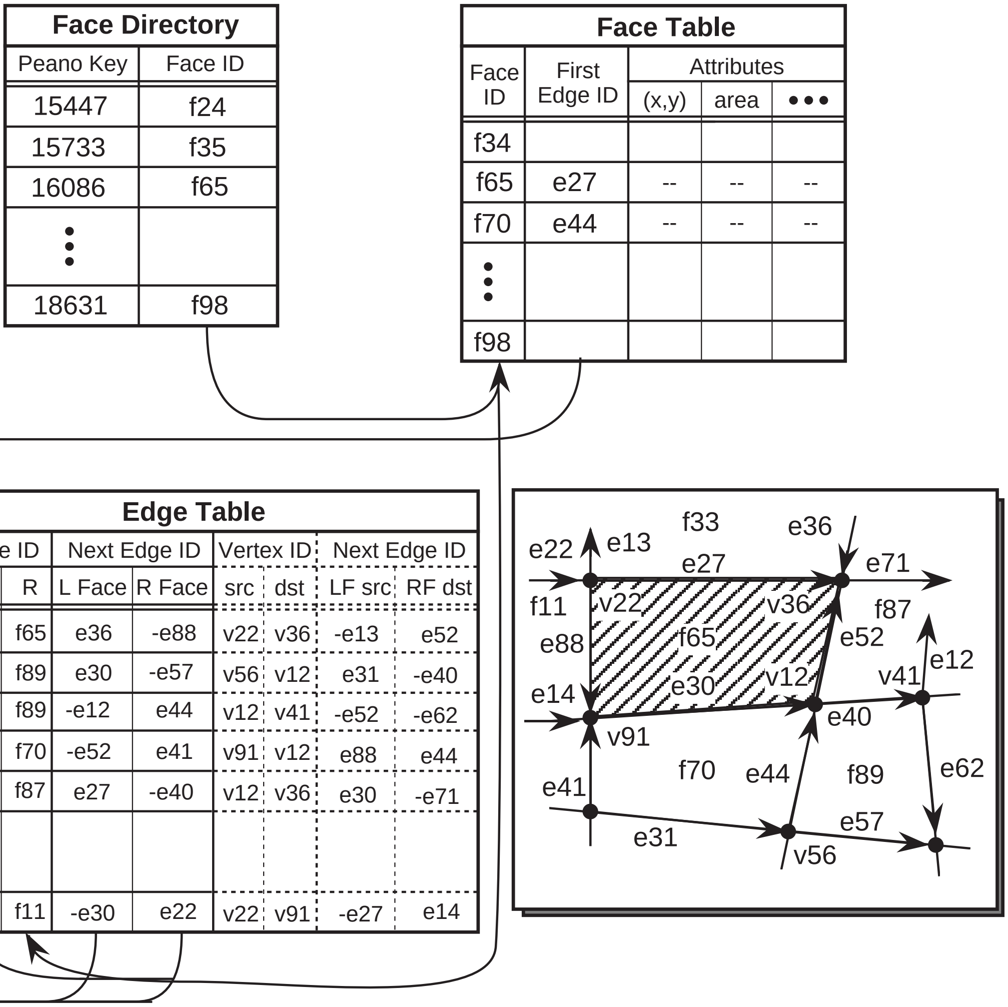
Vertex Table

Vertex ID	First Edge ID	Attributes Coords	•••
v12	-e52	x,y	
v22	-e27	x,y	
v36	e52	x,y	
⋮			
v91	-e30	x,y	

TIGER FACE RELATIONSHIPS

- Face Directory: index by Peano key of a point in a face
- Face Table: correspondence between a face and the first edge in list of edges that comprise it stored in Edge Table
- Edge Table: yields a circular list of consecutive edges in counterclockwise order that make up a face by recording the adjacent faces and the next edges along them
  1. one linked list for each of the 2 faces adjacent to the edge
  2. negative next edge link if the edge is directed away from the destination vertex of the edge and positive otherwise

Ex: all edges around face f65:



Face Directory

Peano Key	Face ID
15447	f24
15733	f35
16086	f65
⋮	
18631	f98

Face Table

Face ID	First Edge ID	Attributes		
		(x,y)	area	•••
f34				
f65	e27	--	--	--
f70	e44	--	--	--
⋮				
f98				

Edge Table

Edge ID	Face ID		Next Edge ID		Vertex ID		Next Edge ID	
	L	R	L Face	R Face	src	dst	LF src	RF dst
e27	f33	f65	e36	-e88	v22	v36	-e13	e52
e44	f70	f89	e30	-e57	v56	v12	e31	-e40
e40	f87	f89	-e12	e44	v12	v41	-e52	-e62
e30	f65	f70	-e52	e41	v91	v12	e88	e44
e52	f65	f87	e27	-e40	v12	v36	e30	-e71
⋮								
e88	f65	f11	-e30	e22	v22	v91	-e27	e14

Copyright © 2002 by Hanan Samet

EXAMPLE OF A COMPLETE TIGER STRUCTURE

Face Directory

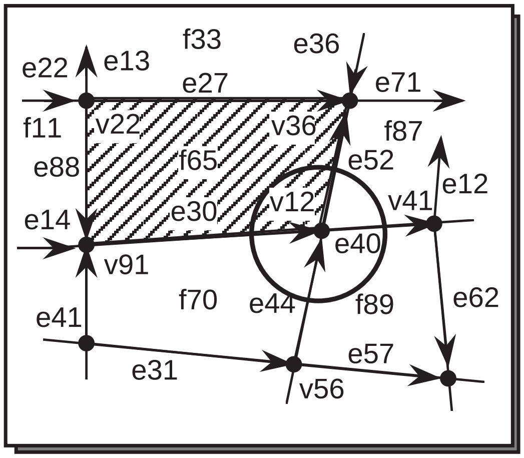
Peano Key	Face ID
15447	f24
15733	f35
16086	f65
⋮	
18631	f98

Face Table

Face ID	First Edge ID	Attributes		
		(x,y)	area	•••
f34				
f65	e27	--	--	--
f70	e44	--	--	--
⋮				
f98				

Edge Table

Edge ID	Face ID		Next Edge ID		Vertex ID		Next Edge ID	
	L	R	L Face	R Face	src	dst	LF src	RF dst
e27	f33	f65	e36	-e88	v22	v36	-e13	e52
e44	f70	f89	e30	-e57	v56	v12	e31	-e40
e40	f87	f89	-e12	e44	v12	v41	-e52	-e62
e30	f65	f70	-e52	e41	v91	v12	e88	e44
e52	f65	f87	e27	-e40	v12	v36	e30	-e71
⋮								
e88	f65	f11	-e30	e22	v22	v91	-e27	e14



Vertex Directory

Peano Key	Vertex ID
15447	v12
15733	v22
16086	v36
⋮	
18631	v91

Vertex Table

Vertex ID	First Edge ID	Attributes	
		Coords	•••
v12	-e52	x,y	
v22	-e27	x,y	
v36	e52	x,y	
⋮			
v91	-e30	x,y	

- Edge Table has an index on Edge ID field

DIGITAL LINE GRAPH (DLG)

Vertex Table

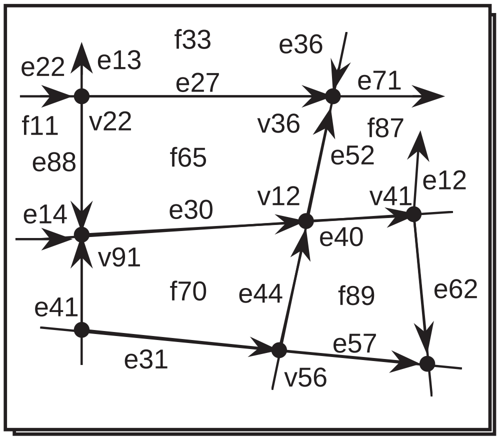
Vertex ID	Coordinate
v12	x,y
v22	x,y
v36	x,y
v41	x,y
⋮	
v91	x,y

Face Table

Face ID	Interior Point
f65	x,y
f70	x,y
f89	x,y
⋮	

Edge Table

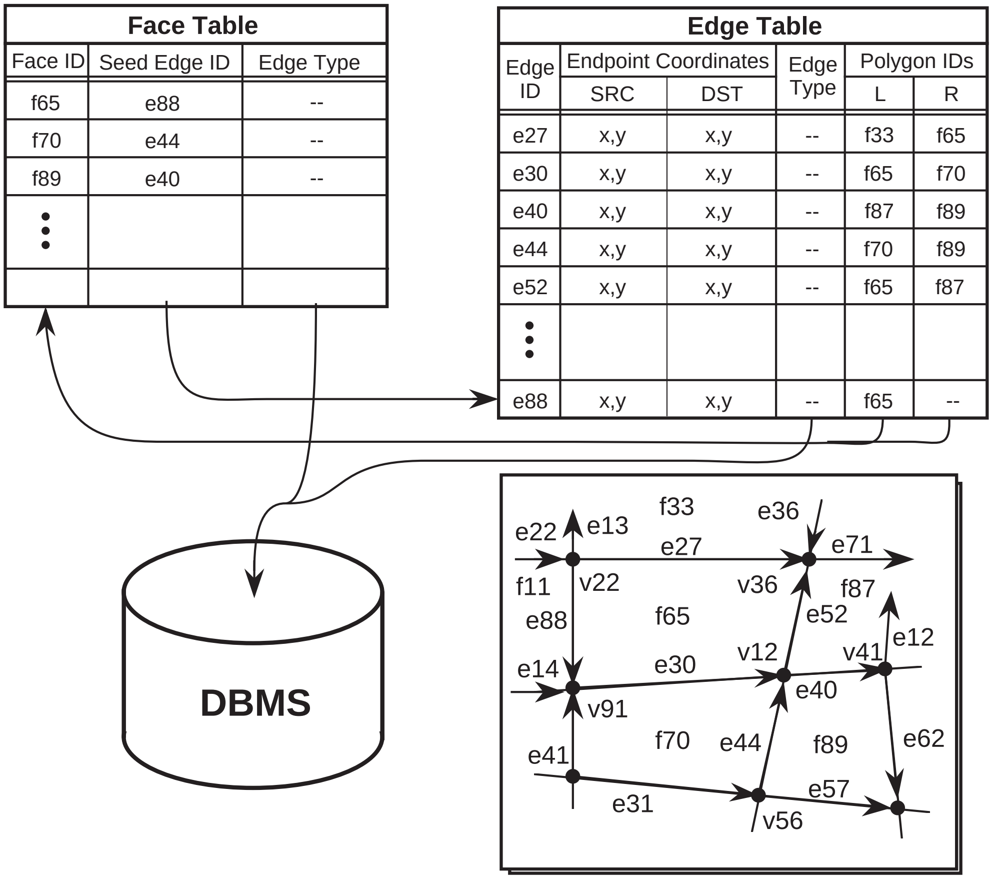
Edge ID	Vertex IDs		Face IDs		Coords	
	SRC	DST	L	R	SRC	DST
e27	v22	v36	f33	f65	x,y	x,y
e30	v91	v12	f65	f70	x,y	x,y
e40	v12	v41	f87	f89	x,y	x,y
e44	v56	v12	f70	f89	x,y	x,y
⋮						
e88	v22	v91	f65	f11	x,y	x,y



- Really just a file format (i.e., data repository)
- Vertex Table contains shape points as well
- Edge Table only contains vertices where real edges meet
- Somewhat cumbersome for performing operations

UNIFIED PMR QUADTREE

- Bucketing method for storing collections of line segments
- Facilitates finding nearest edge to a given point (i.e., spatial indexing)
- Subdivide a block once into four equal blocks whenever it contains more than *s* (splitting threshold) line segments
- If the total number of line segments in four brother blocks is less than *s*, then merge as often as the condition holds
- Edge Table indicates endpoints of line segments and identity of the polygons to their left and right
- Given the identity of a face, get one edge via Face Table and rest of edges by neighbor finding



Face Table

Face ID	Seed Edge ID	Edge Type
f65	e88	--
f70	e44	--
f89	e40	--
⋮		

Edge Table

Edge ID	Endpoint Coordinates		Edge Type	Polygon IDs	
	SRC	DST		L	R
e27	x,y	x,y	--	f33	f65
e30	x,y	x,y	--	f65	f70
e40	x,y	x,y	--	f87	f89
e44	x,y	x,y	--	f70	f89
e52	x,y	x,y	--	f65	f87
⋮					
e88	x,y	x,y	--	f65	--

SAMPLE IMPLEMENTATIONS

1. ARC/INFO (ESRI)
   - topological model with layers
2. TIGRIS (INTERGRAPH)
   - topological model with just one layer
   - decriptions of objects that are part of different attributes are shared
   - primitives are directed edges
3. SYSTEM9 (Wild, Computervision, Prime, UNISYS)
   - topological model with just one layer
   - primitives are nodes, lines, and polygons

STANDARD DATA FORMATS

1. No real standard!

2. Defense Mapping Agency (DMA)
   - Digital Feature Analysis Data (DFAD)
     a. description of land surface in terms of
        - culture
        - geographic (e.g., forests, lakes, etc.)

     b. stored as point, line, and area data
        - lists of or single latitude, longitude pairs
        - accompanied by tables of attribute information

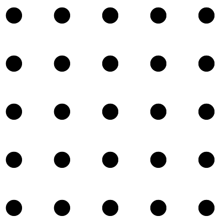
     c. e.g., water tower is stored as a point whose attributes are feature high and structure type
   - Digital Terrain Elevation Data (DTED)
     a. elevation data in latitude, longitude, and elevation
     b. at 3 arc second intervals
     c. referenced to sea level
     d. rounded to nearest meter

3. US Geological Survey (USGS)
   - Digital Line Graph (DLG)
     a. nodes (endpoints)
     b. lines
        - identifier
        - endpoint nodes
        - adjacent areas
     c. areas
        - identifier
        - address in the area
   - Digital Elevation Model (DEM)
   - Land use and land cover data (GIRAS)

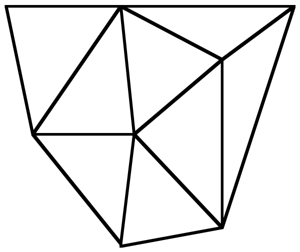
4. CIA World Data Bank
   - just sequences of x and y coordinates
   - no topological structure
   - just coastlines?
5. National Ocean Service (NOS and NOAA)
   - nautical charts depicting hydrography and bathymetry (depths)
     a. latitude
     b. longitude
     c. depth
   - aeronautical charts depicting visual and instrument flight charts
6. National Geophysical Center (NOS)
   - surface elevations and ocean depths
   - at 10 minute increments for latitude and longitude

DIGITAL ELEVATION DATA

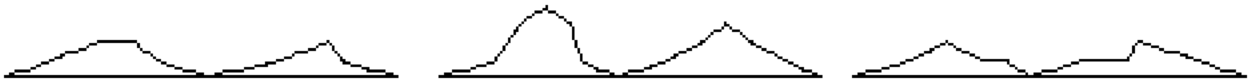
1. Regular grid
   - one elevation value for each of a set of regularly spaced positions
   - disadvantage is that the density of elevations is uniform
   - want high density in complex terrain and sparseness in level areas



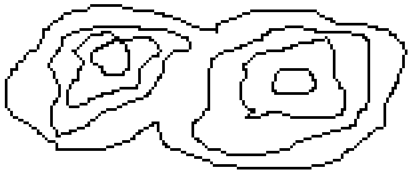
2. Triangulated Irregular Network (TIN)
   - irregularly spaced elevation points
   - topography represented by network of triangular facets
   - preferable when triangles are equilateral in shape



3. Profiles
   - show elevations at points along a series of parallel lines
   - vertical slices

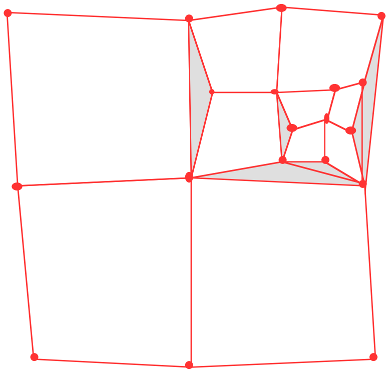


4. Contours
   - topographic surface is represented by series of elevation points taken along individual contour lines
   - horizontal slices are overlaid and do not cross since the 2.5-d is single-valued



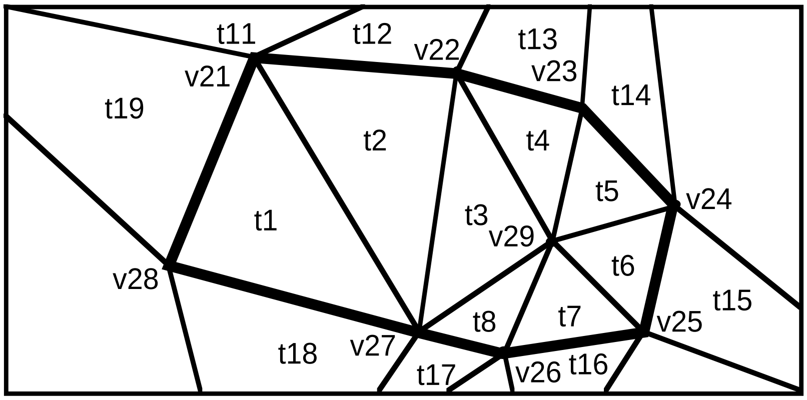
TRIANGULATED IRREGULAR NETWORK (TIN)

1. Map data collection usually tabulates data at points
   - points have “high information content”
     a. mountain peaks
     b. bottoms of valleys and depressions
     c. saddle points
     d. break points in slopes
   - interpolate between points
     a. assume a plane between triplets of points
     b. triplets of points form irregular triangles
     c. connect irregular triangles to form a network
2. Advantages of TIN lie in variable size triangles (facets)
   - extra information is encoded for areas of complex relief
   - simple relief is represented by large triangles
   - avoids cracks
3. Example applications
   - modeling of hill slopes and streams
   - intervisibility problems—i.e., what is visible from where



TRIANGLE TIN REPRESENTATION

- Primary entity is a triangle
- Vertex table
  1. *x* and *y* coordinate values
  2. elevation
- Triangle table
  1. pointers to records of constituent vertices (3)
  2. pointers to records of adjacent triangles (maximum 3)
- No particular order for entries in tables
- Good for display, hidden surface removal

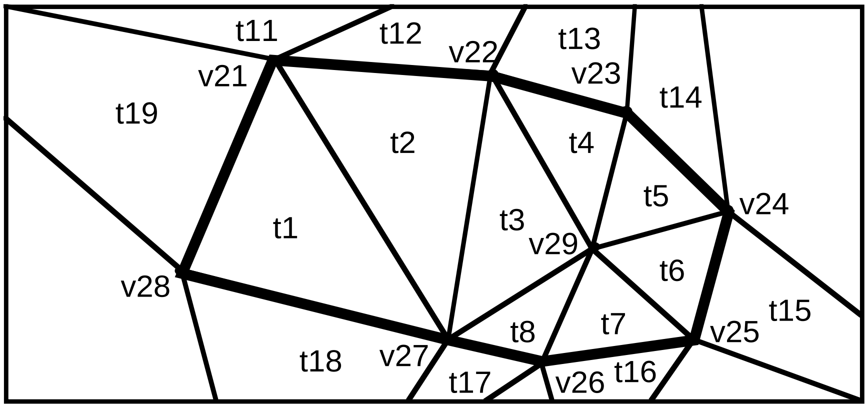


Vertex Table			
Vertex ID	Coordinate		Elevation
	X	Y	
v21	x	y	z
v22	x	y	z
v23	x	y	z
v24	x	y	z
v25	x	y	z
v26	x	y	z
v27	x	y	z
v28	x	y	z
v29	x	y	z

Triangle Table						
Tri. ID	Connected Vert.			Adjacent Triangles		
	1	2	3	1	2	3
t1	v21	v27	v28	t2	t18	t19
t2	v21	v22	v27	t1	t3	t12
t3	v22	v27	v29	t2	t4	t8
t4	v22	v23	v29	t3	t5	t13
t5	v23	v24	v29	t4	t6	t14
t6	v24	v25	v29	t5	t7	t15
t7	v25	v26	v29	t6	t8	t16
t8	v26	v27	v29	t3	t7	t17

CONNECTED VERTEX TIN REPRESENTATION

- For each vertex, keep track of all vertices connected to it
- Vertex table
  1. *x* and *y* coordinate values and elevation
  2. pointer to 1st element in vertex list in connectivity table
- Connectivity table
  1. consecutive pointers to records of adjacent vertices
  2. each list is terminated by 0
- Takes about half the space of triangle TIN representation
- Enables editing—e.g., addition and deletion of points
- Does not keep track of triangles explicitly



Vertex Table

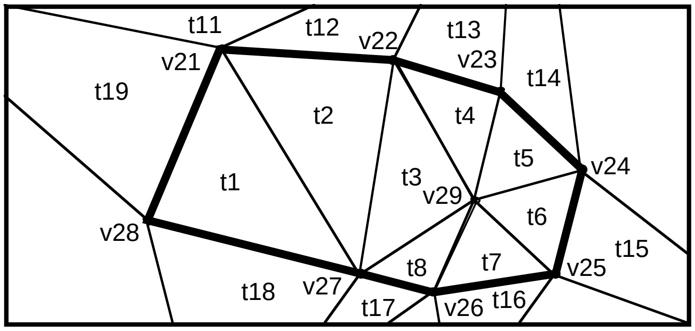
Vertex ID	Coordinate		Elevation	First Vertex
	X	Y		
v21	x	y	z	--
v22	x	y	z	--
v23	x	y	z	8
v24	x	y	z	--
v25	x	y	z	--
v26	x	y	z	--
v27	x	y	z	--
v28	x	y	z	--
v29	x	y	z	1

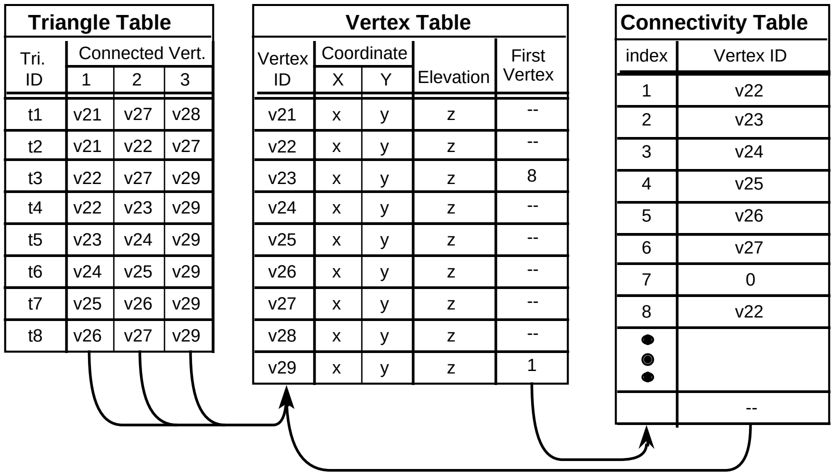
Connectivity Table

index	Vertex ID
1	v22
2	v23
3	v24
4	v25
5	v26
6	v27
7	0
8	v22
● ● ●	
	--

HYBRID TIN REPRESENTATION

- Combine vertex method with a list of triangles
- Does not include the pointers to the adjacent triangles
- Comparable in size to other data structures
- Enables easy display and editing
  1. move point: easy to detect which edges need updating
  2. add point: find enclosing triangle and connect point to its vertices
  3. delete point: remove it from the Vertex Table and its Connectivity Table entry
     - may need to retriangulate
  4. can calculate volume
  5. display using hidden surface removal by maintaining the normal to each triangle and ignoring all triangles which face away from viewer





Triangle Table

Tri. ID	Connected Vert. 1	2	3
t1	v21	v27	v28
t2	v21	v22	v27
t3	v22	v27	v29
t4	v22	v23	v29
t5	v23	v24	v29
t6	v24	v25	v29
t7	v25	v26	v29
t8	v26	v27	v29

Vertex Table

Vertex ID	Coordinate X	Y	Elevation	First Vertex
v21	x	y	z	--
v22	x	y	z	--
v23	x	y	z	8
v24	x	y	z	--
v25	x	y	z	--
v26	x	y	z	--
v27	x	y	z	--
v28	x	y	z	--
v29	x	y	z	1

Connectivity Table

index	Vertex ID
1	v22
2	v23
3	v24
4	v25
5	v26
6	v27
7	0
8	v22
⋮	
	--

SYMBOLS

- Renders visible the features and locations on a map
- Distinguishes between the relevant and the irrelevant
- Three geometric categories
  1. points
  2. lines
  3. regions
- Six visual variables
  1. size
  2. shape
  3. graytone values
  4. texture (pattern)
  5. orientation
  6. hue (color)

USE OF VISUAL VARIABLES

- Often combine them to yield a particular effect
- Shape, texture, and hue highlight qualitative differences - e.g., vegetation, religion, etc.
- Quantitative differences
  1. size for count or capacity (e.g., road network)
  2. graytone for rate or intensity
     - light is associated with less
     - dark is associated with more
     - difficult to achieve with hue
     - e.g., proportional voting
- Orientation symbols (e.g., arrows) are useful for directional quantities
  1. weather such as wind, gulfstream
  2. population shifts such as migration patterns, troop movements
  3. traffic flow
  4. direction of hatching in area symbolization by lines
- Hard to detect differences in graytone, and pattern for point and line symbols while yes for area symbols
- Hue is not good for differentiating between point symbols but yes for line symbols (e.g., type of road)
- Can combine two or more visual variables (e.g., elevation contours)
  1. dense spacing (an element of texture) shows steep slope while wide spacing shows gentle slope
  2. perpendicular to contour line (an element of orientation) shows downhill direction

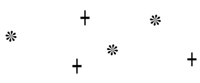
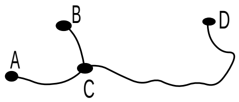
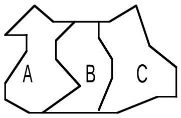
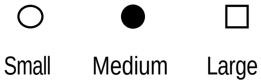
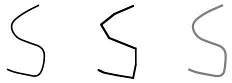
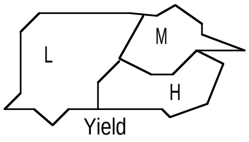
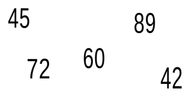
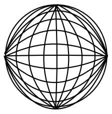
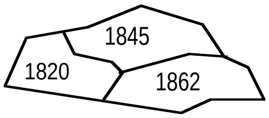
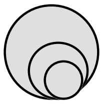
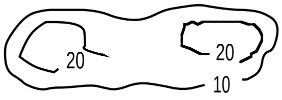
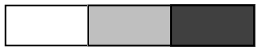
PERCEPTUAL ERROR

- Caused by poor match between data and visual variable
- Improper use of color to illustrate order
  1. layman does not understand ordering implied by wavelengths
  2. use graytone!
  3. many people are color blind
- Certain colors are associated with preconceived notions
  1. blue for water or cold
  2. brown for barren land
  3. green for lush vegetations or forests
  4. red for warm
- Point symbols may be better at capturing size than area symbols
  1. especially true for non-spatial variables such as population
  2. don't fill an area with an area symbol such as hue, texture, or graytone
     - area symbols suggest intensity
     - a small region with a large population is represented in the same way as a large region with a relatively small but equal population
  3. vary size of point symbol

CLASSIFICATION OF CARTOGRAPHIC DATA TYPES

1. Dimension
   - point - described by location and attribute
   - line - string of connected line segments or a mathematical function such as a spline
   - area - predefined with a boundary, topology, and attribute
2. Level of measurements - i.e., by its complexity
   - nominal - e.g., place name
   - ordinal
     a. has a sequence or ranking
     b. can use relations such as 'greater than', 'smaller' etc.
   - interval
     a. measured numerical value
     b. scale need not be absolute
   - can be pegged to an arbitrary zero
   - e.g., BC, AD
   - ratio
     a. measured on a scale with a meaningful zero so mathematical operations can be performed on it
     b. e.g., density, rate of change, percentage

- Robinson: a way of classifying cartographic symbols and map types in common use
- Symbolization depends on
  1. type of data (i.e., dimension)
  2. level of attribute measurement

Content Scaling Level	Defining Relations	FORM OF CARTOGRAPHIC SYMBOL		
		POINT	LINE	AREA
Nominal	Equivalence	 Wholesale and Retail Establishments	 Highway Connectivity	 Land Ownership
Ordinal	Equivalence Greater than	 Population Centers	 Roads by Degree of Improvement	
Interval	Equivalence Greater than Ratio of Intervals	 Spot Elevations	 Latitude/Longitude Grid	 Date of First Settlement
Ratio	Equivalence Greater than Ratio of Intervals Ratio of Scale Values	 Area Proportional to Population	 Population Density Isopleths	 Darkness Proportional to Population Density

Adapted from Figure 8.01 from *Analytical and Computer Cartography* by Keith C. Clarke, Prentice-Hall, Englewood Cliffs, NJ, 1990, p. 133.

