Konstantin Lackner 19.01.2021

Visualisierung 1 VO Stoffzusammenfassung 15.01.21

1. Einleitung

Carbonfaserverstärkte Kunststoffe

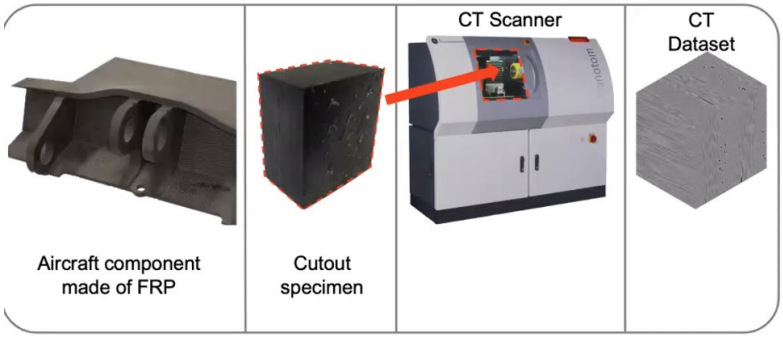
Dabei handelt es sich um ein Beispiel für Einsatzgebiete der Visualisierung.

Wird für Tennisschläger, Schi, Flugzeuge, usw. eingesetzt. Professor arbeitet an solchen Materialien und eben auch Flugzeugen und scheint das sehr relevant für die VO zu finden.

Er spricht dabei von **Advanced Materials**

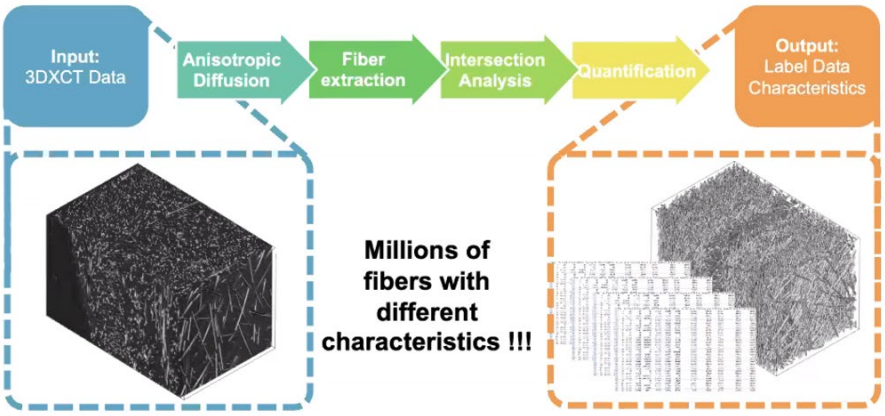
Man kann in Materialien reinschauen mittels Computertomografie. Man sieht dabei die Fasern, was natürlich relevant ist, weil mehr Faserrichtungen auch mehr Stabilität bringen. Interessant ist also -> Wie sind die Fasern orientiert, wie beschaffen? Wie lang, dick, dünn… usw..

Am besten geht das normalerweise indem man cutouts nimmt und anhand dieser misst.



CTs werden da meist nur als Eskalation benutzt. Zuerst wird mit Ultraschall in Objekte geschaut, fallen da Ungereimtheiten auf, wird ein CT gemacht.

Man nimmt die CT Daten und wendet eine Preprocessing Pipeline an.



Man filtert und extrahiert die einzelnen Fasern. Also bekommt jede Faser eine eigene ID, und gewisse Eigenschaften zugewiesen.

Konstantin Lackner 19.01.2021

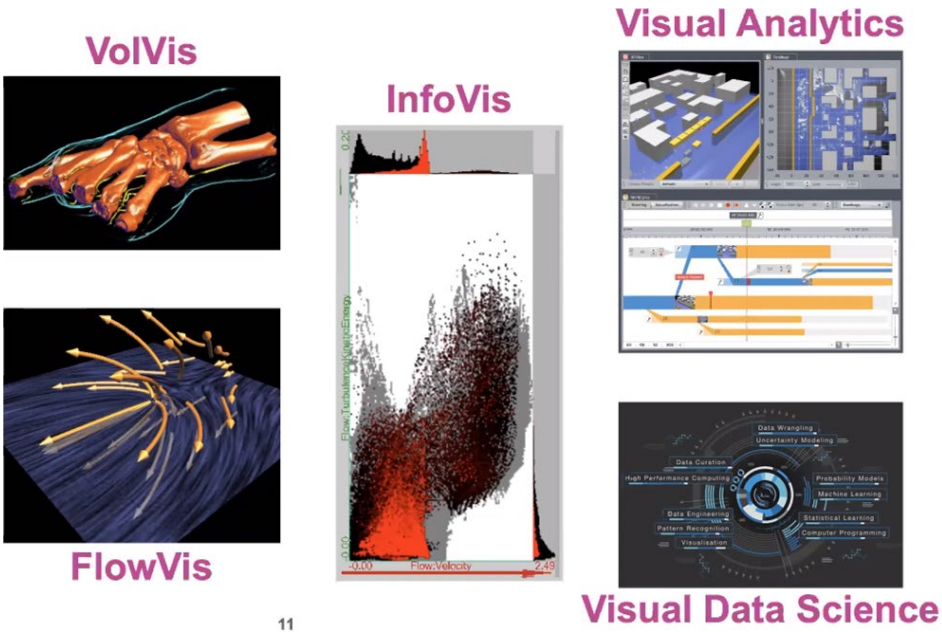
Warum Visualisierung?

Einerseits hat man sehr viele Daten, andererseits hat man sehr verschiedene Daten. Ein Mensch kann sich darauf aber nicht wirklich einen Reim machen. Daher muss man Einblicke in Materialien generieren, die mehr insight erlauben.

„The purpose of computing is insight, not numbers" – Richard Hamming

Man sammelt immer mehr Daten, kann diese als Mensch aber gar nicht aufnehmen.

Arten der Visualisierung – kurzer Überlick



VolVis – Ein Skalarfeld, also ein Bereich wo an jeder Position ein Wert steht (bspw. Dichtewert)

FlowVis – Man hat ein Volumen in dem Richtungen oder Druckveränderungen usw. gemessen werden.

InfoVis – Abstrakte Daten; Hier bekommt man Messdaten, man macht bspw. einen Scatterplot und versucht über zwei Messdatensätze einen Zusammenhang abzubilden. Bspw. Velocity zu kinetic energy.

Visual Analytics – Die Kombination aus verschiedenen Bereichen bspw. dieses Überschwemmungsstaudammding. Man hat eine Volumen, flow und co. Darstellung.

Visual Data Science – Pattern recognition, Machine learning … (+ diversen anderen Bullshitbingobegriffe ^^)

Background

Unseeable Biology – TED Talk zur Darstellung von Molekülen.

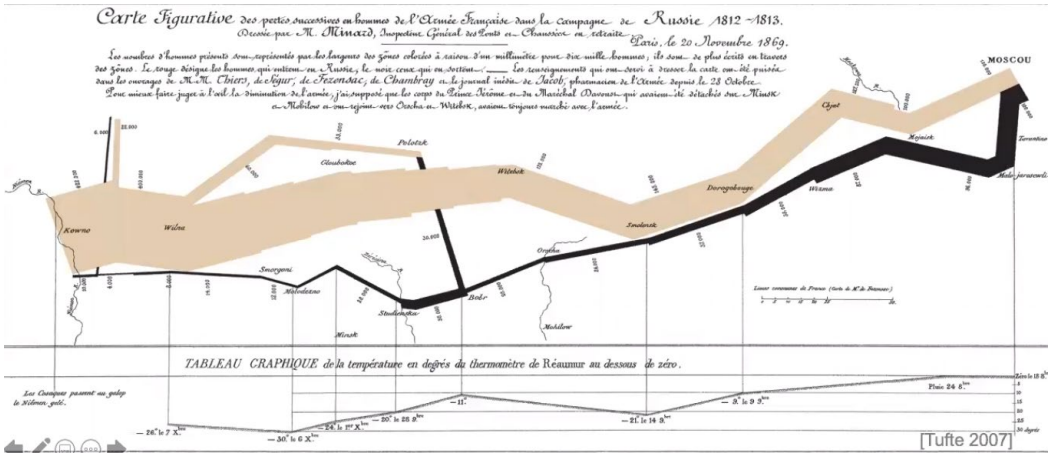
Es geht auch hier darum, dass die Erklärung unsichtbarer Phänomene sehr schwer ist. Wenn man aber Computergrafik benutzt, um diese Dinge darzustellen, wird die Verständlichkeit natürlich enorm verbessert.

Spezifisch werden hier auch Chromosomen gezeigt, und wie DNAstränge um Proteine gewickelt sind.

Vis ist sehr alt. Bereits da Vinci hat schon Vis genutzt um bspw. Flüsse oder Strömungen darzustellen. Man nutzt Vis weil Datenmengen immer größer werden und die Möglichkeit Dinge darzustellen damit immer wichtiger wird.

Beispiel Cholera Epidemic in London – Durch eine Karte der Fälle hat der Wissenschafter John Snow festgestellt, dass die Fälle sich alle um eine zentrale Straße anordnen. Er konnte also den Zusammenhang zwischen einer gewissen Wasserversorgung und Cholerafällen herstellen. Akzeptiert wurde seine Theorie erst Jahre später.

Beispiel Military Campaign of Napoleon – Man zeigt die Truppenstärke in der Dicke eines Strichs und zeigt genau, wie die Truppen schlussendlich in Moskau angekommen sind. Die schwarze Linie hingegen ist der Rückzug.



Zusätzlich hat man unten Zahlen. Man sieht, dass die Temperatur mit der Gruppengröße korreliert.

Definition

Vis hilft ein mentales Bild von komplexen Zusammenhängen darzustellen.

Vis is a tool to enable a user insight into data

the use of computer-supported interactive, visual representations of abstract data to amplify cognition.

Gebiete der Visualisierung heute

Heute Volvis, Flowvis und Infovis – außerdem historical data(archeologist), microscopic data(molecular physics), macroscopic data(astronomy) und extremely large data sets.

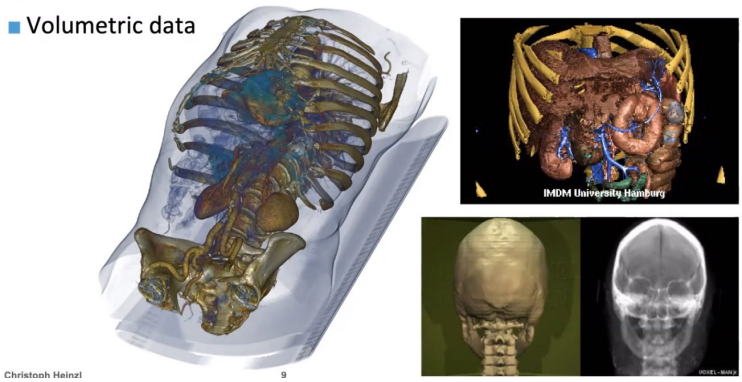
Arten der Visualisierung – Etwas weniger kurzer Überblick

VolVis

Hier versucht man meist mehr Kontext in Daten zu bringen. Bspw. versucht man Farben so einzusetzen, wie diese auch in der echten Welt sind. Andererseits schaut man bspw. auch in Körper rein. Mit Dichtewerten werden einzelne Bereiche ein- und ausgeblendet. Es gibt auch semitransparente Darstellungen, bei denen bspw. mehrere Dinge gleichzeitig als Kontext gezeigt werden.

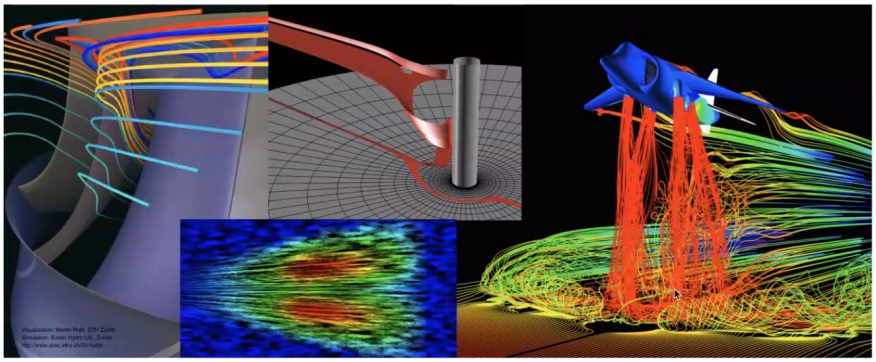
In allen diesen Darstellungen ist die zugrundeliegende Datenbasis immer dieselbe. Ein volumetrischer Datensatz mit drei Dimensionen und an jeder Position hat man einen

Grauwert. (weil die Daten nicht normiert sind, nennt man sie so). Damit kann man dann diverse Darstellungen machen. Bspw. maximum intensity projection.

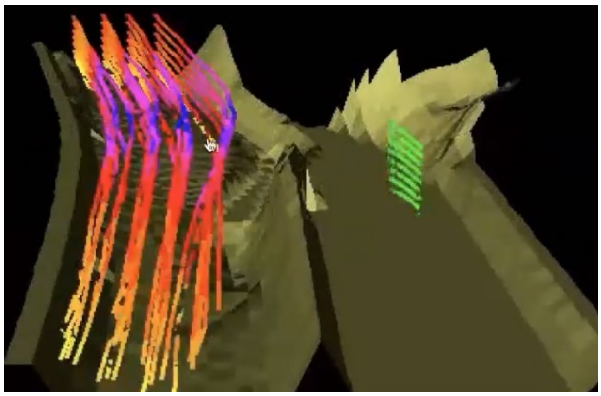


FlowVis

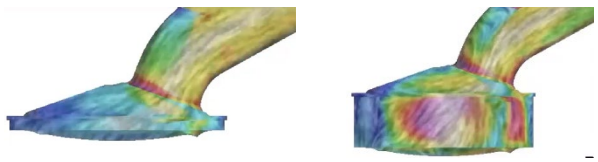
Man hat auch einen räumlichen Bezug. Man visualisiert dann um ein Objekt oder in einem Objekt Daten bzgl. des Flusses von Luft, Wasser, Gasen, was auch immer.



Hat man also indirekte Flowvis, stellt man zb. die Strömung von Blut in Adern dar.



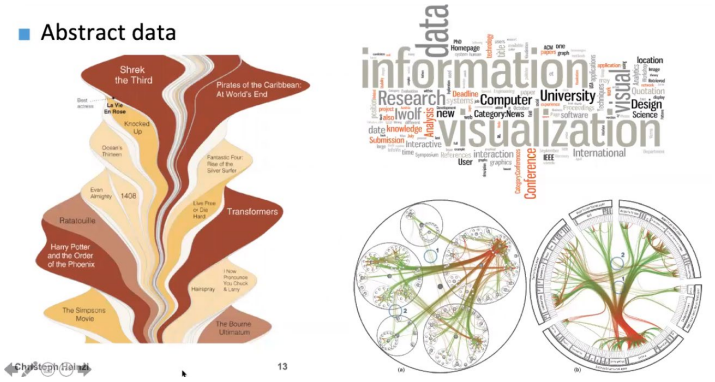
Es gibt aber auch die direkte Flowvis. Dabei wird für jeden Punkt im Raum der Fluss einer Strömung codiert. Bei direkter Flussvisualisierung zeigt man bspw. das Treibstoff-Luftgemisch, dass in einen Motor fließt. (Aber nicht den Motor)



Konstantin Lackner 19.01.2021

InfoVis

Hauptsächlich abstrakte Daten. Bspw. Besucherzahlen in Kinos usw.



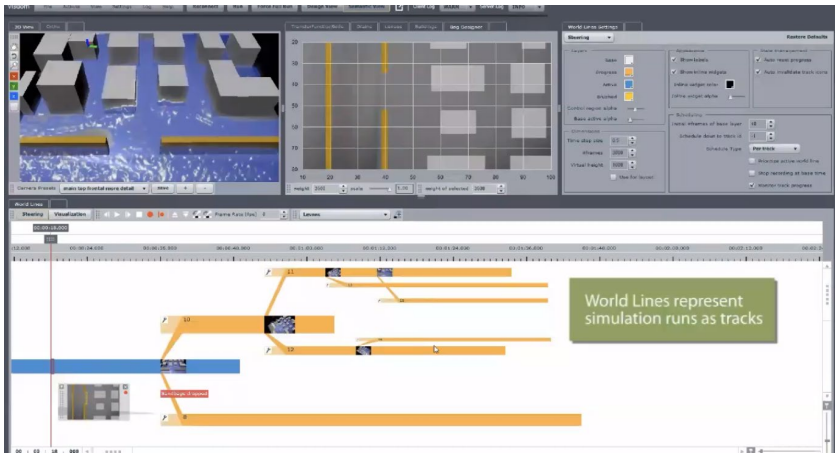
Von oben nach unten werden die Besucherzahlen in der Breite der Flüsse gezeigt. **Techclouds** (rechts oben) zeigen einfach Wörter / Bereiche, und deren Häufigkeit. Diese werden dargestellt mit Farbe und Größe. Man kann theoretisch auch fonts einbauen. Rechts unten sieht man **hierarchical edgebundles**. Man sieht anhand der Dicke wie stark welche Dinge verbunden sind.

Visual Data science / analytics

Hier geht es meist um komplexe Daten. Oft räumliche Daten, zeitliche Daten und eben abstract data. Bspw. Simulationen von Feuer für Filme, oder Fischereidaten. Die Vis Data science ist dabei der Teil, der aus den Daten neue Daten oder andere Daten generiert – machine learning usw., die Vis analytics ist dann der Bereich, in dem durch visuelle Analyse neue Daten generiert, oder Daten interpretiert werden – dabei handelt es sich dann um eine Mischung aus den vorhergenannten Bereichen.

Beispiel World Lines

Das World Lines tool vereint mehrere Simulationstechniken. Das ist dieses Überschwemmungstool.



Wird bspw. für Planung und Vorbereitung zu Hurricanes benutzt. Zugrundeliegend war damals ein großer Dammbruch. Mit den einzelnen Linien sieht man verschiedene Events. Bspw. das Abwerfen von Sandsäcken. Man kann eben testen, ob und wie gewisse Maßnahmen schützen. Man kann auch anzeigen, welche Gebäude am meisten und wie stark betroffen sind.

Konstantin Lackner 19.01.2021

Ziele der Vis

Explore

Nichts ist bekannt, man erforscht Datensätze.

Analyse

Man hat bereits eine Theorie und versucht Hypothesen zu verifizieren.

Present

Es ist bereits alles über die Daten bekannt, man versucht diese darzustellen/kommunizieren.

Scientific Vis vs. Information Vis & Vis Analytics & Vis Data Science

Scientific Vis – Volume, Flow – mit spatial reference in 3D

Rest - Gegenüber dem InformationVis und Visual Analytics als auch Vis data science – oft keine spatial reference und auch ned immer in 3D sondern wechseln, also n Dimensionen.

Prozess der Vis

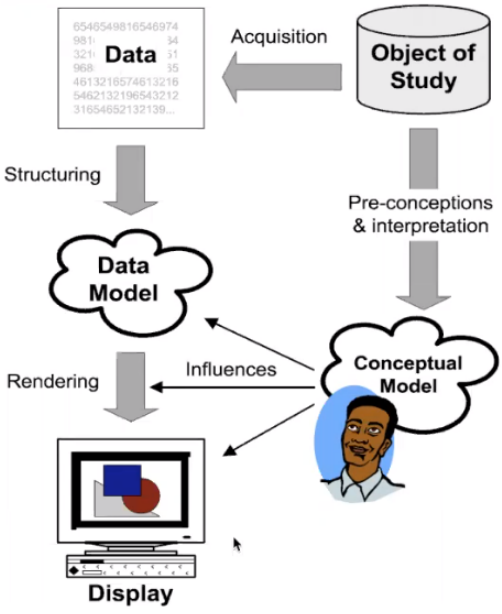
1. Zunächst hat man ein object of study.

2. Dann macht man die data acquisition, dabei hat man schon ein gewisses Konzept dazu.

3. (Gleichzeitig zu 2.) Man macht ein konzeptuelles Modell und überlegt, wie man die darstellen kann.

4. Aus 2. und 3. macht man ein Datenmodell.

5. Das wird dann gerendert, das Modell wird am Display ausgegeben.



Vis Pipeline

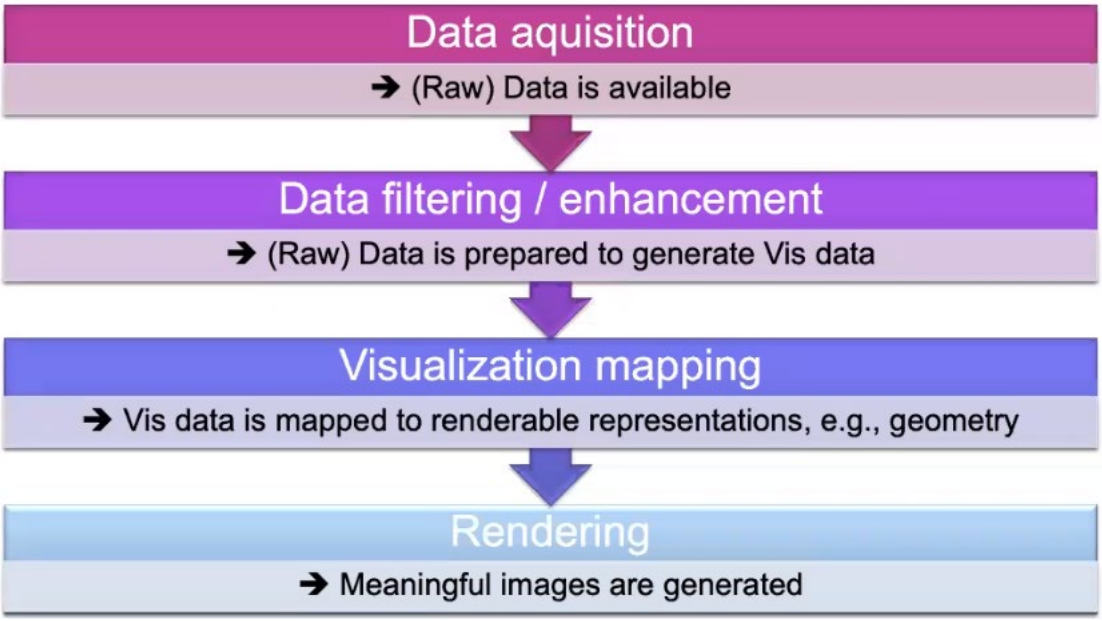
Diese ist aus dem Prozess abgeleitet.

1.) Zuerst erneut die data acquisition – CT, Scanner, Datenbanken… - Danach hat man Rohdaten.

2.) Data filtering and enhancement – Daten werden gefiltert und verbessert – Rohdaten werden in Visualisierungsdaten umgewandelt. Dabei gibt es **clipping**, **cropping**, **slicing**, aber auch **Datenumwandlung**. **Segmentierung** (bspw. Tumorstrutkuren einteilen) **Feature extraction**, **clustering**, **Filtern**(noise reduction), **resampling**(resizing), **Interpolation**(auch Gridconversion).

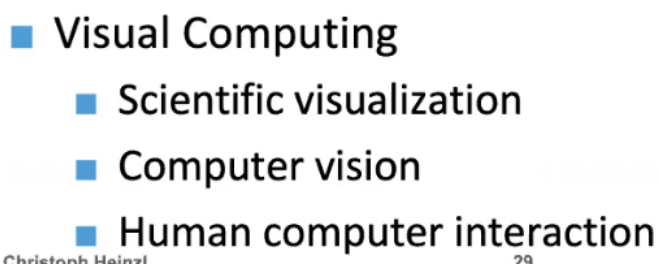
3.) Vismapping – Man führt die Daten in visuelle Repräsentationen über. Daten werden zu Grafikprimitiven. Ein 3D Scalar field wird bspw. zu einem Volumen und isosurface. 3D Daten können bspw. zu 2D Bildern projiziert werden. Man berechnet sich eben das Layout der Vis. Auch colours, Transparenz werden hier zugeordnet.

4.) Rendering – Man hat **renderable representations** im Vorschritt errechnet. Jetzt muss man also Schattierungen, Licht, Shading, Interaktion (Drehung, usw.) festlegen. Auch Animationen fallen in diesen Bereich. View frustum wird auch hier berechnet.

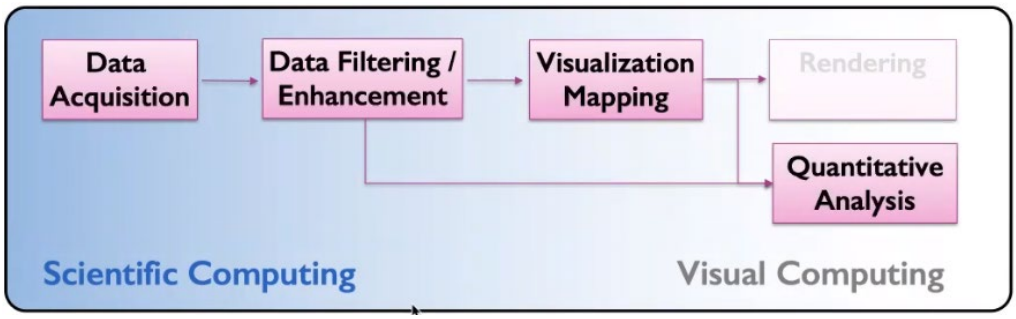


Einordnung der Vis in Computational Sciences

Computational Sciences umfasst einige Bereiche. Vorrangig geht es bei der Vis natürlich um:



Eingebettet in die Pipeline, ist sie natürlich eher auf Seite des Vis Computing. Aber auch die anderen Bereiche sind relevant.



Konstantin Lackner 19.01.2021

2. Daten

Zu Daten muss man sich in der Vis einige Fragen stellen.

Datenraum

Zu den Daten muss man sich überlegen, wo der Datenraum ist, in dem man operiert.

In spatial domain (SciVis) hat man meist 2D/3D space. Bspw. bei medical data, flow simulation data, etc. Hat man hingegen keine inherente spatial reference (infoVis), sondern abstrakte Daten oder gewisse Datenbanken? Dann muss man sich dafür Visualisierungen überlegen.

Die Aspekte um die es hier also geht ist: Die **Dimensionalität** des Datenraums, die **Koordinaten** – ist ein **globaler Überblick** wichtig, oder eine gewisse **Richtung/Bereich**?, **Wofür designt man** das Tool?

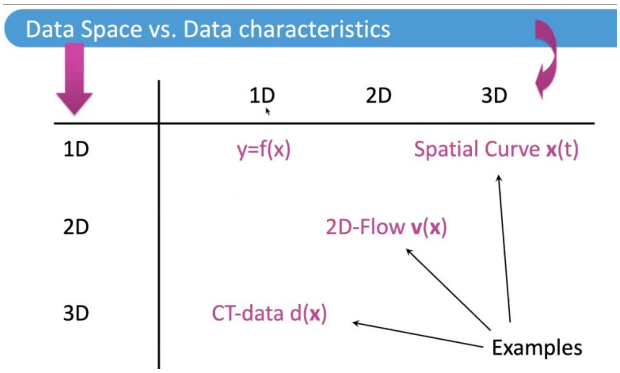
Datentypen

Skalare = numerische Daten, Nicht-numerische Daten, Multidimensionale-Daten, Multimodale Daten (Vektoren mit verschiedenen Typen) - Hier muss man auf die Dimensionalität und die codomains (Also die ranges dazu) achten.

Datenrepräsentation

Dann fragt man sich, wie man die Daten repräsentieren kann. Hat man räumliche Daten, kann man sich z.B. fragen, ob man diesen Raum auch zur Vis nutzt. Was macht hier Sinn? welche Dimension ist wofür verwendet? Gibt es eine Relation zwischen Datenraum und Datencharakteristik? Ist der Datenraum in 2D, 3D, 4D? Gibt es einen speziellen Ort, wo genauer hingesehen werden muss? = Fokus.

Beispiele zu Daten und deren Charakteristiken:



	1D	2D	3D	nD
1D	bar chart, pie chart (line) graphs...		spatial curves...	parallel coordinates, glyphs, Icons...
2D	2D-height map in 3D, contour lines in 2D, false color map...	2D flow...		
3D	iso-surfaces in 3D, volume rendering of CT data...		3D flow...	tensors...
nD	time depending CT data...			

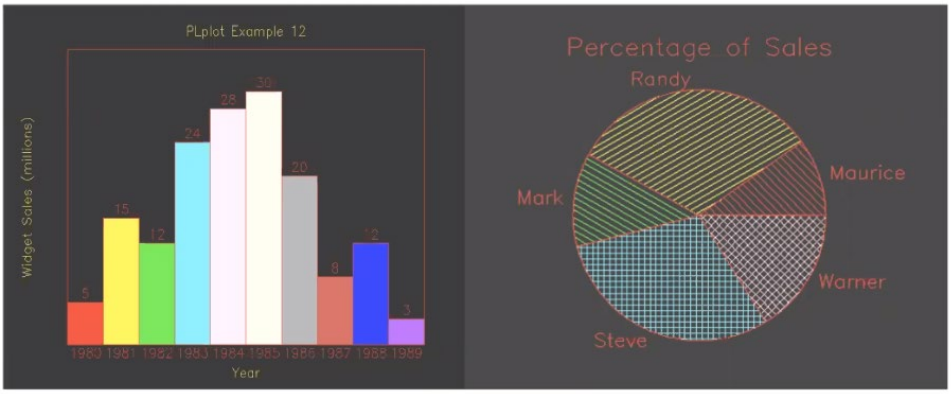
Konstantin Lackner 19.01.2021

Mapping

Infografiken

Hier geht man von **einer** natürlichen Zahl als Datenpunkt aus und wandelt sie in **eine** reelle Zahl um.

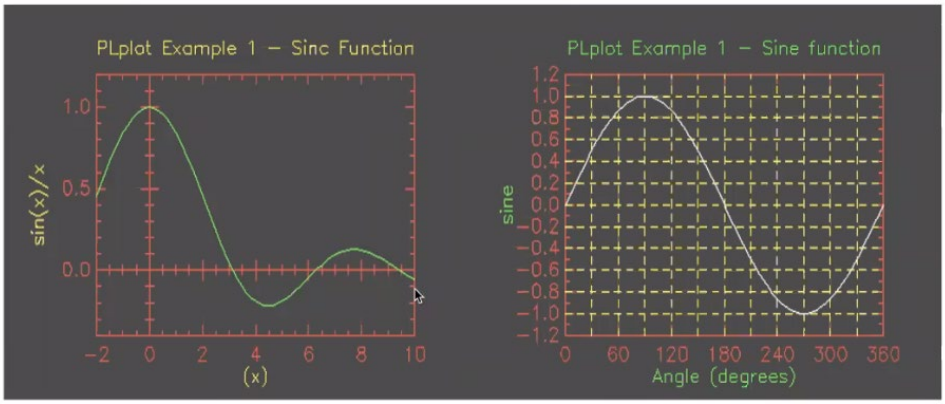
data	description	visualization example
$N^1 \to R^1$	value series	bar chart, pie chart, etc.



Line Graphs

Hier mapt man von reellen Zahlen zu reellen Zahlen.

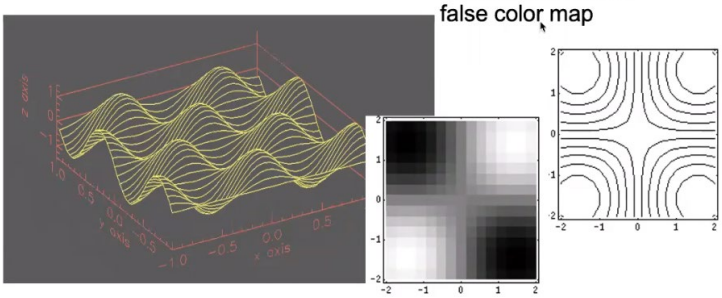
data	description	visualization example
$R^1 \to R^1$	function	(line) graph



2D height maps

Ähnlich wie zuvor. Man geht von R2 zu R1. Man hat also einen Bezug in R2 und für jedes Element in R2 kann man ein Element in R1 codieren.

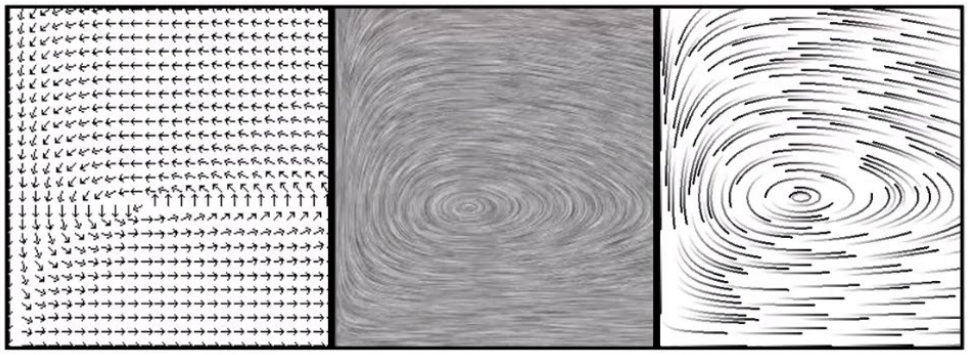
data	description	visualization example
$R^2 \rightarrow R^1$	function over R^2	2D-height map in 3D, contour lines in 2D, false color map



2D Vektorfelder

Wenn man Natürliche Zahlen in N2 auf Reelle in R2 mapt.

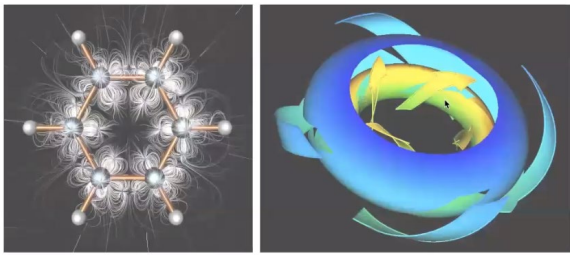
data	description	visualization example
$N^2 \rightarrow R^2$	2D-vector field	hedgehog plot, LIC, streamlets, etc



Streamlines und surfaces

Mapping von R3 auf R3

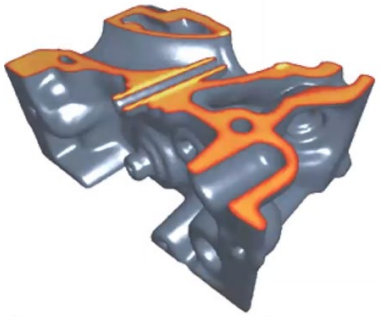
data	description	visualization example
$R^3 \rightarrow R^3$	3D-flow	streamlines, streamsurfaces



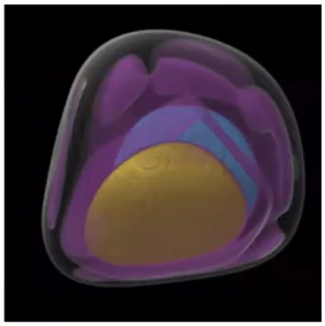
Iso surfaces in 3D, Volumes

Hauptsächlich in R3 aber gemappt auf R1.

data	description	visualization example
$R^3 \rightarrow R^1$	3D-densities	iso-surfaces in 3D, volume rendering



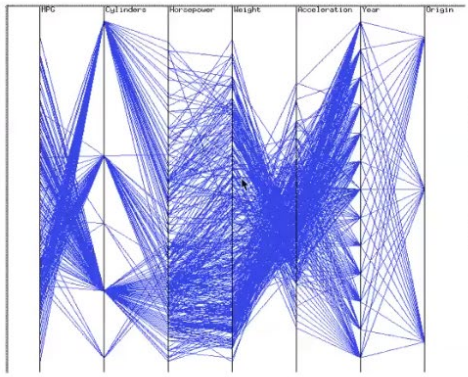
43

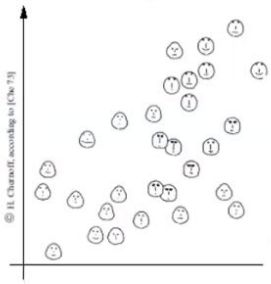


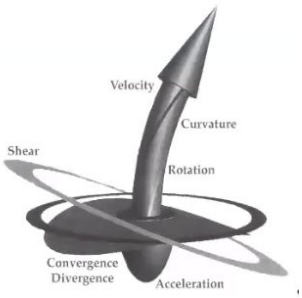
Tupel (erzeugen Glyphen oder icons)

Bei noch komplexeren Daten und evtl. keine Mappings vorhanden.

data	description	visualization example
$(N^1 \rightarrow) R^n$	set of tuples	parallel coordinates, glyphs, icons, etc.







44

Konstantin Lackner 19.01.2021

3. Grids

Man muss sich beim Organisieren von Daten fragen:

1.) Welche Datenorganisation ist optimal?

2.) Woher kommen die Daten?

3.) Gibt es eine Nachbarschaftsbeziehung?

4.) Wie ist die Nachbarschaftsbeziehung gespeichert?

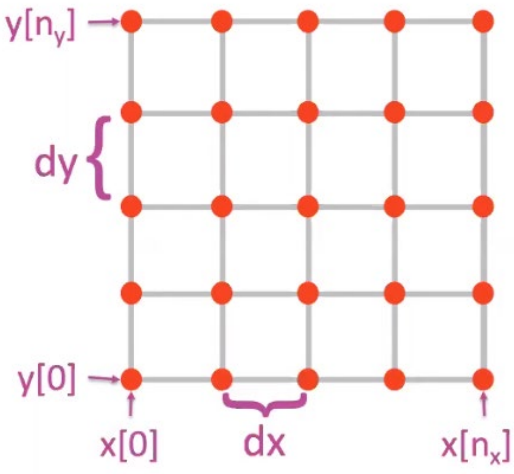
5.) Wie kann man innerhalb der Daten navigieren?

6.) Kann man mit den Daten Berechnungen anstellen?

7.) Sind die Daten strukturiert?

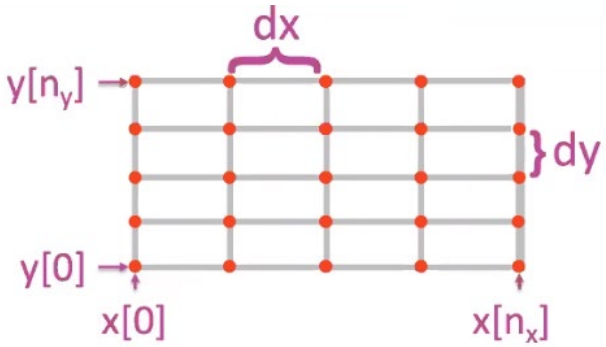
Cartesian Grids

Ist das einfachste Datengrid. Es ist orthogonal(rechtwinklig), man hat gleiche Distanzen in y und x. Für jedes Element gibt es eine explizite Nachbarschaft.



Regular Grids

Ähnlich wie cartesian. Nur ist dx und dy verschieden.

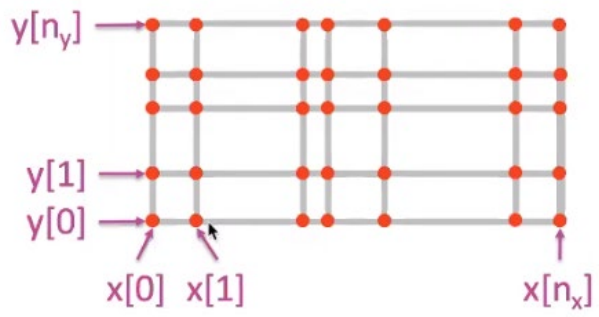


Tritt auf, wenn man bspw. einen Patienten durch ein CT durchschiebt mit Geschwindigkeit X, aber die Auflösung in Y-Richtung des Systems höher ist, als die Abgreifrate im Verhältnis zur Geschwindigkeit in X Richtung.

Konstantin Lackner 19.01.2021

Rectiliniar Grids

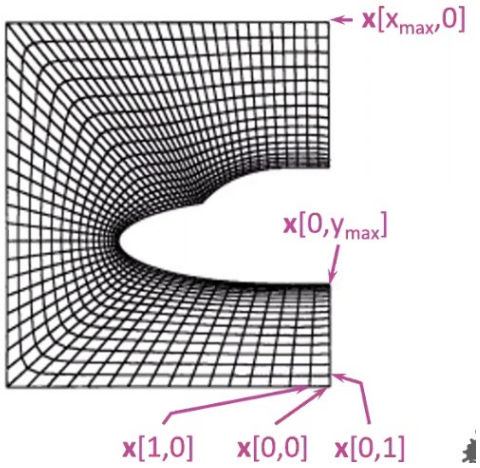
Bei diesen müssen die Distanzen nicht mehr gleich sein.



Immer noch orthogonal und die Nachbarschaften sind auch noch klar.

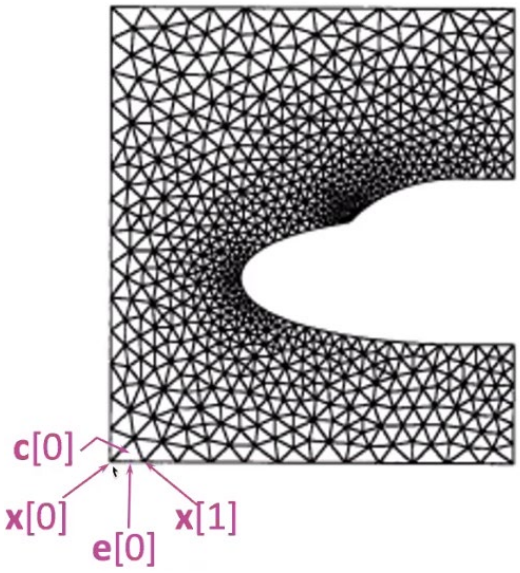
Curviliniear Grids

Man weiß noch wer der Nachbar ist, aber Orthogonalität gilt nicht mehr.



Unstructured Grids

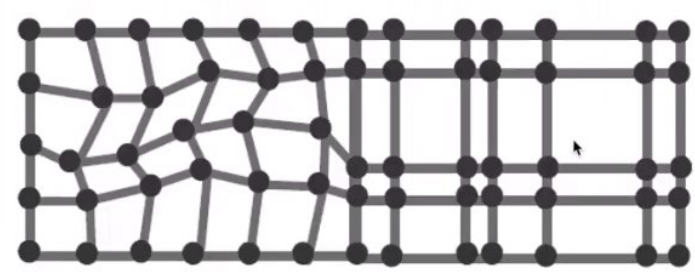
Man weiß wie die Punkte und Kanten angeordnet sind. Hier hat man noch explizit Nachbarschaften. Das Interpolieren ist aber nicht mehr so leicht möglich.



Konstantin Lackner 19.01.2021

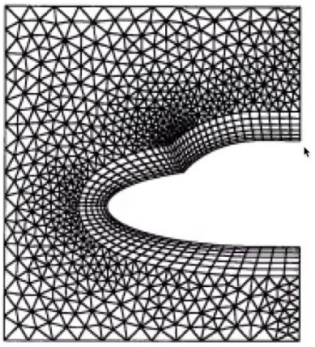
Block-Structured Grids

Man übernimmt gewisse Strukturen für jedes Element. Ist im Endeffekt eine Kombination aus Grids.



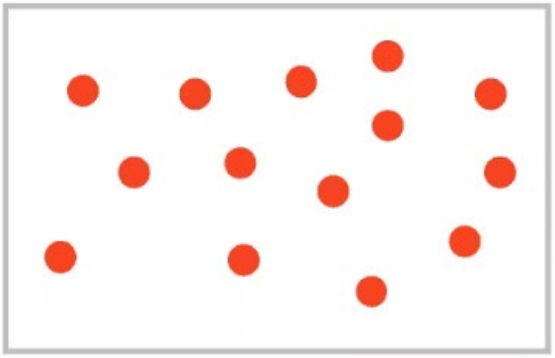
Hybrid Grids

Man will bspw. um einen gewissen Punkt die Nachbarschaft deutlich und einfach darstellen, dann nimmt man diese Kombination.



Scattered Data

Keine Nachbarschaftsbeziehung, kein Grid, keinen Influence.



Konstantin Lackner 19.01.2021

Umrechnung von Grids

- Conversion between grids:
  - physical domain (simulation)
  - computational domain (visualization mapping)
  - image domain (rendering)
  - etc.

- Questions:
  - Accuracy of re-sampling!
  - Design of algorithms

Konstantin Lackner 19.01.2021

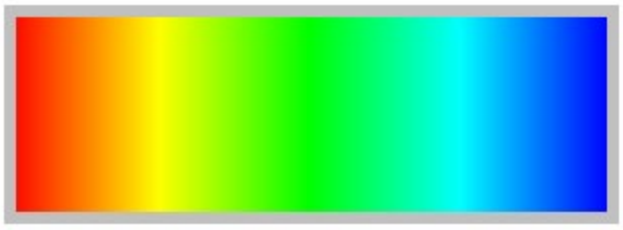
4. Farben

Fakten zu Farben

Farben unterstreichen Informationen und sollen daher im Kontext zu diesen genutzt werden.

Anzahl an unterscheidbaren Farben ungefähr 7+-2. Für jede dieser Farben ungefähr 50-300 Schattierungen (abhängig von Farbe)

Rainbow colour scale != linear -> heißt, dass man zwar von einem Farbton auf den anderen vl. noch unterschieden kann, aber der gleiche Abstand woanders im scale muss nicht unterscheidbar sein.



Usage

Colour perception ist stark vom Kontext abhängig.
Colour blind users are handicapped.
Colour associations sind relevant.

	Sensation	Taste	Temp.	Weight
Blue	bright: soft dark: hard	neutral	cool, cold	bright: light, dark: heavy
Red	rough	spicy, crispy	warm, hot	(as blue)
Green	-	bitter	cool	(as blue)
Yellow	soft	sweet	warm	light
Pink	very soft	sweetish	skintemp	light

Möglichkeit Farben auszuwählen

Colorbrewer 2.0 (printer friendly, colour blind safe)

Colour gamuts

Verschiedene Monitore und Anzeigegeräte haben verschiedene Farbbereiche.

