

methods of computational science

visualization

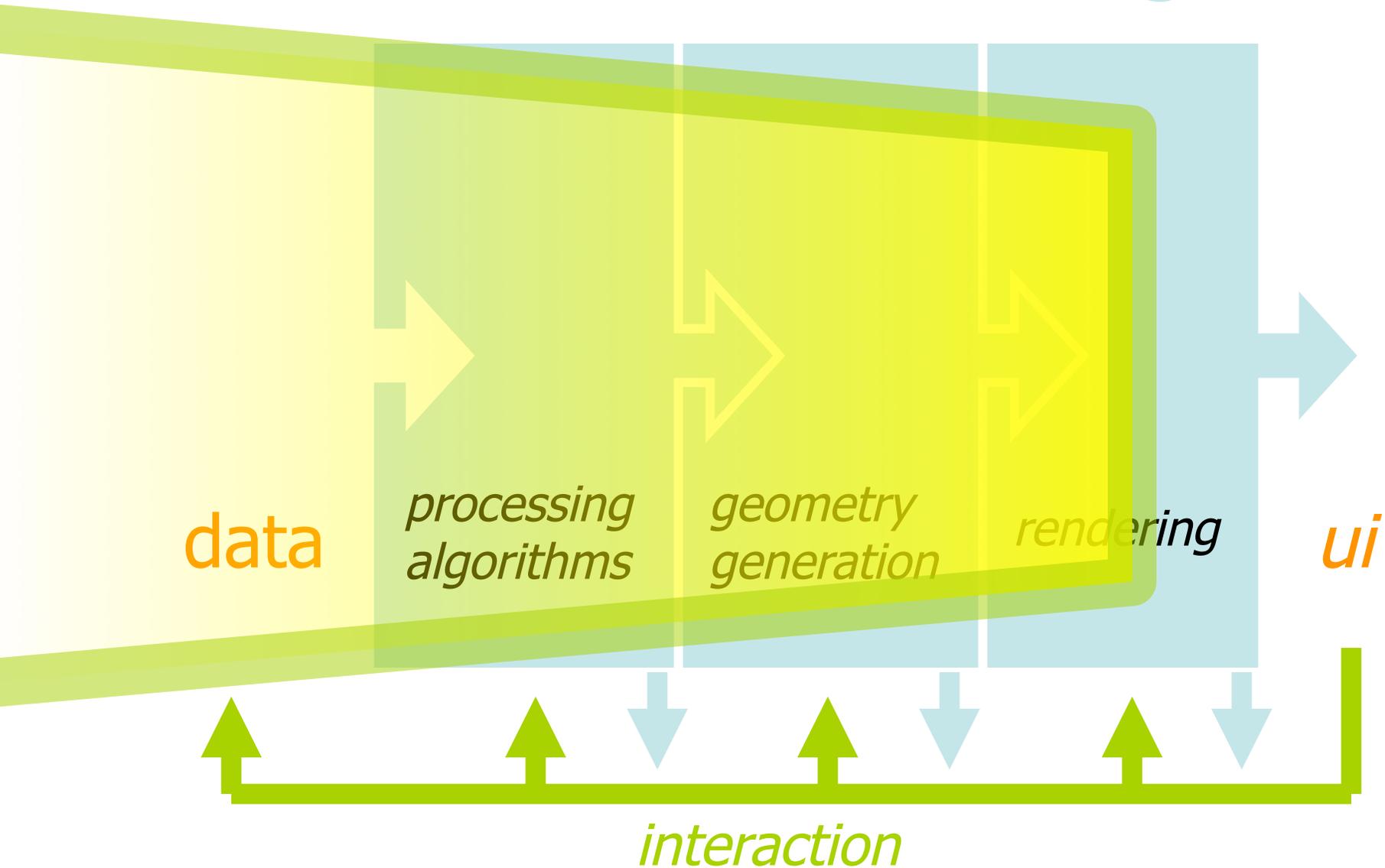
day iii – case studies + advanced viz

santiago v lombeyda

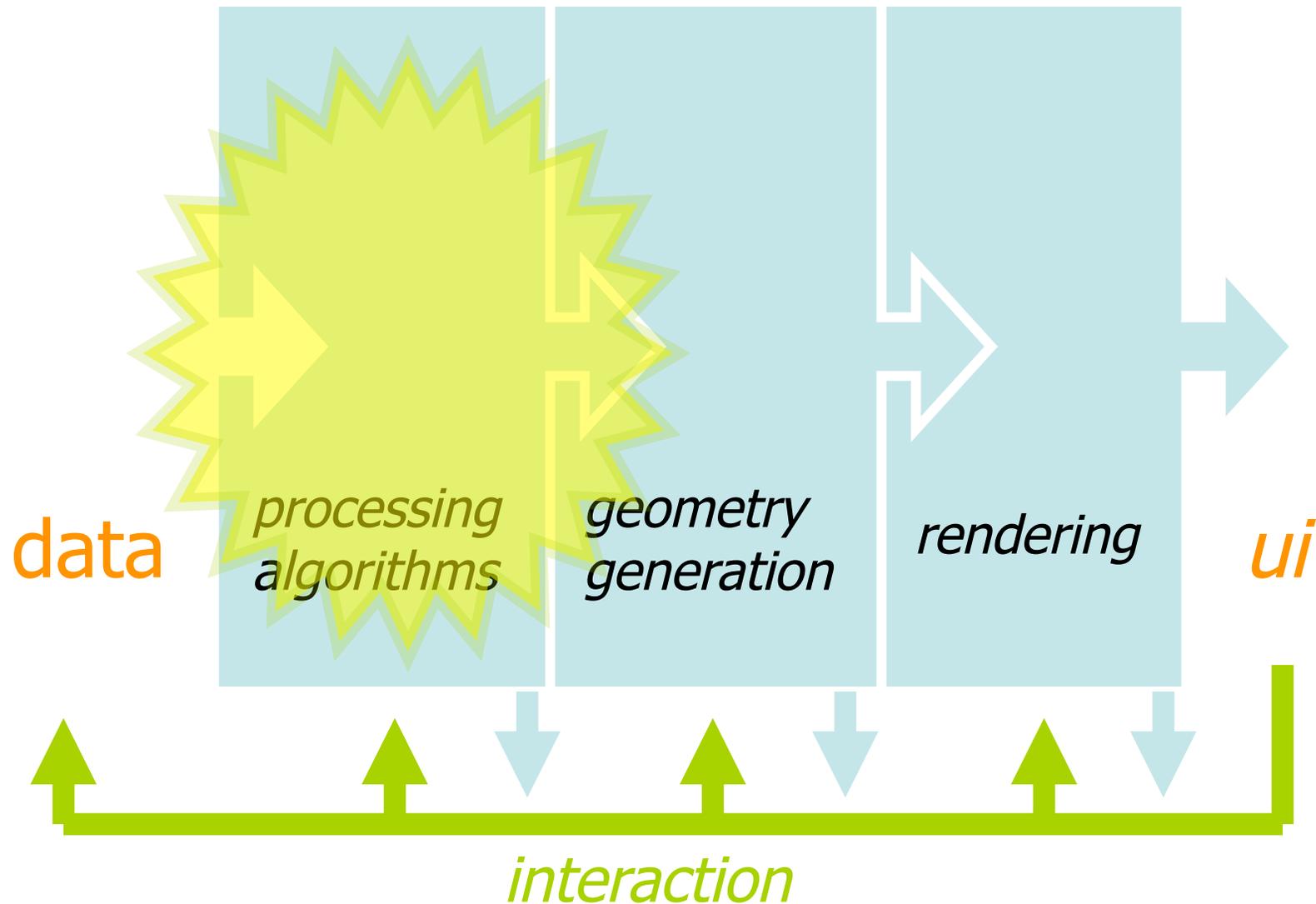
center for advanced computing research

caltech

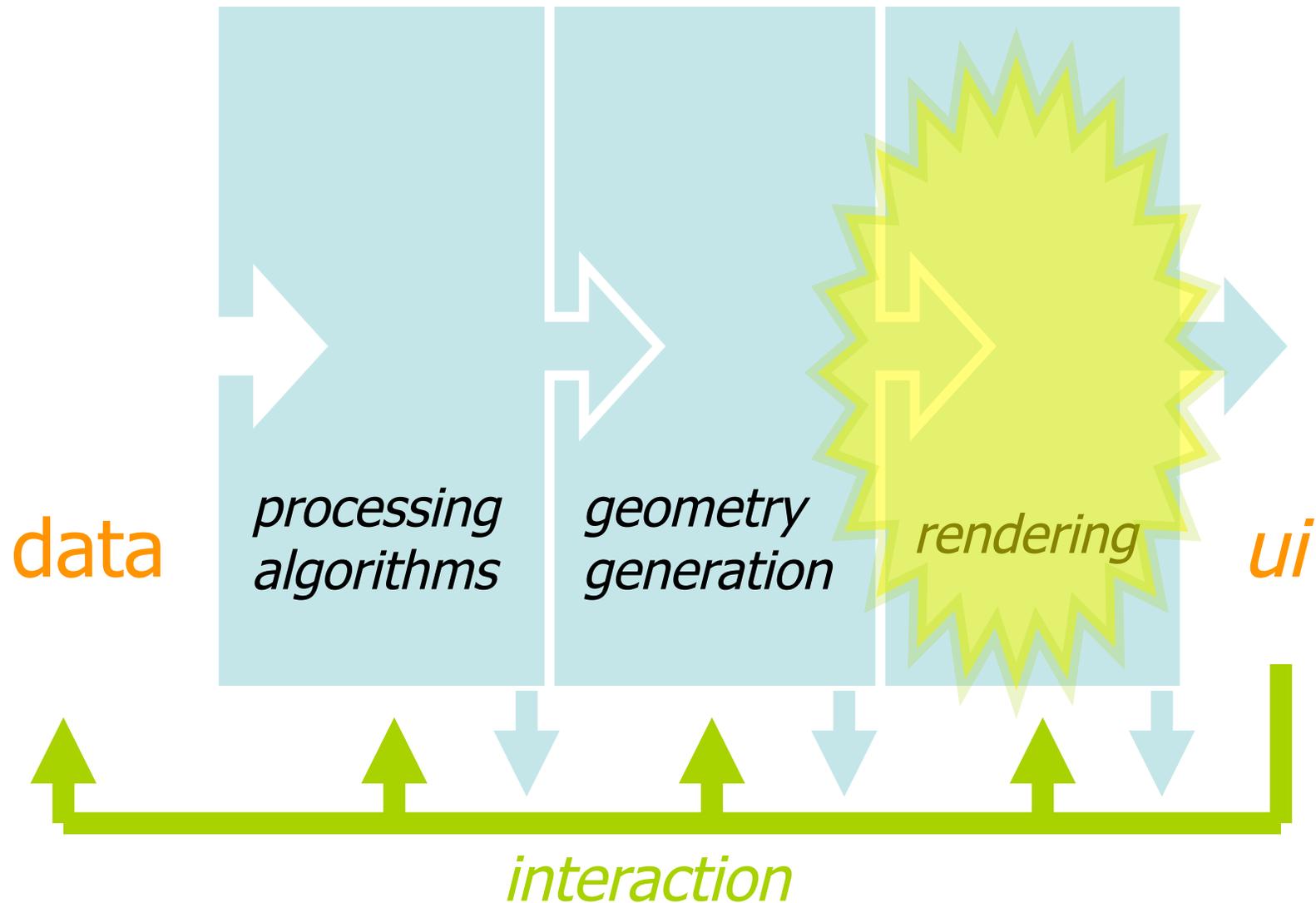
usual visualization "engine"



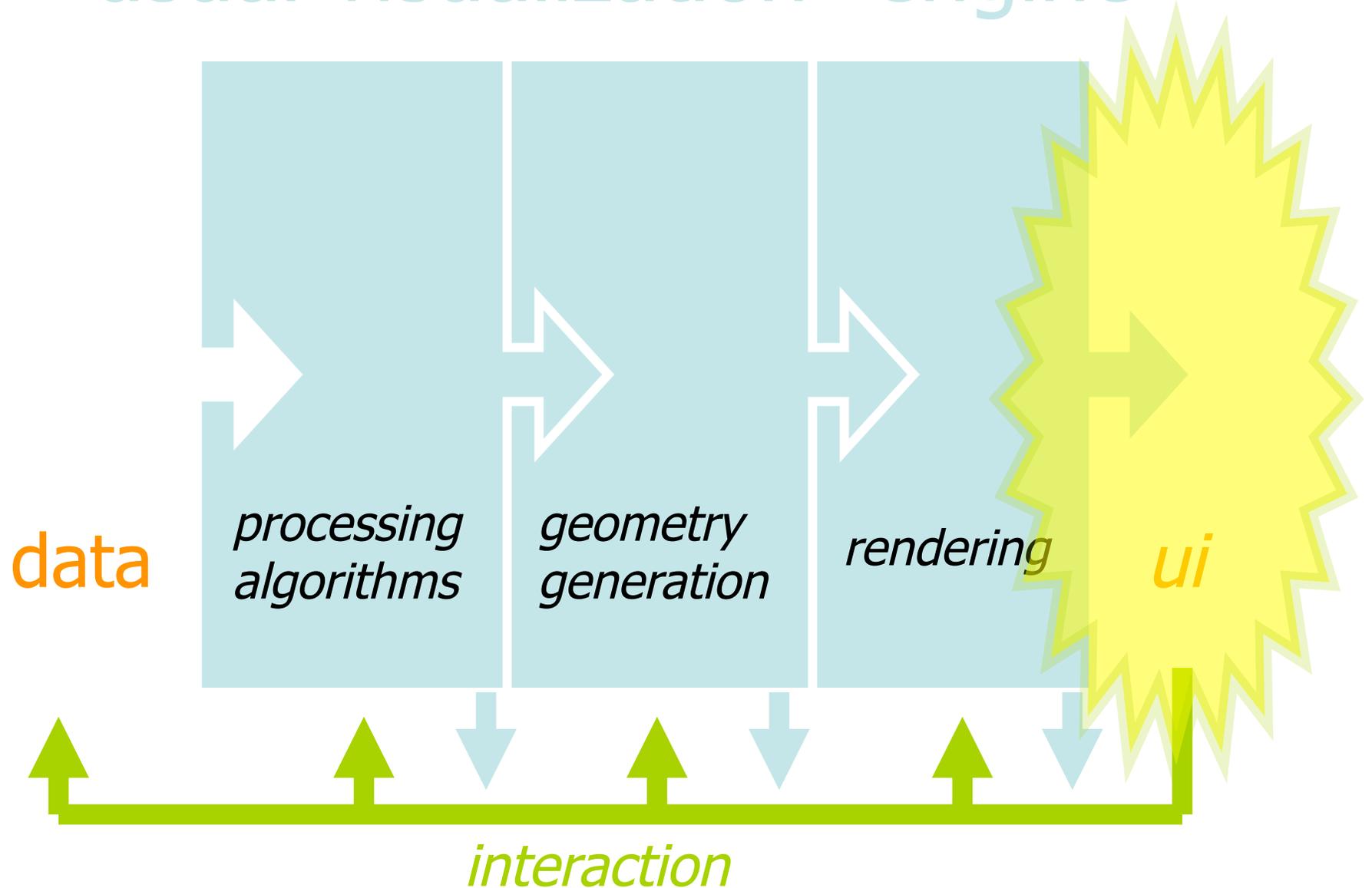
usual visualization "engine"



usual visualization "engine"



usual visualization "engine"



usual visualization "engine"

rearranging your data smartly...