- Unwin: added the distinction between map-data transformations and map-type transformations
  1. data attributes define data types
  2. type of symbolization defines the map type
  3. result is a data-type to map-type symbolization transformation
  4. analytical cartography is the study of the above transformation
- The separation between data type and map types is based on the distinction between cartographic entities and their symbolization

DATA TYPES

	Point	Line	Area	Volume
Nominal	City	Road	Name of Unit	Precipitation or soil type
Ordinal	Large City	Major Road	Rich County	Heavy precip. Good soil
Interval	Total Population	Traffic Flow	Per Capita Income	Precip. in mm or Cation Exchange
Ratio				

MAP TYPES

	Point	Line	Area	Volume
Nominal	Dot Map	Network Map	Colored Area Map	Freely Colored Map
Ordinal	Symbol Map	Ordered Network Map	Ordered Colored Map	Ordered Chromatic Map
Interval	Graduated Symbol Map	Flow Map	Choropleth Map	Contour Map
Ratio				

Adapted from Figure 8.03 from *Analytical and Computer Cartography* by Keith C. Clarke, Prentice-Hall, Englewood Cliffs, NJ, 1990, p. 138.

WHY TRANSFORM CARTOGRAPHIC DATA?

1. Cartographic generalization—i.e., changing scale
2. Conversion of the geometry of the map base
   - in registration—i.e., common origin
   - common statistical basis
3. Change the data structure
   - e.g., contouring may require data to be on a grid
   - converting from raster to vector or vice versa because certain operations are more efficient with them
   - vector to raster transformation is not invertible which is why we store with each block in a PM quadtree the identity of the line passing through it
4. Change the level of measurement
   - not necessarily a map transformation

     Ex: make a map of the US where the states are labeled with their ranking in terms of population density

     - result is a choropleth map since each location in a region (i.e., state) has the same value (i.e., population density)
     - two transformations

       a. ratio (population and state area) to ratio (population density)

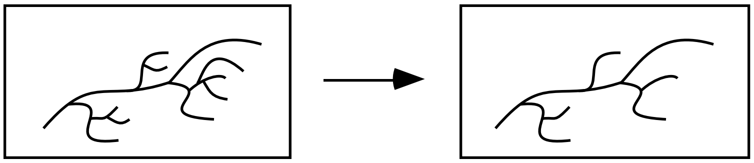
       b. ratio (population density) to ordinal (ranking)
     - ratio to ratio is invertible but ratio to ordinal is not

SCALE TRANSFORMATIONS

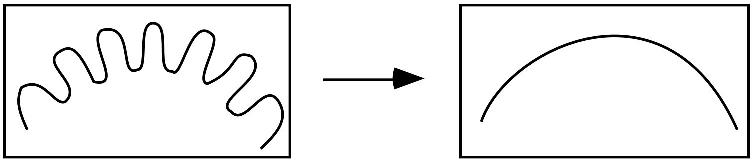
- Large scale means a small area on the ground is covered with much detail (small denominator)
- Generalization is transformation from large scale to small scale
  1. geometric generalization avoids or eliminates overlapping symbols
  2. content generalization filters out details irrelevant to the map's theme
     - selection suppresses some information
     - classification recognizes similarity among a group of features by using one type of symbol to represent them
- Inverse of generalization is “enhancement” (also known as cartographic license)
  1. emphatic enhancement
     - uniformly done over the entire map (though artificial)
  2. synthetic enhancement
     - uses a model to generate the variation
     - e.g., fractal enhancement of coastlines

EXAMPLES OF GENERALIZATION

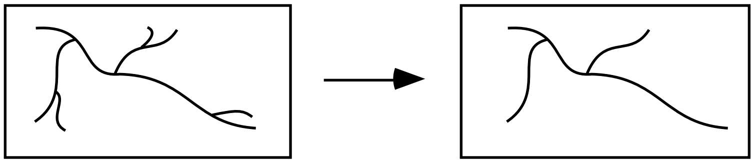
1. Elimination or selection
   - preserve average feature density
   - choose features to be retained and suppressed



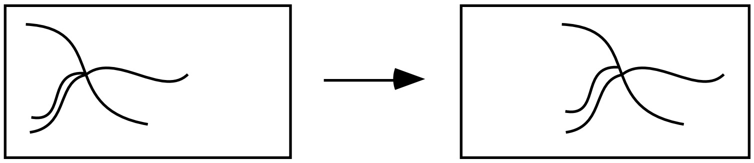
2. Simplification
   - type conversion
     a. using a dot for a city instead of an area
     b. using a line for a wide river instead of an area
   - smoothing a line for a river or road



3. Combination or aggregation
   - joining features—e.g., a river with its tributaries
   - area conversion
     a. convert a group of dots to an area
     b. especially useful when scale reduction is severe (e.g., from 1 : 5,000 to 1 : 1,000,000) since it diverts reader's attention from imperceptible individual occurrences to regions of relative concentration
   - single line for a divided highway



4. Displacement
   - avoid interference due to:
     a. changing symbol and line widths
     b. minimum feature separation requirements
   - reveal structure
   - make room for labels



5. Enhancement
   - add details to give an appearance of realism
   - Ex: use hairpin symbol for a winding road
   - Ex: fractal coastlines

DIFFERENTIATING BETWEEN DIFFERENT GENERALIZATION OPERATIONS

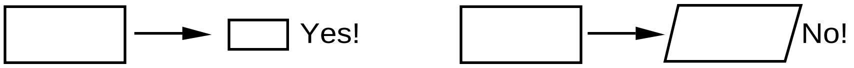
1. Filtering consists of elimination and simplification
   - do not cause locational changes of the features
   - local operations
   - can be done sequentially
   - primarily for data reduction
2. Generalized generalization includes combination and displacement (also known as *resymbolization*)
   - reorganize space
   - best if done in parallel
   - oriented towards graphical output

MAP-BASED TRANSFORMATIONS

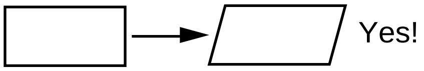
1. Projections
   - projections from a sphere to a plane
   - actually earth is spheroid but difference is small
   - geometry is different on sphere and plane (e.g., spherical triangles)
2. Affine transformations—shifting, rotation, scaling
3. Statistical transformations
   - rubber sheeting—stretch map to fit on a surface
     a. don’t know the transformation exactly
     b. know the transformation of a few points
   - cartograms—draw map so areas of constituent entities are proportional to the quantity measured while preserving the shape
4. Symbolization transformations
   - viewing transformation—e.g., normalization to workstation window
   - symbols for different data types

GEOMETRIC TRANFORMATIONS

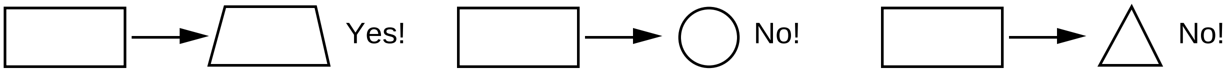
1. Equi-area
   - preserves area of object
2. Similarity
   - preserves shape but not necessarily area



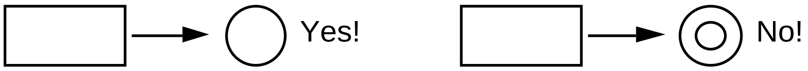
3. Affine
   - permits angular distortion but preserves parallelism of lines and collinearity
   - scaling, rotation, translation



4. Projective
   - permits distortion of shape and angle
   - e.g., number of vertices and edges is preserved



5. Topological
   - preserves neighborhood



- Rubber sheeting: collective term describing the combination of affine, projective, and topological geometric transformations
- Conflation: matching of features on two maps which differ due to distortions, slivers, superfluous points, etc.

AFFINE (LINEAR) TRANSFORMATIONS

- Transformations includes translation, rotation, and scaling
- Assume two-dimensional space
- Combination is facilitated by use of matrix multiplication

$$C = A \cdot B \quad \text{where} \quad c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

- Use homogeneous coordinates so that matrix multiplication can be used for translation instead of addition

1. translation: $x' = x + a$ and $y' = y + b$ $\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix}$

2. scaling: $x' = s \cdot x$ and $y' = s \cdot y$ $\begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix}$

3. rotation: $x' = x \cdot \cos\theta - s \cdot \sin\theta$, $y' = x \cdot \sin\theta + y \cdot \cos\theta$ $\begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$

- Can specify operations in sequence

Ex:
```
rotation(90);
translation(5,2);
scaling(2);
rotation(–45);
...
```

RESAMPLING TRANSFORMATIONS

1. Change scale
   - symbolize cartographic objects at a different scale than the one at which geocoding took place
   - for ease of symbolization
2. Point-to-Point
   - one point represents many points—e.g., centroid, Voronoi diagram
3. Line-to-line
   - reduce number of elements for representing the line
   - purpose: convey character of line to map reader
4. Area-to-area
   - goal: merge multiple data sets into a set of regions so the merged data facilitates a comparison between maps
   - compute a set of greatest common geographic units—i.e., they do not need further partitioning
     a. convert to a grid
     b. polygon overlay
        - find intersection points between lines
        - split chains
        - form new polygons
5. Volume-to-volume: rare as hard to symbolize in 3-d

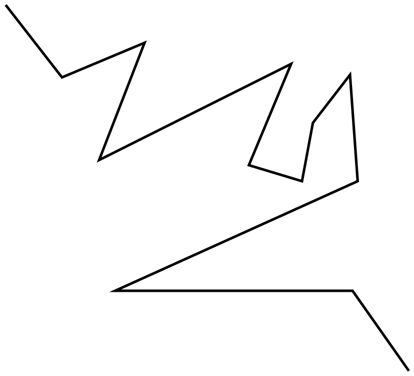
VOLUME-TO-VOLUME TRANSFORMATIONS

1. Change in grid spacing
   - requires interpolation
   - use four neighbors and average

2. Change in TIN surface representation
   - rare for set of points that represent the surface to change as the points are real observations
   - can create a grid from a TIN by using interpolation
   - can determine which triangle contains a particular point by using a point-in-polygon test

LINE-TO-LINE TRANSFORMATIONS

- Bottom-up line generalization
- Purpose: data reduction

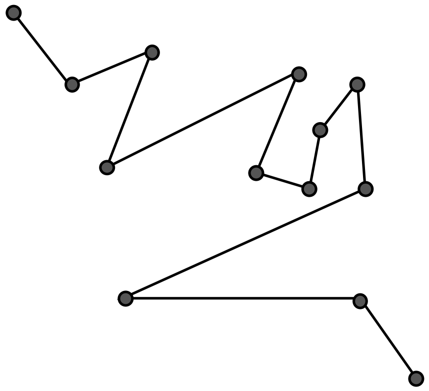


- Algorithm
- Question: Can we guarantee absence of slivers?
- Local methods

Adapted from Figure 10.01 from *Analytical and ComputerCartography* by Keith C. Clarke, Prentice-Hall, EnglewoodCliffs, NJ, 1990, p. 181.

LINE-TO-LINE TRANSFORMATIONS

- Bottom-up line generalization
- Purpose: data reduction

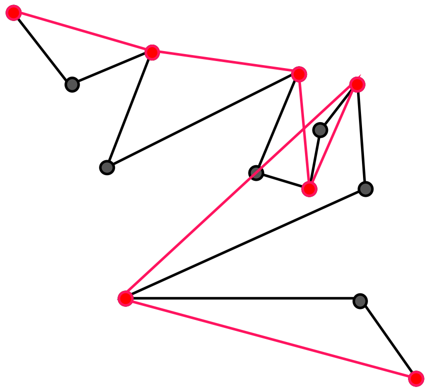


- Algorithm
- Question: Can we guarantee absence of slivers?
- Local methods

Adapted from Figure 10.01 from *Analytical and ComputerCartography* by Keith C. Clarke, Prentice-Hall, EnglewoodCliffs, NJ, 1990, p. 181.

LINE-TO-LINE TRANSFORMATIONS

- Bottom-up line generalization
- Purpose: data reduction



- Algorithm

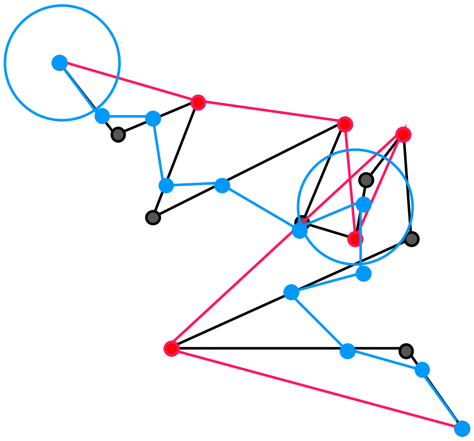
1. *N* th point elimination

- Question: Can we guarantee absence of slivers?
- Local methods

Adapted from Figure 10.01 from *Analytical and ComputerCartography* by Keith C. Clarke, Prentice-Hall, EnglewoodCliffs, NJ, 1990, p. 181.

LINE-TO-LINE TRANSFORMATIONS

- Bottom-up line generalization
- Purpose: data reduction



- Algorithm

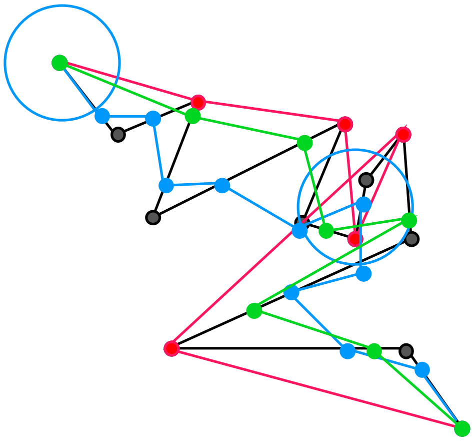
  1. *N* th point elimination
  2. all points equidistant from each other in terms of Euclidean distance

- Question: Can we guarantee absence of slivers?
- Local methods

Adapted from Figure 10.01 from *Analytical and ComputerCartography* by Keith C. Clarke, Prentice-Hall, EnglewoodCliffs, NJ, 1990, p. 181.

LINE-TO-LINE TRANSFORMATIONS

- Bottom-up line generalization
- Purpose: data reduction



- Algorithm

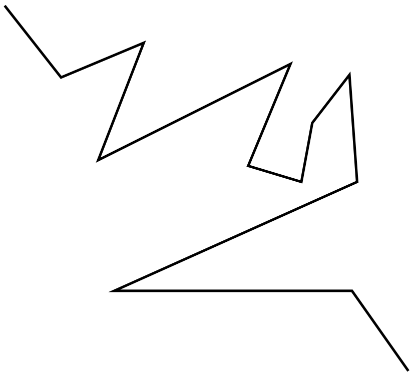
  1. *N* th point elimination
  2. all points equidistant from each other in terms of Euclidean distance
  3. retain a point for every x distance units along the line

- Question: Can we guarantee absence of slivers?
- Local methods

Adapted from Figure 10.01 from *Analytical and ComputerCartography* by Keith C. Clarke, Prentice-Hall, EnglewoodCliffs, NJ, 1990, p. 181.

DOUGLAS-PEUCKER ALGORITHM

- Top-down line generalization
- Algorithm



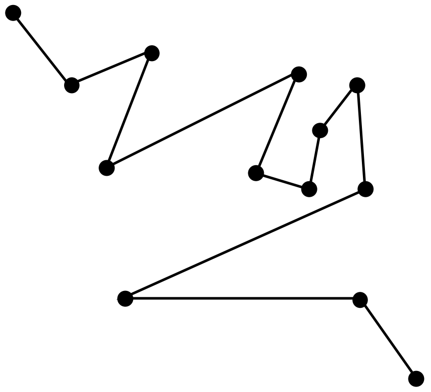
- Global method of data reduction

Adapted from Figure 10.02 from *Analytical and ComputerCartography* by Keith C. Clarke, Prentice-Hall, EnglewoodCliffs, NJ, 1990, p. 184.

DOUGLAS-PEUCKER ALGORITHM

- Top-down line generalization

- Algorithm



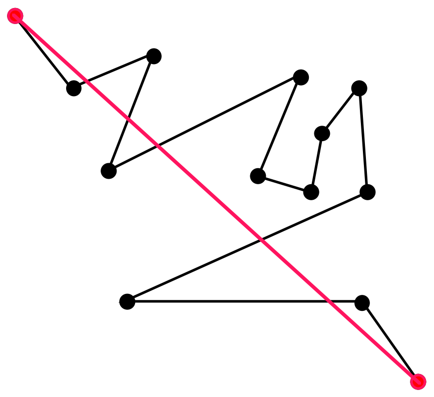
- Global method of data reduction

Adapted from Figure 10.02 from *Analytical and ComputerCartography* by Keith C. Clarke, Prentice-Hall, EnglewoodCliffs, NJ, 1990, p. 184.

DOUGLAS-PEUCKER ALGORITHM

- Top-down line generalization
- Algorithm

  1. Join extreme endpoints



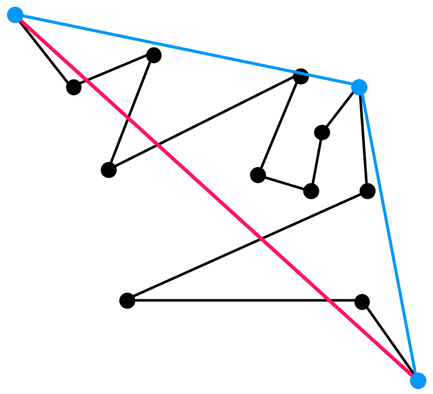
- Global method of data reduction

Adapted from Figure 10.02 from *Analytical and ComputerCartography* by Keith C. Clarke, Prentice-Hall, EnglewoodCliffs, NJ, 1990, p. 184.

DOUGLAS-PEUCKER ALGORITHM

- Top-down line generalization
- Algorithm

  1. Join extreme endpoints
  2. Select point on line with largest orthogonal distance to line approximation and break line at this point



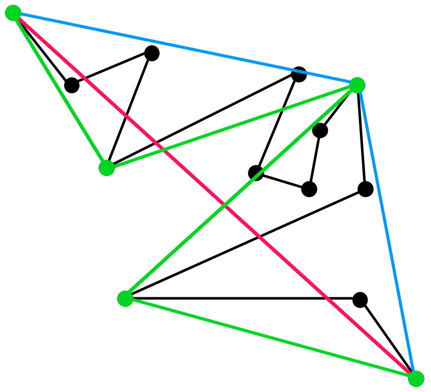
- Global method of data reduction

Adapted from Figure 10.02 from *Analytical and ComputerCartography* by Keith C. Clarke, Prentice-Hall, EnglewoodCliffs, NJ, 1990, p. 184.

DOUGLAS-PEUCKER ALGORITHM

- Top-down line generalization
- Algorithm

  1. Join extreme endpoints
  2. Select point on line with largest orthogonal distance to line approximation and break line at this point
  3. Apply recursively until
     - a minimum number of points remain, OR



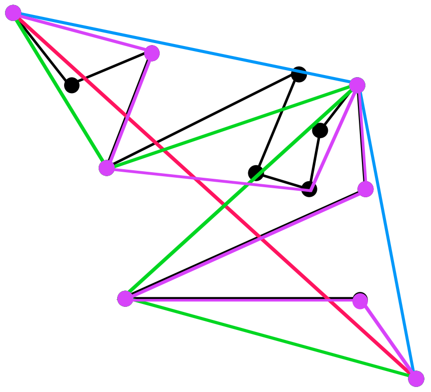
- Global method of data reduction

Adapted from Figure 10.02 from *Analytical and ComputerCartography* by Keith C. Clarke, Prentice-Hall, EnglewoodCliffs, NJ, 1990, p. 184.

DOUGLAS-PEUCKER ALGORITHM

- Top-down line generalization
- Algorithm

  1. Join extreme endpoints
  2. Select point on line with largest orthogonal distance to line approximation and break line at this point
  3. Apply recursively until
     - a minimum number of points remain, OR
     - reach a specified tolerance level such as a fraction of the initial orthogonal distance, OR



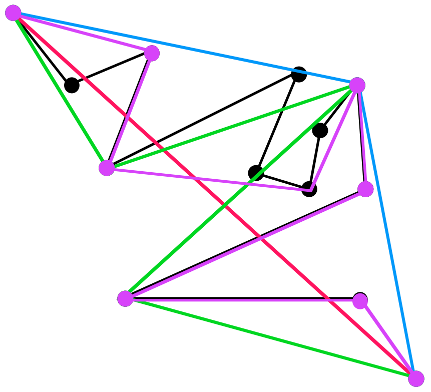
- Global method of data reduction

Adapted from Figure 10.02 from *Analytical and ComputerCartography* by Keith C. Clarke, Prentice-Hall, EnglewoodCliffs, NJ, 1990, p. 184.

DOUGLAS-PEUCKER ALGORITHM

- Top-down line generalization
- Algorithm

  1. Join extreme endpoints
  2. Select point on line with largest orthogonal distance to line approximation and break line at this point
  3. Apply recursively until
     - a minimum number of points remain, OR
     - reach a specified tolerance level such as a fraction of the initial orthogonal distance, OR
     - preservation of the fractal dimension of the line



- Global method of data reduction

Adapted from Figure 10.02 from *Analytical and ComputerCartography* by Keith C. Clarke, Prentice-Hall, EnglewoodCliffs, NJ, 1990, p. 184.