Konstantin Lackner 19.01.2021

Guidelines

1.) Keine entsättigten Linien als borders für coloured areas

2.) Keine gesättigten Blautöne für Details oder Animationen

3.) Niemals gesättigtes Rot und Blau mixen -> Beginnt sich zu bewegen

4.) Keine high colour frequencies benutzen

5.) Vergleichsfarben sollen nah beieinander sein. Wenn man Grautöne vergleicht bspw. sticht eine sehr kleine Änderung stark heraus

6.) Farben müssen an den Kontext angepasst werden

7.) Redundanz kann durch Farben ausgedrückt werden

Konstantin Lackner 19.01.2021

5. Volume Vis

Beispielvideo virtuelle Autopsie. Man sieht wie mehrere Leuten auf einem surface table arbeiten. Sie stellen Transferfunktionen ein, um bspw. Knochen freizulegen, oder Haut anzuzeigen.

Bei Volume Vis will man Daten aus 3D auf 2D mappen (also auf einen Bildschirm).

Dabei benutzt man diverse Verfahren: **Projection**, **slicing**, **vol. rendering**

Die Daten stammen aus dem 3D Raum und man hat für jede Position im 3D Raum genau einen Wert. **Also 3Dx1D**.

Ziel ist insight, Strukturen und co. sehen.

Daten werden mit CTs gesammelt, oder MRTs. Auch bei Material testing werden CTs benutzt. Außerdem können Daten auch simuliert werden bspw. bei finite element methods oder computational fluid dynamics.

Herausforderungen der VolVis

Sehr viel Information und die Pixel am screen sind, im Vergleich zur Datengröße, sehr wenige und klein.

1024*1024*1024 voxel à 16 bit (uint) = 2 GB
1024*1024*1024 voxel à 32 bit (float) = 4 GB
2048*2048*2048 voxel à 16 bit (uint) = 16 GB
2048*2048*2048 voxel à 32 bit (float) = 32 GB
4096*4096*4096 voxel à 16 bit (uint) = 128 GB
4096*4096*4096 voxel à 32 bit (float) = 256 GB

(immer in drei Dimensionen)

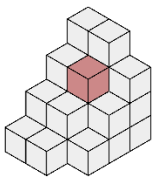
Wenn man an Räumlichzeitliche Daten denkt, muss man ja auch Zeitschritte darstellen. Damit hat man dann bei 1000 Zeitschritten: (bspw. wenn man die Kraft immer erhöht, bis ein Bauteil bricht)

4096*4096*4096 voxel à 32 bit (float) * 1000 steps = 256000 GB

Speed ist ein Problem insofern, dass Interaktion im Render wichtig ist. Also müssen >10fps erreicht werden.

Volumetrische Datensätze

Man hat ein Grid von x*y*z Datenelemente. Diese sind also in einem Volumen angeordnet. Die volumetric elements nennt man dabei Voxel.

 <-Voxel

Man braucht zur Speicherung:

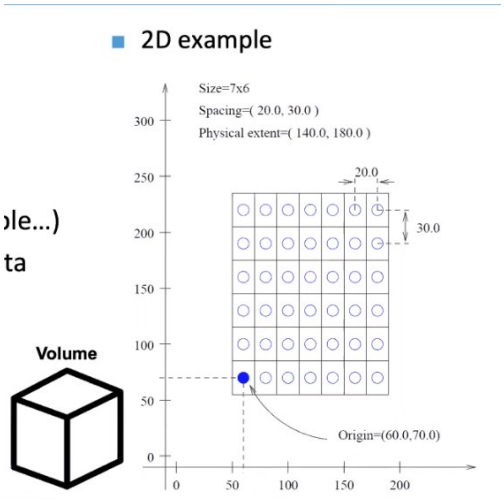
1.) Einen Datenwert für die Speicherung der Werte – float, short, integer, double…

2.) Außerdem braucht man einen Ursprungspunkt = Origin

3.) Eine Größe

4.) Ein Spacing, also den Abstand der Voxels zueinander – ist das bspw. uniform, oder verschieden?

5.) Und letztlich Units. Man muss wissen wovon man sprich… mm, nm, …



Das Beispiel ist in 2D, normalerweise ist es 3D

Image stacks

Die volumetric Datensätze können auch als Image stacks daherkommen.

Das wären dann 2D Daten mit 1D Skalaren.

Beliebige Imageformate – png, jpeg, usw. Am ehesten tiff, weil es nicht komprimiert ist und man in bis zu 32bit speichern kann, sogar auch als Imagestack oder ganzes Volumen.

Dabei hat man eine implizite Definition des Grid. Nämlich die Dimension der Bilder und die Anzahl der Bilder.

Datenorganisierung

Die Daten sind normalerweise karthesische oder reguläre grids. (karthesisch = gleiche Abstände in x wie y und rechte Winkel, regulär is dasselbe ohne gleiche Abstände)

In der Regel hat man oft karthesische Grids bezogen auf x und y, aber die z Richtung ist oft nicht gleich. Das liegt eben daran, dass bspw. durchs CT durchgeschoben wird. x und y sind durch die Detektorzeile fix definiert, aber z liegt an der Geschwindigkeit, mit der durchgeschoben wird. Durch Trägheit usw. kann es dazu kommen, dass das Gerät nicht mm genau schiebt. Deswegen wird die Position zu jeder Aufnahme in solchen Geräten gespeichert und dann rückgerechnet.

Da gibt es auch die erste Möglichkeit zum **data enhancement** – Man kann aus einem regular grid ein carthesian grid machen, sodass dx=dy=dz.

Bei cartesian resp. regular grid sind Daten meist in Form von Zellen(cuboids) definiert. An den Kanten hat man Voxel.

Spezielle Fälle sind curvi-linear grids. Dabei werden die Daten in tetrahedra bzw. hexahedra definiert.

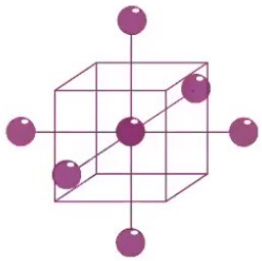
Konstantin Lackner 19.01.2021

Voxel vs Zellen

Man hat zwei Möglichkeiten das Volumen zu interpretieren.

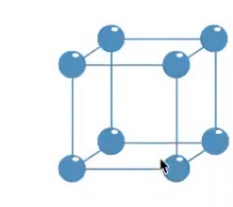
Voxel

Man nimmt den Datenwert als Zentrum des Voxels. Als Nachbarn dann die anderen Datenwerte. Ein Voxel ist also ein pointsample in 3D. Der Würfel ist nur eine Metapher.



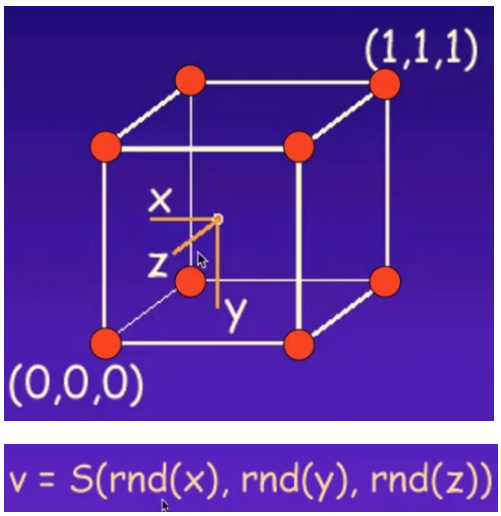
Zellen

Zellen hingegen nehmen immer 8 Werte. Diese 8 Werte bilden dann einen Würfel. Eine Zelle sind also 8 Voxel. Das geht natürlich nur bei karthesischen Grids.

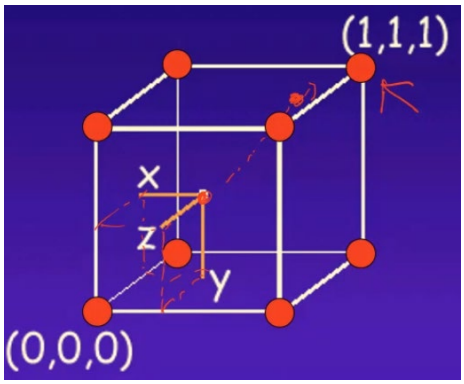


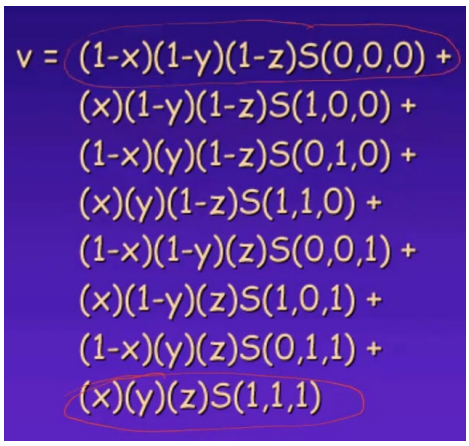
Interpolation

Mit diesen Modellen (Voxel und Cell) kann man gut interpolieren. Einerseits kann man die nearest neighbour Methode benutzen. Also einfach den nächsten Punkt jeweils nehmen.

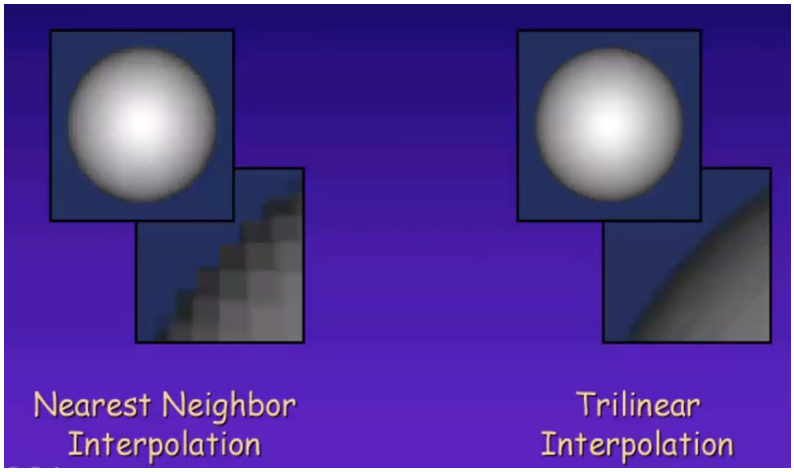


Andererseits kann man auch die Trilineare Interpolation anwenden. Dabei definiert man Gewichte. Man definiert sich also einen Würfel im Würfel. Schaut sich die Entfernungen von neuen Voxel zu den alten an und je näher man ist, desto stärker gewichtet man dann auch.





Beispiele zu Interpolation

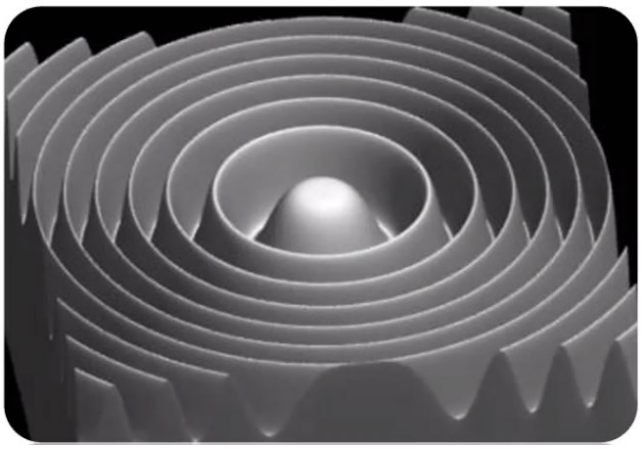


Bei NN starkes aliasing, weil man nur die Werte hat, die bereits in anderen Voxels waren. Bei Trilinear sieht man dann **Bandeffekte**. Je nachdem wie das Bauteil im Raum liegt, werden die Effekte anders dargestellt.

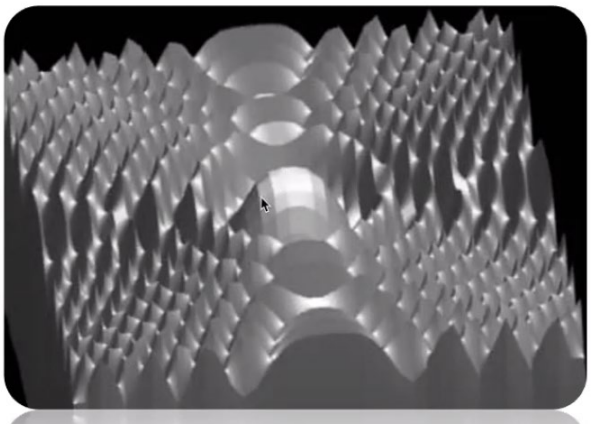
Higher-order Reconstruction

Dabei braucht man höhere Qualität, also bessere Rekonstruktionsfilter. Man hat also eine sehr nahe Frequenz zur **Nyquistfrequenz**. Das Testsignal also muss so abgegriffen werden, dass die doppelte Frequenz des Ursprungs die Abgrifffrequenz ist. Beispiel dazu das Marschner-Lob Testsignal.

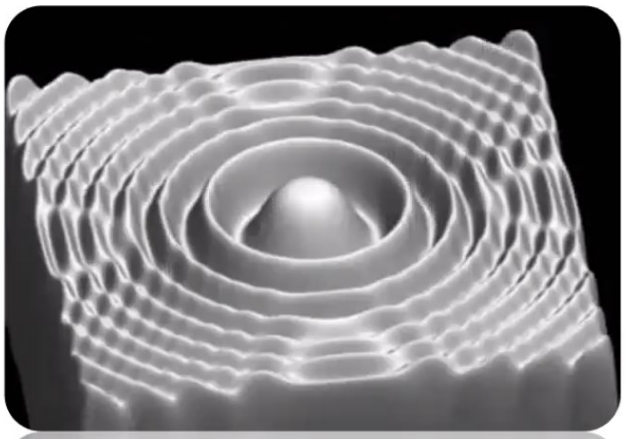
Konstantin Lackner 19.01.2021



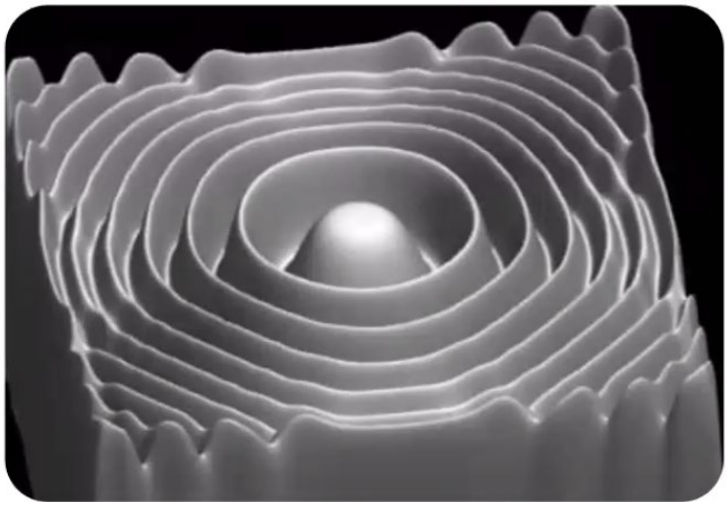
Mit analytischer Rekonstruktion (eben auf Nyquist geachtet)



Hingegen trilinear



Mit B-spline Interpolation. Also nicht nur mit Linien wie zuvor bei trilinear. Allerdings wird hier die Amplitude nicht hoch genug gesetzt.

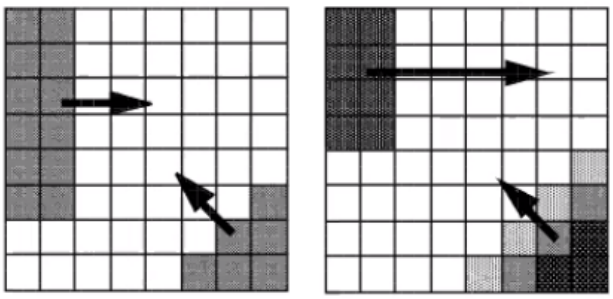


Windowed sync also sinx/x nur eben iwo abgeschnitten (windowed) funktioniert am besten. Man hat allerdings leichte Deformationen in der Form. Man hat einfach zu wenig Daten um das Signal wieder ganz abzubilden.

99% der Systeme benutzen nur Trilinear. Das liegt vor allem daran, dass die Komplexität sehr niedrig ist. Bei 4000 Voxels ist mehr Arbeitsaufwand einfach oft nicht sinnvoll.

Gradienten

Volumsdaten werden als Funktionen dargestellt. Man hat Koordinaten in R3 und jeweils bei den Positionen einen Skalarwert. Der Gradientennadler macht einen Vektor, der in Richtung der größten Änderung zeigt und dessen Länge beschreibt wie stark die Länge ist.



Das ist wichtig, weil starke Änderungen meist dort sind, wo Übergänge von z.B. Haut zu Luft oder Haut zu Knochen sind. Also Übergänge im Medium.

Die Berechnung dieser Gradienten ist aber nicht so einfach.

Üblicherweise werden sie mit der Zentraldifferenz approximiert.

$$\nabla f(x,y,z) = 1/2 \begin{pmatrix} f(x+1) - f(x-1) \\ f(y+1) - f(y-1) \\ f(z+1) - f(z-1) \end{pmatrix}$$

Also die Differenz der Punkte hinter und vor x. Der Faktor ½ ist notwendig, um die Schrittweite einzurechnen. Weil die Schrittweite bei dem davor und dahinter ja eigl. 2 Voxel wäre. Daher eben mal ½.

Alternativen

- Forward differencing: $\nabla f(x)=f(x+1)-f(x)$
- Backwards differencing: $\nabla f(x)=f(x)-f(x-1)$
- Intermediate differencing: $\nabla f(x+0.5)=f(x+1)-f(x)$

Mit den Gradienten kann man dann Shadings machen.

Klassifizierung

1.) Zunächst muss man semantisch unterscheiden. Was ist Haut, was Knochen, was Muskel, etc. – Was gibt es überhaupt?

2.) Dazu benutzt man die Datenwerte (bspw. Dichte), curvature, gradients… - Was ist was?

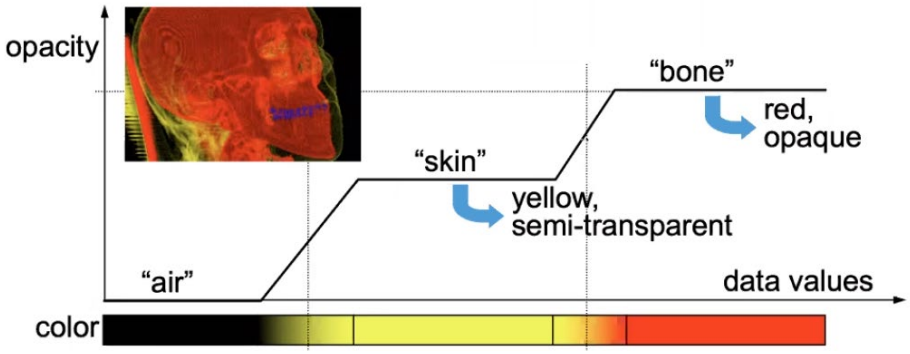
Ziel ist es dabei Bereiche zu segmentiere – bspw. die Knochen.

Oft ist die Zielanwendung dabei semi-automatisch. (Also es gibt schon noch userinput)

Und Approximieren soll automatisch von einer Transferfunktion gemacht werden.

Transferfunktionen

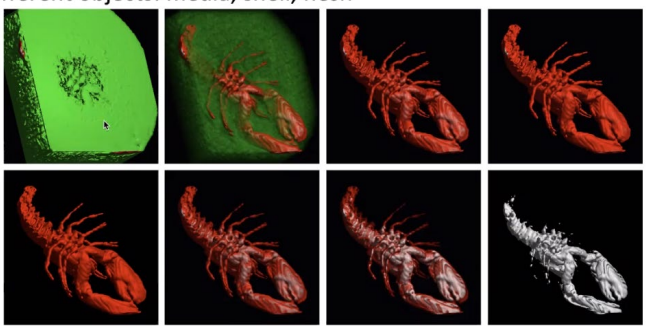
Geben eine Zuordnung für opacity und Farben an.



Man sieht also dass Luft ganz schwarz und durchsichtig ist. Knochen hingegen sind rot und ganz opaque. Haut ist semi-transparent und gelb. Die opacity ist insgesamt an einem Bildpunkt natürlich dann das Gesamtgewicht der Voxels, die der ray durchwandert.

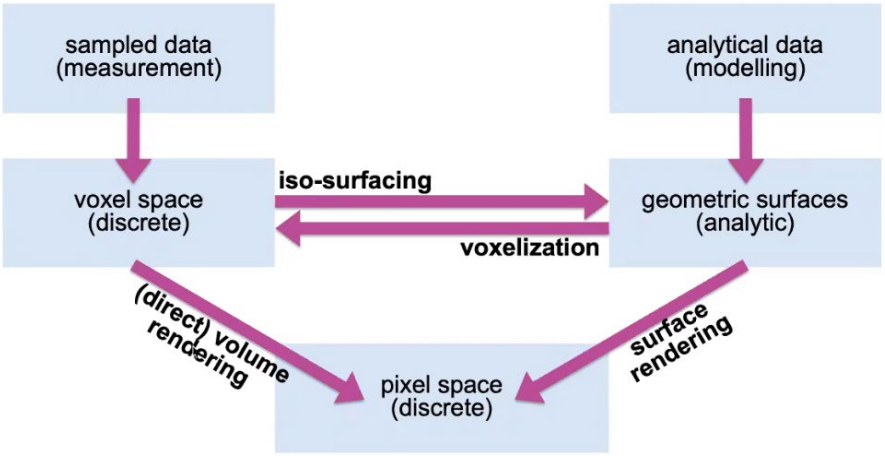
Andere Beispiele:

- Different objects: media, shell, flesh



Konstantin Lackner 19.01.2021

Konzepte und Termini



Hat man gemessene Daten(links oben), kann man damit in den voxel space. Mit direct volume rendering kommt man dann in den pixel space. Hat man hingegen modellierte Daten(rechts oben), kann man daraus geometrische surfaces machen und mittels surface rendering in den pixel space kommen.

Die Verbindung ist einerseits die Voxelisation – also werden innerhalb einer Oberfläche Voxel berechnet. Andersrum kann man mit iso-surfacing aus Voxel eine Oberfläche berechnen.

Beispiele dazu sind

- X-Ray Modelling
- Surface-definition
- Sampling (voxelization), combination
- Direct volume rendering

Slice rendering vs Surface rendering vs Volume rendering

Slice rendering

Hier nimmt man 2D Querschnitte durchs Volumen und stellt diese da.

Surface rendering

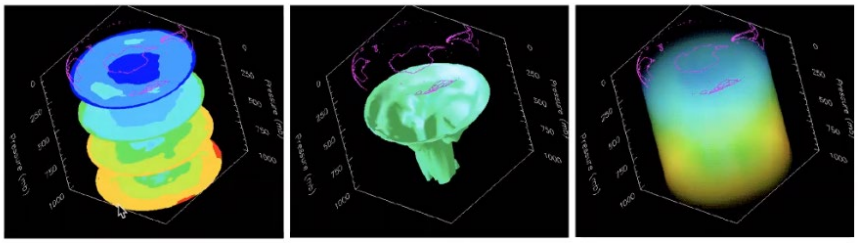
Hier wird hingegen indirekt visualisiert. Man berechnet eine iso-surface in 3D, kann dann shading usw. anwenden und bekommt schließlich mit Hardwarerendering ein Bild.

Volume rendering

Hier benutzt man direkt das Volumen und stellt mittels Transferfunktion die Daten da.

Beispiel dazu:

- Slices: selective (z), 2D, color coding
- Iso-surface: selective (f_0), covers 3D
- Volume rendering: transfer function dependent, "(too) sparse – (too) dense"



Slice macht vier slices und zeigt die Ozonbereiche. Man sieht also nur an diesen Stellen was abgeht.

Die Isooberfläche ist die Oberfläche entlang eines Werts f0. Man schmeißt also alle anderen Daten weg und stellt nur den Wert entlang dieses Werts dar.

Schließlich gibt es noch das Volumenrendering. Da sieht man die gesamten Daten. Allerdings muss man auch eine Transferfunktion definieren. Kennt man, wie bei medizinischen CTs, die Strukturen, man weiß ja wie ein Mensch aussieht, ist das meist sehr leicht. Bei so abstrakten Dingen wie hier, ist es schon schwieriger.

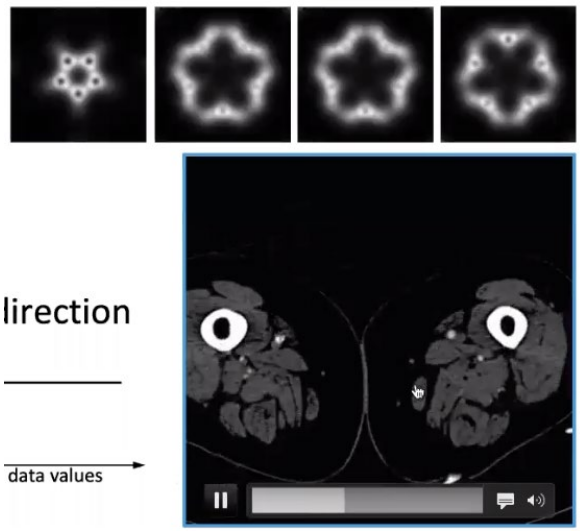
Methoden

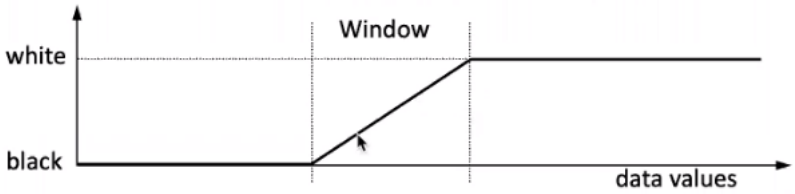
- Simple methods:
  - Slicing, MPR (multi-planar reconstruction)
- Direct volume visualization:
  - Ray casting
  - Shear-warp factorization
  - Splatting
  - 3D texture mapping
  - Fourier volume rendering
- Surface-fitting methods:
  - Marching cubes (marching tetrahedra)

Slicing

Man berechnet achsen-parallele Schnittbilder. Normalerweise braucht man hier keine Transferfunkiton. Für den Kontrast benutzt man meistens ein Window, also einen Übergang von Schwarz zu weiß. Window heißt dabei der Bereich in dem die Rampe ist. Luft wird schwarz = ausgeblendet, weiß = Knochen.

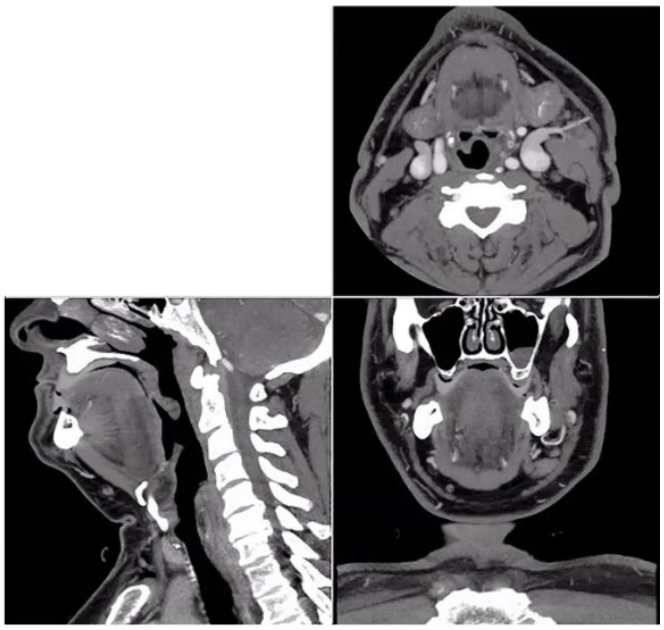
Konstantin Lackner 19.01.2021





Multiplanar reconstruction (MPR)

Man scliced mit verschiedenen Ebenen durch das Volumen und bekommt dann verschiedene Blickwinkel. Manchmal kann man das sogar mit Kurven statt Ebenen, also eigl. gebogenen Flächen machen. Damit kann man z.B. die Luftröhre nachfahren.

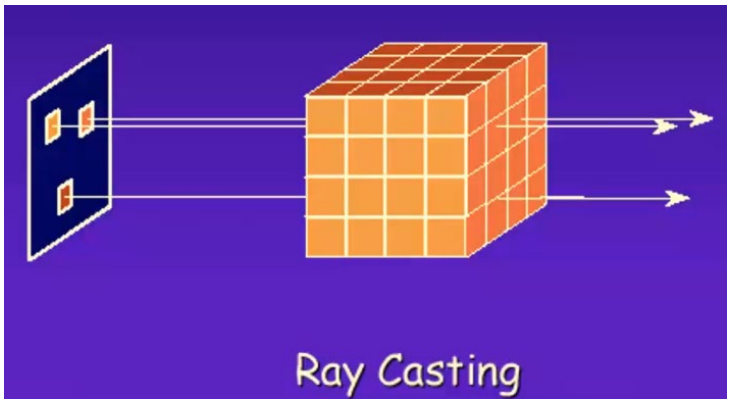


Direct Volume visualisation

Hier gibt es zwei approcahes.

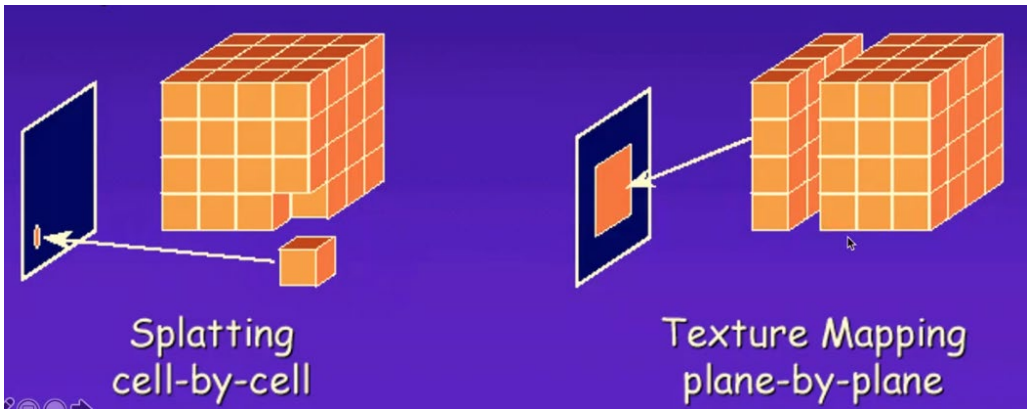
Image-Order approach

Man geht das Volumen Pixel für Pixel durch und berechnet sich die Werte.

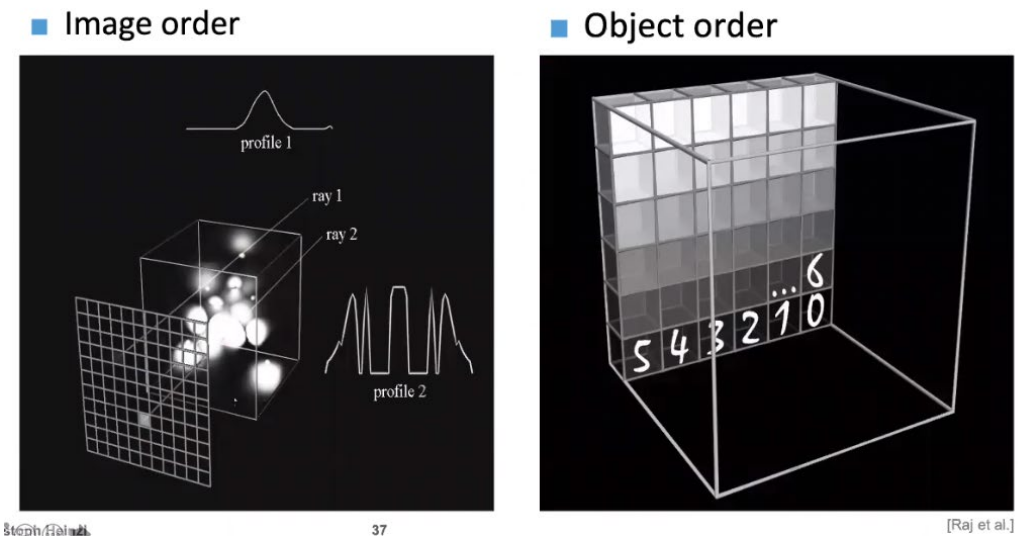


Object-Order approach

Man nimmt einzelne Zellen/features des Datensatzes und projiziert die auf die Image plane. Dabei kann man das echt mit einzelnen Zellen machen = **splatting** – oder man macht es schichtweise = **texture mapping** plane by plane.



Vergleich Object-Order vs Image Order

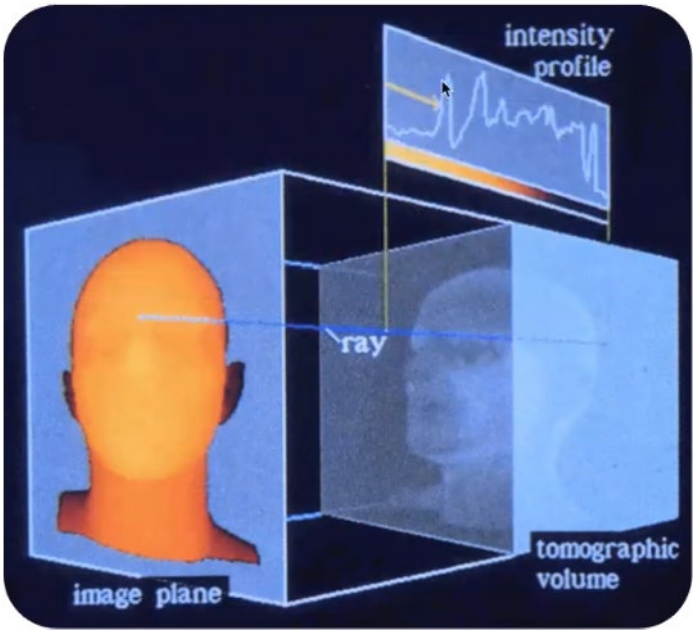


Konstantin Lackner 19.01.2021

Ray Casting vs Ray Tracing

Ray Tracing ist das Verfolgen der Strahlen bspw. in Computerspielen.

Ray Casting hingegen ist das Berechnen von bspw. Dichtewerten entlang eines Strahls. Interpolation braucht man also in jedem Schritt.



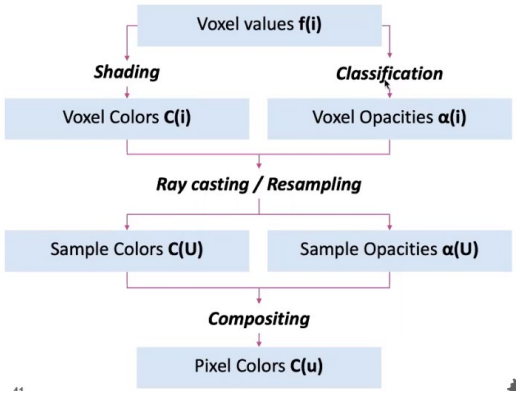
- Volume data: 1D value defined in 3D – $f(\mathbf{x}) \in R^1$, $x \in R^3$
- Ray defined as half-line: $\mathbf{r}(t) \in R^3$, $t \in R^1 > 0$
- Values along Ray: $f(r(t)) \in R^1$, $t \in R^1 > 0$ (intensity profile)

Das Ray profile ist eine Funktion entlang der Halblinie des Rays. Es zeigt also die gemessenen Werte am Weg des Rays.

Konstantin Lackner 19.01.2021

Raycasting Methoden

Standard war beim Ray Casting einst Die Methode nach Levoy'88

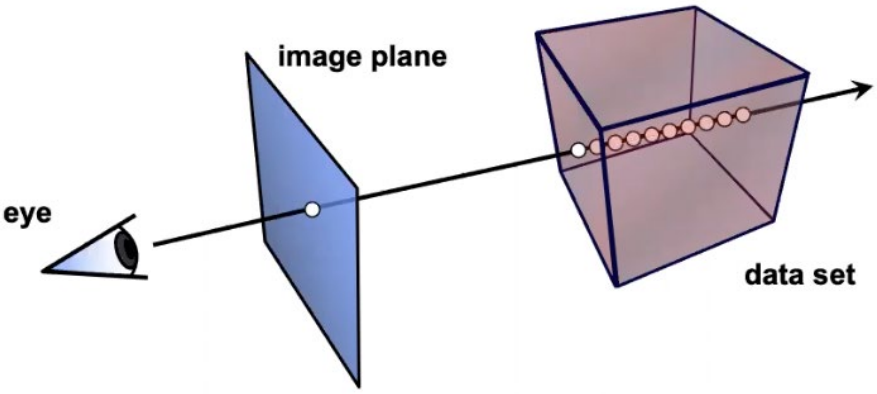


Man geht von drei Komponenten aus. Farben und opacity, das ray profile und schließlich das compositing, also Zusammenführen der Komposition.

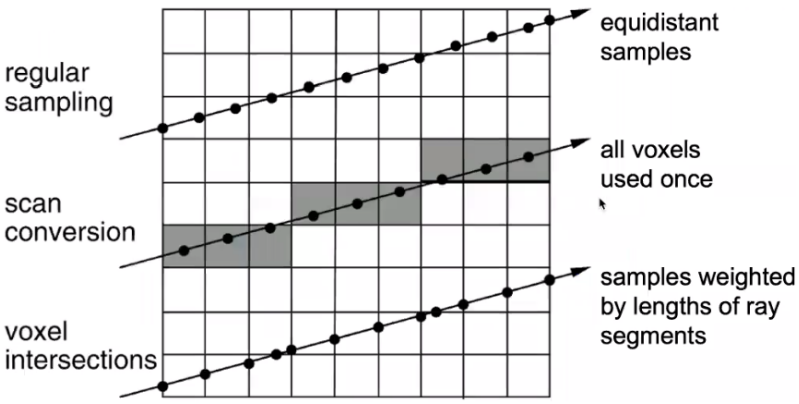
Man startet mit dem Volumen, hat also Voxelwerte.

Dann wird Shading gemacht, also die Farbtransferfunktionen werden zugeordnet und dann auch die opacitytransferfunktionen. Dann hat man eine Funktion entsprechend der Farben.

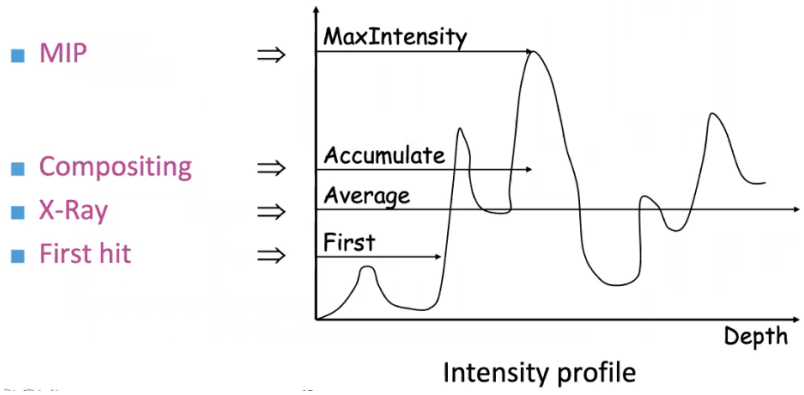
Hat man das, kann man durch das Volumen durchgehen und entsprechend die ray profiles berechnen und schauen, wie die Werte entlang der rays aussehen. C(U) sind dabei die sample positions entlang des Strahls. Das ist das Ray traversal.



Bei diesem Schritt, dem Ray traversal, gibt es wieder drei approaches

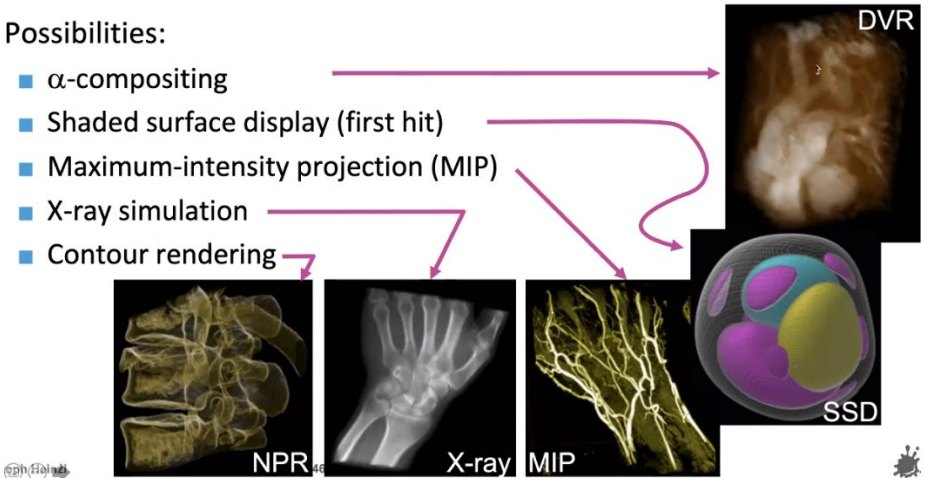


Schlussendlich muss man die Bildfarbe berechnen, dafür gibt es dann wieder verschiedene Methoden.



Intensity profile

Hier hat man Übergangswerte an den steilen Stellen. Also wird hier gezeigt, was was ist. Verschiedene Methoden sind:



Bei first hit benutzt man einfach die erste Position, die einen Schwellwert überschreitet.

Bei x-ray nimmt man den Durchschnittswert.

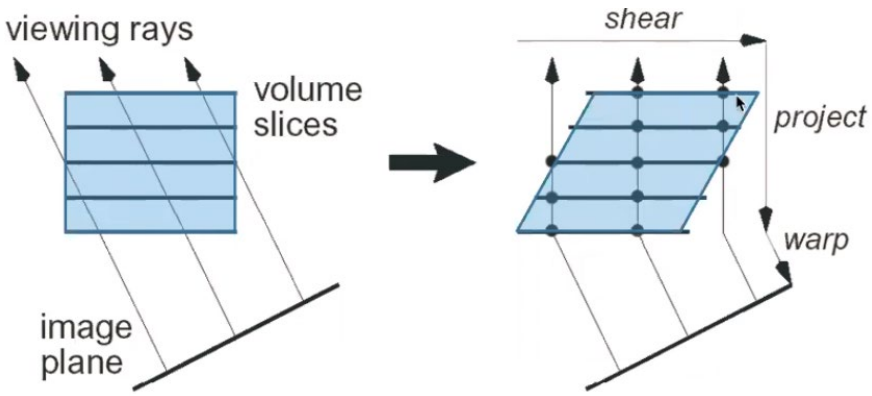
Max intensity nimmt den Maximalwert. Das Problem bei sowas ist, dass man den Sinn für Vorne und Hinten sehr stark verliert, daher braucht man oft Animationen. (drehende Ratte)

Alpha compositing macht transparente layers visible.

Dieser approach (Levoy'88) die Einteilung zuerst und dann erst das ray profile zu machen, ist mittlerweile eher veraltet. Jetzt ist es so, dass zuerst die Daten gesammelt werden und dann erst eingeteilt wird. Der Unterschied ist dabei wann interpoliert und wann zugeteilt wird. Mehr dazu später.

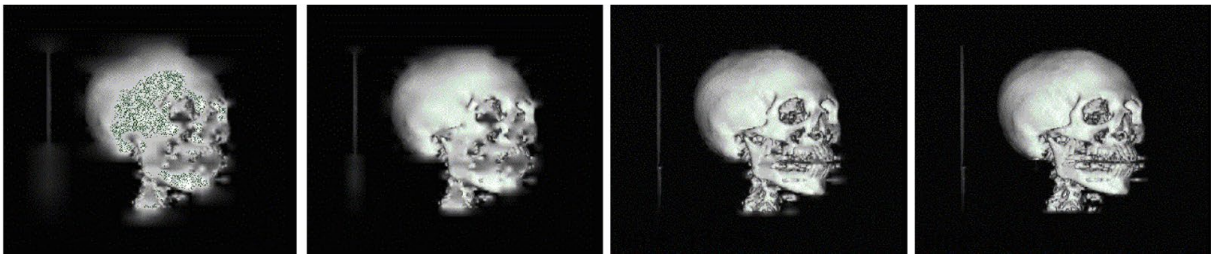
Beschleunigung

Der größte Zeitaufwand ist die Interpolation durch das Volumen. Das geht gut, wenn man genau entlang der Achsen schaut. Wenn aber die rays nicht genau entlang der Achsen liegen, hat man ein Problem. Eine Lösung dafür ist, dass man eine Scherung anwendet.



Problem dabei ist aber, dass man dann das Image schließlich noch warpen muss, nämlich um den Scherwinkel.

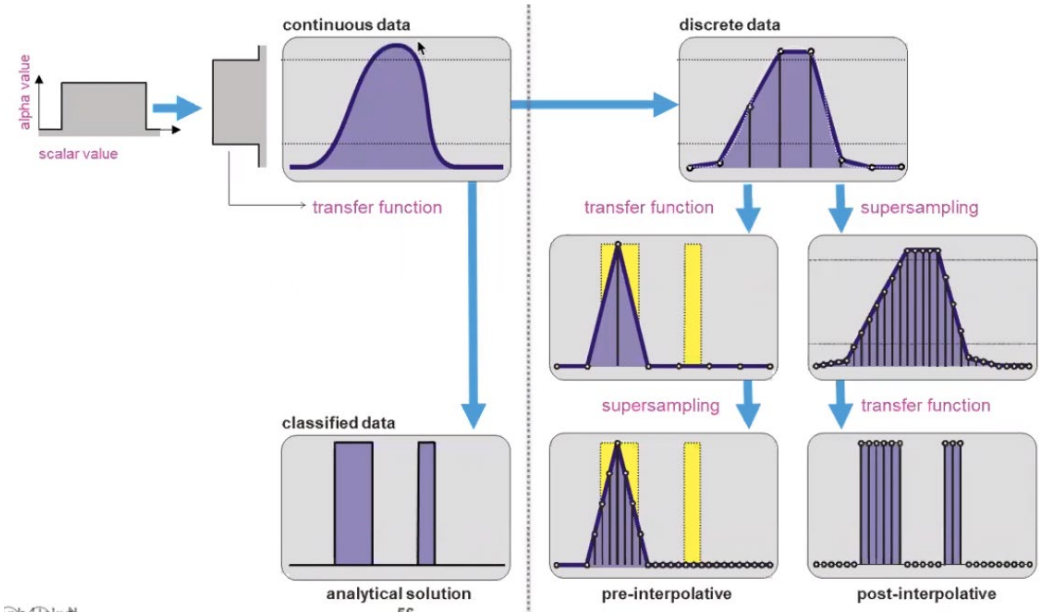
Alternativ kann man progressive refinement benutzen. Man rendert also zuerst nur jedes xte Pixel und verfeinert erst dann. Man berechnet zuerst nicht alle Positionen, sondern kontinuierlich mehr.



