Basic Mapping Techniques

Mapping is one of the most important parts of visualization. In this chapter the most popular Mapping Techniques are shown, which are used to represent data.

In this part you will learn to handle 3 dimensional objects with matrices, designing 3D Scenes using scene graphs, Projections, z-buffer Algorithm, lighting techniques...

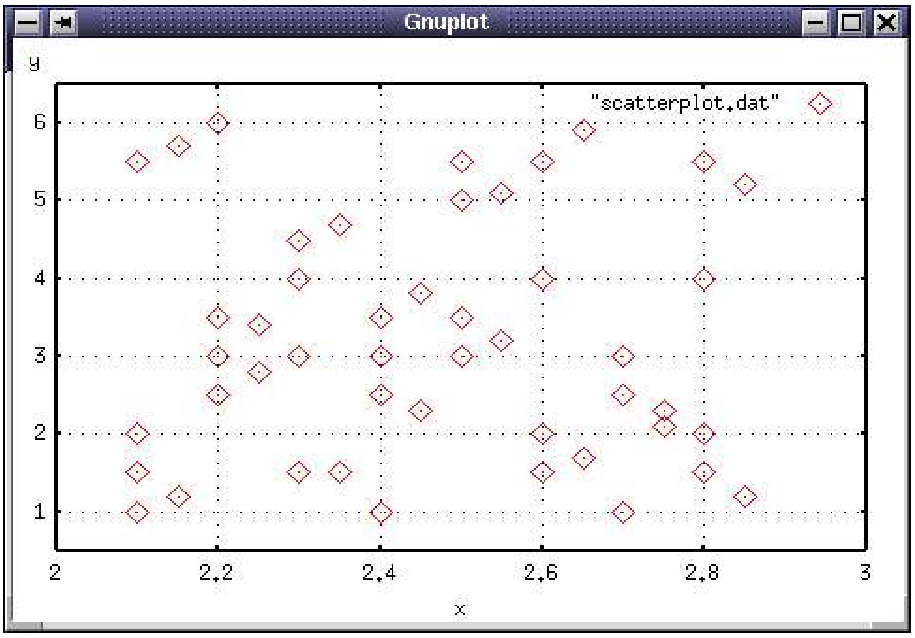
1 Diagram Techniques

In this part you get an insight of the most common mapping techniques like scattered plots, Bar and Pie graphs, line graphs, and 3D height fields.

Typically a data set of information is given in an complex array of numbers. It is aspired to vizualize a set of data in a way, to be easily understood by a human. Dependant of the data structure, it is possible to choose between several different Techniques.

1.1 Scatter plots

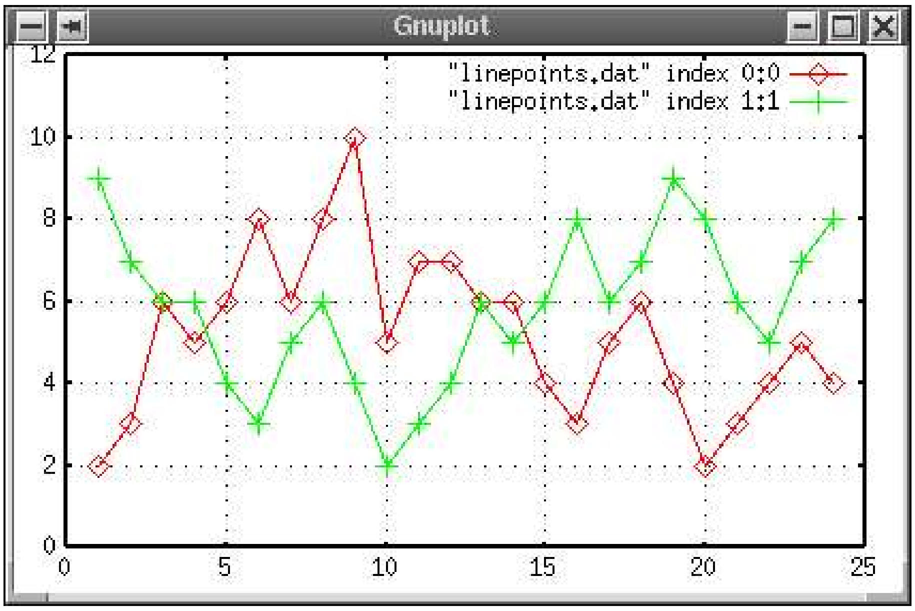
The easiest and most intuitive way to visualize complex, quantitative data is a diagram, based on a two-dimensional coordinate system. One possible option is to use **scatter plots** to illustrate correlations of discrete data.



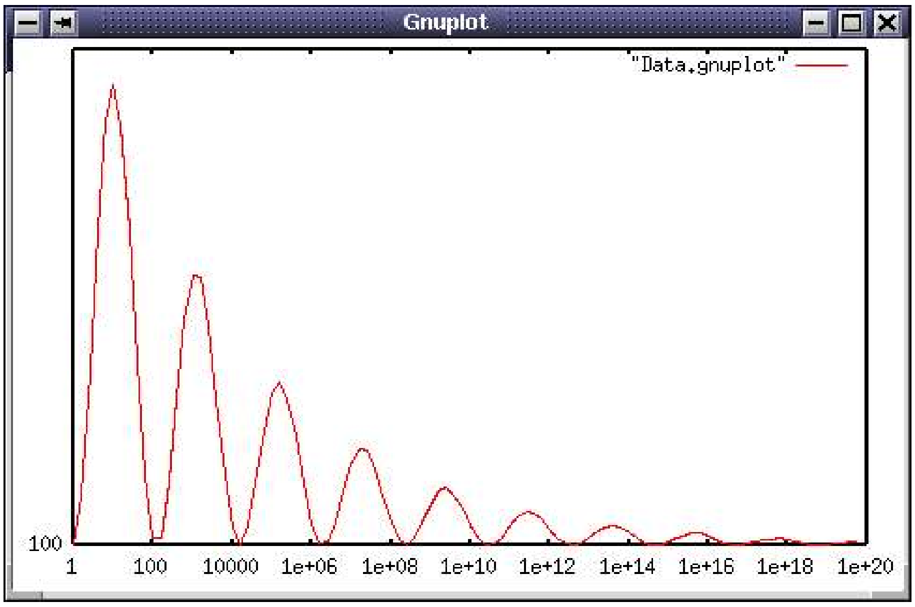
This picture shows a scatter plot generated with gnuplot. The x- and y-axis are scaled, a simple grid and bigger points are used to improve recoginization of important parts.

1.2 Line graph

A simple **line graph** or a line connection between discrete points can be used to display a continuous function, that for example changes its value over time.