data

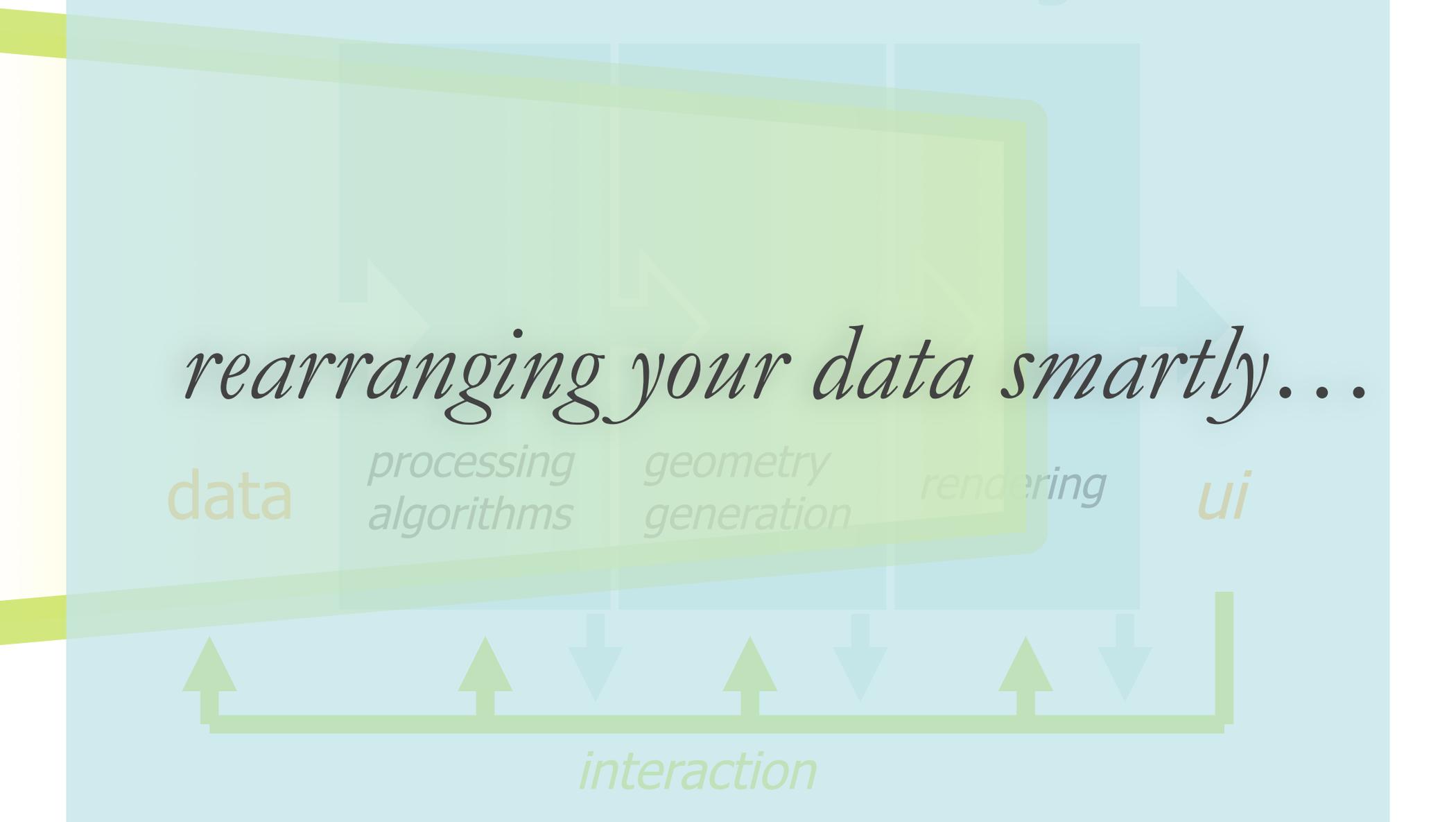
*processing
algorithms*

*geometry
generation*

rendering

ui

interaction

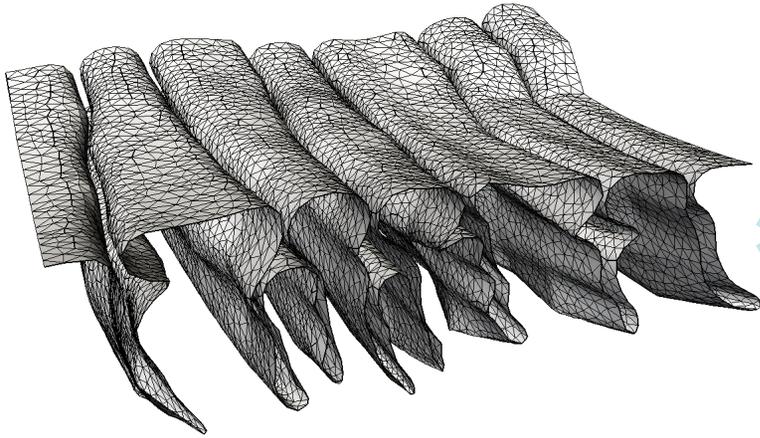


advanced topics: *case study*:
TETSPLAT

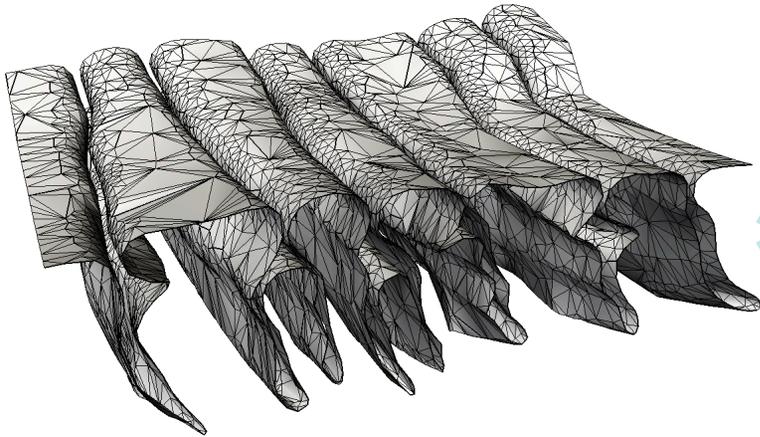
addressing *throughput bottlenecks*

- * level of detail (LOD)
 - * requires preprocessing
 - * requires larger storage (*original+...*)
- * parallel processing/rendering
 - * requires a parallel system
 - * increases sw complexity
 - * less likely to be “portable”

