Visualização de Dados
Data Visualization
2021/2022

Mestrado em Engenharia Informática
Mestrado em Informática
Especialização em Informática
Outros Mestrados e pós-graduações

1st semester

Course slides

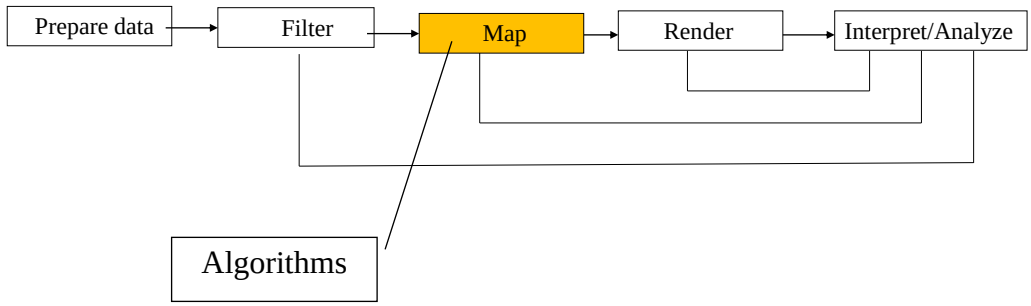
Beatriz Carmo
DI-FCUL GU-VD-21-3

Outline

- Algorithms classification according to:
  - Structural transformations
  - Dataset type
- Algorithms for the visualization of scalar quantities
  - **Color mapping** (*Mapeamento em cores*)
  - **Contouring** (*Traçado de isolinhas e isosuperfícies*)
  - **Carpet Plots** (height plots) (*Deformação de superfícies*)
- Other algorithms
  - Scalar generation
  - Probing (*Sondagem*)

Scientific Data Visualization

Conceptual model of the visualization process
(Visualization pipeline)



Algorithms

- The algorithms that transform the data are the heart of visualization
- The **algorithms** can be **classified** taking into account:
  - The **effect that transformation has on the topology and geometry** of the dataset - **structural transformations**
  - The **type of dataset** that the algorithm operates on

Algorithms

Structural transformations

- Geometric transformations
- Topological transformations
- Attribute transformations
- Combined transformations

Algorithms

Geometric transformations

- alter input geometry
- but do not change the topology of the dataset
- nor the attributes

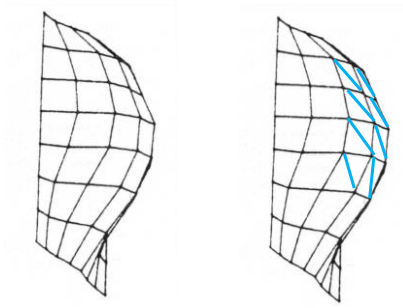
Ex: applying translations, rotations, scaling

Topological transformations

- alter input topology
- but do not change the geometry of the dataset
- nor the attributes

An example of topological transformation is the to convert a structured grid into a unstructured grid without changing points location.

An example of topological transformation



Adding more edges in a grid may transform the grid in a smoother representation of the object

When we add edges, we change the topology

Algorithms

Attribute transformations

- do not alter the geometry
- nor the topology of the dataset
- convert data attributes

- Convert data attributes from one form to another or create new data attributes

For instance,

- Calculating vector magnitude
- Creating a vector attribute, by combining 3 scalar attributes as components of a 3D space.

Algorithms

Combined transformations ——— Change both dataset structure and attribute data

For instance, computing contour lines creates a new data structure associated with attributes

Algorithms

Taking into account the **type of dataset** that the algorithm operates on:

- Scalar algorithms
  The generation of contour lines using temperature data
- Vector algorithms
  The generation of oriented arrows to represent vector data
- Tensor algorithms
  The generation of oriented icons to represent the components of stress in a material

Algorithms

Modelling algorithms correspond to the algorithms that do not fit into any other category.

