Spatial Layout

Maneesh Agrawala

CS 294-10: Visualization
Fall 2007

Assignment 3: Visualization Software

Create an interactive visualization application – you choose data domain and visualization technique.

1. Describe data and storyboard interface
due Oct 3 (before class)
2. Implement interface and produce final writeup
due Oct 15 (before class)
3. Submit the application and a final writeup on the wiki

Can work alone or in pairs
Final write up due before class on Oct 15, 2007

Final project

Design new visualization method

- Pose problem, Implement creative solution

Deliverables

- Implementation of solution
- 8-12 page paper in format of conference paper submission
- 2 design discussion presentations

Schedule

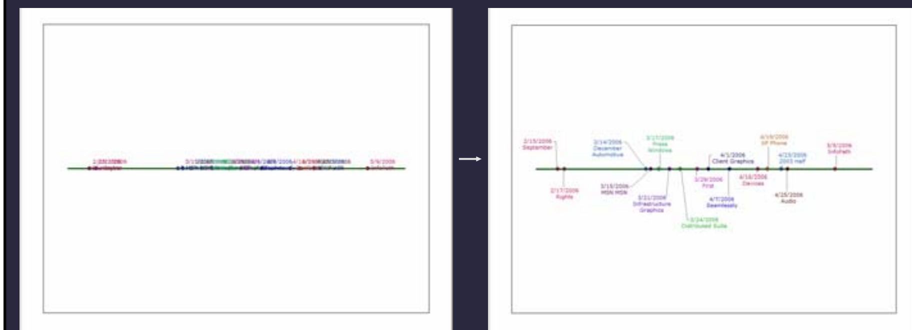
- Project proposal: 10/24
- Initial problem presentation: 10/24, 10/29 or 10/31
- Midpoint design discussion: 11/19, 11/21 or 11/26
- Final paper and presentation: To be determined

Grading

- Groups of up to 3 people, graded individually
- Clearly report responsibilities of each member

Spatial Layout

Example: Timeline label layout

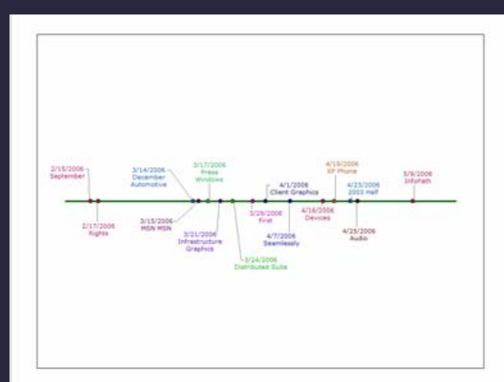


Problem

Input: Set of graphic elements (scene description)

Goal: Select visual attributes for elements

- **Position**
- Orientation
- Size
- Color
- …

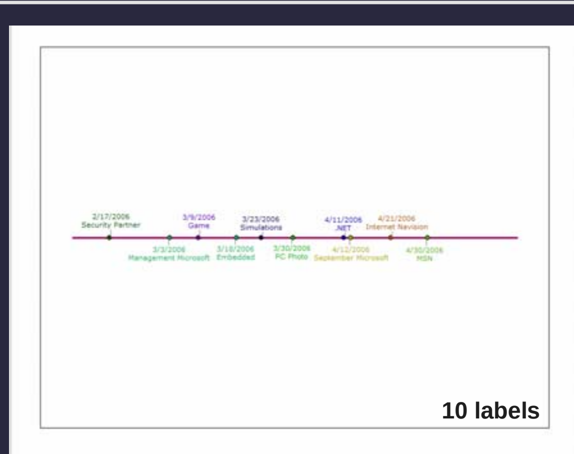


Topics

Direct rule-based methods
Constraint satisfaction
Optimization
Example-based methods

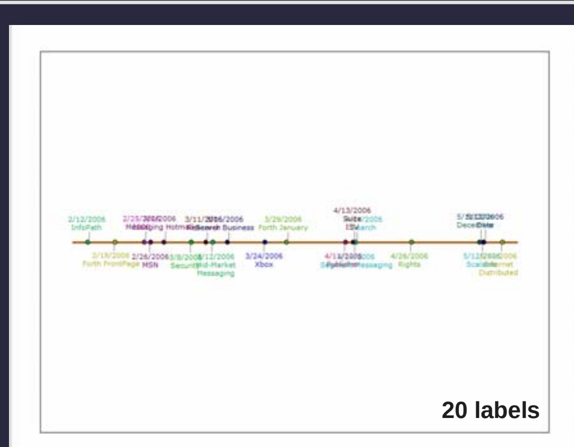
Direct Rule-Based Methods

Rule-based timeline labeling



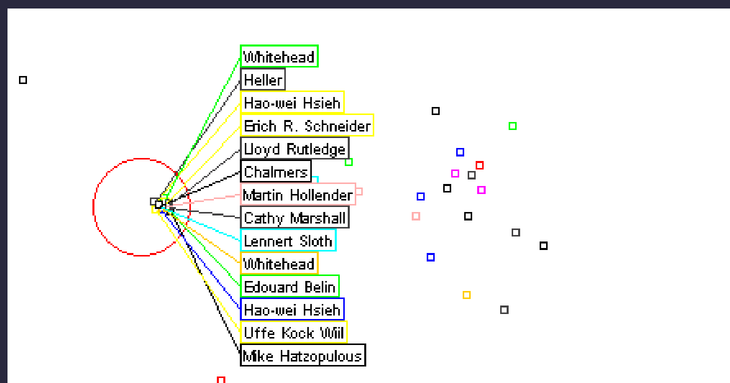
- Alternate above/below line
- Center labels with respect to point on line

Rule-based timeline labeling



- Alternate above/below line
- Center labels with respect to point on line

Excentric labeling [Fekete & Plaisant 99]



http://www.cs.umd.edu/hcil/excentric/

Dynamic space management [Bell 00]

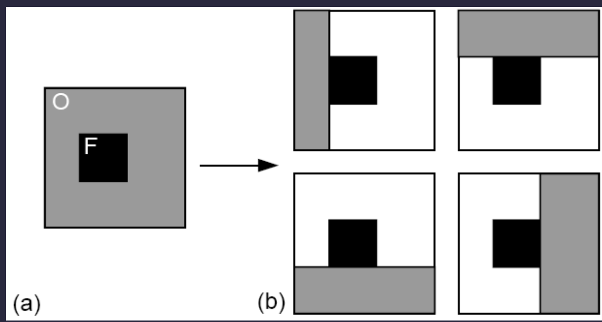
Manage *free space* on desktop to prevent window overlap