Außerdem gibt es splatting. Dabei splattet man jedes Voxel auf die Bildebene (werfen wie ein Schneeball) und dort, wo sich dann mehrere Sammeln, ergibt sich ein Bild. Der splat auf der Bildebene ist ein gausian Radius.

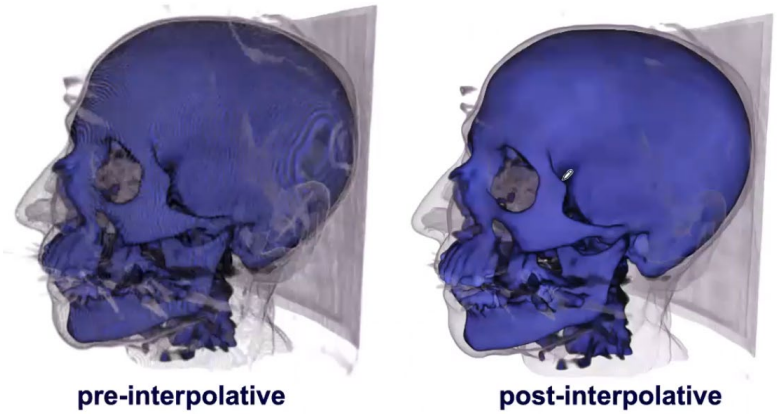
Classification Order

Teilt man wie bei Levoy zuerst allen Werten Kategorien zu, noch bevor man interpoliert, so hat man natürlich andere Ergebnisse, als wenn man wie heute üblich zuerst interpoliert und dann einordnet.



In der analytischen Lösung bekommt man das linke Ergebnis – das ist sozusagen jetzt die ground truth.

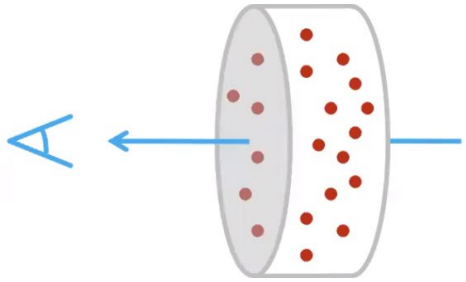
Wenn man sich wie nach Levoy (pre-interpolative) zuerst die Daten ansieht und einteilt, erst dann sampelt, passiert der linke Zweig im rechten Bereich. So geht also der komplette rechte Ausschlag verloren, weil man eben nur an gewissen Punkten, eben den groben Samplingpunkten, einteilt. Hingegen mit der post-interpolativen Methode bekommt man gleich mehr Samplingpunkte, weil man zuerst interpoliert. Teilt man dann ein, bekommt man ein genaueres Resultat.



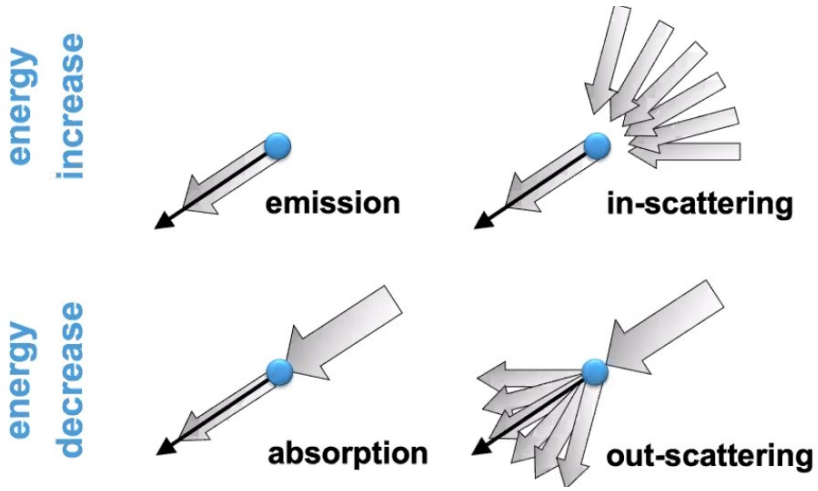
Alpha compostiting

Man bildet Modelle der Natur ab. Schickt wieder Blickstrahlen durch einen Raum und schaut, wie die Partikel im Raum Licht emittieren, dieses Abdecken, scattern, usw..

- Emission only (light particles)
- Absorption only (dark fog)
- Emission & absorption (clouds)
- Single scattering, w/o shadows
- Multiple scattering



Es gibt dabei verschiedene Arten des **radiativen Transfers**, also wie Energie erhöht oder verringert wird.



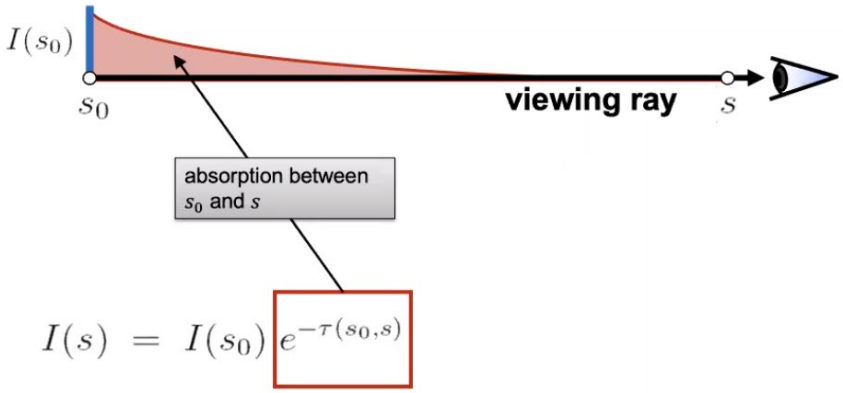
Relevant sind dabei für uns jeweils das erste aus beiden Kategorien. Nämlich Emission und Absorption.

Um solches alpha compositing zu berechnen, gibt es zwei Zugänge. Das analytische Modell und die numerische Approximation.

Analytisches Modell

Wenn man von einem Ausgangspunkt so einen Blickstrahl zum Augpunkt wirft, sendet man am Ausgangspunkt eine initiale Energy/intensity aus. Ohne Abschwächung, also im leeren Raum, würde beim Augpunkt natürlich dasselbe ankommen, wie auch losgeschickt wird. In der Natur aber ist das natürlich anders. Das Medium schwächt solche Lichter ab.

Die Formel lautet hier:

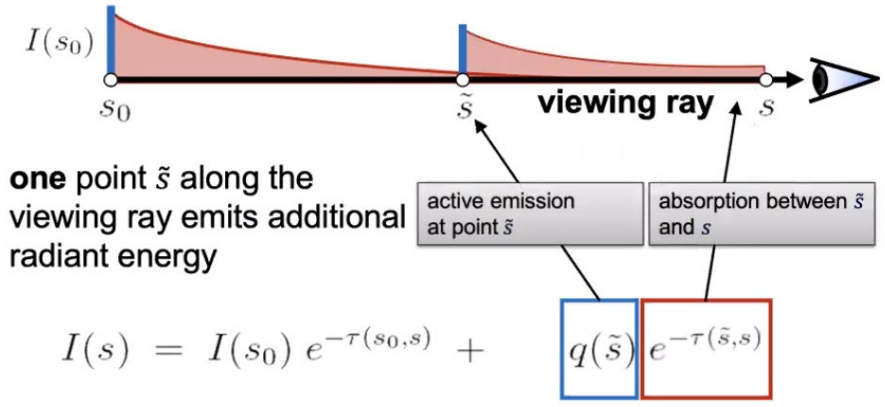


Dieses T (Tau) ist die Abschwächung. Die ist nicht immer gleich, und lässt sich berechnen mit:

Extinction τ
Absorption κ

$$\tau(s_1, s_2) = \int_{s_1}^{s_2} \kappa(s)\, ds.$$

Es gibt auch oft einen Punkt wo zusätzliche Energie/Intensität generiert wird.



q(s) gibt dabei an wie viel Licht ausgesendet wird an der Stelle. Das ist die Farbtransferfunktion in der praktischen Anwendung. Im Endeffekt hat man meist einige

Punkte, die Einfluss haben. Sprich man muss dann für die Endintensität mehrere Werte zusammenzählen:

$$I(s) \;=\; I(s_0)\, e^{-\tau(s_0,s)} \;+\; \int_{s_0}^{s} q(\tilde{s})\, e^{-\tau(\tilde{s},s)}\, d\tilde{s}$$

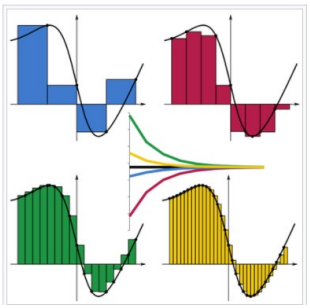
Nummerische Approximation

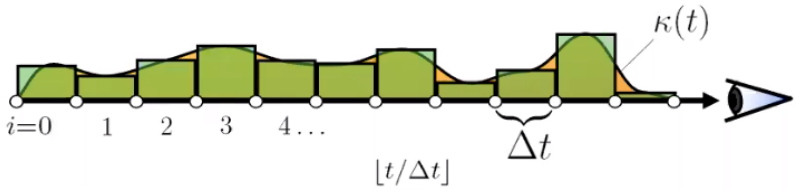
Weil solche Integrale schwer zu berechnen sind, gibt es auch die Möglichkeit das numerisch zu approximieren.

Man hat also ein Medium mit gewissen Absorptionswerten:

Dort wo's höher ist, wird mehr absorbiert.

Das Approximieren passiert durch eine Riemannsumme. Man macht also eine stückweise Approximation.


<-Riemannsumme



$$\tau(0,t) \;\approx\; \tilde{\tau}(0,t) = \sum_{i=0}^{\lfloor t/\Delta t \rfloor} \kappa(i\cdot\Delta t)\,\Delta t$$

Hier wird also an gewissen Punkten gemessen und dann wird einfach damit summiert. Tau Schlange steht dafür, dass es approximiert wurde.

$$e^{-\tilde{\tau}(0,t)} \;=\; \prod_{i=0}^{\lfloor t/\Delta t \rfloor} e^{-\,\kappa(i\,\cdot\,\Delta t)\,\Delta t}$$

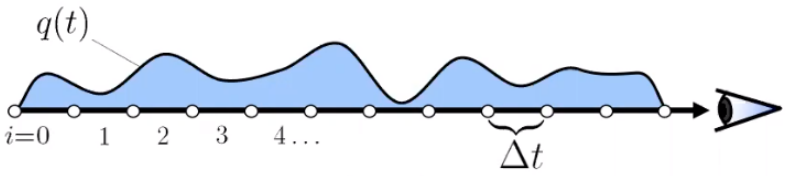
now we introduce opacity:

$$A_i \;=\; 1 \;-\; e^{-\kappa(i\cdot\Delta t)\,\Delta t}$$

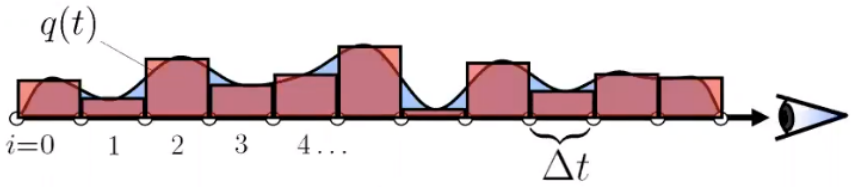
Dann kann man ersetzen:

$$e^{-\tilde{\tau}(0,t)} = \prod_{i=0}^{\lfloor t/\Delta t \rfloor} (1 - A_i)$$

Selbiges kann man auch mit den Farben machen…



q(t) ist die Farbe an einem Punkt/die Stärke des Lichts…whatever



$$e^{-\tilde{\tau}(0,t)} = \prod_{i=0}^{\lfloor t/\Delta t \rfloor} (1 - A_i)$$

$$q(t) \approx C_i = c(i \cdot \Delta t)\,\Delta t$$

$$\tilde{C} = \sum_{i=0}^{\lfloor T/\Delta t \rfloor} C_i\, e^{-\tilde{\tau}(0,t)}$$

und auch hier kann man wieder einsetzen…

$$\tilde{C} = \sum_{i=0}^{\lfloor T/\Delta t \rfloor} C_i \prod_{j=0}^{i-1} (1 - A_j)$$

Und man kann das dann rekursiv berechnen.

$$C'_i = C_i + (1 - A_i)C'_{i-1}$$

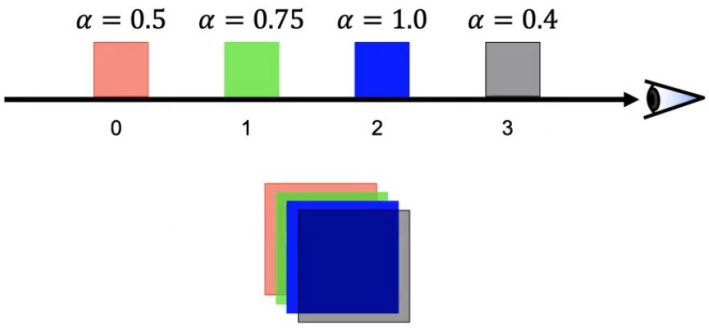
radiant energy observed at position i | radiant energy emitted at position i | radiant energy observed at position $i-1$

Dieses Prinzip nennt man dann **back to front compositing.** Also vom Ursprungspunkt starten, bis zum Augpunkt gehen und jeweils die neue Farbe berechnen. Es ginge auch anders rum:

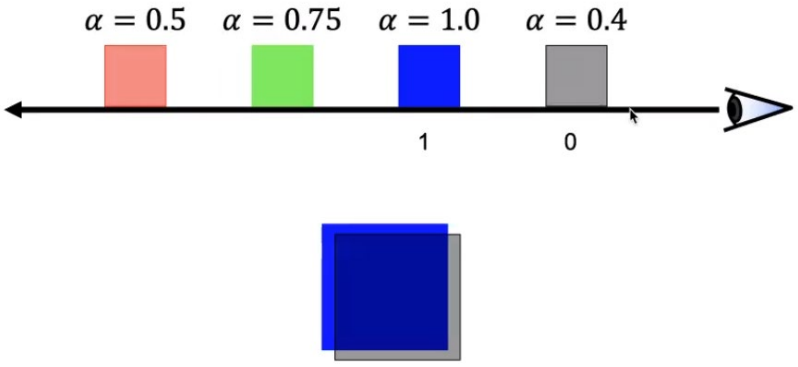
front-to-back compositing

$$C'_i = C'_{i+1} + (1 - A'_{i+1})C_i$$
$$A'_i = A'_{i+1} + (1 - A'_{i+1})A_i$$

Vorteil von **front to back** ist, dass man evtl. früher abbrechen kann. Ist nämlich der Alphawert schon 1, so ist das Ergebnis eh schon nicht mehr durchsichtig und somit würden die Dinge dahinter eh nicht mehr gesehen.



Wenn ich also **back to front** mache muss ich alle Platten zeichnen. Bei **Front to back** würde nur bis zur blauen gezeichnet werden. Da ist nämlich die alphasumme schon über 1 und damit brauch ich gar ned weiter machen.



Analytisches vs. Numerisches Modell

- Emission Absorption Model

$$I(s) = I(s_0)\, e^{-\tau(s_0,s)} + \int_{s_0}^{s} q(\tilde{s})\, e^{-\tau(\tilde{s},s)}\, \mathrm{d}\tilde{s}$$

- Numerical Solutions [pre-multiplied alpha assumed]

back-to-front iteration

$$C'_i = C_i + (1 - A_i)C'_{i-1}$$

front-to-back iteration

$$C'_i = C'_{i+1} + (1 - A'_{i+1})C_i$$
$$A'_i = A'_{i+1} + (1 - A'_{i+1})A_i$$

Beim Analytischen Modell startet man mit einer Anfangsintensität $I(s_0)$, rechnet dann die Abschwächung hinzu von s_0 bis s und dann macht man dasselbe für alle weiteren emittierenden Punkten bzw. Farbwerten entlang des Blickstrahls. Auch diese werden abgeschwächt.

Bei back to front oder front to back werden die einzelnen Werte aufsummiert und mit ihrer opacity abgeschwächt.

Konstantin Lackner 19.01.2021

Pre-multiplied alpha

Man rechnet zuerst die alpha Werte zusammen und rechnet sie dann auf die Farben auf.

Color values are stored pre-multiplied by their opacity: $(\alpha r, \alpha g, \alpha b)$
Consequence: transparent red is the same as transparent black, etc.
Simplifies blending: color and alpha values are treated equally
Can result in loss of precision

Wenn man einen RBG Wert zuerst darstellt und dann mit 0.5 abschwächt, kommt was anderes raus, als wenn man die einzelnen Werte, also R, G und B, mit 0.5 abschwächt und daraus dann eine Farbe kombiniert.

Hardaware-Volume Vis

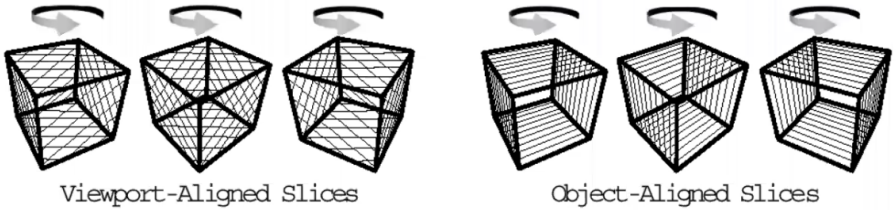
Beschleunigung mit Hardware. Dazu gibt es zwei Zugänge.

3D-textures:
- Volume data stored in 3D-texture
- Proxy geometry (slices) parallel to image plane, are interpolated tri-linearly
- Back-to-front compositing

2D-textures:
- 3 stacks of slices (x-, y- & z-axis), slices are interpolated bi-linearly
- Select stack (most "parallel" to image plane)
- Back-to-front compositing

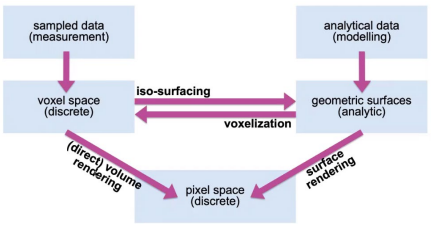
Dabei ist eigl. nur die Aufbereitung interessant. Man hat einfach sllices genommen um nicht-achsen-parallele Sachen zu berechnen.

- 3D-textures:
  - Number of slices varies
- 2D-textures:
  - Stack change: discontinuity



Indirekte Volumen Visualisierung

Hier geht es um Iso Oberflächendarstellung. Also um den Weg von geometric surfaces zu pixel space.



Konstantin Lackner 19.01.2021

Wenn man Iso Oberflächen berechnet, macht man

1.) zuerst startet man von einem CT Datensatz.

2.) dann eine Iso-stack-conversion ()

3.) danach berechnet man die Isooberflächen

4.) schlussendlich rendert man mit openGL

Aber diese Isooberflächen sind stetes eine Zwischenrepräsentation. Man braucht einen Isowert der die Oberflächen definiert und die Interfaces entlang des Volumen repräsentiert. Nachteil ist, dass kleine Änderungen in die Iso-Werte sehr schnell alles verändern.

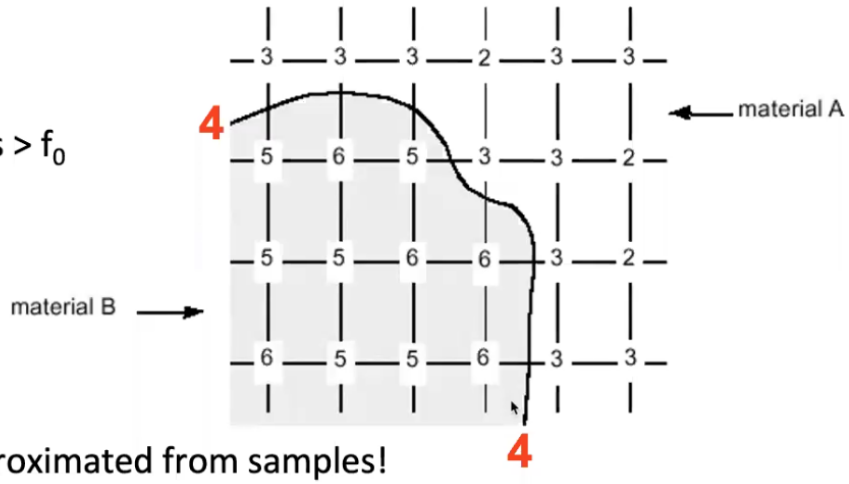
Isooberflächen sind also gut um Transitions gut hervorzuheben.

Aspects:

- Preconditions:
  - expressive Iso-value, Iso-value separates materials
  - Interest: in transitions
- Very selective (binary selection / omission)
- Uses traditional hardware
- Shading ⇒ 3D-impression!

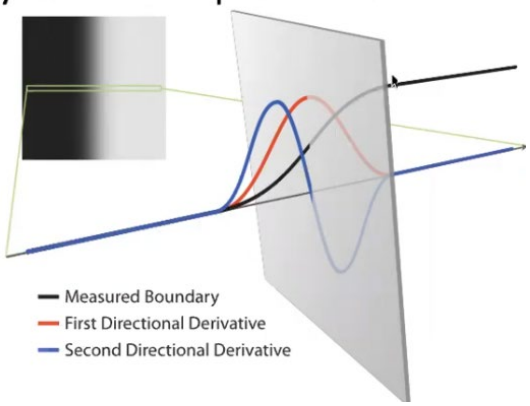
Bei volumetrischen Daten muss man zuerst eine Fläche finden, wo die Isofläche durchgeht. Dazu braucht man einen Grundisowert. Der definiert dann, welche Werte innerhalb des Objekts sind und welche außerhalb.

- Iso-Surface:
  - Iso-value f_0
  - Separates values > f_0 from values ≤ f_0
  - Often not known →
  - Can only be approximated from samples!
  - Shape / position dependent on type of reconstruction



Eben deswegen ist eine kleine Änderung schon sehr ausschlaggebend. Wenn der Isowert ein bisschen höher gewählt wird, fehlt gleich ein Stück des Modells. Diesen Wert muss man natürlich approximieren. Das macht man indem man sich die Isowerte als Grauwerte ausgibt. Also für jede Achse ein Grauwertprofil macht und dann nachschaut, wo die Änderung am größten ist.

- Iso-value f_0 interesting at transitions
- Typically greyvalue at steepest transition

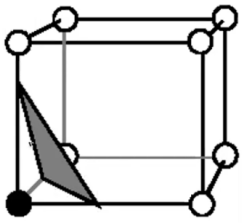


Man leitet also das Grauwertprofil ab und schaut sich das Maximum an. Den Wert an dieser Stelle nimmt man dann als Startisowert.

Extrahieren der Iso Oberflächen

Marching Cubes

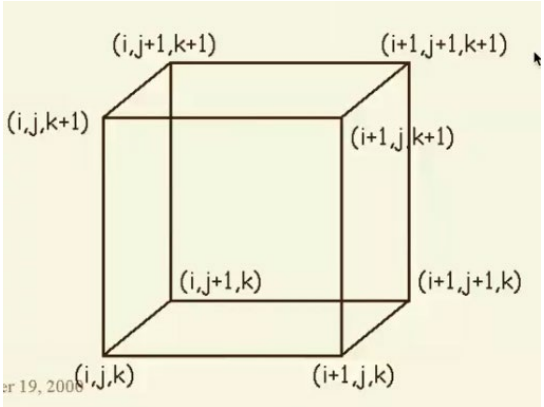
Dabei wird nichts anderes gemacht, als das Volumen in Zellen eingeteilt. (vgl. Voxel vs. Zellen) In jeder Zelle versucht man dann zu extrahieren, wie die Oberfläche durch die jeweilige Zelle durchgeht.



Das wird mit folgenden Schritten gemacht:

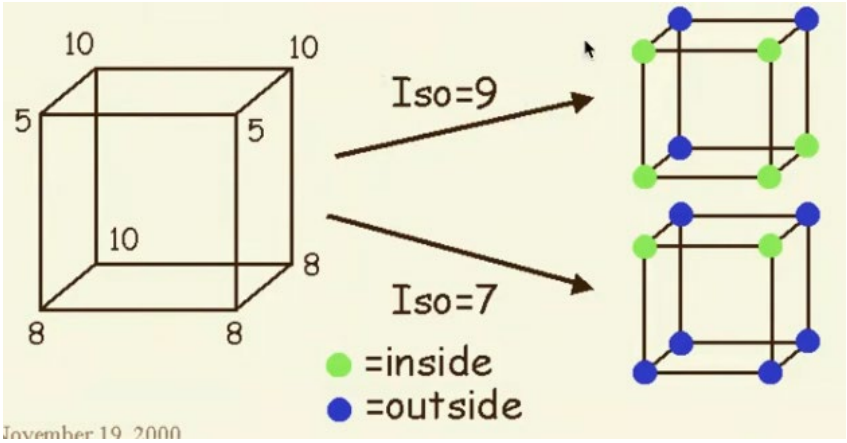
Volumen in Zellen einteilen – bei 3D also 8 Voxelwerte.

Man definiert den Cube also in seine 8 Ecken.



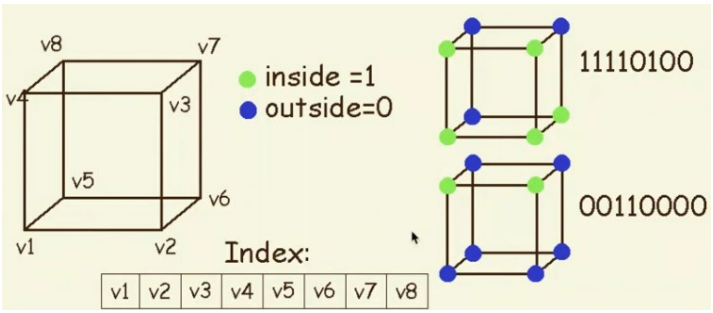
Anschließend schaut man, ob die Eckpunkte innerhalb oder außerhalb der Fläche liegen.

Man schaut also den Isowert am Eckpunkt an und schaut, ob der höher oder niedriger ist als der Startisowert.



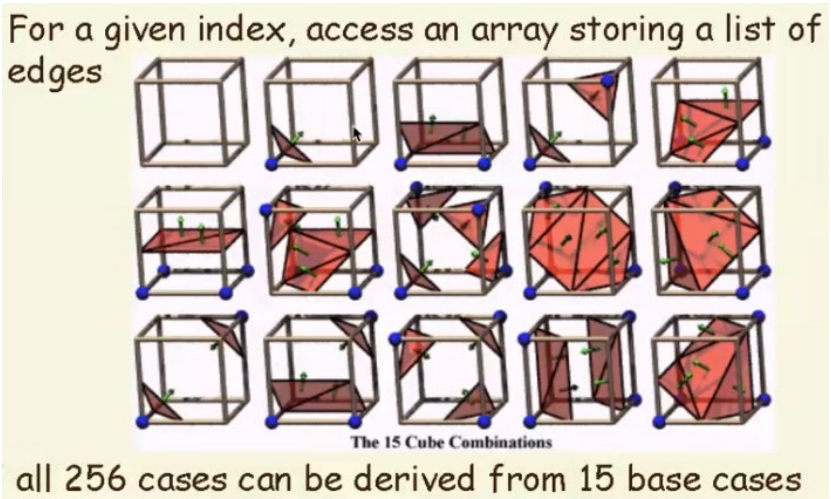
Dann setzt man den Index auf 1 oder 0 je nachdem, ob die Punkte drinnen sind.

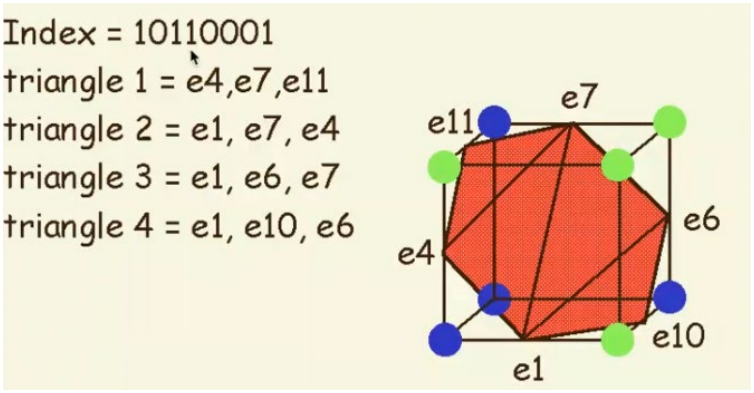
Der index wird dann ein 8 Stellen langer Binärwert.



Dann wird die edgelist aufgerufen und man schaut für die Indexwerte, was passt.

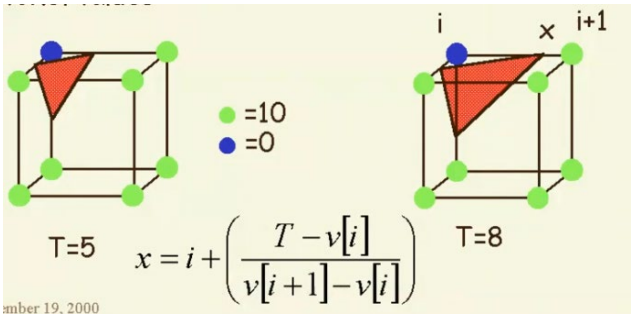
Die Edgelist ist dabei eine Liste an möglichen Flächen im Würfel. Es gibt 256 Möglichkeiten, wie die Fläche im Würfel liegt. Man kann aber auf 15 base cases runterbrechen und diese dann rotieren und co.. Man schaut dann einfach, welcher base case zum Index passt.



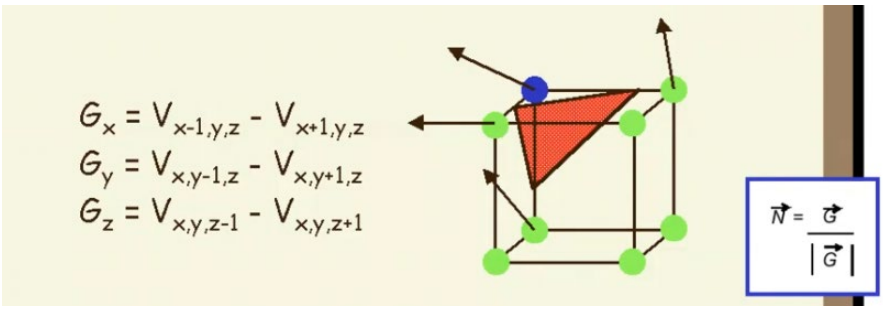


Dann muss man sich anschauen, wie das Dreieck genau durch die Zelle geht.

Man schaut also nochmal die Werte an den Punkten an und verschiebt dann nochmal anhand des Datenwerts. Es ist also, wie so oft, eine lineare Interpolation.

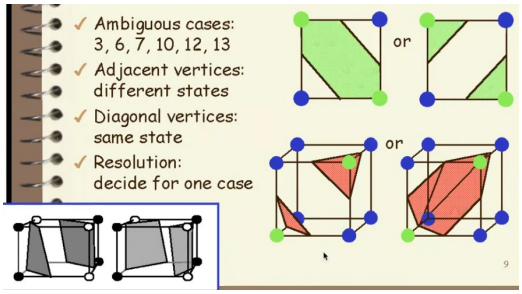


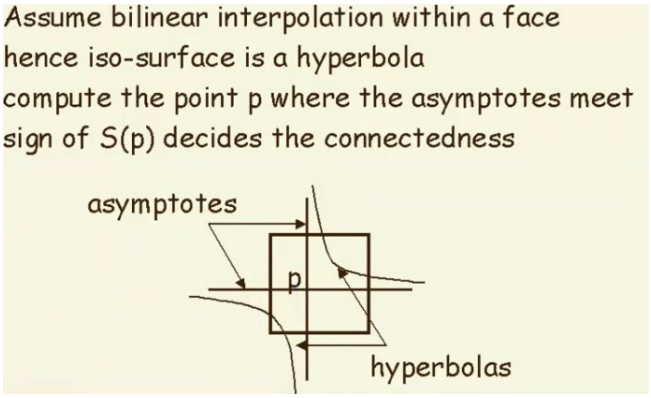
Dann muss man dann noch die Normalen berechnen. Das geht, indem man den Gradienten nimmt und wie folgt rechnet:



Schlussendlich muss man noch mehrdeutige Werte bereinigen.

Um zu entscheiden, schaut man sich einen asymptotischen decider an. Man legt eine Hyperbel durch und schaut, ob die Asymptoten der Hyperbeln an und bestimmt dann, ob der Schnittpunkt dieser innerhalb oder außerhalb der Fläche ist. (Nicht, dass das iwer verstehen würde…)

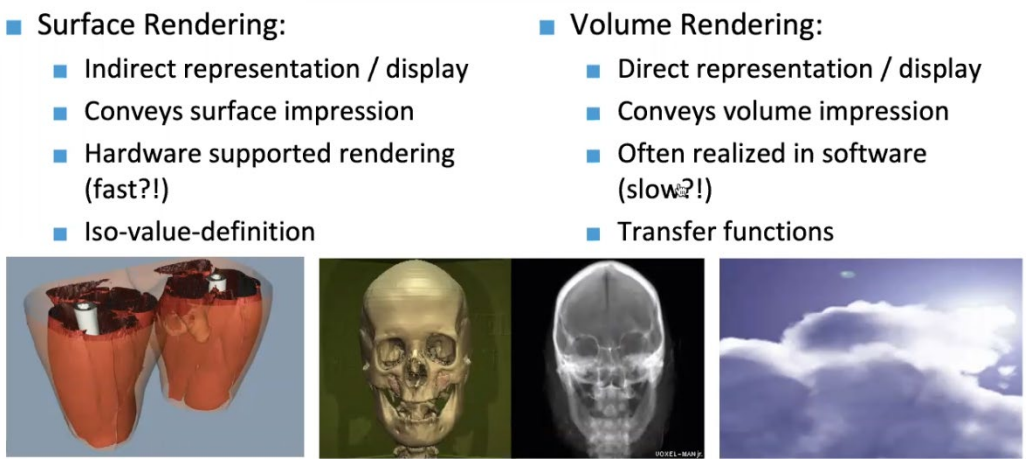




Zu beachten

Man kann pro Zelle bis zu 4 Dreiecke generieren, also kann das alles sehr groß werden. Außerdem handelt es sich nicht um Messungen, sondern Approximationen.

Surface vs. Volume rendering



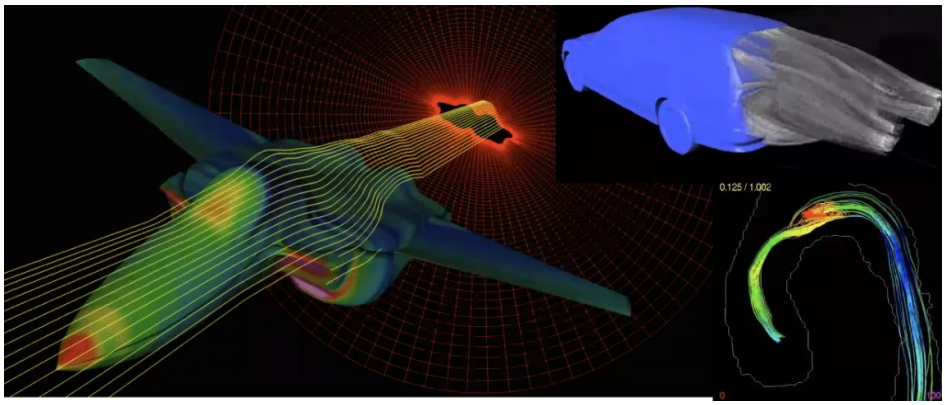
6. Flow Vis

Einleitung

Meistens um Veränderungen darzustellen. Veränderungen eben in Strömungen usw..

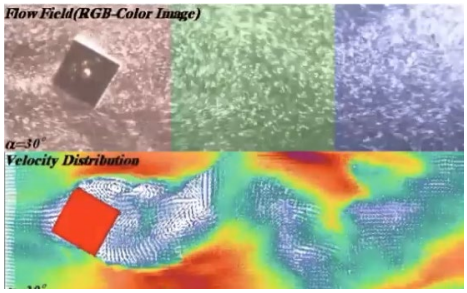
Man hat meist mehr als drei Dimensionen und die Daten sind komplexer als jene aus Volvis. Man hat nDxnD Daten. Im Endeffekt meist Vektordaten (nD) im nD Datenraum.

Man hat oft iso surfaces und zeichnet ein, wo wie viel Druck entsteht bspw.

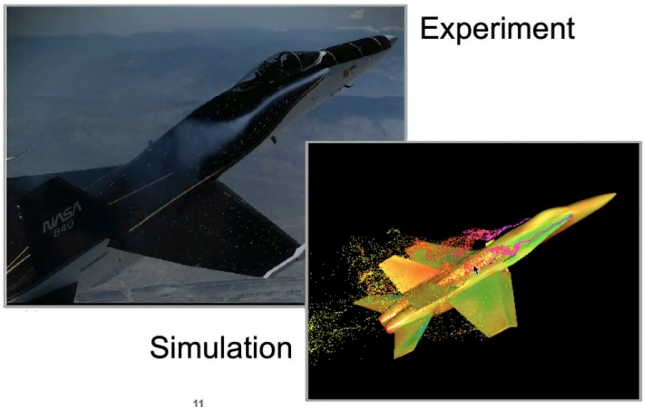


Auch bei Autos wird der Windwiederstand gemessen. Man schaut auf Verwirbelungen hinter dem Fahrzeug. Diese sind schlecht, weil sie mehr Treibstoffverbrauch erzeugen. Unten rechts sieht man noch die Simulation eines Blutgefäßes.

Man kann Flussmessungen machen, indem man in das Medium leuchtende Punkte einführt und deren Position dann trackt.



Man unterscheidet zwischen Experiment und Simulation.



Dazu benutzt man bei Experimenten oft Öl oder Rauch als Flussexperiment.

Konstantin Lackner 19.01.2021

2D vs 3D Visualisierungen

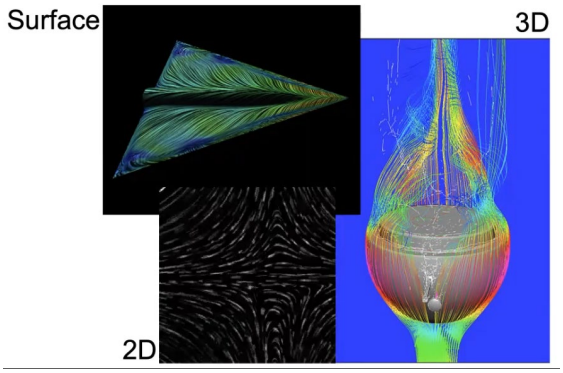
Bei 2Dx2D Flows hat man einen 2D Fluss in einem 2D Bild mit Flussinfo als 2D Vektoren.

Ebenso kann man das mit mehreren slices machen um ein 3D slice flow Ding zu bekommen.

Man kann aber auch auf der Oberfläche den Fluss kodieren. Das geht in 2D oder 3D.

Schließlich kann man auch 3D Simulationen machen.

Beispiele:



Bei der Oberfläche hat man eben auf der Oberfläche eine Strömung, bei 2D einfach Vektoren in 2D und bei 3D hat man Strömungslinien im 3D Bereich um ein Objekt.

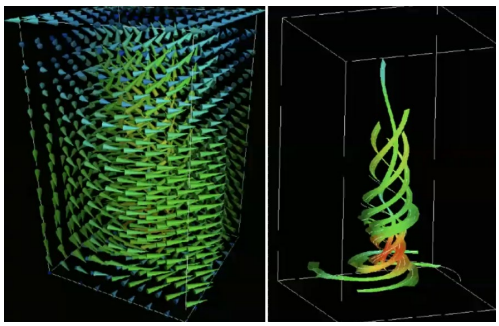
Stetig vs. zeitabhängig

Es gibt stetige Strömungsfelder und auch zeitabhängige Strömungsfelder. Stetige Strömungsfelder verändern sich mit der Zeit nicht. Zeitabhängige eben schon. Zeitabhängige flows werden oft mit Animationen und bspw. Pfeilen dargestellt. also Pfeile, die ihre Richtung über die Zeit verändern. Hingegen steady flows kann man über Bilder und fix abbilden.

Direkte und indirekte Möglichkeiten

Bei der direkten Vis wird ein Überblick über das ganze Flussfeld dargestellt, meist mit Vektorpfeilen und smearing techniques.

Bei indirekten wird hingegen meist die zeitliche Änderung gezeigt durch Integrale über Strömungsfelder usw.



Das ist derselbe Datensatz. Links direkt, rechts indirekt.

Experimentelle Methoden um Flowdaten zu generieren

Man kann Partikel in ein System einführen und dann mit einer Kamera deren Bewegung tracken. Das nennt man **particle image velocimetry PIV**.

Man kann aber auch mit Farben arbeiten. Ebenso mit Viskositäten und Farben (Verwirbelungen in flüssigen Medien bspw.)

Man kann aber auch Bilder aus Bildsequenzen korrelieren, also schauen, wie sich was ändert.

Ebenso kann man mit Flugzeugen durch Nebelwände fliegen und die Verwirbelungen betrachten.

Visualisierung von dynamischen Systemen

Man hat Veränderungen von Positionen über Zeit. Es geht hier auch wieder um flow vis.

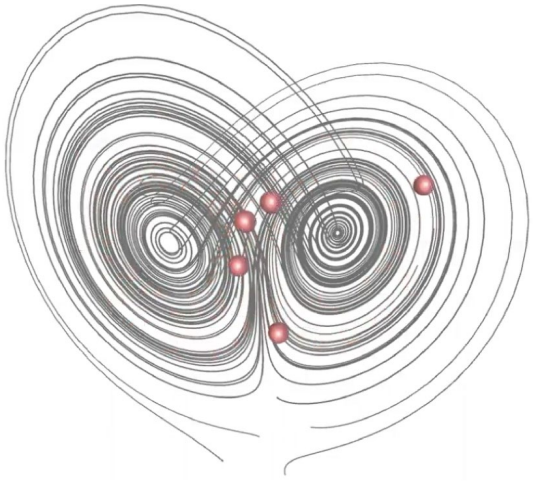
Man hat bspw. dx/dt = v(x) und generiert dazu Strömungsdaten. Man hat es dabei mit **Navier-Strokes Gleichungen** zu tun, das sind Gleichungen die Strömungen in Newtonschen Flüssigkeiten oder Gasen modellieren. Das sind Systeme nicht linearer Differentialgleichungen zweiter Ordnung.

Außerdem gibt es das **Lorenz-system**, dabei hat man Veränderungen über Zeit in verschiedenen Koordinaten. Es ist deswegen interessant, weil es sich dabei um Attraktoren handelt (was auch immer) – man hat gekoppelte, nicht-lineare, gewöhnliche Differentialgleichungen. Diese werden für Vorhersagen in der Meteorologie verwendet.

$$dx/dt = \sigma(y-x)$$
$$dy/dt = rx-y-xz$$
$$dz/dt = xy-bz$$

(Lorenz-system)

Randbemerkung – Fixpunkte in Strömungsfeldern sind Punkte ohne Strömung, also Geschwindigkeit 0. Interessant sind diese Punkte und was in ihrer Nähe passiert. Man schaut also wie sich die Strömung im Bereich dieses Attraktors verhalten. Attraktoren dürften iwie die Zentren sein… (Würd Sinn machen, die Punkte scheinen anzuziehen)

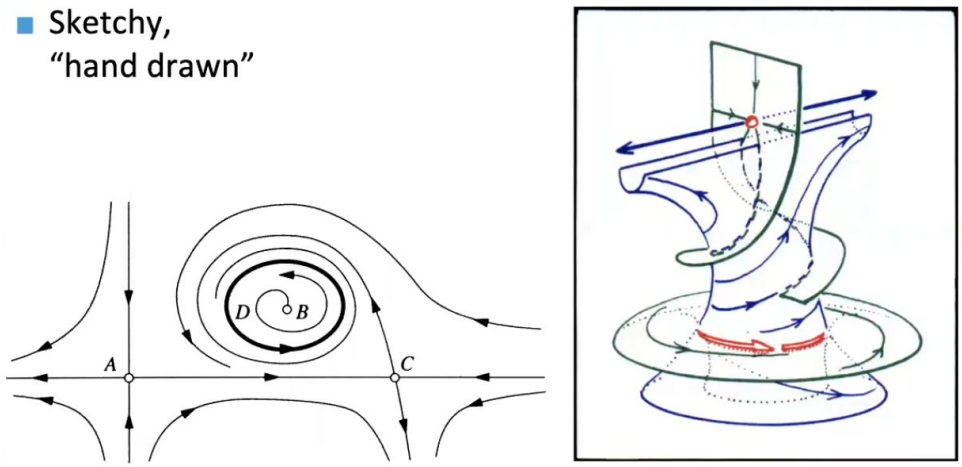


Die Attraktoren dürften hier die Zentren sein. Ein **Lorenzattraktor** ist eng verbunden mit dem **Butterflyeffect**, meint er… Kleine Änderungen haben massive Auswirkungen auf das Gesamtsystem. Für Lorenzsysteme ist es so, dass die Kugeln im Bild nie im Ausgangspunkt wieder ankommen. Es ist also **nicht periodisch**, aber **deterministisch**. Ich glaub er will sagen, dass es keine „Lösung“ gibt. Es ist ein chaotisches System und insofern auch interessant bei Meteorologie.