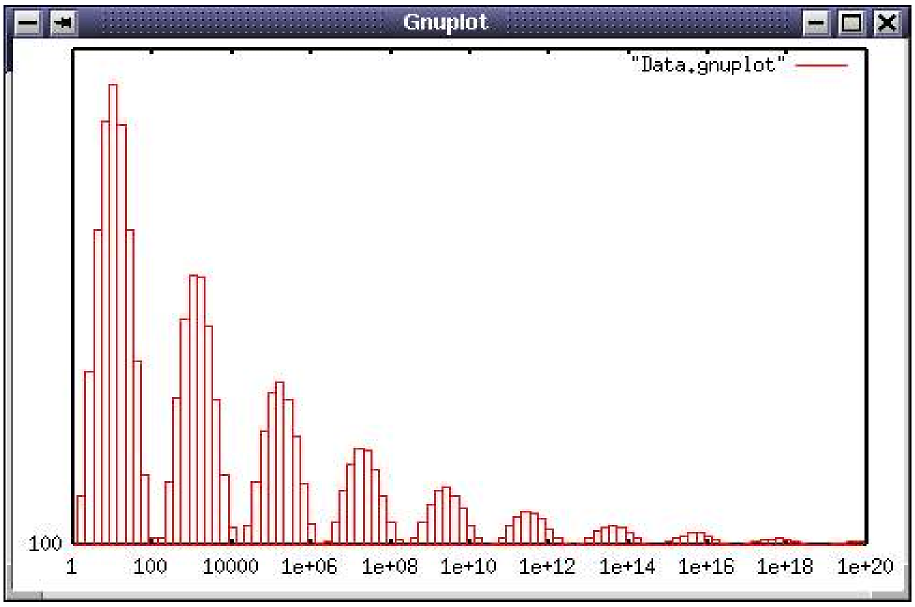
This Graphic illustrates a point-graph with connections between the single points.



This plot represents a continuous line graph.

1.3 Bar graph

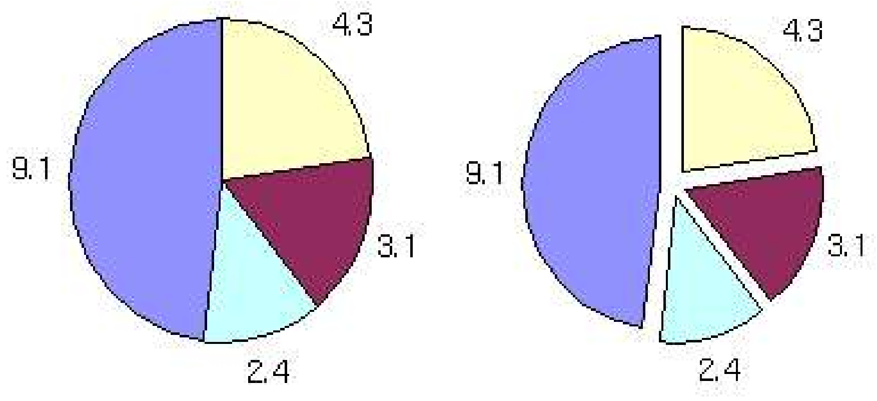
If a set of discrete data is given, in some cases it is better to visualize it with a simple **bar graph**.
In this kind of diagram the domain is a discrete, independent variable, labeled with nominal, ordinal or quantitative value. Whereas the range is a quantitative dependent variable, naturally the data set.



Repersentation of the same data set with a discrete bar graph.

1.4 Pie graph

To visualise quantitative data that adds up to a fixed number, or a condition that is calculated in percent, the use of a **pie graph** is favorably.



Pie charts for the representation of data that adds up to a fixed number. The size of a piece is propoprtional to the percentage of its portion of the whole pie.

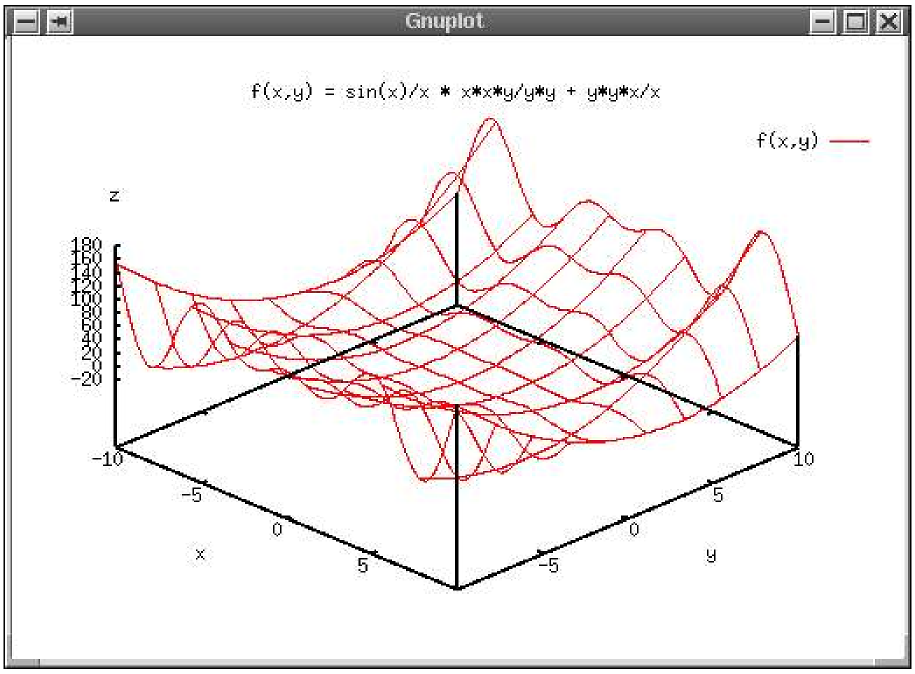
2 Function Plots and Height Fields

Visualization of 1D or 2D scalar fields.
1D scalar field: $\Omega \subset \mathbb{R} \rightarrow \mathbb{R}$
2D scalar field: $\Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}$

A **Heightfield** is a two-dimensional array in X and Y direction. The array values represent the Z value of a surface at each point. Heightfields are used to model any kind of surface, e.g. Landscapes.
The surface is smoothly interpolated between the defining points using a linear filter.

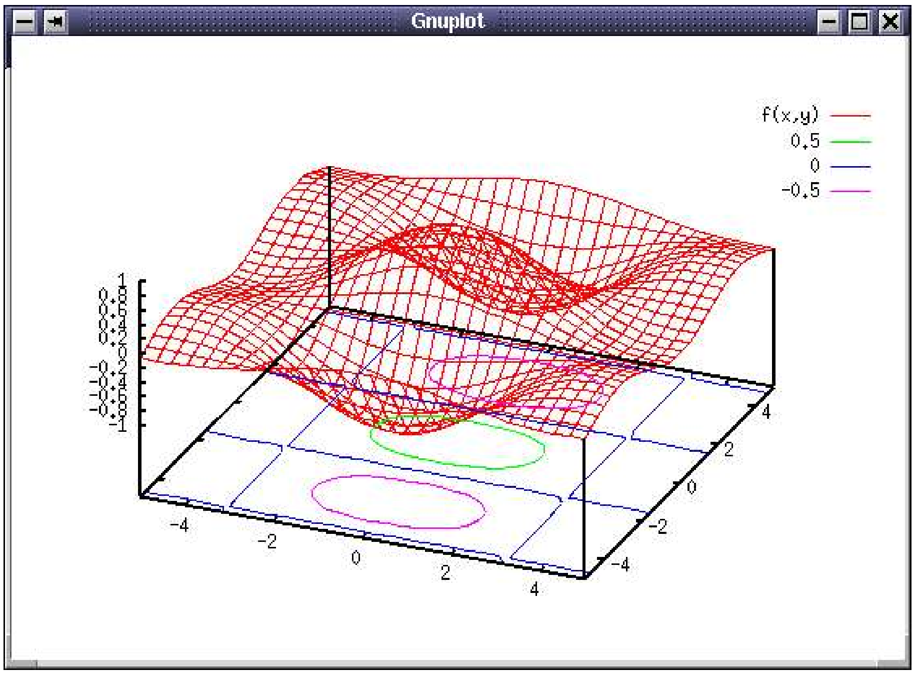
There are three different types for the representation of surfaces:
- Wireframe
- Hidden Lines
- Shaded surface

Wireframe representation:
The wireframe representation requires the specification of viewing parameters. Every edge of the grid is drawn, even if it is hidden by a face lying in front. For the viewer the result appears as a transparent surface.