Video (0:46s)

Dynamic space management [Bell 00]

Goal: Place new elements to avoid overlap

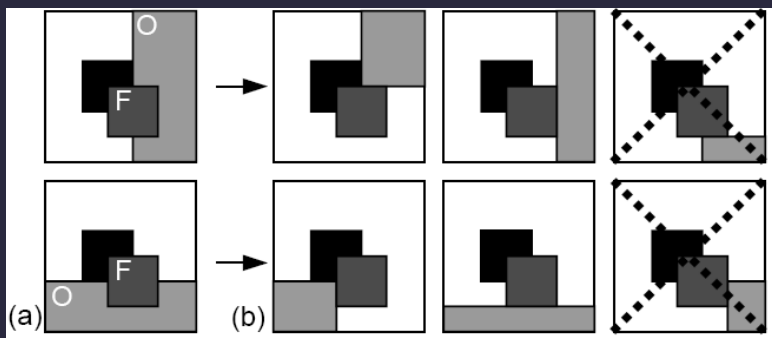
- Elements are axis-aligned rectangles
- Keep track of largest empty space rectangles



Dynamic space management [Bell 00]

Goal: Place new elements to avoid overlap

- Elements are axis-aligned rectangles
- Keep track of largest empty space rectangles



Pros and cons

Pros

- Designed to run extremely quickly
- Simple layout algorithms are easy to code

Cons

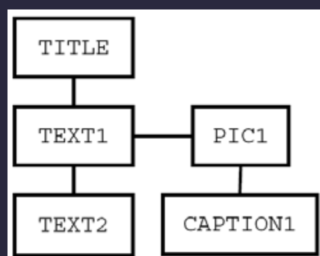
- Complex layouts require large rule bases with lots of special cases

Linear Constraint Satisfaction

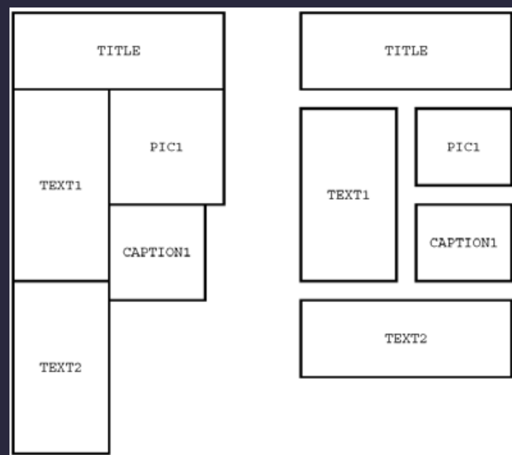
Network of layout constraints

TITLE ABOVE TEXT1
TITLE FULL PAGE WIDTH
TEXT1 LEFT OF PIC1
CAPTION1 BELOW PIC1
TEXT2 BELOW TEXT1

Constraints



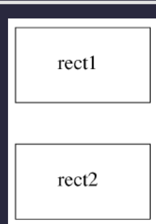
Network



Two possible layouts

[from Lok and Feiner 01]

Constraints as linear equations



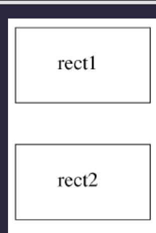
C1: rect2.top = rect1.top + rect1.height + 10
C2: rect2.height = rect1.height
C3: rect2.bottom = rect2.top + rect2.height

Local propagation
- Set any variable
- Update other variables to maintain constraints

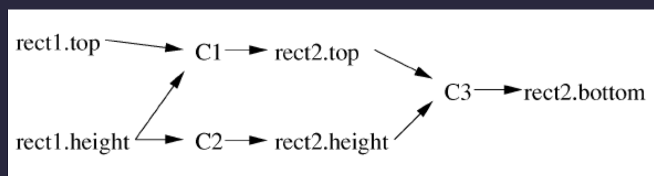
One-way
- Each constraint has 1 output variable
- Update output when any input changes

Multi-way
- Each constraint can be written so that any variable is output
- More complicated to maintain

One-way constraints



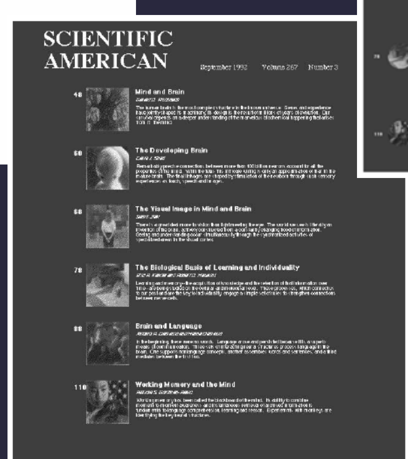
C1: rect2.top = rect1.top + rect1.height + 10
C2: rect2.height = rect1.height
C3: rect2.bottom = rect2.top + rect2.height



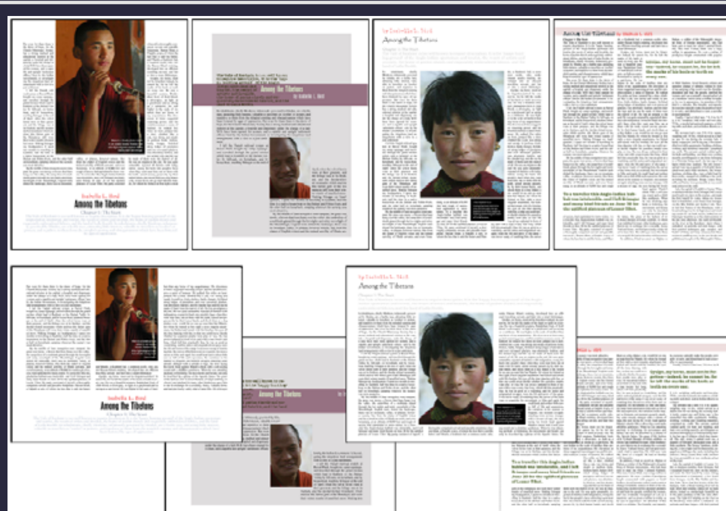
One-way constraints form a directed acyclic graph (DAG). Given the value for any variable we propagate it's value locally through the graph updating the other variable.