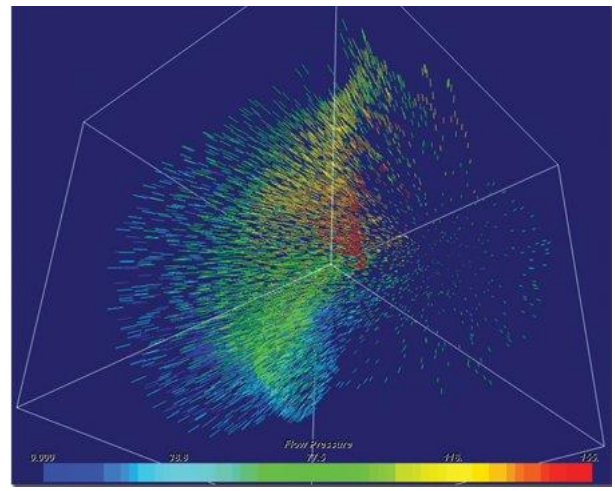
For instance, an algorithm that creates an oriented ***glyph*(*)** using a vector and then scaled **or** colored according to a scalar.

It combines a scalar algorithm with a vector algorithm.

(*) A *glyph* is a graphical representation whose size, orientation, shape or graphical attributes can change according with the attributes it represents.

Vector algorithm combined with a scalar algorithm

- **oriented *glyph*** to represent a vector
- **coloured** according to a scalar



Scalar Algorithms

Scalars are single data values associated with each point and/or cell of a dataset

The most common algorithms to visualize scalars are:

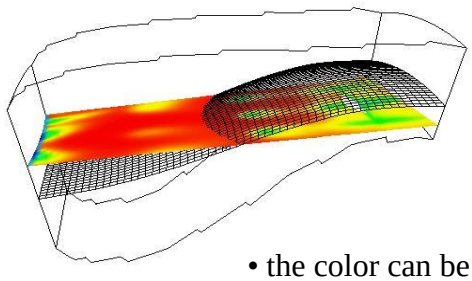
- **Color mapping** (*Mapeamento em cores*)
- **Contouring** (*Traçado de isolinhas e isosuperfícies*)
- **Carpet Plots** (height plots) (*Deformação de superfícies*)

Other algorithms

- Scalar generation
- Probing (*Sondagem*)

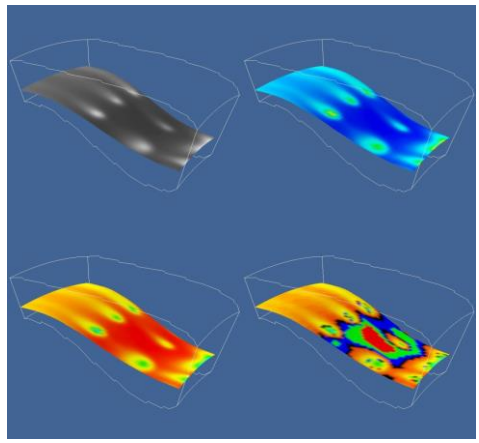
Color mapping (*Mapeamento em cores*)

It maps scalar data to colors



- the color can be applied to an object that already exists (for instance, a grid slice)
- or to an object that it is created for that purpose (for instance, a cutting plane)

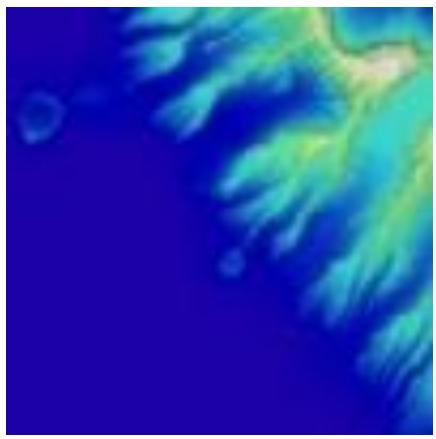
Color mapping (*Mapeamento em cores*)



• In this case the color mapping is applied to a **slice** of the grid

• Several color maps can be used

Color mapping (*Mapeamento em cores*)



Visualizing terrain using a color map based on height.
(Schroeder, Martin, Lorensen et al.)