This graphic shows a 3D height field. The given plot represents the function

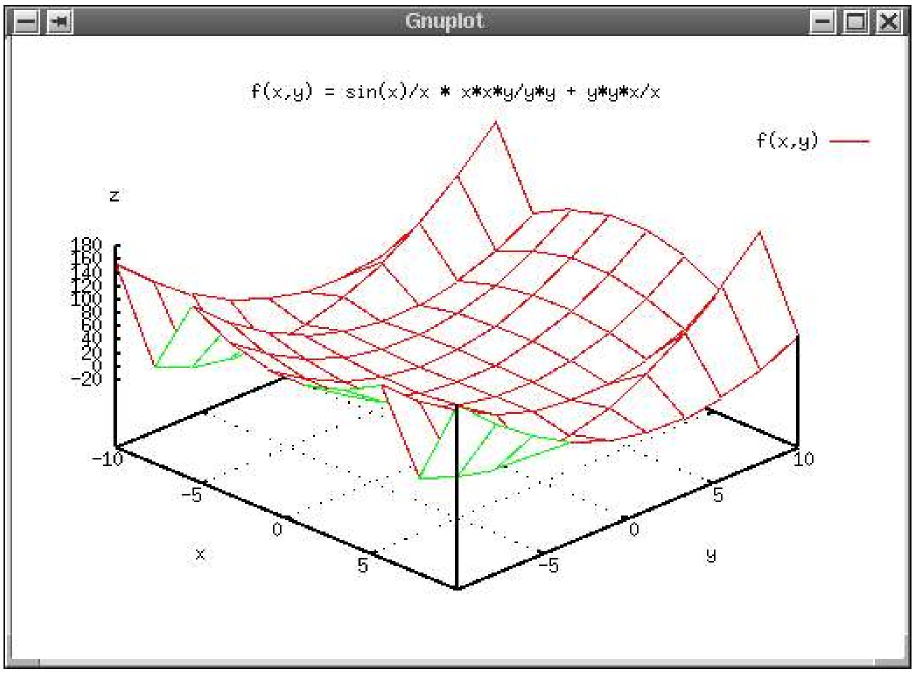
$$f(x,y)=\frac{\sin(x)}{x}+\frac{x^2 y}{y^2}+\frac{y^2 x}{x}$$



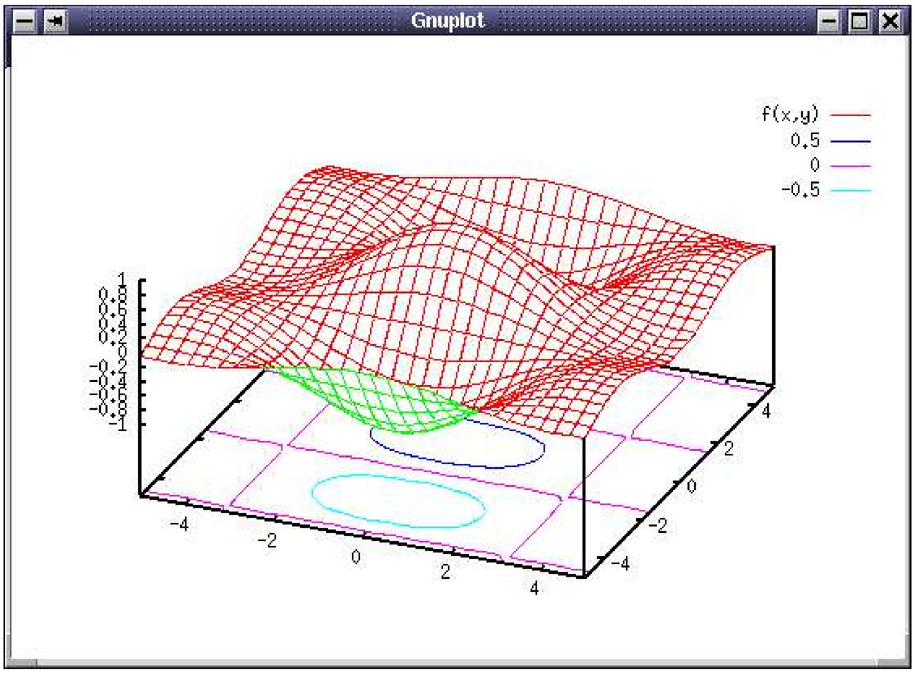
Plot of the function $f(x,y)=\sin(x)\cdot\cos(y)$ *in wireframe representation.*

$$f(x,y)=\frac{\sin(x)}{x}+\frac{x^2 y}{y^2}+\frac{y^2 x}{x}$$

Hidden lines representation:
Contrary to a wireframe the hidden lines representation removes all edges that belong to hidden faces.
Due to this occlusion it gives the viewer a better spatial orientation.



Plot of the function $f(x,y)=\frac{\sin(x)}{x}+\frac{x^2 y}{y^2}+\frac{y^2 x}{x}$ *with hidden lines.*

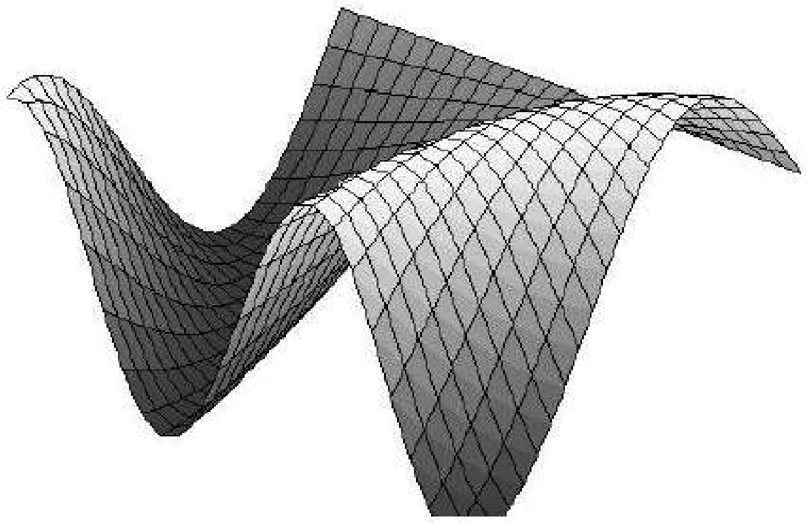


Plot of the function $f(x,y)=\sin(x)\cdot\cos(y)$ *in hidden lines representation.*

$$f(x,y)=\frac{\sin(x)}{x}+\frac{x^2 y}{y^2}+\frac{y^2 x}{x}$$

$$f(x,y)=\sin(x)*\cos(y)$$

Shaded surface representation:
The shaded surface is an extended version of the hidden lines representation. In addition to removed hidden faces it shades every polygon dependant on the influence of incident light. Therefore it requires the specification of a lighting/shading model.