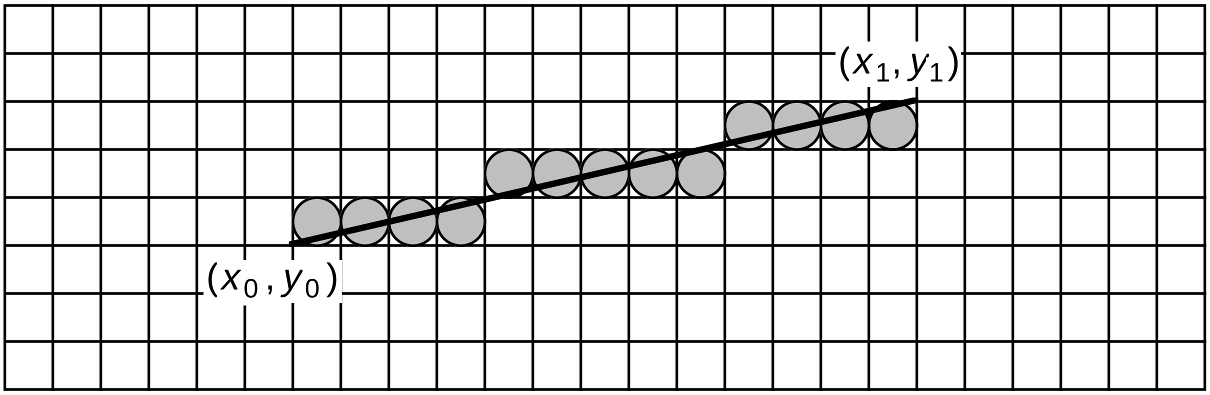
VECTOR TO RASTER CONVERSION (RASTERIZATION)

1. Read the data
   - map projections
   - resampling transformations
2. Apply necessary scaling and map transformations
3. Determine nonzero pixels
   - if distance between two points is less than half the grid spacing, then ignore the second point
   - use Bresenham's algorithm—only requires integers
   - any pixel that is partially touched by a line is nonzero
     a. could cause spurious holes
     b. avoid holes by checking if a pixel is necessary to preserve connectivity
4. Store the array in an appropriate manner
   - bit map
   - runlength encoding
   - quadtree
5. If need to symbolize, then can use line thickening
   - change to black any pixel that borders one of the line pixels
   - in order to avoid “jaggies”, use anti-aliasing which assigns lower intensities to neighboring pixels

BRESENHAM’S ALGORITHM

1. Skeletonization or line thinning
   - necessary because vector lines have zero width from a theoretical standpoint, yet are one pixel wide in raster mode
   - line consists merely of connected pixels (8-connected)
2. Line extraction
   - determine where the lines start and end in the thinned image
3. Topological reconstruction
   - generate topological connectivity of lines to build a topological definition of the lines and polygons

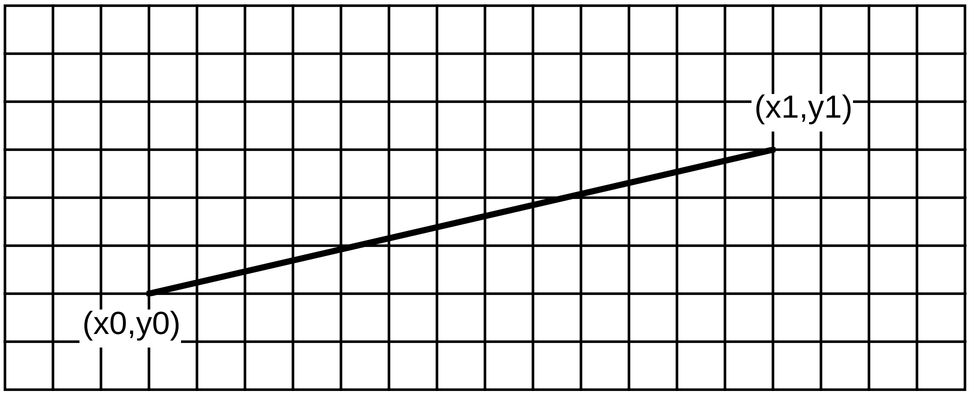
Ex:



- start plotting at (x_0,y_0)
- each time the model line moves to a new row (v keeps track of it), then start plotting there
- as column is incremented by one, the row is incremented by the fractional slope of the line, $(y_1–y_0)/(x_1–x_0)$

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0

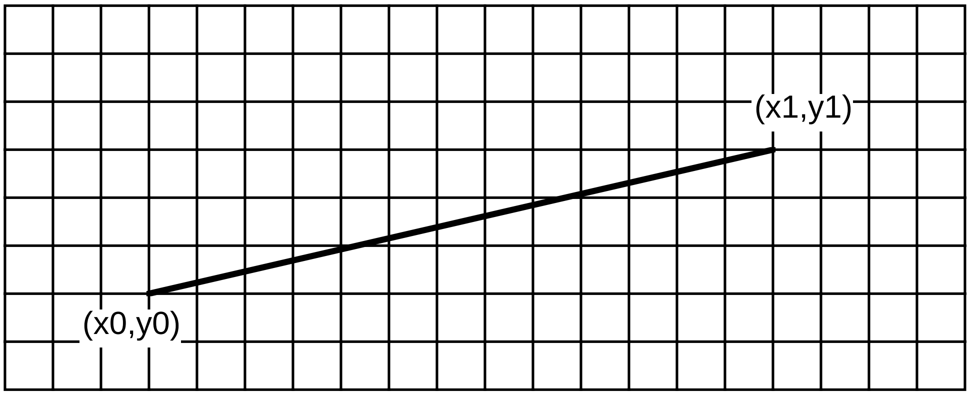


- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

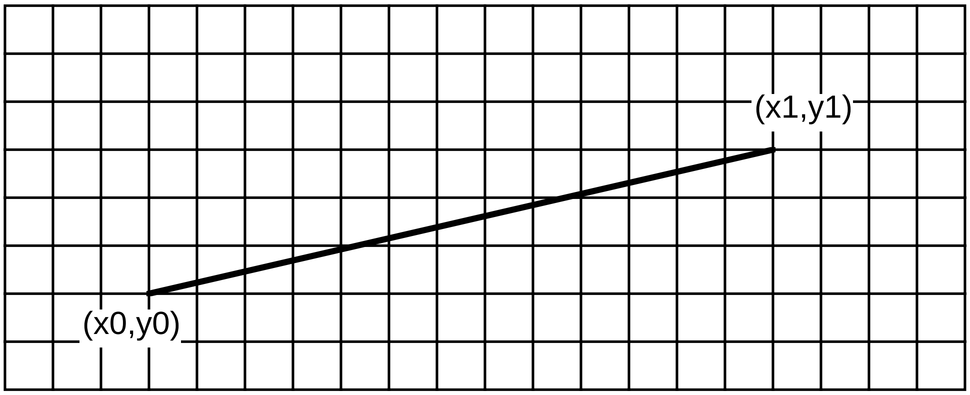
```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

- prefer integer calculations
- rewrite as:

```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

- prefer integer calculations
- rewrite as:

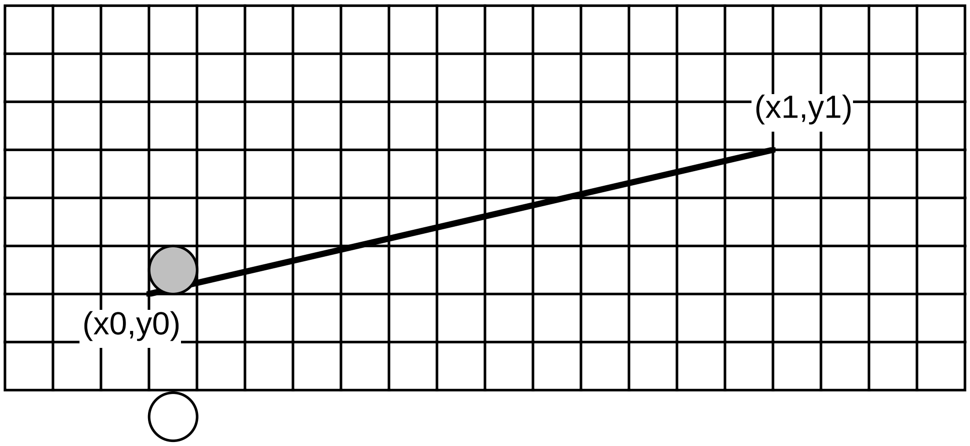
```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

- Substituting V for (Bv)

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
V ← 0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   V ← V + A;
   if V ≥ B then
   begin
      y ← y + 1;
      V ← V – B;
   end;
end;
```

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

- prefer integer calculations
- rewrite as:

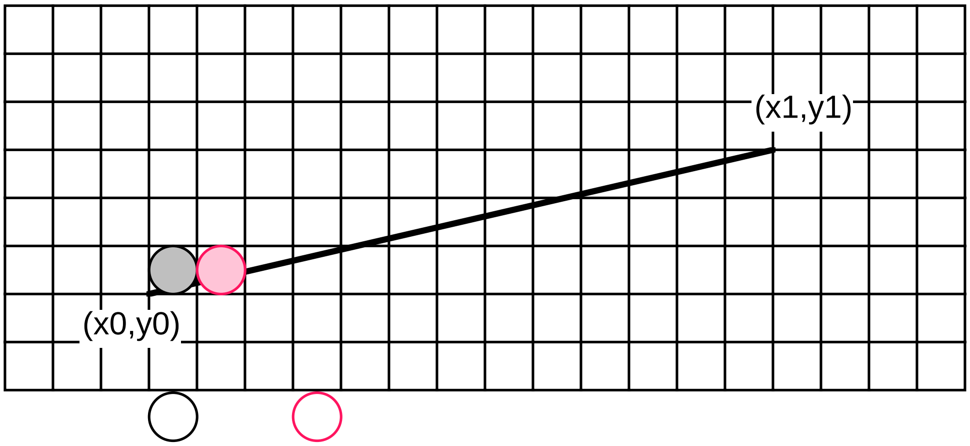
```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

- Substituting V for (Bv)

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
V ← 0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   V ← V + A;
   if V ≥ B then
   begin
      y ← y + 1;
      V ← V – B;
   end;
end;
```

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When $v \geq B$, it is reset to 0



- Ex: A=3, B=13

```
$A \leftarrow y1 - y0$;
$B \leftarrow x1 - x0$;
$x \leftarrow x0$;
$y \leftarrow y0$;
$v \leftarrow 0.0$;
while $x < x1$ do
begin
   plot(x, y);
   $x \leftarrow x + 1$;
   $v \leftarrow v + A/B$;
   if $v \geq 1.0$ then
   begin
      $y \leftarrow y + 1$;
      $v \leftarrow v - 1.0$;
   end;
end;
```

- prefer integer calculations
- rewrite as:

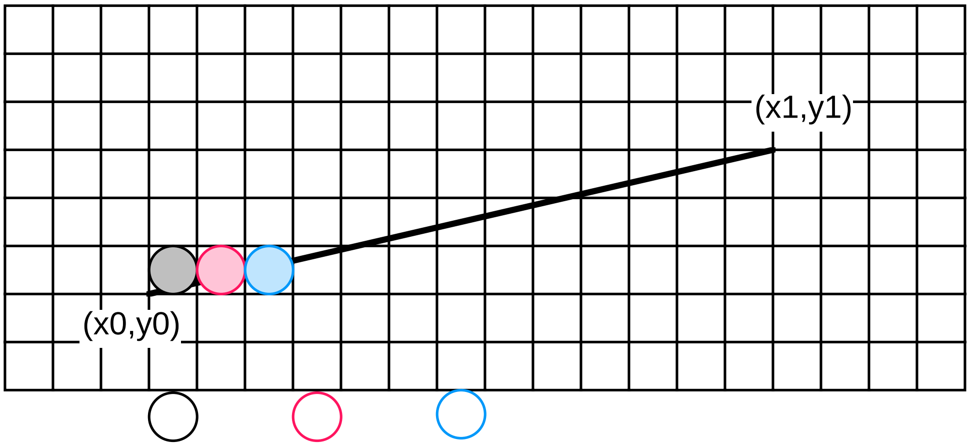
```
$(Bv) \leftarrow (Bv) + A$

if $(Bv) \geq B$ then

  $(Bv) \leftarrow (Bv) - B$;
```

- Substituting V for (Bv)

```
$A \leftarrow y1 - y0$;
$B \leftarrow x1 - x0$;
$x \leftarrow x0$;
$y \leftarrow y0$;
$V \leftarrow 0$;
while $x < x1$ do
begin
   plot(x, y);
   $x \leftarrow x + 1$;
   $V \leftarrow V + A$;
   if $V \geq B$ then
   begin
      $y \leftarrow y + 1$;
      $V \leftarrow V - B$;
   end;
end;
```

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
    plot(x, y);
    x ← x + 1;
    v ← v + A/B;
    if v ≥ 1.0 then
    begin
        y ← y + 1;
        v ← v – 1.0;
    end;
end;
```

- prefer integer calculations
- rewrite as:

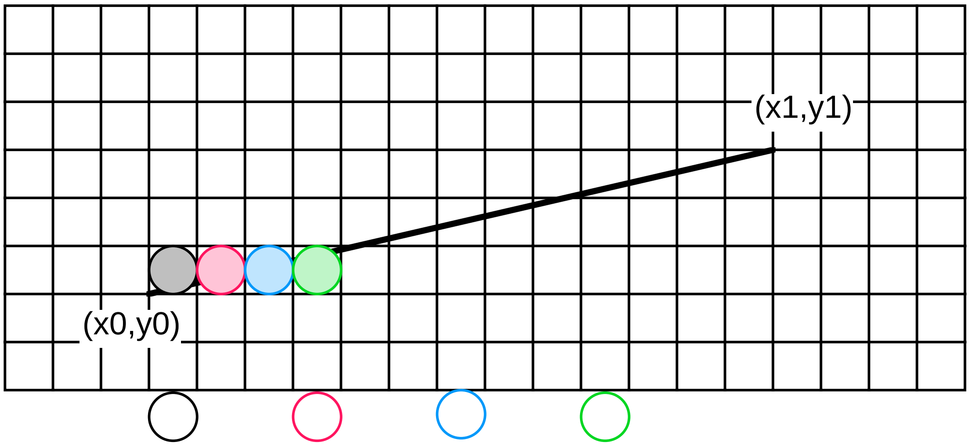
```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

- Substituting V for (Bv)

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
V ← 0;
while x < x1 do
begin
    plot(x, y);
    x ← x + 1;
    V ← V + A;
    if V ≥ B then
    begin
        y ← y + 1;
        V ← V – B;
    end;
end;
```

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

- prefer integer calculations
- rewrite as:

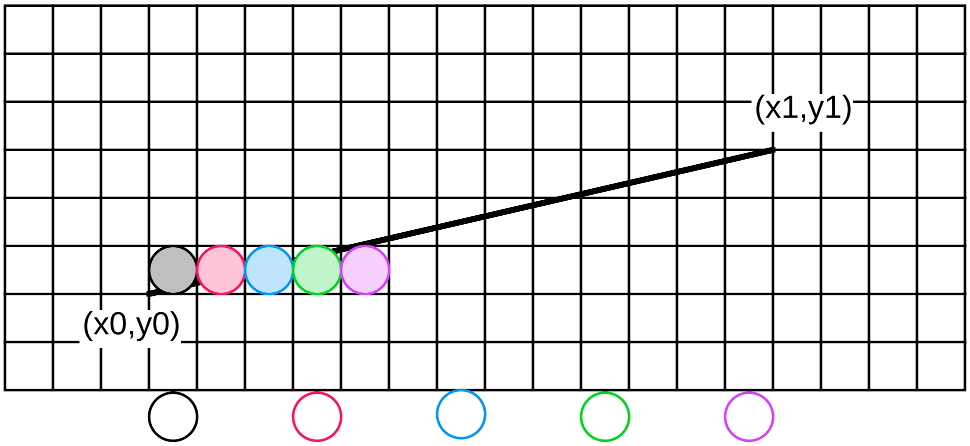
```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

- Substituting V for (Bv)

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
V ← 0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   V ← V + A;
   if V ≥ B then
   begin
      y ← y + 1;
      V ← V – B;
   end;
end;
```

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

- prefer integer calculations
- rewrite as:

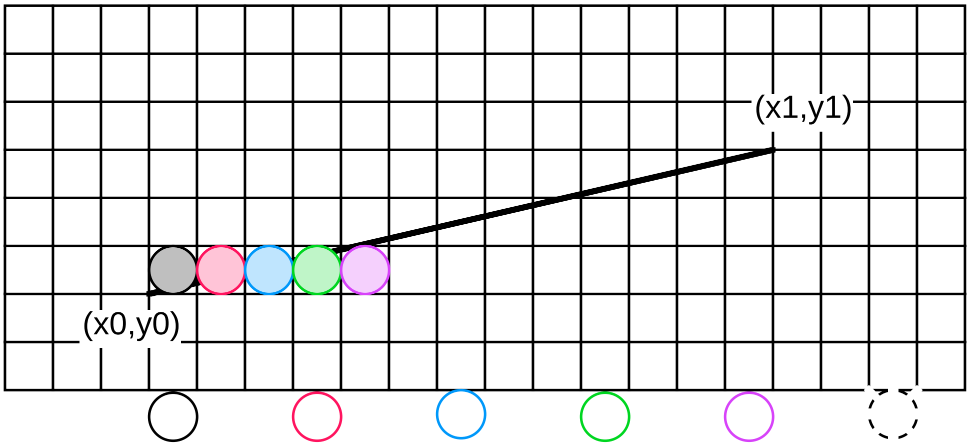
```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

- Substituting V for (Bv)

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
V ← 0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   V ← V + A;
   if V ≥ B then
   begin
      y ← y + 1;
      V ← V – B;
   end;
end;
```

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

- prefer integer calculations
- rewrite as:

```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

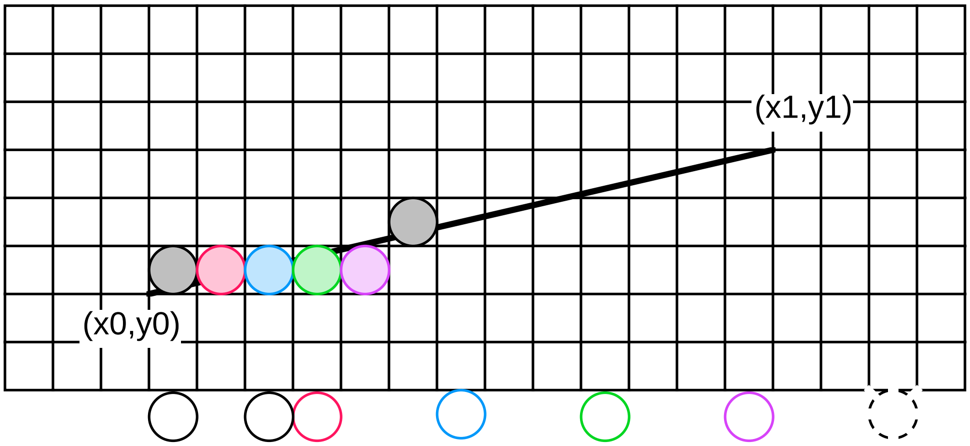
- Substituting V for (Bv)

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
V ← 0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   V ← V + A;
   if V ≥ B then
   begin
      y ← y + 1;
      V ← V – B;
   end;
end;
```

10	9	8	7	6	5	4	3	2	1
b	b	v	g	z	r	b	z	r	b

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

- prefer integer calculations
- rewrite as:

```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

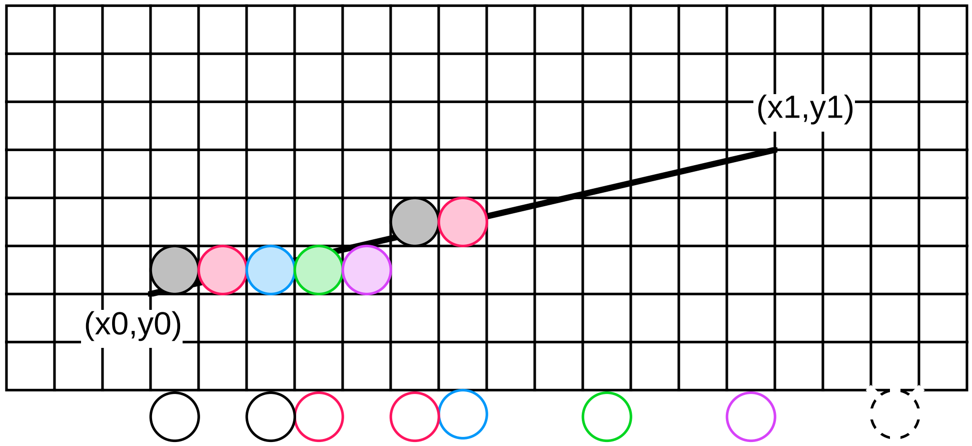
- Substituting V for (Bv)

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
V ← 0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   V ← V + A;
   if V ≥ B then
   begin
      y ← y + 1;
      V ← V – B;
   end;
end;
```

11	10	9	8	7	6	5	4	3	2	1
r	b	b	v	g	z	r	b	z	r	b

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

- prefer integer calculations
- rewrite as:

```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

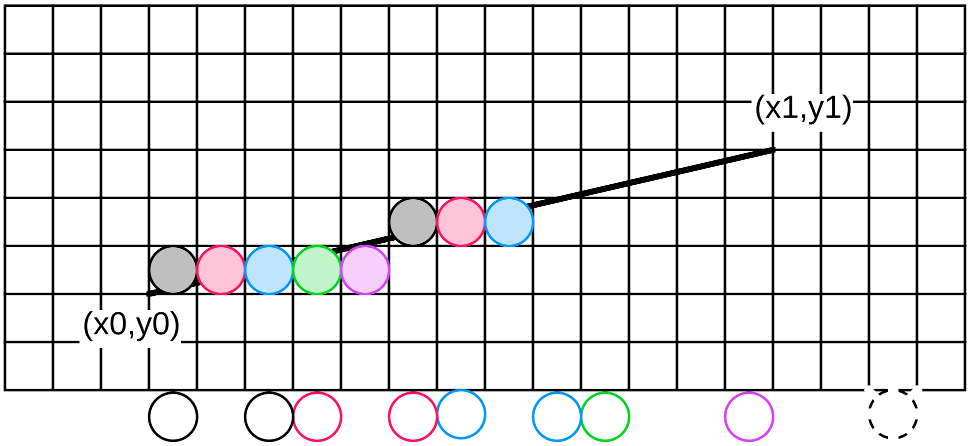
- Substituting V for (Bv)

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
V ← 0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   V ← V + A;
   if V ≥ B then
   begin
      y ← y + 1;
      V ← V – B;
   end;
end;
```

12	11	10	9	8	7	6	5	4	3	2	1
z	r	b	b	v	g	z	r	b	z	r	b

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

- prefer integer calculations
- rewrite as:

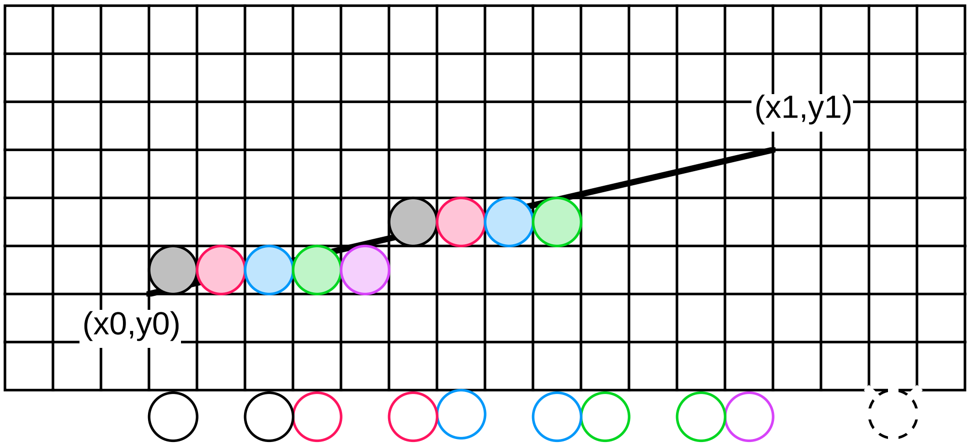
```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

- Substituting V for (Bv)

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
V ← 0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   V ← V + A;
   if V ≥ B then
   begin
      y ← y + 1;
      V ← V – B;
   end;
end;
```

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

- prefer integer calculations
- rewrite as:

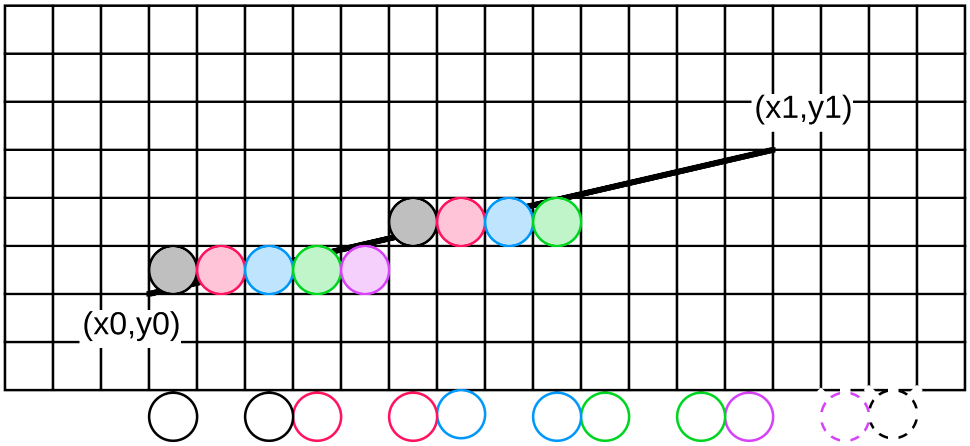
```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

- Substituting V for (Bv)

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
V ← 0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   V ← V + A;
   if V ≥ B then
   begin
      y ← y + 1;
      V ← V – B;
   end;
end;
```

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

- prefer integer calculations
- rewrite as:

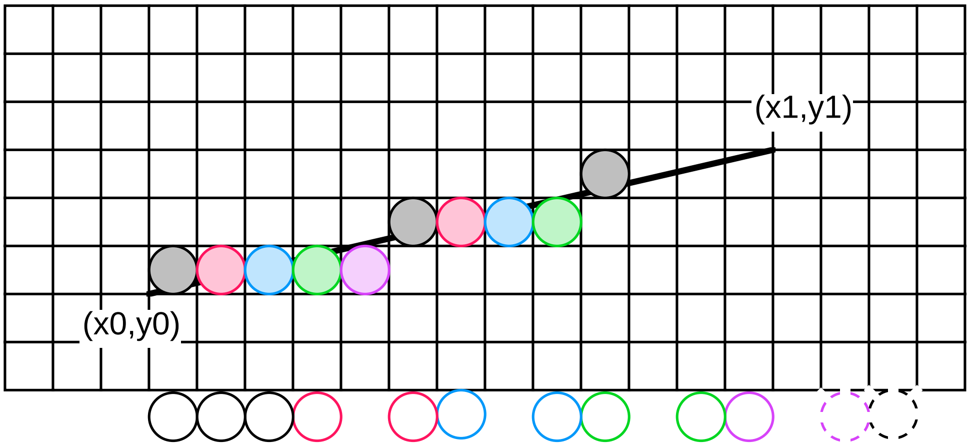
```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

- Substituting V for (Bv)

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
V ← 0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   V ← V + A;
   if V ≥ B then
   begin
      y ← y + 1;
      V ← V – B;
   end;
end;
```

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

- prefer integer calculations
- rewrite as:

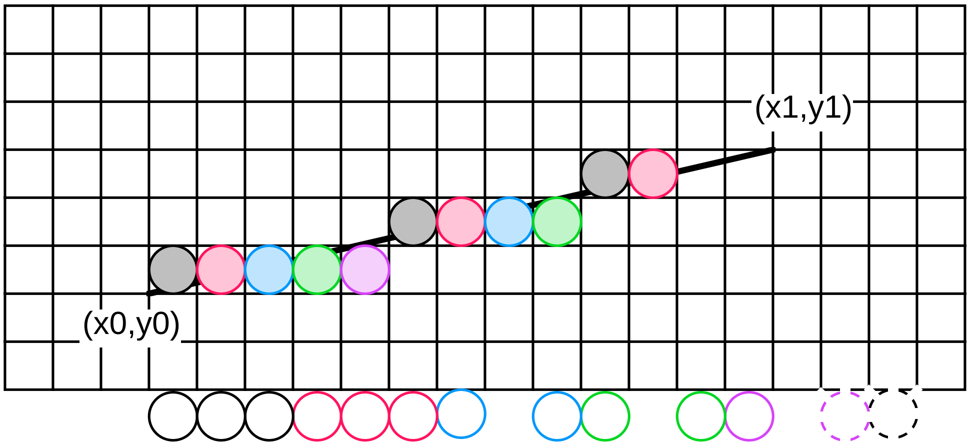
```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

- Substituting V for (Bv)