Color mapping (*Mapeamento em cores*)

Color mapping is implemented by indexing a **scalar** into a **color lookup table**

To convert a discrete variable

Establish a one-to-one correspondence between a value and a color

The scalar value is the index of the lookup table

Color mapping (*Mapeamento em cores*)

To convert a continuous variable (ex. Temperature**),** it is necessary apply a scalar value into an index of the color lookup table

Associated with the table is a minimum (v_{min}) and maximum (v_{max}) scalar range

Considering a table with n colors

If the scalar is less than a v_{min}, index = 0

If the scalar is greater or equal to v_{max}, index = n – 1

For intermediate values, s,

index = floor **(** n [(s - *min*) / (*max* - *min*)] **)**

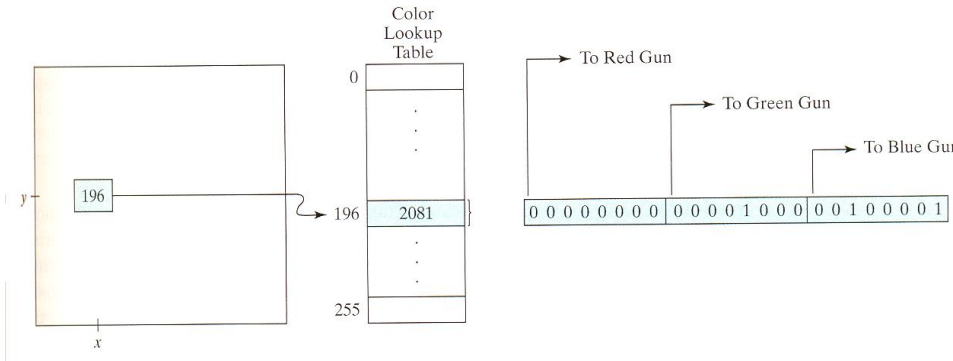
Obs: The floor function of x gives the largest integer less than or equal to x (função característica)

Number of colors: n = 5
Vmax = 100
Vmin = 50

	50	60	70	80	90	100
Values						
indexes	0	1	2	3	4	
color	Cor0	Cor1	Cor2	Cor3	cor4	

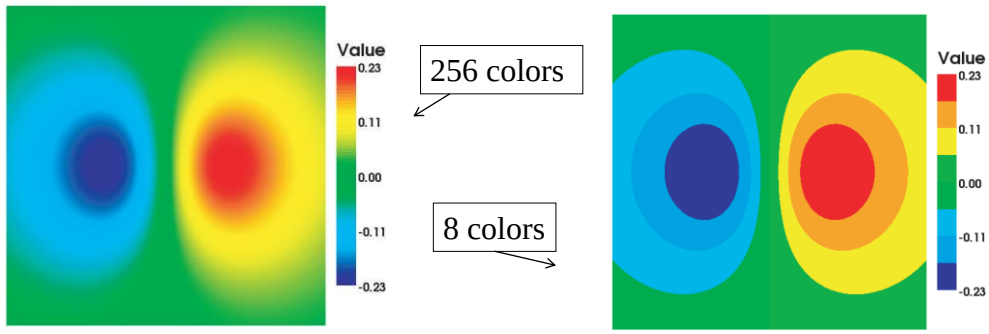
s = 75
index (75) = floor (5 x (75-50)/ (100-50))
=2

From index to color



A *LookUp Table* (LUT) with 24 bits for each color is accessed by an 8 bits *frame buffer* . It allows a maximum of 2^{24} (~17 millions) of different colors but in sets of 256 colors each (2^8=256)

Color bands



The dataset is visualized using the **same rainbow color scale** but with a **different number of colors in the look-up table**
With fewer colors, equal color bands become visible in both the image and the color legend [Telea2014]

Color mapping (*Mapeamento em cores*)

- The scalar mapping is implemented by indexing into a color lookup table
- A more general form of the lookup table is called a **transfer function (*função de transferência*)**
- A **transfer function** is any expression that maps scalar values into a color specification.

  Actually a lookup table is a discrete sampling of a transfer function.
- Transfer functions can be used to map scalar data into other information, like **transparency.**
- An important issue to color mapping is to **choose the color scale carefully**.