Beschreiben von solchen Systemen

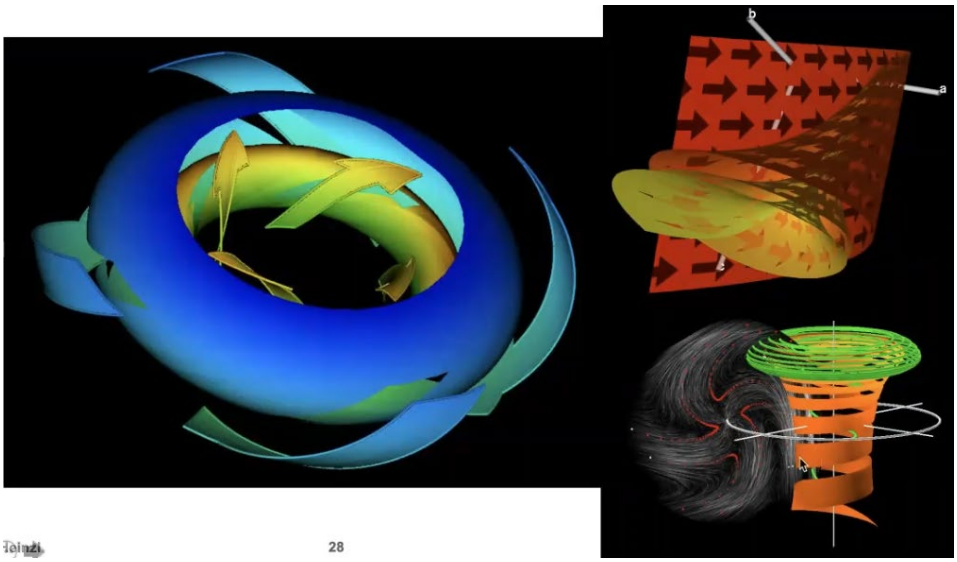
Man kann den sketchy approach wählen, also handzeichnen.

Man sucht zuerst die zwei Fixpunkte.



Also dort, wo man die Geschwindigkeit 0 hat. Dann verbindet man die Fixpunkte und damit bekommt man Sektoren. Im linken Bild verbindet man A und C. Die Linien, die die Regionen trennen sind Separatritzen. Die trennen völlig verschiedene Strömungsverhalten. In der Mitte im linken Bild hat man einen Orbit um DB. Der Fluss um den Orbit bleibt erhalten und wird sich nie ändern.

Andererseits kann man auch 3D Modelle erstellen.

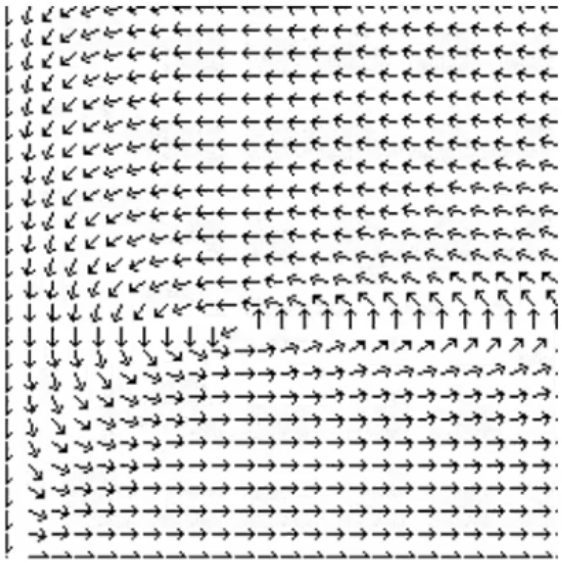


Rechts oben stream surfaces, rechts unten das Grüne stream lines, drunter die Bändern in Orange stream ribbons und das Graue ist line integral convolution. Links wurde wieder mit Pfeilen gearbeitet.

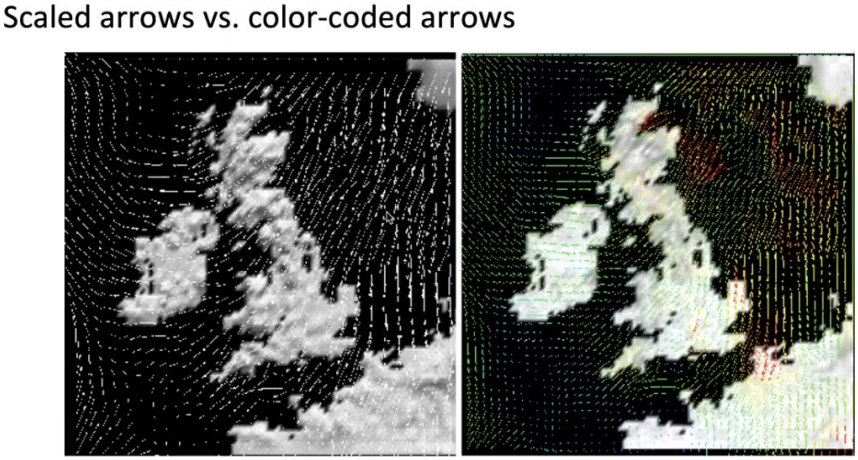
Konstantin Lackner 19.01.2021

Strömungsfeldvisualisierung mit Pfeilen

Es gibt wieder direkte und indirekte Verfahren. Ein direktes Verfahren ist das Darstellen mit Pfeilen. Man hat also ein reguläres Grid und kann an jeder Stelle eine Pfeilposition wählen. Dabei kann man die Länge der Pfeile entweder normalisieren, oder man passt die Länge an die Strömungsstärke an. In 2D ist das ganz einfach, weil der 1D Pfeil gut auf 2D legbar ist. Bei 2D wird das schwieriger, weil man das ja iwie räumlich präsentieren müsste (bspw. Zylinder mit Torus und shading). Außerdem muss man sich festlegen, welche Pfeile man visualisieren will, sonst sieht man nicht mehr was im Inneren passiert, weil der äußere Bereich alles abdeckt. Das nennt man Overplotting.

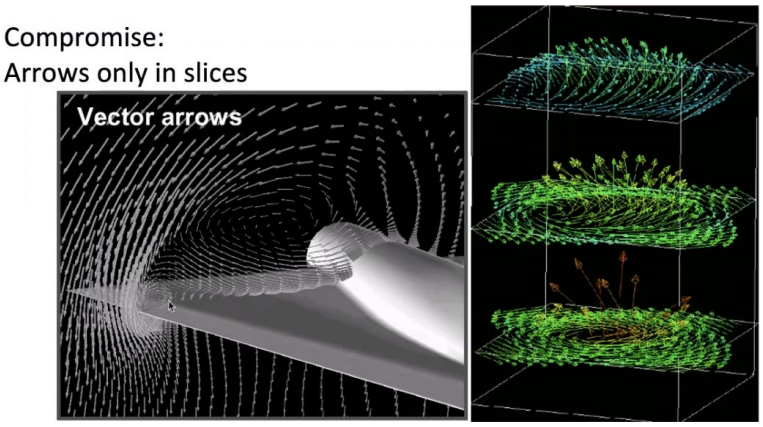


So etwas kennt man aus Wetterberichten



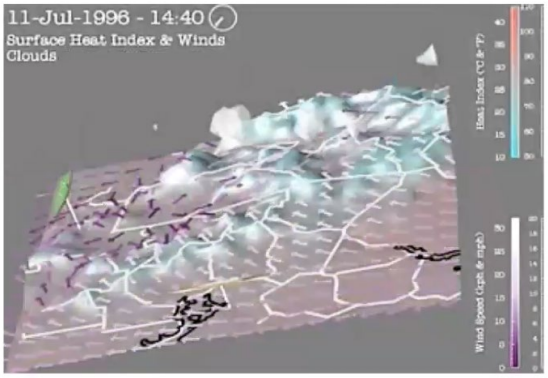
Oft eben **overplotting** (man zeichnet ein Objekt übers andere) oder **visual clutter** – einfach viele Objekte, was das Verstehen schwer macht. Bei diesen Darstellungen hat man auch oft im Hintergrund, als Referenz Basisdaten, bspw. topologische wie hier Großbritannien. (Ja, das sind keine Wolken, who knew)

Um solche Probleme mit cluttering und co zu vermeiden, kann man bspw. Ebenen einfügen:



Das macht's bissl besser.

Man kann auch auf Topologien Pfeile einfügen, das macht das Bild auch verständlicher.



Die Referenzgeometrie ist dabei die Geometrie unten, also die Landschaft.

Solche Lösungen kann man auch in der echten Welt zur Visualisierung nutzen. Bspw. kann man Fäden an ein Auto in einem Windtunnel kleben und hat dasselbe Ergebnis.

Streamlines

Hier haben wir jetzt eine indirekte Strömungsvisualisierung. Man hat flow data die räumliche Änderung über Zeit sind. Man generiert dann Linien daraus indem man Integriert.

$$\mathbf{s}(t) = \mathbf{s}_0 + \int_{0 \le u \le t} \mathbf{v}(\mathbf{s}(u))\,du;$$

seed point $\mathbf{s}_0$, integration variable u

s(t) ist die aktuelle Position der Strömungslinie. Man startet also vom Startpunkt s0 und geht bis zum Zeitpunkt t. Das Problem damit ist, dass das Resultat s auch im Integral vorkommt. Damit muss das Ganze approximiert werden, weil eine genaue Lösung eben nicht möglich wäre. Das macht man mit **Eulerintegration**. Heißt man geht vom Startpunkt aus, integriert über das Vektorfeld und geht jeweils in die Richtung s_i für einen sehr kurzen Schritt und berechnet sich dann damit den jeweils nächsten s_i Punkt.

$$\text{Euler integration: } \mathbf{s}_{i+1} = \mathbf{s}_i + dt \cdot \mathbf{v}(\mathbf{s}_i),$$

integration of small steps (dt very small)

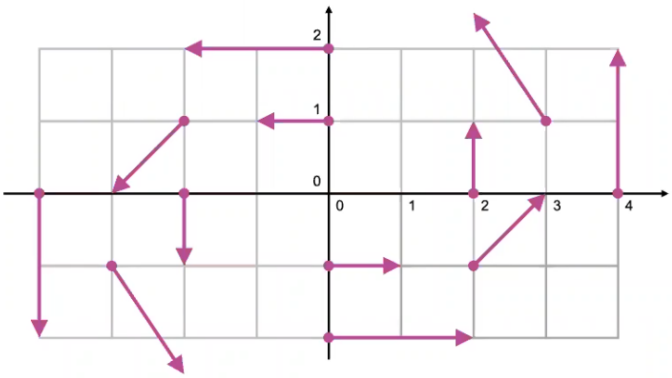
dt ist der Zeitschritt. Man wählt das möglichst klein für ein sehr genaues Ergebnis.

Beispiel dazu:

2D model data: $v_x = dx/dt = -y$
$v_y = dy/dt = x/2$

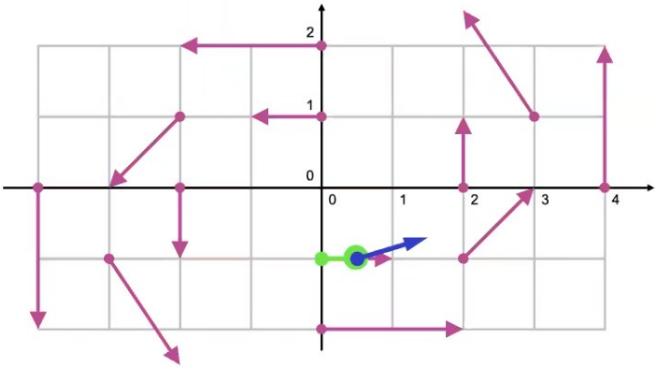
Sample arrows:

True solution: ellipses!



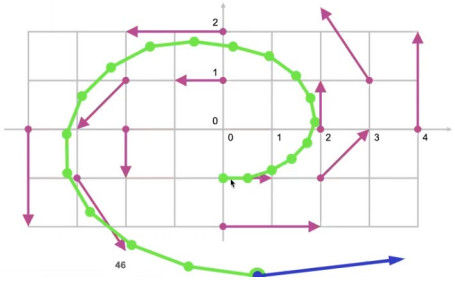
Man hat hier in rosa Strömungsvektoren und die Lösung wäre eine Ellipse. Benutzt man jetzt die Eulerintegration, startet man bei (0, -1) und schaut dort den Vektor an der Position an. Der zeigt zu (1,0). Man geht dabei einen halben Schritt vor, zu (½,-1). Dort schaut man auch wieder:

New point $s_1 = s_0 + v(s_0) \cdot dt = (1/2 \mid -1)^T$;
current flow vector $v(s_1) = (1 \mid 1/4)^T$;



Das macht man einfach immer so weiter, bis man den Fluss komplett verfolgt hat und kommt iwann aber relativ falsch weit draußen raus.

$s_{19} \approx (0.75 \mid -3.02)^T$; $v(s_{19}) \approx (3.02 \mid 0.37)^T$;
clearly: large integration error, d*t* too large!
19 steps

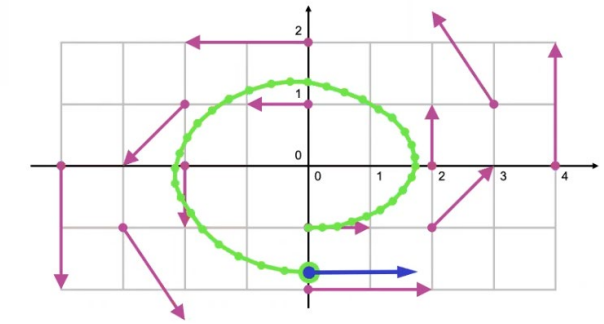


Man wertet ja immer genau an einer Position aus und geht dann in die Richtung. Im Kleinen is das egal, aber zusammen wird das ein ordentlicher Fehler. Eine Lösung um das besser zu machen, wäre, wenn man mehr Schritte wählt.

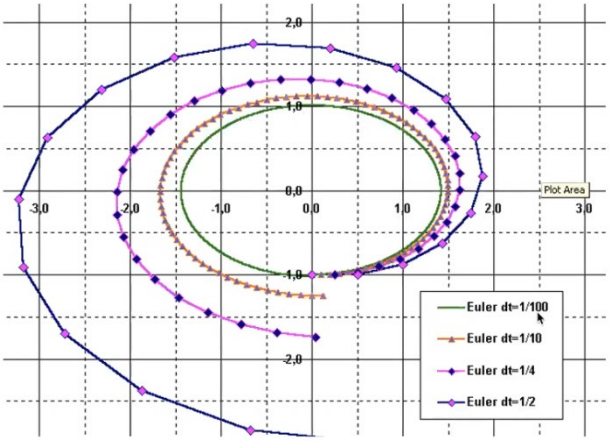
dt smaller (1/4): more steps, more exact!

$\mathbf{s}_{36} \approx (0.04 \mid -1.74)^T$; $\mathbf{v}(\mathbf{s}_{36}) \approx (1.74 \mid 0.02)^T$;

36 steps



Nachteil ist halt, dass man mehr Schritte braucht.



Beim Hundertstelschritt erreicht man schon wieder fast den Ausgangspunkt genau.

dt	#steps	error
1/2	19	~200%
1/4	36	~75%
1/10	89	~25%
1/100	889	~2%
1/1000	8889	~0.2%

Alternativ Runge-Kutta Approach

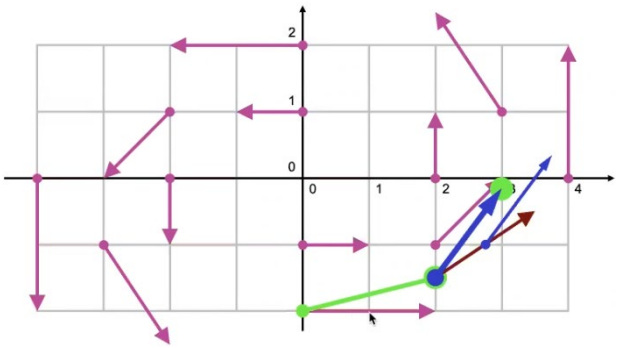
Versucht das besser als die Eulerintegration zu machen.

Man versucht einen Halbschritt vorauszuschauen, dort den Vektor anschauen und dann den verwenden für die aktuelle Position, anstatt den wirklichen.

RK-2 (two evaluations of **v** per step):

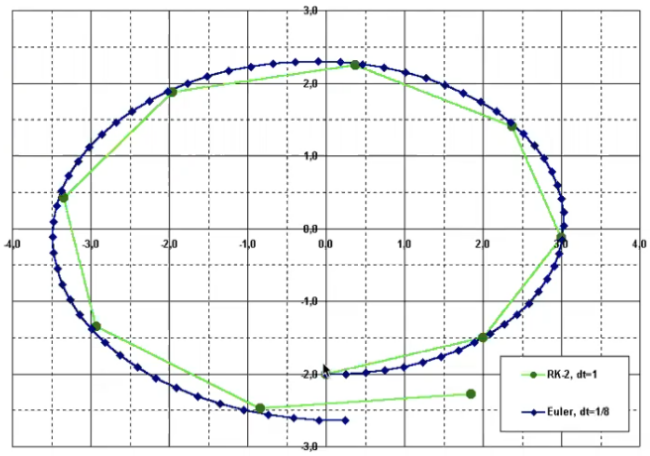
$$\mathbf{s}_{i+1} = \mathbf{s}_i + \mathbf{v}(\mathbf{s}_i + \mathbf{v}(\mathbf{s}_i)\cdot dt/2)\cdot dt$$

Seed point $\mathbf{s}_1 = (2 \mid -1.5)^T$;
current flow vector $\mathbf{v}(\mathbf{s}_1) = (1.5 \mid 1)^T$;
preview vector $\mathbf{v}(\mathbf{s}_1 + \mathbf{v}(\mathbf{s}_1)\cdot dt/2) \approx (1 \mid 1.4)^T$;
$dt = 1$



Man nimmt also zuerst den echten Vektor, geht den einen Halbschritt vor, nimmt dann den dortigen Vektor und setzt den dann schließlich am ursprünglichen Punkt ein.

RK-2: even with dt=1 (9 steps)
better
than Euler
with dt=1/8
(72 steps)



Bei RK braucht man nur dt = 1 um schon ein Ergebnis zu bekommen, das ähnlich gut wie Euler mit 1/8 ist.

Verschiedene Arten von Lines in Vis

Streamlines – Tangential zum Vektorfeld

Pathlines – Trajektorien von masselosen Partikeln im Flow (also ohne Wiederstand, deswegen masselos)

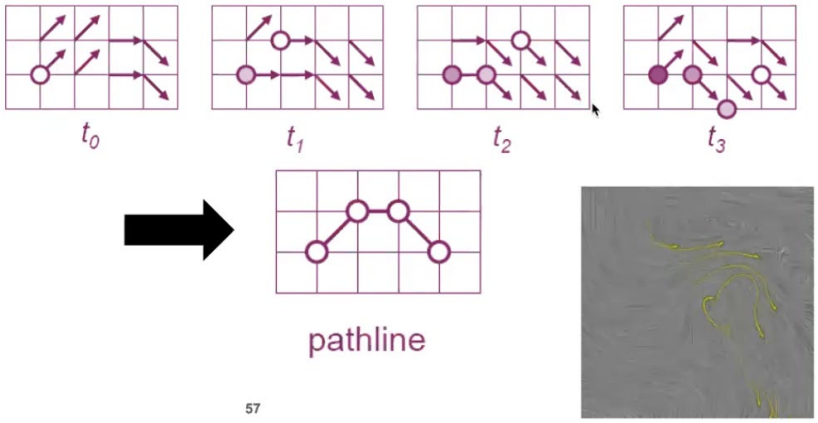
Streaklines – Tinte im Waschbecken auslaufen lassen, einfach streaks nachverfolgen, bzw. Öl am Flugzeug wie zuvor erwähnt.

Timelines – einen Fluss anschauen und verfolgen, wie er sich über Zeit entwickelt.

Konstantin Lackner 19.01.2021

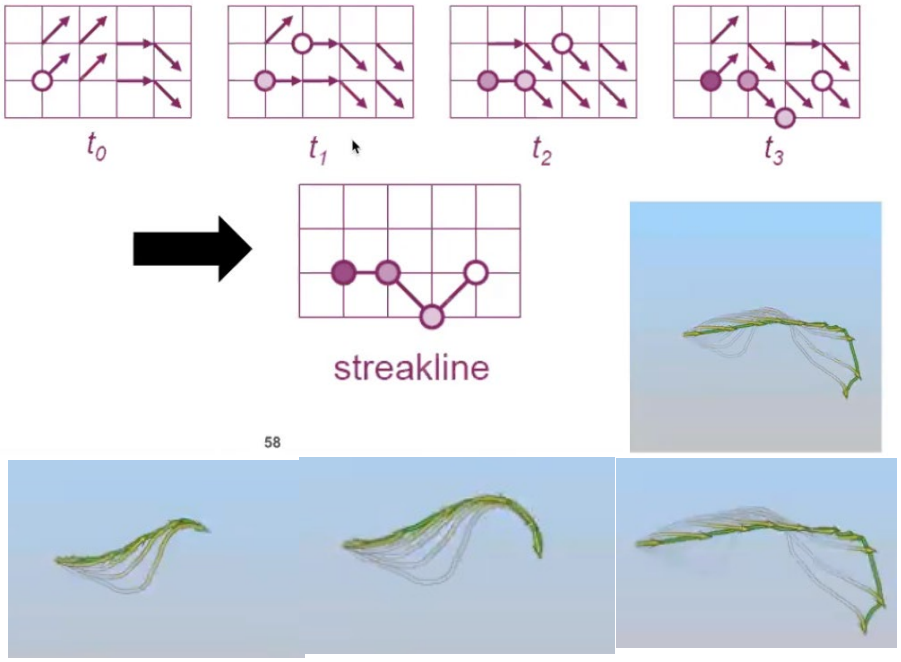
Pathline

Masseloses Partikel im Fluss. An einem Zeitschritt t0 hat man eine Position, an t1 …



Streaklines

Verfolgen die Punkte des Flusses und generieren dann immer weitere.



Sind einfach die Endpositionen der einzelnen Partikeln verbunden.

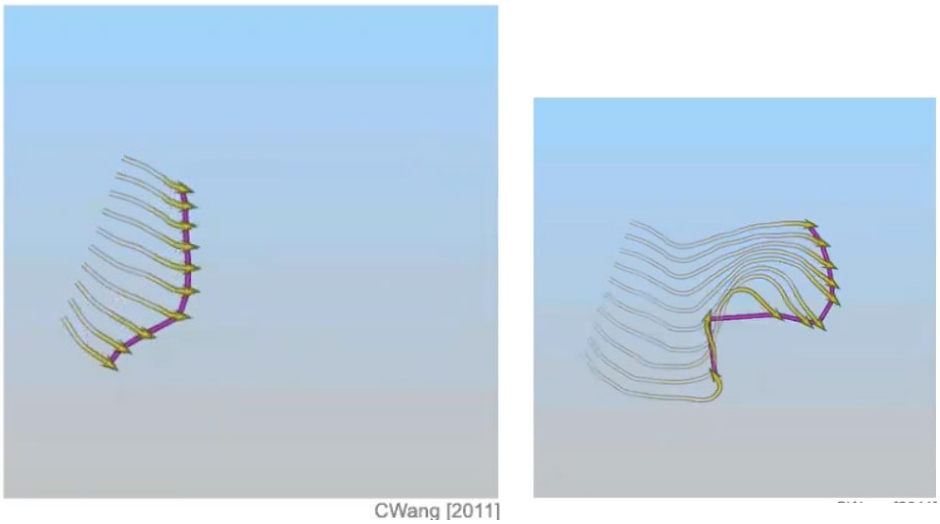
Streamline

Betrachtet nur einen Zeitschritt.

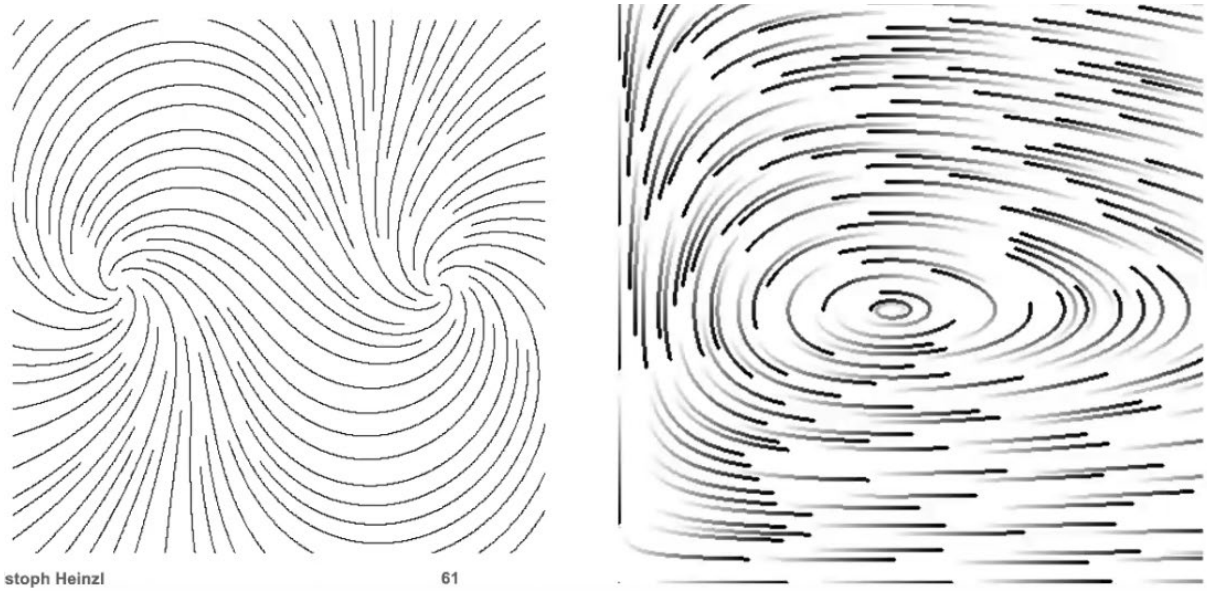
Konstantin Lackner 19.01.2021

Timelines

Verfolgen Linien im Flussfeld. Die Timelines sind ein set von Fluidpartikeln die sich über eine Zeit weiterbewegen.



Streamlines in 2D



Links streamlines die nach dem flowfield getraced werden, rechts streamlets, also kleine Segmente von streamlines, die über gewisse Zeit verfolgt werden. Je schwächer die opacity. desto weiter die Zeit her.

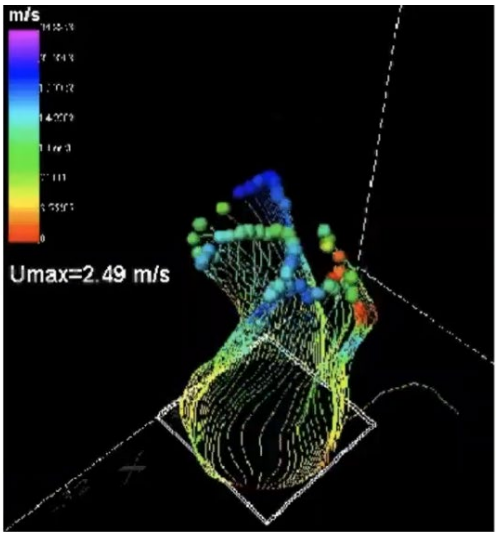
Streamlets zeigen nicht nur die Strömungsrichtung, sondern man kann auch die traces verfolgen. Man kann aber nicht genau sagen, wo Partikel am Ende ankommen. Das kann man bei streamlines aber schon gut verfolgen.

In 3D

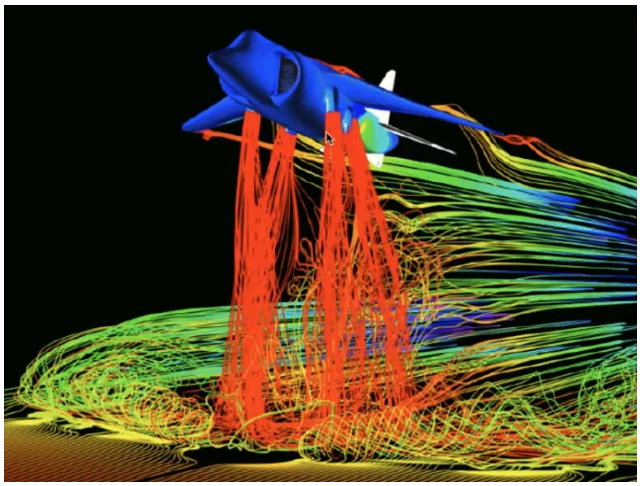
Particle paths = streamlines (steady flows)

Variants (time-dependent data):

- **streak lines:** steadily new particles
- **path lines:** long-term path of one particle

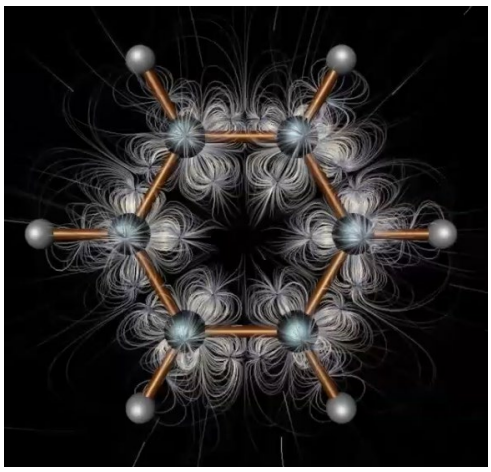


Bei Streamlines hat man auch oft das Problem von visual clutter.



Colour coding hilft dabei natürlich - wenngleich nicht sehr.

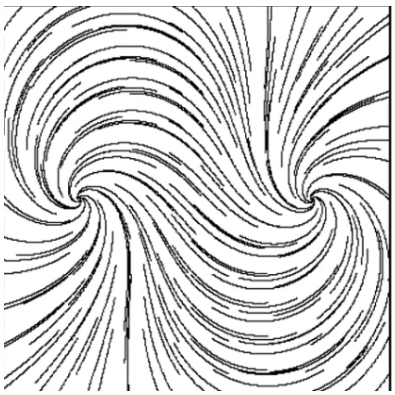
Auch illuminated Streamlines können helfen. Das sind einfach beleuchtete Streamlines. Man nimmt da einfach als gradient das, was mit dem Betrachtungsfeld den kleinsten Winkel bildet.



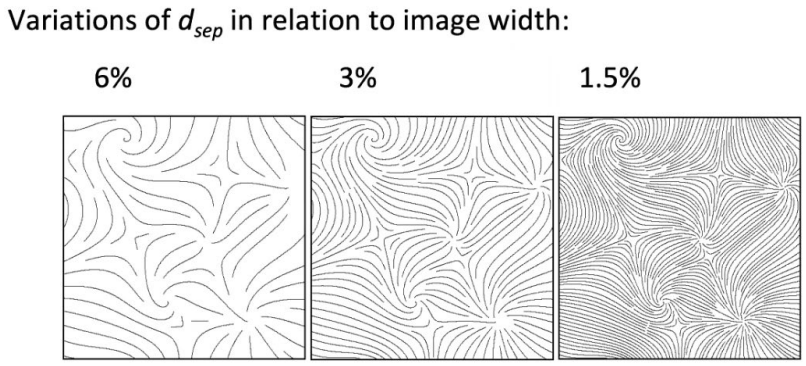
Konstantin Lackner 19.01.2021

Placement von Streamlines

Normalerweise geht man von jeder Position im grid aus und schaut dort, wo die Werte liegen.

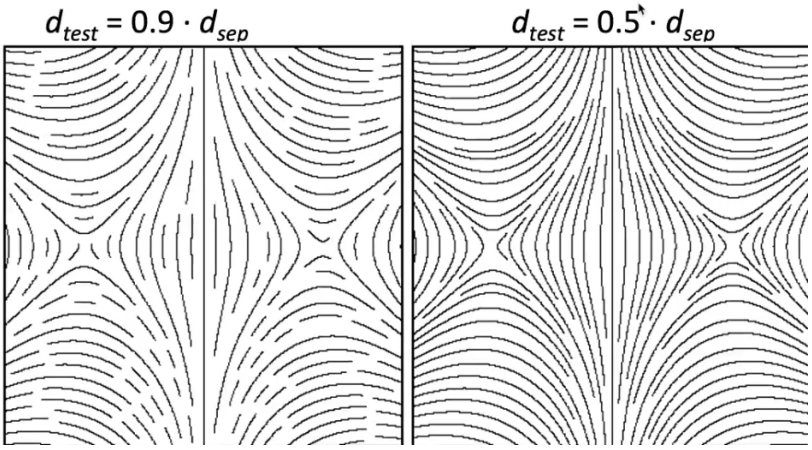


Um das aber zu verbessern, legt man fest, dass man die Streamlines nicht zu nah aneinanderlegt. Man macht sich einen **seed point** für die erste Streamline und einen für die nächsten, den man um einen gewissen separate Wert verschiebt. Als Code macht man das so, das man eine streamline berechnet und dann eine while-schleife macht. in der Schleife holt man sich einen neuen seed Punkt für einen neue Streamline, bis man eine hat, die mindestens den separate Wert weg ist, dort macht man eine neue und fängt wieder an. Wenn man aber Linien zieht, können sie ja auch erst später zueinander kommen. Dafür gibt es den d_{test} Wert (vgl. mit d_{sep} als separator von vorher) – Dieser Wert ist dafür da, dass man, nach dem initial Schauen, ob der Abstand eh nicht zu klein ist, auch weiter kontrolliert beim Linieziehen. Wenn iwann der d_{test} Wert unterschritten wird zur nächsten Linie (dieser ist immer <= d_{sep}) dann wird die Linie abgebrochen.



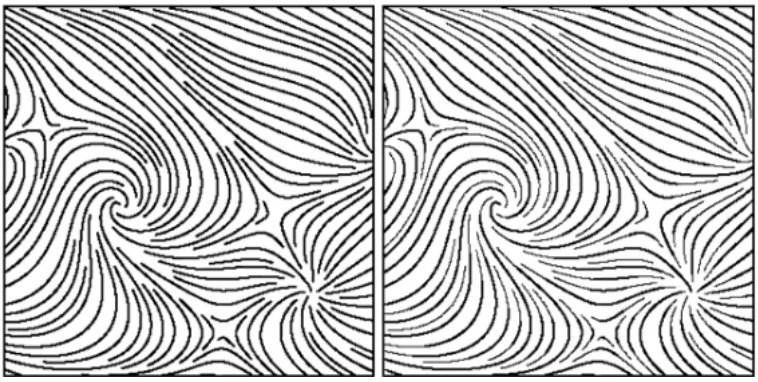
d_{sep} ist der separator. Man muss sich einen guten Wert aussuchen, damit das Bild nicht entweder zu dicht, oder zu uninformativ ist. Der zweite Wert, den man einstellen muss ist der d_{test} Wert. Dieser legt fest, wie tolerant man bzgl. späteren Annäherungen ist. Auch dieser

muss richtig gewählt werden.



Wählt man ihn zu groß, brechen die Streamlines zu früh ab. Zu klein, könnte man wieder zu gedrängte Ergebnisse bekommen.

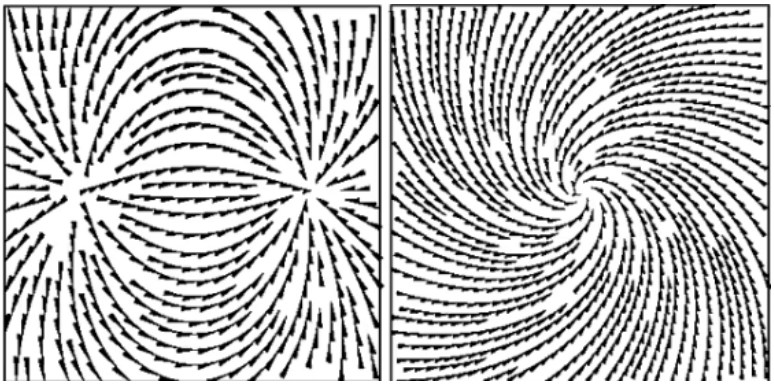
Man kann auch die Dicken von Strömungslinien einfließen lassen. Bspw. kann man sie von d_{sep} abhängig machen. Je weiter weg, desto dicker.



Thickness in relation to distance.

$$\begin{matrix} 1.0 & \forall d \geq d_{sep} \\ \frac{d - d_{test}}{d_{sep} - d_{test}} & \forall d < d_{sep} \end{matrix}$$

Man kann auch Richtungen mappen. Diese Glyphen sind gut und schlecht. Man sieht die Richtung besser, aber es is messy.



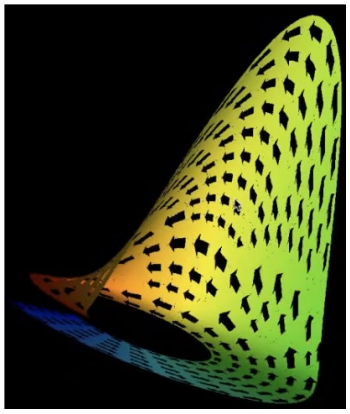
Konstantin Lackner 19.01.2021

Stream ribbons

Man startet mit zwei Streamlines. Die erste wird normal berechnet. Die zweite hat einfach immer einen konstanten Abstand zur ersten.

Stream surfaces

Man gibt eine Trajektorenlinie. Diese verfolgt man über die Zeit und bekommt dann eine Oberfläche.



Volumetrisches Visualisieren von Flussdaten

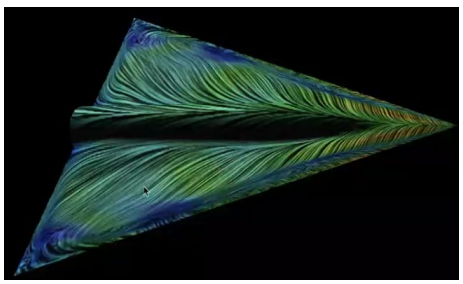
Einfach sehr viele Partikel in einem System simulieren und beobachten.

Seedobjektarten bei Streamlines und co.

IntegralObject	Dimension	SeedObject	Dimension
Streamline,…	1D	Point	0D
Streamribbon	1D++	Point+pt.	0D+0D
Streamtube	1D++	Pt.+cont.	0D+1D
Streamsurface	2D	Curve	1D
Flow volume	3D	Patch	2D

Line Integral Convolution

Man will eine Korrelation zwischen Textur, Oberfläche und Fluss darstellen.

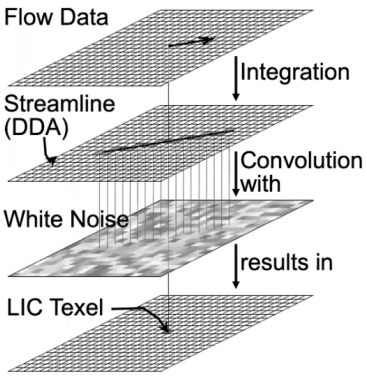


Für eine Textur sieht man sich die texel an und korreliert diese entlang des Flusses. Man korreliert sie aber nur in Richtung der Flussrichtung, nicht gegen oder quer dem Fluss. Als Grundtextur dient Whitenoise, dieses wird verwischt.

Man schaut sich wieder eine Streamline entlang der Flussdaten an und faltet diese mit der Textur. Damit bekommt man ein Texel für die Darstellung.

Calculation of a texture value:

- look at streamline through point
- filter white noise along streamline

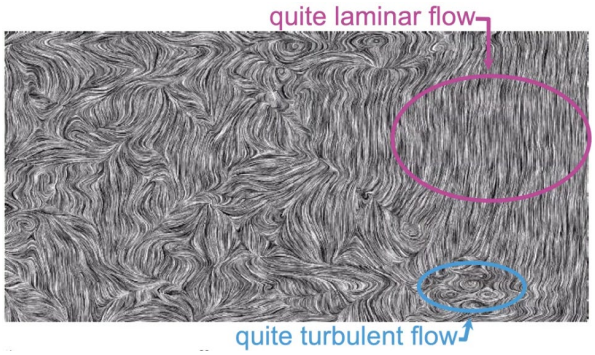


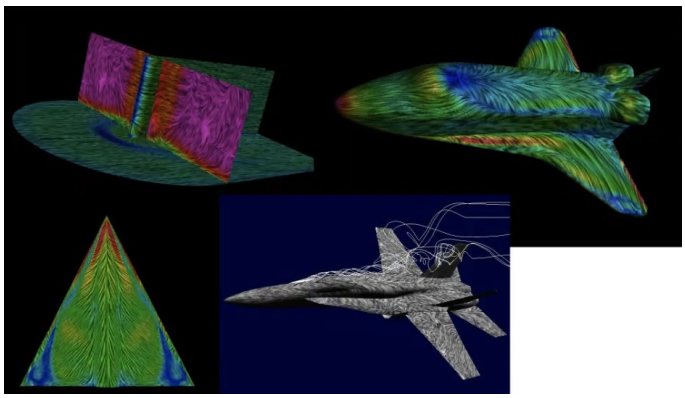
Faltung ist eine Mittelwertberechnung. Den errechneten Wert trägt man dann auf der fertigen Textur ein. Man hat also zwei Inputs. Flussdaten und Whitenoise Daten. Man macht die Streamline und faltet dann Flussdaten mit Whitenoise. man kann dann danach noch einen Gaußfilter anwenden um festzulegen, wie stark der Einfluss jeder einzelnen Position ist.

LIC – lic(**x**) – is a convolution of

- white noise *n* (or …)
- and a smoothing filter *h* (e.g. a Gaussian)

The noise texture values are picked up along streamlines $\mathbf{s}_x$ through **x**

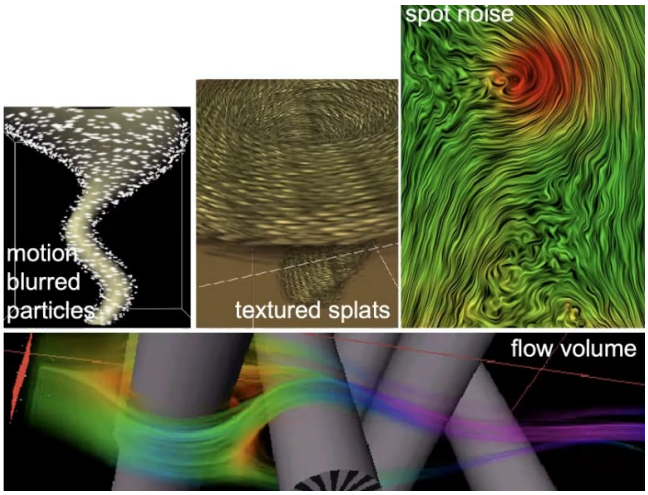




Weitere approaches

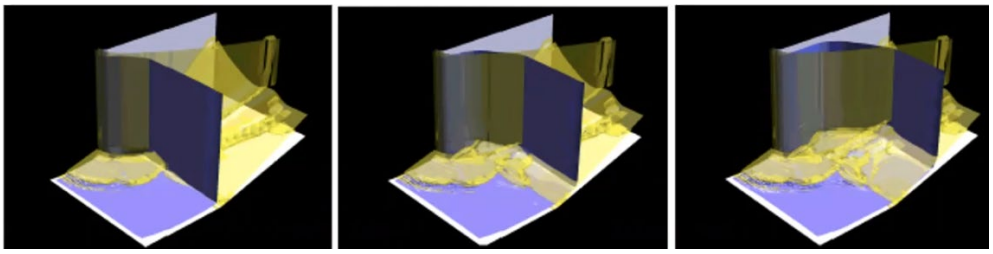
Similar approaches:

- spot noise
- vector kernel
- line bundles / splats
- textured splats
- particle systems
- flow volumes
- texture advection



Time surfaces

Hier macht man eine Oberfläche, die sich über die Zeit deformiert.



Konstantin Lackner 19.01.2021

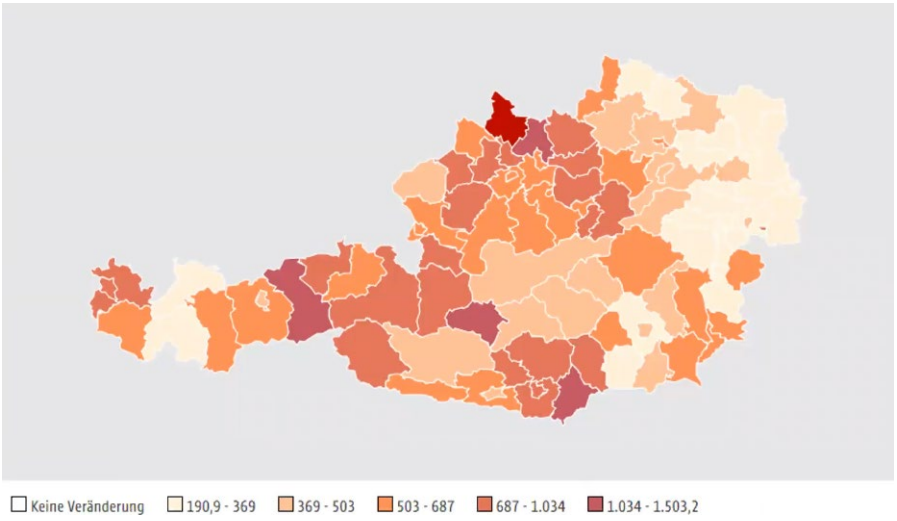
7. InfoVis

Einleitung

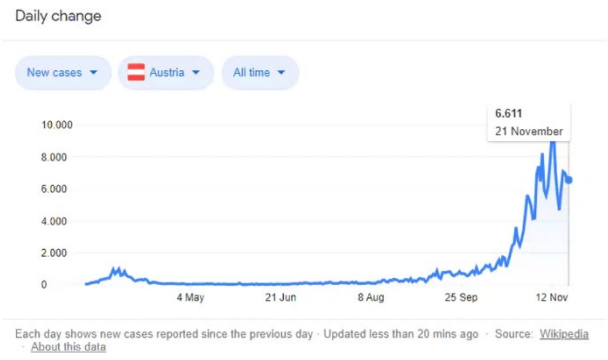
Beispiel John Snow Cholera Epedemie. Bei InfoVis werden eben abstrakte Daten und Daten aus diversen Bereichen visualisiert.