Page layout example [Weitzman and Wittenburg 94]

```
(Defrule (Make-Article The-Grammar)
  Article -> Text Text Text Number Image
    0        1    2    3     4      5
  (Author-Of 2 1)
  (Description-Of 4 1)
  (Page-Of 4 1)
  (Image-Of 5 1)
  (article-name 0) = r
  (article-image 0) = 5
  :OUT
  (right-of 1 5)
  (top-aligned 1 5)
  (top-aligned 5 4)
  (spaced-below 2 1)
  (spaced-below 3 2)
)
```

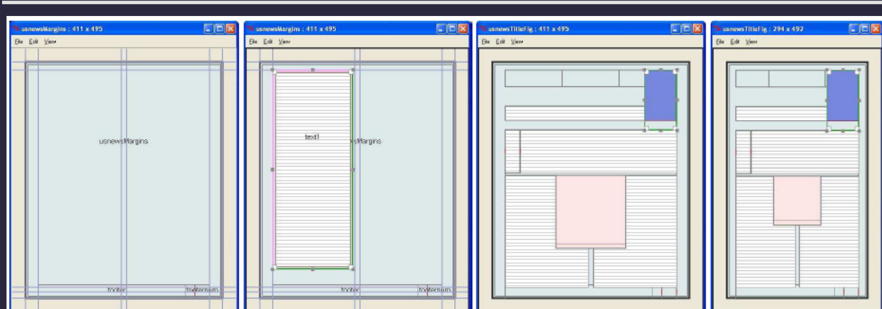


Adaptive document layout [Jacobs 03]



Users authors templates which use one-way constraints to adapt to changes in page size

ADL template authoring [Jacobs 03]



Video

Pros and cons

Pros

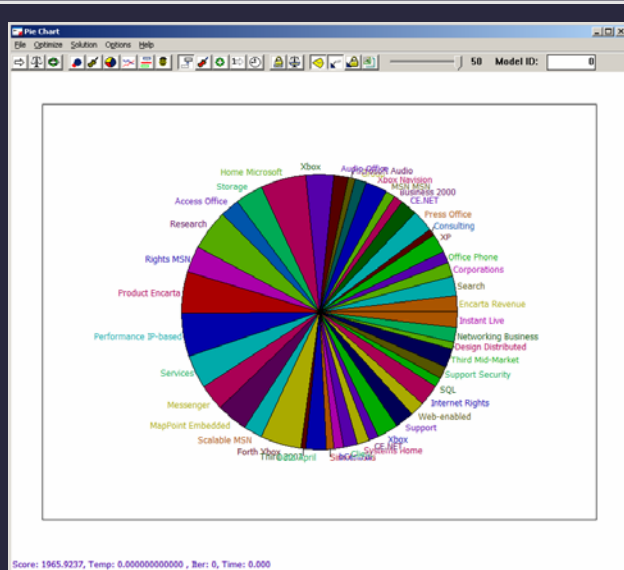
- Often run fast (at least one-way constraints)
- Constraint solving systems are available online
- Can be easier to specify relative layout constraints than to code direct layout algorithm

Cons

- Easy to over-constrain the problem
- Constraint solving systems can only solve some types of layout problems
- Difficult to encode desired layout in terms of mathematical constraints

Optimization

Demo



Layout as optimization

Scene description

- **Geometry:** polygons, bounding boxes, lines, points, etc.
- **Layout parameters:** position, orientation, scale, color, etc.

Large design space of possible layouts

To use optimization we will specify …

- **Initialize/Perturb functions:** Form a layout
- **Penalty function:** Evaluate quality of layout
- **.. and find layout that minimizes penalty**

Optimization algorithms

There are lots of them:

line search, Newton's method, A*, tabu, gradient descent, conjugate gradient, linear programming, quadratic programming, simulated annealing, …

Differences

- Speed
- Memory
- Properties of the solution
- Requirements

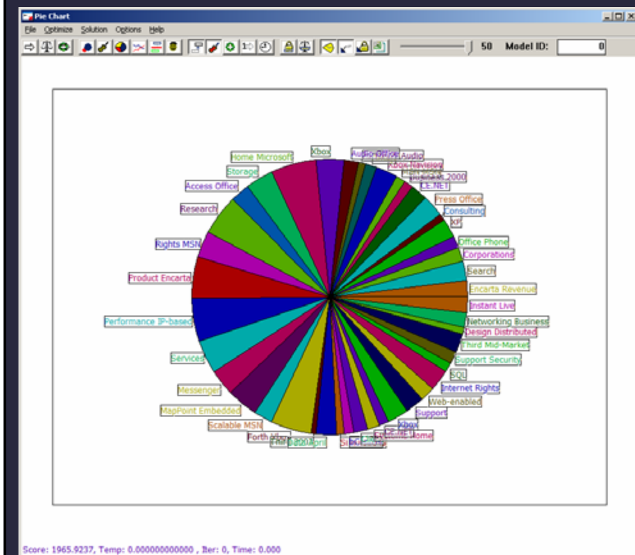
Simulated annealing

```
currL ← Initialize()                          → Form initial layout
while(! termination condition)
  newL ← Perturb(currL)                       → Perturb to form new layout
  currE ← Penalty(currL)                      → Evaluate quality of layouts
  newE ← Penalty(newL)
  if((newE < currE) or                        → Always accept lower penalty
     (rand[0,1) < e^(-ΔE/T)))                 → Small probability of accepting higher penalty
     then currL ← newL
  Decrease(T)
```