lod: decimation

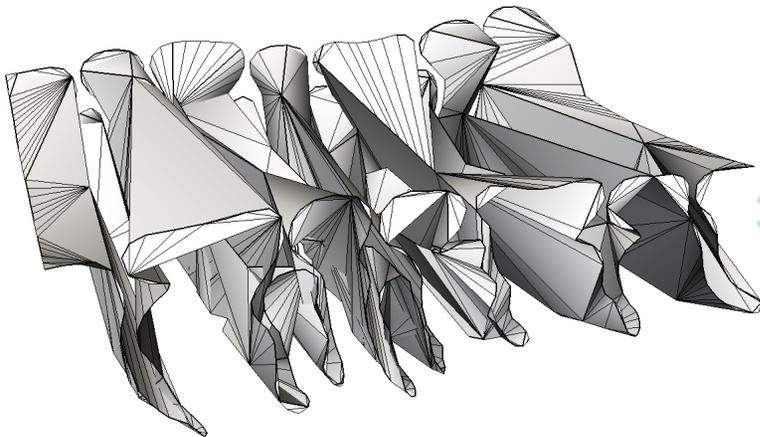


* 25365



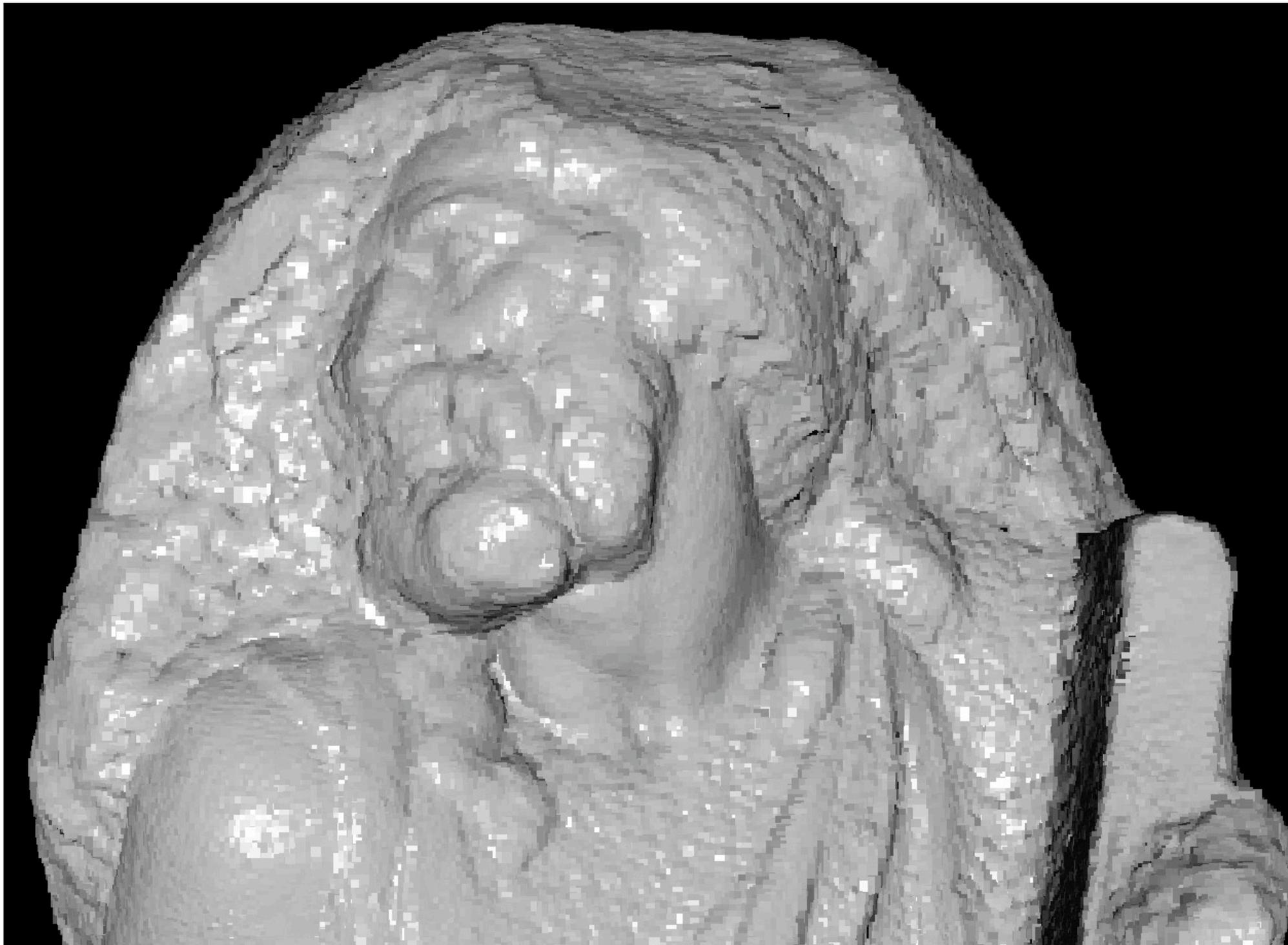
* 12681

* 50%



* 1011

* 4%

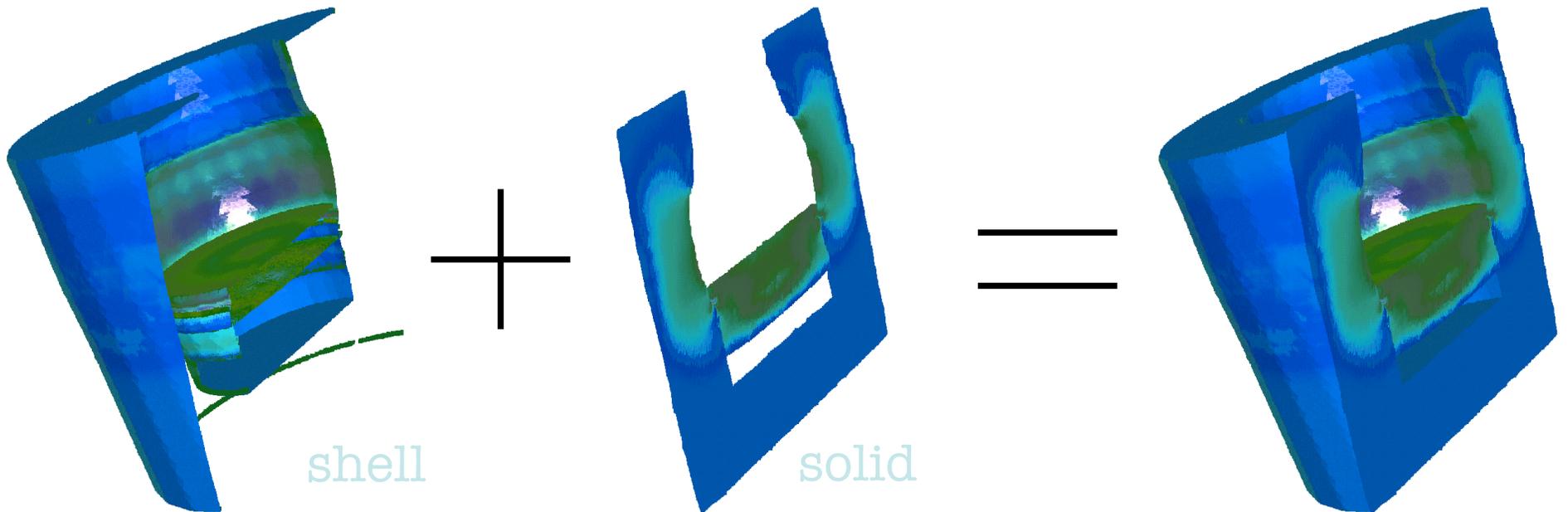
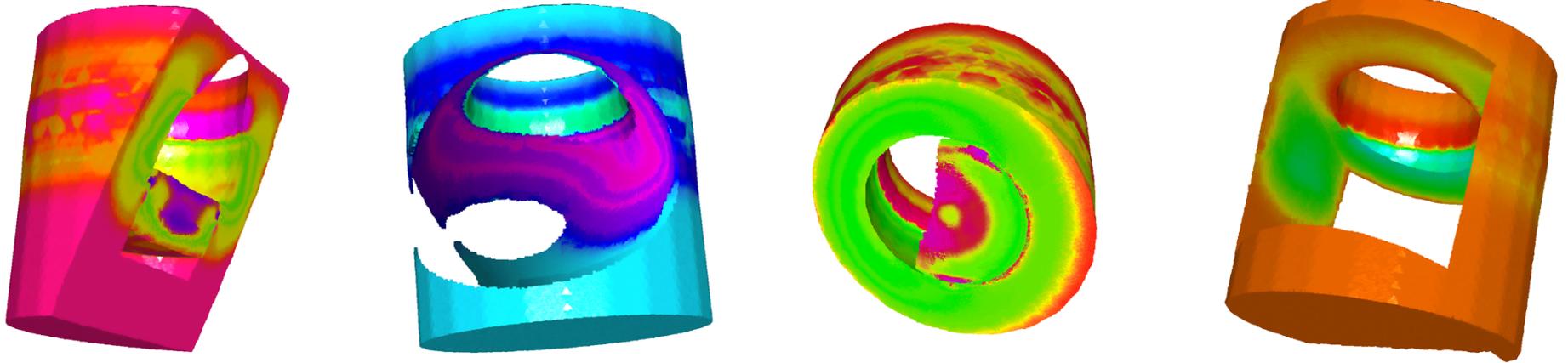






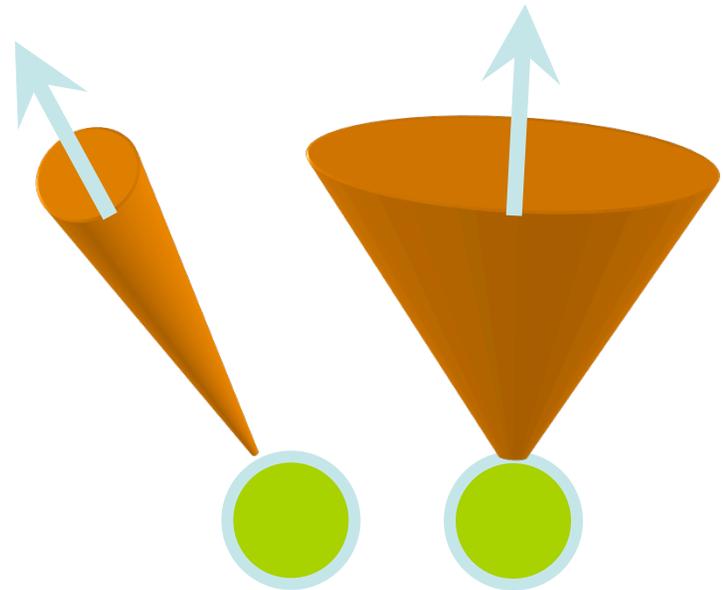


exploring the shock-test cannister



tet-splat?

- * tet-mesh (volumetric) exploration with csg proves
 - * based on qsplat [rusinkiewicz00]
- * hierarchical internal mesh representation
 - * keep simplified version of data
- * render:
 - * splats! (points)
 - * as far into tree
 - * as possible in time

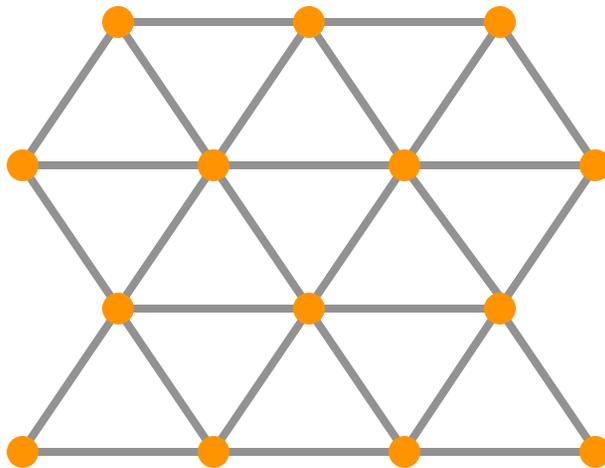


strategies for building hierarchical trees

- * separate trees for **solid** and **shell**
 - * unbalanced **octree** for **solid**
 - * unbalanced **quadtree** for **shell**
- * construct trees bottom-up
 - * **bi-sect longest edge** of axis-aligned bounding box
 - * internal nodes have **averaged** attributes
- * compress trees to-down
 - * hierarchical **delta encoding**
 - * **bit quantization**
- * flush trees to file-system breadth first
 - * allows LOD by progressive loading

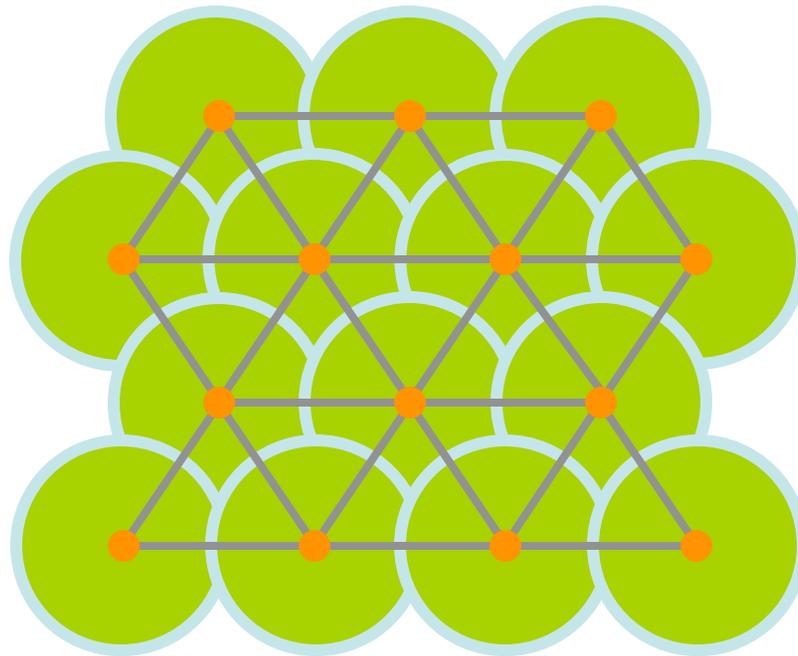
constructing trees bottom-up

- * start with mesh from solid and shell

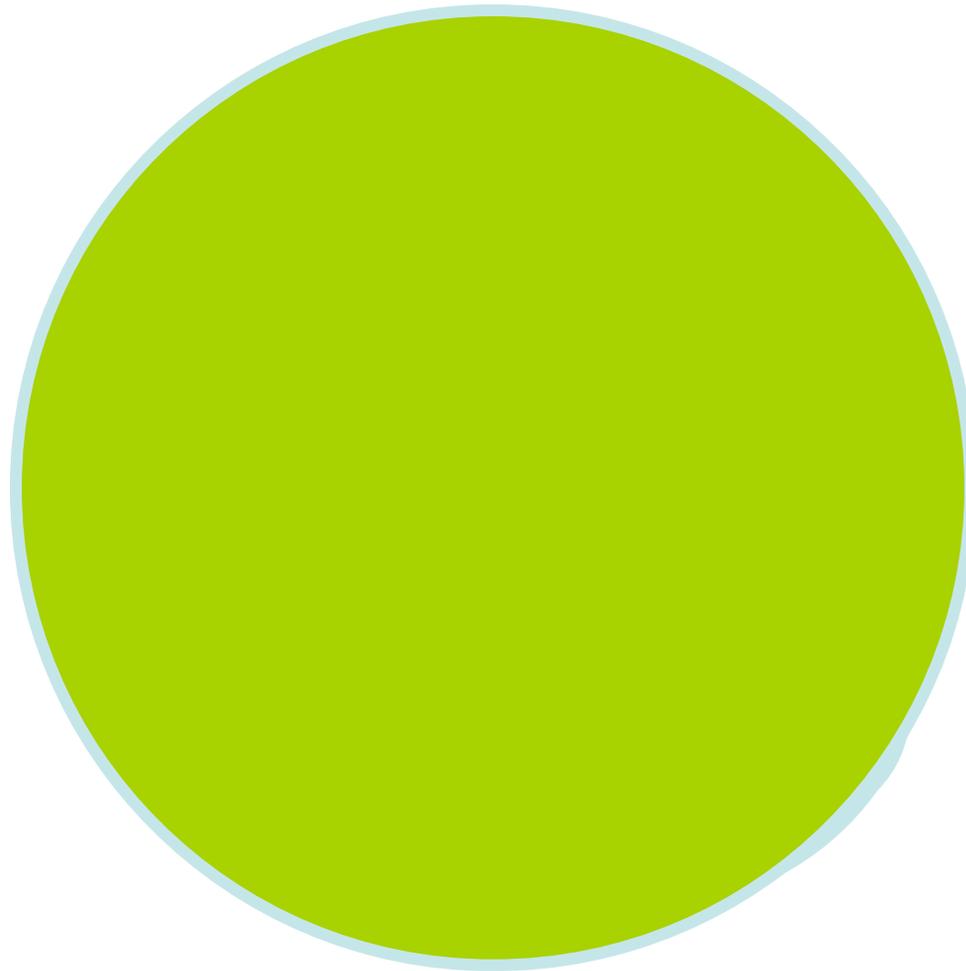


constructing trees bottom-up

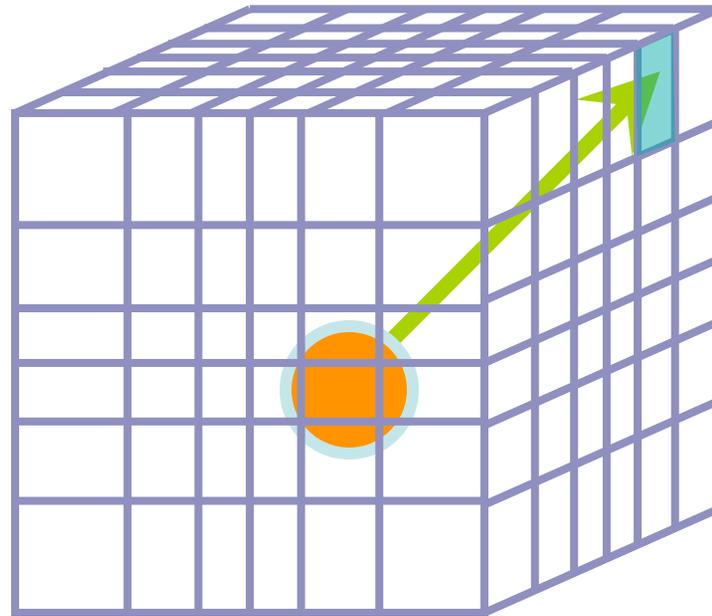
- * geometric primitive -> define circle/sphere



- * build up hierarchy:
 - * successive bisections along longest axis
 - * merging nodes -> bounding circles/spheres