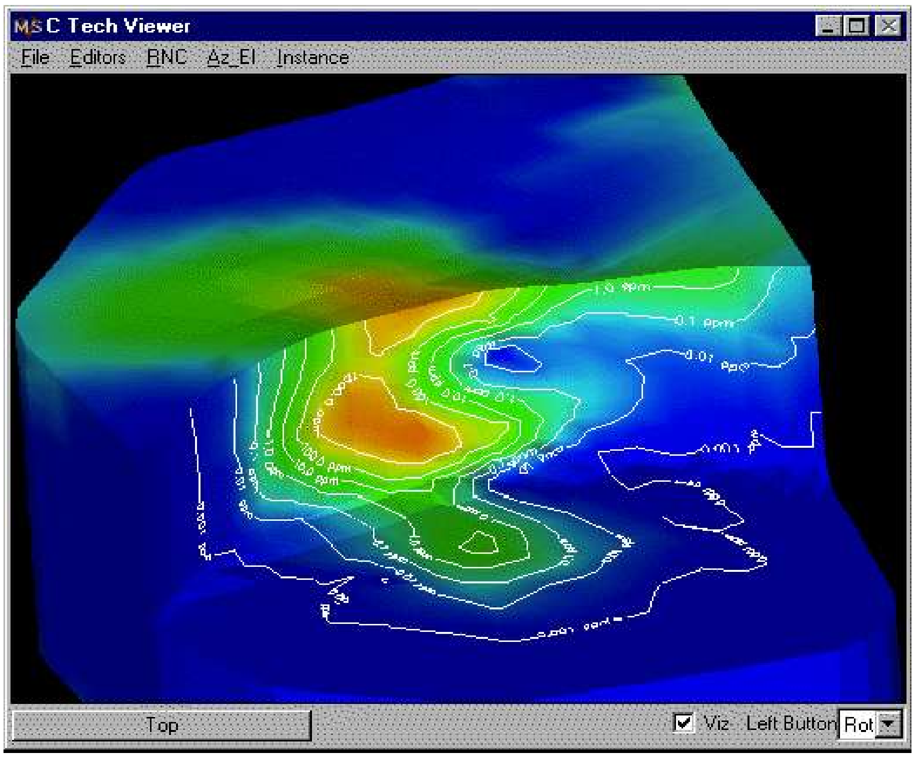


This picture shows a shaded surface with the light source in the top right corner.

3 Isolines

3.1 Isolines

Isolines are used for the visualization of 2D scalar fields, given as a scalar function $f: x \in \mathrm{IR}$ and a scalar value $c \in \mathrm{IR}$. Isolines consists of points $\{(x,y) | f(x,y) \in c\}$ which result in contour lines if connected. If $f()$ is differentiable and $grad(f) \neq 0$, then isolines are curves.



Isolines are often used for weather charts, to illustrate flow paths of high and low pressure areas.

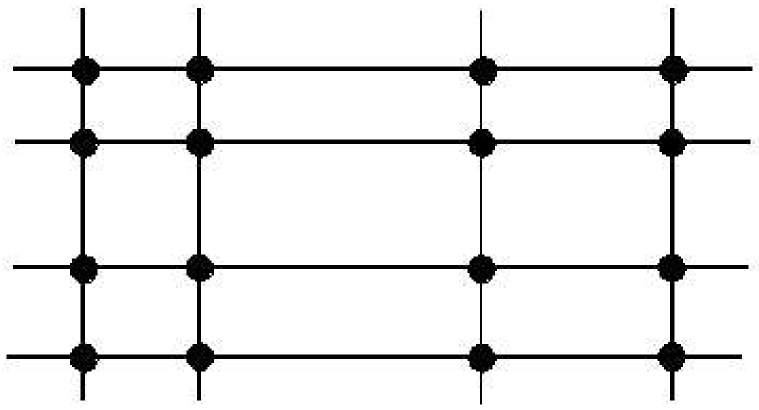
Isolines are often used for maps, weather charts, thermal diagrams or any other kind of 2 or 3 dimensional data representation.
Isolines can be implemented with a simple pixel by pixel contouring algorithm. It is a straightforward approach, scanning all pixels for equivalence with a given isovalue.

Problem: Isolines can be missed if the gradient of $f()$ is too large (despite range ϵ)

```
Input:
         $f:(1,\dots,x_{max})\times(1,\dots,y_{max})\rightarrow\mathbb{R}$
         Isovalues $I_1,\dots,I_n$ and isocolors $c_1,\dots,c_n$
Algorithm:
         for all $(x,y)\in(1,\dots,x_{max})\times(1,\dots,y_{max})$ do
           for all $k\in\{1,\dots,n\}$ do
             if $|f(x,y)-I_k|<\epsilon$ then
                  draw $(x,y,c_k)$
```

3.2 Marching squares

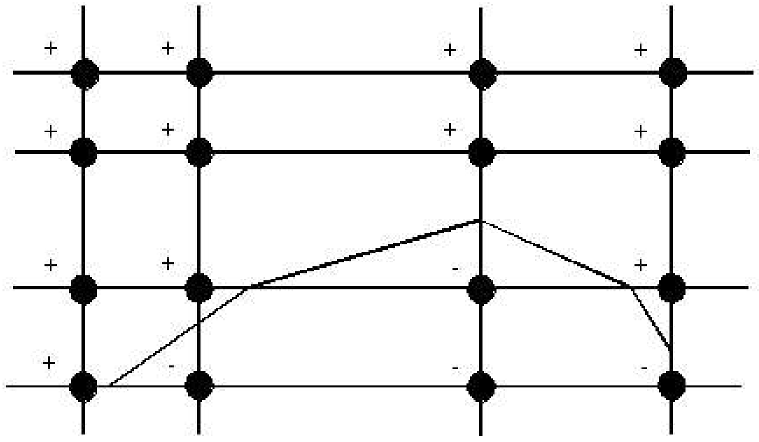
The **Marching squares** algorithm is used to compute the representation of a scalar function on a rectilinear grid. Due to the discrete scalar value at each vertex of a cell, interpolation of points within cells is necessary. The marching squares algorithm is the 2D equivalent to the marching cubes algorithm, described in chapter 5. It uses a devide and conquer approach by considering cells independently of each other.



Rectilinear grid with scalar values represented as points.

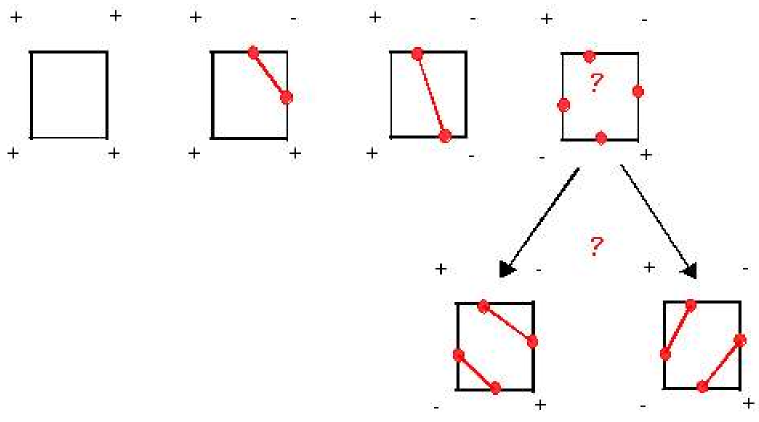
Basic idea of the algorithm:

- Which cells will be intersected ?
    - Initially mark all vertices by + or - , depending on the condition $f_{ij} \geq c, f_{ij} < c$
    - c is the user defined isofvalue.
- No isoline passes through cells (rectangles) which have the same algebraic sign at all four vertices.
    - So we only have to determine the edges with different signs at their adjacent vertices.