Perturb: Efficiently cover layout design space

Penalty: Describes desirable/undesirable layout features

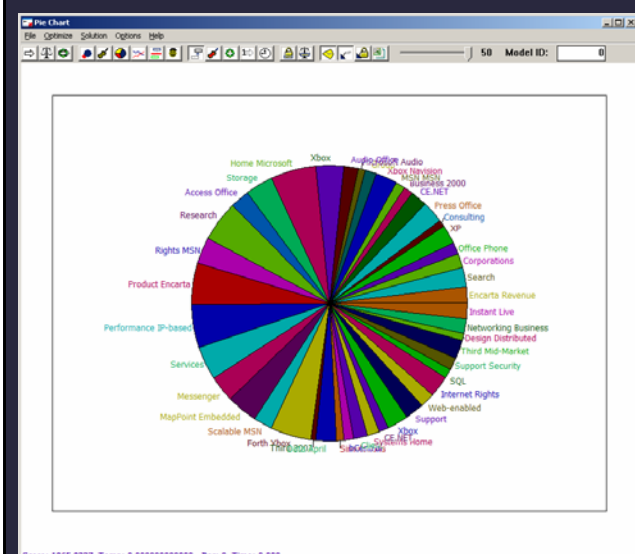
Scene description



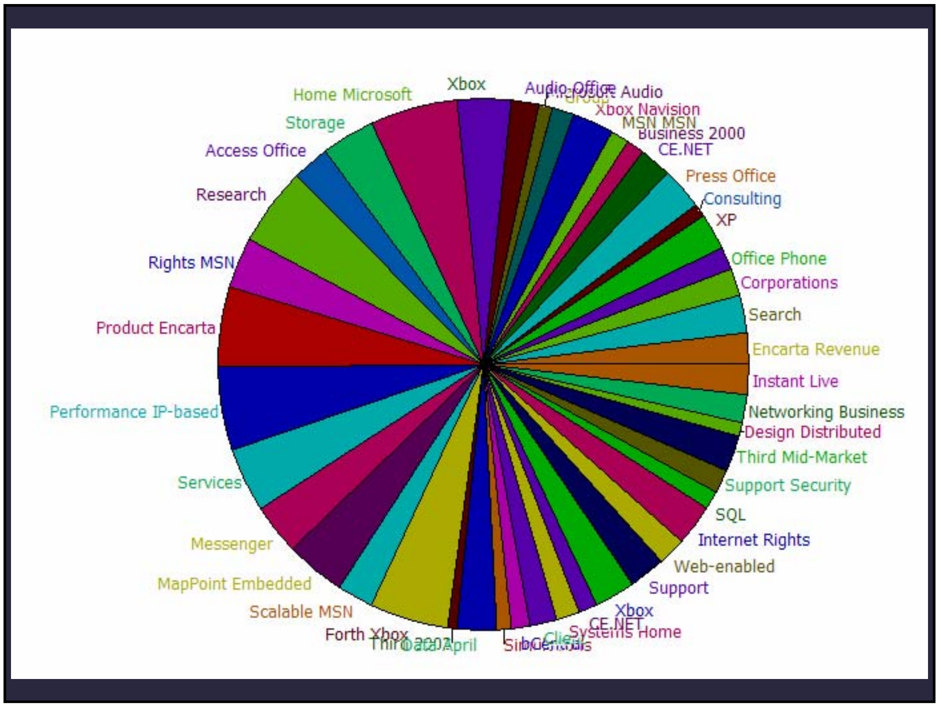
Geometry

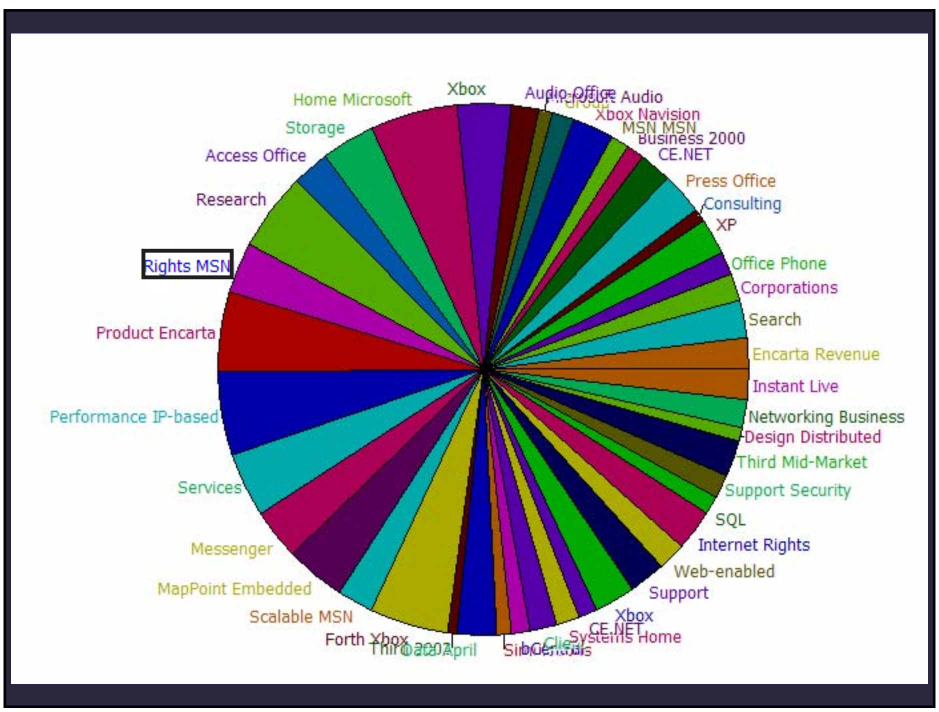
- **Pie slices**
  anchors for labels
- **Labels**
  bounding boxes