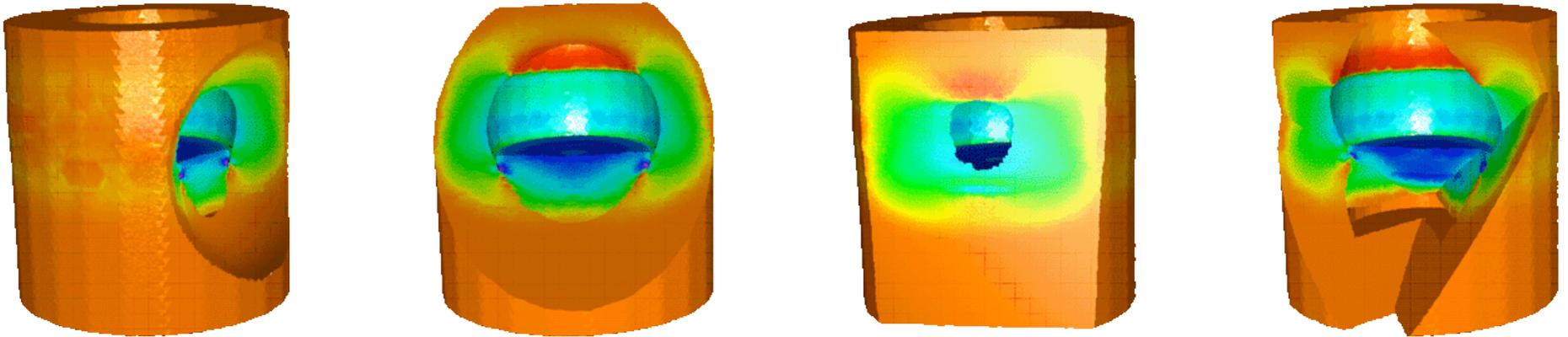


quantizing nodes



- * normals quantized to 52x52x6 values
- * width of bounding cone of child normals is quantized

results



* input: cannister data

* 5,378,048 tetrahedra

* 947,127 vertices

* 13 different field values

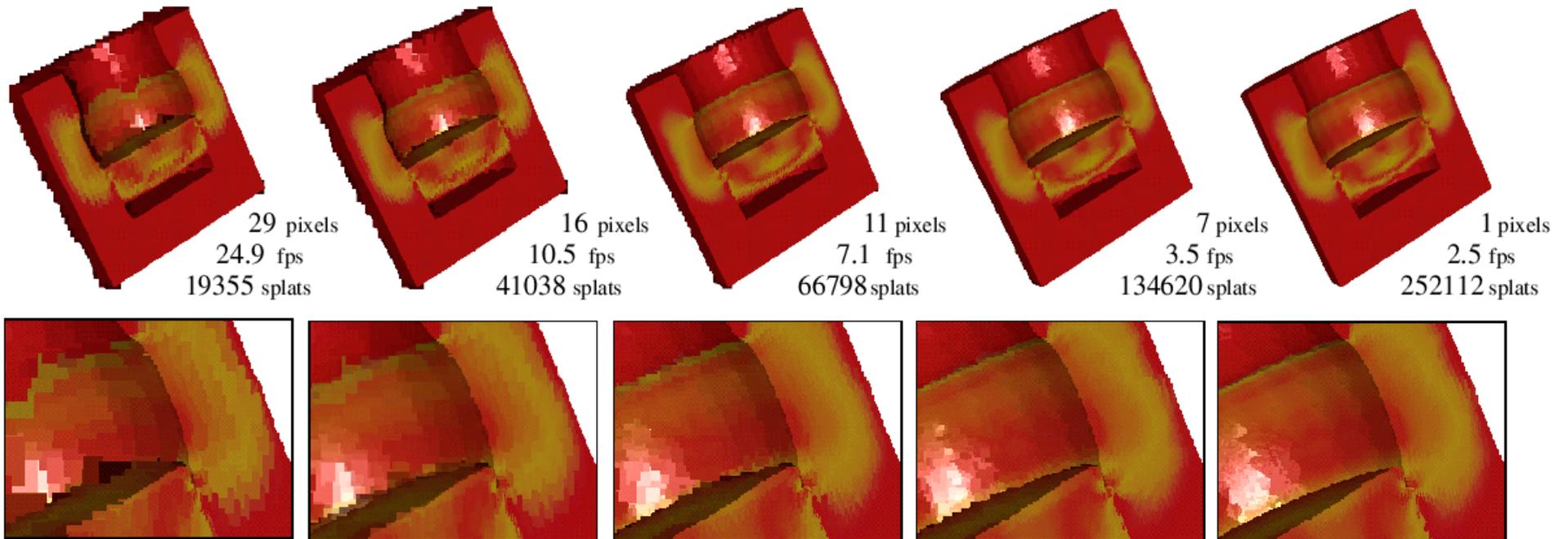
* 275 mb

* output: shell and solid

* size 10-100mb

* nodes 1-15 million

frame rate vs quality examples



usual visualization "engine"

presenting your data descriptively...