Auf der Landkarte von London alle Fälle aufgezeichnet. Eine wasserquelle, die kontaminiert war, war der ausschlaggebende Faktor.

Heute bei Corona:



Infektionszahlenveränderung.

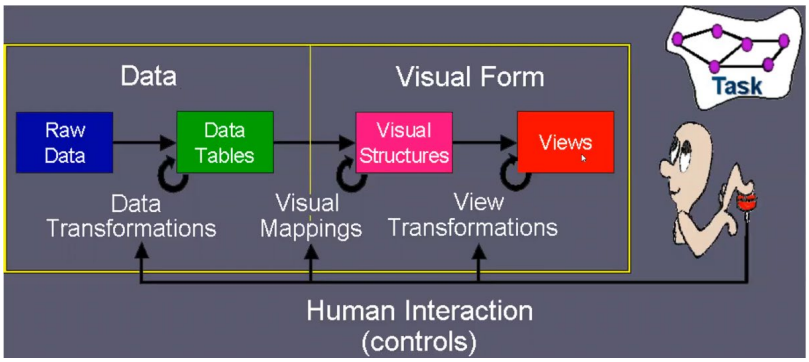


Vis spielt in vielen Bereichen eine Rolle, ob Infografiken mit wenig Interaktivität, bis hin zu plotting tools mit viel Interaktivität. Zweiteres ist dann explorative Analyse. Wo man noch nicht weiß, was die Daten einem sagen, aber eben mit den vis tools nachforscht.

Man kann Infovis also definieren als:

„The use of computer supported, interactive, visual representations of abstract data to amplify cognition"

InfoVis Pipeline



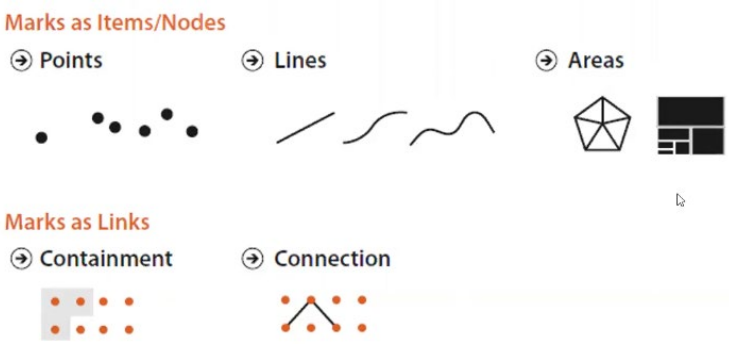
Man beginnt bei Rohdaten, die in Tabellen transformiert werden, um mit diesen zu arbeiten (bei uns wurde in task 2 die grüne Box vorgegeben) – in der Praxis ist das aber eigl. ein riesen Schritt und Aufwand. Dann passiert das visual mapping. Man hat also Tabellen und will daraus 2D Strukturen finden, oder 3D, usw.. Dann muss man Views generieren. Das sind dann Zooms, Transformationen, Ausschnitte, Fokussierungen…

Was Vis von charts unterscheidet, ist, dass der Nutzer bei jedem dieser Schritte die Möglichkeit hat einzugreifen.

Visual Mapping – Fachbegriffe?

Marks and **channels**

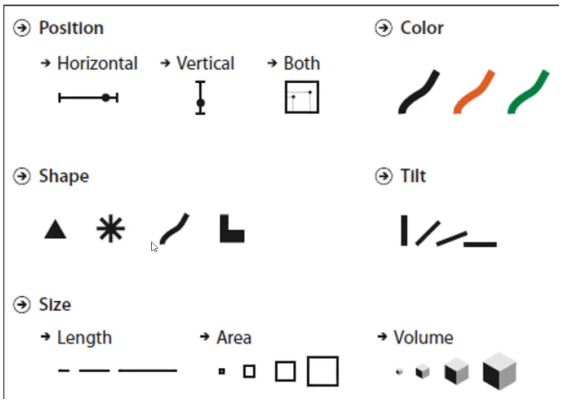
Marks sind die Dinge, mit denen man Items darstellt.



Wenn man Daten mit Verbindungen hat, braucht man auch für die links marks

Channels sind dann für die marks Eigenschaften. Nämlich Farbe, Position, Shape, tilt, Größe…

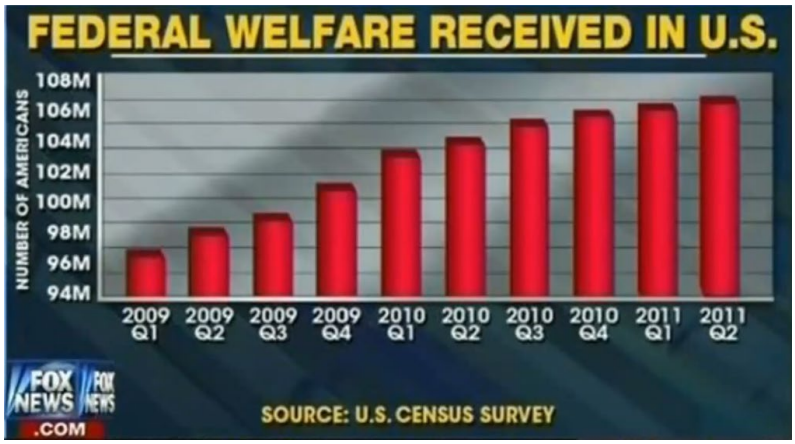
Konstantin Lackner 19.01.2021



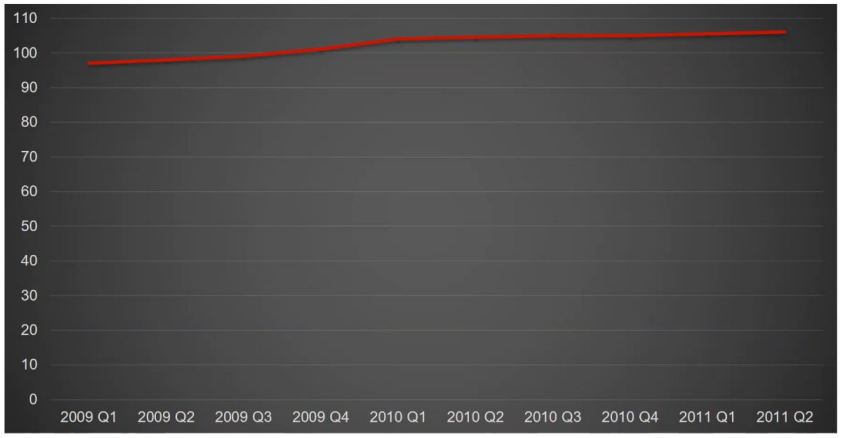
Univariate Daten

Sind solche, wo man ein Attribut hat, das man darstellen will (bspw. Verkehrsunfälle in einem Jahr)

Beispiel von Abschneiden von Y-Achsen



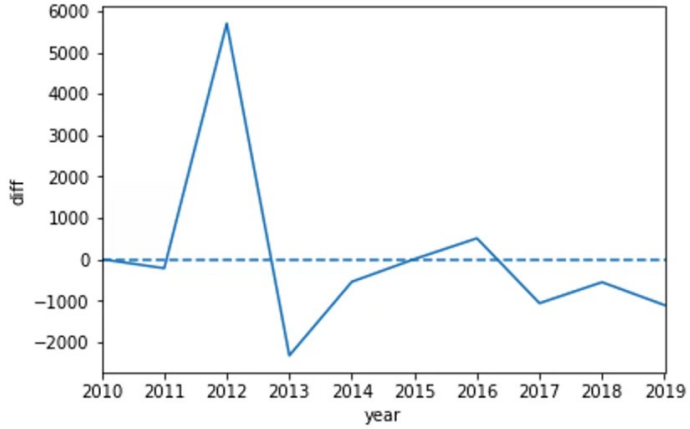
Wenn man die Y-Achse nicht abschneidet, und einen line chart verwendet, kommt das raus:



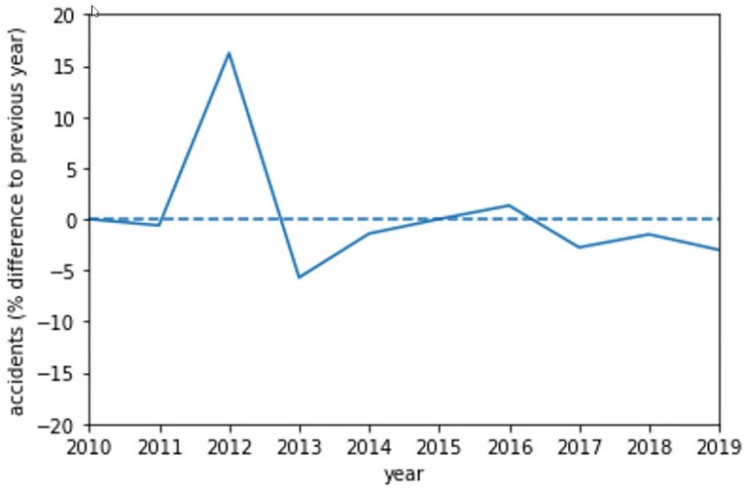
„The representation of number, as physically measured on the surface of the graphics itself, should be directly proportional to the quantities represented." – Sagt Tufte dazu

Will man also erreichen, dass in einem Jahr eine große Veränderung war, obwohl es im Gesamten ned oag aussieht, muss man eine Datentransformation machen.

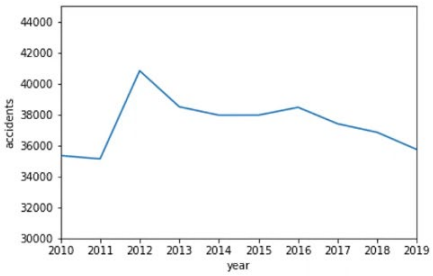
Konstantin Lackner 19.01.2021

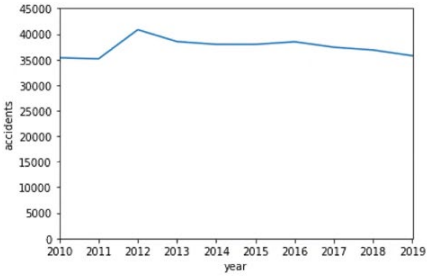


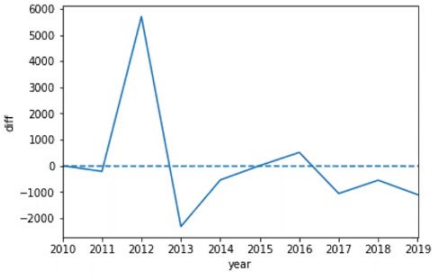
Man kann stattdessen den Unterschied zum Vorjahr anschauen.

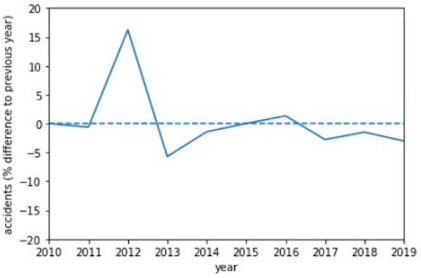


Oder dasselbe prozentuell.



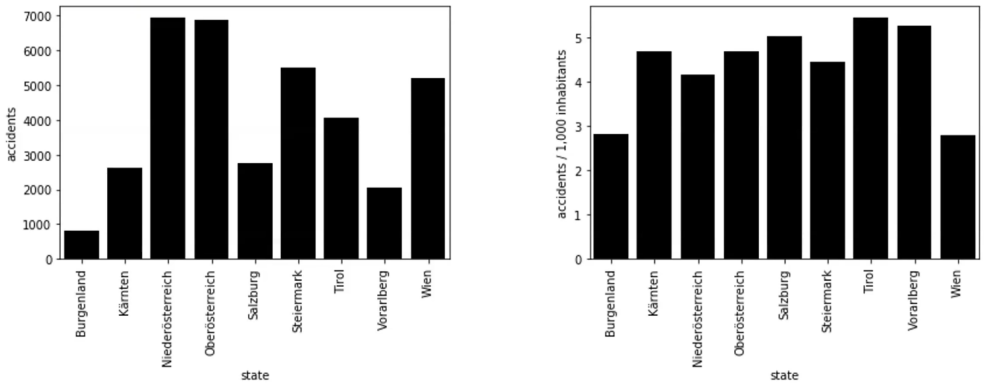






Es gibt also viele Möglichkeiten und alle geben einen anderen Eindruck. Bis auf das oben links ist nichts falsch.

Beispiel Verkehrsunfälle:



Links einfach die Zahl pro Bundesland, rechts Zahl pro 1000 Einwohner.

Bivariate Daten

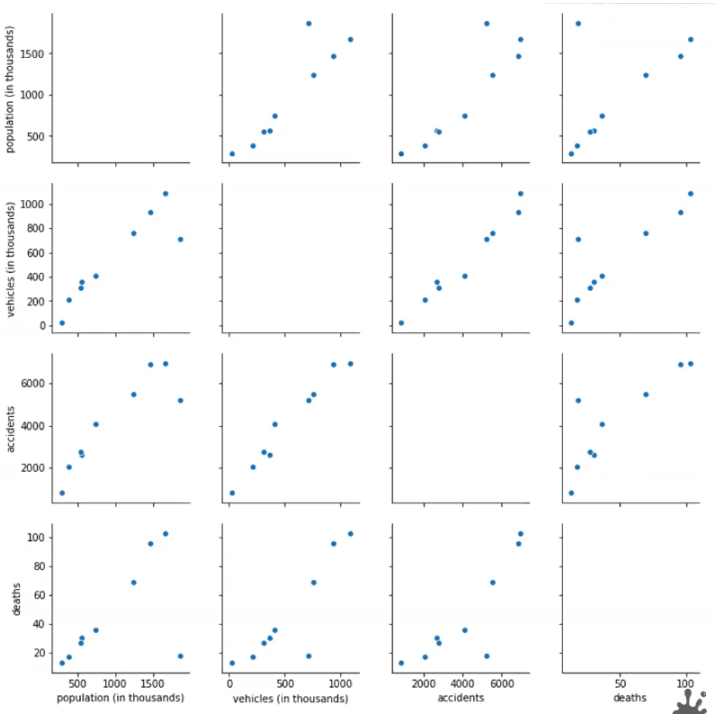
Das hier sind Daten die Zusammenhänge darstellen.

Bspw. Scatterplots. Deren Stärken sind, dass man Korrelation gut erkennt, Outliers gut finden kann, Extremwerte gut finden kann und auch Cluster gut finden kann.

Mit einem Scatterplot kommt man aber schnell an Grenzen, konkret kann man nicht viel mehr als 3-4 quantitative Werte darstellen. Braucht man aber mehr, so empfiehlt sich Nesting.

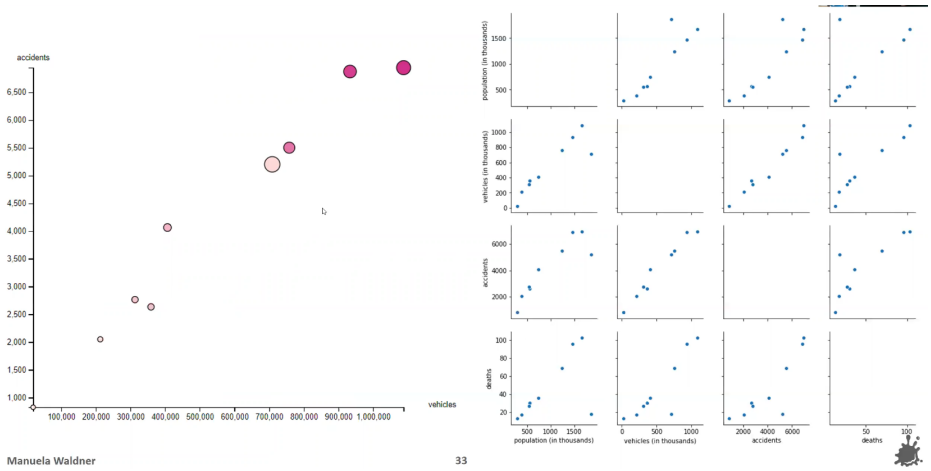
Multivariate Daten

Hier gibt es zum Beispiel **Scatterplot Matrizen**. Dabei handelt es sich einfach um eine Reihe von Scatterplots, die man nebeneinanderstellt, um den Zusammenhang zwischen allen möglichen Zweierkombis aus Daten darzustellen.



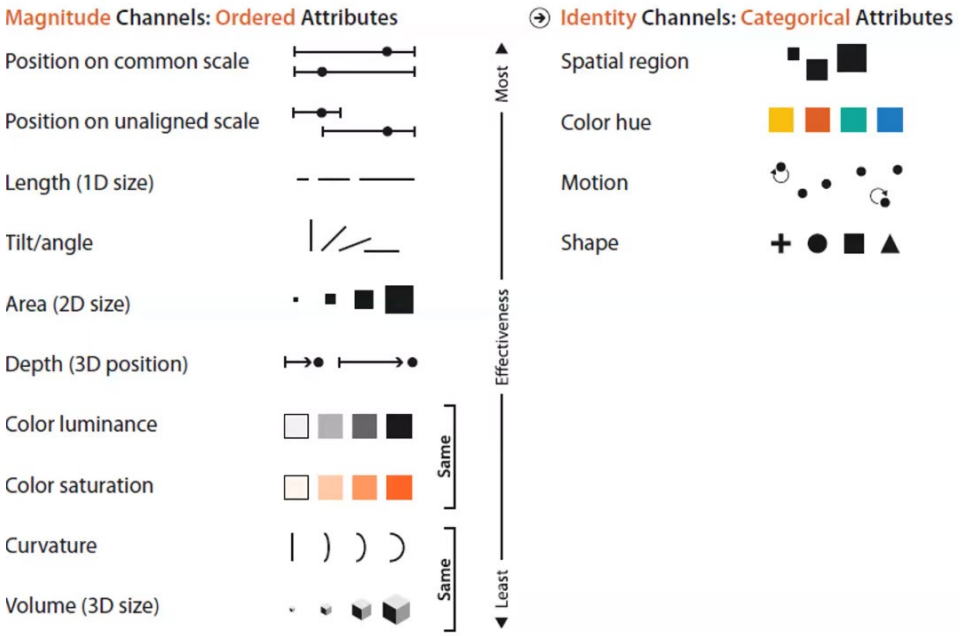
Dabei wird also der Scatterplot die mark.

Stellt man dieselben Daten wie vorher dar, dann sieht man jetzt die Ausreißer viel besser.

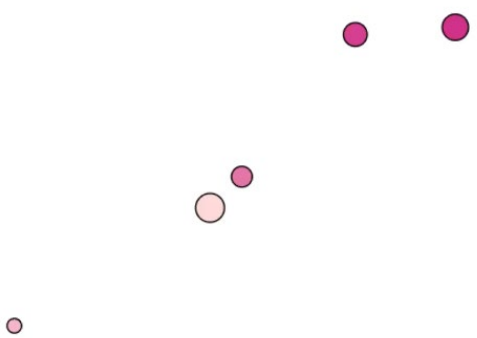


Das liegt daran, dass man zu visual channels verschiedenen sensitiv ist.

Sensitivität zu channels



Man ist also sehr gut darin Positionen zu unterscheiden. Allerdings nur bei Magnitude channels. Im Beispiel vorher war der helle Punkt hier in der Mitte ein mega Ausreißer:



Weil er sehr groß ist und sehr hell. Aber die Helligkeit fällt kaum auf. Die Position ist viel mehr im Fokus.

Konstantin Lackner 19.01.2021

Interaktivität

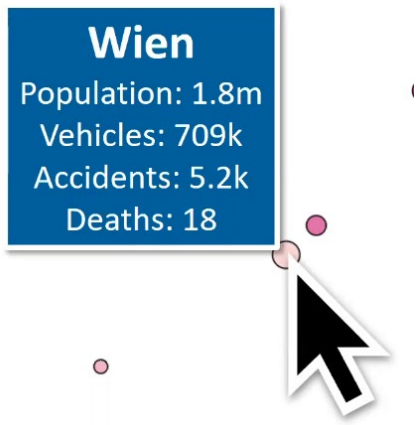
Welche Art von Interaktionsmöglichkeit ist sinnvoll und was wird benutzt?

Üblicherweise läuft es so ab:

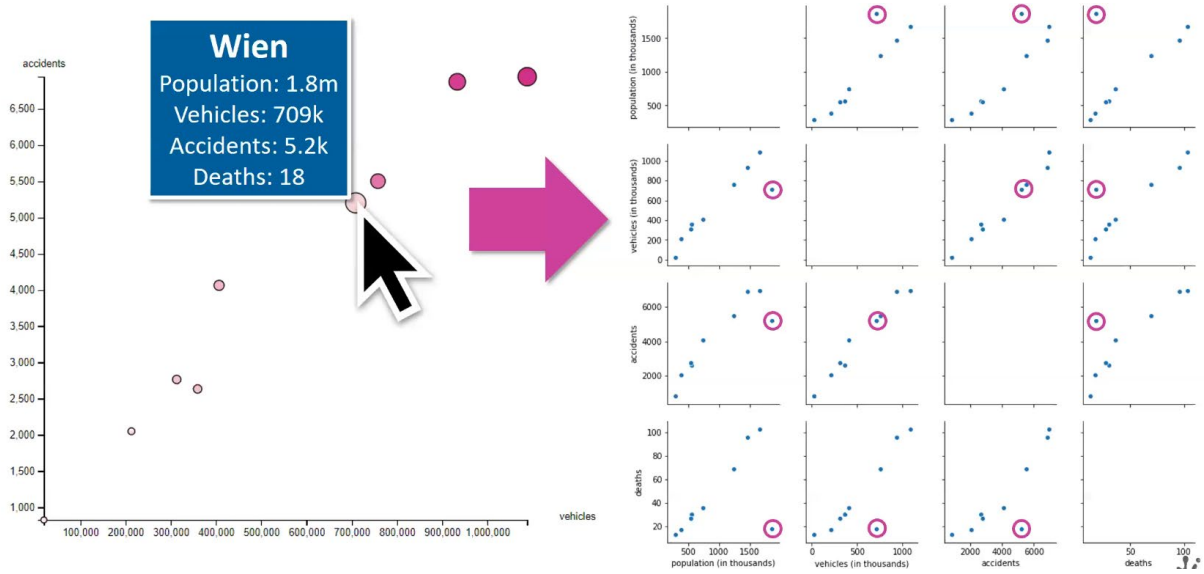
Overview First
Zoom and Filter
Details on Demand

Zuerst Overview, dann die Möglichkeit zu zoomen und filtern und dann feature on demand.

Zuerst also der Grobe overview, dann kann ich genauere Bereiche filtern und zoomen und schließlich kann ich bspw. über den hellen Punkt hovern und bekomm das: (einen tool tipp)



Brushing and linking ist dabei dann noch die Möglichkeit in einer Vis etwas zu markieren, um das dann in der anderen zu markieren:

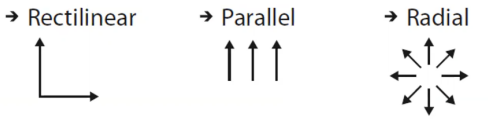


Noch mehr Features und tabular data

Das Nesting von zuvor ist ganz sinnvoll um noch ein paar mehr Features darzustellen. Nichtsdestotrotz kann man damit nicht mehr als eben ein paar wenige mehr anzeigen. Will man aber noch größere Zahlen von Features visualisieren, so gibt es andere Möglichkeiten.

How to visually encode more than 3-4 quantitative attributes?

- Nesting
- Axis orientation

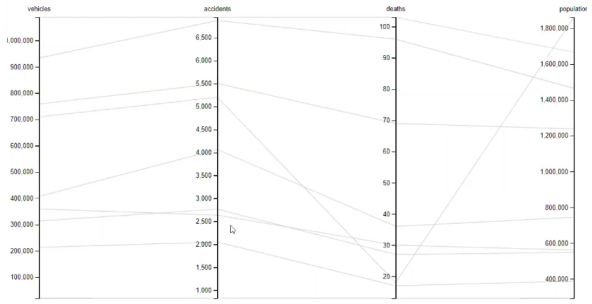


[Munzner, 2014]

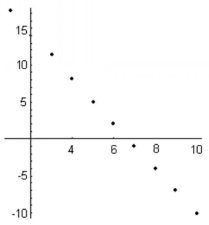
Statt Nesting und statt eben dem normalen rectilinear system, kann man Achsen parallel darstellen.

Parallel Coordinates:

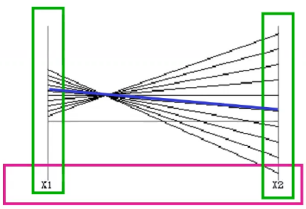
- Data:
  - N quantitative value attributes
    → multivariate data
- Visual encoding:
  - Marks: lines
  - Visual channels: y position
  - Axes layout: parallel
- Strengths:
  - Establish patterns, examine interactions



Zur Prüfung könnte bspw. kommen, dass man einen cartesian graph bekommt und daraus einen mit parallel coordinates zeichnen muss.



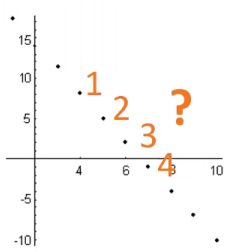
Dataset in a Cartesian graph



Same dataset in parallel coordinates

[Inselberg]

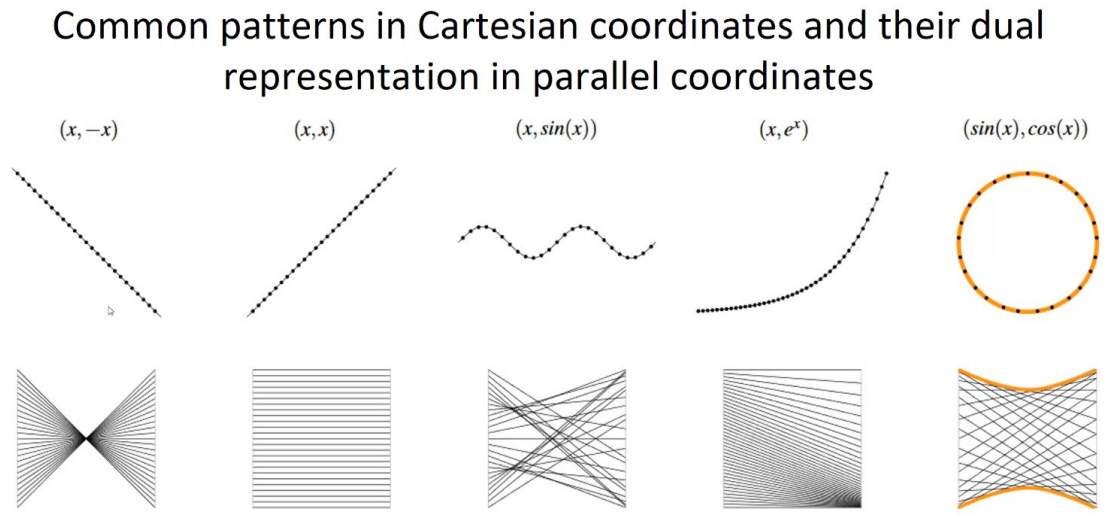
Encode variables along a horizontal row
Vertical line specifies single variable
Blue line specifies a case



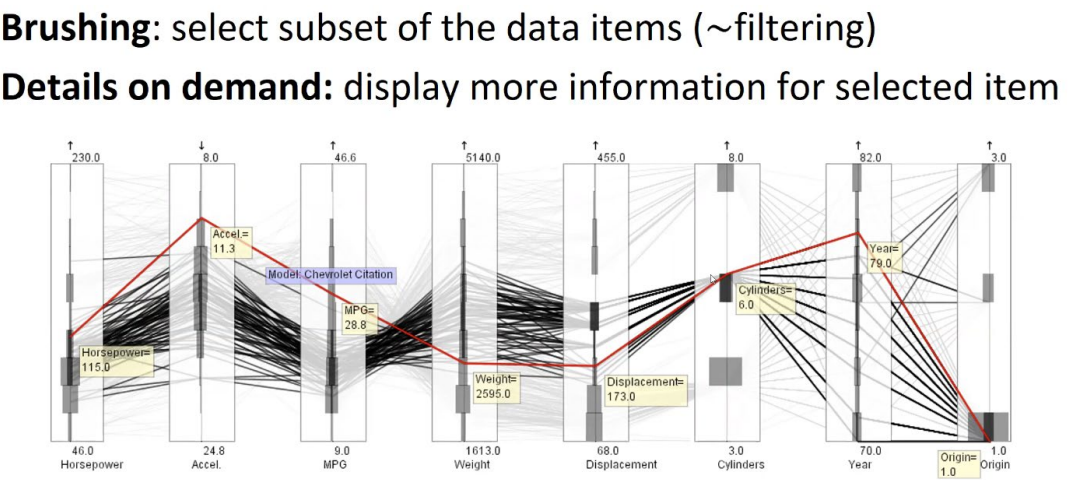
Gefragt könnte auch werden, um welchen Punkt es sich bei der blauen Linie handelt.

Die blaue Linie ist hier Punkt 3.

Es gibt sehr klare Beispiele, wie Dinge in cartesian zu parallel aussieht.

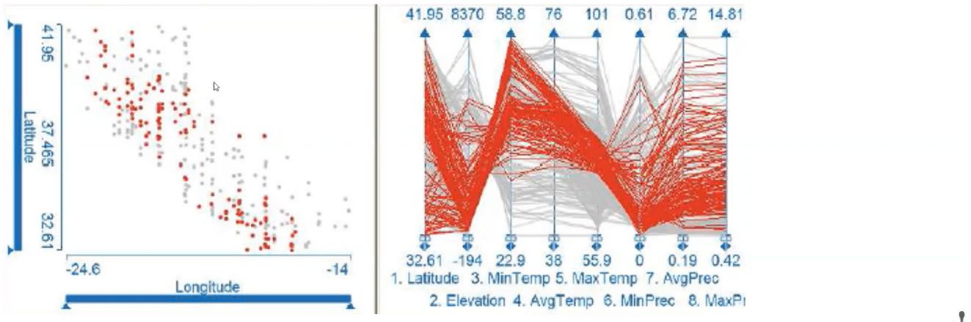


Brushing bei parallelen Koordinaten



Linking and brushing:

- Interactive changes in one view automatically reflected in other views

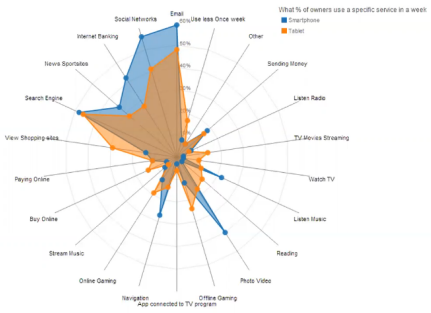


dner 44 [Matkovic et al., IV 2008]

Radiales Anordnen von Achsen

Star Plots / Radar Chart

- Data:
  - N quantitative value attributes
    → multivariate data
- Visual encoding:
  - Marks: lines
  - Visual channels: position on axis
  - Axes layout: radial



https://www.visualcinnamon.com/2013/09/making-d3-radar-chart-look-bit-better.html

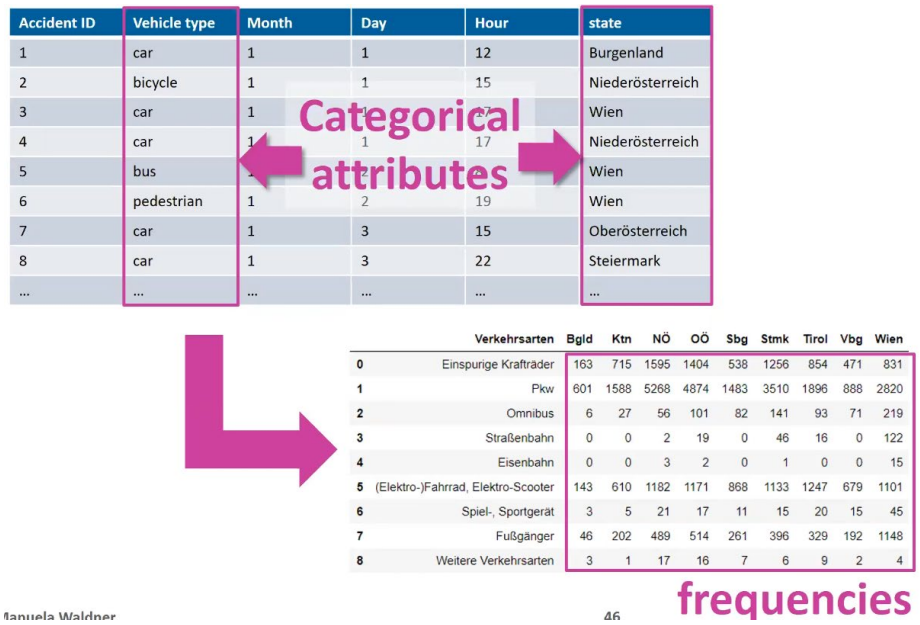
[Ward, Multivariate Data Glyphs: Principles and Practice, 2008]

Man hat eigl. wieder ein Item als Polylinie, wie zuvor. Marks sind hier die lines, die channels sind die positions auf Achsen.

Das rechts unten wäre die Möglichkeit für jedes Element einen eigenen Radar chart zu machen und sozusagen eine kleine Visualisierung für jeden Wert zu machen.

Multidimensionale Daten

Bisher hatten wir tables, wie die funktionieren, ist klar. Will man beim table jetzt aber bspw. den Zusammenhang zwischen zwei kategorischen Variablen anschauen (nicht quantitative, nicht numerische Variablen) – dann transformiert man um in einen neuen Table, wo die Zellen nicht mehr das Attribut selbst darstellen, sondern Frequenzen.

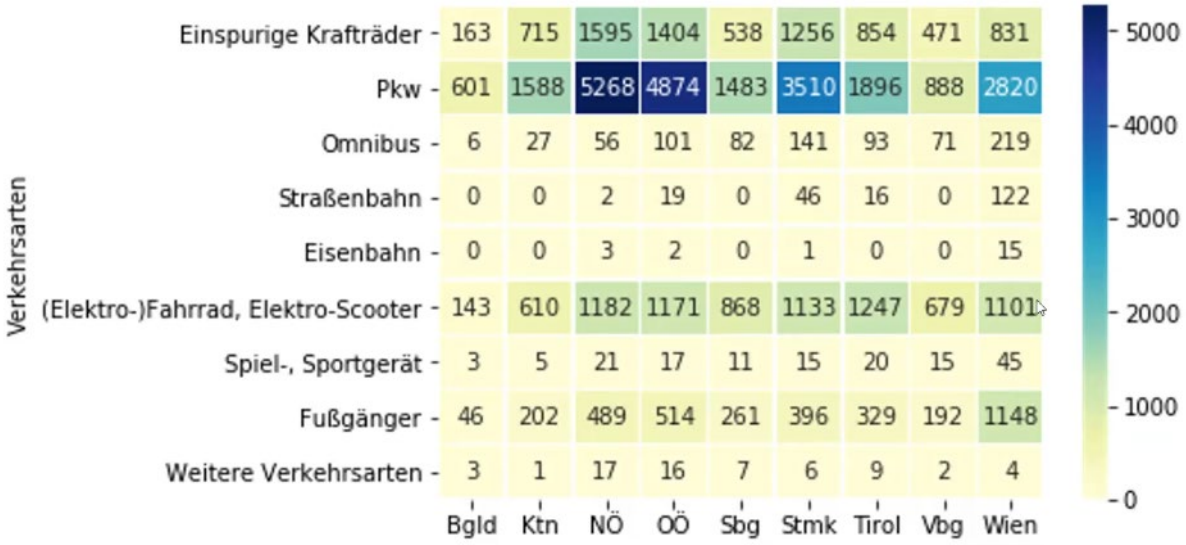


Accident ID	Vehicle type	Month	Day	Hour	state
1	car	1	1	12	Burgenland
2	bicycle	1	1	15	Niederösterreich
3	car	1	1	17	Wien
4	car	1	1	17	Niederösterreich
5	bus	1	1	18	Wien
6	pedestrian	1	2	19	Wien
7	car	1	3	15	Oberösterreich
8	car	1	3	22	Steiermark
...	...	...	...	...	...

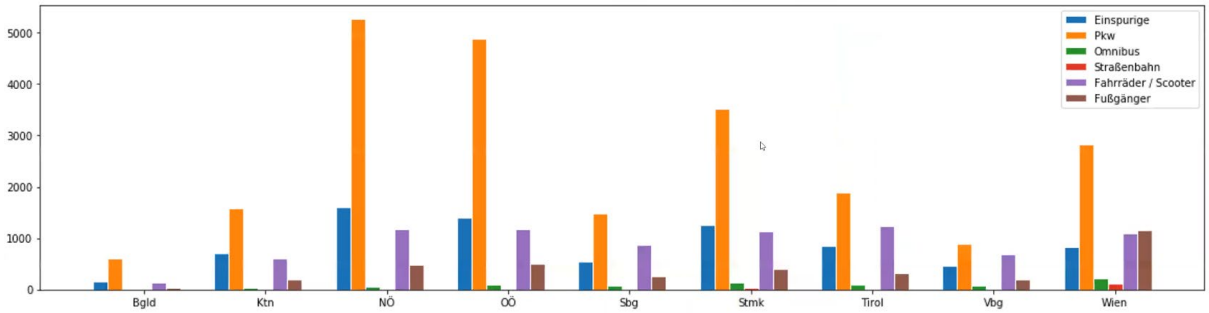
	Verkehrsarten	Bgld	Ktn	NÖ	OÖ	Sbg	Stmk	Tirol	Vbg	Wien
0	Einspurige Krafträder	163	715	1595	1404	538	1256	854	471	831
1	Pkw	601	1588	5268	4874	1483	3510	1896	888	2820
2	Omnibus	6	27	56	101	82	141	93	71	219
3	Straßenbahn	0	0	2	19	0	46	16	0	122
4	Eisenbahn	0	0	3	2	0	1	0	0	15
5	(Elektro-)Fahrrad, Elektro-Scooter	143	610	1182	1171	868	1133	1247	679	1101
6	Spiel-, Sportgerät	3	5	21	17	11	15	20	15	45
7	Fußgänger	46	202	489	514	261	396	329	192	1148
8	Weitere Verkehrsarten	3	1	17	16	7	6	9	2	4

Man zählt jetzt also nicht mehr die Gesamtumfälle pro Bundesland, sondern man schaut die Unfälle pro Bundesland mit einem gewissen Fahrzeugtypen an.

daraus kann man Heatmaps machen.

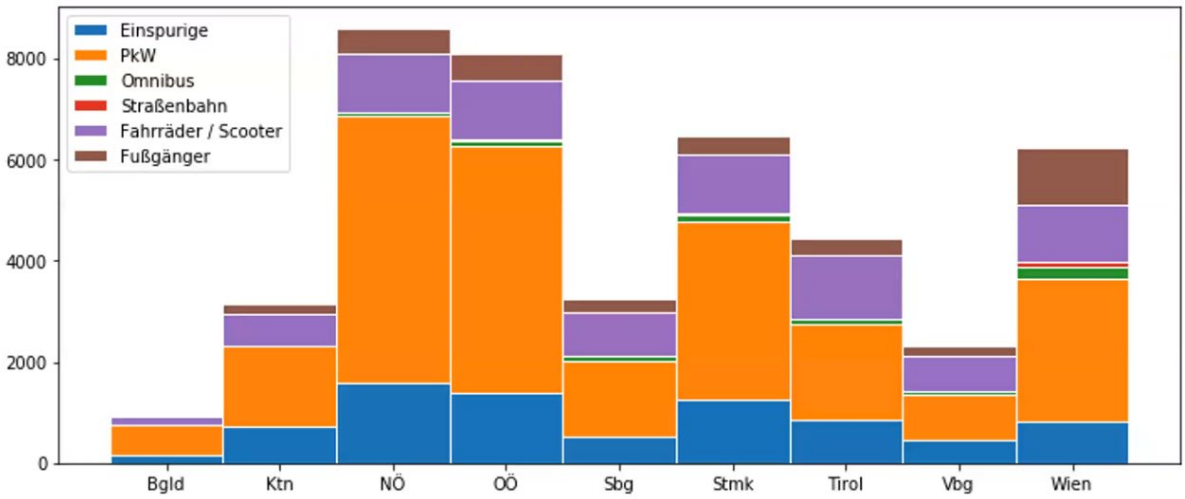


Oder man macht wieder ein Nesting mit bar charts ein gruppiertes bar chart. (**grouped bar chart**)



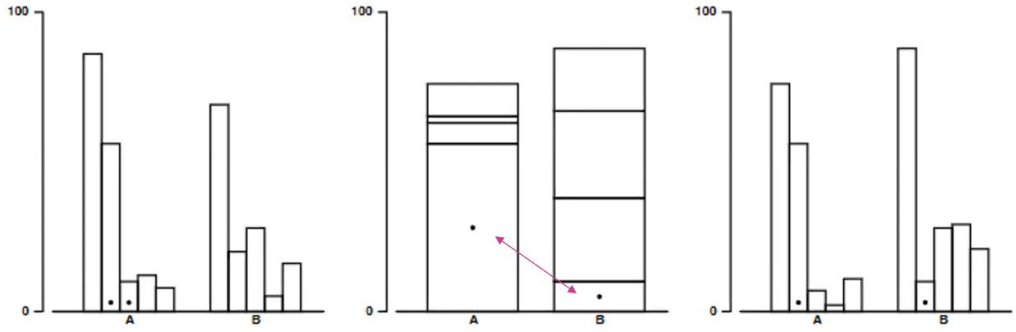
Also ein bar chart pro Bundesland wo man die Unfälle pro Kategorie darstellt.

Oder ein stacked bar chart



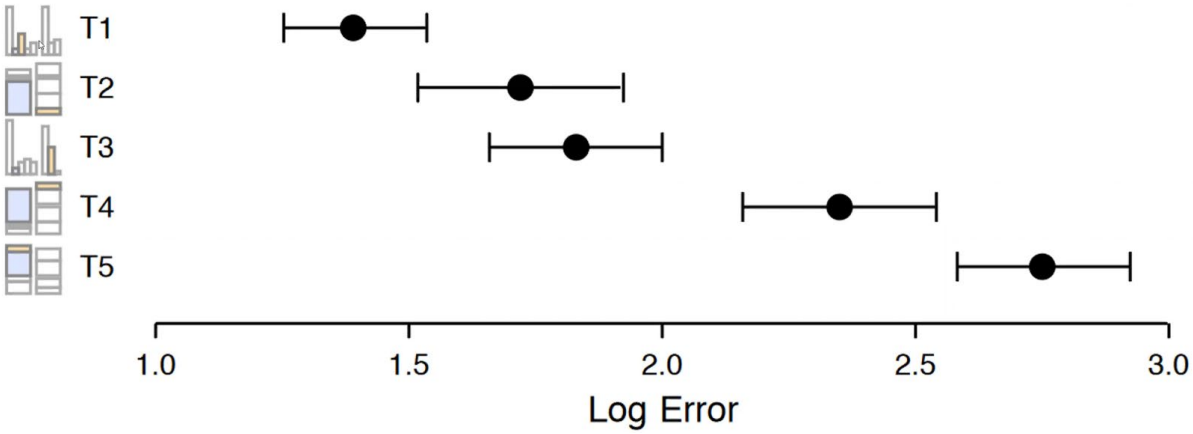
Also ein bar chart wo die einzelnen bars übereinander sind und die einzelnen bars übereinander.

Auch hier hat man wieder verschiedene Vor- und Nachteile. Einerseits kann man in den grouped bar charts schwer sagen, welches Bundesland bspw. insgesamt die drittmeisten Unfälle hat. Das geht in der zweiten Variante leicht. Hingegen die Frage welches Bundesland die zweitmeisten Fahrradunfälle hat, ist im zweiten kaum zu beantworten, im ersteh eher noch.



Zu genau diesem Phänomen gab es Untersuchungen, wo Leute raten mussten, wie groß der prozentuelle Unterschied in bar charts ist. Dabei wurde eben untersucht, welche Art am ehesten gut visualisiert, was die Datenlage ist.

Dabei kam raus:



Wenn die bars nebeneinander sind, raten die Leute gut.

Wenn die bars gestacked sind, aber dennoch nebeneinander, is es schon schwerer.

Ähnlich schlimm wenn die bars nicht direkt nebeneinander sind.

Richtig oag wird es, wenn die bars nicht auf der selben y Ebene starten.

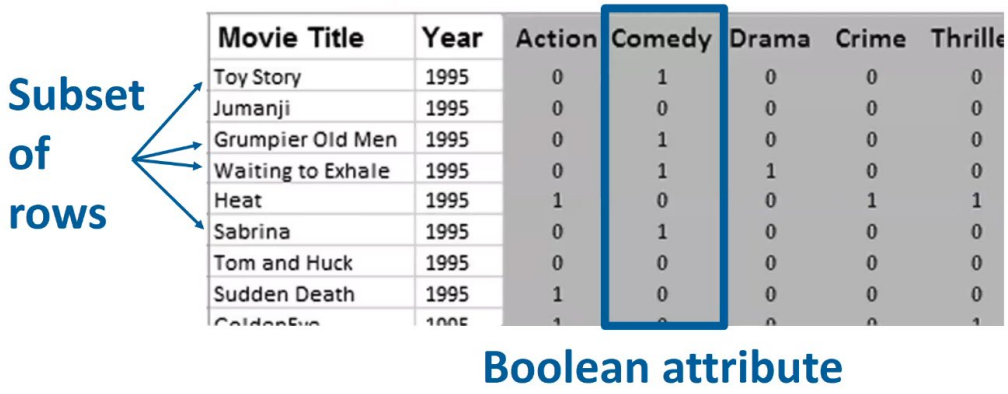
Und am ärgsten, wenn die bars innerhalb einer Gruppe sind.

Daraus folgt also, dass man sich immer überlegen muss, was man darstellen will.

Boolsche Attribute

Ein weiters Beispiel für kategorische Variablen sind boolsche Variablen. Hat man boolsche Attribute, spricht man von sets.

- Set: items classified into one or more categories
- Sets can overlap
- Set theory: $A \cup B$, $A \cap B$ ...