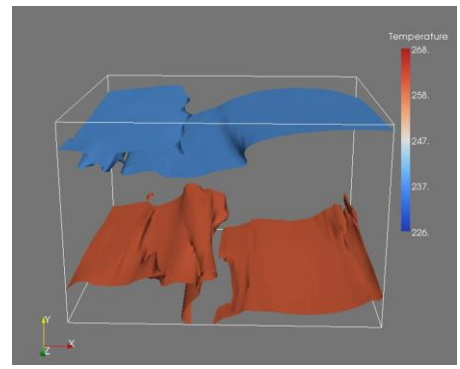
Desired properties for color scales (*escalas ou gamas de cores*):

Given a sequence of numerical values (v1≤v2 ≤… ≤vn) represented by colors (c1, c2,…, cn) we can consider the following properties:

1) Order – the colors chosen to represent the numerical values must be perceived as having the same order as them, i.e., if the values are ordered,the colors chosen to represent them must also seem ordered.

Ex1: the representation of a temperature scale by using the notions of **cold and warm colors** and their proportional mixtures inorder to obtain a scale from cold to hot temperatures.

Ex2: vegetation density – green shades



Kelvin units: K = °C + 273,15

Special case of **nominal data:**

It is not ordered, so there should be no perceptual ordering in the representation.

Desired properties for color scales:

2) Uniformity and representative distance —The color representation of two values should convey the distance between them.

Moreover

- clearly separated values must be represented by distinguishable colors
- close values must be represented by colors perceived to be closer

3) Boundaries (*fronteiras*) – If there are no boundaries on the represented numerical data the color scale should not create this effect, i.e., the color scale must be able to represent **continuous scales**.

Color scales to represent univariate data

It involves only one variable

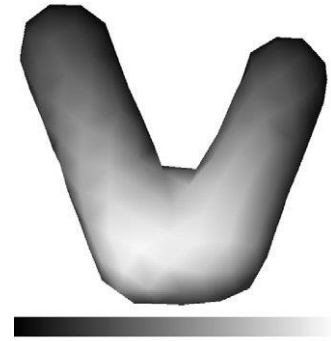
- When using a color scale to represent **univariate data**, each color represents a single scalar value
- It can be a **continuous color scale**, if color varies along the scale in such a way that adjacent colors are similar to one another
- Otherwise, it is a **discontinuous color scale**

The most common color scales to represent univariate data
Grey scale (*escala de cinzentos*)

This color scale **maps scalars** to **brightness**.
It consists in a variation from black to white, with black representing, in general, the lowest value and white the highest.

Advantage: easy to perceive ordering

Disadvantage: it displays a low contrast between the different colors, which limits its use in quantitative tasks.

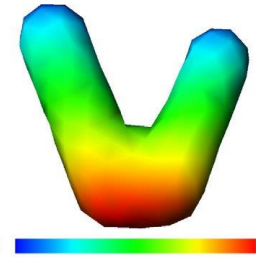


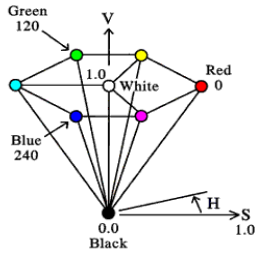
(Silva, 2011)

Note: How to translate brightness into Portuguese?
Brilho é quantidade de luz emitida pela superfície de um objecto luminoso
Luminosidade é a intensidade da luz reflectida pela superfície dos objectos (Brisson Lopes, 2006)

The most common color scales to represent univariate data
Rainbow scale (*escala* do arco-íris)

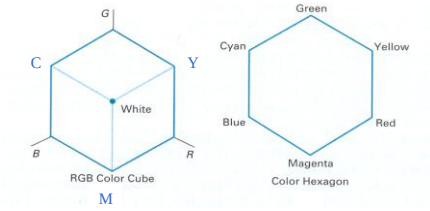
It consists in a color path along the different colors of the rainbow, built by varying hue while keeping saturation and contrast at fixed values

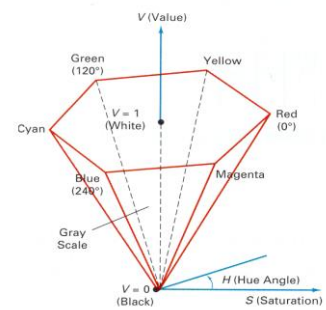




In HSV (hue, saturation and value) color model, the rainbow scale consists in a complete rotation around the value (V) axis.