data

*processing
algorithms*

*geometry
generation*

rendering

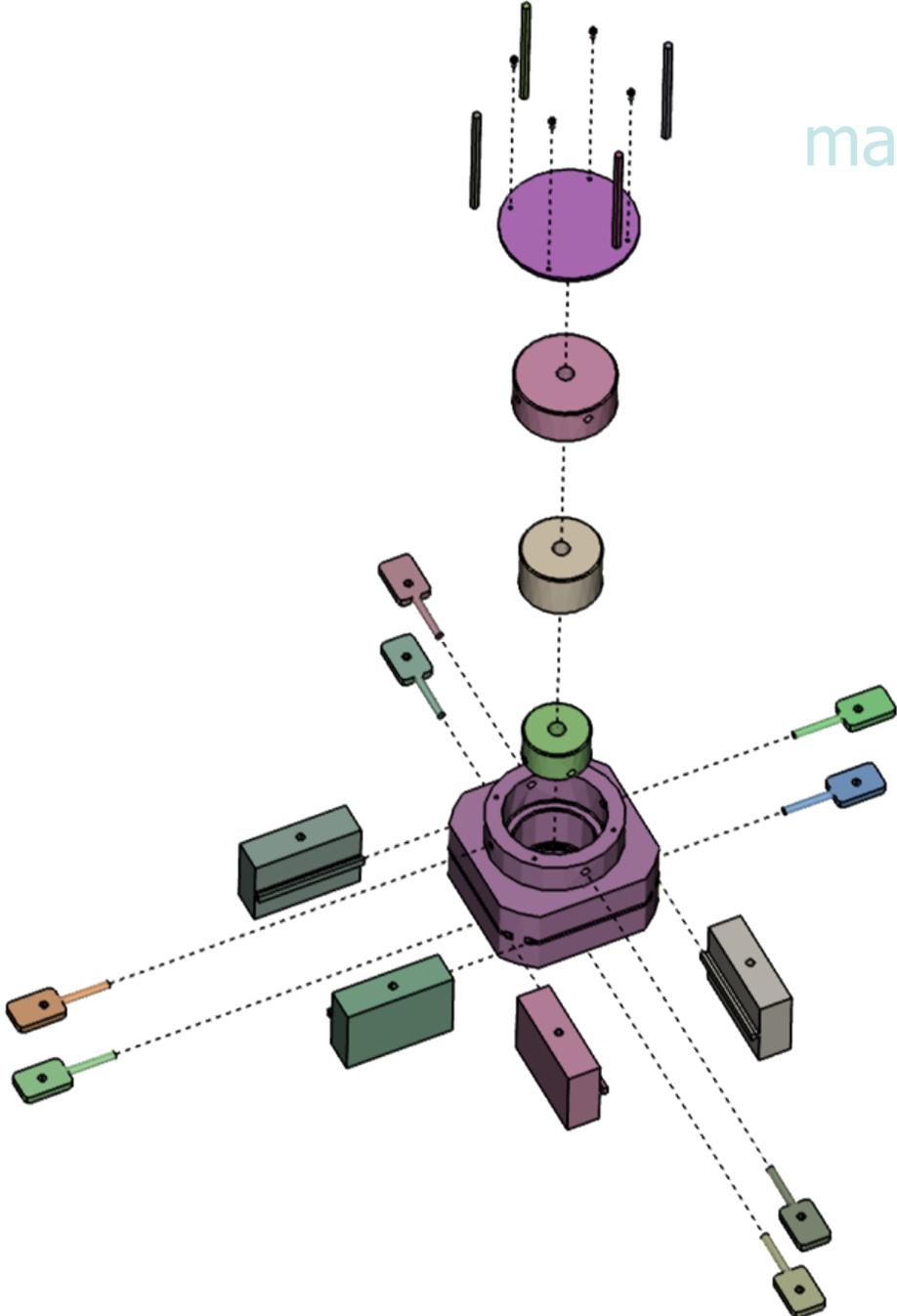
ui

interaction

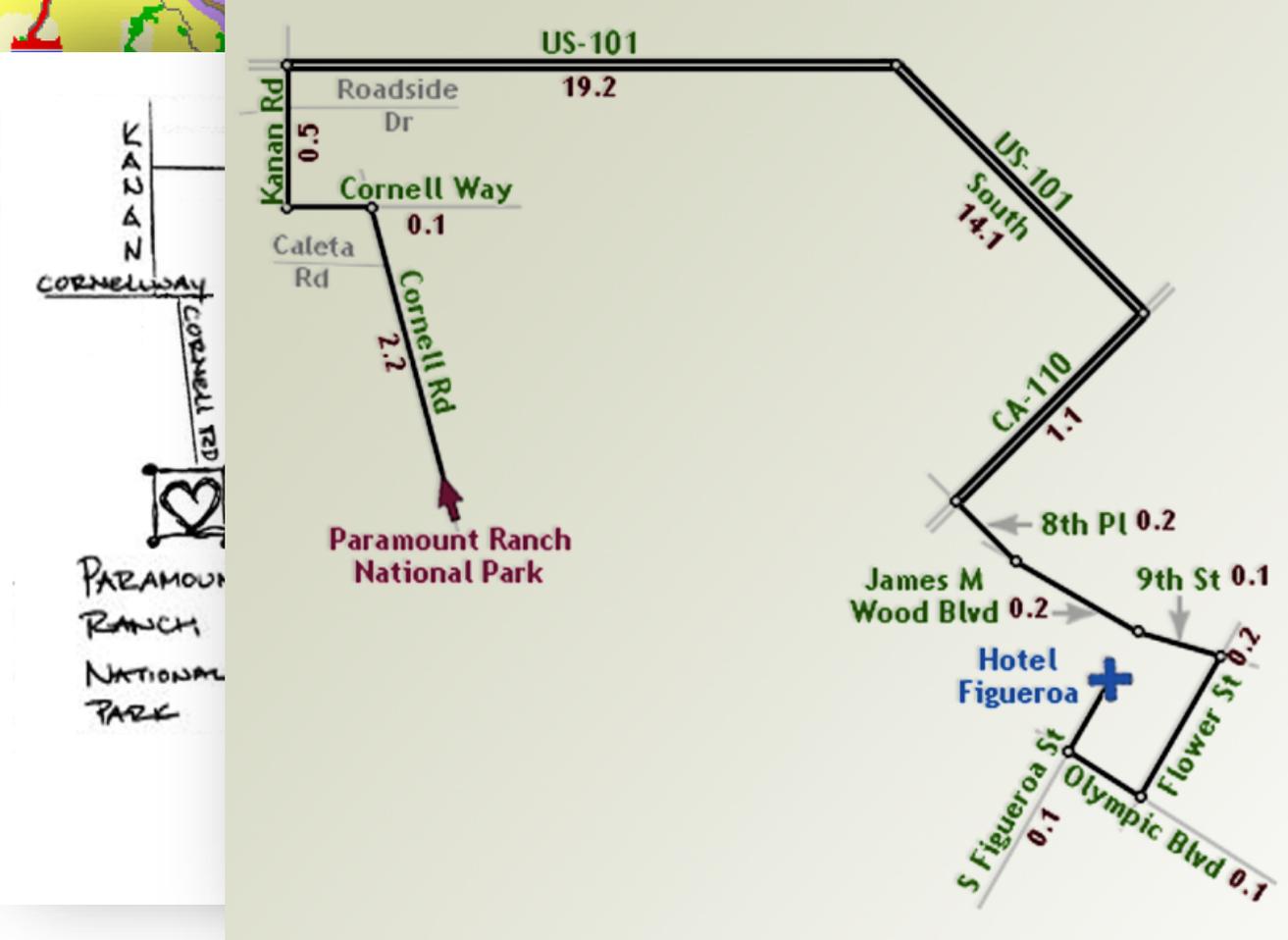


advanced topics:
INTERACTIVE CUTAWAYS

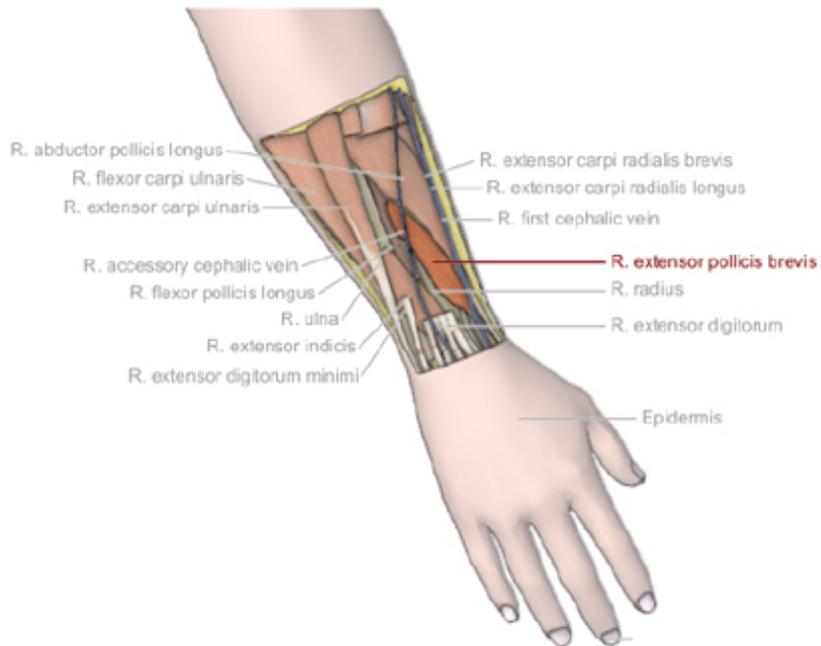
maneesh agrawala



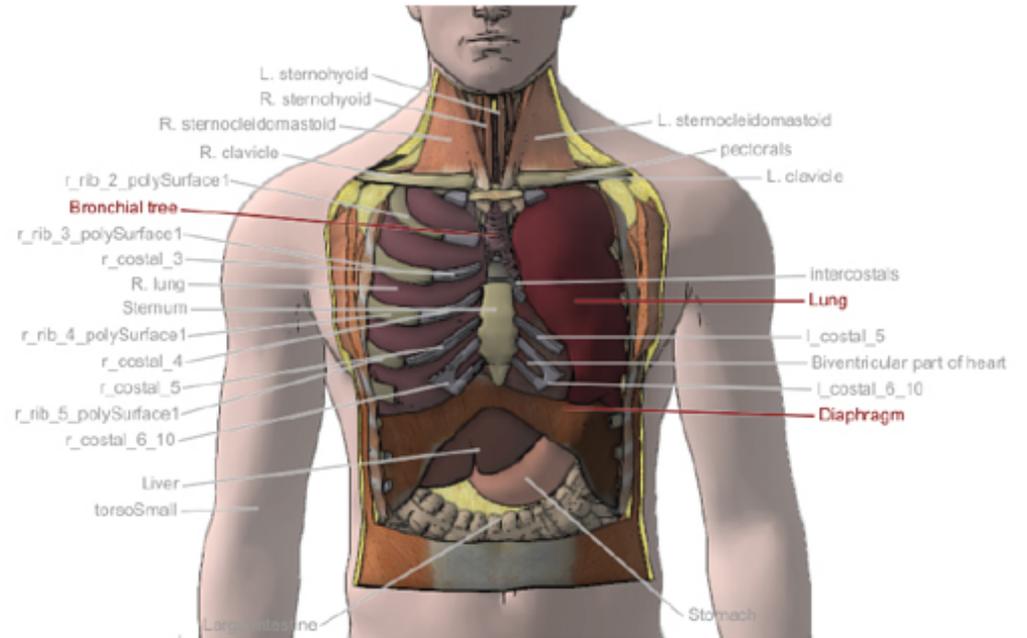
maneesh agrawala



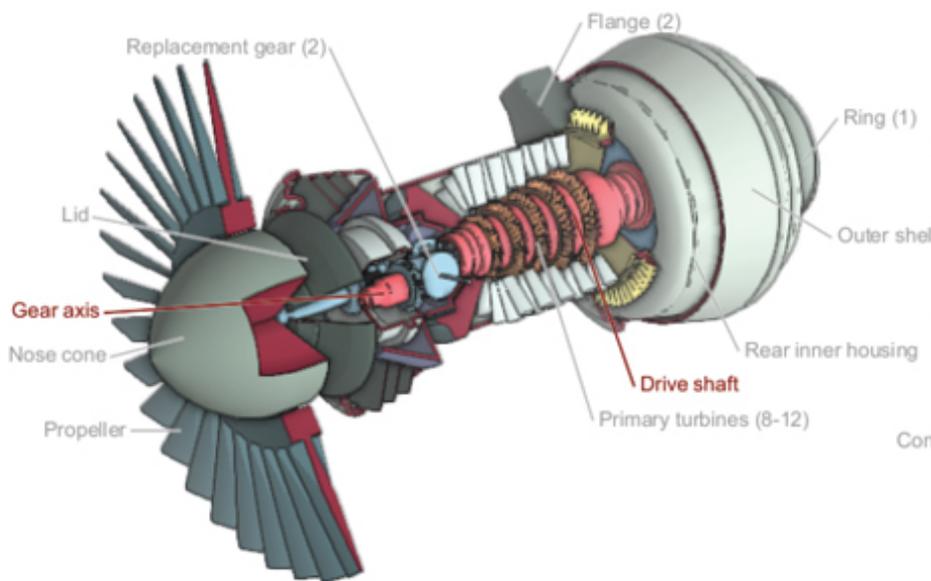
KANAN
CORNELLWAY
CORNELL RD
PARAMOUNT RANCH NATIONAL PARK



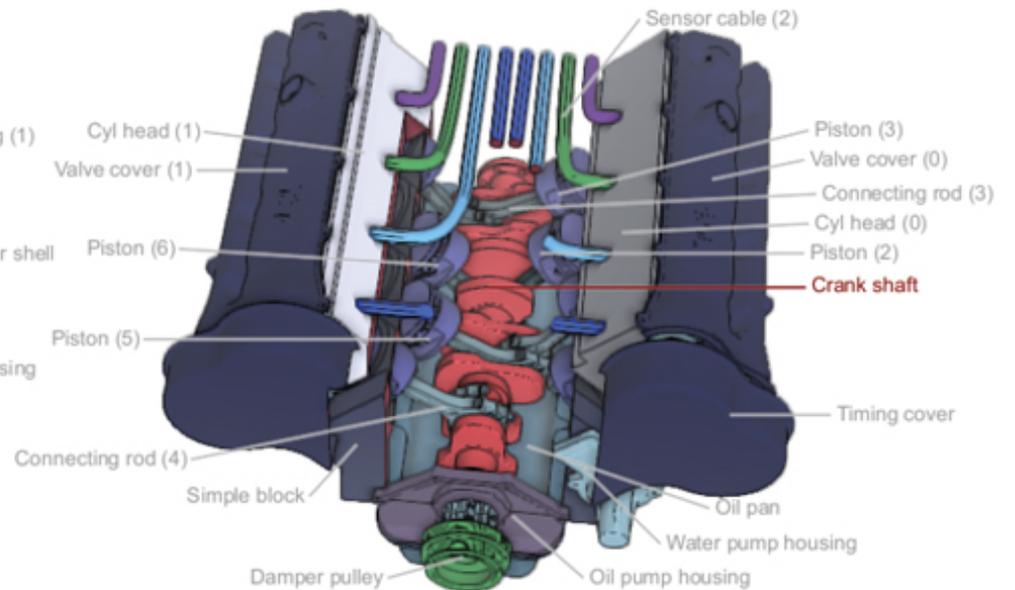
(a) Arm



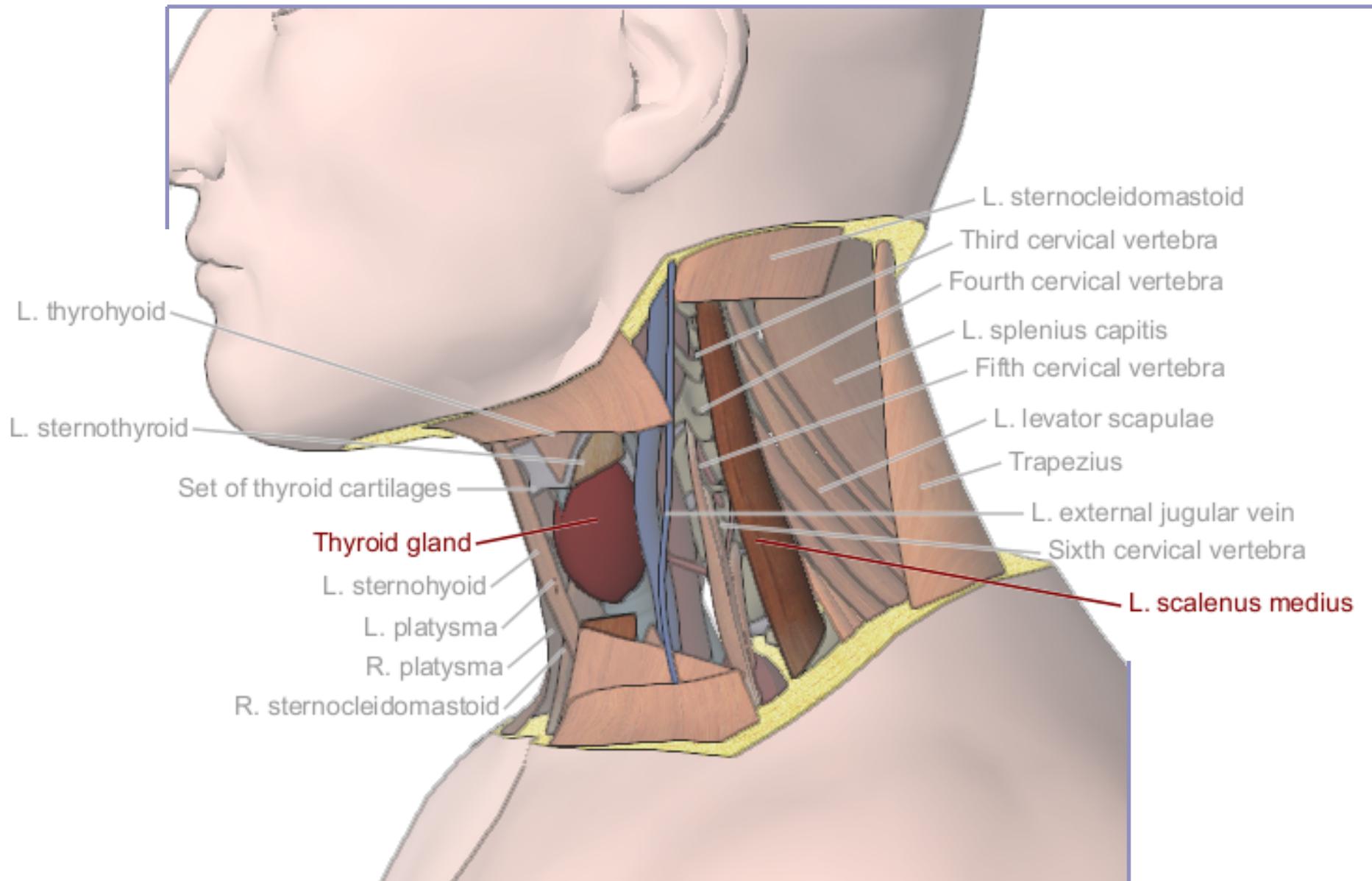
(b) Thorax



(c) Turbine



(d) Engine



usual visualization "engine"

rendering your data intuitively...