Ein set von comedy Filmen wären bspw. alle, wo comedy auf 1 ist. Ebenso wären bspw. Filme wie hier Waiting to Exhale ein overlapping set zwischen comedy und drama. Damit kann man gut visualisieren.

Bspw. in Venn oder Euler Diagrammen.

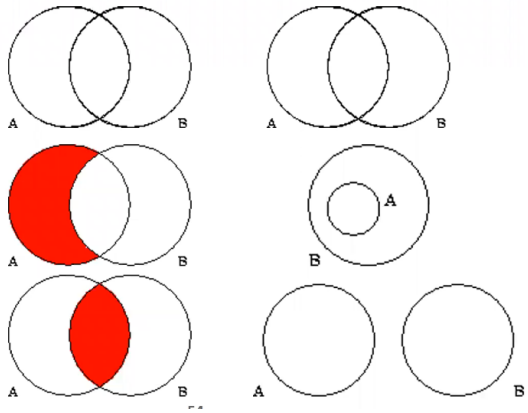
Venn Diagrams: shows all possible logical relations between sets represented as closed curves

Euler Diagrams: a region is present if and only if it contains at least one item

$\{\emptyset, A, B, AB\}$

$\{\emptyset, B, AB\}$

$\{\emptyset, A, B\}$

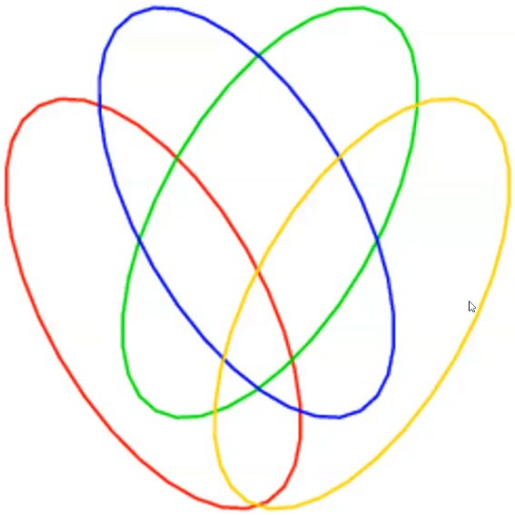


[Ruskey and Weston, The Electronic Journal of Combinatorics, 2005]

Venndiagramme zeigen alle Möglichen Zusammenhänge zwischen A und B, Euler die tatsächlichen. Hat man wie im zweiten Beispiel also keine Elemente, die nur mit a assoziiert sind, so zeit das das Euler Diagramm auch. Bei Venn sieht man das nicht. Ebenso zeigt das Eulerdiagramm im letzten Beispiel keine Schnittmenge, weil es diese nicht gibt. Beim Venn wurde mit shading gezeigt, welches set leer ist.

Nachteil an Venn Diagrammen ist, dass sie nicht sehr gut skalieren. Bei vier Sets hat man schon das:

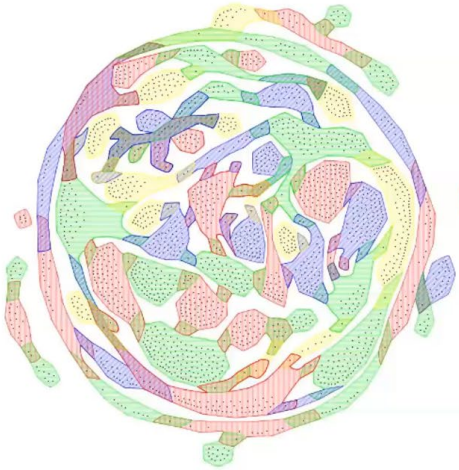
$n = 4$



Bei 60 sets muss man auf Euler-like Diagramme zurückgreifen…

Konstantin Lackner 19.01.2021

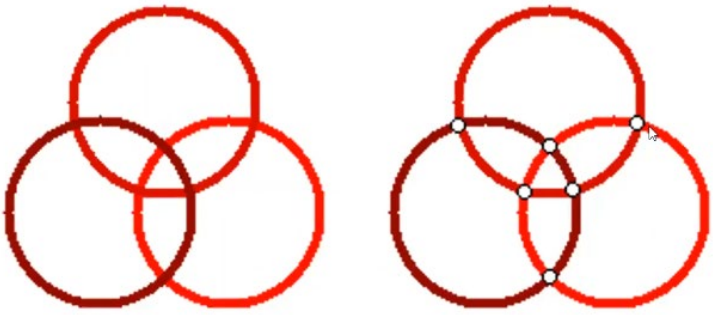
$n = 60$



60 top films from IMDB
(dots represent > 2,000 actors)

Venn Grafen

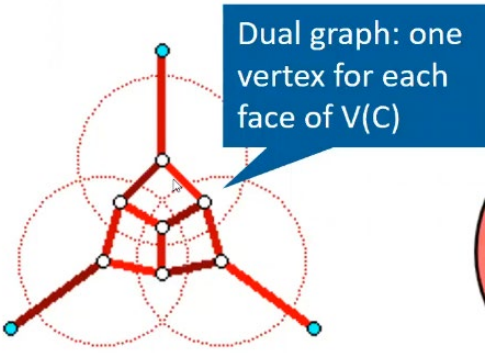
Die Vertices sind dabei die Schnittpunkte der Kreise, die Edges sind die Kurven segmente:



The Venn Diagram, C

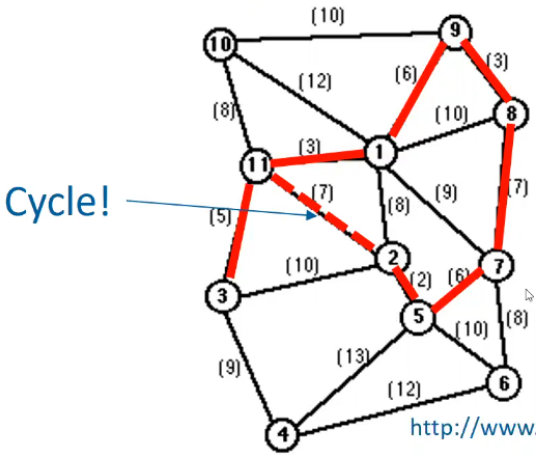
The Venn Diagram V(C)
as an edge labelled Graph.

Das Rechte ist jetzt eine Grafendarstellung eines Venndiagramms. Diese Darstellung ist **planar**, die Kanten überkreuzen sich nicht. Aus **jedem planaren Grafen kann man einen dualen Grafen machen**. Ein dualer Graf wiederrum ist einer, bei dem **alle Flächenmittelpunkte zu einem Vertex** werden und **edges jene Regionen verbinden**, die vorher eine geteilte Grenze hatten. Man macht also das:



The Venn Dual, a 3-cube, D(C)
The cyan vertices are identified.

Man macht also zuerst diesen dualen Grafen, **dann bringt man die Kanten auf ein Minimum**. Das macht man mit einem Kruskal. (**minimal spanning tree** … immer die kleinstgewichtete Kante hinzufügen, außer es entsteht ein Ring und das bis man alle Punkte hat.)



Der entstehende Baum kann auch nichtüberlappend gezeichnet werden.

Man macht dann **einen planaren Intersection Graf**. Man darf eben keine überlappende Edges haben. Man benutzt hier aber nicht den Kruskal sondern eine abgewandelte Version, wo man auch noch sicherstellt, dass sich nichts überlappt. Außerdem nimmt man die **maximalen Kantengewichte statt minimale.** Die Kantengewichte stellt man fest, indem man einfach sagt, dass die Vertices die set relations sind. Die Kantengewichte definiert man durch die Anzahl der sets, die durch diese Kante verbunden werden würden. (Professorin fängt da sehr zum Stottern an und verweist aufs paper… I call bullshit.)

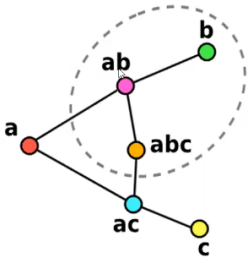
Edge weights: $\omega(e) = c(e) - p_1 u(e) - p_2 v(e)$

$c(e)$: number of disconnected sets

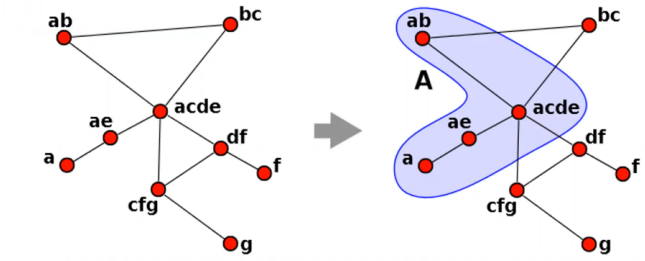
$u(e)$: number of unrelated sets connected

$v(e)$: difference in number of sets -1

p_1 and p_2 are penalty weights



Man hat am Ende einen planaren Grafen.

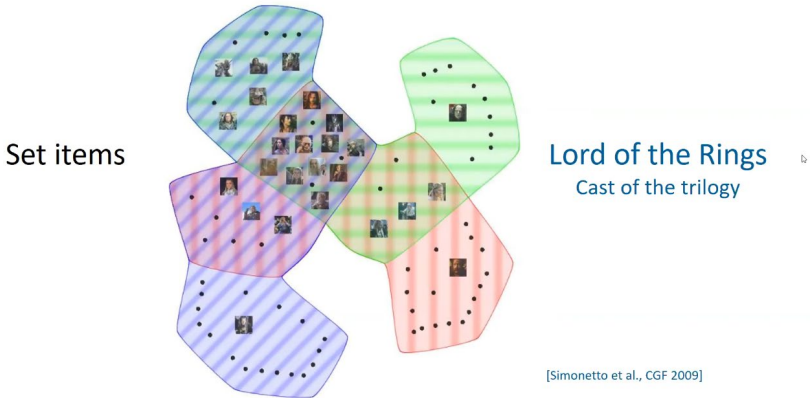


Planar graph drawing algorithm (force-directed layout)

Set boundaries around all nodes in same class (Bézier curves)

Schließlich muss man noch die bubbles zeichnen. Dann hat man so ein Euler-like Diagramm.

Konstantin Lackner 19.01.2021



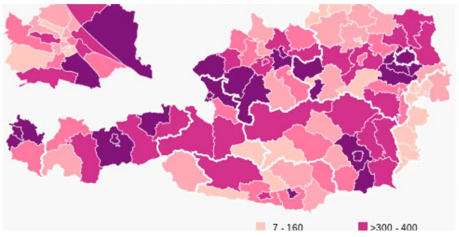
Geospatial Data

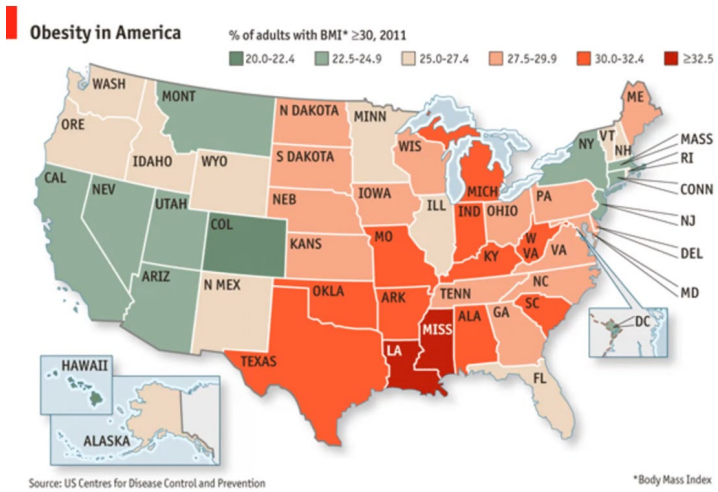
Solche, die geografischen Bezug haben. Damit sind die Daten inhärent in 2 oder 3D strukturiert.

Choropleth maps

Choropleth maps

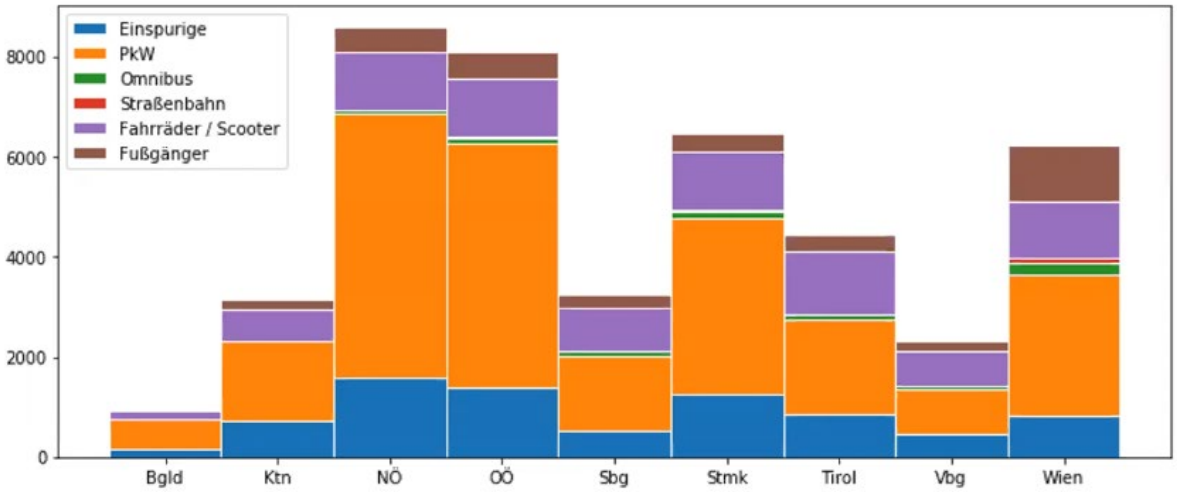
- Shows numeric attributes within predefined discrete regions
- Classed or unclassed color scheme for encoding





Hier ein Negativbeispiel. Man verwendet **sequentielle Daten**, es geht von 0 bis theoretisch 100%. Die Farben hingegen sind aber nicht sequentiell. Man bräuchte eine **diverging colour scheme**. So eines hätte einen Nullpunkt, eben ein Normalgewicht.

Es gebe auch noch kategorische Daten, das wäre bspw. sowas:

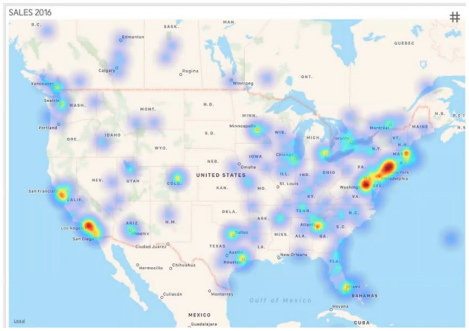


Die müssen einfach gut unterscheidbar sein.

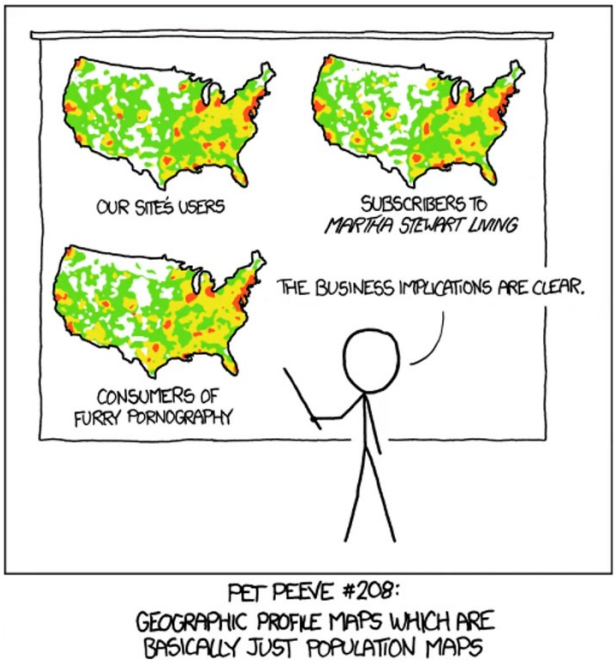
Heatmaps

Man kann geografische Heatmaps machen.

Values represented as color overlays



Heatmaps haben immer das Problem, dass dort wo viele Leute sind, viele Dinge passieren…



Konstantin Lackner 19.01.2021

Spatio Temporal Data

Wenn geografische Daten räumlichen und zeitlichen Bezug haben, gibt es drei Arten:

Spatial time series

Räumliche Zeitserien. Man hat eine Location und die Zeit als key values und ein Attribut als value, der mit Raum und Zeit assoziiert ist. Bspw. Coronainfektionszahlen an gewissen Ort zu gewisser Zeit.

Spatial event data

Man hat ein diskretes item, ein Event, das charakterisiert ist durch Zeit und Ort. Also hat man einen flat table mit items als Zeile und Zeit und Ort sind Spalten.

Movement data

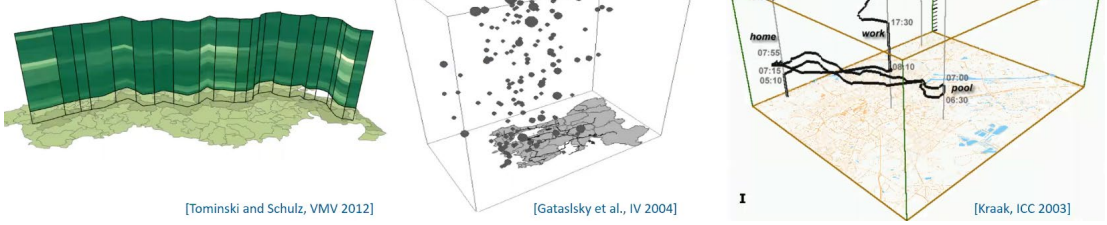
Die Daten zeigen hier sich bewegende items. Man liest also aus, welches item wann wo war. Item und Zeit sind also keys, space ist das Attribut.

Space-time cube

Ist eine Art zur Darstellung solcher Daten.

- Geographic base representation (x- and y-axis)
- Time axis (z)

Time Series | Events | Movement

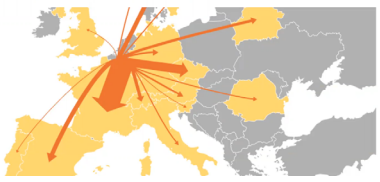


Nicht auf 2D Bildschirmen zu empfehlen.

Flow Maps

Auch eine Vis Technik für sowas.

Cartographic representation of aggregated movement items
Lines or curves connecting location pairs
Line thickness proportional to item counts



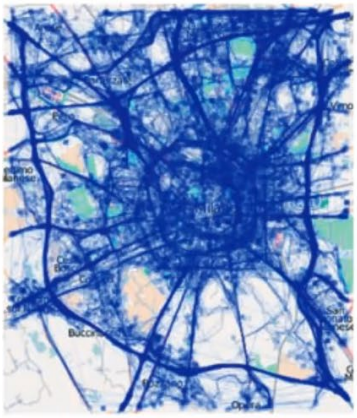
Man hat hier bspw. eine Startposition und verschiedene Ziele. Je dicker der Pfeil, desto höher die Anzahl der Waren/Person, was immer da dargestellt wurde. Skaliert aber nicht gut.

Konstantin Lackner 19.01.2021

Generalized flow maps

Nehmen Trajektorien und aggregieren daraus eine bessere Darstellung

Extended flow maps also take intermediate positions into account



~6.000 car trajectories

Aggregation along trajectory segments

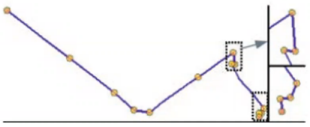
All trajectories aggregated (segmented into 1.500 m bins)

[Andrienko & Andrienko, TVCG 2010]

Zuerst **extrahiert man keypoints**. Das sind Start und Endpunkte, oder solche wo es einen significant turn gab (starke Richtungsänderung) oder eine lange Pause, usw. Wenn es lange segmente gibt, werden die unterteilt.

Extract key points from trajectories

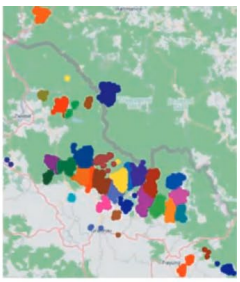
- Start & end points
- Points of significant turns
- Points of significant stops (pauses)
- Long straight segments are split

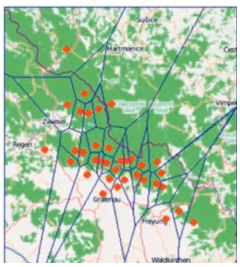


Mit den keypoints wird dann im nächsten Schritt **gruppiert**. Man schaut also wo viele keypoints nah beieinander sind.

Generalized flow maps

- Grouping of key points based on spatial proximity
  - Proximity threshold defines level of detail
- Extract group centroids
- Generate Voronoi cells
  - Using Delaunay triangulation





[Andrienko & Andrienko, TVCG 2010]

Dann berechnet man sich **pro cluster den Mittelpunkt und generiert sich Voronoi Diagramme.**

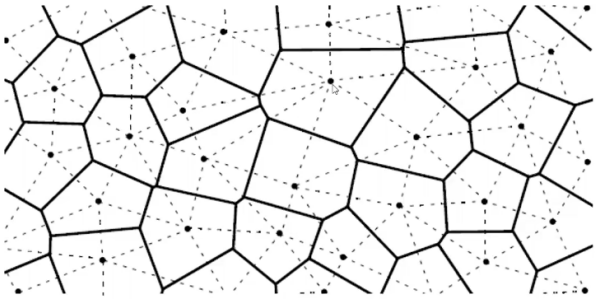
Konstantin Lackner 19.01.2021

Voronoi Diagramme bekommen als Input n Punkte auf einer Ebene und der Output sollen n Polygone sein. Die Eigenschaft der Polygone soll sein, dass jeder Punkt auf der Polygonlinie näher ist am Mittelpunkt, um den er entstanden ist, als zu irgendeinem anderen.

Partitioning of a plane with n points into convex polygons

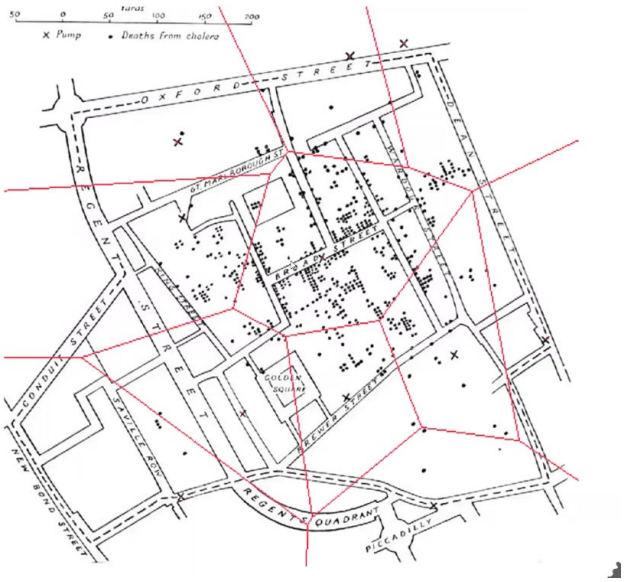
Each polygon contains exactly one point

Every point in a polygon is closer to its generating point than to any other



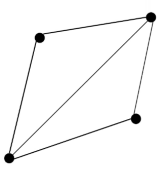
https://www.quora.com/How-do-you-make-a-Voronoi-diagram

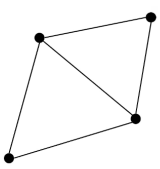
Die Sinnhaftigkeit an einem Beispiel. Zurück zu John Snow. Man nehme als Punkte alle Wasserbrunnen.



Hier sieht man dann für jeden Choloratoten den nähesten Wasserbrunnen.

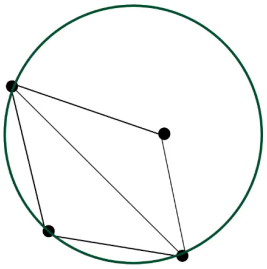
Man macht solche Volonoi Diagramme, indem man eine Delaunay Triangulierung macht. Man hat eine Punktwolke und macht daraus ein Dreiecksmesh. Dabei muss in jedem Umkreis um ein Dreieck herum kein anderer Punkt sein.



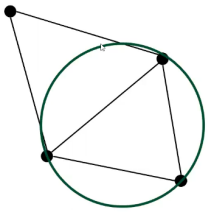


Rechts wäre also gültig, links nicht. Das ist auch eine häufige Prüfungsfrage!

Non-Delaunay Triangulation



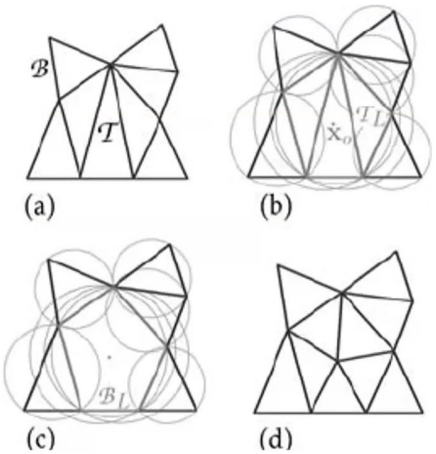
Delaunay Triangulation



Man kann solche Delaunay Triangulierungen mit dem Bowyer-Watson Algorithmus erzeugen.

Incremental algorithm to compute Delaunay triangulation:

- Start with very simple initial triangulation enclosing all points
- For every added point:
  - Delete all triangules violating circumcircle property
  - Insert new triangles

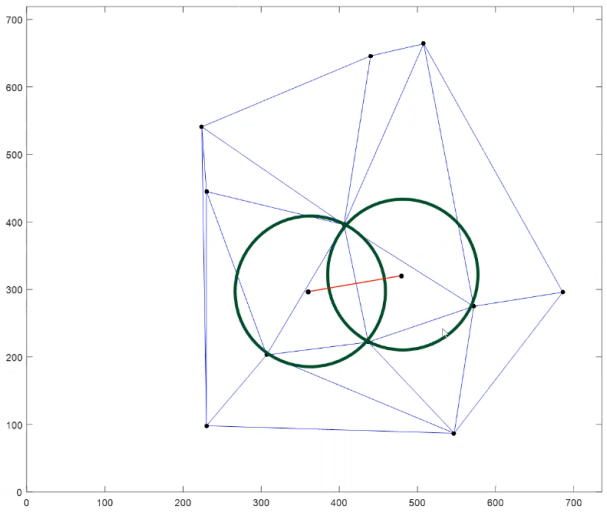


Das ist einfach eine schrittweise Antastung.

Das ist deswegen alles notwendig, weil die Vertices des Voronoi Diagramms immer genau im Zentrum der Kreise der Delaunay Triangulierung liegen. Die Edges(hier rot) des Voronoi Diagramms gehen immer genau durch die Mitte der Edges der Delaunay Triangulierung und das immer im rechten Winkel.

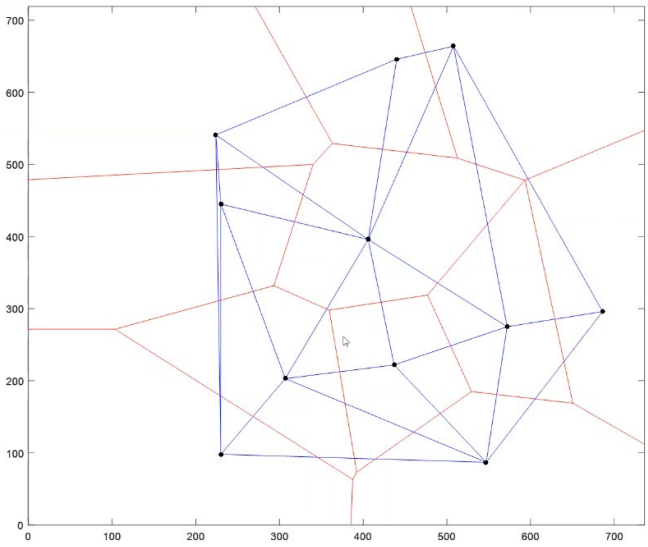
Delaunay triangulation and Voronoi diagram in $\mathbb{R}^2$ are dual:

- Voronoi polygon associated with a polygon vertex is defined by perpendicular bisectors of its Delaunay edges
- Vertices of Voronoi polygons are circumcenters of Delaunay triangles



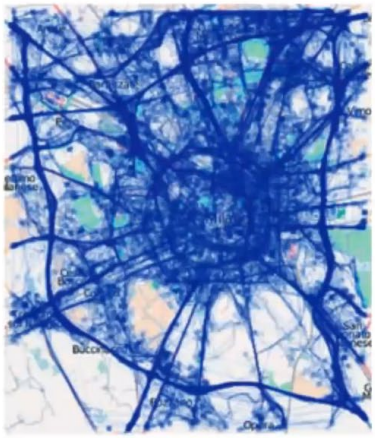
Wenn man das also für alle Dreiecke macht, bekommt man das Voronoi Diagramm.

Konstantin Lackner 19.01.2021



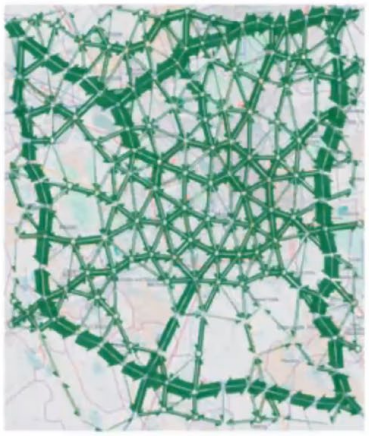
Das ist eben der letzte Schritt. Man ordnet noch alle trajectories diesen Voronoizellen zu – also den nähersten generating point finden und so zuteilen. Dann kann man das alles zeichnen und je nachdem, wie viele trajectories zwischen zwei so Zellen hin und hergegangen sind, zeichnet man dann eben stärker die Wege ein.

Generalized flow maps



~6.000 car trajectories

All trajectories aggregated
(segmented into 3.000 m bins)



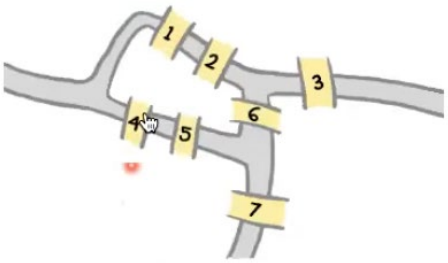
All trajectories aggregated
(segmented into 1.500 m bins)

Finale take-away message

Man kann nicht nur anhand der Daten sagen, wie die Vis aussehen soll. Sondern auch an den Usern, dem task, usw..

8. Network Vis

Königsbergproblem

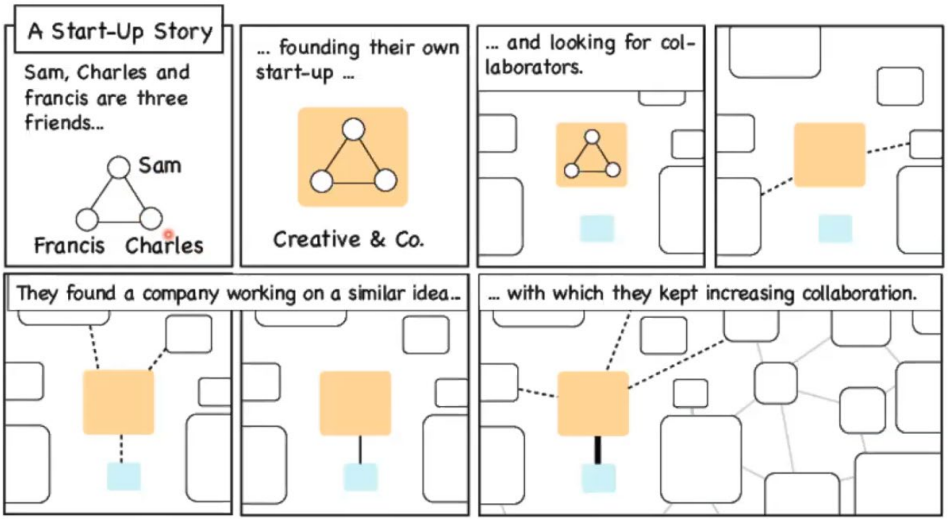


Wie kann man einen Pfad finden ohne je eine Brücke zweimal zu benutzen?

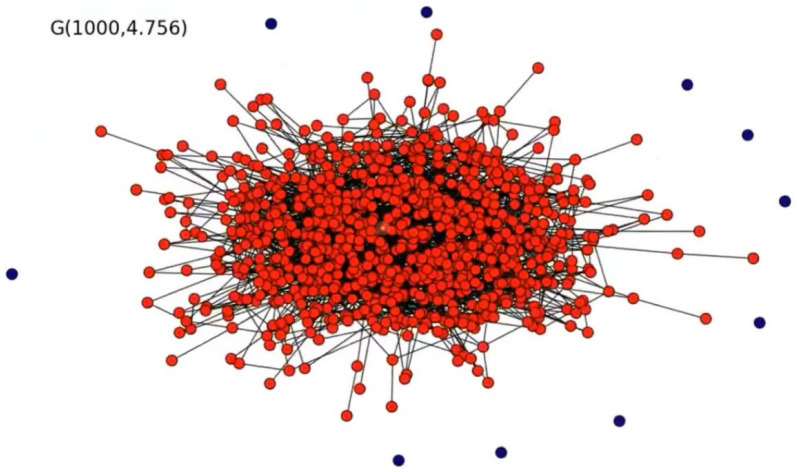
Hier hat man eine Simplifizierung. Am Ende hat man islands als Nodes.

Vortrag is auf Englisch… ich wechsel jz auf Englisch.

New example – start up company



The three devs consider themselves as strongly connected square. They also have a slightly less strong connection to their contractors, as well as their competitors. **This concept is called graph comic**. This is only feasible for a certain amount of connections.



This effect is called **hairball effect** – it basically looks like a hairball.

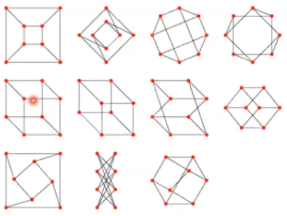
Recap terminology

Vertices (nodes) connected by **Edges** (links)

- Graph edges can be directed or undirected
- The degree of a vertex is the number of edges connected to it
  - In-degree and out-degree for directed graphs
- Graphs can have cycles
- Graph edges can have values: edge weights on them (nominal, ordinal or quantitative)

Related to

- Data structure, discrete math, graph theory, computational geometry, optimization, and … etc.

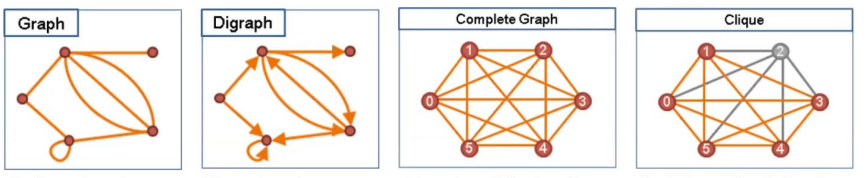


Graphs with various layout
http://mathworld.wolfram.com/GraphEmbedding.html

In directed graphs, you can differentiate between in and out edges and hence can differentiate the in and out degree as well.

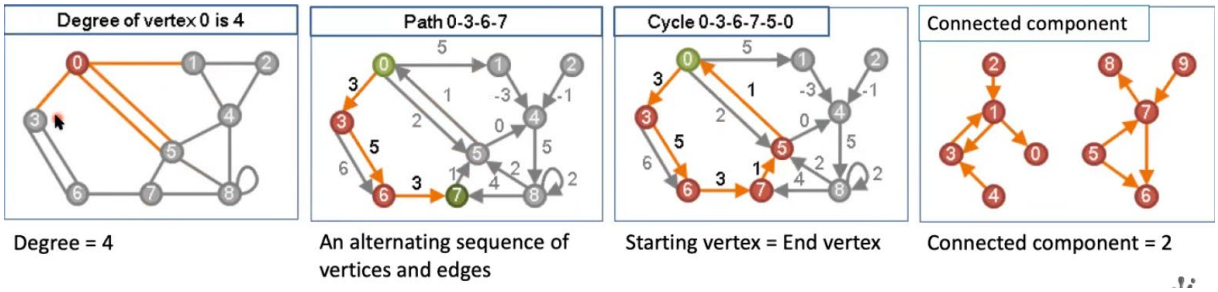
Definition of graphs

- Graph $G = (V, E)$
- Vertices $V = \{v_1, v_2, \dots, v_n\}$
- Edges $E = \{\{v_i, v_j\} | v_i, v_j \in V\}$



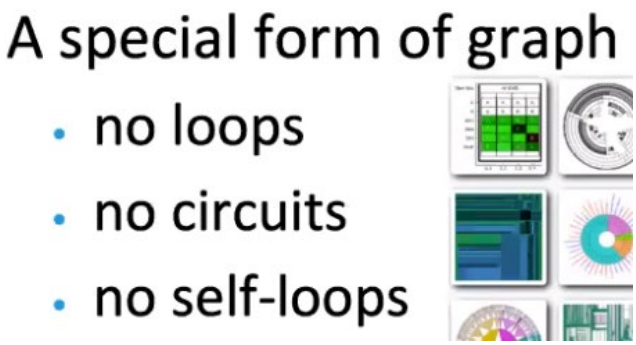
Undirected graph | Directed graph | Every two distinct vertices are connected | Apply to a subset of vertices of an undirected graph

Graphs consist of a set of vertices and a set of edges. In complete graphs, every vertex is connected to every vertex but itself. A clique is not inherently a subset of a graph. Apparently applying a set makes it so the graph is reduced to a subset of itself that is still complete.



The total connected components in the image on the right is two, as there's two graphs in it.

Trees



Why graphs?

It's the simplest way to represent relations.

Some more terms

Graph drawing – a pictorial representation of a graph. Only vertices and edges.

Node-Link diagram – special case of a graph drawing. vertices/Nodes are disks or labels and edges/links are lines or curves in the euclidean space.

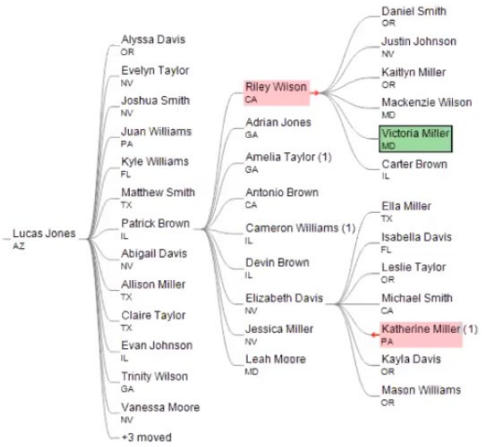
Network visualisations – Just again a more complex version. – she doesn't really explain any further.

Konstantin Lackner 19.01.2021

Examples for graph visualisations

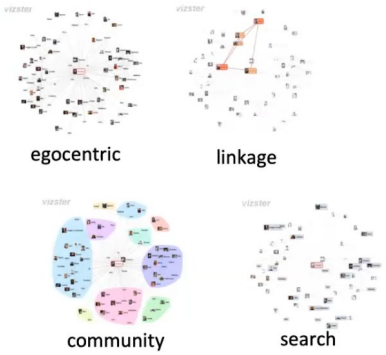
Tree-based

- TreePlus [Lee et al.: 2006]



network-based – We see egocentric view (based on yourself), linkage – so connections between e.g. friends, community which shows the groups in the graph or search which looks for certain features.

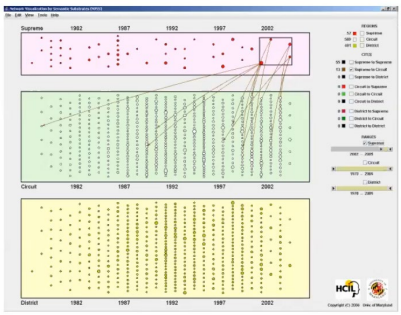
- Social networks
  - VIZSTER [Heer et al. 2005]

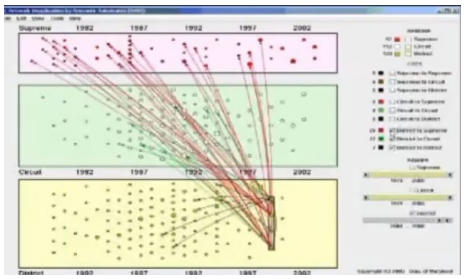


Citation networks – The vertices are aligned in parallel in subgroups.

Citation network

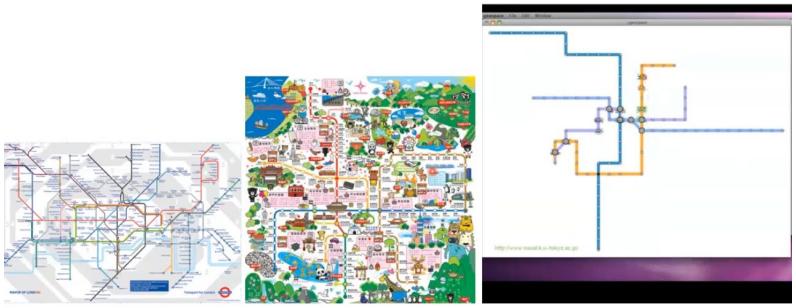
- Network Vis by Semantic Substrates





Traffic networks – extra info can be added.

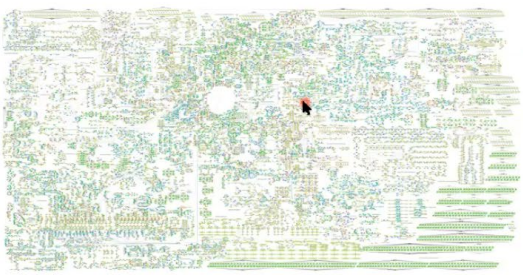
- Traffic networks
  - User customized metro map [Wu et al.: 2013]



Harry Beck's map | Milu Design company | [http://yun-vis.net]

Metabolic networks - This is the largest human drawn graph.

- Metabolic Networks
- Reflect the set of biochemical reactions in a cell
  - Nodes: metabolites
  - Edges: biochemical reactions
- Derived through:
  - Knowledge of biochemistry
  - Metabolic flux measurements



ReconMap 2.01 (Human Metabolism)
Website: https://vmh.uni.lu/minerva/?id=ReconMap-2.01

Konstantin Lackner 19.01.2021

Understanding relationships

To prevent the hairball effect, one needs to untangle networks by understanding what is and what isn't necessary. There's a few ways to do this.

(a) Minimize node-node/node-edge occlusion

(b) Minimize edge crossings

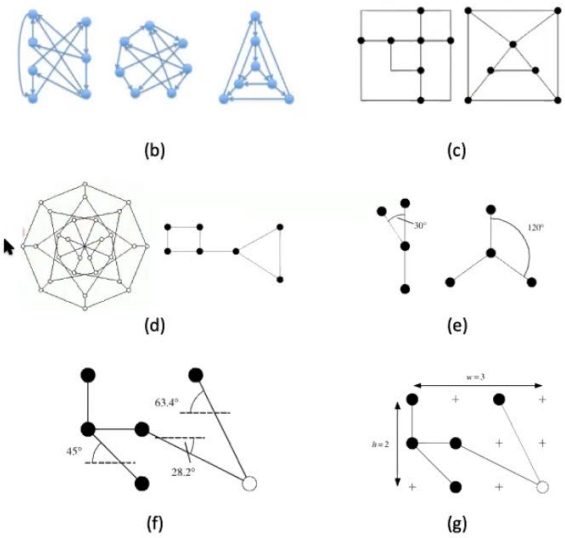
(c) Minimize edge bends

(d) Maximize symmetry

(e) Maximize the minimum angle between neighboring edges

(f) Maximize edge orthogonality

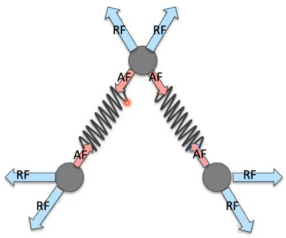
(g) Maximize node orthogonality



Algorithms overview

Node-Link-Diagram

This is created using a **forced based algorithm**. This means that theoretical physical forces are applied to simulate nodes pulling at each other.



A concept image

So there's a spring force and a RF force. the springs pull the vertices together, the RF pulls them apart, this should, in the end, make for a balanced graph.

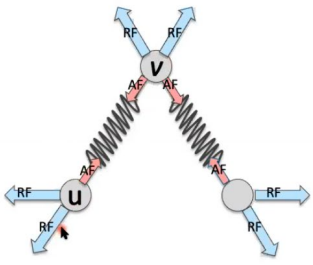
- Replace an edge by a spring with unit natural length

$$f(u,v) = k_a \log(l)$$

- Connect nonadjacent vertices with additional springs with infinite natural length

$$g(u,v) = k_r / d^2$$

where



A concept image

k_a, k_r : constants $\quad d$: distance between vertices

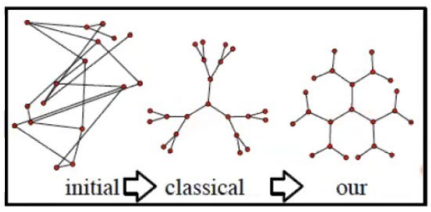
The **RF** = **repulsing force** is always directed away from the other nodes.

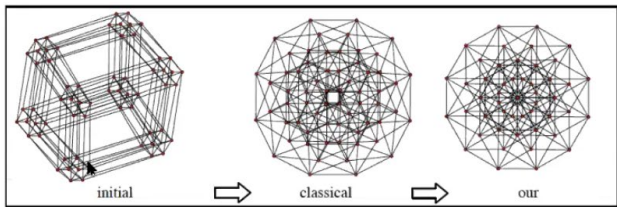
The force on vertex v is

$$F(v) = \sum_{(u,v)\in E} f_{uv} + \sum_{(u,v)\in V\times V} g_{uv}$$

However, the method needs a lot of computing power and doesn't yield all too great results – especially considering the effort.

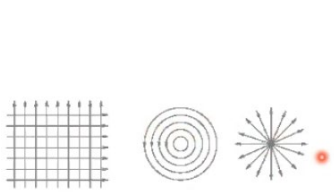
A way to improve this is edge-edge repulsion. **Not only the nodes act repulsive now, so do the edges**. Edges now repulse each other as well.