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
V ← 0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   V ← V + A;
   if V ≥ B then
   begin
      y ← y + 1;
      V ← V – B;
   end;
end;
```

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

- prefer integer calculations
- rewrite as:

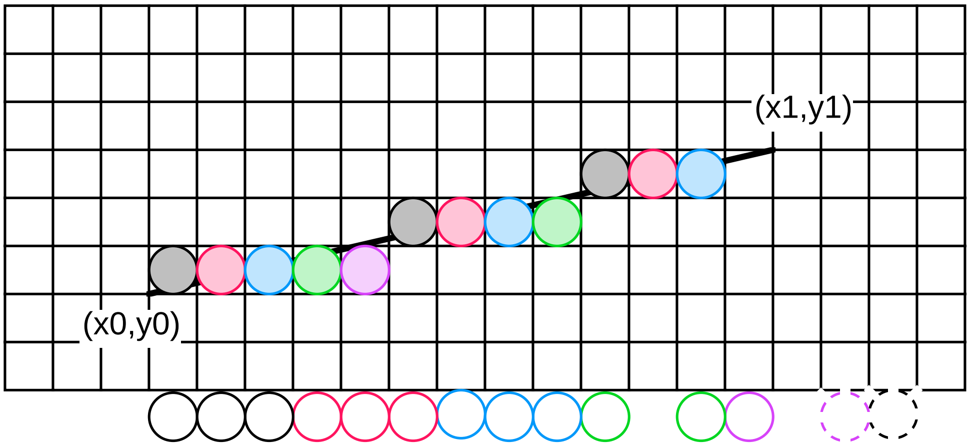
```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

- Substituting V for (Bv)

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
V ← 0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   V ← V + A;
   if V ≥ B then
   begin
      y ← y + 1;
      V ← V – B;
   end;
end;
```

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

- prefer integer calculations
- rewrite as:

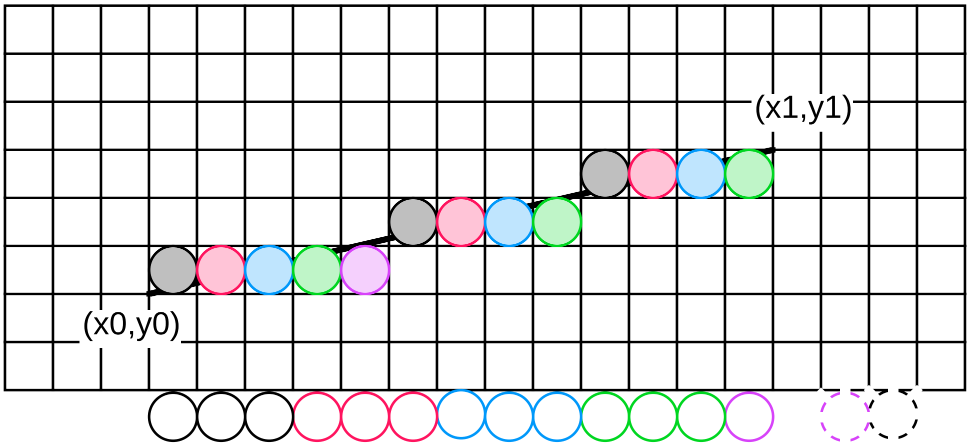
```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

- Substituting V for (Bv)

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
V ← 0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   V ← V + A;
   if V ≥ B then
   begin
      y ← y + 1;
      V ← V – B;
   end;
end;
```

MECHANICS OF BRESENHAM'S ALGORITHM

- Horizontal ball accumulates changes in v
- When v ≥ B, it is reset to 0



- Ex: A=3, B=13

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
v ← 0.0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   v ← v + A/B;
   if v ≥ 1.0 then
   begin
      y ← y + 1;
      v ← v – 1.0;
   end;
end;
```

- prefer integer calculations
- rewrite as:

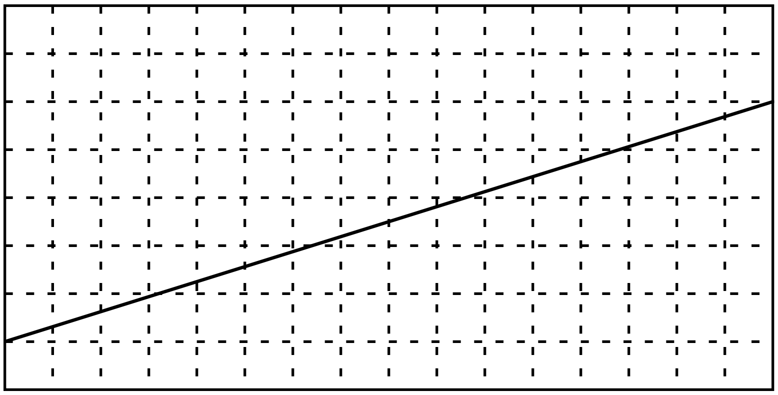
```
(Bv) ← (Bv) + A

if (Bv) ≥ B then

  (Bv) ← (Bv) – B;
```

- Substituting V for (Bv)

```
A ← y1 – y0;
B ← x1 – x0;
x ← x0;
y ← y0;
V ← 0;
while x < x1 do
begin
   plot(x, y);
   x ← x + 1;
   V ← V + A;
   if V ≥ B then
   begin
      y ← y + 1;
      V ← V – B;
   end;
end;
```

PROPERTY OF BRESENHAM’S ALGORITHM

- Does not mark all cells intersected by a line
- Ex:

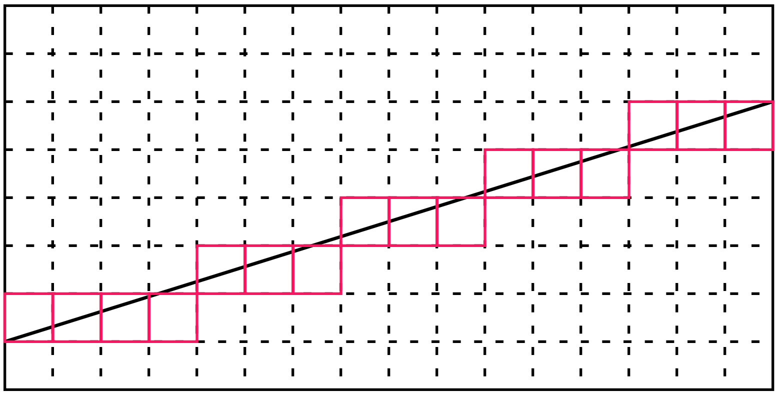


- Rationale:

  1. algorithm is discrete

PROPERTY OF BRESENHAM'S ALGORITHM

- Does not mark all cells intersected by a line
- Ex:

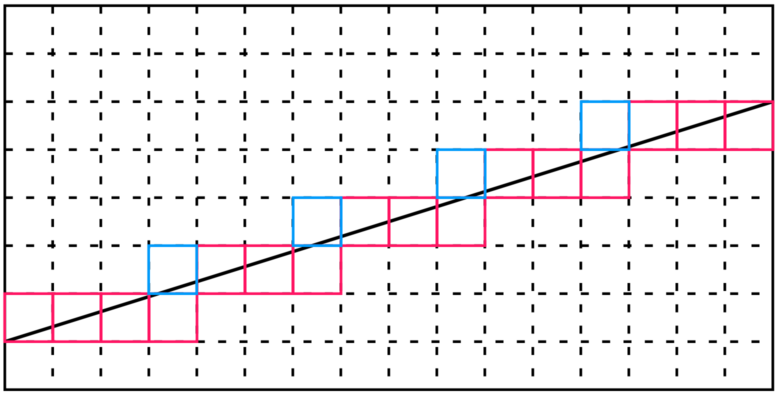


- Rationale:

1. algorithm is discrete
2. checks whether the line passes through a cell when it first encounters the extreme left boundary of the cell

PROPERTY OF BRESENHAM'S ALGORITHM

- Does not mark all cells intersected by a line
- Ex:

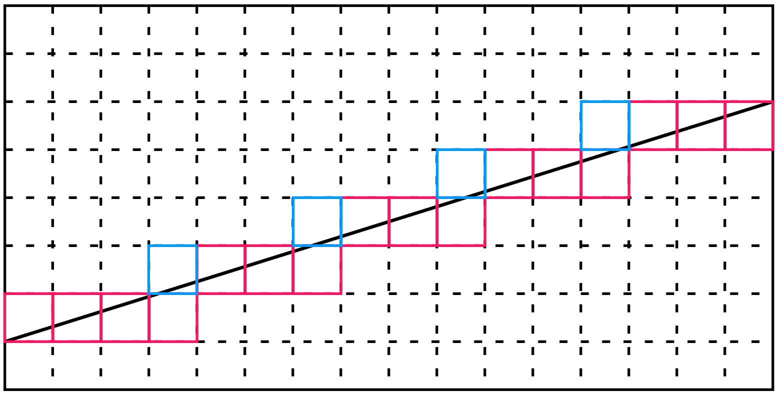


- Rationale:

  1. algorithm is discrete
  2. checks whether the line passes through a cell when it first encounters the extreme left boundary of the cell
  3. misses cells that are crossed after the line has passed through the left boundary of a cell at (*x*,*y*) but before passing through the vertical line at its right boundary

PROPERTY OF BRESENHAM’S ALGORITHM

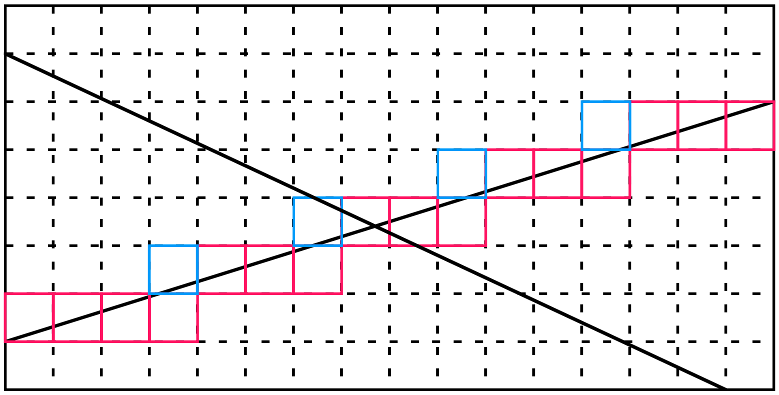
- Does not mark all cells intersected by a line
- Ex:



- Rationale:
  1. algorithm is discrete
  2. checks whether the line passes through a cell when it first encounters the extreme left boundary of the cell
  3. misses cells that are crossed after the line has passed through the left boundary of a cell at (*x*,*y*) but before passing through the vertical line at its right boundary
     - if we were to count these cells as part of the line, then two intersecting lines may be reported as intersecting in more cells than before

PROPERTY OF BRESENHAM'S ALGORITHM

- Does not mark all cells intersected by a line
- Ex:

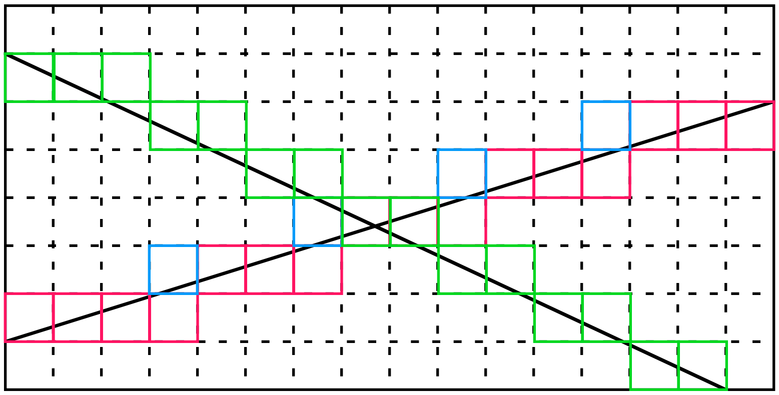


- Rationale:

1. algorithm is discrete
2. checks whether the line passes through a cell when it first encounters the extreme left boundary of the cell
3. misses cells that are crossed after the line has passed through the left boundary of a cell at (x,y) but before passing through the vertical line at its right boundary
   - if we were to count these cells as part of the line, then two intersecting lines may be reported as intersecting in more cells than before
   - Ex: add an intersecting line

PROPERTY OF BRESENHAM’S ALGORITHM

- Does not mark all cells intersected by a line
- Ex:



- Rationale:

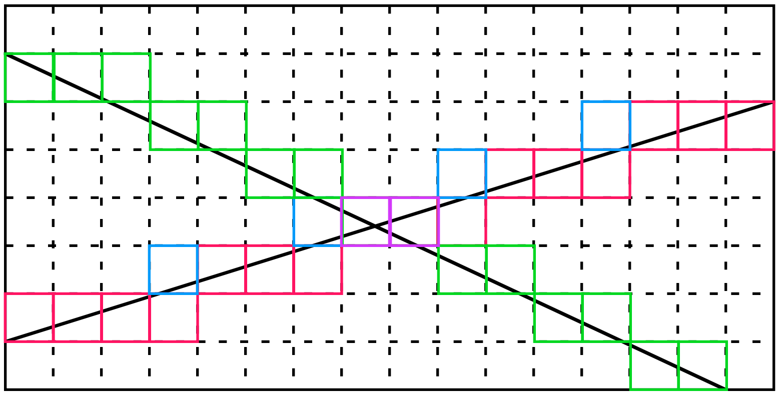
  1. algorithm is discrete
  2. checks whether the line passes through a cell when it first encounters the extreme left boundary of the cell
  3. misses cells that are crossed after the line has passed through the left boundary of a cell at (x,y) but before passing through the vertical line at its right boundary

     - if we were to count these cells as part of the line, then two intersecting lines may be reported as intersecting in more cells than before
     - Ex: add an intersecting line

       1. result of Bresenham digitization

PROPERTY OF BRESENHAM’S ALGORITHM

- Does not mark all cells intersected by a line
- Ex:



- Rationale:

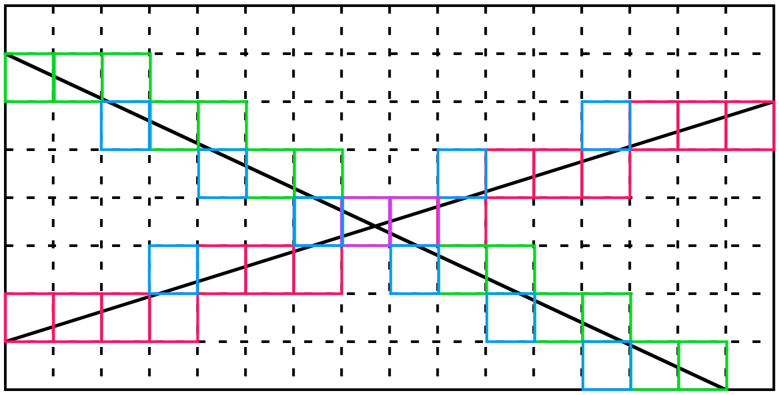
  1. algorithm is discrete
  2. checks whether the line passes through a cell when it first encounters the extreme left boundary of the cell
  3. misses cells that are crossed after the line has passed through the left boundary of a cell at (*x*,*y*) but before passing through the vertical line at its right boundary

     - if we were to count these cells as part of the line, then two intersecting lines may be reported as intersecting in more cells than before
     - Ex: add an intersecting line

       1. result of Bresenham digitization
          - digitized representations intersect at two cells

PROPERTY OF BRESENHAM'S ALGORITHM

- Does not mark all cells intersected by a line
- Ex:



- Rationale:

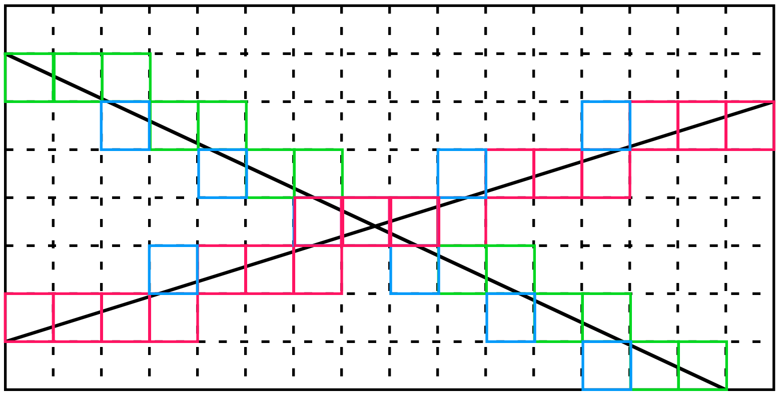
  1. algorithm is discrete
  2. checks whether the line passes through a cell when it first encounters the extreme left boundary of the cell
  3. misses cells that are crossed after the line has passed through the left boundary of a cell at (x,y) but before passing through the vertical line at its right boundary

     - if we were to count these cells as part of the line, then two intersecting lines may be reported as intersecting in more cells than before
     - Ex: add an intersecting line

       1. result of Bresenham digitization
          - digitized representations intersect at two cells
       2. result of marking all cells intersected by the line

PROPERTY OF BRESENHAM'S ALGORITHM

- Does not mark all cells intersected by a line
- Ex:



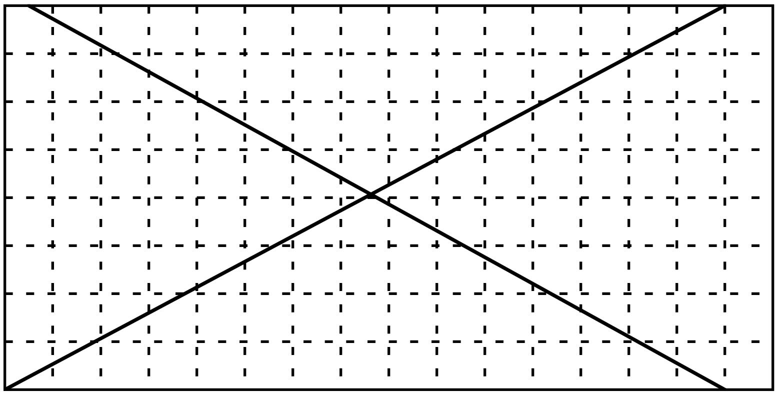
- Rationale:
  1. algorithm is discrete
  2. checks whether the line passes through a cell when it first encounters the extreme left boundary of the cell
  3. misses cells that are crossed after the line has passed through the left boundary of a cell at (x,y) but before passing through the vertical line at its right boundary
     - if we were to count these cells as part of the line, then two intersecting lines may be reported as intersecting in more cells than before
     - Ex: add an intersecting line
       1. result of Bresenham digitization
          - digitized representations intersect at two cells
       2. result of marking all cells intersected by the line
          - digitized representations intersect at three cells, instead of two

ANOMALIES OF BRESENHAM’S ALGORITHM AND OTHER DIGITIZATION METHODS

- Intersecting lines may fail to have a cell in common, thereby causing the intersection to be missed

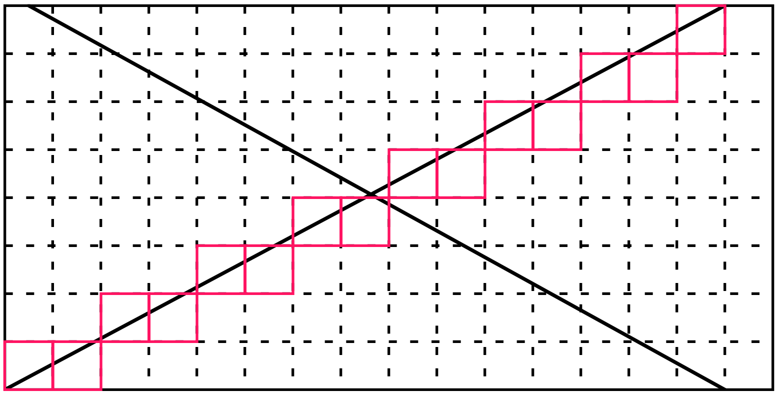
- With Bresenham’s algorithm, this happens relatively frequently, as cells of a line are not four-connected when the line crosses top or bottom edge of a cell

- Ex:



ANOMALIES OF BRESENHAM’S ALGORITHM AND OTHER DIGITIZATION METHODS

- Intersecting lines may fail to have a cell in common, thereby causing the intersection to be missed
- With Bresenham’s algorithm, this happens relatively frequently, as cells of a line are not four-connected when the line crosses top or bottom edge of a cell
- Ex:

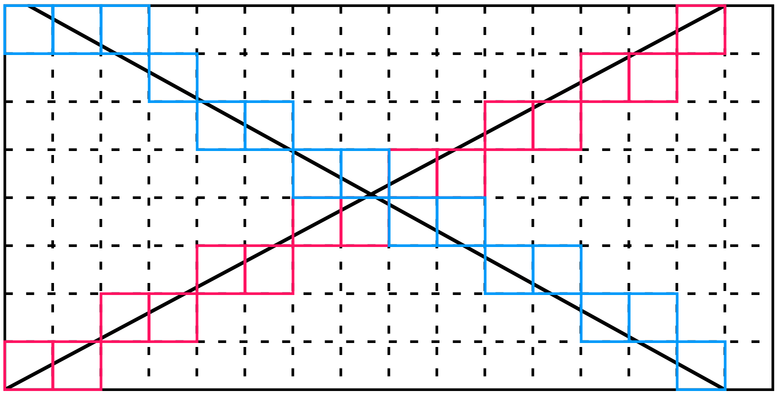


Bresenham digitization

Bresenham digitization

ANOMALIES OF BRESENHAM’S ALGORITHM AND OTHER DIGITIZATION METHODS

- Intersecting lines may fail to have a cell in common, thereby causing the intersection to be missed
- With Bresenham’s algorithm, this happens relatively frequently, as cells of a line are not four-connected when the line crosses top or bottom edge of a cell
- Ex:



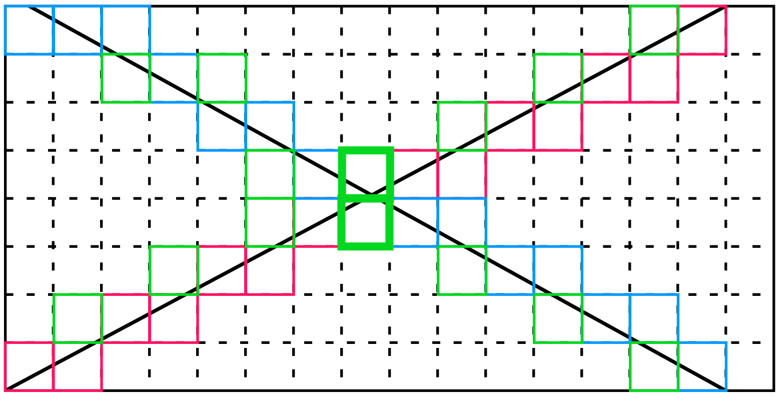
Bresenham digitization

Bresenham digitization

Bresenham digitization

ANOMALIES OF BRESENHAM’S ALGORITHM AND OTHER DIGITIZATION METHODS

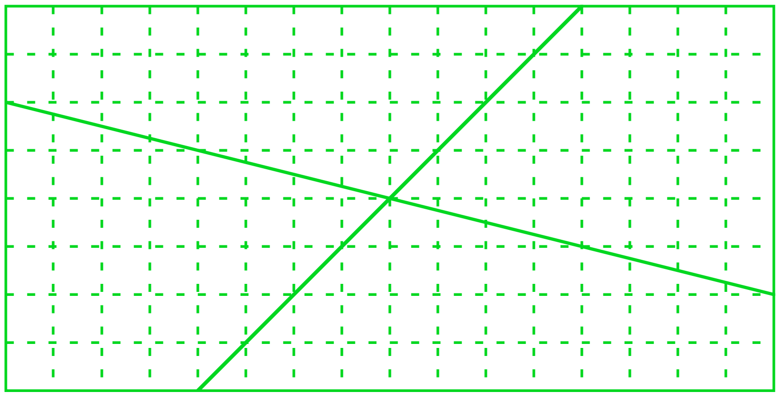
- Intersecting lines may fail to have a cell in common, thereby causing the intersection to be missed
- With Bresenham’s algorithm, this happens relatively frequently, as cells of a line are not four-connected when the line crosses top or bottom edge of a cell
- Ex:



Bresenham digitization

Bresenham digitization

- In this example, two cell intersections are detected if all cells that are entered are marked. However, what happens when the lines cross at the corner of a cell?
- Ex:



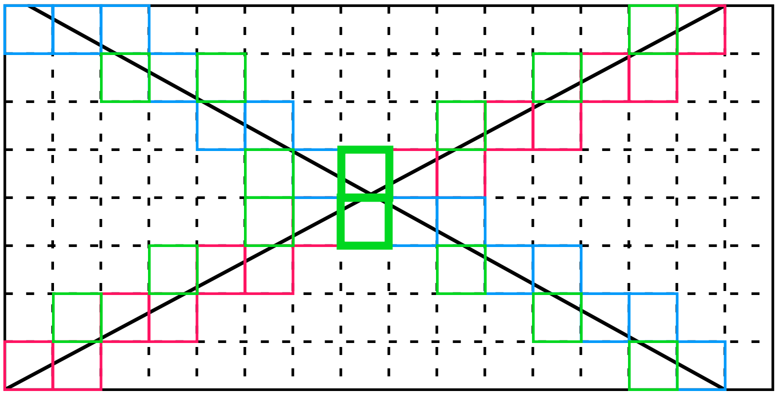
Bresenham digitization

ANOMALIES OF BRESENHAM'S ALGORITHM AND OTHER DIGITIZATION METHODS

- Intersecting lines may fail to have a cell in common, thereby causing the intersection to be missed

- With Bresenham's algorithm, this happens relatively frequently, as cells of a line are not four-connected when the line crosses top or bottom edge of a cell

- Ex:

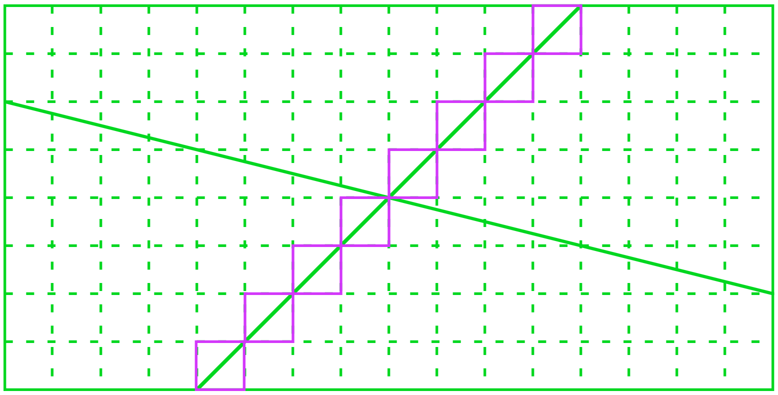


Bresenham digitization

Bresenham digitization

- In this example, two cell intersections are detected if all cells that are entered are marked. However, what happens when the lines cross at the corner of a cell?