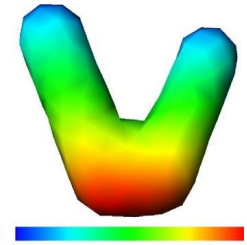
HSV Model
(Hue Saturation Value)





The hexagon corresponds to the projection of the RGB cube seen along the main diagonal

Rainbow scale (*escala* do arco-íris)



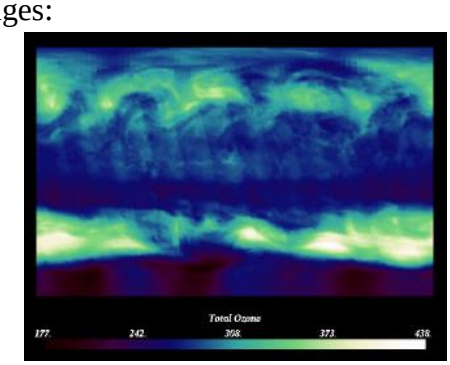
- Advantages: we can easily distinguish distint values associated with different colors
- Rainbow scale presents several **problems**

- To some users it might not present an intuitive ordering unless they are familiar with the color progression (light spectrum).
- This color scale goes from red to violet.
  Since these colors are quite similar, both extremes will get visually close.
  To avoid this problem the color scale is usually cut at blue
- Yellow is present half way through the color scale. Since yellow has an highlighting effect being perceived as brighter than the other colors, if one is interested in depicting extreme values the middle values might interfere.
- Yellow has the smallest number of perceived saturation steps. Therefore, users find it harder to distinguish small saturation variations for yellow than, for example, for blue

Redundant color scales

Using multiple display parameters, for instance, hue and brightness, to represent data may have several advantages:

- It might help in dealing with situations where one of the parameters becomes ambiguous (e.g., due to a visual deficiency) and is compensated by the others.
- It might also allow a better distinction between values by reinforcing their visual differences.



Redundant hue/lightness scale [Rheingans00]

Double-ended color scales (diverging color scales)

Such color scales are created by joining two monotonically increasing scales at a common end point.

For example, one can join

a scale from grey to red

and a scale from grey to green

building a scale from red to grey to green

With such color scales it is possible to visually represent high, low and middle values clearly, since they exhibit three distinct groups of colors

It can be useful to indicate on which side of the zero a region lies.

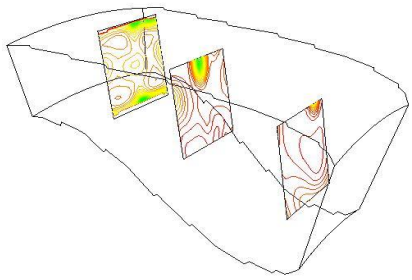
Another example of a diverging color scale combines blue, white in the middle and red

Double-ended color scales (diverging color scales)

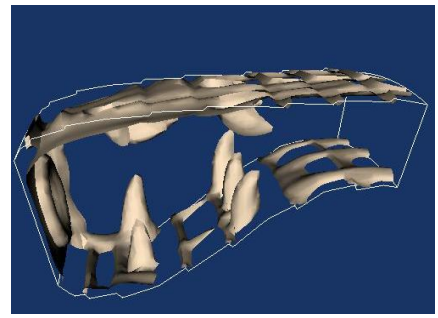
Diverging color scale with
cmin = green
cmax = red
cmid = brightyellow

[Telea2014]

Contouring (*Traçado de isolinhas e isosuperfícies*)



Isolines (*isolinhas*)



Isosurfaces (*Isosuperfícies*)

Contouring (*Traçado de isolinhas e isosuperfícies*)

Isolines and isosurfaces, respectively, in **2D** and in **3D**, construct boundaries between regions. They are contours that connect points associated with a given scalar value.