Rectilinear grid with isolines intersecting cells, depending on the algebraic signs of the vertices.

- Concerning symmetry characteristics there are only 4 different cases of combinations of signs.
- Symmetry characteristics are: rotation, reflection, change + and -
- Compute intersections between isoline and cell edge, based on linear interpolation along the cell edges.



4 different cases of intersecting isolines, two of them are ambiguous and must be solved somehow.

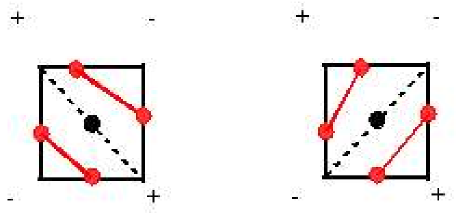
- Cases of ambiguity can be distinguished by a decider.

Mid point decider:

- Determine the function value in the center by bilinear interpolation

$$f_{center} = \frac{1}{4}(f_{i,j} + f_{i+1,j} + f_{i,j+1} + f_{i+1,j+1})$$

- If $f_{center} < c$ we choose the right case, otherwise we choose the left case.



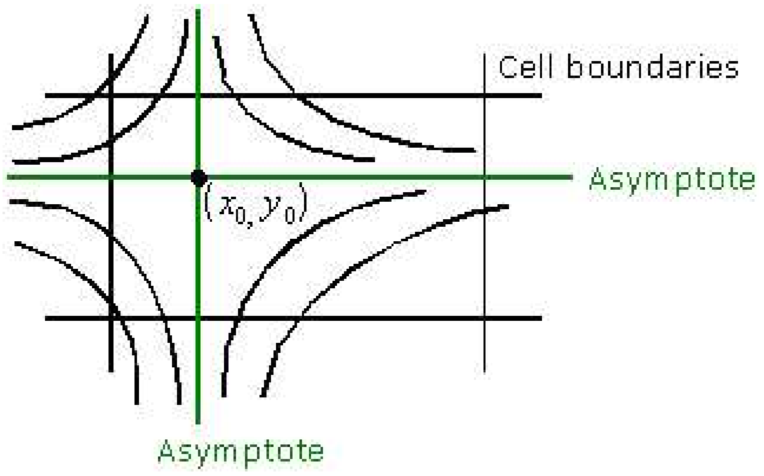
Mid point decider can be used to solve the ambiguity problem.

- This is not always the correct solution.

Asymptotic decider:

The Asymptotic Decider resolves the ambiguity in Marching squares.

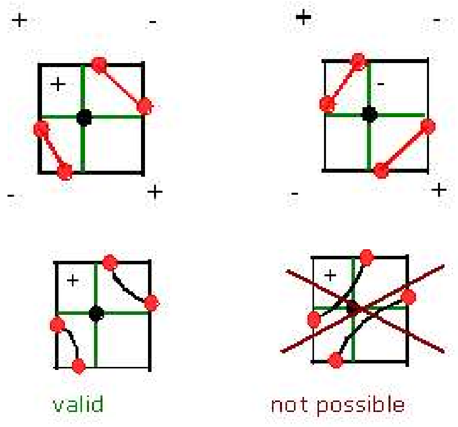
- Consider the bilinear interpolant within a cell.
- The true isolines within a cell are hyperbolas.



The asymptotic decider can be used to solve ambiguity more reliable.

- Interpolate the functon bilinearly
  $f(x,y)=f_{i,j}(1-x)(1-y)+f_{i+1,j}x(1-y)+f_{i,j+1}(1-x)y+f_{i+1,j+1}xy$
- Transform $f()$ to
  $f(x,y)=\eta(x-x_0)(y-y_0)+\gamma$
- γ is the function value in the intersection point of the asymptotes.

- If $\gamma \le c$ we choose the right case, otherwise we choose the left case.



By interpolating with the asymptotic decider the two possibilities of intersecting isolines resulting from the ambiguity can be solved.

Dependent on the sign which results from the computed formula the arrangement of the lines is clearly given.

- Explicit transformation $f()$ to
  $f(x,y)=\eta(x-x_0)(y-y_0)+\gamma$ can be avoided
- Idea: investigate the order of intersection points either along x or y axis.
- Build pairs of first two and last two intersections.

Cell order approach for marching squares:
- Traverse all cells of the grid.
- Apply marching squares technique to each cell.

Disadvantage
- Every vertex (of the isoline) and every edge in the grid is processed twice.
- The output is just a collection of pieces of isolines which have to be postprocessed to

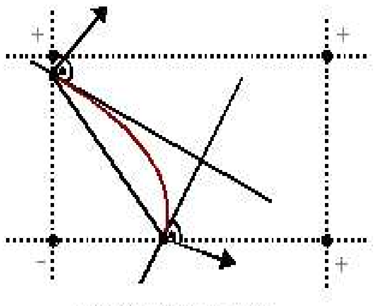
get (closed) polylines.

Contour tracing approach:
- Start at a seed point of the isoline
- Move to the adjacent cell which is entered by the isoline.
- Trace isoline until

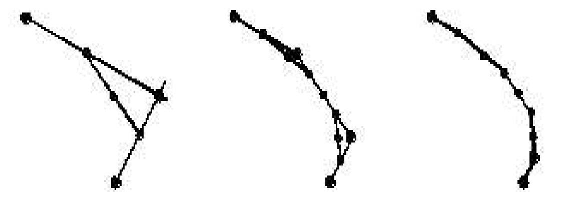
- Bounds of the domain are reached or
- Isoline is closed.

Problems:

1. How to find seed points efficiently:
   - In a preprocessing step, mark all cells which have a sign change.
   - Remove marker from cells which are traversed during contour tracing (unless there are 4 intersection edges !).
2. How to smooth isolines:
   - Evaluate the gradient at vertices by central differences.
   - Estimate tangents at the intersection points by linear interpolation (Note that the gradient is perpendicular to the isoline !).
   - Draw a parabolic arc which is tangentail to the estimated tangents at the intersections:
     - Quadratic Bezier curve
     - Approximmation with 2-3 subdivision steps is sufficient.



Quadratic Bezier



Quadratic Bezier approximation with subdivision

Example of a Quadratic Bezier approximation in three steps.

4 Color Coding

Colors are an essential component of visualization. Most visualization techniques contain a step in which different data values are mapped to different colors to make the range of the data visible. Since the mapping of data values to colors involves color coding, we will describe this in some detail. Issues of the visualization prozess are what kind of data can be color-coded and what kind of information can be efficiently visualized?