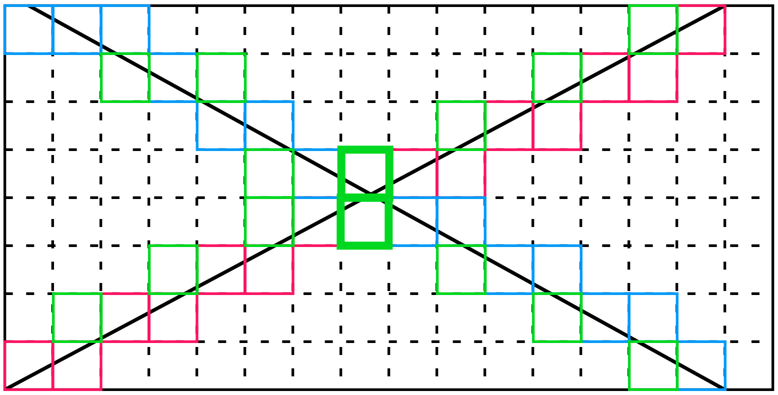
- Ex:



Bresenham digitization

ANOMALIES OF BRESENHAM'S ALGORITHM AND OTHER DIGITIZATION METHODS

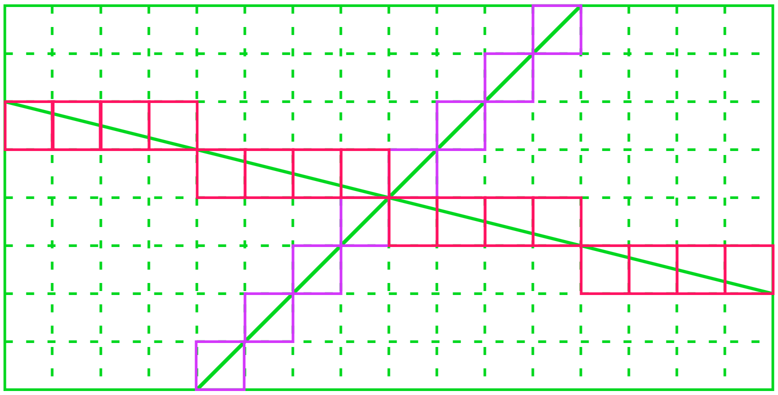
- Intersecting lines may fail to have a cell in common, thereby causing the intersection to be missed
- With Bresenham's algorithm, this happens relatively frequently, as cells of a line are not four-connected when the line crosses top or bottom edge of a cell
- Ex:



Bresenham digitization

Bresenham digitization

- In this example, two cell intersections are detected if all cells that are entered are marked. However, what happens when the lines cross at the corner of a cell?
- Ex:



Bresenham digitization

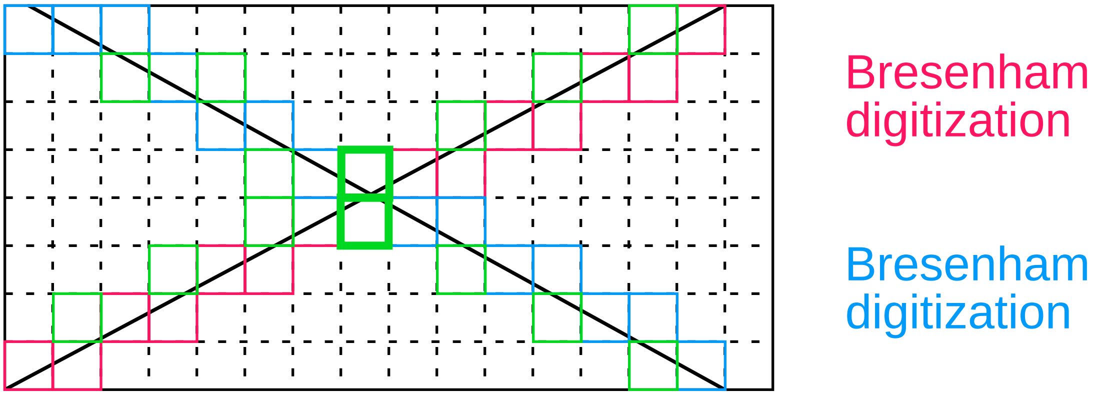
Bresenham digitization

ANOMALIES OF BRESENHAM'S ALGORITHM AND OTHER DIGITIZATION METHODS

- Intersecting lines may fail to have a cell in common, thereby causing the intersection to be missed

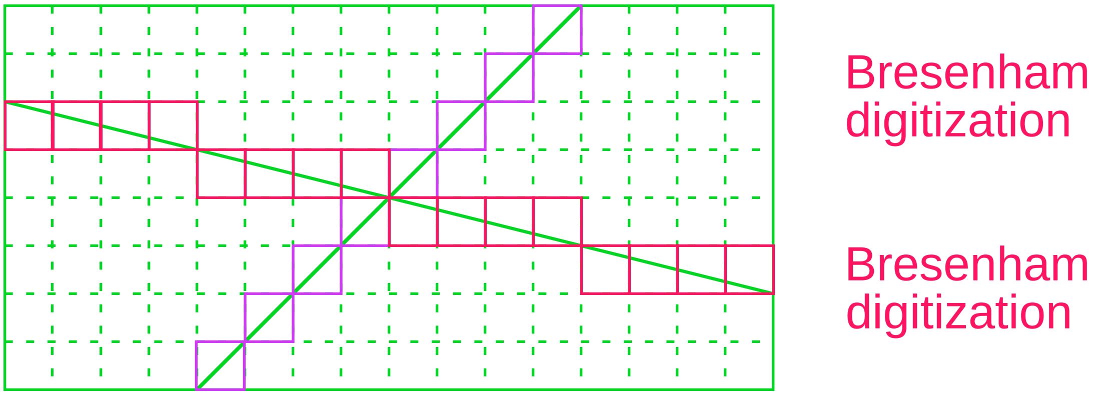
- With Bresenham's algorithm, this happens relatively frequently, as cells of a line are not four-connected when the line crosses top or bottom edge of a cell

- Ex:



- In this example, two cell intersections are detected if all cells that are entered are marked. However, what happens when the lines cross at the corner of a cell?

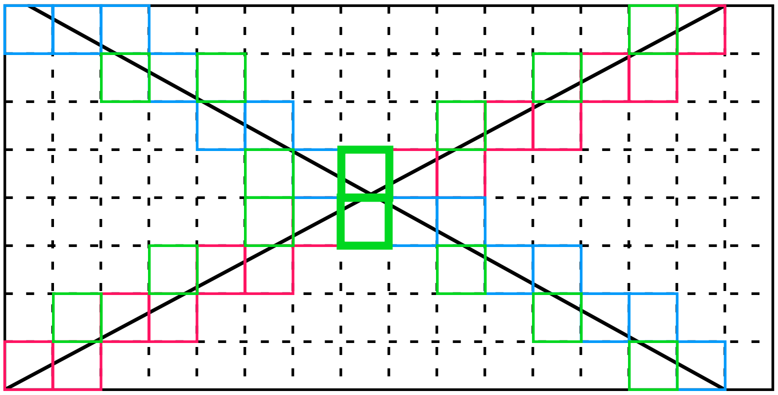
- Ex:



- No other cells are entered except at a corner

ANOMALIES OF BRESENHAM'S ALGORITHM AND OTHER DIGITIZATION METHODS

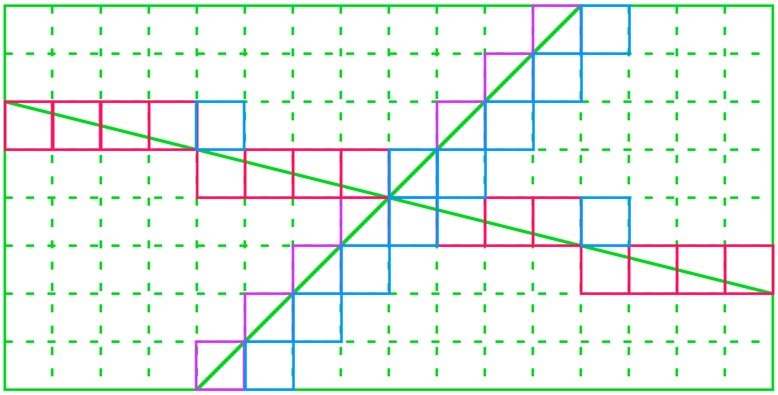
- Intersecting lines may fail to have a cell in common, thereby causing the intersection to be missed
- With Bresenham's algorithm, this happens relatively frequently, as cells of a line are not four-connected when the line crosses top or bottom edge of a cell
- Ex:



Bresenham digitization

Bresenham digitization

- In this example, two cell intersections are detected if all cells that are entered are marked. However, what happens when the lines cross at the corner of a cell?
- Ex:



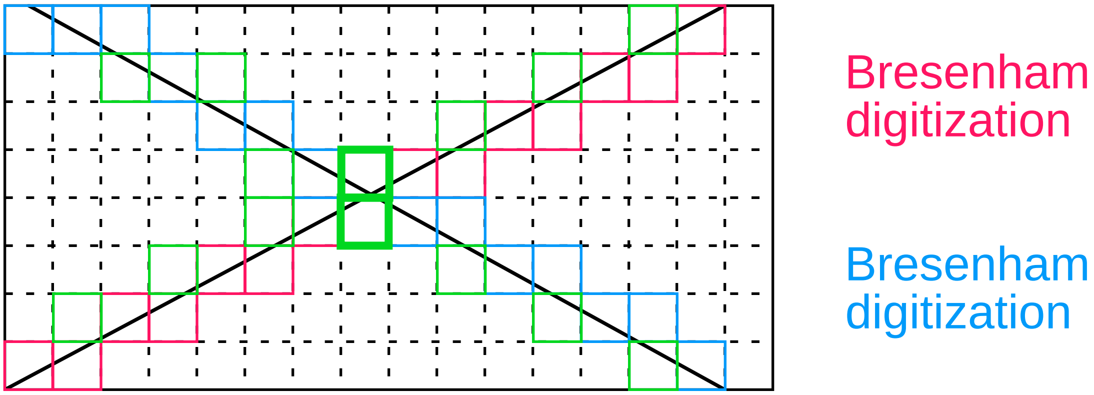
Bresenham digitization

Bresenham digitization

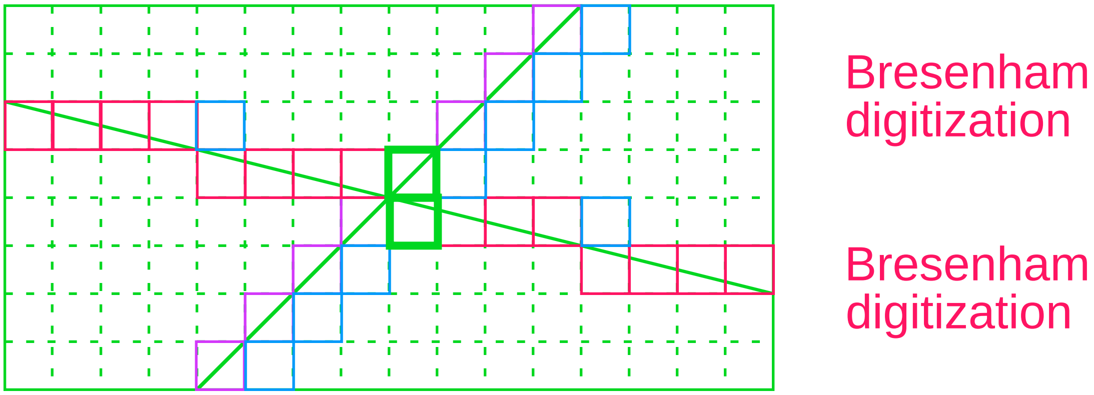
- No other cells are entered except at a corner
- Could adopt the convention that a cell is included if the line touches the left or bottom edge at a point

ANOMALIES OF BRESENHAM'S ALGORITHM AND OTHER DIGITIZATION METHODS

- Intersecting lines may fail to have a cell in common, thereby causing the intersection to be missed
- With Bresenham's algorithm, this happens relatively frequently, as cells of a line are not four-connected when the line crosses top or bottom edge of a cell
- Ex:



- In this example, two cell intersections are detected if all cells that are entered are marked. However, what happens when the lines cross at the corner of a cell?
- Ex:



- No other cells are entered except at a corner
- Could adopt the convention that a cell is included if the line touches the left or bottom edge at a point

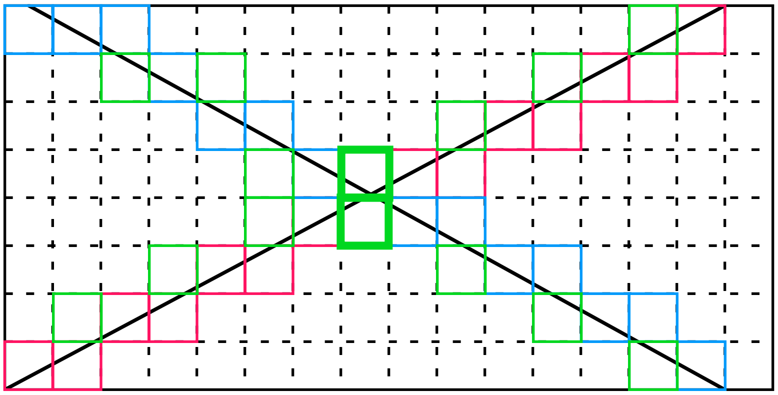
  1. two cell intersections

ANOMALIES OF BRESENHAM’S ALGORITHM AND OTHER DIGITIZATION METHODS

- Intersecting lines may fail to have a cell in common, thereby causing the intersection to be missed

- With Bresenham’s algorithm, this happens relatively frequently, as cells of a line are not four-connected when the line crosses top or bottom edge of a cell

- Ex:

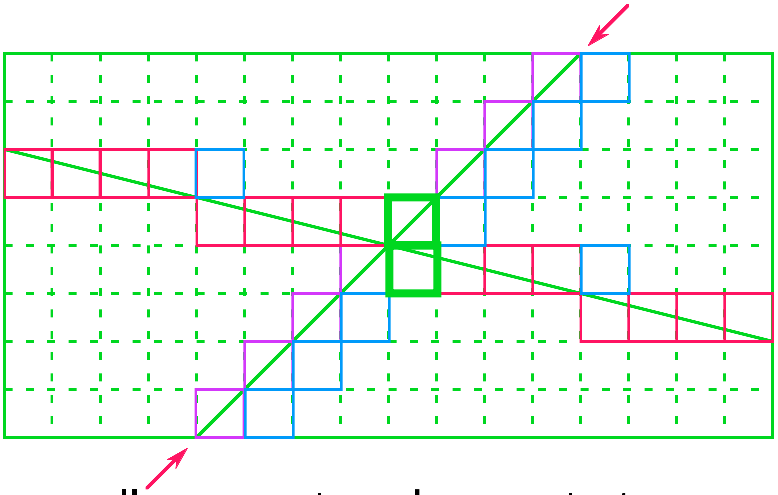


Bresenham digitization

Bresenham digitization

- In this example, two cell intersections are detected if all cells that are entered are marked. However, what happens when the lines cross at the corner of a cell?

- Ex:



Bresenham digitization

Bresenham digitization

- No other cells are entered except at a corner

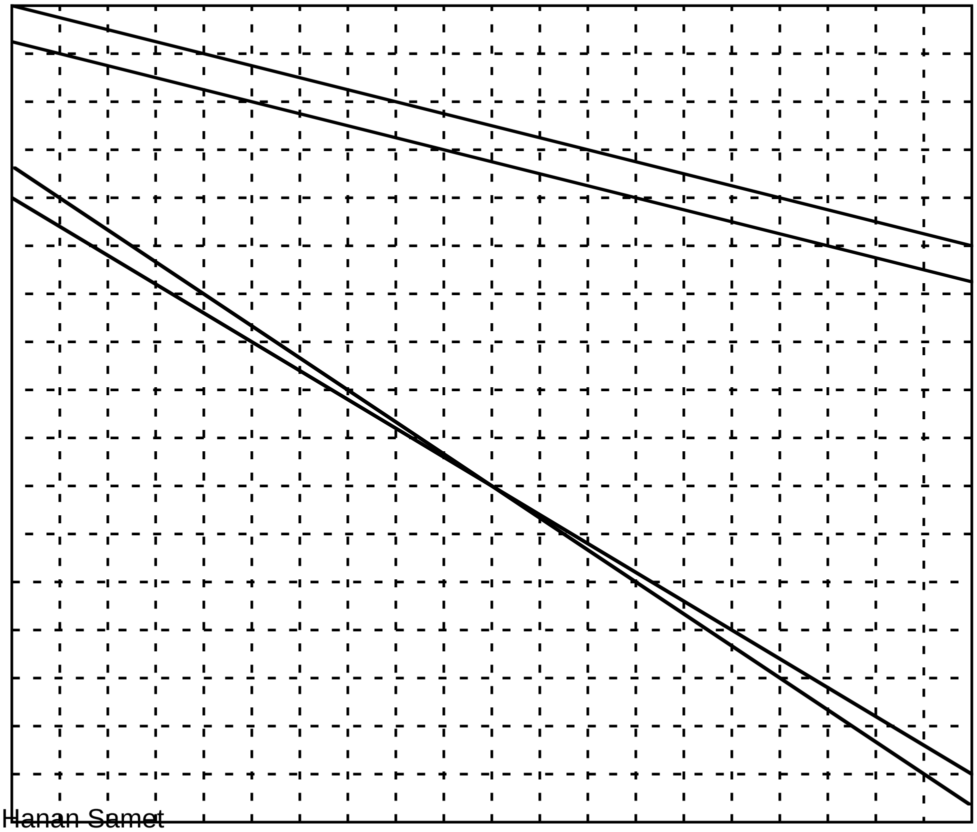
- Could adopt the convention that a cell is included if the line touches the left or bottom edge at a point

  1. two cell intersections

  2. this causes some lines to appear thicker than others!

LINE SEGMENT INTERSECTION

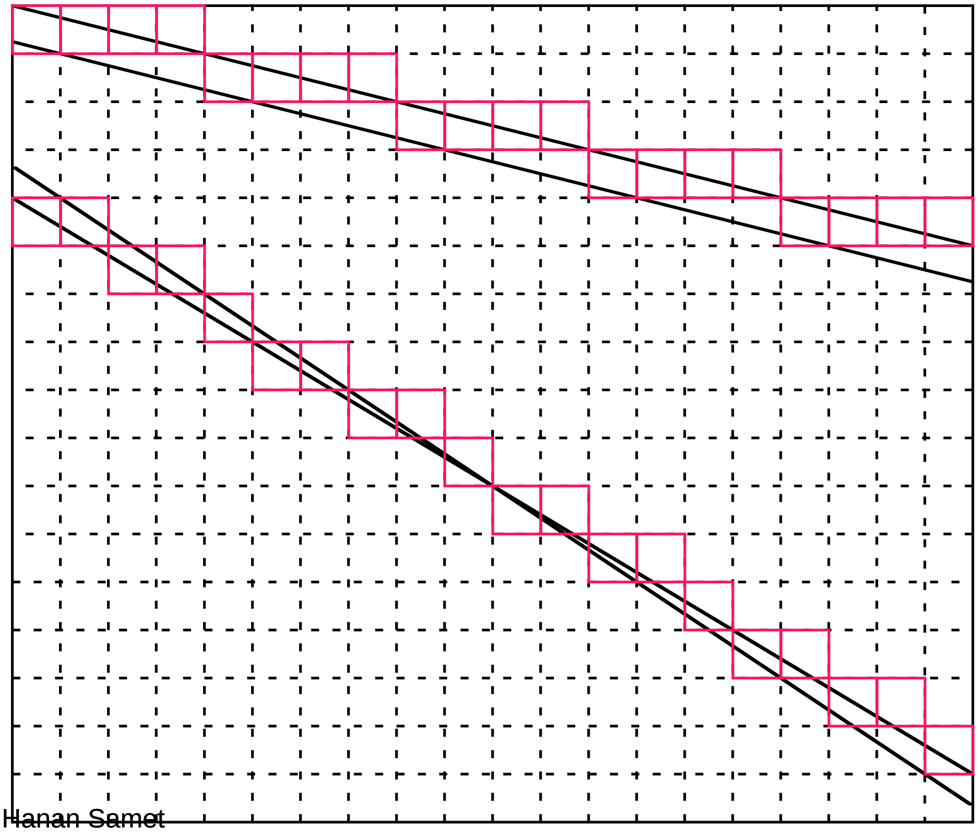
- Simple solution is to solve a pair of simultaneous equations
- Need special cases
  1. intersection is computed by using the infinite extension of the line segments and the point of intersection may possibly lie on a point which is not on the line segments that are being considered
  2. vertical lines have infinite slopes
  3. parallel lines
- May not have enough precision to calculate intersection
- Digital representation may cause multiple intersections as in the braiding of lines
- Ex: 1. parallel lines
  2. intersecting lines



LINE SEGMENT INTERSECTION

- Simple solution is to solve a pair of simultaneous equations
- Need special cases
  1. intersection is computed by using the infinite extension of the line segments and the point of intersection may possibly lie on a point which is not on the line segments that are being considered
  2. vertical lines have infinite slopes
  3. parallel lines
- May not have enough precision to calculate intersection
- Digital representation may cause multiple intersections as in the braiding of lines
- Ex: 1. parallel lines

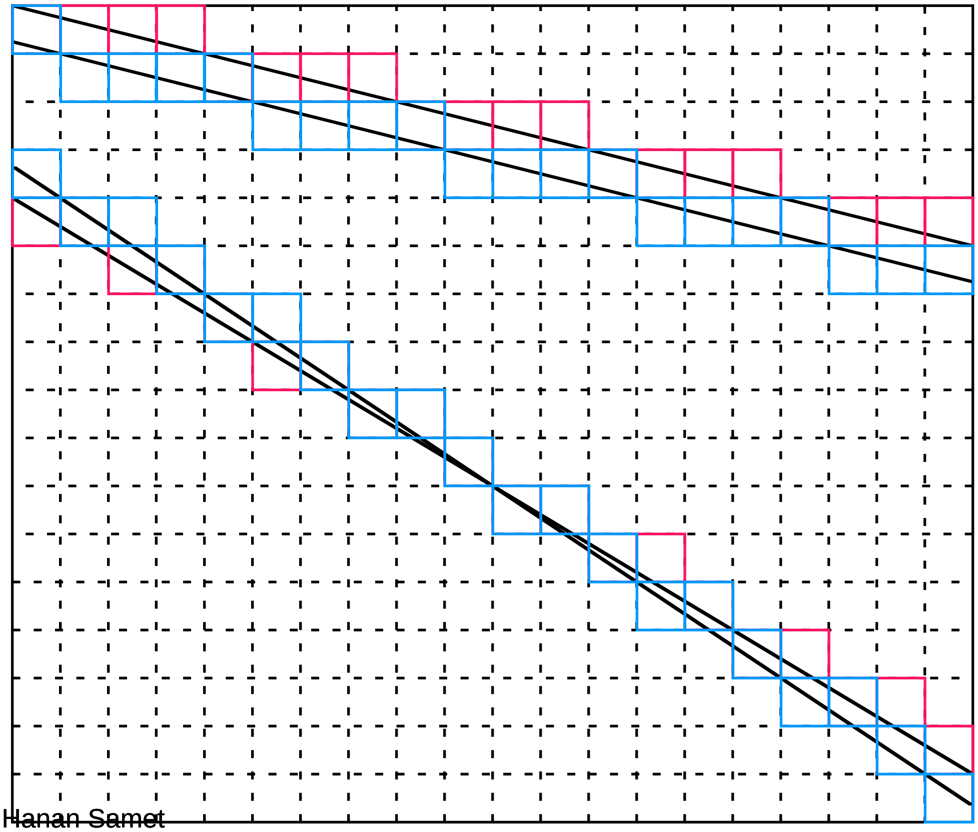
  2. intersecting lines



LINE SEGMENT INTERSECTION

- Simple solution is to solve a pair of simultaneous equations
- Need special cases
  1. intersection is computed by using the infinite extension of the line segments and the point of intersection may possibly lie on a point which is not on the line segments that are being considered
  2. vertical lines have infinite slopes
  3. parallel lines
- May not have enough precision to calculate intersection
- Digital representation may cause multiple intersections as in the braiding of lines
- Ex: 1. parallel lines

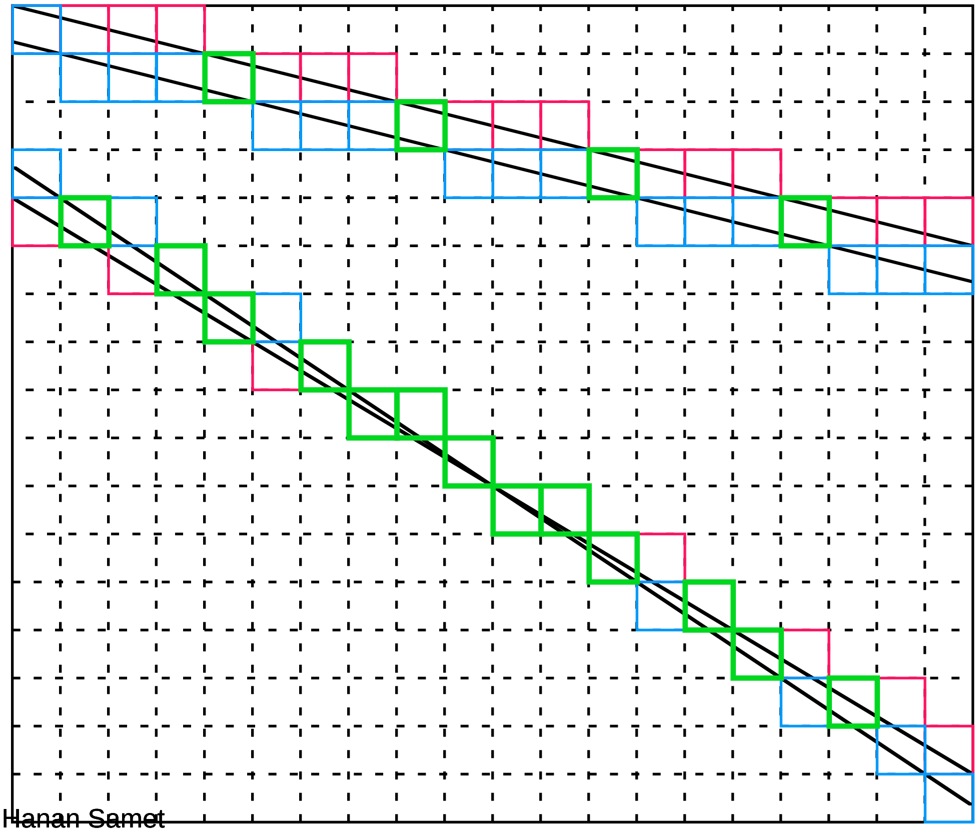
  2. intersecting lines



LINE SEGMENT INTERSECTION

- Simple solution is to solve a pair of simultaneous equations
- Need special cases
  1. intersection is computed by using the infinite extension of the line segments and the point of intersection may possibly lie on a point which is not on the line segments that are being considered
  2. vertical lines have infinite slopes
  3. parallel lines
- May not have enough precision to calculate intersection
- Digital representation may cause multiple intersections as in the braiding of lines
- Ex: 1. parallel lines