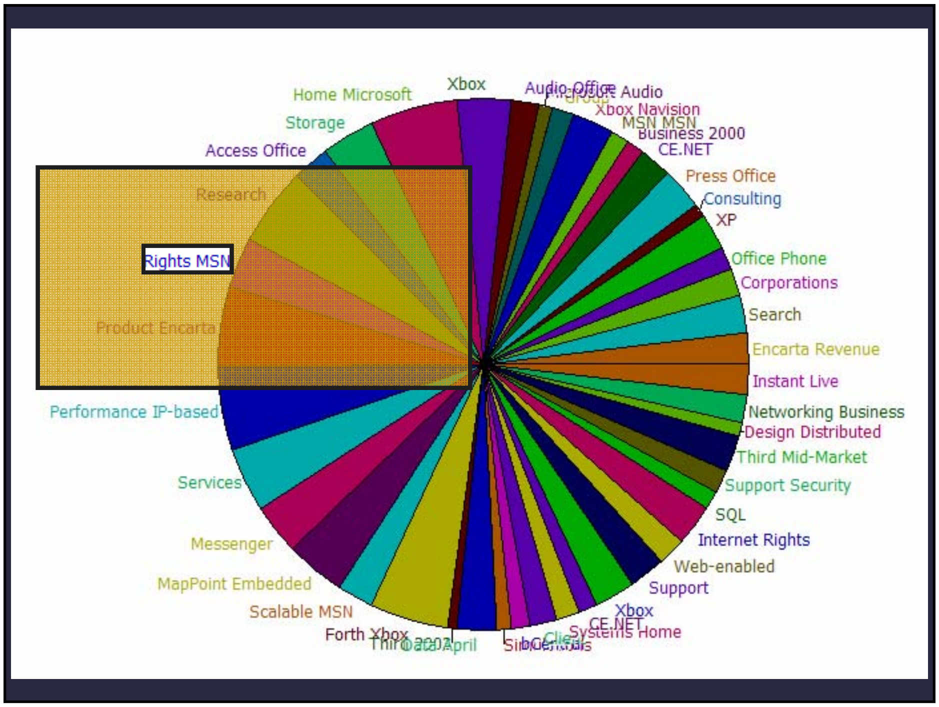
Layout parameters

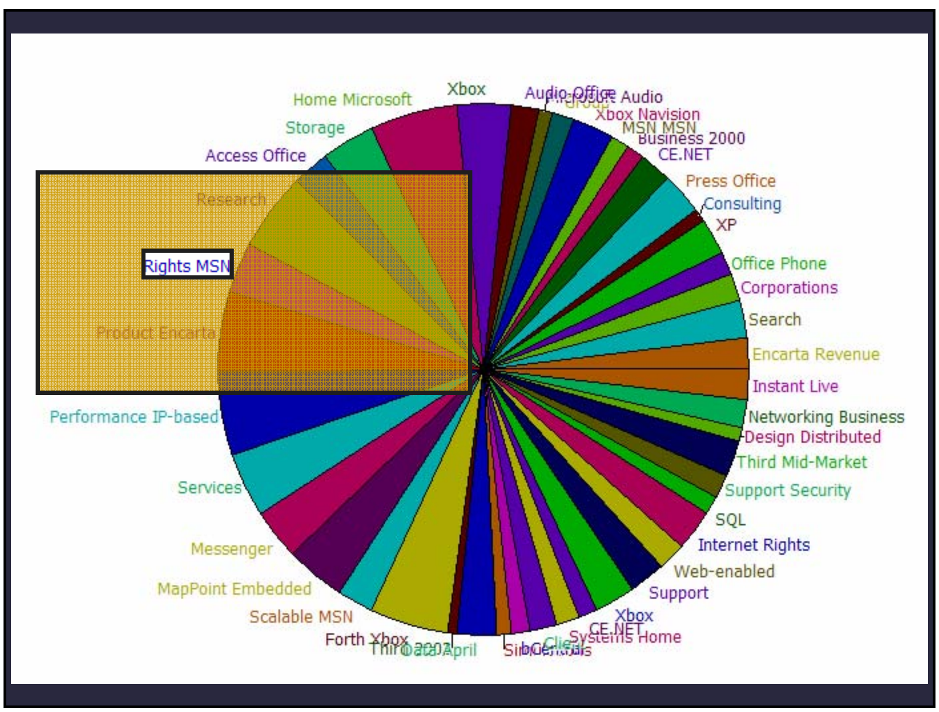


- **Position *(x, y)***
- **Leader line**
- **Word wrap**
- Color
- Alignment
- Orientation
- Scale

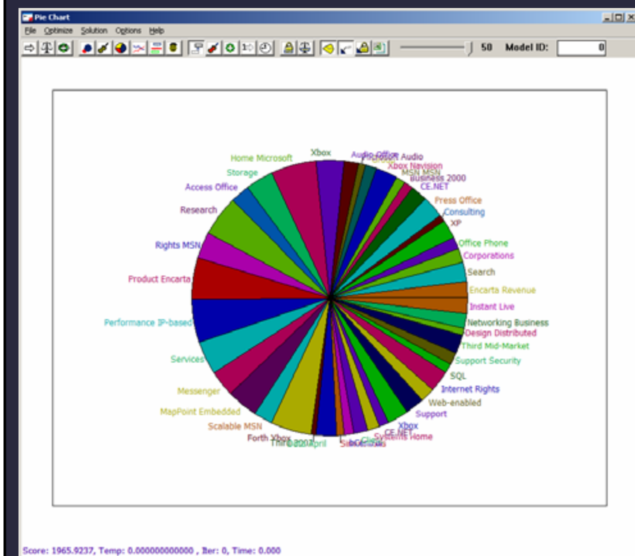

Xbox
Home Microsoft
Storage
Access Office
Research
Rights MSN
Product Encarta
Performance IP-based
Services
Messenger
MapPoint Embedded
Scalable MSN
Xbox Navision
MSN MSN
Business 2000
CE.NET
Press Office
Consulting
XP
Office Phone
Corporations
Search
Encarta Revenue
Instant Live
Networking Business
Design Distributed
Third Mid-Market
Support Security
SQL
Internet Rights
Web-enabled
Support
Xbox
Systems Home


Xbox
Home Microsoft
Storage
Access Office
Research
Rights MSN
Product Encarta
Performance IP-based
Services
Messenger
MapPoint Embedded
Scalable MSN
Xbox Navision
MSN MSN
Business 2000
CE.NET
Press Office
Consulting
XP
Office Phone
Corporations
Search
Encarta Revenue
Instant Live
Networking Business
Design Distributed
Third Mid-Market
Support Security
SQL
Internet Rights
Web-enabled
Support
Xbox
Systems Home


Xbox
Home Microsoft
Storage
Access Office
Research
Rights MSN
Product Encarta
Performance IP-based
Services
Messenger
MapPoint Embedded
Scalable MSN
Forth Xbox
Xbox Navision
MSN MSN
Business 2000
CE.NET
Press Office
Consulting
XP
Office Phone
Corporations
Search
Encarta Revenue
Instant Live
Networking Business
Design Distributed
Third Mid-Market
Support Security
SQL
Internet Rights
Web-enabled
Support
Xbox
Systems Home