Color coding extends over a broad application scope. Some examples are:

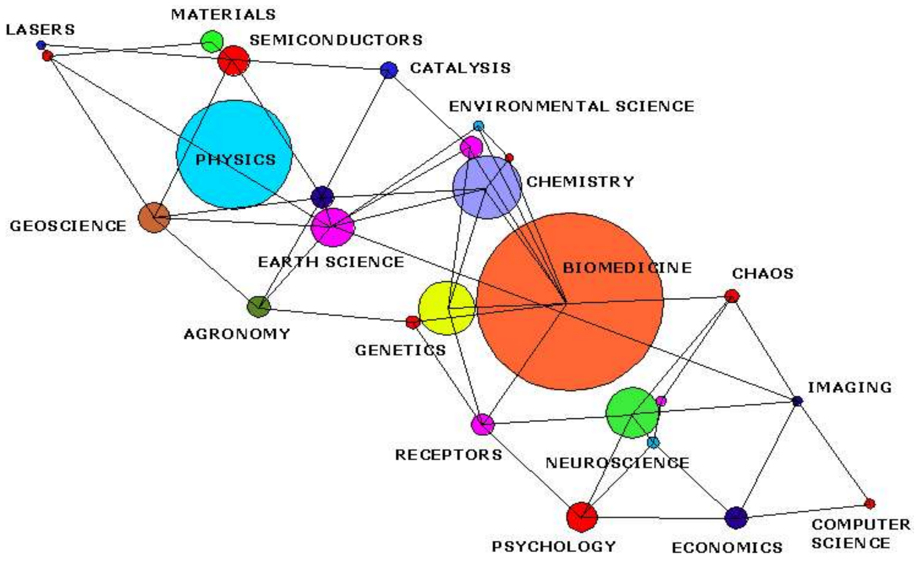
- Provide information coding
- Designate or emphasize a specific target in a crowded display
- Provide a sense of realism or virtual realism
- Provide warning signals or signify low probability events
- Group, categorize, and chunk information
- Convey emotional content
- Provide an aesthetically pleasing display

Problems:

- Distract the user when inadequately used
- Dependent on viewing and stimulus conditions
- Ineffective for color deficient individuals

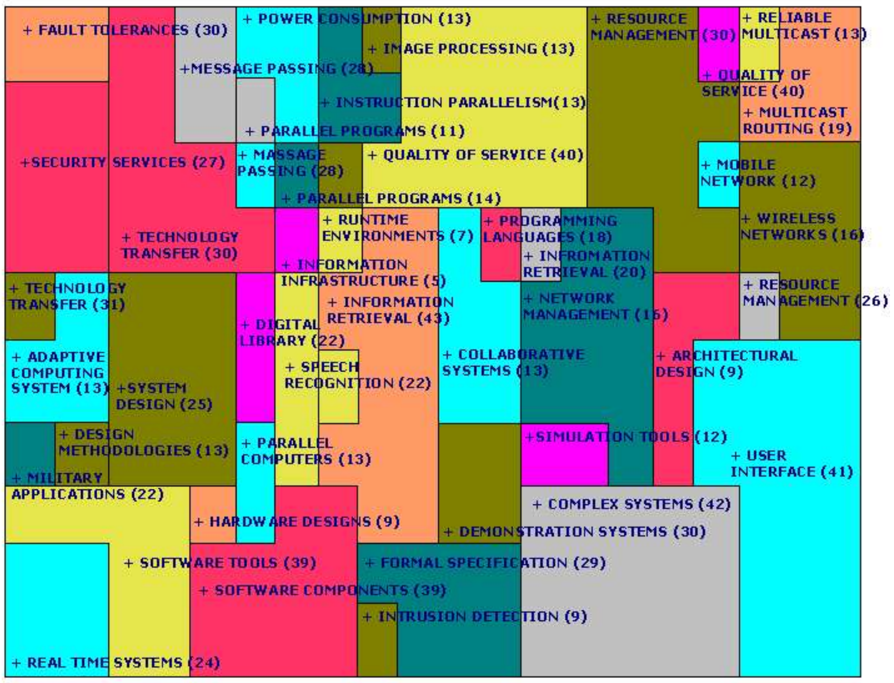
- Results in information overload
- Unintentionally conflict with cultural conventions
- Cause unintended visual effects and discomfort

Nominal data color coding:
By **nominal color coding** one understands the aesthetic and functional use of color to impart qualitative information and psychological aspects in graphical environments.
- Assignment of colors needs to be well distinguished.
- According to standard is the use of arround 8 different basis colors.
- Localisation of data.



Co-citation analysis displayed with a nominal data diagram.

Fields of research are shown as colored circles, relations are marked with lines.

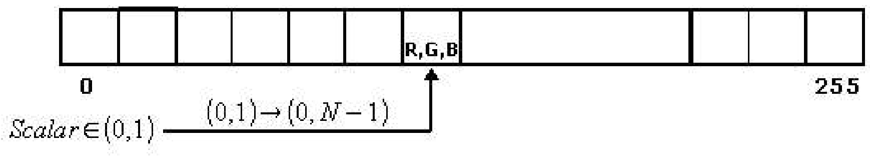


Another example of a nominal data diagram. Fields of research are given as different colored and sized quadratic regions.

Ordinal and quantitative data:

- The order of data should be represented by the order of colors
- Stand for psychological aspects
- $x_1 < x_2 < \ldots < x_n \rightarrow E(c_1) < E(c_2) < \ldots < E(c_n)$

Color coding for scalar data:
- Assign to each scalar value a different color value
- Assign via transfer function T
  $T: scalarvalue \rightarrow colorvalue$
- Code color values into a color lookup table
  default range of a color lookup table is 256



The simple structure of a color lookup table is given by a array with 256 fields.

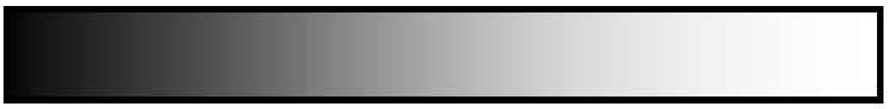
Pre-shading vs. post-shading
- **Pre-shading**
  - Assign color value to original function value (e.g. at vertices of a cell)
  - Interpolate between color values (within a cell)
- **Post-shading**
  - Inerpolate between scalar values (within a cell)
  - Assign color values to interpolated scalar values

Linear transfer function for color coding
- Specify color for f_{min} and for f_{max}
  - $(R_{min}, G_{min}, B_{min})$ and $(R_{max}, G_{max}, B_{max})$
- Linearly interpolate between them (same idea as inverse distance weighting):

$$f \rightarrow \frac{f - f_{min}}{f_{max} - f_{min}} (R_{max}, G_{max}, B_{max}) + \frac{f_{max} - f}{f_{max} - f_{min}} (R_{min}, G_{min}, B_{min})$$

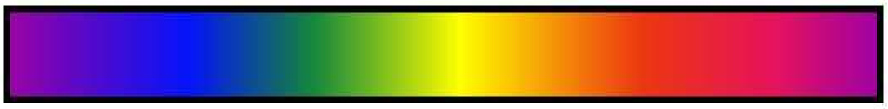
- Different color spaces lead to different interpolation functions
- In order to visualise (enhance/suppress) specific details, non-linear color lookup tables are needed

Gray scale color table
- Intuitive Ordering



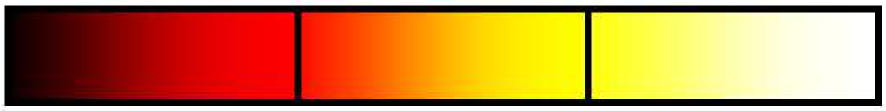
Gray scale color table.