- First a scalar value is selected
- Then, to generate the contour, some form of interpolation must be used

Because we have scalars values at a finite set of points in the dataset and the contour lines may lie between the point values

The **contour points** are generated by linear interpolation along the edges

Contouring (*Traçado de isolinhas e isosuperfícies*)

There are two approaches to connect the contour points

- ***edge-tracking***
- ***marching-squares* (*marching-cubes* em 3D).**

Contouring (*Traçado de isolinhas e isosuperfícies*)

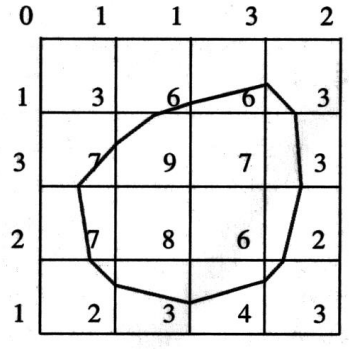
Edge-tracking (in the case of isolines):

- Detect an edge intersection (ie, the contour passes through an edge)
- Then "tracks" this contour as it moves across cell boundaries (if a contour edge enters a cell, it must exit a cell as well)
- The contour is tracked until it closes back on itself or exits a dataset boundary
- If it is known that only a single contour exists, the process stops
  Otherwise, every edge in the dataset must be checked to see whether other contour lines exist

This approach is more complex when working in 3D to build isosurfaces

Contouring (*Traçado de isolinhas e isosuperfícies*)

Example of isoline with contour value 5

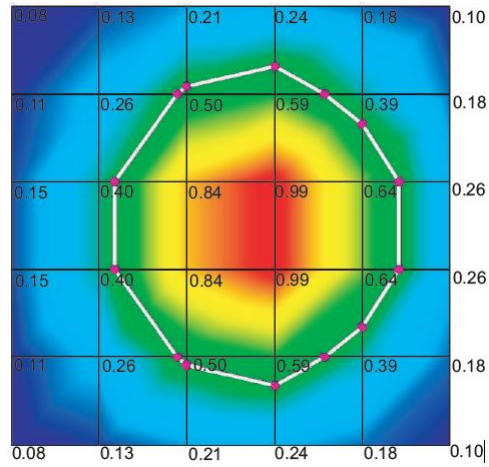


Contouring and color mapping

Isoline for the scalar value $v = 0.48$.

The numbers in the figure indicate scalar values at the grid vertices.

[Telea2014]



Contouring (*Traçado de isolinhas e isosuperfícies*)

The **marching-squares algorithm** uses a "**divide and conquer" technique** treating each cell independently and, at the end, connecting segments obtained for each cell

The basic assumption of this technique is that a **contour** can **only pass through a cell** in a **finite number of ways**, taking into account the topology of the grid

For instance

– For 2D grids, if a cell has 4 vertices and each vertex can be either inside or outside the contour, there are

$2^4 = 16$

Number of different sequences of 4 bit values (0 or 1)

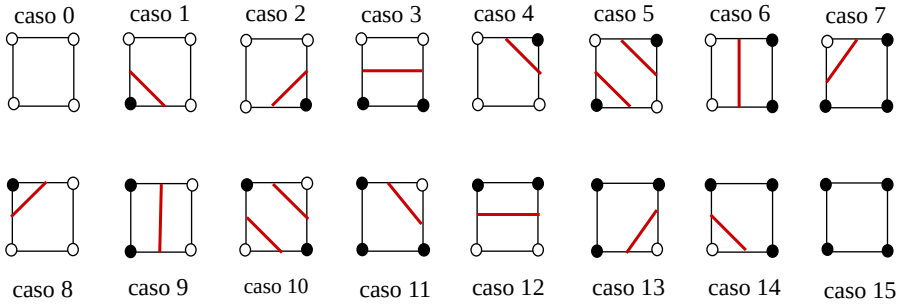
possible ways that the contour passes through the cell

– For 3D grids, if a cell has 8 vertices, there are $2^8 = 256$ possible ways.

Contouring (*Traçado de isolinhas e isosuperfícies*)

Next figure shows the 16 combinations for a square cell

Dark vertices indicate that the scalar value is **above** contour value



Marching squares algorithm

- For each cell
  - Calculate the inside /outside state of each vertex of the cell
  - Create an index by storing the binary state of each vertex in a separate bit
  - Use the vertex to look up the topological state of the cell in a case table
  - Calculate the contour location (by interpolation) for each edge in the case table

- This procedure will construct independent geometric primitives for each cell
  It is necessary to connect all the segments of the isoline

- As the cells are treated independently, at the cell boundaries duplicate vertices and edges may be created.
  So interpolation along each edge should be done in the same direction. Otherwise, numerical round-off will generate, on a edge, intersection points not coincident

Comparison of algorithms
edge-tracking vs *marching squares*

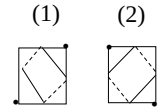
Marching squares algorithm

- Easy to implement
  - This is important when we extend the technique into 3D, where isosurface tracking becomes much more difficult.