Xbox
Home Microsoft
Storage
Access Office
Research
Rights MSN
Product Encarta
Performance IP-based
Services
Messenger
MapPoint Embedded
Scalable MSN
Forth Xbox
Xbox Navision
MSN MSN
Business 2000
CE.NET
Press Office
Consulting
XP
Office Phone
Corporations
Search
Encarta Revenue
Instant Live
Networking Business
Design Distributed
Third Mid-Market
Support Security
SQL
Internet Rights
Web-enabled
Support
Xbox
Systems Home

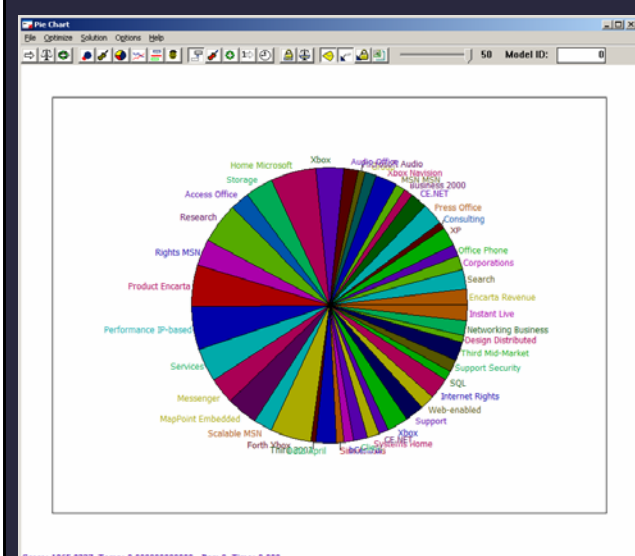
Many dimensions → large space



- **Position *(x, y)***
- Leader line
- Word wrap
- Color
- Alignment
- Orientation
- Scale

2D x 50 labels → 100D space

Penalties



Overlap & Distance

- Label – anchor slice
- Label – other slices
- Label – label

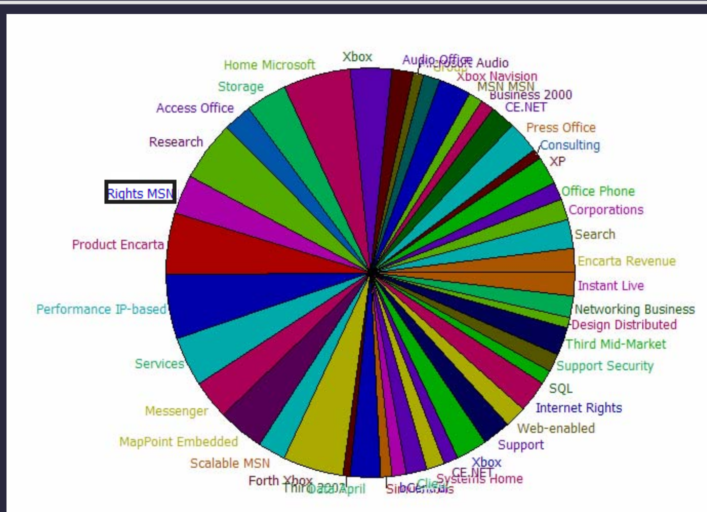
Leader lines

- Length
- Intersections

Word Wrap

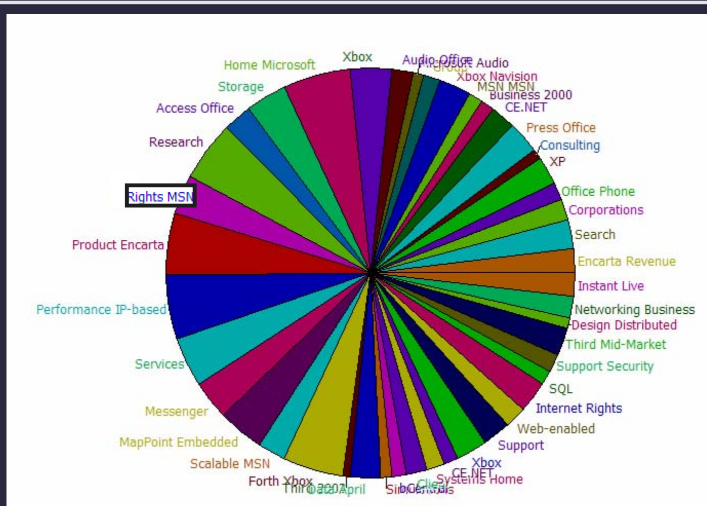
Annealing minimizes sum of all penalties

Overlap: Label – Anchor Slice



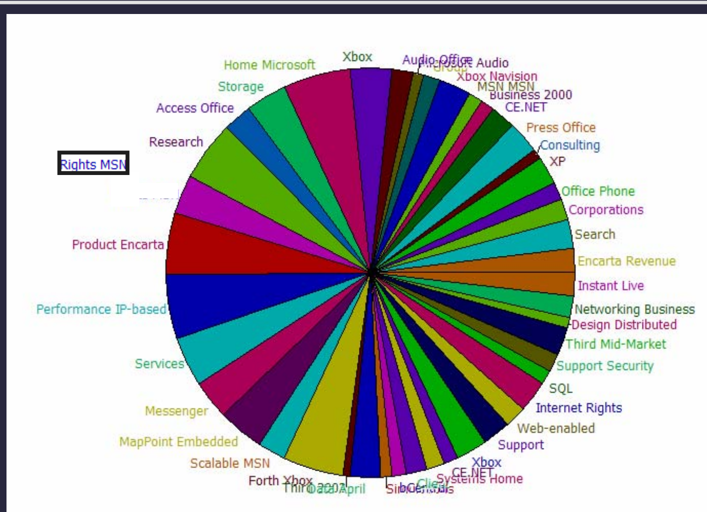
Avoid partial overlap: No penalty if fully inside /outside

Overlap: Label – Anchor Slice



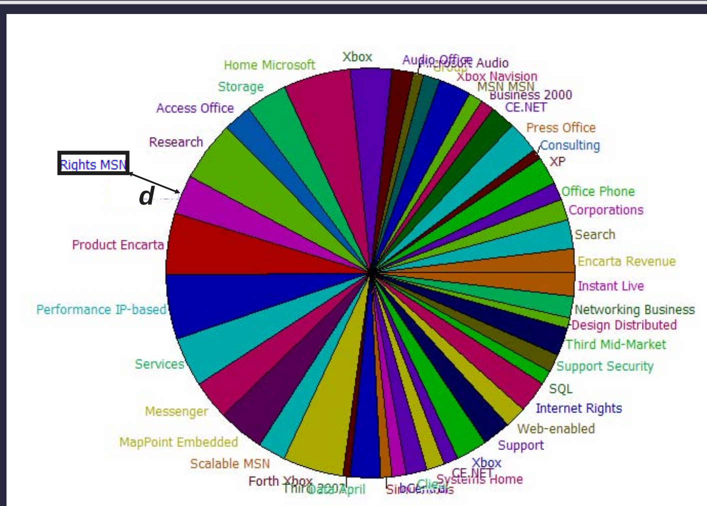
Penalize partial overlap by overlap amount