  2. intersecting lines



LINE SEGMENT INTERSECTION

- Simple solution is to solve a pair of simultaneous equations

- Need special cases

  1. intersection is computed by using the infinite extension of the line segments and the point of intersection may possibly lie on a point which is not on the line segments that are being considered

  2. vertical lines have infinite slopes

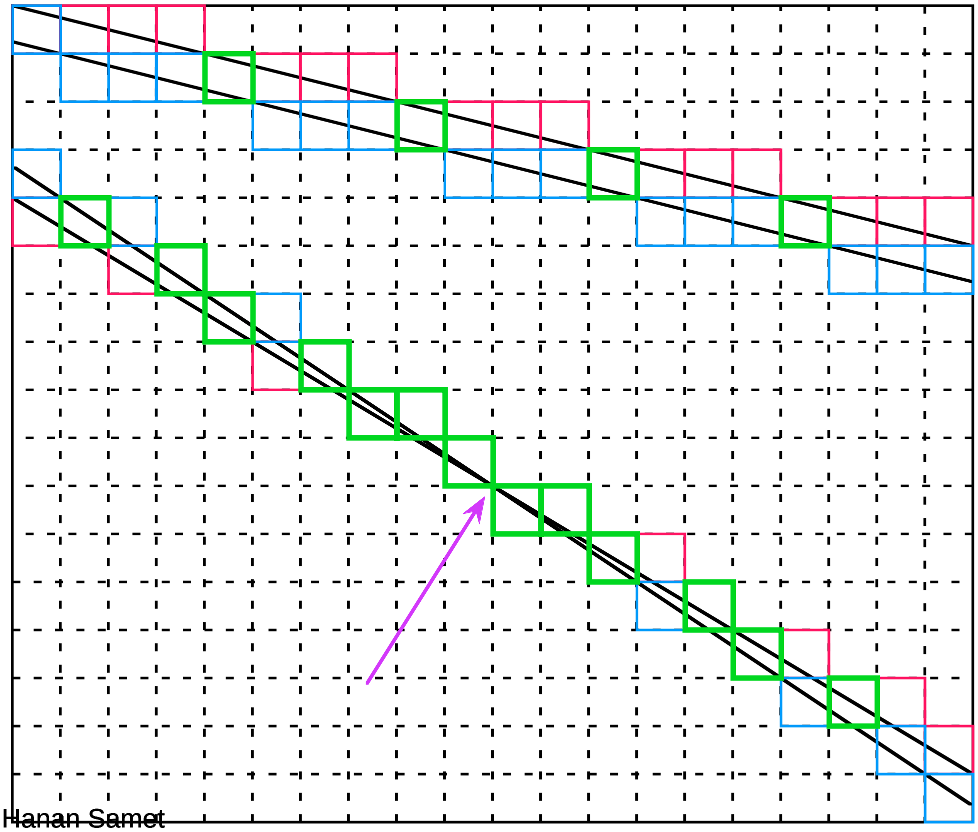
  3. parallel lines

- May not have enough precision to calculate intersection

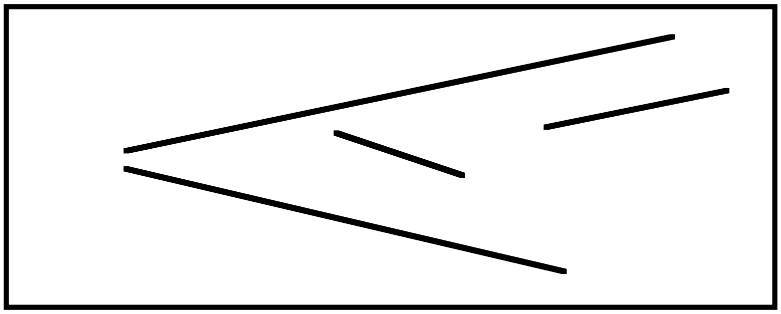
- Digital representation may cause multiple intersections as in the braiding of lines

- Ex: 1. parallel lines

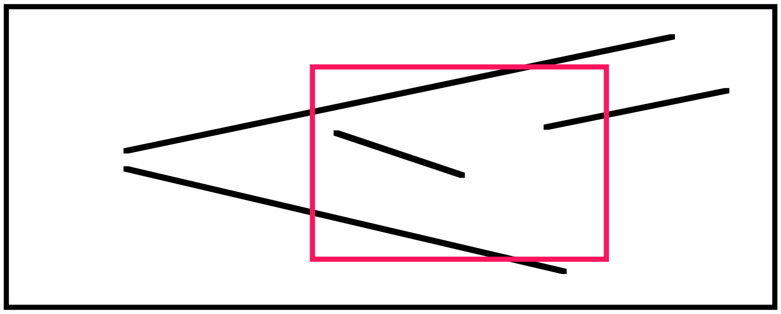
  2. intersecting lines (true intersection point)



WINDOWING AND CLIPPING

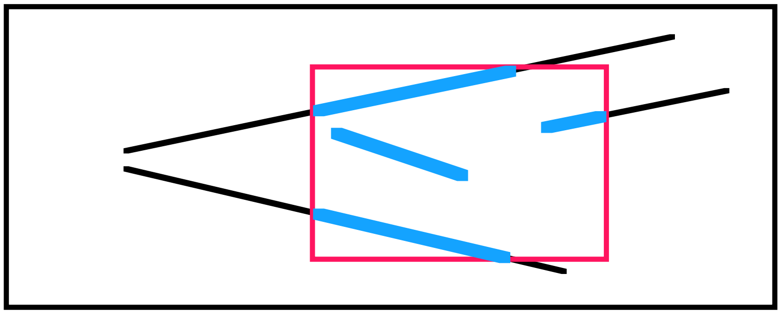


WINDOWING AND CLIPPING



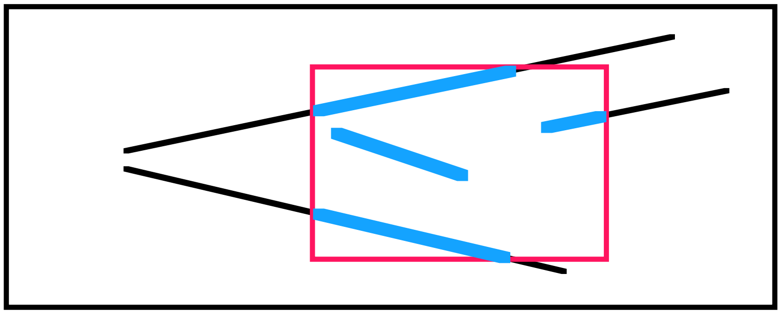
- Windowing: process of extracting the information in a subset (*window*) of the map (usually rectangular)

WINDOWING AND CLIPPING



- **Windowing:** process of extracting the information in a subset (*window*) of the map (usually rectangular)
- **Clipping:** process of eliminating the objects (or parts of objects) that are not in the window

WINDOWING AND CLIPPING

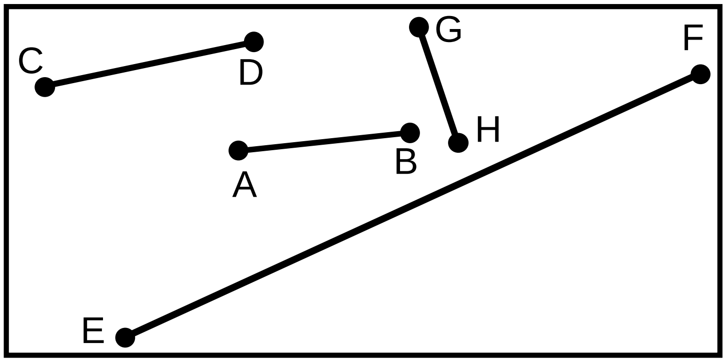


- Windowing: process of extracting the information in a subset (*window*) of the map (usually rectangular)

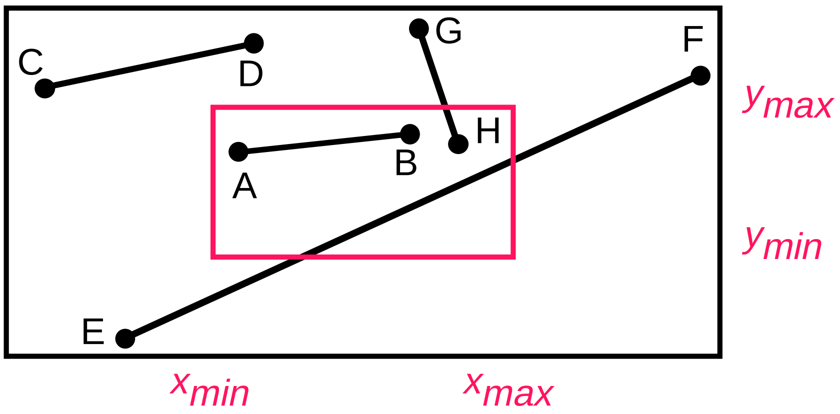
- Clipping: process of eliminating the objects (or parts of objects) that are not in the window

- Brute force solution intersects every line segment with the line segments corresponding to the window boundaries
  1. use equations of line segments
  2. very complex and cumbersome for vertical line segments
  3. use parametric representations of lines:
     - line 1: $x = x_{s1} + t_1 \bullet (x_{e1} - x_{s1})$ and
       $y = y_{s1} + t_1 \bullet (y_{e1} - y_{s1})$
     - line 2: $x = x_{s2} + t_2 \bullet (x_{e2} - x_{s2})$ and
       $y = y_{s2} + t_2 \bullet (y_{e2} - y_{s2})$
- obtain point of intersection by setting the two *x* and *y* coordinate values to each other and solving for t_1 and t_2
- intersection point lies within the ranges of the two lines if t_1 and t_2 are in [0,1]

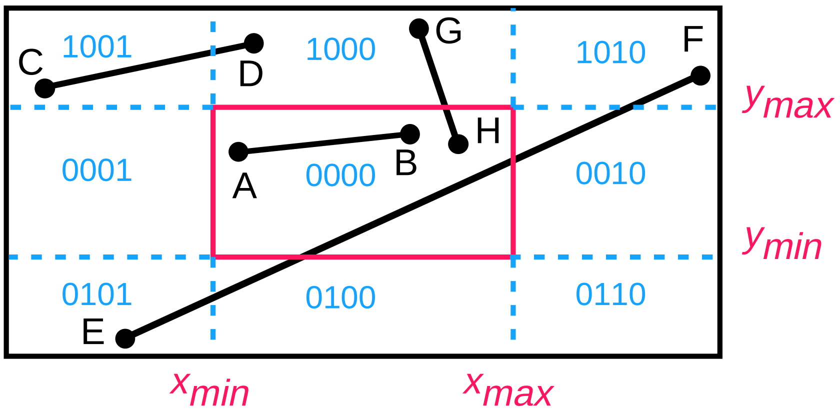
COHEN-SUTHERLAND-LANE ALGORITHM



COHEN-SUTHERLAND-LANE ALGORITHM

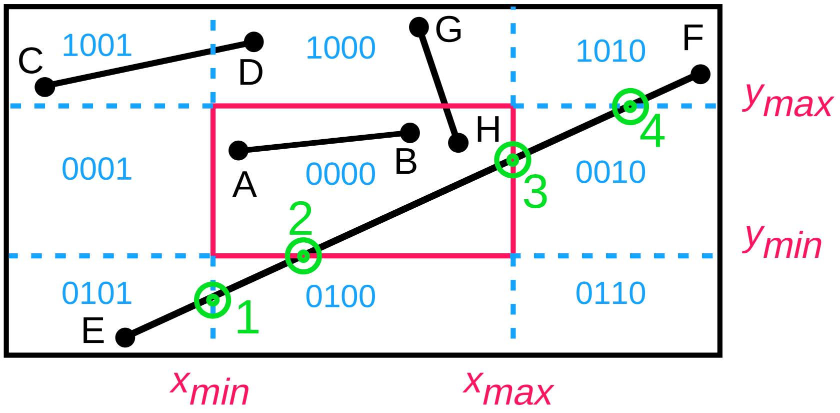


COHEN-SUTHERLAND-LANE ALGORITHM



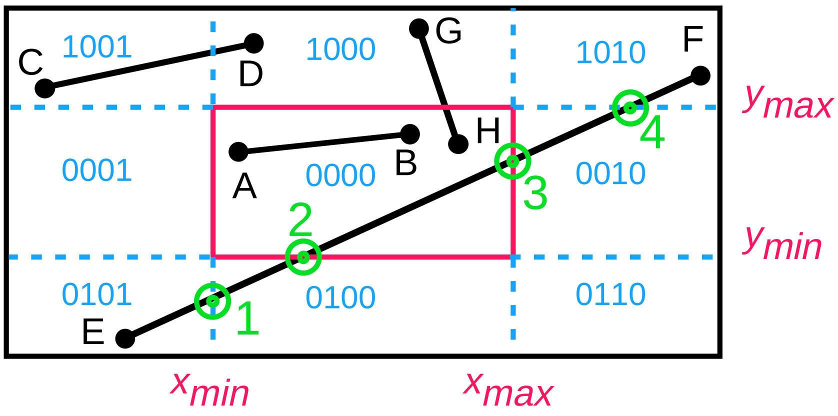
1. Classify regions of endpoints using a 4 bit string (left to right)
   - bit 1: point above top border of window ($y > y_{max}$)
   - bit 2: point below bottom border of window ($y < y_{min}$)
   - bit 3: point right of right border of window ($x > x_{max}$)
   - bit 4: point left of left border of window ($x < x_{min}$)
2. Can use subtraction to classify
3. Accept entire line if both endpoints in region 0000 (e.g., AB)
4. Reject line if logical *and* of codes of endpoints is nonzero (e.g., CD)
5. Otherwise, subdivide line into two parts at a window boundary or its extension using the code of an endpoint that is nonzero and reapply the clipping procedure (e.g., EF)

COHEN-SUTHERLAND-LANE ALGORITHM



1. Classify regions of endpoints using a 4 bit string (left to right)
   - bit 1: point above top border of window ($y > y_{max}$)
   - bit 2: point below bottom border of window ($y < y_{min}$)
   - bit 3: point right of right border of window ($x > x_{max}$)
   - bit 4: point left of left border of window ($x < x_{min}$)
2. Can use subtraction to classify
3. Accept entire line if both endpoints in region 0000 (e.g., AB)
4. Reject line if logical *and* of codes of endpoints is nonzero (e.g., CD)
5. Otherwise, subdivide line into two parts at a window boundary or its extension using the code of an endpoint that is nonzero and reapply the clipping procedure (e.g., EF)
   - at most four iterations (e.g., EF)

COHEN-SUTHERLAND-LANE ALGORITHM



1. Classify regions of endpoints using a 4 bit string (left to right)
   - bit 1: point above top border of window ($y > y_{max}$)
   - bit 2: point below bottom border of window ($y < y_{min}$)
   - bit 3: point right of right border of window ($x > x_{max}$)
   - bit 4: point left of left border of window ($x < x_{min}$)
2. Can use subtraction to classify
3. Accept entire line if both endpoints in region 0000 (e.g., AB)
4. Reject line if logical *and* of codes of endpoints is nonzero (e.g., CD)
5. Otherwise, subdivide line into two parts at a window boundary or its extension using the code of an endpoint that is nonzero and reapply the clipping procedure (e.g., EF)
   - at most four iterations (e.g., EF)

- Improvements (all use the parametric approach)
  1. Cyrus-Beck
     - can clip to an arbitrary polygon or polyhedra
  2. Liang-Barsky
     - very efficient for rectangular regions
  3. Lee-Nichol-Lee
     - most efficient but only works in two dimensions

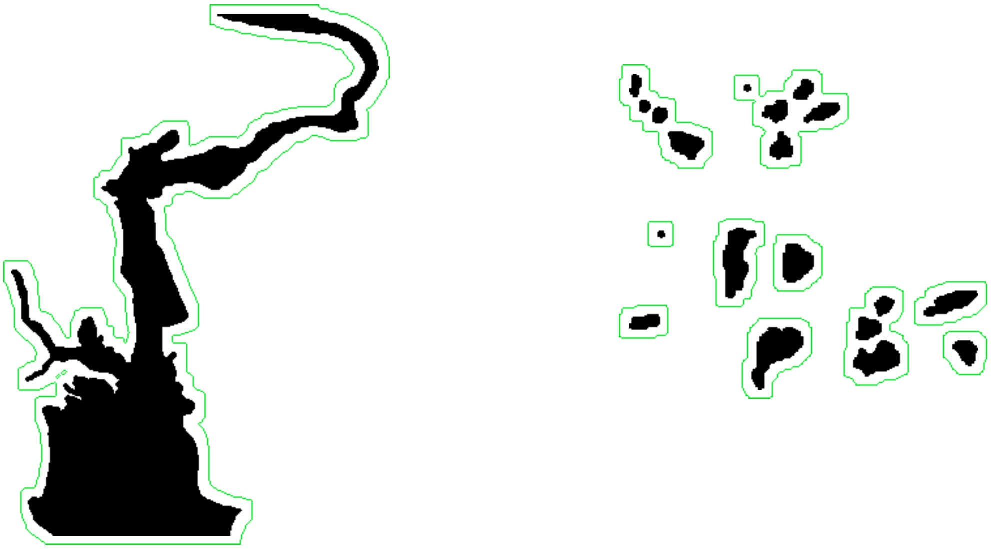
SPATIAL RANGE QUERIES

- Ex: find all objects within 10 miles of St. Louis, the Mississippi River, Lake Michigan, ...

- Analog of range query in conventional databases
  1. ex: find all people with height between 1.5 and 1.8 meters and weight between 60 and 70 kilos
  2. difference is that the range query is not rectangular
  3. instead, region is spatially defined
- Known as a buffer or a corridor in GIS, image dilation or region expansion in image processing
- Can be computed very efficiently for a quadtree by using the L_∞ (i.e., chessboard) distance metric since locus of all points within a given distance is a square
  1. dist(a,b) = $\max(|a_x - b_x|,|a_y - b_y|)$

SPATIAL RANGE QUERIES

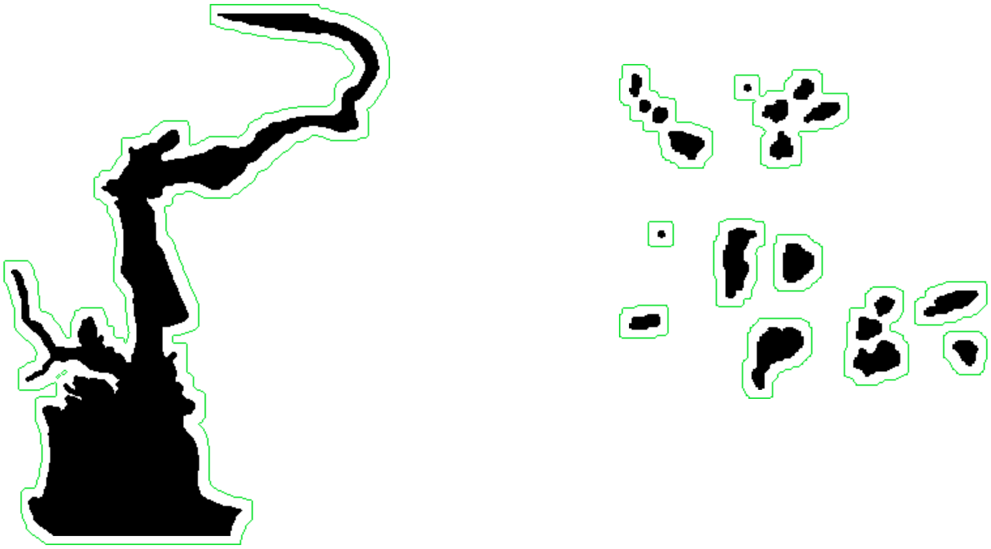
- Ex: find all objects within 10 miles of St. Louis, the Mississippi River, Lake Michigan, ...



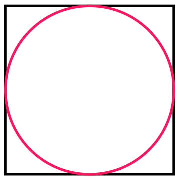
- Analog of range query in conventional databases
  1. ex: find all people with height between 1.5 and 1.8 meters and weight between 60 and 70 kilos
  2. difference is that the range query is not rectangular
  3. instead, region is spatially defined
- Known as a buffer or a corridor in GIS, image dilation or region expansion in image processing
- Can be computed very efficiently for a quadtree by using the L_∞ (i.e., chessboard) distance metric since locus of all points within a given distance is a square
  1. dist(a,b) = $\max(|a_x - b_x|,|a_y - b_y|)$

SPATIAL RANGE QUERIES

- Ex: find all objects within 10 miles of St. Louis, the Mississippi River, Lake Michigan, ...

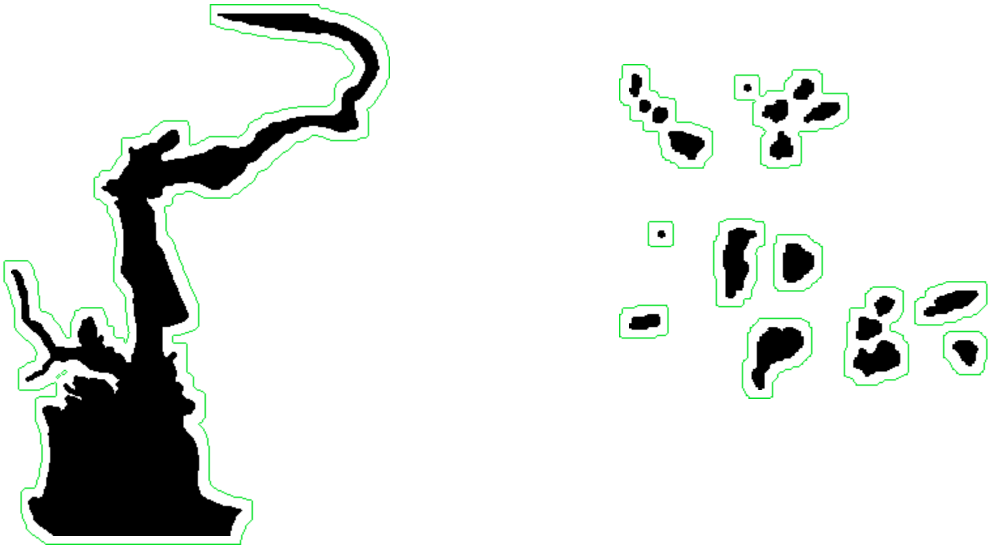


- Analog of range query in conventional databases
  1. ex: find all people with height between 1.5 and 1.8 meters and weight between 60 and 70 kilos
  2. difference is that the range query is not rectangular
  3. instead, region is spatially defined
- Known as a buffer or a corridor in GIS, image dilation or region expansion in image processing
- Can be computed very efficiently for a quadtree by using the L_∞ (i.e., chessboard) distance metric since locus of all points within a given distance is a square
  1. $\text{dist}(a,b) = \max(|a_x - b_x|,|a_y - b_y|)$
  2. an overapproximation to L_2 (Euclidean) distance metric — i.e., $\text{dist}(a,b) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}$

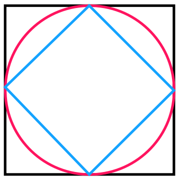


SPATIAL RANGE QUERIES

- Ex: find all objects within 10 miles of St. Louis, the Mississippi River, Lake Michigan, ...