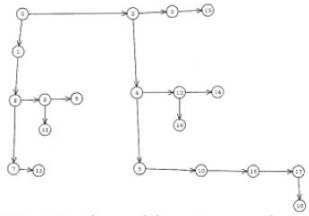


initial is without any force-based algorithm, classic without **edge-edge repulsion** and the final version, our, is the edge-edge improvement.

You can, obviously also use magnetic fields as way to calculate the attraction.

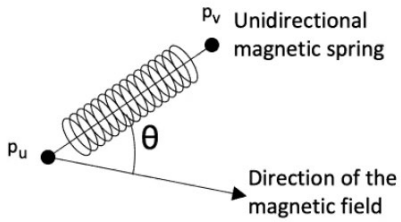


Types of magnetic field



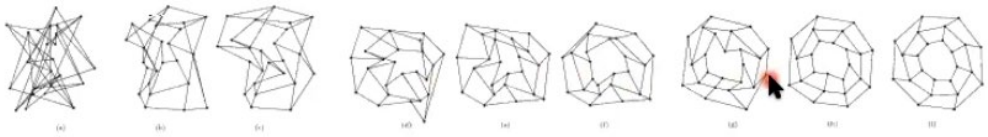
Vertical and horizontal magnetic field

eld



Magnetic spring

An energy-based approach would mean that you can introduce a concept called **cooling factor**. So at the beginning of your simulation everything moves quite fast until it cools down in an ordered manner. The movement is vertex movement. you start by moving random vertices and check, after moving, if the new result is better than the first one. You do this, until your system cooled down to a certain point.



Konstantin Lackner 19.01.2021

Mental Map

Not an algorithm per se but a way to preserve graph shapes when organising a graph.

If there's different relationships in different times, a dynamic visualisation is needed. There's a few objectives here: (not all of these must apply)

1.) Ranking – If the system changes, the relationship between two relative positions, i.e. vertices, I guess, shouldn't change too much

2.) k-matrix – Preserve the clockwise or counter clockwise orientation of the vertices.

3.) unweighted nearest neighbour within – A node's nearest neighbour should remain its nearest neighbour.

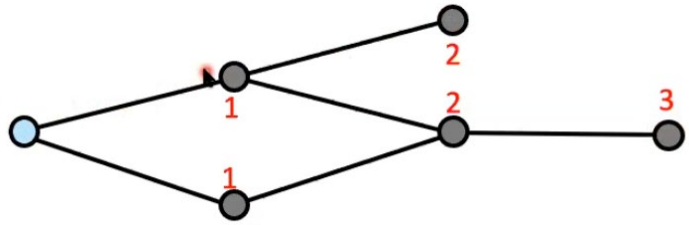
4.) Weighted NN between – A node should remain close to its original position.

5.) shape – preserve edge orientation.

Multidimensional Scaling (MDS Layout)

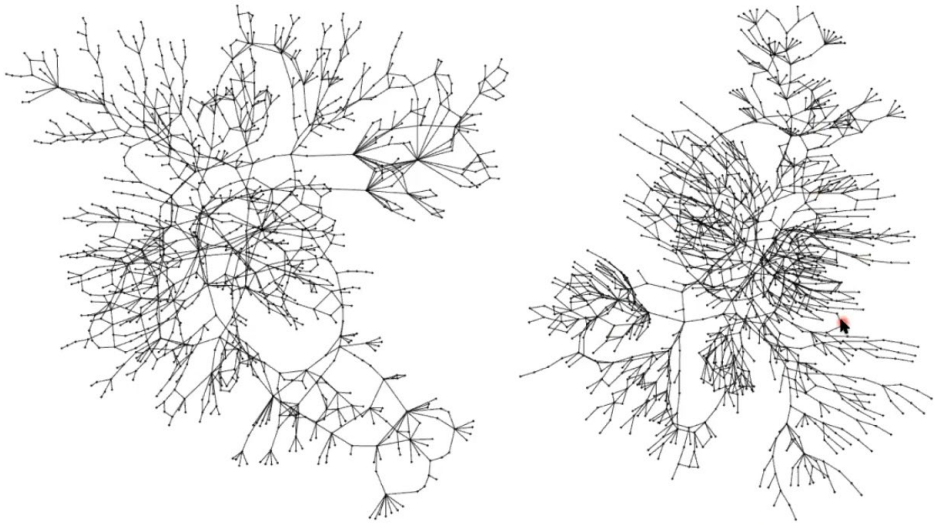
Stress majorization

Here you want the distance between two nodes in the 2D representation as close to their actual distance in m-d space as possible.



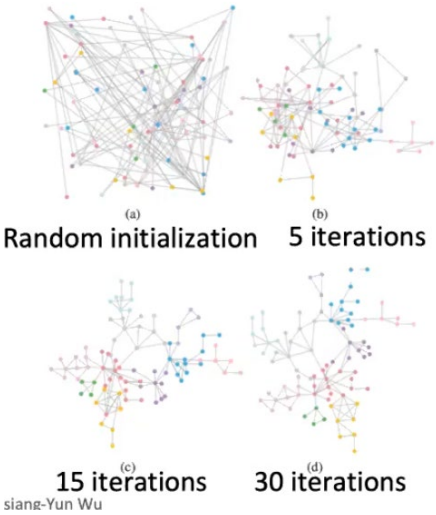
$$Stress(X) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{C}{{d_{ij}}^2} (\|X_i - X_j\| - d_{ij})^2$$

So the distance between the blue vertex and the grey vertex with the red 1 next to it (the top one) is 1 unit. So is the distance between that grey vertex and the next grey vertex is 1 again, making the distance between the light blue vertex and the second grey vertex two units.



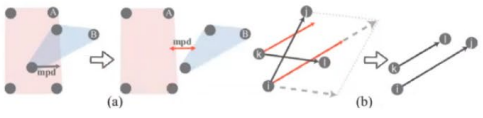
Weights can also influence the distance between the nodes when using this stress formula.

In an approach to make this mess even more complicated, somebody invented a way to include edge orientation into the system. So basically the edge orientations would be reduced to only a few directions, making the image clearer.



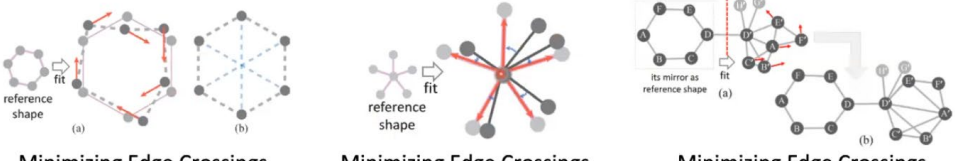
There's also other constrains here…

- Metrics-based Constraints



Non-overlap Constraint Minimizing Edge Crossings

- Shape-based Constraints



Minimizing Edge Crossings Minimizing Edge Crossings Minimizing Edge Crossings

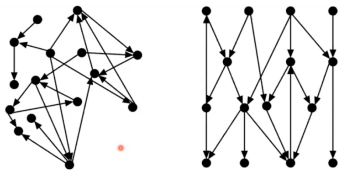
Konstantin Lackner 19.01.2021

Hierarchical Layout

Layered drawing

This is a very old approach. It is done by breaking cycles heuristically. The algorithm has 4 steps:

Step1: Optimal Rank Assignment
Step2: Vertex Ordering Within Ranks
Step3: Node Coordinates
Step4: Drawing Edges

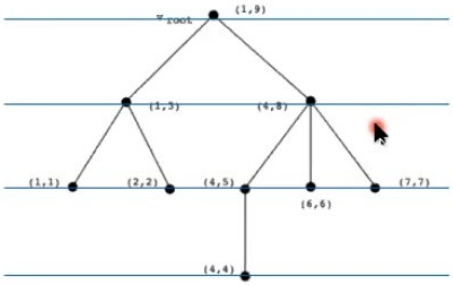


If the hierarchy is not in the data, you'll have to find an order that's step one.

In step 2 you have to assign that number/order to the vertices. Step 3 is to find the node coordinates and step 4 to draw the edges. This doesn't explain anything yet though.

Step 1

If you have a root, go through the nodes and assign them a hierarchy level.



Step 2

Swap vertices within the ranks to prevent edge crossings. Assign them a new order.

Step 3

After reordering the vertices, calculate their new final positions.

Step 4

Finally, draw the edges and incorporate e.g. edge bends to resolve issues in your graph.

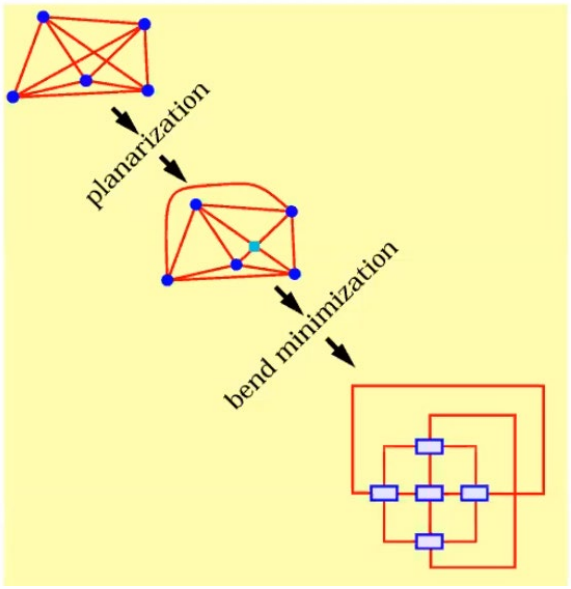
Schematic layout

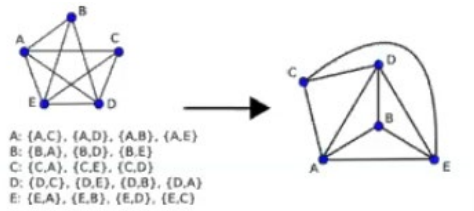
Orthogonal drawing

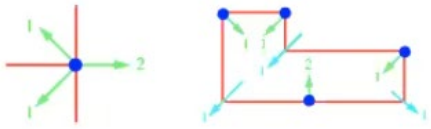
This is only possible on graphs that have a maximum of 4 edges per vertex. What you're trying to do is to minimise edge bends. You do this by finding a planar graph (one without edge crossings) first. (with edge bends at first) then you try to minimise the number of bends. This somehow works by looking at the corners and doing something, she doesn't really manage to explain.

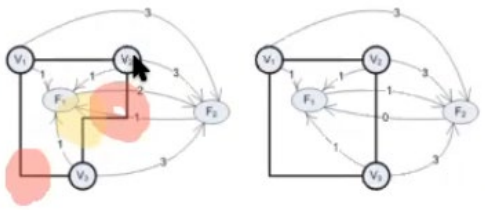
Algorithm

- Find planar embedding
  - Boyer-Myrvold Planarity Embedding
- Minimize edge bends
  - Can be formulated as a network flow problem

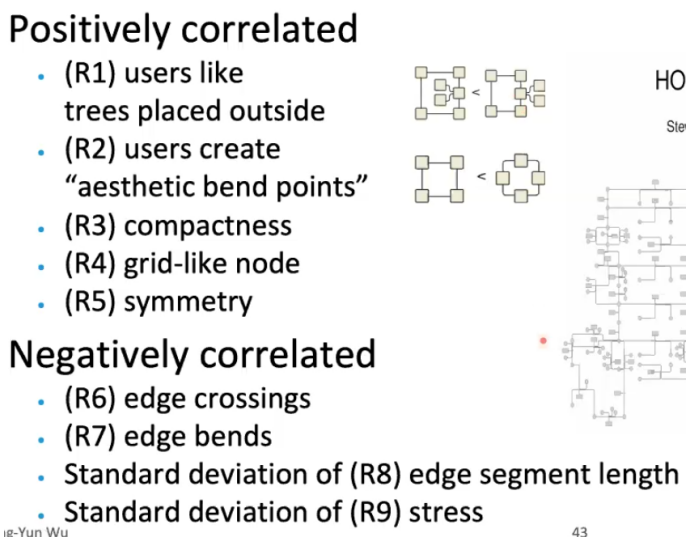








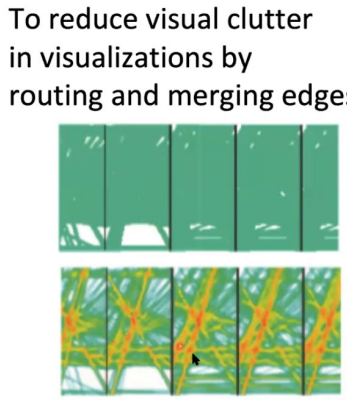
You're checking for 90 degrees corners. If you find one, you label the edge 1, if you find two, you label it 2… and so on.



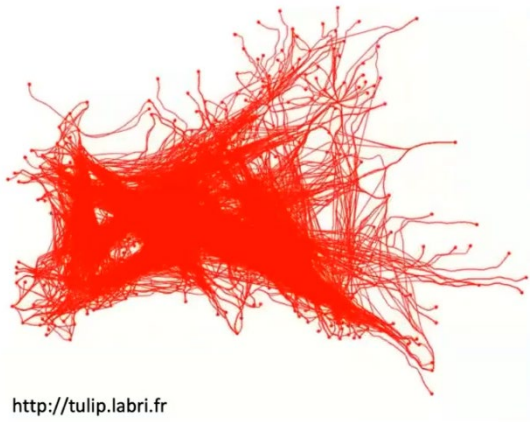
Edge Drawing

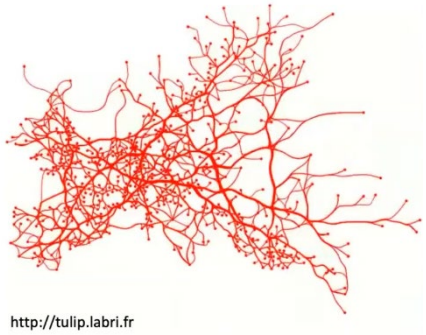
Edge bundling

Instead of spreading edges and trying to place them far enough apart of each other, if there's simply too many edges, bundling them can be an option.

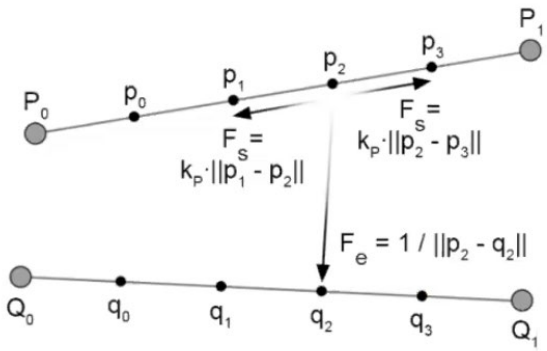


This is edge splatting, where you don't draw all the edges with the same opacity and colour but instead change colour and opacity according to the amount of edges drawn at that point.





One way to realise this is a force-directed edge bundling. This works with springs again as before. There's springs between certain points on the edges here, not just at the vertices. The spring forces pull the edges together.

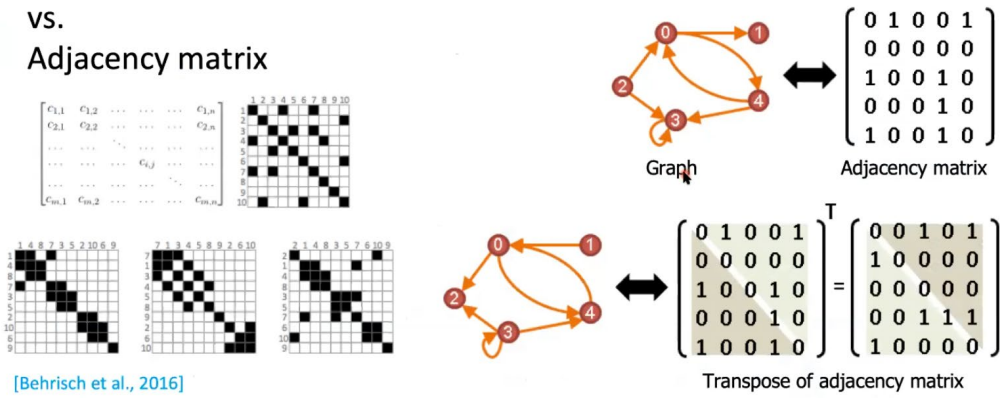


Forces added to the edges

Matrix Representation

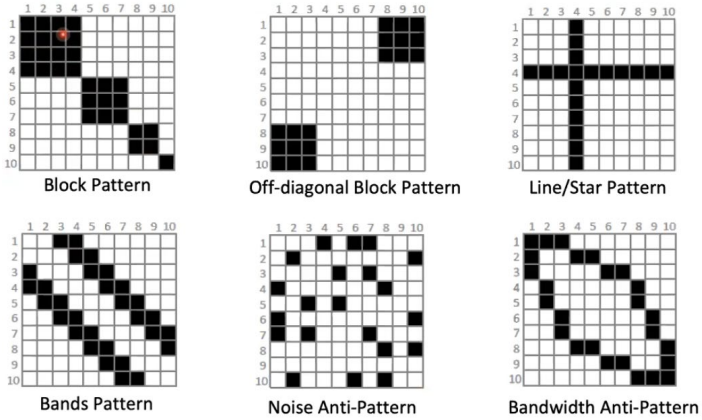
Adjacency matrix

- Node-link diagram
  vs.
  Adjacency matrix



A graph can be represented as a matrix. The columns are vertices and the rows are as well. This also means that you can easily transpose graphs. (as they're matrices anyway)

With the matrix patterns can be drawn. Doing this, you'll notice some common patterns.

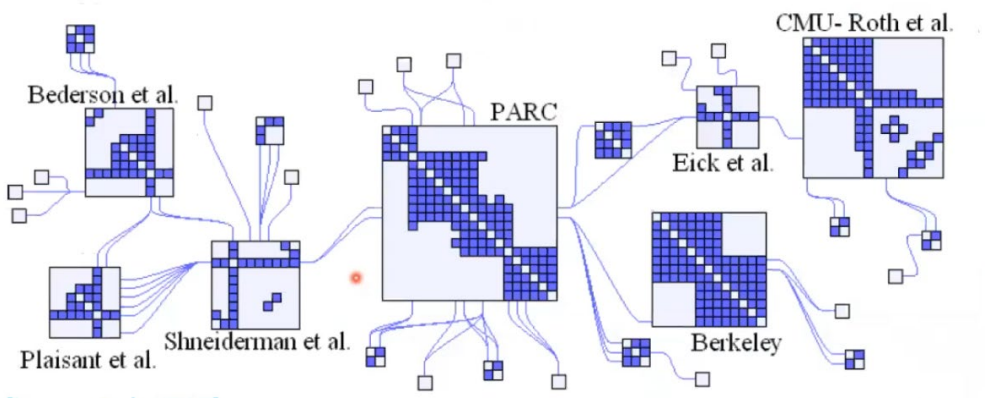


Block patterns for example usually show very busy, cluttered parts in a graph.

Hybrid representation

Combining different representations.

e.G. NodeTrix – This is a way to present large social networks.

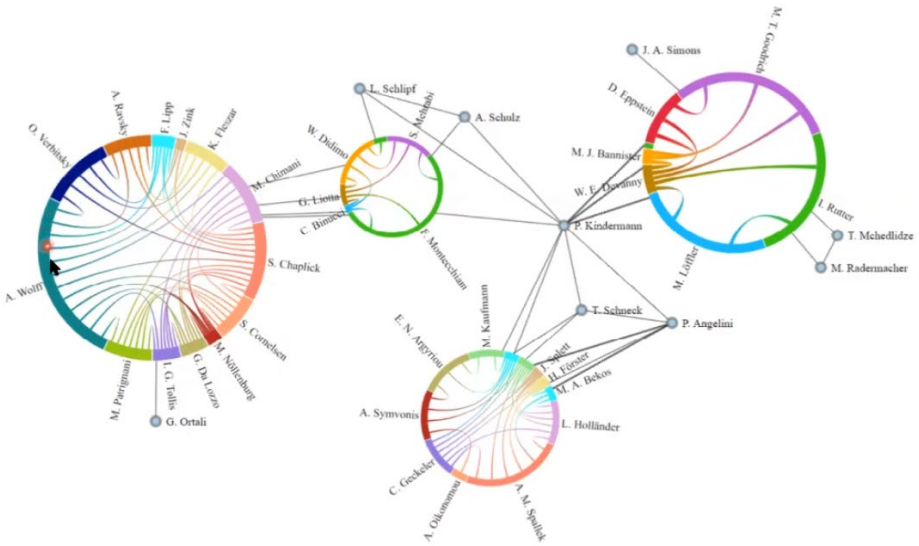


Large social network visualization
Aggregate / split, order, merge matrices

They tried to merge node-link diagram and matrix approaches.

ChordLink

This is another hybrid approach. The vertices are ordered along a circle.



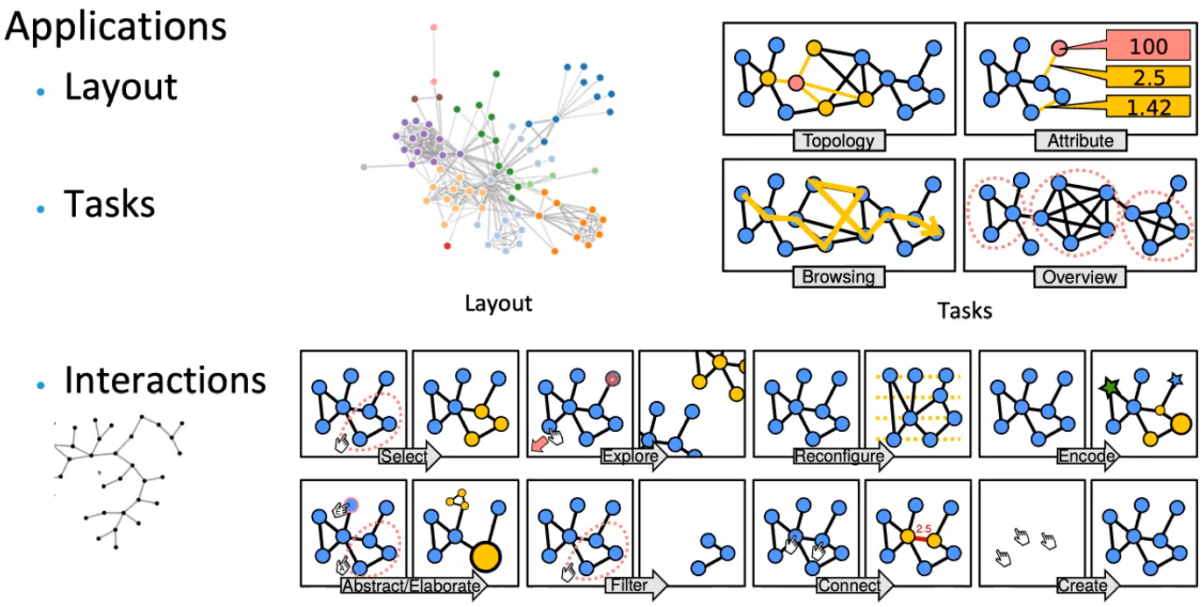
The interconnections withing the circles are easily visible here.

This is useful if there is a lot of interconnection and not too much connections between the circles. So a clear grouping.

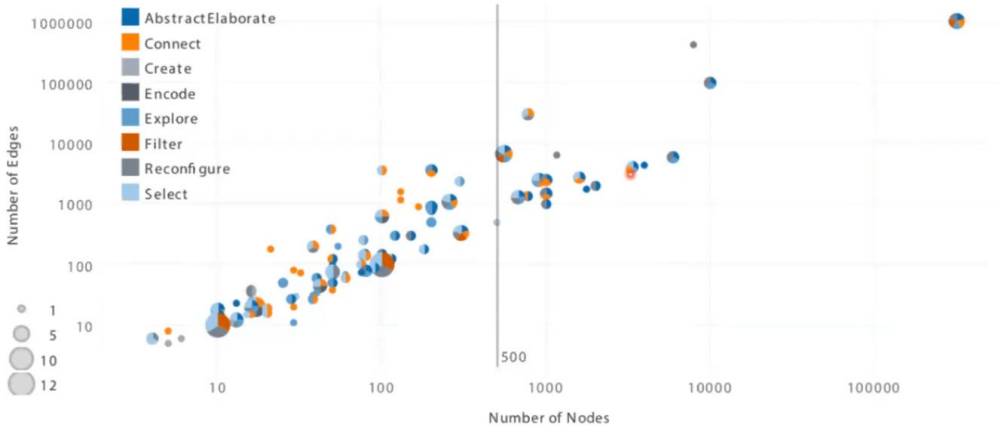
Konstantin Lackner 19.01.2021

State of the vis

State of the art visualisations are always combinations between different approaches, with interactivity features and the like.

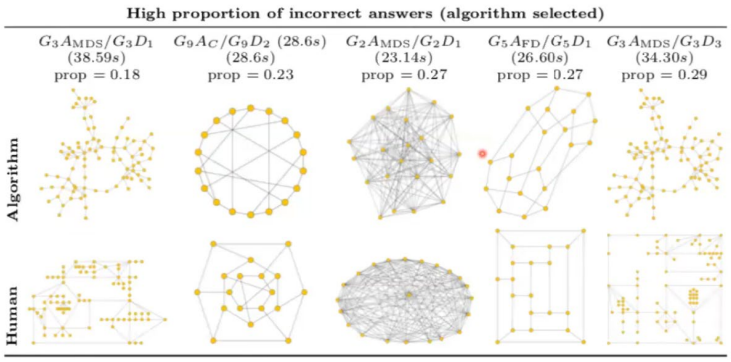


Limits of complexity



Human vs Machine

A normal human can't identify if a graph is drawn by a human or not.

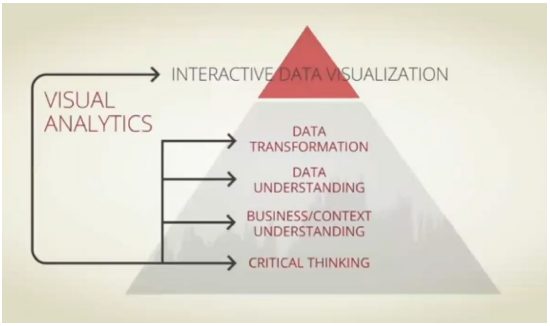


Konstantin Lackner 19.01.2021

9. Visual Analytics und Visual Data Science

Einführung

Visual Analytics ist die Kombination, die visuelle Intelligenz und Analysefähigkeit mit Technologie verbindet, um Daten zu verstehen. Unterschied zwischen interactive vis und visual analystics ist, dass das eine ein Teil des anderen ist. Analytics ist das breite Feld, interactive vis nur ein Grundgebiet davon.

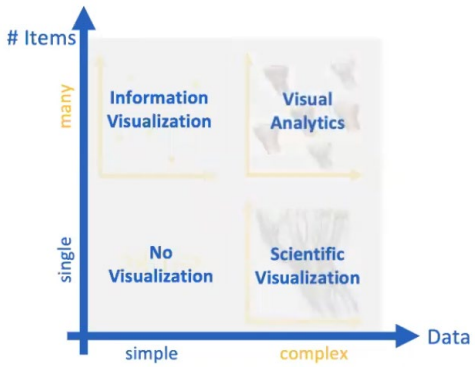


Außerdem ist vis analytics sehr back and forth lastig und ein verstrickter Prozess.

Es ist human centric – der Mensch ist im Mittelpunkt des Geschehens.

An sich gibt es verschiedene Zugänge…

Visual Analytics ist das Bindeglied zwischen all den vorgestellten Visualisierungen. Es ist die Wissenschaft des analytischen Schlussfolgerns. die unterstützt wird durch interaktive visuelle Interfaces. **Man hat immer ein Interface**, ein multiple view interface, mit views, die alle verbunden sind.



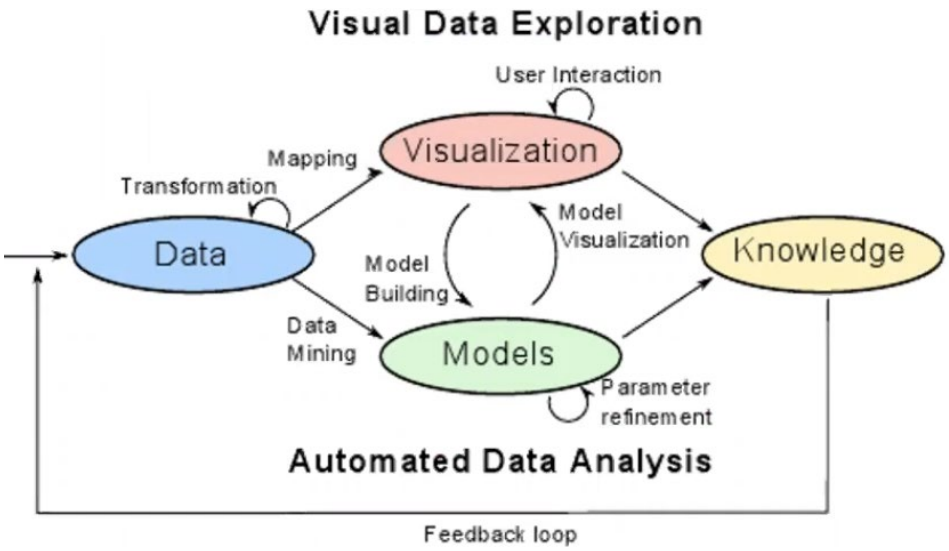
Visual Analytics ist also mit komplexen Daten und vielen Items(/Datenobjekten) gleichzusetzen.

"Visual Analytics is the science of analytical reasoning supported by a highly interactive visual interface." [Wong and Thomas 2004]

"Visual Analytics combines automated analysis techniques with interactive visualization for an effective understanding, reasoning and decision making on the basis of very large and complex datasets" [Keim 2010]

Konstantin Lackner 19.01.2021

Visuelle Analyse Prozess



Man hat 4 Komponenten **Daten**, **Visualisierung**, **Modelle** und **Wissen**

Daten können transformiert und aufbereitet werden für die Aufnahme.

Aus den Daten kann man Modelle ableiten mit Data mining, man kann die Daten auch direkt mappen in renderable content und in die vis einfließen lassen.

Bei den Modellen hat man wieder viel User interaction zum Verfeinern oder Ableiten von Wissen durch die Modelle. Bei den Modellen ist es wichtig Parameter, bspw. den Isowert, richtig einzuteilen. Das kann ein Prozess sein, wo der Isowert schon berechnet wurde, also für jeden Isowert eine Darstellung, oder man verwendet das Modell um den perfekten Isowert zu finden. **Model vis** is die Darstellung des aktuellen isowerts, **Model building** ist, wenn man über die Vis den richtigen Isowert einstellt und findet.

Generell ist das **Ziel Wissen zu generieren**. Man kann das datenbasiert oder visualisierungsbasiert machen. Datenbasiert könnt eine KI bspw. alle Daten durchforsten und dann iwas ausrechnen. Vis. eh klar.

Der erste Schritt im visuellen Analyseprozess ist das Datenvorbearbeiten. Bspw. normalisieren. Der Knochen bspw. sollt immer denselben Grauwert haben. Daten müssen dabei auch korrigiert werden. Datenfusion ebenso. Hat man bspw. mehrere Modalitäten von Objekten. Also z.B. CT Daten von Patienten, aber auch Röntgenbilder und Ultraschalldaten… Bei der Vorverarbeitung hat man hauptsächlich automatische Methoden. Diese skalieren sehr gut, also mehr Daten sind generell kein Problem und können parallelisiert werden. Dennoch gibt es oft **lokale Optima** in denen man feststecken kann. Hat man bspw. Schwellwertoperationen, also innerhalb Schwelle ist das Objekt das eine, außerhalb das andere, so kann man sich sehr gut ausrechnen, was der Schwellwert sein muss. Wenn man aber eine komplexe Datenvorverarbeitung hat, bspw. viele Filter davor mit vielen Parametern, so gibt es einfach sehr viel anzupassen, was dann dazu führt, dass man zwar ein Optimum findet, aber das **kein globales Optimum** ist. **Kleine Variationen im Input führen also zu großen Änderungen auf globalem Level.**

Außerdem gibt es oft Blackboxes – bspw. KI Netzwerke. Die sind zwar schön und gut, aber leider totale Blackboxes.

Aus diesen Daten muss dann mittels Datamining weitere Info berechnet werden. Bspw. quantitative Information. Hat man bspw. Tumorgewebe?

Dann gibt es die Visualisierung, bei der interaktiv Daten analysiert werden. Man kann also visuell manuell vergleichen. Das ist allerdings nicht skalierbar. Weil eben Menschen arbeiten.

Die Idee hinter Visual Analytics ist, dass man eben beide Methoden verbindet, also sowohl die automatisierten, als auch die manuellen.

Ziele der Visual Analytics

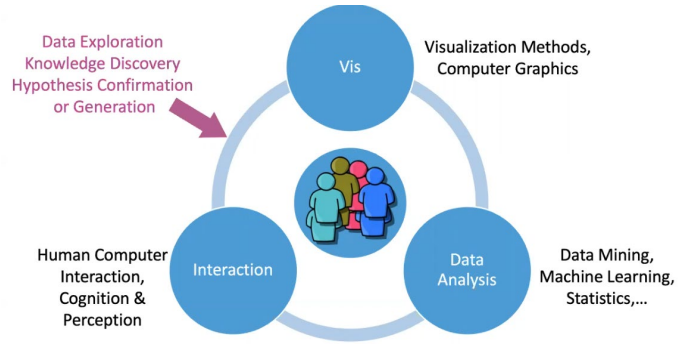
Man will **synthetisieren** – Nämlich Informationen, aus denen Einblicke in die Daten abgeleitet werden können. Die Daten sind dabei sehr groß, dynamisch, mehrdeutig und widersprechen sich teilweise. Außerdem hat man die Daten zum Analysezeitpunkt manchmal anfangs noch nicht, sondern muss auf Basis von predictions arbeiten.

Man will erwartete **Phänomene detektieren**, aber **auch unerwartete darstellen können**. Bei einem Dammbruch bspw. will man nicht nur die vorhergesehen Auswirkungen sehen können, also die Bereiche der Stadt zeigen, die am ehesten überflutet werden. Das Entdecken unerwartet Phänomene bezieht sich bspw. auf das Platzieren von Sandsäcken – kann man durch richtiges Platzieren vl. verhindern, dass eine Bibliothek überschwemmt wird?

Man will **schnell belastbare und verständliche Bewertungen** – Wettervorhersagen, muss schnell gehen, bringt bald nix mehr, ebenso der Dammbruch.

Bewertungen müssen kommuniziert werden können – Ergebnisse und Erkenntnisse müssen richtig transportiert werden. Informationen so aufzubereiten, dass sie auch von unskilled usern verständlich ist, ist die Kühr.

Visual Analytics Loop



Der Mensch ist im Mittelpunkt und will schnell belastbare Daten haben. Bspw. will er wissen, was die Wettervorhersage ist.

Man bedient sich dabei Methoden aus der Vis. Es ist also auch ein Bereich davon die Datenanalyse. Bspw. Big data Methoden – Machine learning, data mining, statistics.

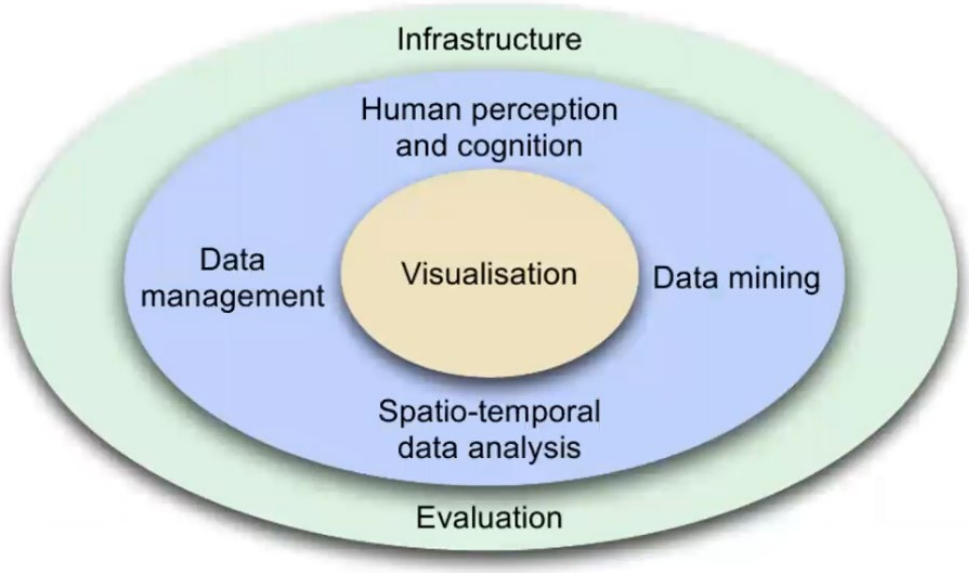
Durch die Interaktion wird das Entdecken von erwarteten und unerwarteten Dingen möglich.

Die Visualisierung dient hauptsächlich dazu die Ergebnisse an den User weiterzutragen und diese zu kommunizieren.

Darüber hinaus besteht die Möglichkeit im Loop Daten zu explorieren und zusätzliches Wissen zu entdecken, Hypothesen aufzustellen und zu bestätigen.

Konstantin Lackner 19.01.2021

Interdisziplinarität



Die Vis steht in der Mitte, aber will man Visual Analytics machen, braucht man viele andere Gebiete. Bspw. Data mining – hat man sehr komplexe Daten, die sehr kompliziert sind, kann data mining daraus neue Daten generieren und über die Daten gewisse Dinge aufdecken. Man hat räumliche, zeitliche usw. Daten. Man muss stets auch mitdenken, wie man Daten bestmöglich für User zu Verfügung stellt. Auch die Infrastruktur ist sehr relevant – welche Infrastruktur braucht man, um diese Probleme zu lösen. Auch augmented und virtual reality sind hier relevant. Man kann damit die Evaluierung fördern.

Auch die **Visualisierung selbst ist interdisziplinär**, nicht nur die Analyse. Hat man Daten im Hintergrund, die man visualisieren muss, hat man immer das Applikationsfeld im Hintergrund, für das man die Vis entwickelt. (bspw. Medizin)

Herausforderungen

Daten sind sehr **groß** und **variabel**. Sie sind in **Qualität unterschiedlich** und **sehr heterogen**. Man hat also oft auch **abstrakte Daten** und **gemischte Daten**.

Verschiedene User haben **verschiedene requirements**. Die **Nutzerziele** sind **von höchster Wichtigkeit**. Das **Skilllevel** bspw**. der User ist sehr relevant**.

Design ist wichtig. Das Tool muss entsprechend der **user requitements** aussehen. Es muss eben auch gut benutzbar sein.

Die **Technologie ist wichtig**. Die Infrastruktur an sich, also nicht nur die **Ausgabedevices**, sondern auch die **Hardwareinfrastruktur**, **wo die Daten gelagert sind**, sind relevant. Festplatte? Lokal? Cloud? Verteilt? **Müssen Daten zuerst aggregiert werden**? Wie sind die **Eingabegeräte**? Maus und Tastatur? Weiterführende Geräte? Ein Controller? Wie kann man die Daten ausgeben? VR Headset? 2D Monitor? 3D Brille? AR Headset? Tablet?

Konstantin Lackner 19.01.2021

Daten Management Challenges

Big Data – Sehr komplexe große Daten, aus denen man zuerst, bspw. mit Data Mining genauere Infos minen muss.

Uncertainty – Man hat bspw. bei CT Daten gewisse Unsicherheit. CTdaten können nicht per se für bare Münze genommen werden. Man muss die Daten auch hinterfragen. Ein Grauwert bspw. hat Einfluss auf das Messergebnis. Bewertet man bspw. eine Tumorgröße, muss man sehr genau sein.

Semantik – Die Bedeutung der Daten ist relevant. Gibt es Taxonomien?(Für die nicht-eloquenten = einheitliche Strukturen, also eigl. eh Klassifiaktionsschemata) Kann man Klassifikationsschemen anlegen? Kann man logische Netzwerke finden?

Data Streaming – Wie entstehen die Daten? Laufend, oder sind sie am Anfang schon alle da? Hat man eine komplexe Simulation, muss man bspw. je nach Komplexität des Modells das sehr zeitaufwändig. Die Simulationen laufen durch und Daten werden Stück für Stück weiteregegeben.

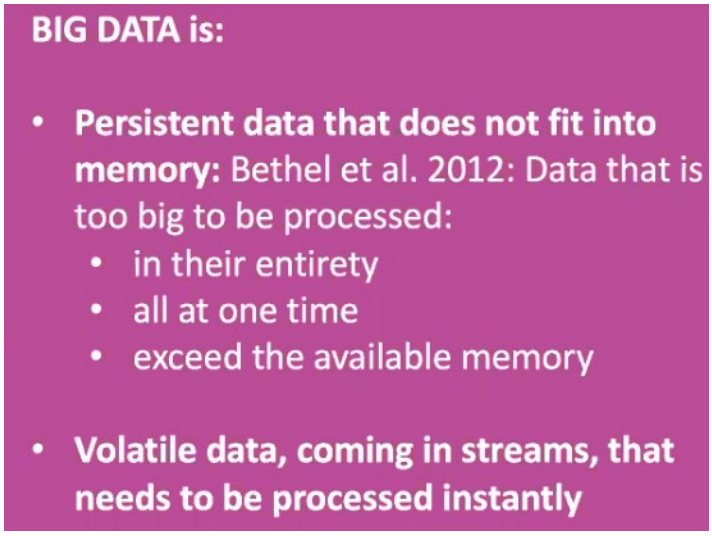
Distributed and Collaborative VA – Man kann Analyseprozesse auf mehreren devices bzw. mit mehreren Leuten auswerten. Welche Techniken funktionieren in diesem Bereich? Wie kann man die Techniken verbinden, sodass die Leute gut zusammen arbeiten können?

VA fort he Masses – Durch die Verfügbarkeit von Devices und die Verbreitung, hat man auch viel mehr Zugang zu diesen Visualisierungen. Smartphones und PCs werden dabei immer wichtiger.

Big Data

Big Data beginnt nicht bei einem fixen Wert. Die Grenze zu ziehen macht man eher mit der Komplexität der Objekte. Diese Grenze zu ziehen ist ein „moving Target“.

Wenn man Daten hat, die nicht mehr in den Speicher passen, spricht man von big data.



Man kann die Daten nicht mehr in der Gesamtheit betrachten, man kann sie nicht gleichzeitig betrachten und man überschreitet den Speicher.
Außerdem gilt es für Daten, die in Stream daherkommen und kontinuierlich bearbeitet werden müssen.

Es gibt aber sehr viele Felder, in denen man big data hat:

Meteorologie – Leibnizrechenzentrum in München für Wettersimulationen.

Genomics – Gene sind super kompliziert, Genomics aber für Krebsforschung und personalisierte Medizin sehr wichtig.

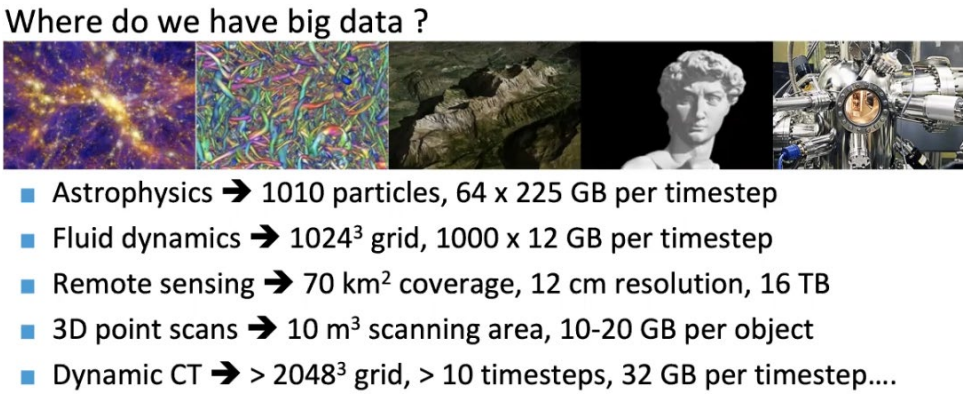
Connectomics – Analyse von Nervenzellen im Gehirn und besonders das Netzwerk der Nervenzellen im Hirn. Man will das Gehirn besser verstehen und nachbilden können.

Komplexe Physiksimulationen – Bspw. Untersuchung der Materienverteilung im Universum.

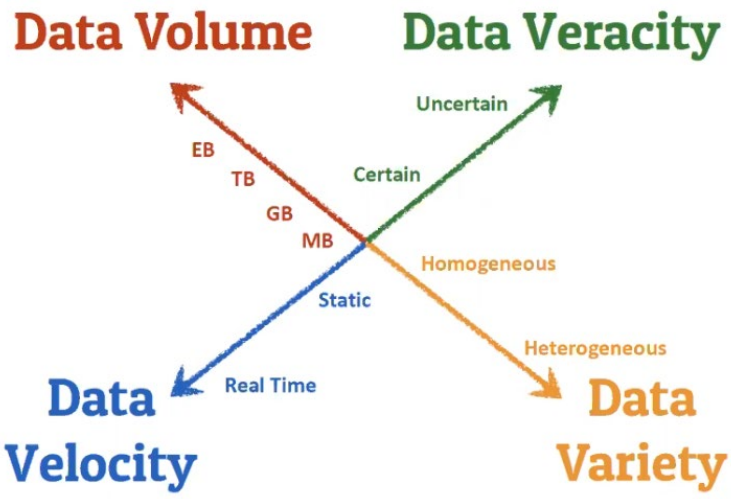
Biologische und Umweltforschungsthemen – Klimawandel bspw.

Business intelligence – Fähigkeit einer Organisation Daten zu organisieren und Pflegen.

Größte Datenquelle sind tatsächlich Handys. Bewegungsprofile, Fotos… sind extrem reich vorhanden. Auch Remote sensing – Vermessungsdaten sind sehr relevant. Kameras und Mikrofone im öffentlichen Raum – Ubahn usw.. RFID Sensoren und Netzwerke … Systemproduktionen mit CTs usw..