Distance: Label – Anchor Slice



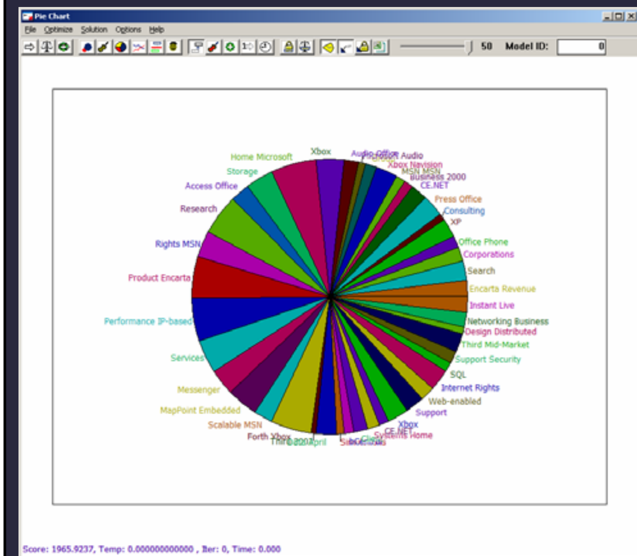
Ensure label near center of edge of anchor slice

Distance: Label – Anchor Slice



Minimize distance *d*

Penalties



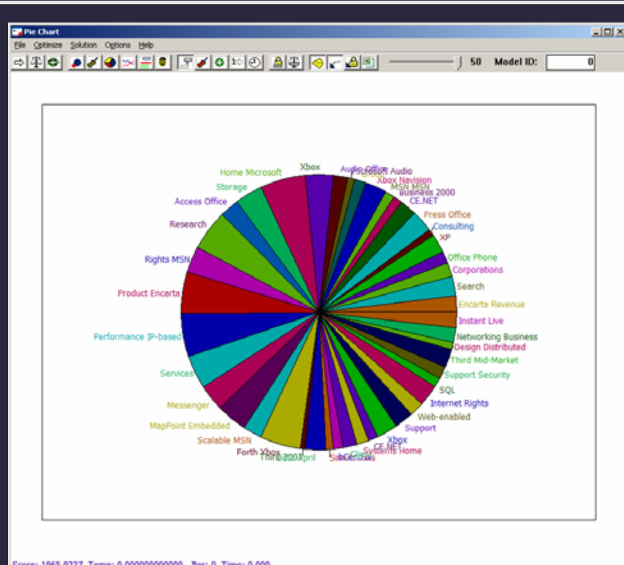
Overlap & Distance
- Label – anchor slice
- Label – other slices
- Label – label

Leader lines
- Length
- Intersections

Word Wrap

Annealing minimizes sum of all penalties

Demo



Pros and cons

Pros

- **Much more flexible than linear constraint solving systems**

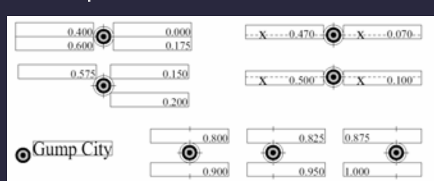
Cons

- **Can be relatively slow to converge**
- **Need to set penalty function parameters (weights)**
- **Difficult to encode desired layout in terms of mathematical penalty functions**

Design principles

Sometimes specified in design books

- Tufte, Few, photography manuals, cartography books …
- Often specified at a high level
- Challenge is to transform principles into constraints or penalties



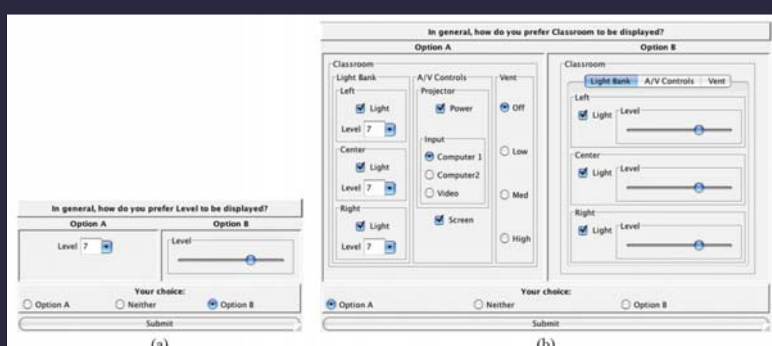
Cartographer Eduard Imhof's labeling heurists transformed into penalty functions for an optimization based point labeling system [Edmondson 97]

Example-Based Methods

Preference elicitation [Gajos and Weld 05]

Learn characteristics of good designs

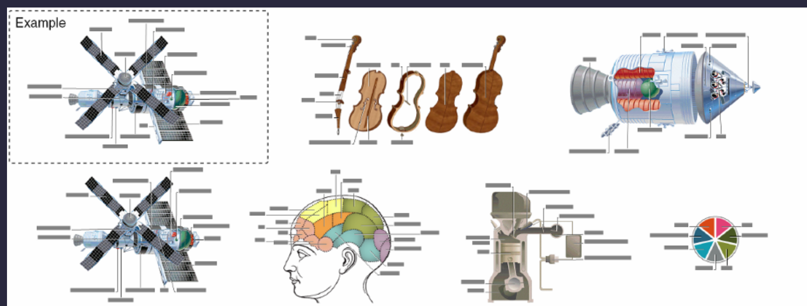
- Generate designs based on a parameterized design space
- Ask designers if they are good or bad
- Learn good parameters values based on responses



(a) (b)

Nonlinear Inverse Opt. [Vollick et al. 07]