- Analog of range query in conventional databases
  1. ex: find all people with height between 1.5 and 1.8 meters and weight between 60 and 70 kilos
  2. difference is that the range query is not rectangular
  3. instead, region is spatially defined
- Known as a buffer or a corridor in GIS, image dilation or region expansion in image processing
- Can be computed very efficiently for a quadtree by using the L_∞ (i.e., chessboard) distance metric since locus of all points within a given distance is a square
  1. dist(a,b) = $\max(|a_x - b_x|,|a_y - b_y|)$
  2. an overapproximation to L_2 (Euclidean) distance metric — i.e., dist(a,b) = $\sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}$
  3. compare with the L_1 (i.e., city block) distance metric which underapproximates the Euclidean distance — i.e., dist(a,b) = $|a_x - b_x| + |a_y - b_y|$



POINT IN POLYGON TEST

- Shoot a ray from the point and count how many boundaries it intersects

  1. even number means it is outside
  2. odd number means it is inside

- Special cases

  1. ray is collinear with polygon
  2. ray only touches a vertex of the polygon

DATA STRUCTURE TRANSFORMATIONS

1. Only by scale
   - line generalization
   - grid resampling
2. Dimensional change
   - can represent a point by an area (e.g., Voronoi diagram)
   - can represent an area by a point but this is not necessarily invertible—i.e., not every polygon is a Voronoi polygon
3. Change data structure without changing scale or dimension
   - digital elevation model to TIN

SPATIAL INTERPOLATION

- Interpolation is the process of estimating an unknown value of a property at unsampled sites within an area covered by existing point observations
- Extrapolation is an estimate outside an area covered by existing observations
- Based on observation that points that are close in space have a higher probability of having similar values than points that are far apart
- Applications
  1. calculate contours
  2. calculate property of a surface at a specific point
  3. change means of comparison when using different scales or data structures on different map layers
- Types
  1. discrete interpolation—change occurs at the boundary
  2. continuous interpolation—change occurs everywhere

DISCRETE INTERPOLATION

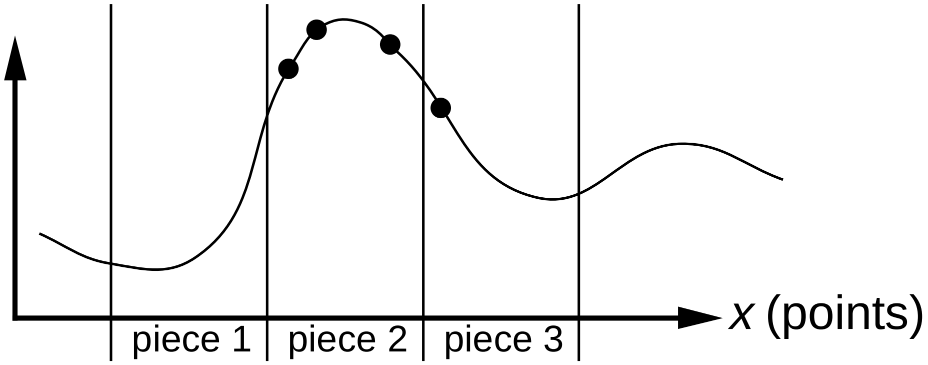
- Areal interpolation—surface has been measured in a set of zones (e.g., density)
  1. different regions have different elevation values—resulting in a skyline
  2. determine the values in another set of zones that overlay the source but do not coincide with it
     - e.g., given populations for school districts, estimate the populations for the police districts
- Best information is obtained from nearest point—e.g., Voronoi diagram
  1. estimate uses a sample of one!
  2. not in line with notion that points close to each other are more likely to be similar than those far apart
  3. good for nominal values since interpolation is irrelevant for them

CONTINUOUS INTERPOLATION

- Point interpolation—surface has been measured at sample points
- Global techniques—use observations at all points of the study area
  1. trend surface analysis—e.g., least squares
  2. Fourier series
- Local techniques—use observations in a small neighborhood
  1. moving averages—really a smoothing technique
     - based on a window
     - weighted by distance
  2. splines
     - approximation by a sequence of piecewise continuous functions

SPLINES

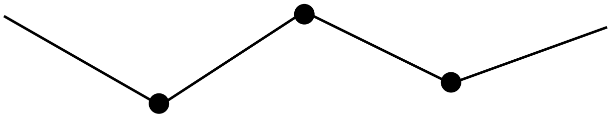
- Assume lines but more general
- Use polynomials to fit a curve through the values (*knots*) of a number of given points while ensuring that the transitions are continuous (thus *piecewise continuous*)
- Ex: *y* (knots)



- Degree of polynomials depends on the number of points used and denotes the number of continuous derivatives (i.e., continuity of the transitions between the pieces)

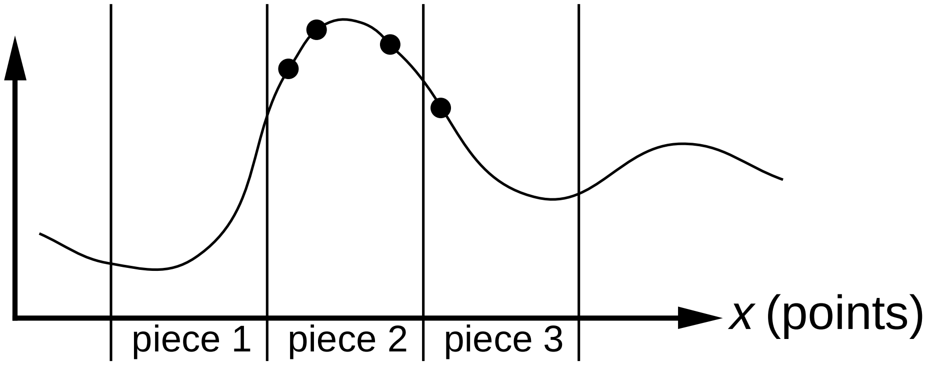
1. linear - 2 points and no continuous derivatives
2. quadratic - 3 points and continuous first derivatives
3. cubic - 4 points and continuous first and second derivatives
   - bicubic spline describes three-dimensional situation where a surface is being fit instead of a line

- If values of a few points change, then can modify a small part of the curve without having to recompute entire curve



SPLINES

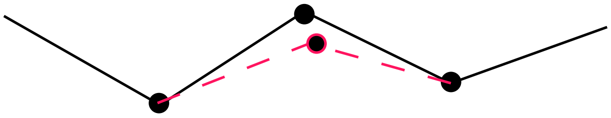
- Assume lines but more general
- Use polynomials to fit a curve through the values (*knots*) of a number of given points while ensuring that the transitions are continuous (thus *piecewise continuous*)
- Ex: *y* (knots)



- Degree of polynomials depends on the number of points used and denotes the number of continuous derivatives (i.e., continuity of the transitions between the pieces)

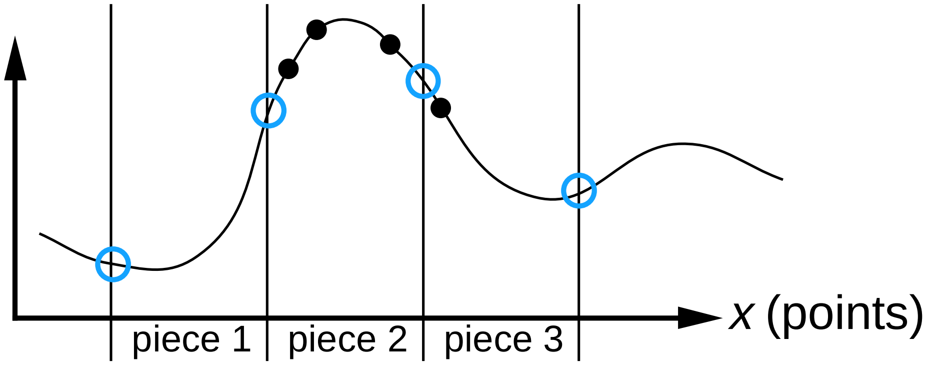
1. linear - 2 points and no continuous derivatives
2. quadratic - 3 points and continuous first derivatives
3. cubic - 4 points and continuous first and second derivatives
   - bicubic spline describes three-dimensional situation where a surface is being fit instead of a line

- If values of a few points change, then can modify a small part of the curve without having to recompute entire curve

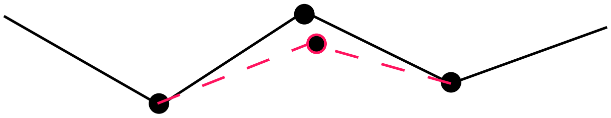


SPLINES

- Assume lines but more general
- Use polynomials to fit a curve through the values (*knots*) of a number of given points while ensuring that the transitions are continuous (thus *piecewise continuous*)
- Ex: *y* (knots)



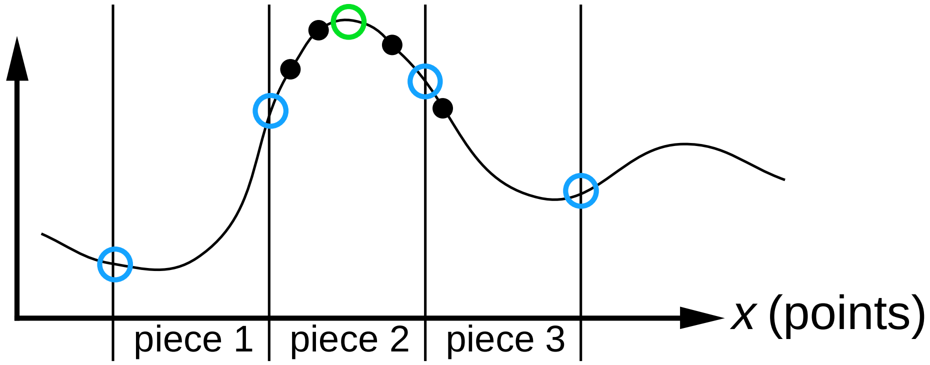
- Degree of polynomials depends on the number of points used and denotes the number of continuous derivatives (i.e., continuity of the transitions between the pieces)
  1. linear - 2 points and no continuous derivatives
  2. quadratic - 3 points and continuous first derivatives
  3. cubic - 4 points and continuous first and second derivatives
     - bicubic spline describes three-dimensional situation where a surface is being fit instead of a line
- If values of a few points change, then can modify a small part of the curve without having to recompute entire curve



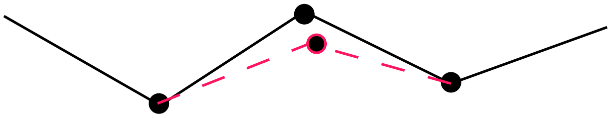
- The junctions need not occur at the data points although they usually do

SPLINES

- Assume lines but more general
- Use polynomials to fit a curve through the values (*knots*) of a number of given points while ensuring that the transitions are continuous (thus *piecewise continuous*)
- Ex: *y* (knots)



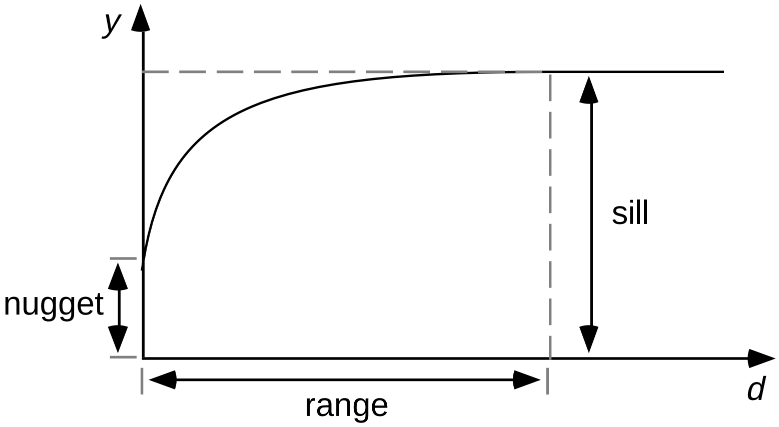
- Degree of polynomials depends on the number of points used and denotes the number of continuous derivatives (i.e., continuity of the transitions between the pieces)
  1. linear - 2 points and no continuous derivatives
  2. quadratic - 3 points and continuous first derivatives
  3. cubic - 4 points and continuous first and second derivatives
     - bicubic spline describes three-dimensional situation where a surface is being fit instead of a line
- If values of a few points change, then can modify a small part of the curve without having to recompute entire curve



- The junctions need not occur at the data points although they usually do
- Maxima and minima need not necessarily occur at the data points

KRIGING

- Makes use of semivariance—variance of differences only depends on the distance between the sites
- Tabulates the average variance (e.g., difference in elevation) between any two points that are a distance *d* apart



1. y is $E(z_i - z_j)^2$
2. d is the distance (e.g., in the plane) between points i and j
3. *sill* is the maximum variance
4. *range* is the maximum distance between the points that makes any contribution to the interpolation
   - this is where the differences are spatialy-dependent
   - use the mean outside the range
   - a good estimate for the size of the window in the moving-average method
5. *nugget* is the y-intercept
   - non-zero value indicates that repeated measurements at the same point will yield different values—i.e., a residual error
   - if the nugget value dominates, then data is so noisy that spatial interpolation is meaningless and just use the mean

MECHANICS OF KRIGING AND VARIOGRAMS

- Nature of variation on the surface affects the variogram
  1. simple Kriging assumes a constant mean (i.e., that all variation is statistical)
  2. uniform Kriging assumes the existence of trends
     - i.e., different means in different regions implies that the variogram and the interpolation are archieved by detrending the original data through subtraction of appropriate values corresponding to the means in the different regions
- Constructing the variogram
  1. partition the distance into distinct intervals—i.e., form bins
  2. compute the distance and squared difference in *z* values for every possible pair of sample points and associate the difference with the bin corresponding to its distance interval
  3. calculate the average squared difference for each bin and plot it at the midpoint of the bin’s interval
- Using the variogram
  1. use a sum of weighted values of known points
  2. weights depend on distance between the point to be interpolated and the known points
  3. *y* - values on the variogram can provide the weights
- Computationally expensive when the number of known points is large

THREE-DIMENSIONAL TRANSFORMATIONS

1. Between data structures
   - interpolation to a grid
   - surface-specific point sampling
2. Transformation between scales
   - surface generalization
3. Analytical transformations
   - elevations to slope and aspect
   - terrain partitioning—e.g., TIN creation
     a. point to grid, point to TIN, and TIN to grid are interpolation problems and relatively simple
     b. grid to TIN is difficult
        - TIN is compact
        - must be able to detect critical points in terrain
   - intervisibility
     a. given a point, find visible and invisible regions
     b. useful in siting towers and planning
4. Terrain symbolization
   - automated contouring
   - hill shading
     a. enables detecting if in shadow
     b. enables generating realistic perspective views
   - gridded perspective views—3-d in 2-d

INTERPOLATION TO A GRID

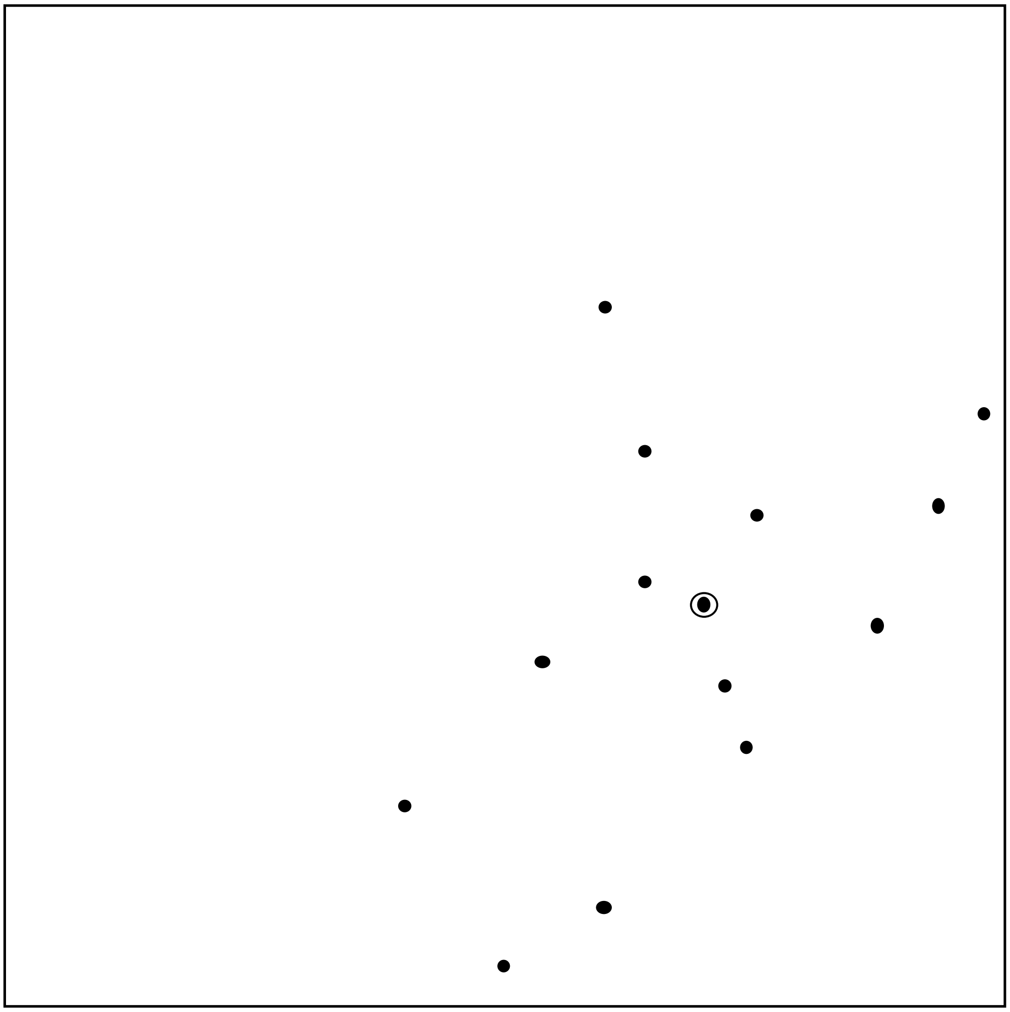
- Problem: given a set of data, build a grid for them
- Use neighborhood property of a surface
  1. elevations are continuously distributed
  2. no sudden cliffs, caves, overhangs, etc.
  3. values are closely related to those in proximity

INTERPOLATION ISSUES

- What is a neighborhood?
  1. a given number of points satisfying a particular condition, OR
  2. all the points that satisfy a particular condition
- How to weight the contributions of the neighboring data points?
  1. equally
  2. inversely with distance from the grid point
     - violates neighborhood property which says that the elevation at any point is most closely related to the points which are closest to it and ignores points further away
- Where are the maxima and minima of the interpolated surface?
  1. at the data points, OR
  2. at interpolated points
- How should the grid relate to the data points (i.e., spacing)?
  1. data rich and data poor areas complicate the problem
  2. sampling theorem implies that once the grid spacing has been chosen, all features with a spatial size (e.g., extent) less than twice the spacing can be ignored