Rainbow color table
- Less intuitive
- HSV color model



Rainbow color table.

Temperature color table
- black – red – yellow - white



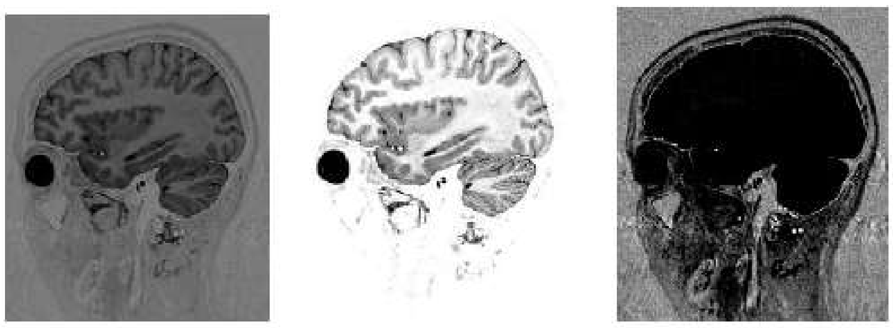
Temperature color table.

- Bivariate and trivariate color tables are not very useful:
    - No intuitive ordering
    - Colors are hard to distinuish

- There exist many more color tables for specific applications
- Design of good color tables depends on psychological / perceptional issues

- Frequently transfer functions are specified interactive, in order to extract important characteristics

  Example:
  - Special color table to visualize the brain tissue
  - Special color table to visualize the bone structure



Original, Brain, Tissue.

5 Glyphs and Icons

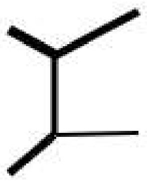
Glyphs belong to the family of Iconography. They are appointed to assign data values on graphic attributes. Each multi-dimensional data element is represented by a Glyph.

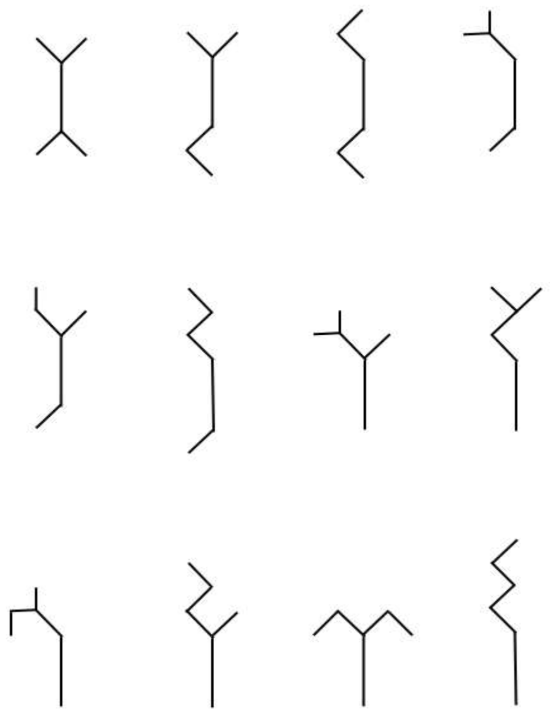
Features should be easy to distinguish and combine
- Icons should be seperated from each other
- Mainly used for multivariate data

Stick-figure icon:
The stick figure was developed by ***[Pickett-1988-IDV]***. The icon consists of several lines forming some kind of manakin. It consists to be added of a base line and at their ends further lines which one can understand as extremities. Its dimension is represented by the number and length of the lines and the angle to each other. Variables can represent the

length, thickness and color of the lines. A representation of different dimensions is possible. If the data records compared to the screen dimensions, lie relatively close together, the resulting visualization shows a structure sample, which varies in accordance with the data characteristics.

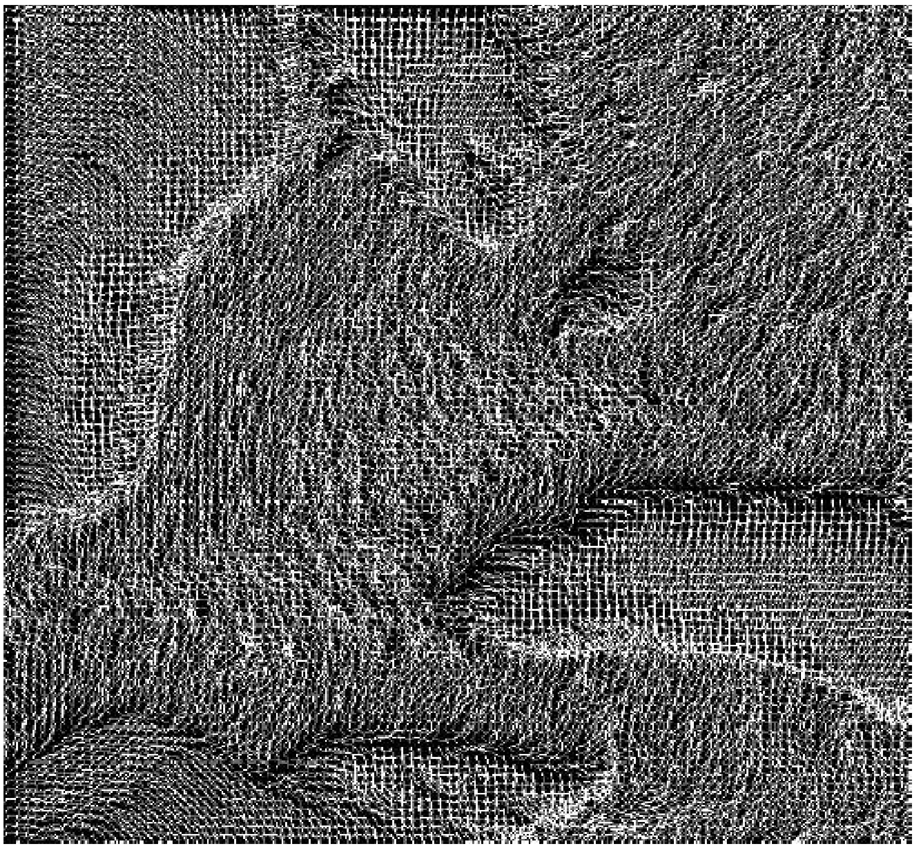
- 2D figure with 4 limbs
- Coding of data via
    - Lenght
    - Thickness
    - Angle with vertical axis
- 12 Attributes
- Exploits the human capability to recognize patterns / textures





These pictures show some examples of how stick figure icons can look like.

The characteristic extremities at the end of the trunk are well recognizable.

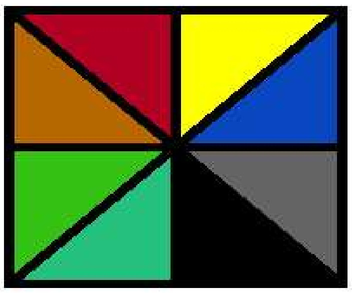

Several stick figure icons arranged next to each other on a surface supplie a surface texture.