- However it creates disconnected line segments and points
  - It requires a merging operation that performs extra computation

Contouring (*Traçado de isolinhas e isosuperfícies*) contouring ambiguity

Cases (1) and (2) are ambiguous
(both for marching squares and edge-tracking)



A contour ambiguity arises when adjacent edge points are in different states, but diagonal vertices are in the same state

In both cases there can be both a "valley" (*vale*) between the two vertices (corresponding to draw solid lines) as a "ridge" (*crista*) (corresponds to draw the dashed lines)

Any of the possibilities may be chosen, since in the absence of accurate information, both hypotheses are valid

Contouring (*Traçado de isolinhas e isosuperfícies*) contouring ambiguity

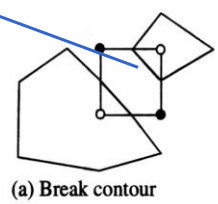
Depending on the choice to solve ambiguities (1) and (2),

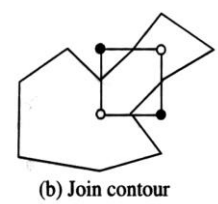
"ridge" (*crista*) or "valley" (*vale*) ,

- The contour may break (a)
- Or the contour may extend (b)

Solution of case (2)

"ridge" (*crista*)- dark vertices correspond to scalar values **above** the contour value

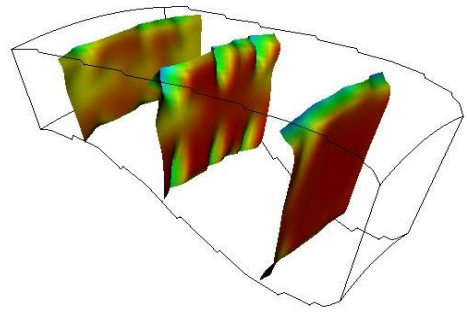

(a) Break contour


(b) Join contour

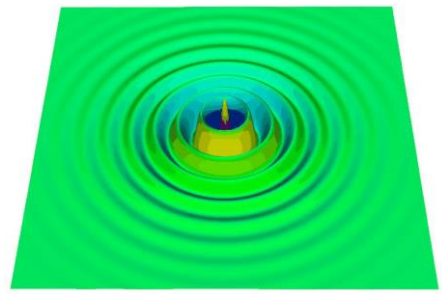
Carpet Plots (height plots)
(*Deformação de Superfícies*)

- A carpet plot is created by warping a 2D surface in the direction of the surface normal
- The amount of warping is controlled by the scalar value, possibly in combination with a scale factor
- It is common to use also color mapping on the surface to visualize the same or another variable

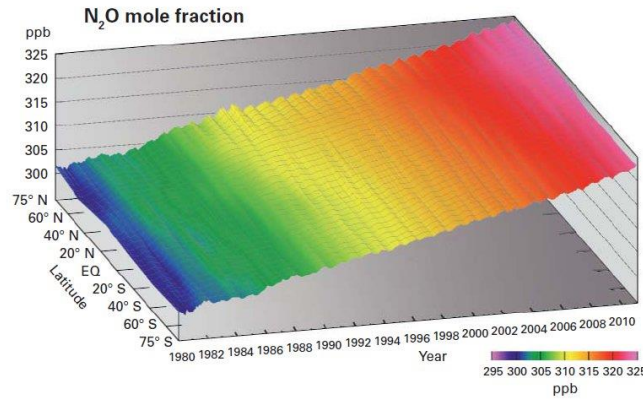
Carpet Plots (*Deformação de Superfícies*)



Carpet Plots (*Deformação de Superfícies*)