data

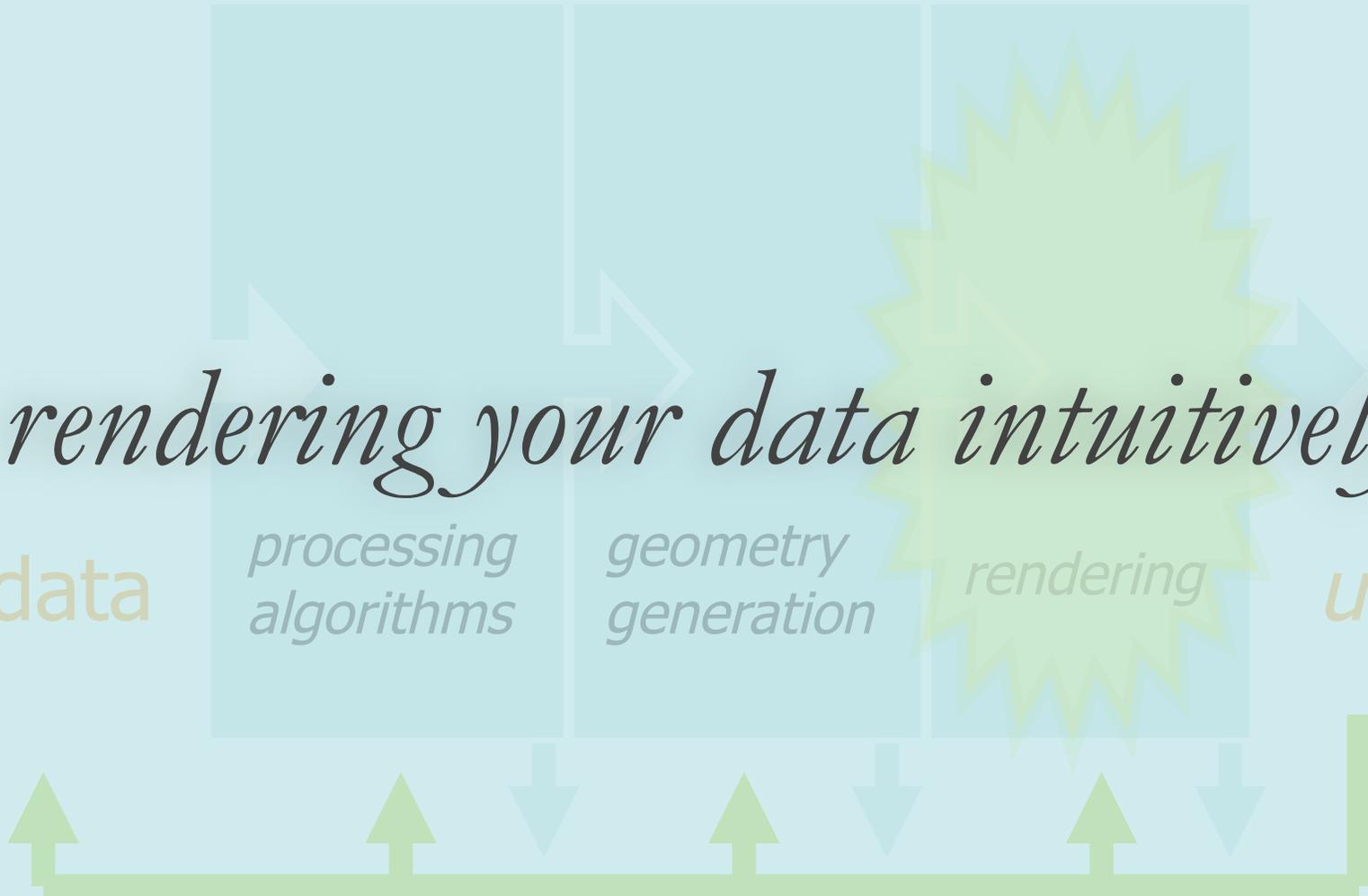
*processing
algorithms*

*geometry
generation*

rendering

ui

interaction



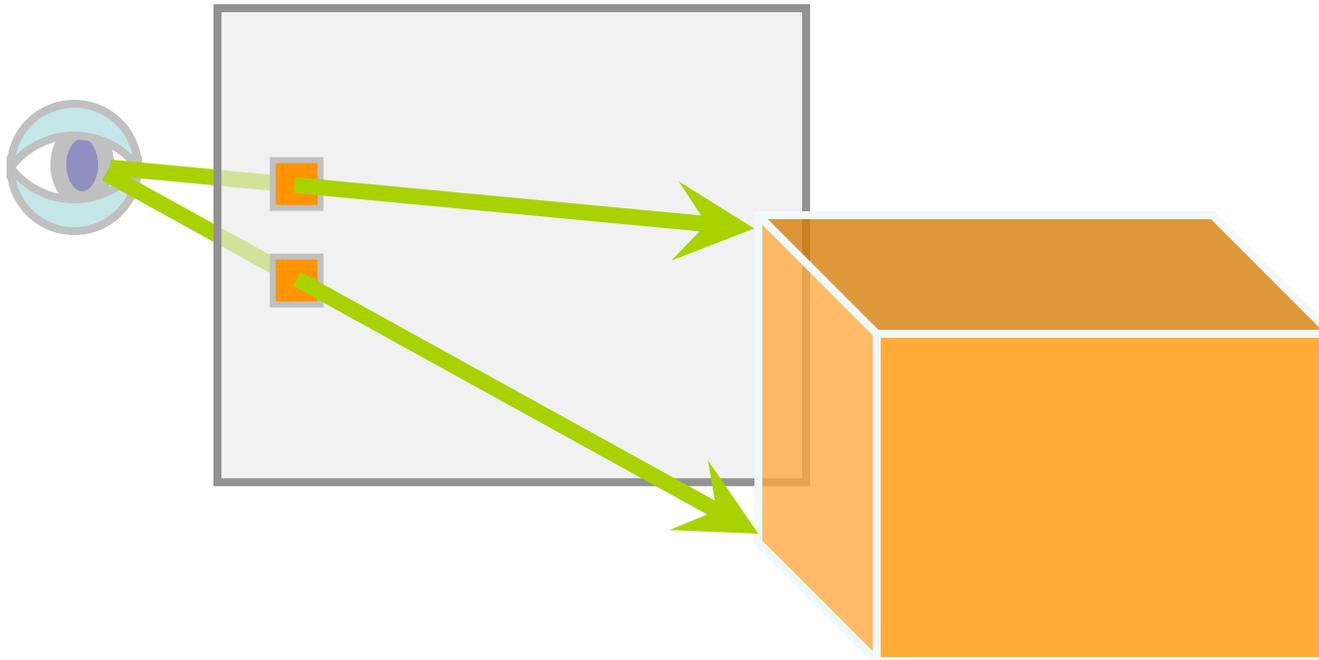
advanced topics: *closer look:*
VOLUME RENDERING

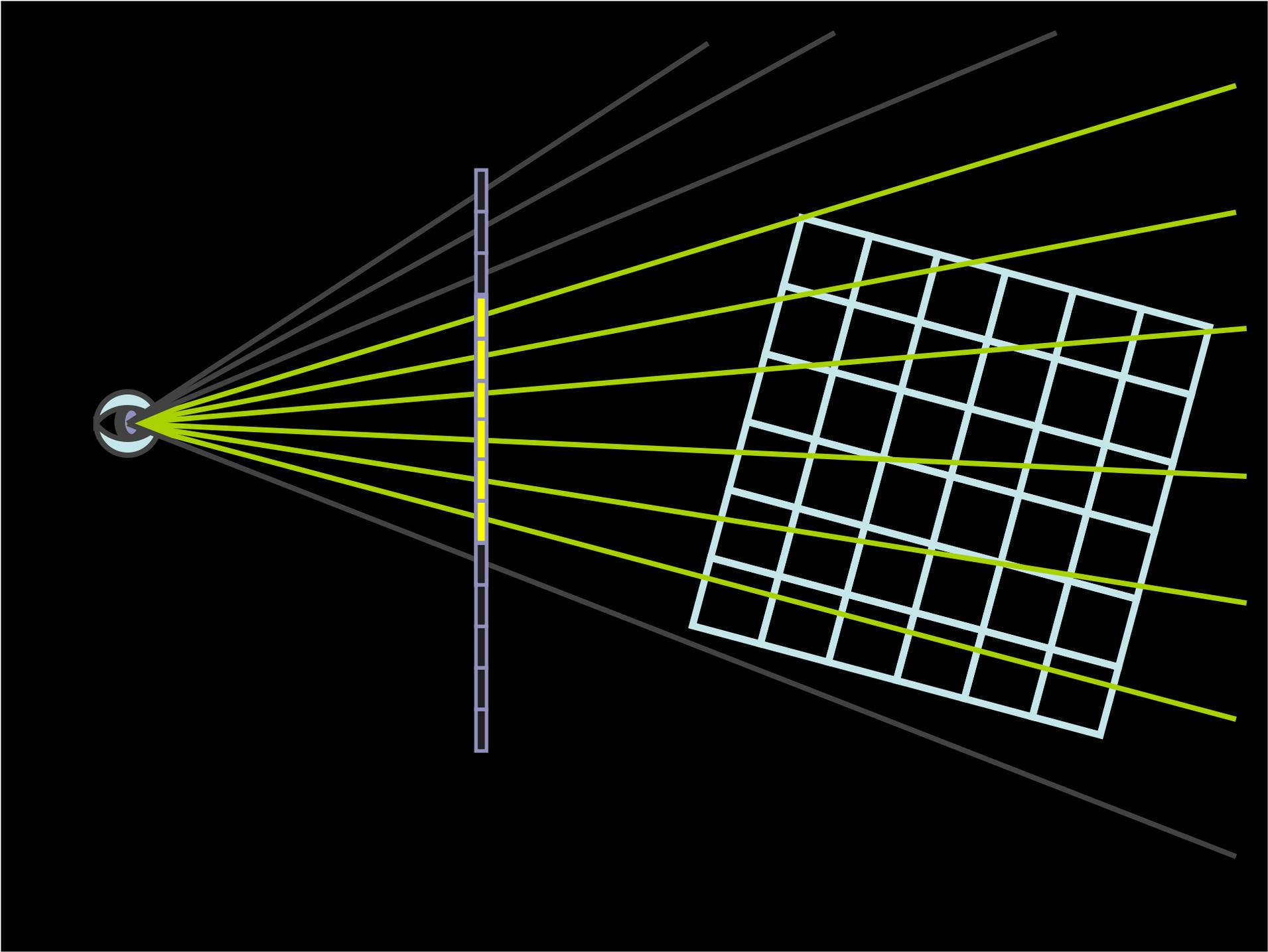
volume rendering

- * volumetric data
 - * *voxels*: volume elements
 - * usually regular 3D grid
 - * scalar field
 - * **volume rendering**
 - * projected semi-transparent representation
 - * *scalar field + color transfer function (R,G,B,α) -> pixels*
 - * image space algorithm
 - * object space algorithm