Color icons:

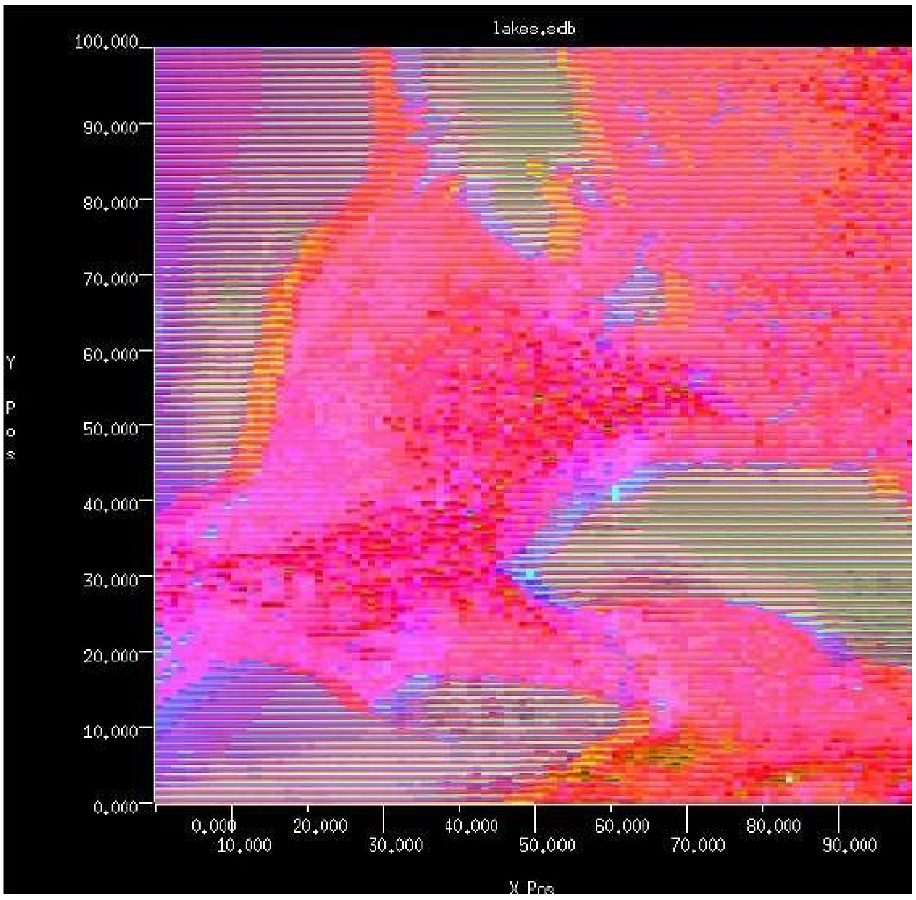
The fundamental principle of color icons (published by ***[Levkowitz-1991-CIC])*** is similar to the stick figures. A square surface is divided into several different colored regions, each region stands for an attribute, each color for a value. Arrangeing several squares on a surface also produces surface texture.

If one generates a color icon, the following characteristics should be considered:

- Subdivision of a basic figure (triangle, square, ...) into edges and faces
- Mapping of data via color tables
- Grouping by emphasizing edges or faces



An example for a color icon is a square divided in eight equal parts.

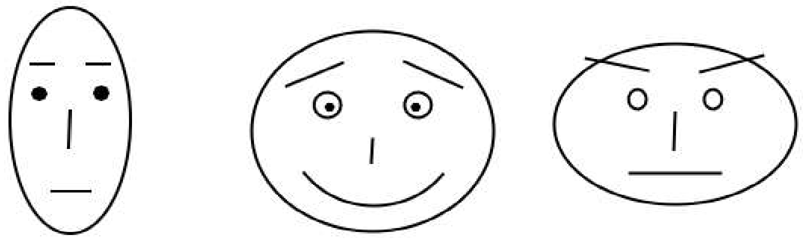


Many of these quadratic color icons combined on a surface genarate a texture as well.

Chernoff Faces:

Another kind of glyphs are the Chernoff faces. In case of geometrical coding schematic faces are used, in order to illustrate trends in multi-dimensional data. Since humans are very good in interpreting faces and they can easily sense their emotions, this metaphor is a simple however effective way of visualizing n-dimensional data in order to pick out trends from the data. This method was developed by ***[Chernoff-1973-UOF]***. The size, form and distances between the parts of the face represent the extents of the individual variables.

- Possible assignement in the decreasing order of importance:
    - Area of the face
    - Shape of the face
    - Length of the nose
    - Location of the mouth
    - Curve of the smile
    - Width of the mouth
    - Location, seperation, angle shape and width of the eyes
    - Location of the pupil
    - Location, angle and width of the eyebrows
- Coding of 15 attributes
- Additional variables could be encoded by making faces asymmetric



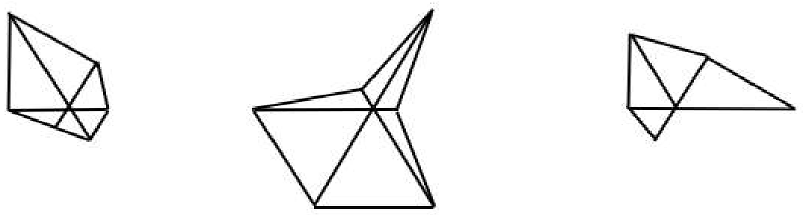
Chernoff Faces.

Chernoff and sun-ray plots lose their simplicity as the sample size gets large.

Circular icon plots like star plots, sun ray plots, etc ... follow a "spoked wheel" format. Values of variables are represented by distances between the center ("hub") of the icon and its edges.

Star Glyphs:
Star glyphs have been established by ***[Fienberg-1979-GMS]***.With star glyphs data values of any dimension are stretched in length by lines. Each line has the same origin and spreads radially from the center. The termination points of the lines are connected, in order to form a polygon.

- A star is composed of equally spaced radii, stemming from the center
- The length of the spike is proportional to the value of the respective attribute
- The first spike / attribute points to the right
- Subsequent spikes are arranged counterclockwise
- The ends of the rays are connected by a line



Star glyphs of some famous cars (from left to right: Audi 500, Dodge St. Regis and Ford Mustang 4).

Sun ray plots:
Sun ray plots are similar to star glyphs, they also have a star-shaped structure and are used to compare more than two endpoints or data types. They are most effective when the reference condition is incorporated into the axes.

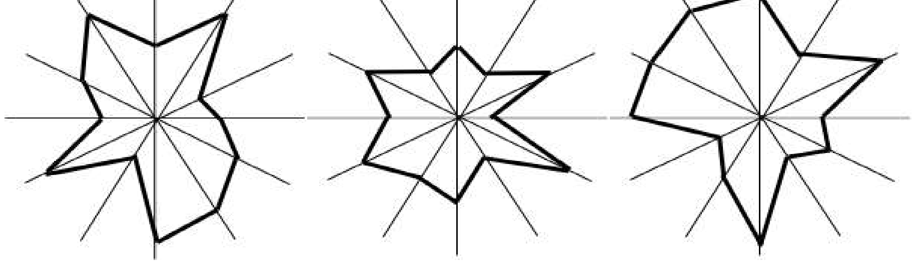


Illustration of some sun ray plots.

6 Multiple Attributes

Another approach to visualize multivariate data is to map data values to different visual primitives. **Multiple attributes** are a typical combination of:

- Geometric position, e.g. height field
- Color: saturation, intensity, tone
- Texture