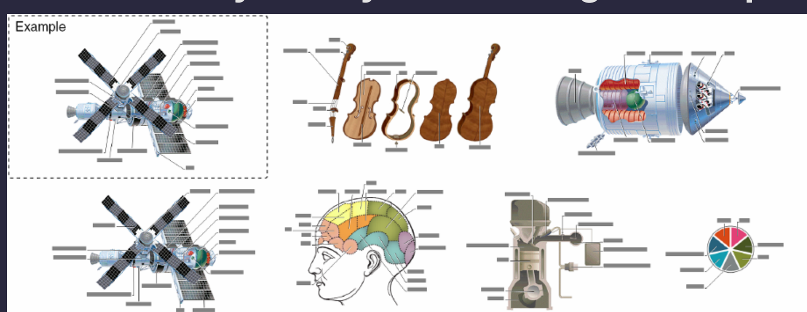
Learn label layout style from single example



Horizontal/Vertical

Nonlinear Inverse Opt. [Vollick et al. 07]

Learn label layout style from single example



Parallel Leader Lines

Artistic Resizing

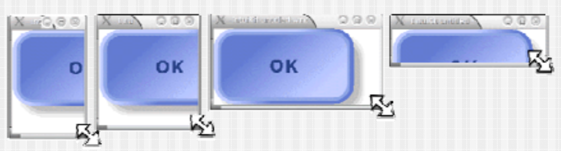
A Technique for Rich
Scale-Sensitive Vector Graphics

Pierre Dragicevic
Stéphane Chatty
David Thevenin
Jean-Luc Vinot

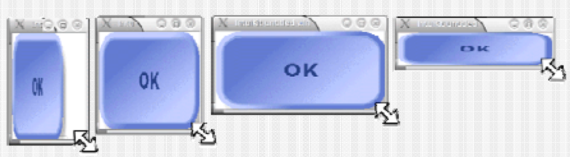
Direction Générale de l'Aviation Civile
dgac

The Resizing Problem

- Fixed size



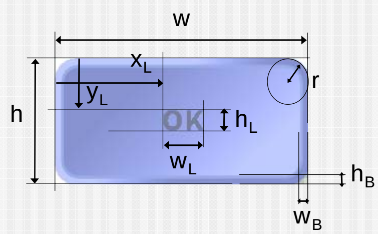
- Naive scaling



- Artistic resizing

Expressing Artistic Resizing

- Commonly described using formulae



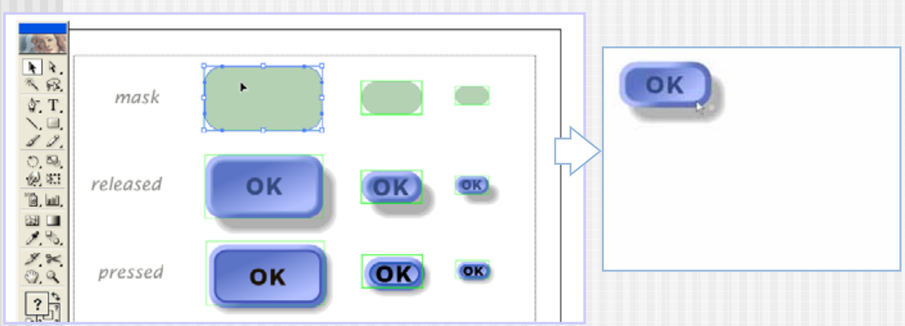
- $x_L = (w-w_L) / 2$
- $y_L = (h-h_L) / 2$
- $w_L = 20$
- $h_L = 10$

- $w_B = 5$
- $h_B = 5$
- $r = 20$

- These formulae are:
  - Translated into code by the programmer
  - Or used as an input to constraint-solving systems

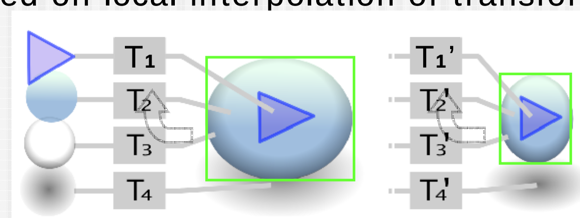
Example-Based Approach

1. Designers produce variants using their authoring tool
2. System interprets the example set



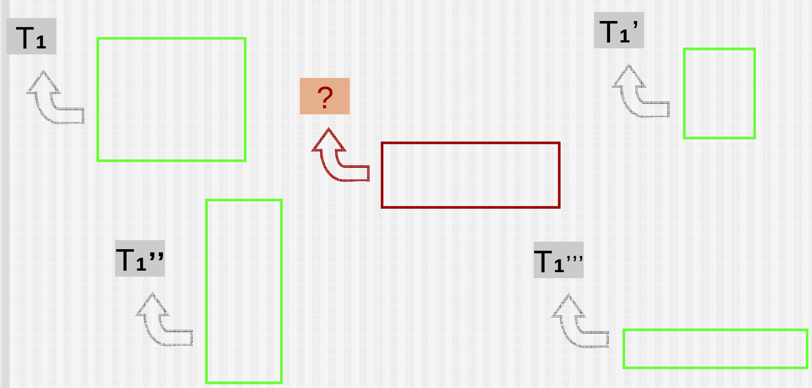
Artistic Resizing
How does it work?

- Assumes the exclusive use of:
  - Copy & paste for adding new examples
  - Affine transformation tools (move, scale, rotate, shear)
- Based on local interpolation of transformations



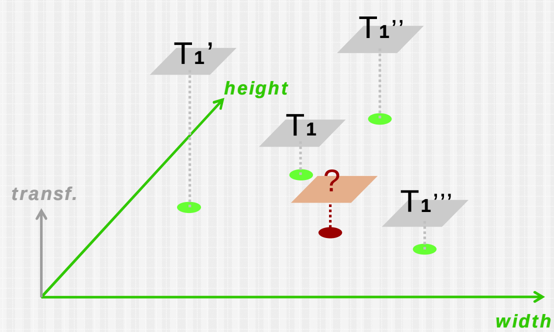
Artistic Resizing
How does it work?

- Each variant of T1 is associated with the example's bounding box



Artistic Resizing
How does it work?

- Problem of multivariate interpolation



Pros and cons

Pros

- Often much easier to specify desired layout via example

Cons

- Usually requires underlying model
- Model will constrain types of layouts possible
- Large design spaces likely to require lots of examples to learn parameters well