image space

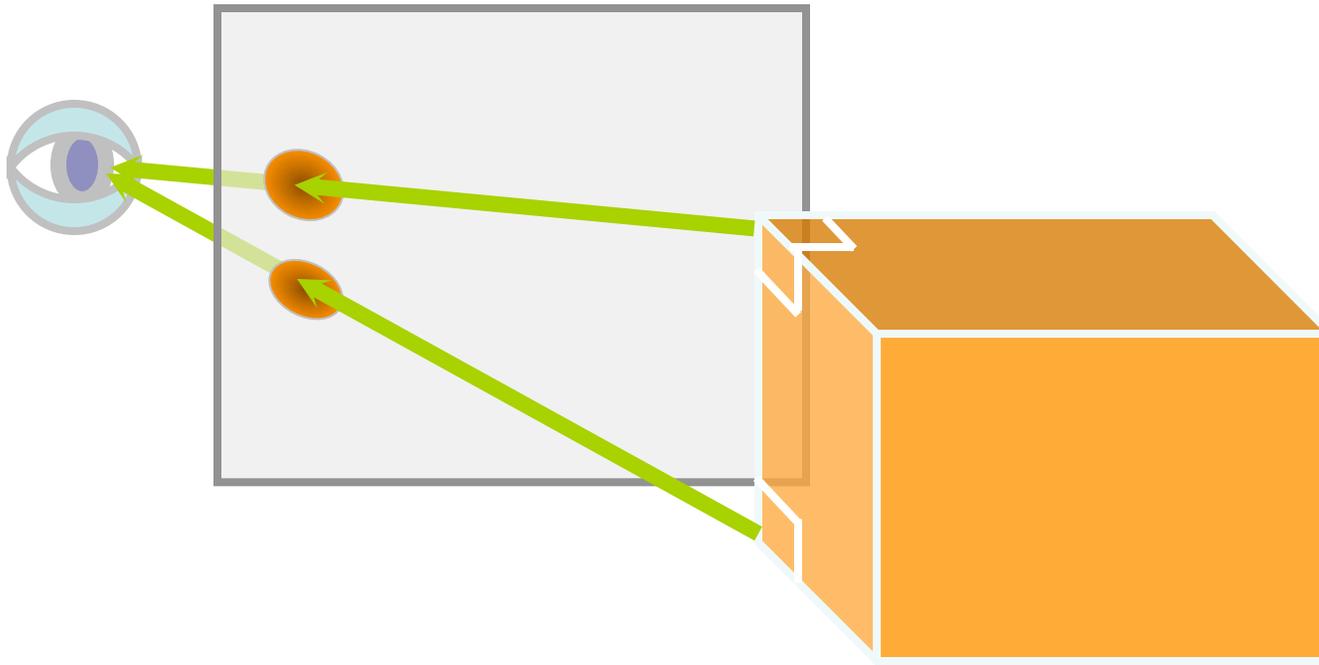
* feed backward: ray casting

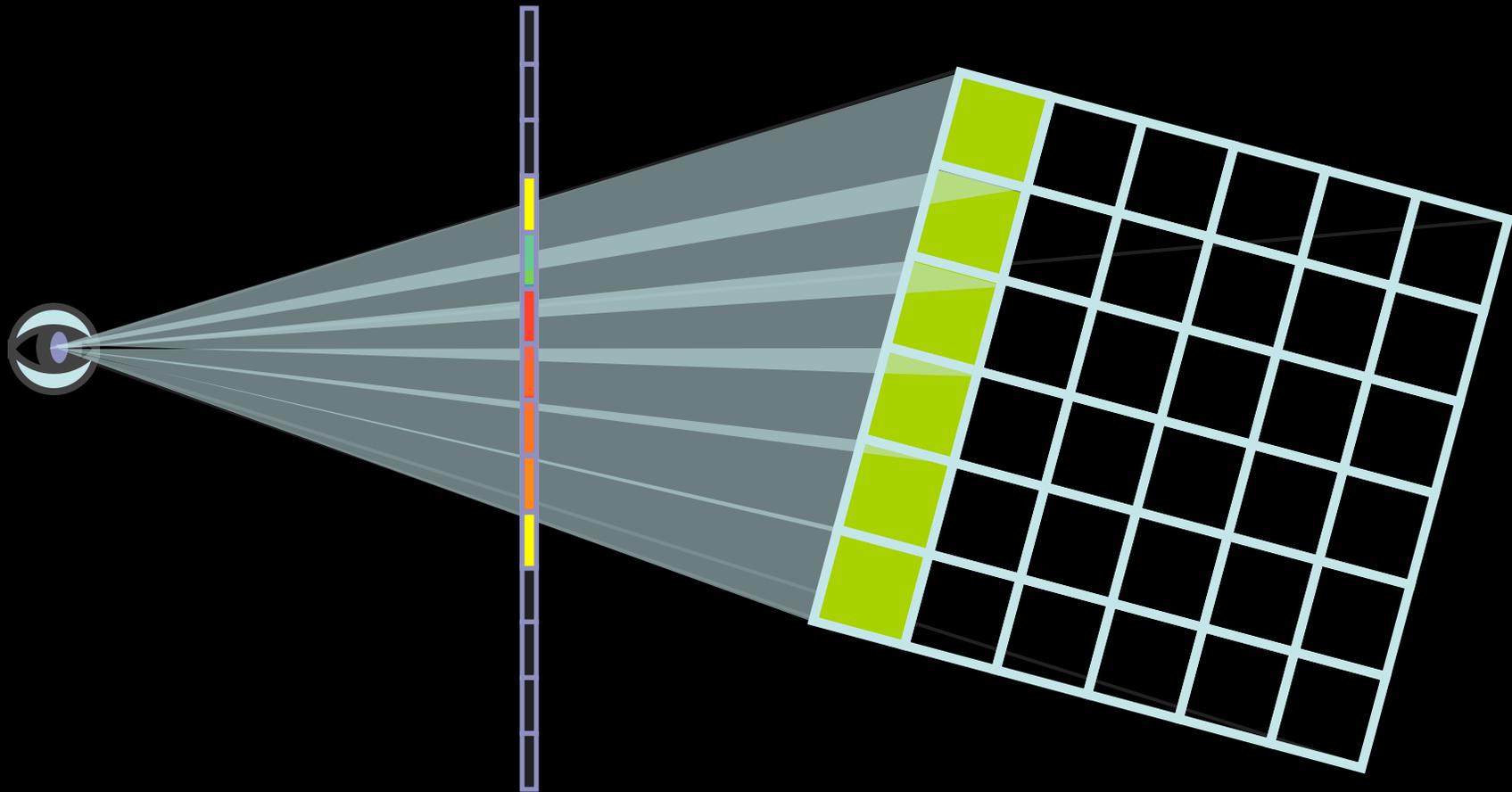




object space

* feed forward: **splatting**

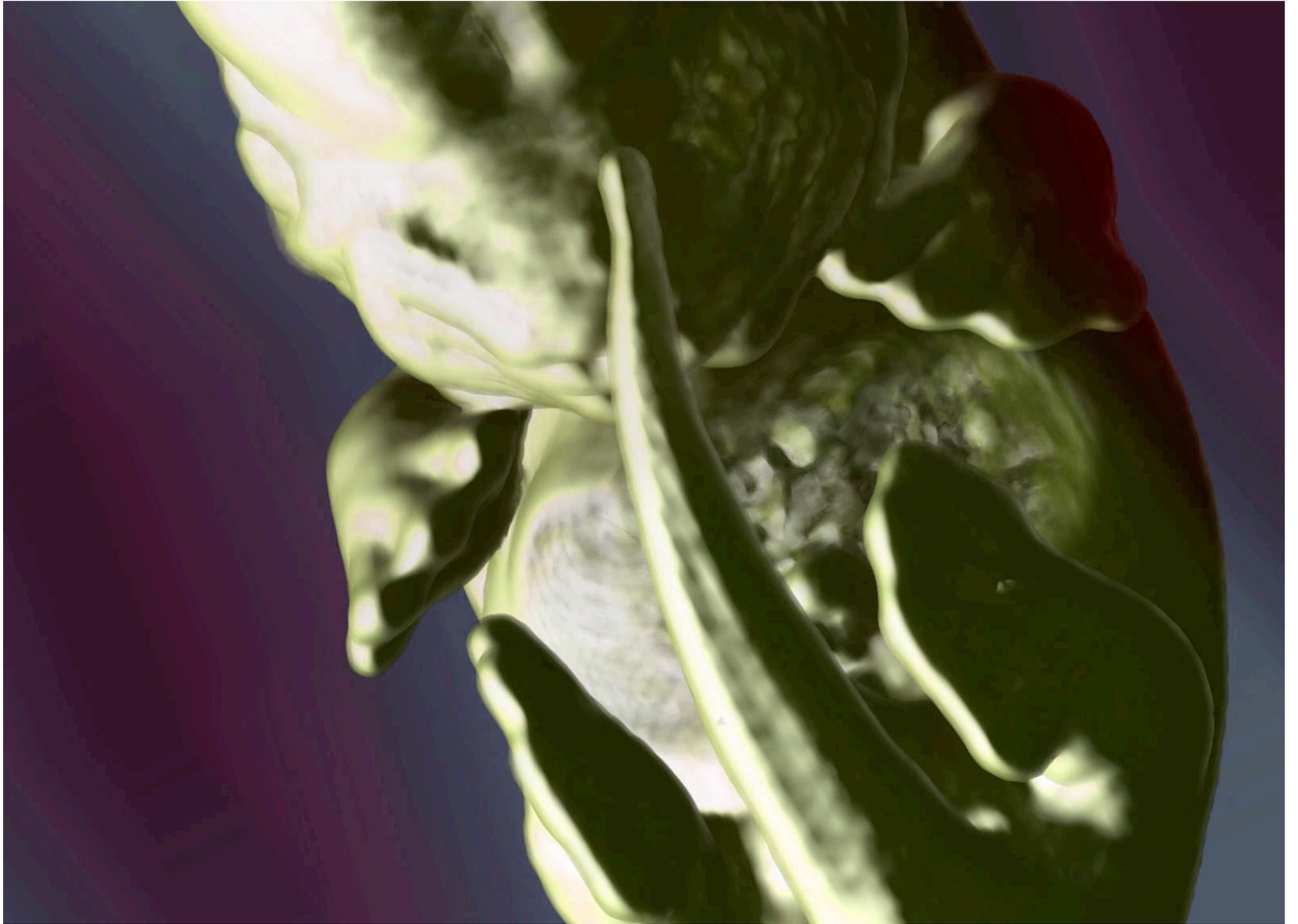




dealing with fuzziness

- * hard to detect on “cloudy” images
 - * better color transfer functions?
 - * gradient based enhancing?
- * **isosurfaces!**
 - * = iso contour in 3d
 - * easy to construct
 - * (also possible with volume rendering)
 - * *triangle throughput vs texture memory*

quick demo:
VOLUME RENDERING



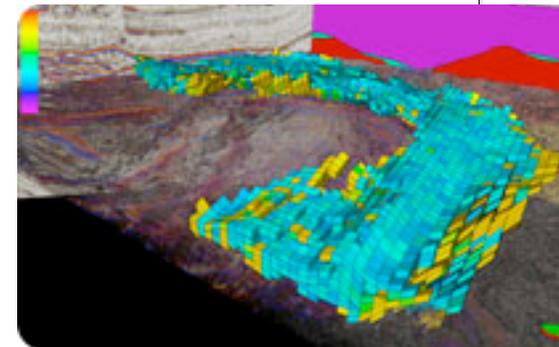
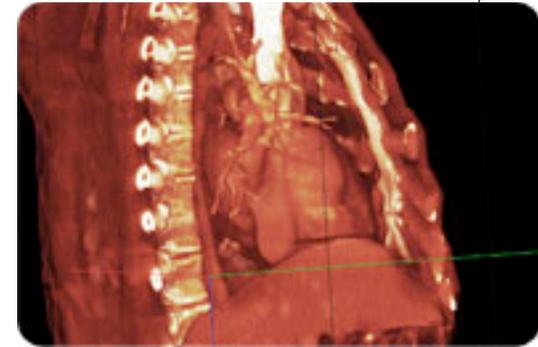
why volume visualization?

- * exploration of full 3D data
- * stored in texture memory (GPU)
 - * also specialized hardware
- * surfaces can be generated
- * modeling through level-set operations

why not volume rendering?

- * semi-transparent volumes not intuitive
 - * hard to visually find features
 - * less visual cues
- * sw not as stable as for triangle meshes
- * not all hw supports volumetric data
- * not always possible to embed meshes
- * harder to find good AMR support

- * nvidia quadro fx 5800
- * 240 processors on board
- * 4GB of memory
- * memory bandwidth 102GB/s
- * 300,000,000 triangles/s
- * 10M x 30 times /sec



state of the art

usual visualization "engine"

interacting your data efficiently...

data

*processing
algorithms*

*geometry
generation*

rendering

ui

interaction



addressing the *UI "bottleneck"*

- * get better "U"s
- * create better "I"s
 - * borrow (steal) good ideas!
 - * create more intuitive interaction
 - * minimize/avoid clutter
 - * mind what is important
 - * visual "read" order
 - * iterate/refine your design

advanced topics:
SKETCH BASED INTERACTION

sketch based interfaces

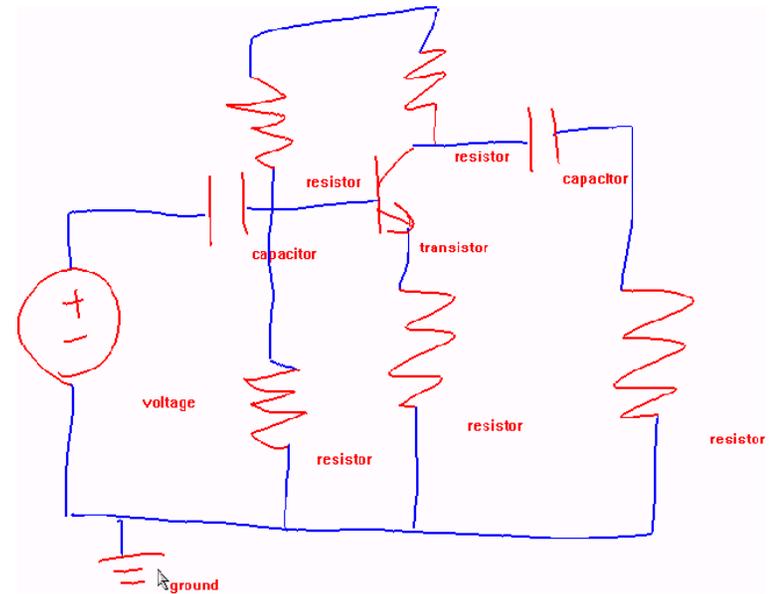
* buttons/menus -> sketch lines

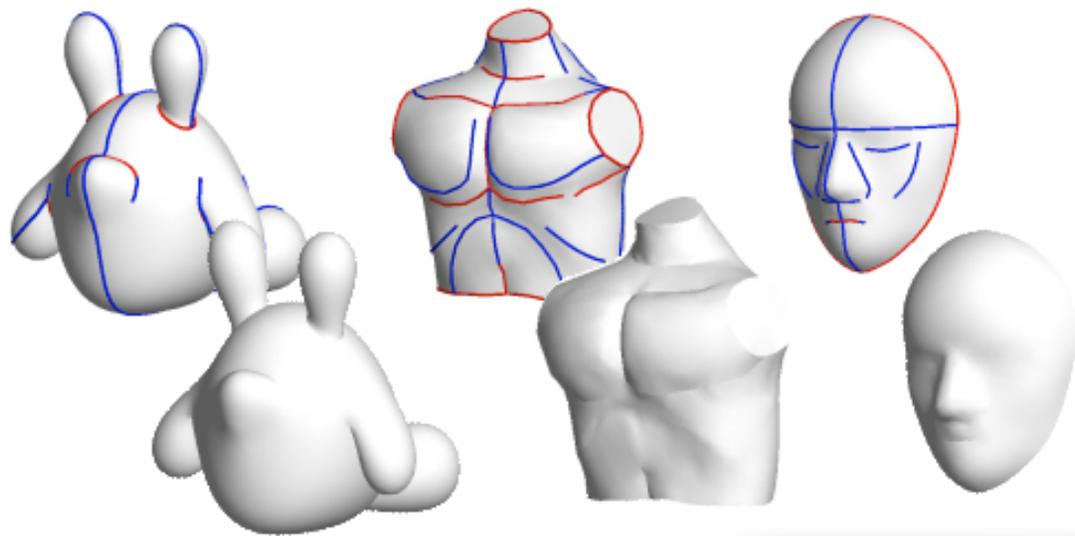
* i.e. direct interaction!

* more intuitive

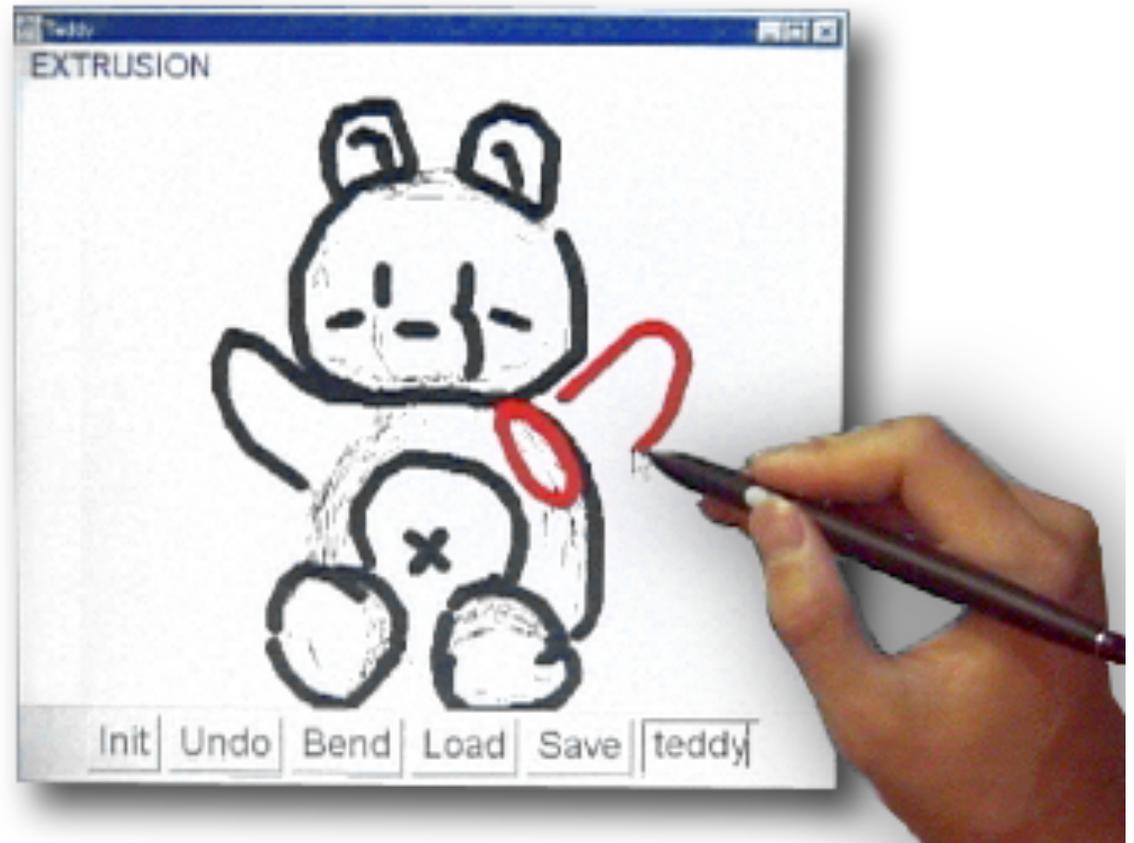
* more efficient

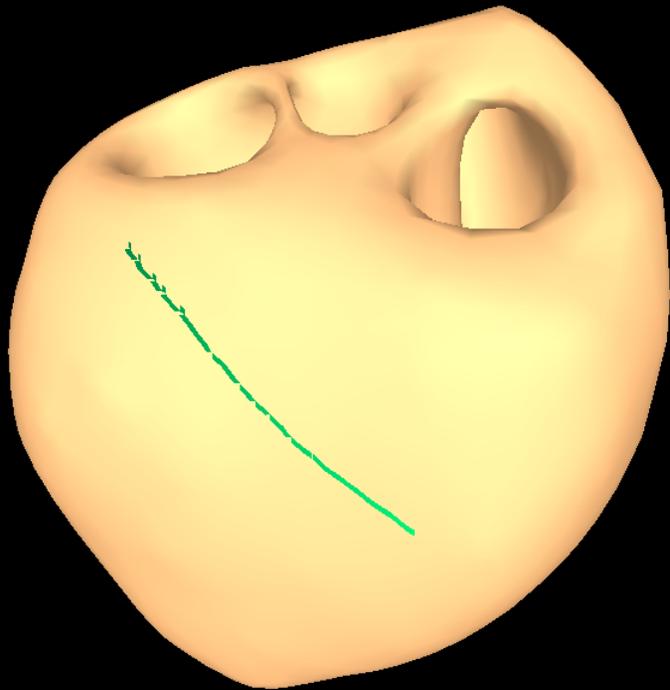
* more elegant



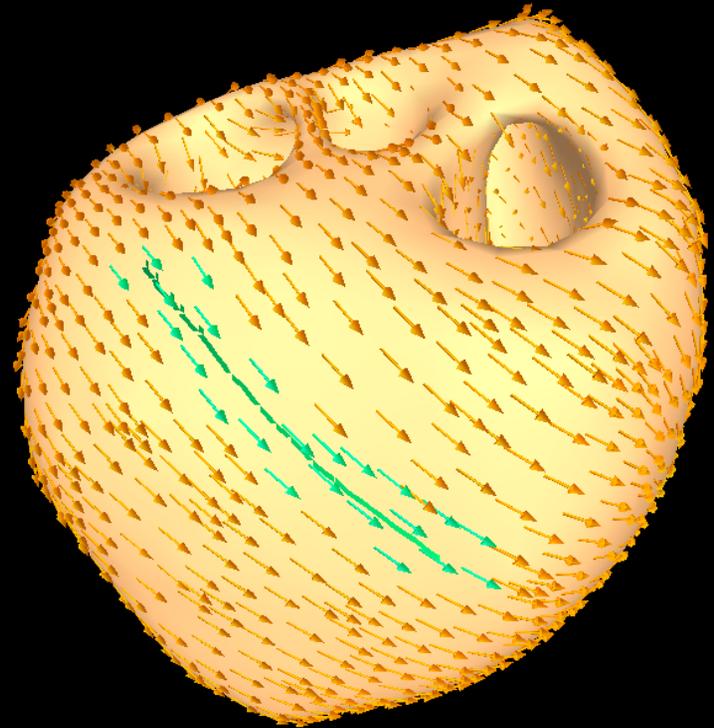
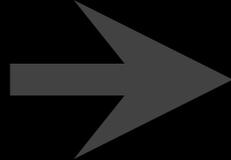