Klassifizieren von Big Data – Four Vs



CT bspw. ist bei GB-TB, iwo zwischen certain und uncertain (eher certain), eher homogen, aber wenn man Zusatzinfos (bspw. zusätzliche scans) added, dann eher heterogen,

außerdem zumeist statisch – zumindest dahingehend, dass man den Standardfall hat -> CT machen, Schlüsse ziehen. Bei Produktion hingegen wird in manchen Prozessen allerdings realtime inline oft gemacht.

Uncertainty

„What is not surrounded by uncertainty cannot be the truth." – Feynman

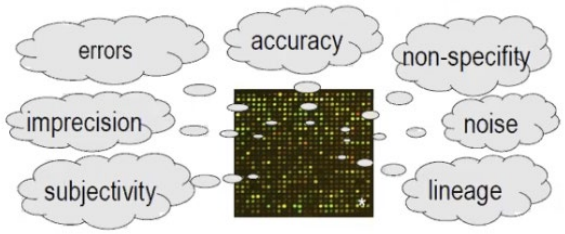
"True genius resides in the capacity for evaluation of uncertain, hazardous, and conflicting information." – Churchill

"Doubt is not a pleasant condition, but certainly is absurd" – Voltaire

Die Genauigkeit bezieht sich darauf, wie nah man an einer Ground truth ist, die **Präzession** hingegen ist wie genau ich beim nächsten Messen wieder denselben Wert bekomme.

Uncertainty an sich ist der Grad, zu welchem das mangelnde Wissen über das Ausmaß eines Fehlers für das Zögern verantwortlich ist, Ergebnisse zu akzeptieren. Genau das muss mit der Uncertainty-Bewertung gemacht werden.

Beispiel DNA microarray expression data



Es gibt typische Ausreißer (errors)
Dann gibt es die accuracy (Genauigkeit != precision)
subjectivity – Hat man nur einen gewissen Bereich eines Datensatzes, kann man in einem lokalen Optimum stecken.
non-specifity – false positives bspw. Die Auswirkung des Testresultats ist vl. nicht so schlimm im Großen und Ganzen. False negative (beides bei Corona) ist hingegen weit schlimmer.
Noise – Kann natürlich zur Unsicherheit beitragen, weil es so schwerer wird Dinge festzumachen.

Die uncertainties kann man sowohl im data oder view space behandeln.

Die Ziele zu Uncertainty sind

Die Evaluierung des **uncertainty budgets** = Woher kommt wie viel uncertainty?

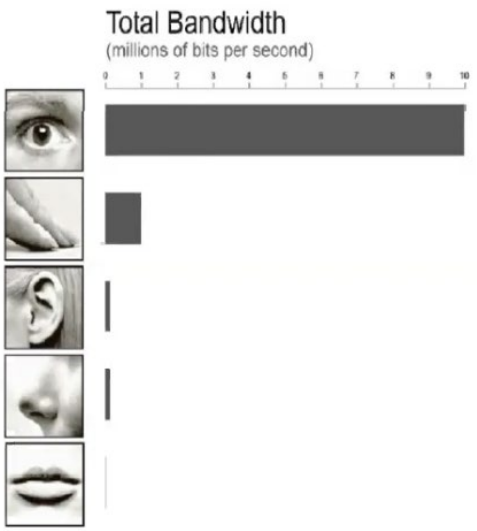
Evaluierung von **precision** und **accuracy** = Wiederholgenauigkeit und Genauigkeit in Hinblick auf die ground truth sind wesentlich.

Communication of uncertainty = eh klar…

10. Vis Abschluss

Grafiken

Man bedient sich zur Visualisierung/Kommunikation der Ergebnisse an Grafiken. Diese können Infos vereinfachen und verdichten. Mit weniger visual clutter.



Beispielgrafik zu den Sinnen des Menschen.

Figures are richer; provide more info with less clutter and in less space.

Figures provide 'Gestalt' effect: they give an overview; make structure more visible.

Figures are more accessible, easier to understand, faster to grasp, more comprehensible, more memorable, more fun, and less formal.

„Gestalt" – Psychologie ist eine Sparte der Psychologe, die sich mit menschlicher Wahrnehmung befasst. Mit Sinneswahrnehmungen, Eindrücken, usw..

6 bzw. 9 Gesetzte der Gestalt

Nähe – Elemente mit geringem Abstand erkennt man oft als zusammenhängend.

Ähnlichkeit – Sind zwei Objekte ähnlich zueinander, sieht man diese eher als zusammengehörend, als unähnliche.

Einfachheit/Prägnanz – Es werden bevorzug Gestalten wahrgenommen, die einer einfachen und einprägsamen Gestalt entsprechen.

Gute Fortsetzung – Denkt man an eine Linie, wird diese so weitergedacht, das die Linie den einfachsten Weg weiternimmt.

Geschlossenheit – Es werden eher geschlossene Strukturen wahrgenommen als offene.

Gemeinsames Schicksal – Wenn sich zwei oder mehr Elemente in eine Richtung bewegen, werden sie oft als eine Gestalt/Einheit wahrgenommen.

In den 90ern wurden die Gesetze noch erweitert

Gemeinsame Region – Elemente in abgegrenzten Gebieten werden als zusammengehörend empfunden

Gleichzeitigkeit – Werden Elemente gleichzeitig verändert oder verschwinden, nimmt man diese als zusammengehörig wahr.

Verbundene Elemente – Verbundene Elemente werden als ein Element wahrgenommen.

Grafiken können sehr viel mehr Zugang zu Daten generieren. Man kann die Daten schneller begreifen und erfassen/merken. Beispiel ist **Anscombe's Quartett**.

Als Zahlenfolge schwer zu erfassen.

Anscombe's quartet

I		II		III		IV	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

Man hat vier Datensätze und jeweils zwei Tupel, die zusammengehören, also insgesamt 11 Elemente in der Datenserie. Neben den mathematischen Statistiken, ist auch die grafische Darstellung der Daten wichtig.

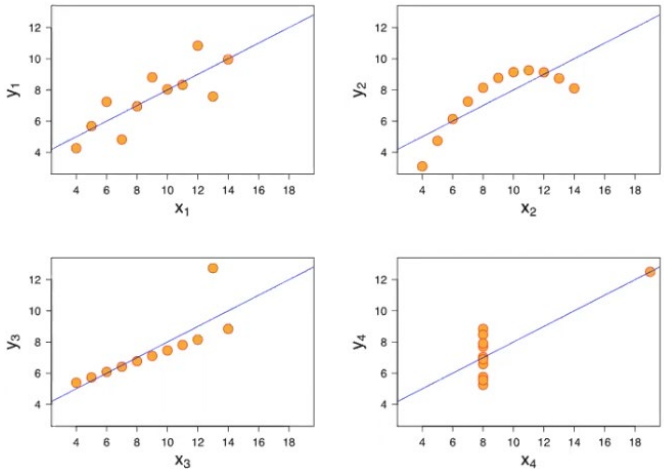
Statistics profile is the same for all!

Property	Value
Mean of *x* in each case	**9** (exact)
Variance of *x* in each case	**11** (exact)
Mean of *y* in each case	**7.50** (to 2 decimal places)
Variance of *y* in each case	**4.122 or 4.127** (to 3 decimal places)
Correlation between *x* and *y* in each case	**0.816** (to 3 decimal places)
Linear regression line in each case	$y = 3.00 + 0.500x$ (to 2 and 3 decimal places, respectively)

Die Statistiken sind für alle vier Datensätze gleich.

Korrelation ist hier, wenn der Wert gleich 1 ist, dass die zweite Variable 100% aus der ersten abgeleitet werden kann. -1 würde dasselbe bedeuten. 0 hingegen würde bedeuten, dass die Variable 2 keinen Zusammenhang mit Variable 1 hat.

Plottet man aber den Datensatz, sieht man gleich auf Anhieb, dass es sehr unterschiedliche Datensätze sind.



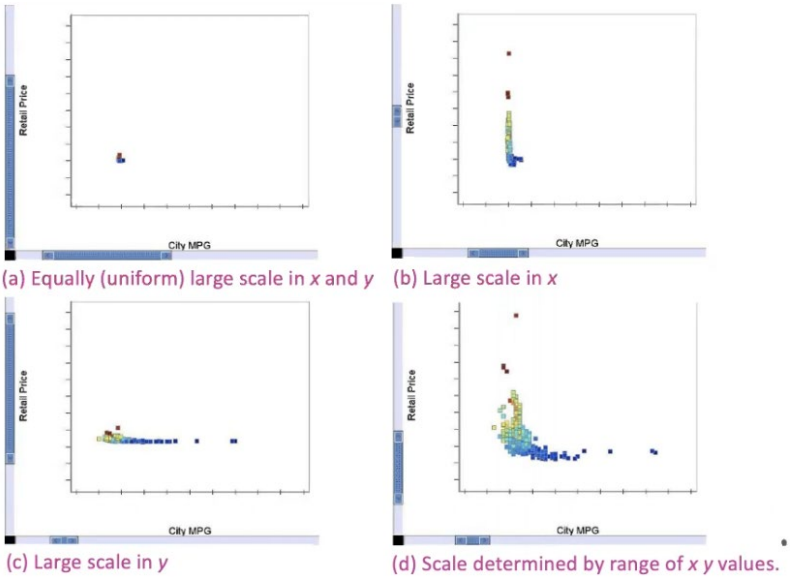
Four datasets with identical simple statistical properties

Yet they appear very different when graphed…

Biases in Visualisierung

Wenn man Daten plottet, muss man aufpassen, dass man keinen Bias reinbringt.

Dieselben Daten plotted mit verschiedenen scales kann sehr verschieden preceived werden.



(a) Equally (uniform) large scale in *x* and *y* (b) Large scale in *x*

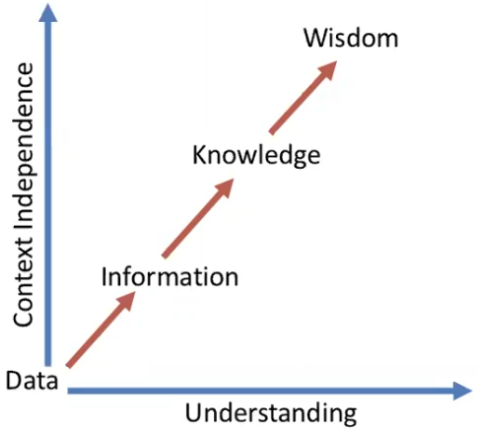
(c) Large scale in *y* (d) Scale determined by range of *x y* values.

Konstantin Lackner 19.01.2021

Diagramm vs Visualisierung

A diagram represents information.

A visualization represents data.



The DIKW-Hierarchy
according to [Bellinger 2004]

From Epistemology:
Data = ground truth
Information = phenomena
Knowledge = causes
Wisdom = possible (inter-)actions

"Above all else, show the data"
- Edward R. Tufte

Daten != Informationen. Diagramme zeigen Informationen, Vis Daten. Es gibt die DIKW. Daten sind sehr stark in den Kontext eingebunden. Für das Verständnis von Daten braucht man Informationen. Ein CT Datensatz ohne Info, was das ist und wie es angeordnet ist, kann man nicht benutzen. Man **braucht also Information um Daten einzuteilen**.

Weisheiten hingegen ist Wissen für die Diagnostik – Was bedeutet es für neue, zukünftige Patienten, wenn man dieses Bild sieht.

Mantras der Vis

Mantra = oft wiederholter Grundsatz (für die nicht-Eloquenten…)

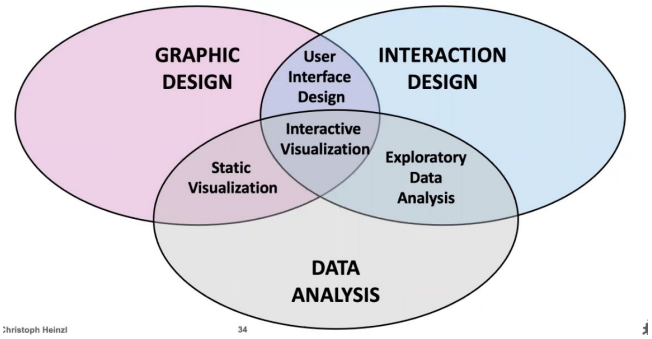
Overview first, zoom/filter, details on demand. – Schneiderman

Ist nicht immer gültig. Bei big data ist overview bspw. einfach nicht mehr so leicht möglich.

Erweitert – **Analyse first, show the important, zoom/filter, analyse further, details on demand** – Keim

Interaction

Interaktive Vis hat drei Bereiche.



Datenanalyse – Daten auf die man zugreifen will, die man mit interaktiven Möglichkeiten beeinflussen will.

Interaction design – Man braucht Interaktionstechniken.

Graphics design – Aufbereiten der Daten.

Die möglichen Zweierkombis sind jetzt auch jeweils Bereiche.

Statische Vis ist eine Vis, die nicht modifiziert werden kann, mit der nicht interagiert werden kann – bspw. ein bar chart in einer Zeitung.

Interaction design/Grafikdesign – Man hat keine Daten oder Datenanalyse, es geht ums UI Design. Da interessieren einen genauer die Zugänge „Was bewirkt meine Interaction und was kann ich damit darstellen".

Exploratory Dataanalysis – Man erforscht Daten ohne Vis.

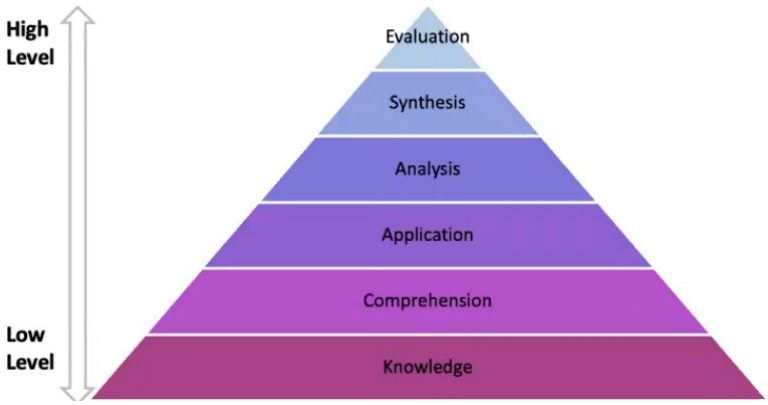
Cognition and Perception

Wichtiger Bereich bei Vis Analytics Konzepten.

Cognition beschreibt die mentalen Konzepte, die dabei helfen Wissen zu generieren, mentale Konzepte festzulegen usw.
Perception hingegen ist nur das Wahrnehmen.

Cognition: the **mental processes** which assist us to remember, think, know, judge, solve problems, etc.

Perception: the process by which we interpret the things around us **through sensory stimuli.**



Das ist **bloom's cognitive domains taxonomy** (= einheitliche Verfahren, mit dem Modell oder Objekte bestimmt werden können)

Die Anordnung ist also hierarchisch. Die Erkenntnispyramide beschreibt, dass, je höher die Einsicht in die Daten, desto höher der Erkenntnisgewinn.

Zunächst hat man **gelerntes Material**, das man abfragen kann.

Die zweite Stufe ist das **Verstehen** – Die Fähigkeit den Sinn eines Materials zu begreifen. Bspw. durch das Umsetzen von einer Form in die andere, bspw. Worte in Zahlen.

Drei ist die **Anwendung** – Die Fähigkeit das gelernte Material in neuen Situationen umsetzen zu können. Methoden, Konzepte und Prinzipien umsetzen zu können.

Vier ist die **Analyse** – Die Fähigkeit das gelernte Material in einzelne Teile zu zerbrechen. Was sind die einzelnen Teile unseres Problems. Dinge herunterbrechen/differenzieren/diskriminieren…

Fünf ist die **Synthese** – Die Fähigkeit Unterteilungen aus der Analyse zusammenzusetzen und neues Wissen zu generieren. Kategorisieren, kombinieren, kreieren und Erklären.

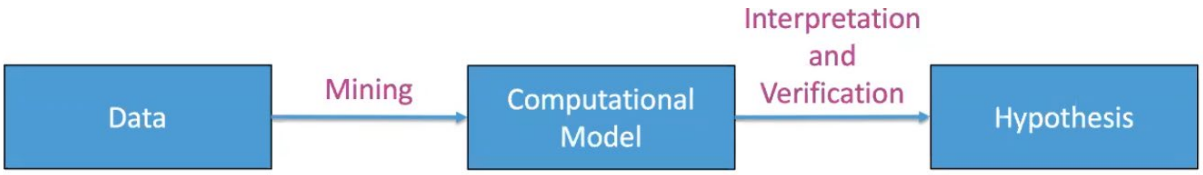
Letztlich – **Evaluation** – **Bewertung**. Beispielsweise kann der Wert eines Materials festgesetzt werden. Beschreiben, Erklären.

Eines der Probleme beim maschinellen Sehen ist, dass man als Mensch Erfahrung im Bereich der visuellen Wahrnehmung hat. Man hat ein Konzept für Dinge. Bei Computern liegt das eigl. nur an der Sensorik, die man zu Verfügung hat.

Data Analysis

Hauptsächlich data mining, machine learning und statistics.

Data mining – Man holt aus dem Datensatz neue Erkenntnisse um neue Daten zu generieren. Es ist eine automatische, algorithmische Extraktion von Information aus Rohdaten.



Data mining tasks können in **predictive tasks** und **descriptive tasks** unterteilt werden.

predictive ist Klassifikation und Regression.

descriptive ist Rule discovery, clustering, pattern mining.

Generell sind **predictive tasks** solche, wo ein globales Modell erstellt wird. Bspw. die Werte eines Targets vorherzusagen und einzelne Objekte zuzuordnen.

Descriptive tasks erstellen Zusammenhänge, man will Charakteristika und Muster erkennen.

Bei der **deskriptiven Statistik** geht es darum, dass man **summary statistics** macht oder **Massenstatistik** - Beispiel: Man evaluiert den IQ von Schülern, die einen neue Lernmethode versuchen. Dort stellt man fest, dass der IQ um 0.1 pro Schüler steigt, wenn man diese Methode anwendet. Das ist kein großer Gewinn für einen Schüler, aber in der Gesamtheit eine sehr wesentliche Änderung. Für so etwas braucht man knowledge discovery oder data mining.

Leider sind das meisten Blackboxen. Man hat voll oder halbautomatische Analyse von Datensätzen. Das ist leider auch ein Problem. Wenn man eine Blackbox analysiert und man intern nicht feststellen kann, wie die Blackbox funktioniert, kann man diese meistens auch nicht zertifizieren. Man kann nämlich damit dann die Genauigkeit des Systems nicht festlegen.

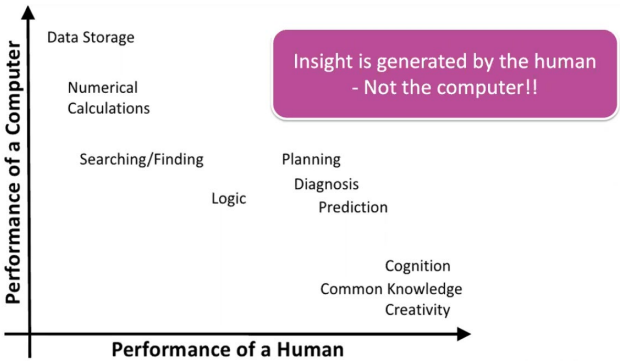
Black-box methods in the hands of end users

- Users need to understand the algorithms for using them
- What attributes to use? What similarity measure? etc.
- Often trial and error

Vis kann helfen einen Einblick in diese Blackboxen zu bekommen. Man kann bspw. Neuronen in Blackboxen darstellen und sich anschauen, wie diese schalten.

Ability Matrix

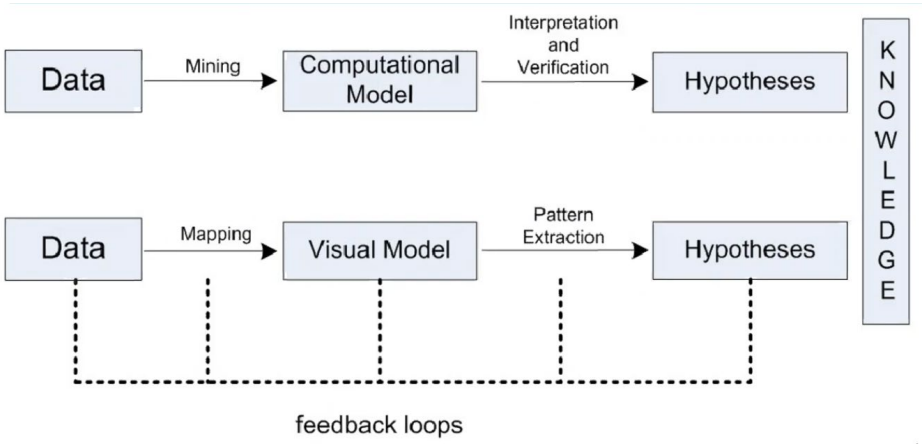
Leistung eines Computers im Vergleich zur Leistung des Menschen.



Hier sieht man also, dass Menschen sehr gut in **Kreativität** und **Verständnis** sind. Computer aber gar nicht. Hence – **Insight is generated by the human, not the computer**.

Der Computer kann die **Daten aufbereiten** und **zur Verfügung stellen**, aber versteht nicht.

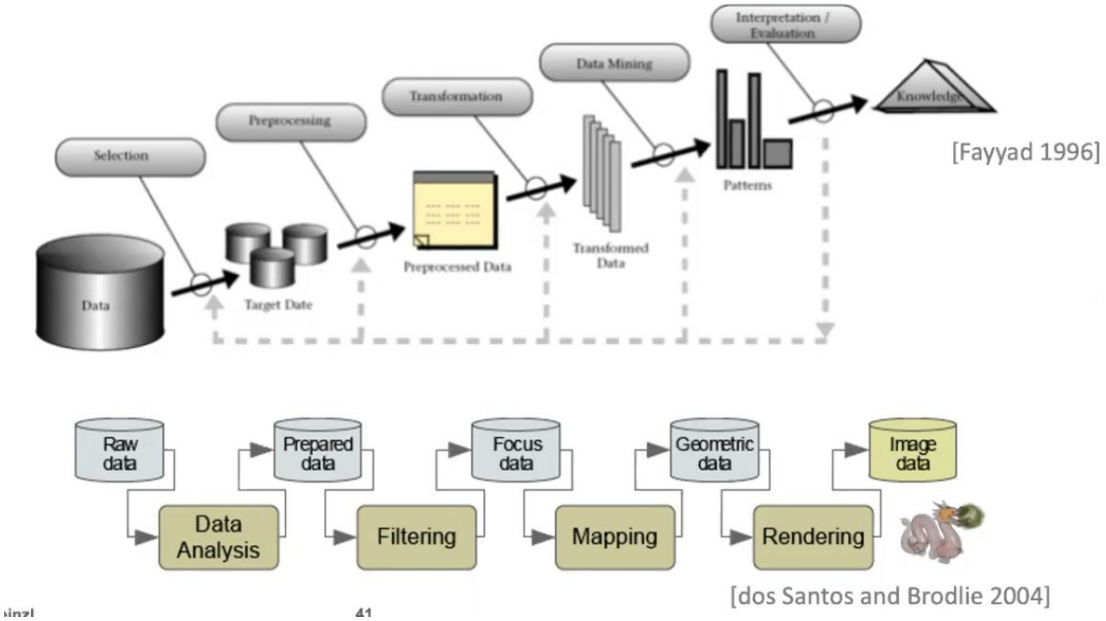
Data mining vs. visual analysis



Data mining **geht von einem computational model aus**, das man vom Datensatz abgeleitet hat.

Die vis analysis hingegen leitet keine Modelle und Zusammenhänge direkt aus den Daten ab. **Man mappt die Daten und macht ein visuelles Modell**. In diesem visuellen Modell können dann patterns abgeleitet werden und analysiert.

knowledge discovering data basis Pipeline vs vis pipeline



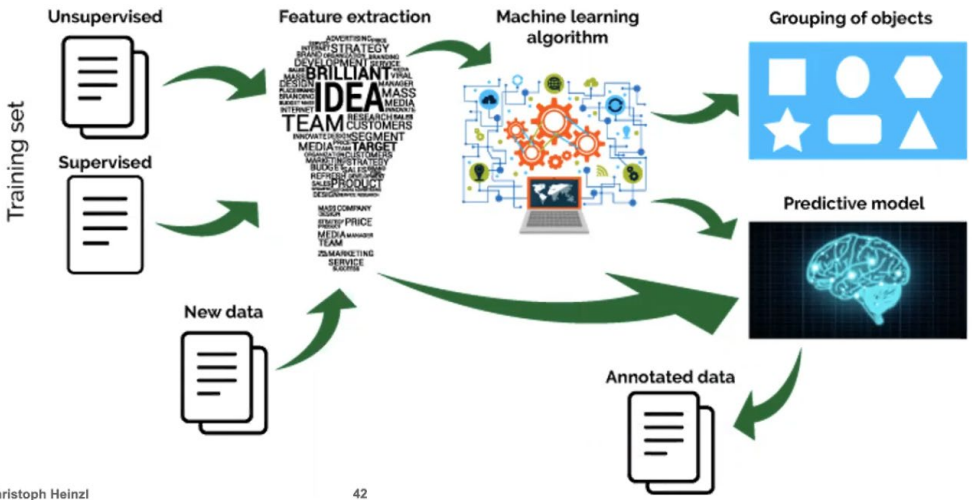
Bei KDD(oben) startet man mit Daten, selektiert aus denen einige, muss die vorverarbeiten, damit das data mining dann ausgeführt werden kann, dann wird data mining gemacht, dann werden die gefundenen Patterns evaluiert.

Vis startet bei Rohdaten, macht dann eine Analyse, hat dann vorbereitete Daten (bspw. 2D Bilder in 3D Volumen umrechnen) – dann Bereich herausfiltern (bspw. Tumor). Dann mappt man die Geometrien, damit man die geometrischen daten dann rendern kann. Dann bekommt man ein Bild, aus dem weiteres Wissen generiert werden kann.

Machine learning

Man will natürliche Muster in Daten finden. Diese helfen Einblicke in Daten zu generieren.

Man verwendet es in der medizinischen Diagnostik, im Börsenhandel, bei Energielastzuständen… Auch für Suchvorschläge…



Es gibt dabei zwei Techniken des machine learning. **Supervised** bzw. **unsupervised learning**.

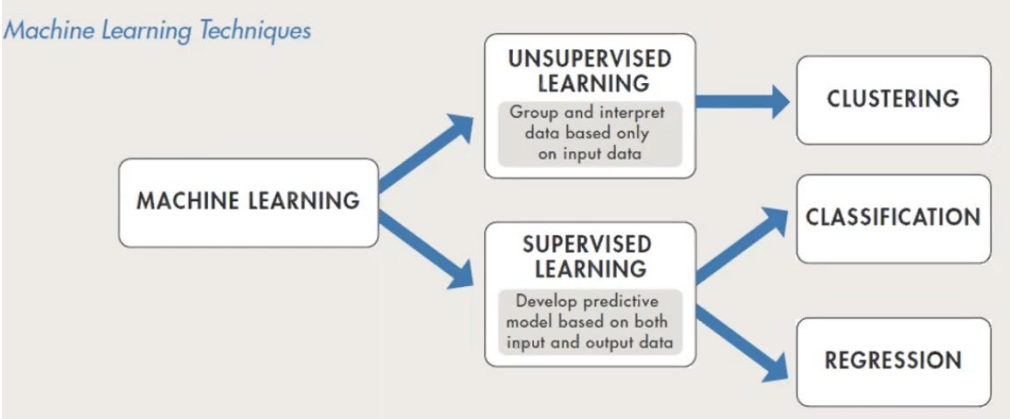
Supervised ist dabei das sowohl Ein- als auch Ausgabe im Training bekannt sind. Man hat also einen Lehrer, der einem sagt „Für diese Eingabe habe ich folgende Ausgabe“.

Unsupervised ist hingegen kennt man die Ausgangsdaten nicht. Man kann damit Ausreißeranalysen und Korrelationsanalysen machen, aber man kann nicht wie im supervised learning auf eine Ausgabe schließen, diese hat man ja nicht verfügbar.

Anschließend gibt es die feature extraction (**also nach dem training**) – man wirft neue Daten rein und holt zusätzliche Features raus, die relevant sind. Dann wirft man all das in einen Machine learning algorithmus – dort drinnen sind Faltungsoperatoren mit Gewichten. Die Gewichte entscheiden, wie stark ein Neuron auf etwas einwirkt und ob es schaltet oder nicht. Sobald man das trainiert hat, kann man eben Gruppierungen zusammenstellen – was gehört zusammen, oder auch predictive models machen, aus welchen wiederrum annotierte Daten gemacht werden können, mit welchen man neue Daten (nicht die Trainingsdaten) neu bewerten kann.

Das Training eines solchen Algorithmus ist sehr aufwändig. Für kleine Trainingssets kann das schon locker 40-50 Stunden dauern… Wenn man aber das Modell schon hat und nur noch die Gewichte anwendet, ist die Klassifizierung sehr schnell. Bei einem kleinen Set 20-30min.

Zwei Arten des Machine Learnings erneut



Unsupervised ist dann hilfreich, wenn man mehr über die Daten wissen will, aber noch kein spezifisches Ziel hat – bspw. eine Ableitung eines Segmentierungsmodells.

Beim **Supervised learning** hat man immer das Ziel ein Modell abzuleiten, das Hilft Vorhersagen über Daten zu erzeugen. Bspw. eine diskrete Antwort - Klassifikation von Daten – ist ein Tumor gefährlich? Oder kontinuierliche Antworten, also Regression – Bspw. Electricityloadbewertungen – ist es noch im Rahmen, oder schon kritisch?

Supervised leraning wird auch benutzt um bspw. Herzattacken vorherzusagen. Man weiß welche Daten zu einem Herzinfarkt geführt haben und kann anhand dieser Daten ablesen, ob man es mit einem hohen Risiko zutun hat.

Algorithmen für supervised oder unsupervised learning

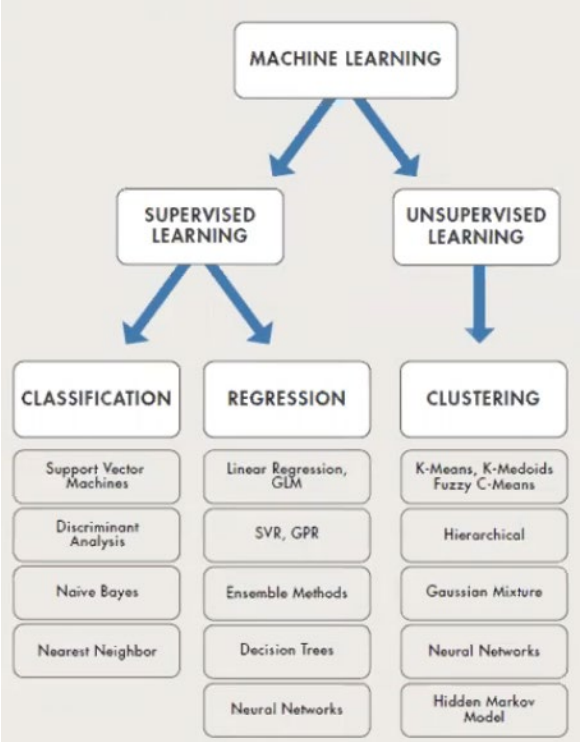
Es gibt sehr viele Methoden und keine beste.

Oft ist es so, dass ein Gutteil der Auswahl bzgl. Algorithmen Trial and Error ist. Man untersucht, welche Algorithmen sich für einen Zweck anbieten und muss diese dann auch noch fine-tunen.

Machine Learning wird benutzt, wenn man ein sehr komplexes Problem hat, oder sehr viele Daten (oder beides) und auch wenn es keine Gleichungen oder Formeln gibt um ein System

zu beschreiben (Gesichtserkennung, oder Texterkennung, Spracherkennung…) Es gibt eben kein festes Modell.

Man verwendet es auch, wenn sich die Gesetzmäßigkeiten eines Tasks täglich ändern. Egal ob menschengemacht, oder durch die Natur. Wetter forecasts oder trend forecasts.

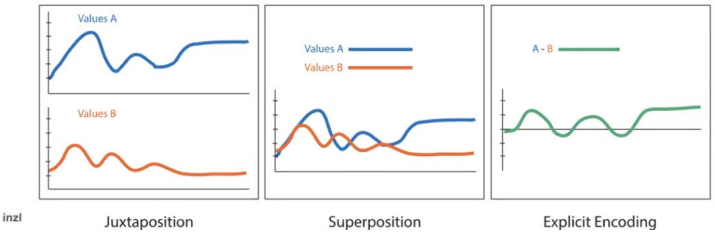


Comperative Vis

Wenn man sehr viele Daten zu vergleichen hat, ist dieser Ansatz wichtig.

es gibt drei Bereiche:

Juxtaposition (“side-by-side placement”)
Superposition (“stacked placement”)
Explicit Encoding (marking, computations etc.)



inzl

Explicit encoding ist bspw. eine **Differenzberechnung**.

Explicit encoding kann sinnvoll bei vielen Werten sein, weil man eben nicht viele Linien bekommt, sondern Zusammenhänge, Extrema, usw.. Man kann durch functional boxplots tausende Linien gleichzeitig vergleichen. Nachteil einer solchen Darstellung ist, dass man halt auch Information verliert. Man bekommt ja bspw. keine Extrema der einzelnen Funktionen. Gerade bei vielen Werten nicht mehr.

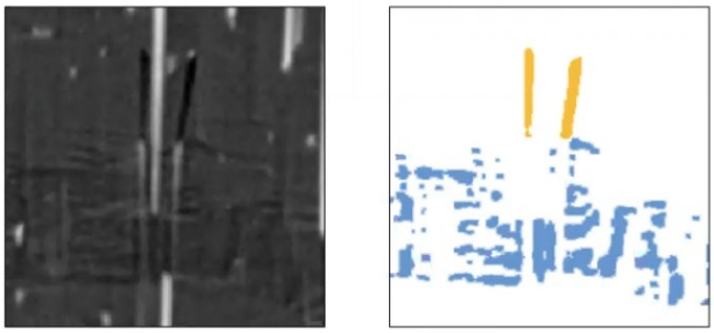
Bei Juxtaposition hat man mit vielen Werten das Problem von visual clutter.

Superposition ebenso.

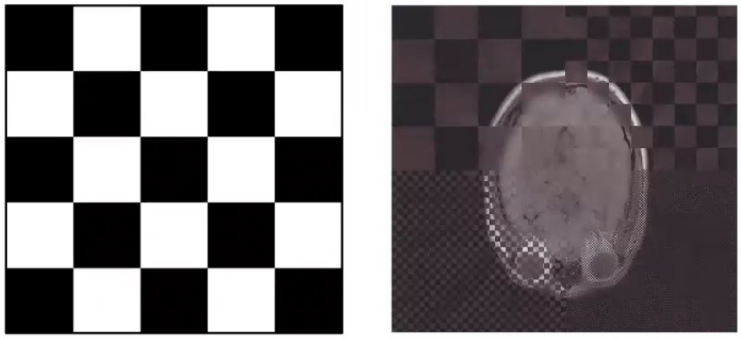
Konstantin Lackner 19.01.2021

Beispiele (juxta, superposition, explicit encoding)

Juxta

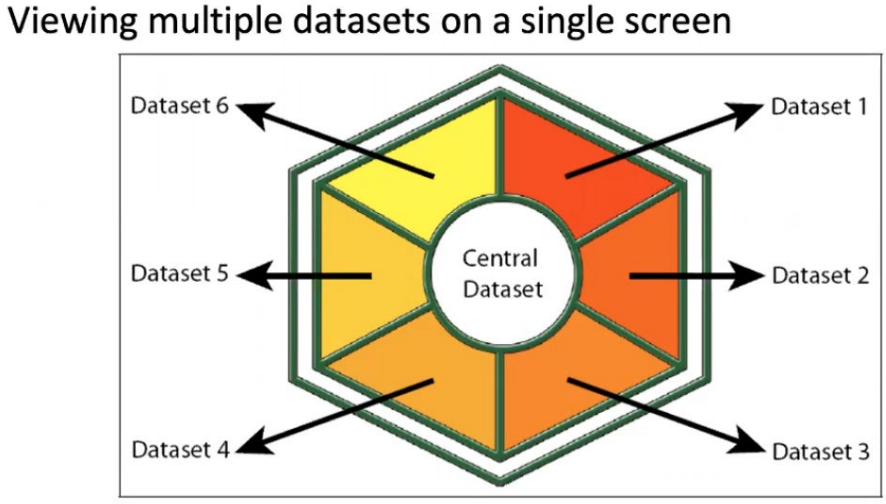


Fasern in einer Matrix. Side by side view. Man sieht viele kleine cracks und auch richtige Risse. Man kann sich also hier die Defekte neben dem Originalbild anschauen.

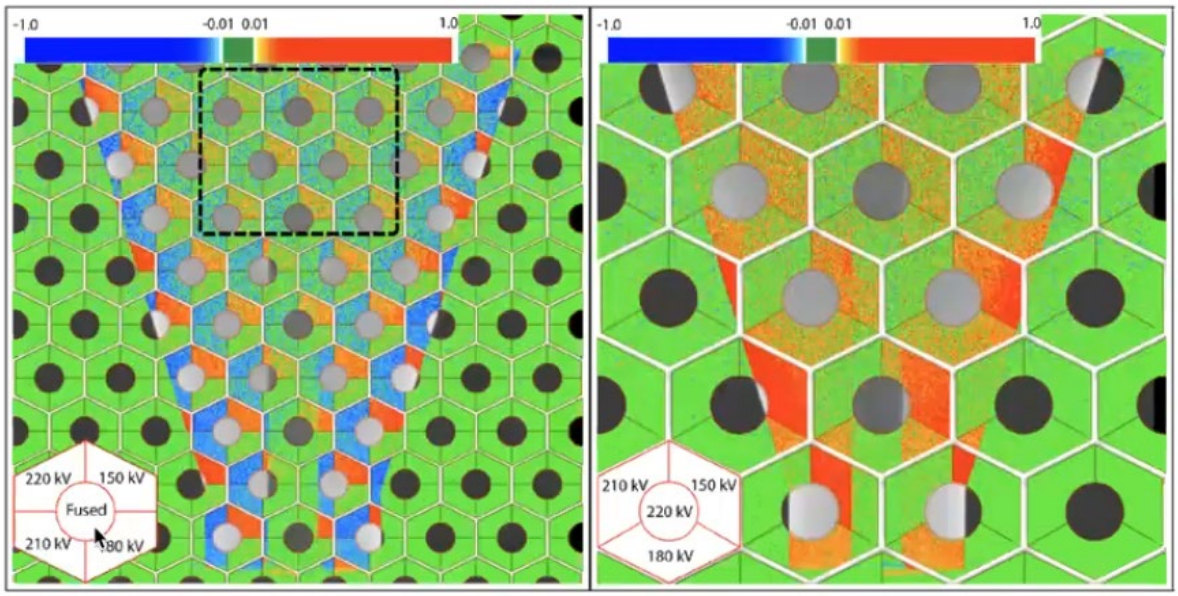


Hier plottet man zwei Datensätze übereinander. Es ist eigl. schon juxta position, weil man eben in einem checkerboardmuster Die einzelnen Teile nebeneinander legt. Konkret hat man hier CT und MRT. MRT ist sehr schlecht aufgelöst, CT sehr gut zusammen sieht man hier dann Unterschiedliche Informationen beide gut dargestellt.

Würde man **mehr als zwei Datensätze** miteinander vergleichen, bräuchte man bspw. statt einem **Quadrat ein Hexagon**.



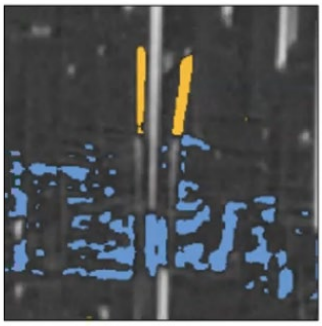
Iwann ist da aber Schluss… 10-12 Datensätze erkennt man einfach nicht mehr. Die tilesize kann da noch zu groß/klein sein.



Der Datensatz in der Mitte ist fusioniert. Es ist also eine Kombi zwischen explicit encoding und Rest.

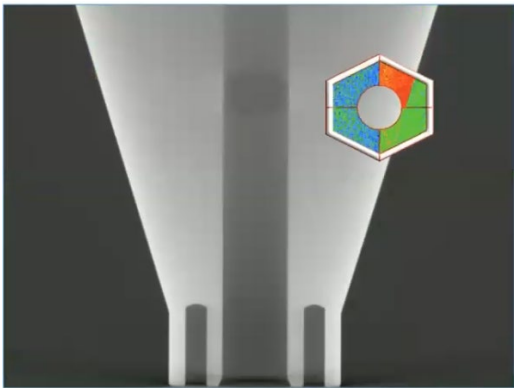
Superposition

Stacked placements.



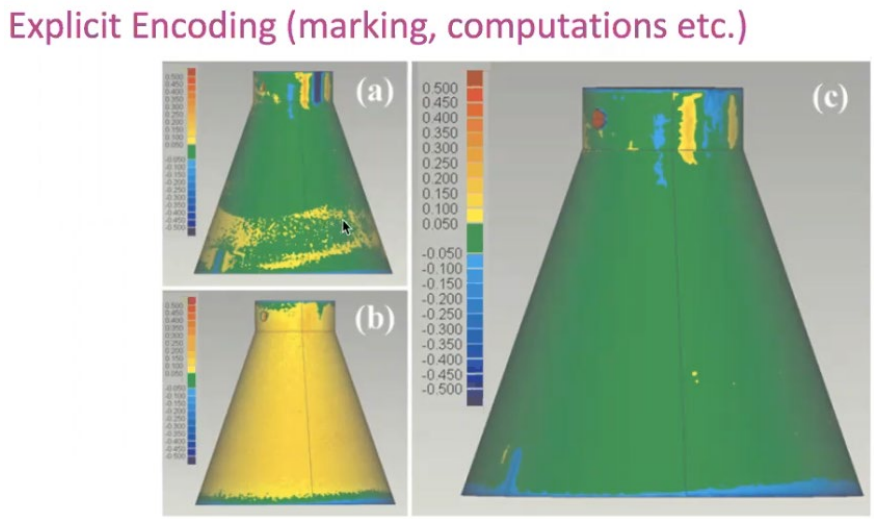
Ist vor allem für wenige Datensätze gut geeignet.

Superposition Hexagon.



Explicit Encoding

Oft um Unterschiede in Geometrien zu zeigen. Man sieht schnell wo der Datensatz mit einem Referenzdatensatz übereinstimmt. Man schaut also wo der Unterschied groß ist (Gelb)

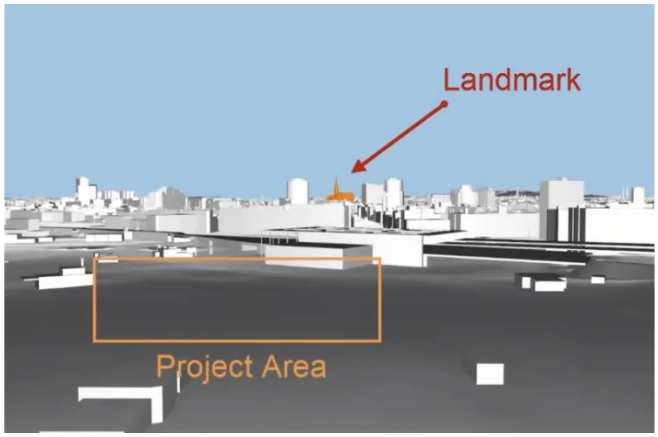


Beispiel Vis Analytics tool – Vis-A-Ware

Man will eine räumliche Planung für neue Gebäude in der Städteplanung. Wenn man ein neues Gebäude plant, wird normalerweise ein Architektenwettbewerb ausgeschrieben und man bekommt 20-30 Kandidaten. Man will also analysieren, welchen Eindruck ein Gebäude auf das Stadtbild macht, welche Landmasse man braucht, usw..

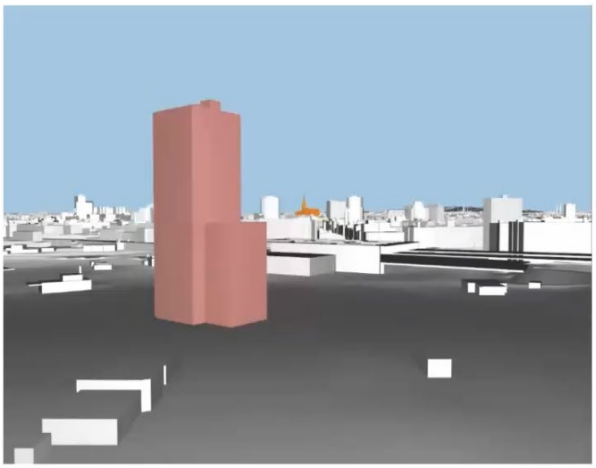
Benutzt sind dabei:

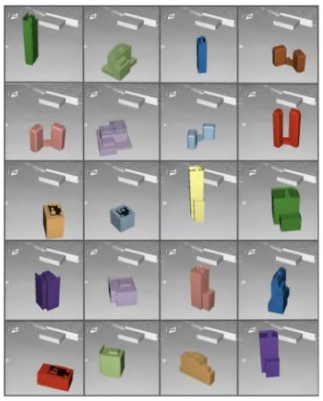
- Visual Analytics
- Comparative Vis.
- Quantitative Vis.
- Scalable Vis.
- Linked/Integrated Views



Hier stellt sich die Frage, ob der Blick auf den Stephansdom verstellt wird, oder ob man von dem Gebäude aus den Stephansdom sieht.

Konstantin Lackner 19.01.2021





20+ candidate buildings

Ohne so ein Tool hat man nur statische Vis von den Gebäuden und müsste selbst überlegen, ob man noch sieht und so.

VIS Aware Ziele

Quantifizieren des visuellen Impacts.

Vergleichen von Gebäuden.

Integrated view on quantitative and qualitative data and spatial component.

Das Tool hat einen 3D View, über die Map Overlays mit entsprechenden viewpoints.

Man hat auch einen Filmstrip view, wo man sich die einzelnen Kandidaten im Vergleich anschauen kann. Man hat auch eine transposable ranking view, also eine Art Tabelle mit Werten wie bspw. „candidate visibility“ …