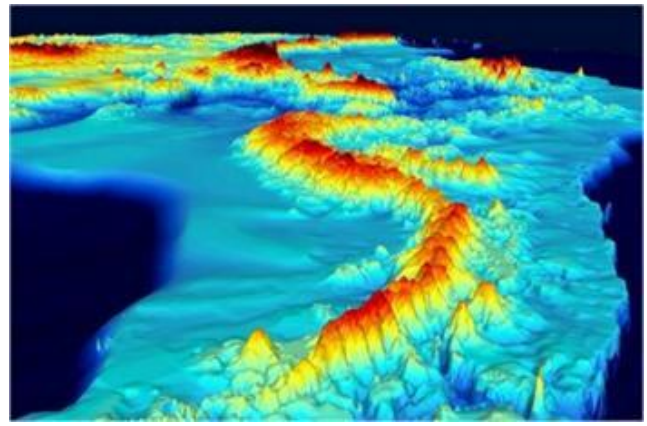


Carpet Plots (*Deformação de Superfícies*)



Zonally averaged nitrous oxide (N_2O) abundance from WMO/GAW air sampling sites is plotted as a function of latitude from 1980 to 2010. Nitrous oxide is now the third most important contributor to radiative forcing of long-lived greenhouse gases (World Meteorological Organization, 21-11-2011)

Carpet Plots (*Deformação de Superfícies*)



Projecto BEDMAP2 - Antarctic relief hidden by the ice (2011)

Scalar generation

- Often our data is not single-valued (i.e., scalar) or it may be a mathematical or other complex relationship.
- So we have to use algorithms to convert data into a form that we can visualize

- Examples:
  - Using the z component as a scalar value (terrain data)
  - Computing vector magnitude
  - Determining the distance between points

Probing (*Sondagem*)

- Probing (*sondagem*) obtains dataset attributes by sampling one dataset (the input) with a set of points (a probe)

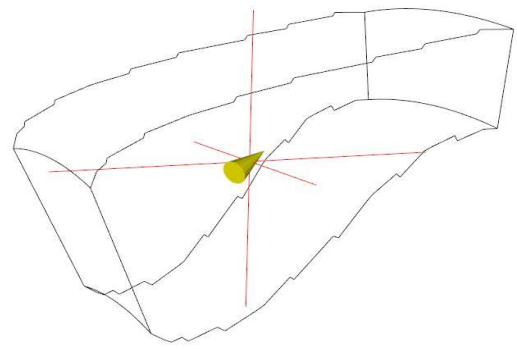
- **Probing** is also called **resampling** as we can obtain a new dataset (the output) with the topological and geometric structure of the probe dataset, and point attributes interpolated from the input dataset.

Probing (*Sondagem*)

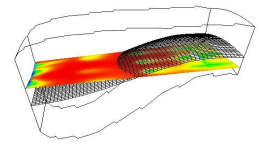
- Probing is not limited to scalar data

- The probe is represented according with the variable that we are studying (ex, a numerical value, a glyph)

- Probing can be controlled manually, for example, using the cursor or the arrow keys or using a scanning algorithm.



Probing (*Sondagem*)



- We can consider that the use of colored or warped cutting planes is a probing where the results, instead of being shown one by one, are accumulated and presented in an integrated manner.

- The probe is a sampling, so it can be subjected to
  - under-sampling (drop some data): data in a region can miss important high-frequency information or localized data variations, because the sample do not include all the data.
  - or oversampling (add new values by interpolation): while not creating error, can give false confidence in the accuracy of the data

Referências

- João Cunha, Guiões VI, DI-FCUL
- Silva, S., Santos, B. S., Madeira, J., "Using Color in Visualization: a Survey", Computers & Graphics, 35(2011), pp 320-333
- [Rheingans00] P.Rheingans,"Task-based color scale design" in Proc.SPIE, 28thAIPR Workshop: 3DVisualization for Data Exploration and Decision Making, vol.3905, pp.35–43,2000.
- Brisson Lopes, Cor e Luz, notas da disciplina de Computação Gráfica, IST,2006/2007
- [Telea2014] A. Telea, Data Visualization, Principles and Practise, 2nd edition
- "The Visualization Toolkit – an Object-Oriented Approach to 3D Graphics", W. Schroeder, K. Martin, B. Lorensen, Prentice Hall PTR,
  2ª edição, 1998,4ª edição, 2006
- "The VTK User's Guide", Kitware, Inc., 2006
- http://www.vtk.org