RESTRICTING NEIGHBORHOOD OF INTEREST

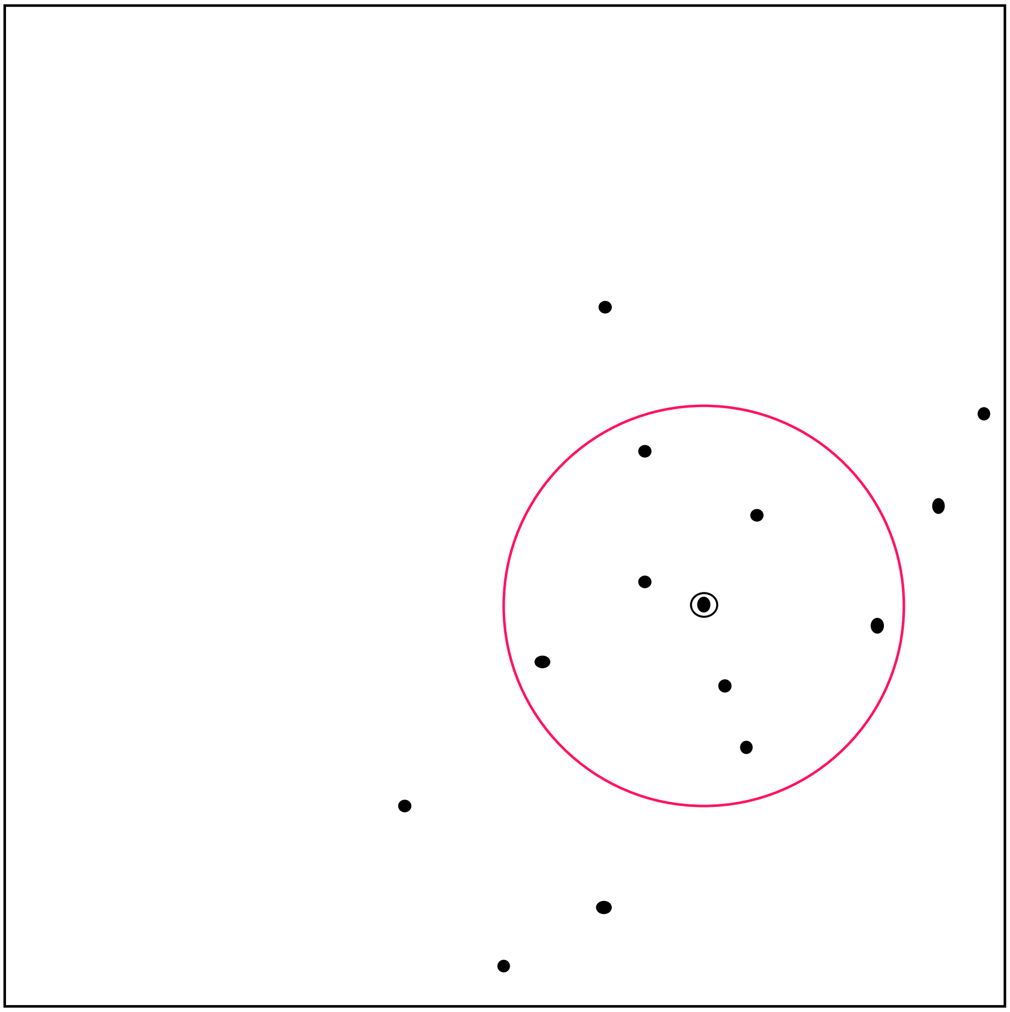
- Limit number of data points



RESTRICTING NEIGHBORHOOD OF INTEREST

- Limit number of data points

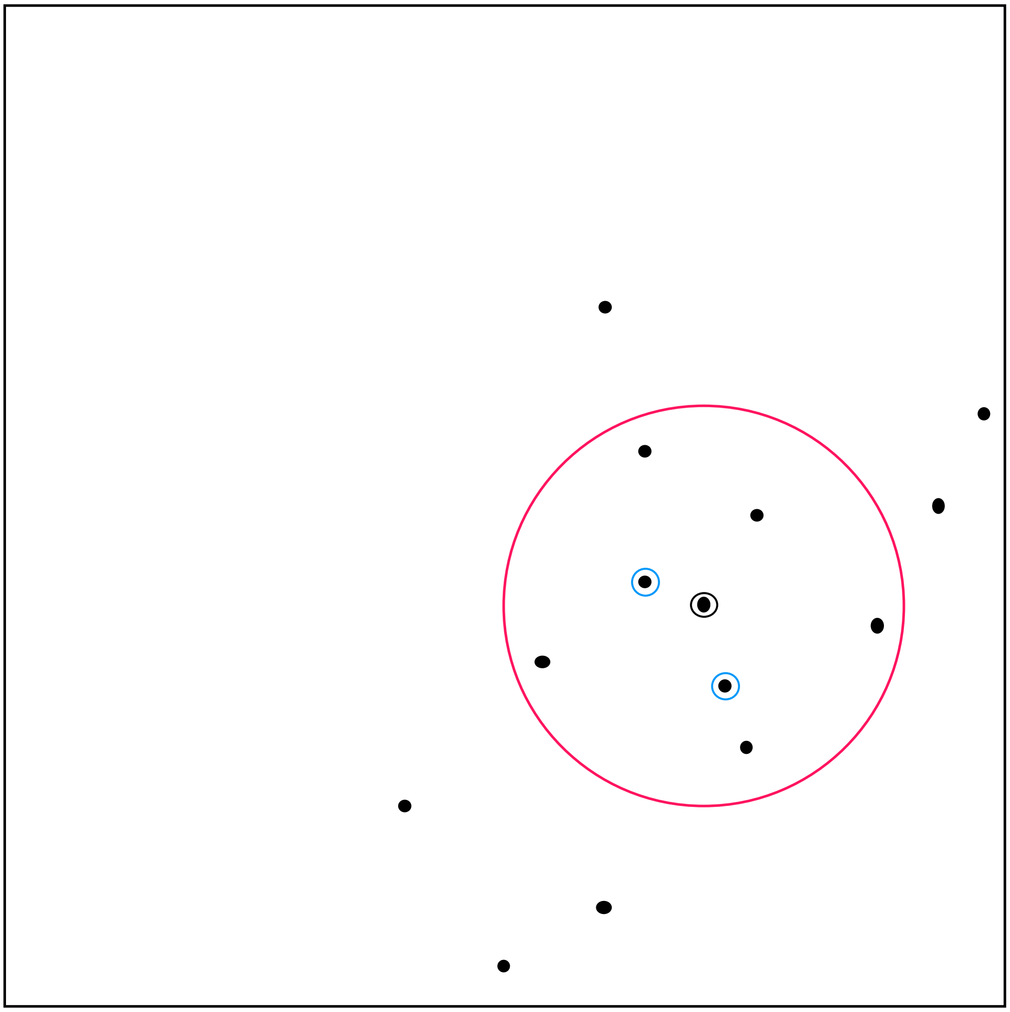
  1. by radius



RESTRICTING NEIGHBORHOOD OF INTEREST

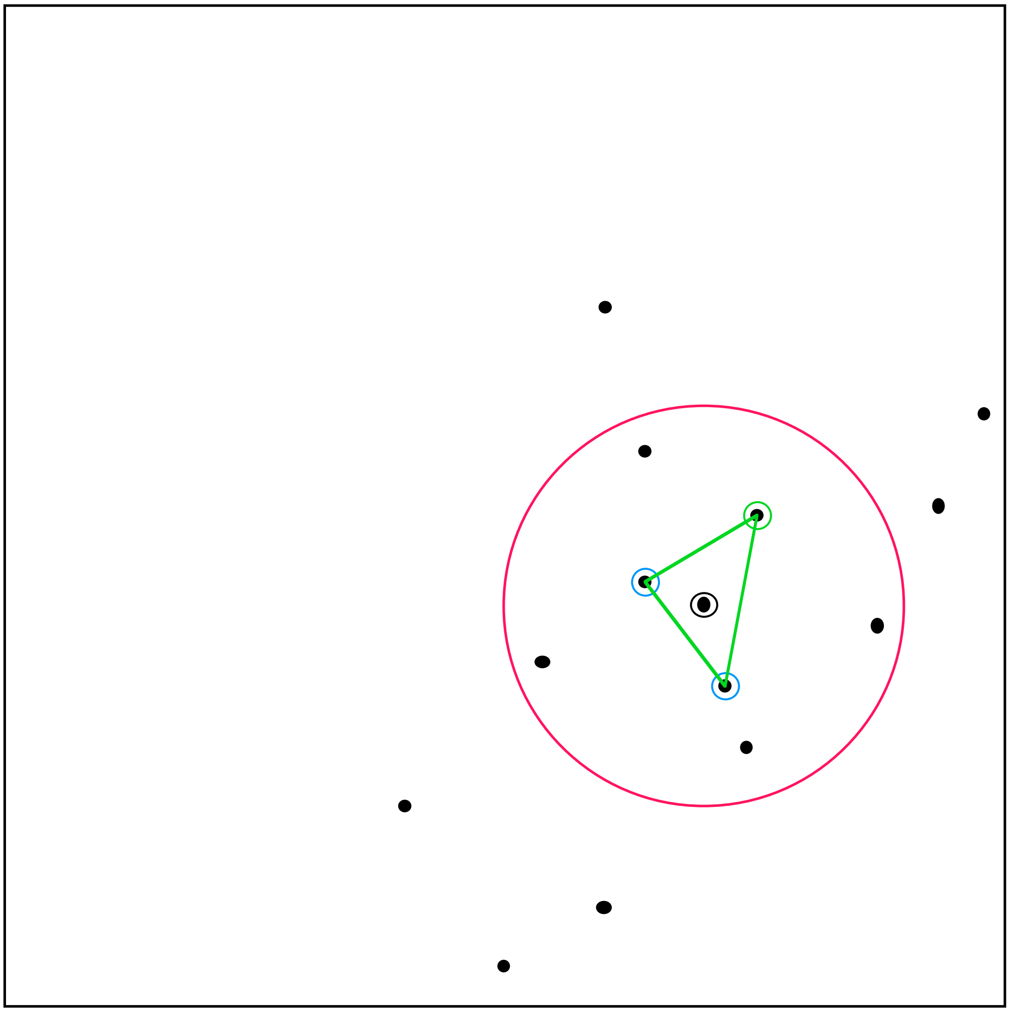
- Limit number of data points

  1. by radius
  2. take nearest *n* (e.g., 2) points



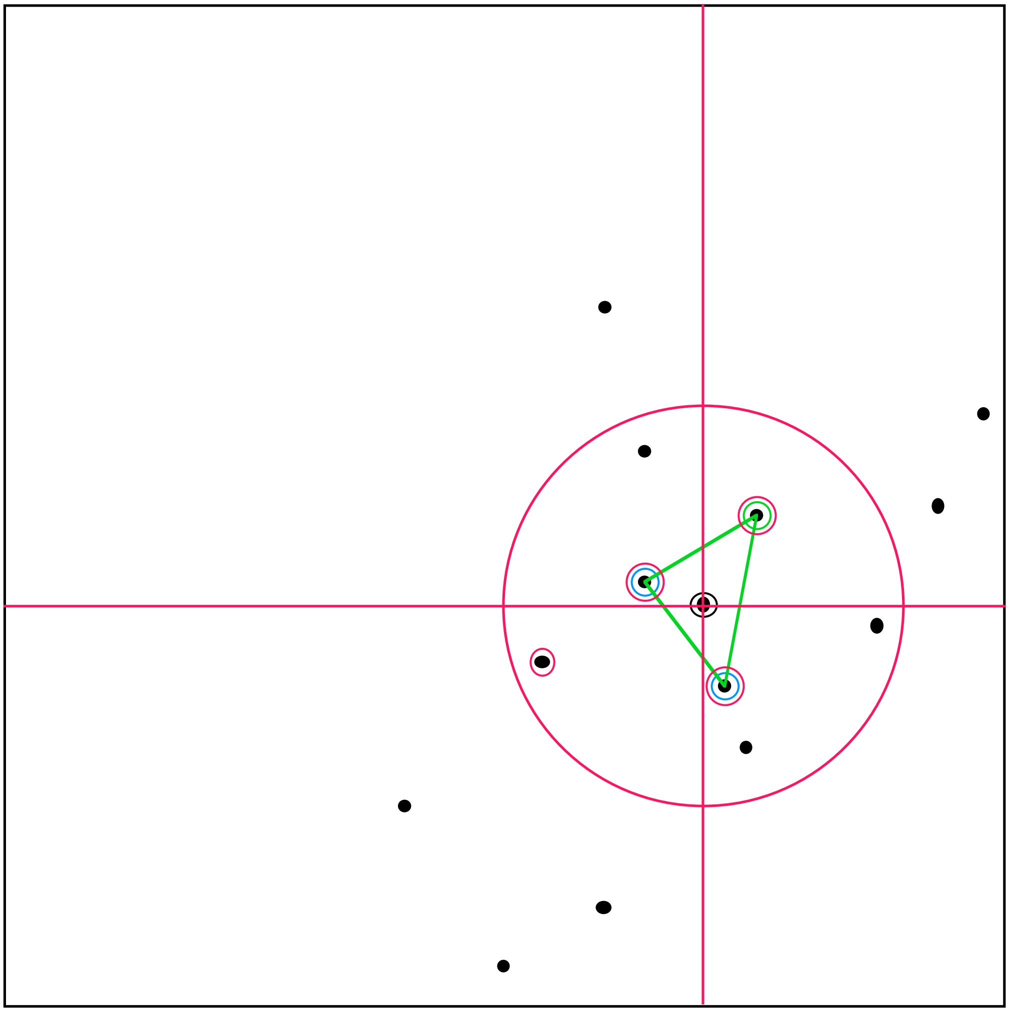
RESTRICTING NEIGHBORHOOD OF INTEREST

- Limit number of data points
  1. by radius
  2. take nearest *n* (e.g., 2) points
     - problem when data points are clustered
     - if inverse weight is linear, neighborhood is 3, and neighborhood surrounds grid point, then equivalent to interpolation over a TIN



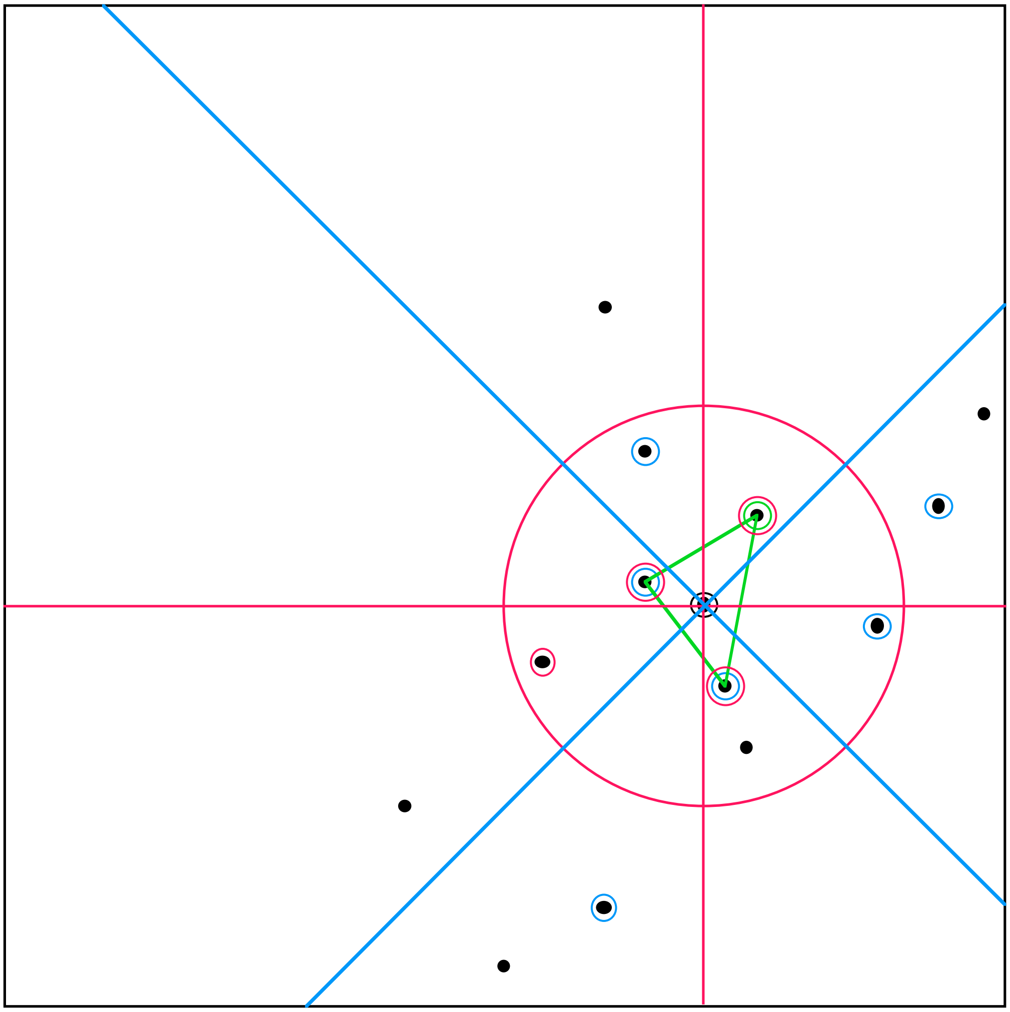
RESTRICTING NEIGHBORHOOD OF INTEREST

- Limit number of data points
  1. by radius
  2. take nearest *n* (e.g., 2) points
     - problem when data points are clustered
     - if inverse weight is linear, neighborhood is 3, and neighborhood surrounds grid point, then equivalent to interpolation over a TIN
- Avoid directional clustering
  1. choose nearest points in each quadrant



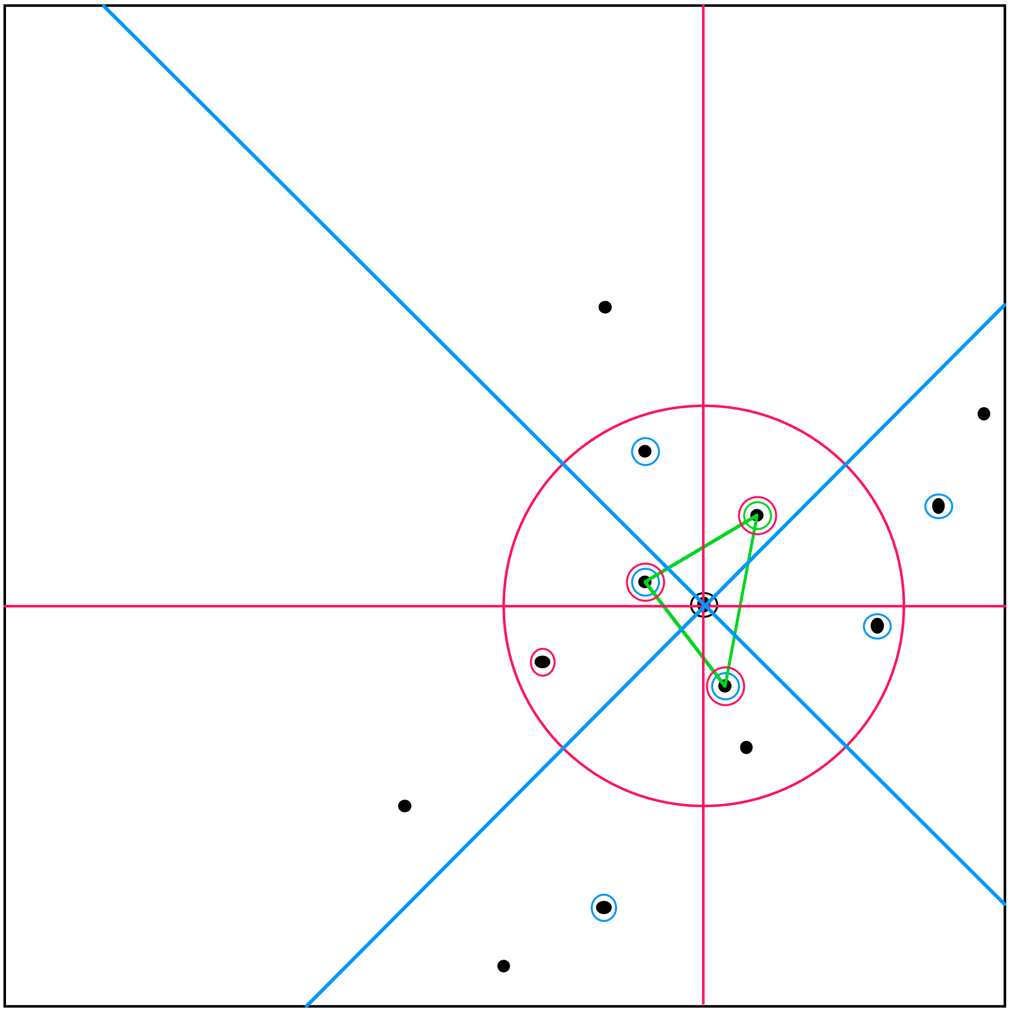
RESTRICTING NEIGHBORHOOD OF INTEREST

- Limit number of data points
  1. by radius
  2. take nearest *n* (e.g., 2) points
     - problem when data points are clustered
     - if inverse weight is linear, neighborhood is 3, and neighborhood surrounds grid point, then equivalent to interpolation over a TIN
- Avoid directional clustering
  1. choose nearest points in each quadrant or octant



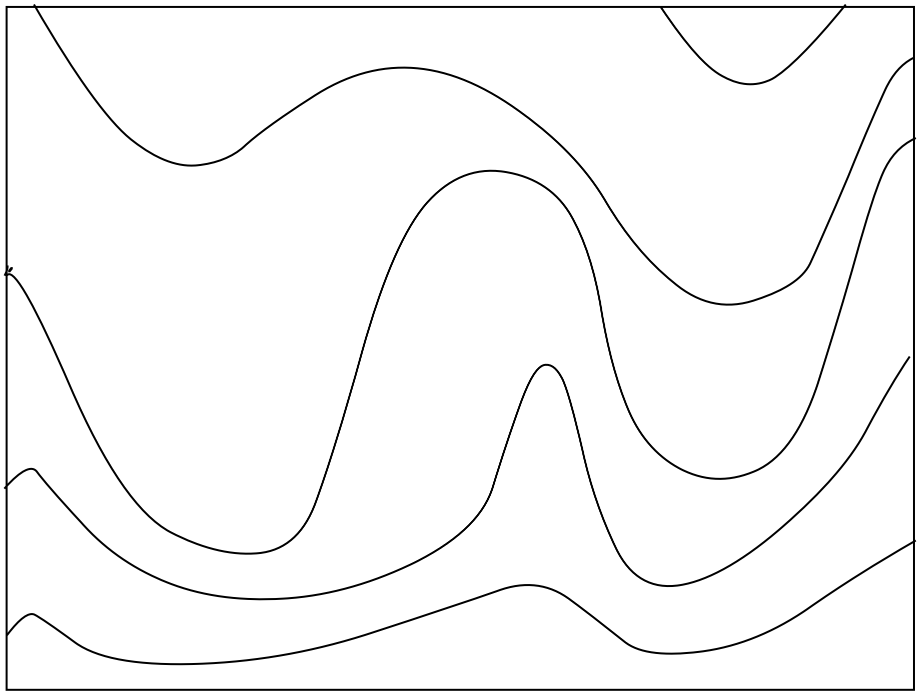
RESTRICTING NEIGHBORHOOD OF INTEREST

- Limit number of data points
  1. by radius
  2. take nearest *n* (e.g., 2) points
     - problem when data points are clustered
     - if inverse weight is linear, neighborhood is 3, and neighborhood surrounds grid point, then equivalent to interpolation over a TIN
- Avoid directional clustering
  1. choose nearest points in each quadrant or octant
  2. problematic when no points in some quadrants or octants
- Use Kriging



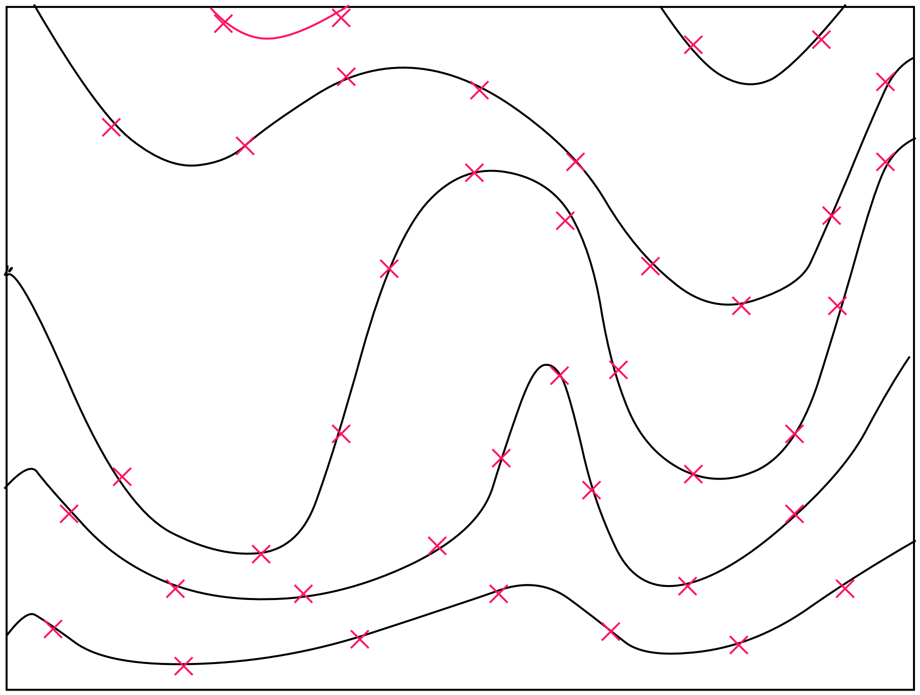
SURFACE-SPECIFIC POINT SAMPLING

- If data is from a contour map, placing a grid on a map and simply digitizing the elevations at the grid points can lead to significant errors
- Errors can result in missing streams, ridges, summit points, saddle points, and bottoms of depressions
- Digitization along the contour is one solution
  1. OK when the spacing between data points along the contour is similar to that between the contours
  2. OK in rough terrain



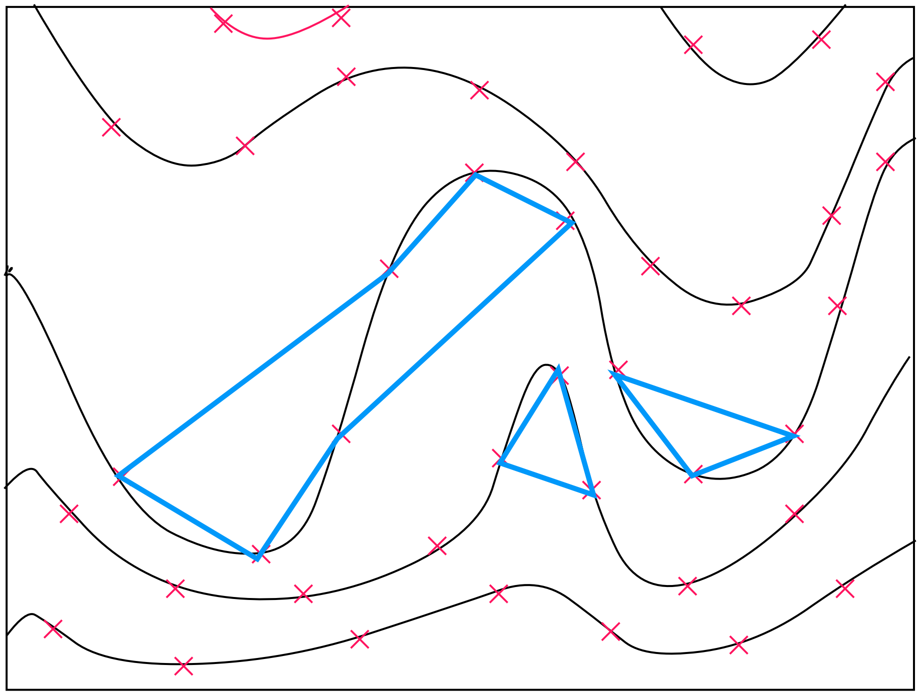
SURFACE-SPECIFIC POINT SAMPLING

- If data is from a contour map, placing a grid on a map and simply digitizing the elevations at the grid points can lead to significant errors
- Errors can result in missing streams, ridges, summit points, saddle points, and bottoms of depressions
- Digitization along the contour is one solution
  1. OK when the spacing between data points along the contour is similar to that between the contours
  2. OK in rough terrain
  3. problem with interpolating using points
     - if only use a few points in the neighborhood search, then results in the creation of artificial plateaus within the loops of the contours



SURFACE-SPECIFIC POINT SAMPLING

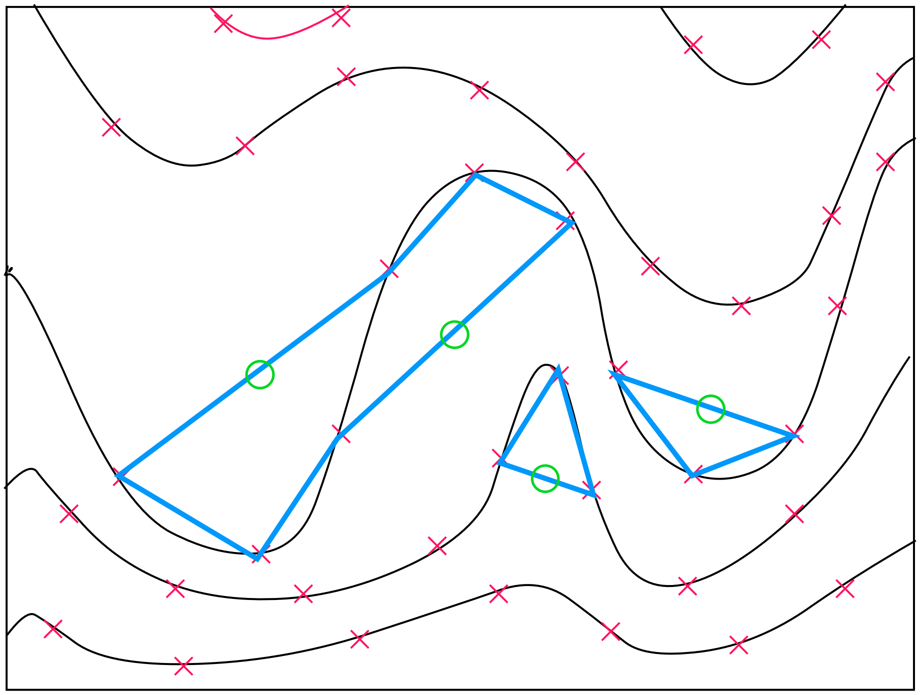
- If data is from a contour map, placing a grid on a map and simply digitizing the elevations at the grid points can lead to significant errors
- Errors can result in missing streams, ridges, summit points, saddle points, and bottoms of depressions
- Digitization along the contour is one solution
  1. OK when the spacing between data points along the contour is similar to that between the contours
  2. OK in rough terrain
  3. problem with interpolating using points
     - if only use a few points in the neighborhood search, then results in the creation of artificial plateaus within the loops of the contours



- need to interpolate using lines

SURFACE-SPECIFIC POINT SAMPLING

- If data is from a contour map, placing a grid on a map and simply digitizing the elevations at the grid points can lead to significant errors
- Errors can result in missing streams, ridges, summit points, saddle points, and bottoms of depressions
- Digitization along the contour is one solution
  1. OK when the spacing between data points along the contour is similar to that between the contours
  2. OK in rough terrain
  3. problem with interpolating using points
     - if only use a few points in the neighborhood search, then results in the creation of artificial plateaus within the loops of the contours



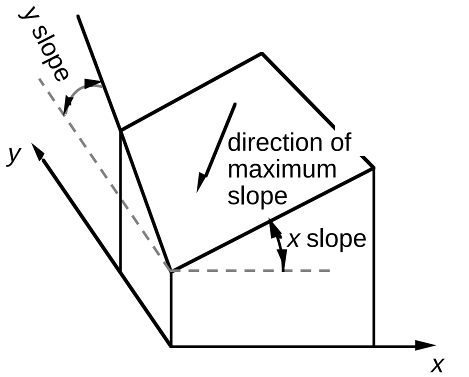
- need to interpolate using lines
- plateau is caused by using the elevation of the closest point along a contour

SURFACE OR SPATIAL FILTERING

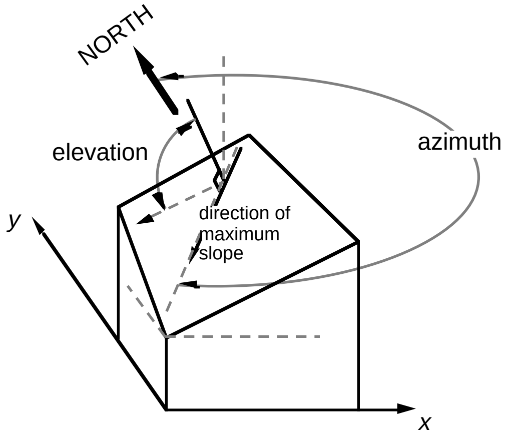
- Performed exclusively on gridded data
- Process each grid cell by a filter (i.e., window) which weights each grid cell by its neighboring values (positive and negative weights)
- Weights must sum to unity as otherwise the terrain is
  1. damped (< 1.0)
  2. amplified (> 1.0)
- Special-purpose filters can be used to enhance particular features on the basis of size, shape, or orientation
  1. horizontal linear can amplify erroneous scan lines
  2. filter specific shapes (e.g., circle)
  3. enhance features of width of one grid cell
     - 3 by 3 window
     - 8 negative weights and one positive weight
     - good approximation of random noise and then subtract it from original to yield “enhanced” image
  4. computationally complex

SLOPE AND ASPECT TRANSFORMATIONS

- Slope is defined by a plane tangent to a surface at a specific point
  1. rate of change of elevation
  2. can calculate in *x* direction, *y* direction, or in the direction of maximum slope (termed *gradient*)



- Aspect is the direction faced by the surface and measured by both the horizontal and vertical angles faced by the surface:
  1. the horizontal angle (azimuth) is measured by moving clockwise from due north to the direction of maximum slope (i.e., gradient)
  2. the elevation angle is measured by moving from horizontal plane to line drawn normal to surface (= 90°– gradient?)



- Maximum slope is known to skiers as the fall line and the aspect is the direction in which the fall line tends
- Slope of zero implies flat terrain and aspect is undefined
- Slope is computed by best-fit plane through points in neighborhood

- TIN
  1. each triangle has uniform slope and a single aspect
  2. slope and aspect are discontinuous at the triangle boundaries

ERROR

1. Source errors
   - often digital data is an “unbelievably faithful reproduction of incorrect maps”
   - in the digital cartographic data
   - in the data capture and geocoding process
     a. missing parts of an area
     b. inappropriate density (e.g., elevation data)
2. Use errors
   - lack of information about mapping system
   - unexpected deviation from cartographic convention
   - use of maps with too small a scale
   - use of out-of-date maps
   - lack of documentation on the quality of the data

3. Process errors

- numerical errors
  a. floating point precision
  b. e.g., line intersection problem
- errors in the geometry of the map
  a. features mislocated in three-dimensional geometry of the world
  b. often caused by controlled distortion
     - changing map projection
     - statistical space fitting
- scale change
  a. ideally one high resolution data base instead of many sources at different scales
  b. some errors cannot even be detected
     - removal of islands at smaller scale
- transformation of cartographic objects in digital form between data structures
  a. not really an unknown source since the process is known
     - gridding error in vector-to-raster conversion
     - map overlay can result in many small meaningless polygons
  b. can use topology to perform consistency checks
- changes in data dimension
  a. intentional errors
  b. but OK since permit analysis or display which is otherwise impossible
- type of symbolization used