takeo igarashi

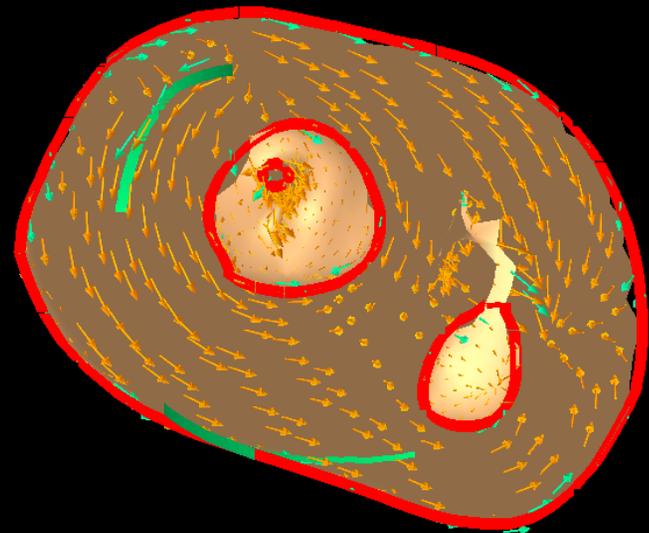
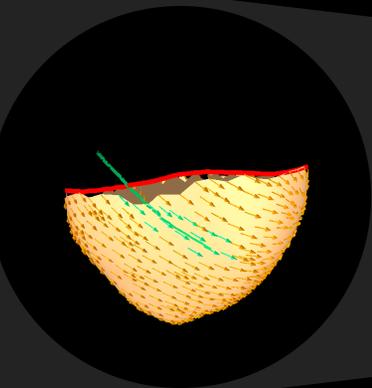
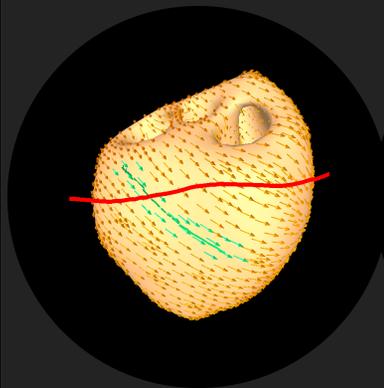




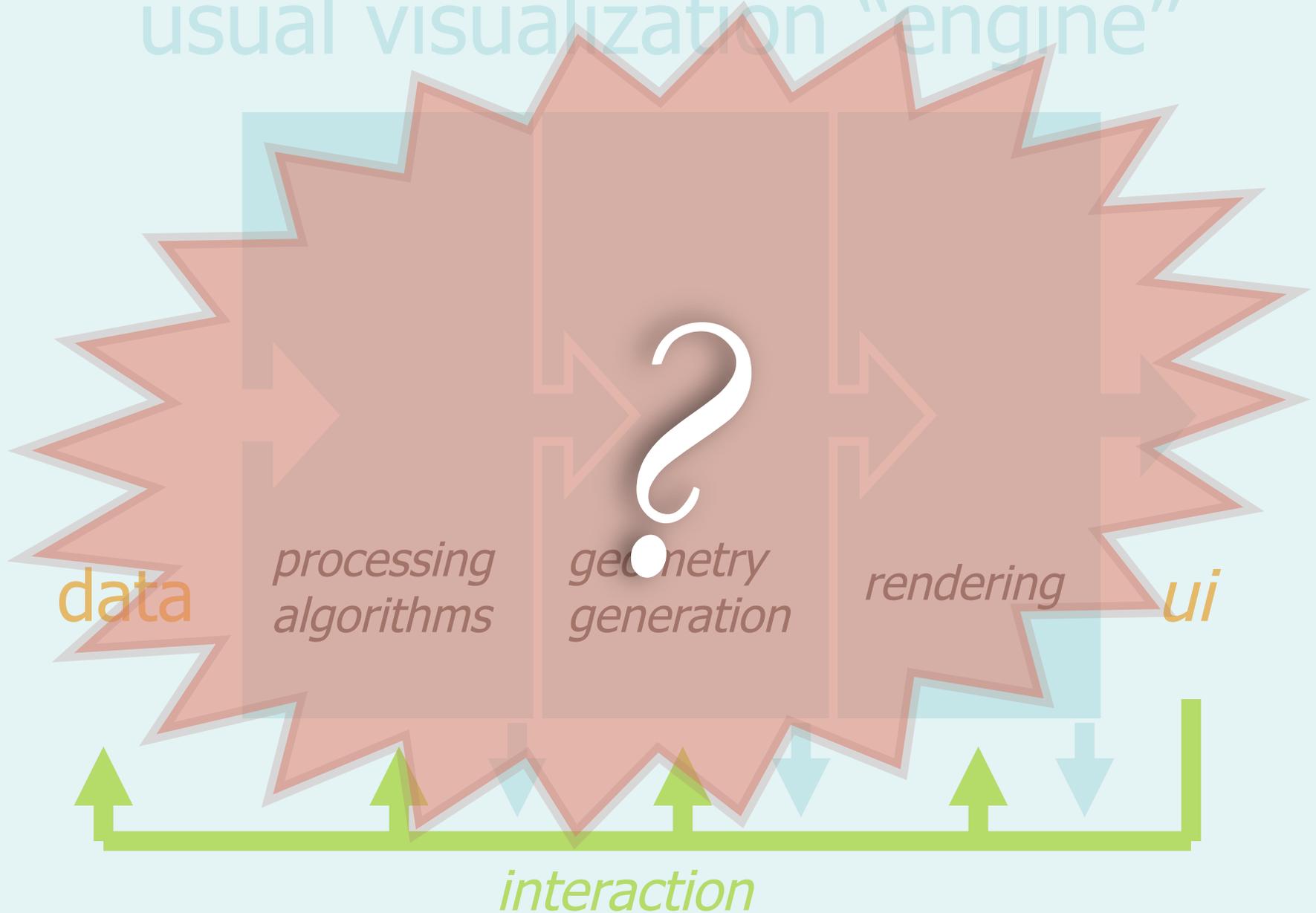
SPECIFY FIBER
ORIENTATIONS ON
THE SURFACE



MYOCARDIAL FIBER ORIENTATIONS FOR
ELECTROPHYSIOLOGICAL SIMULATION OF THE HEART 



usual visualization "engine"

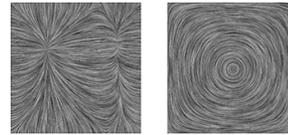
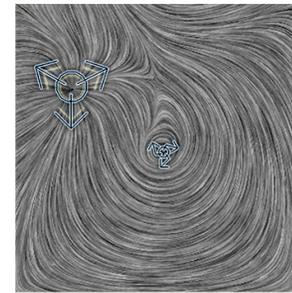
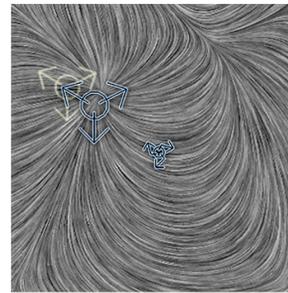
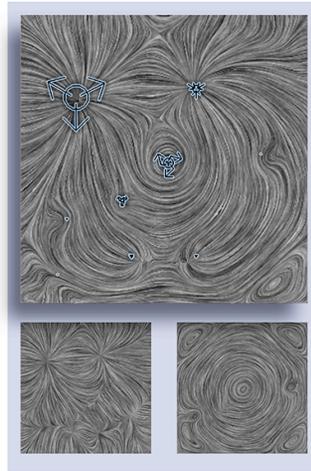


advanced topics: *closer look:*
VECTOR VISUALIZATION

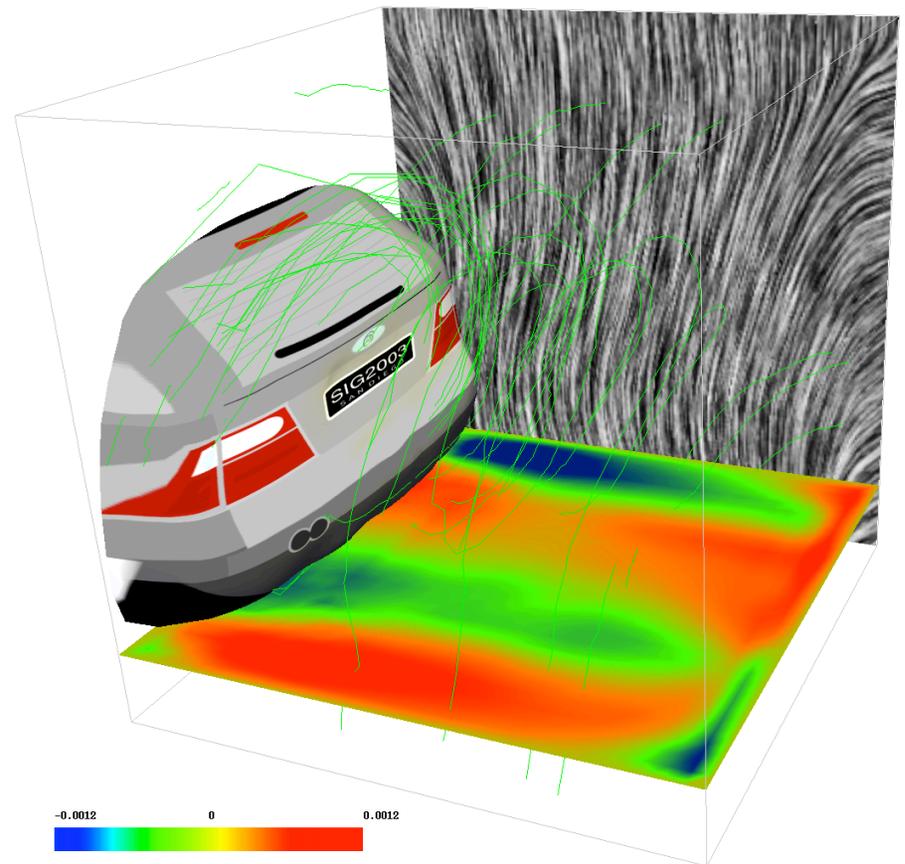
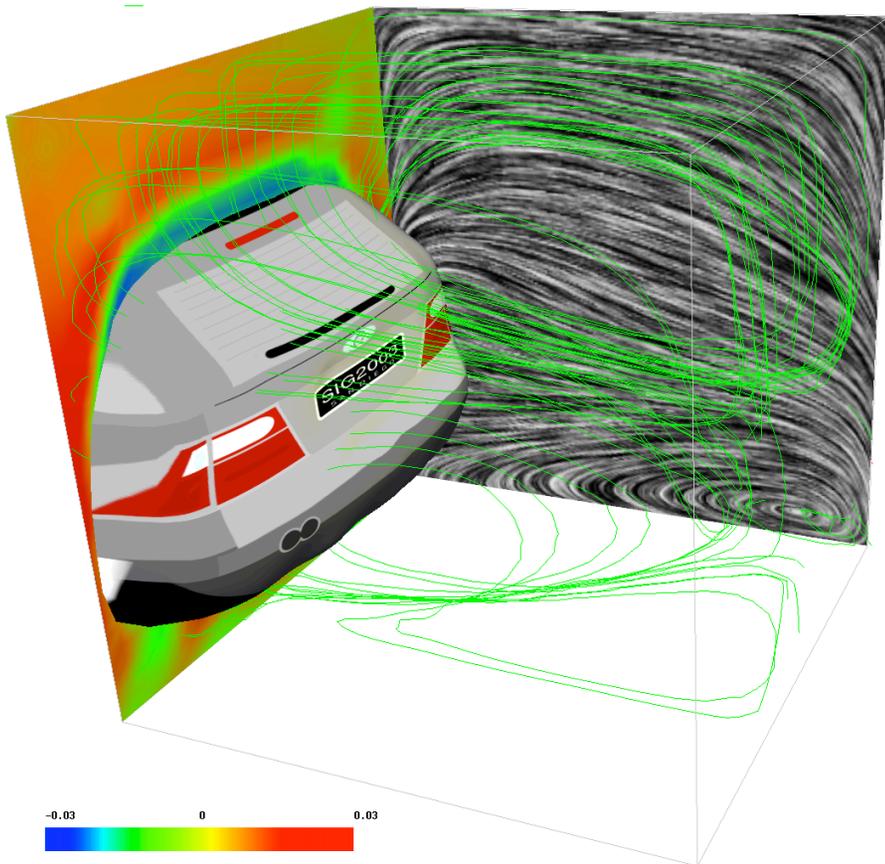
3d vector visualization

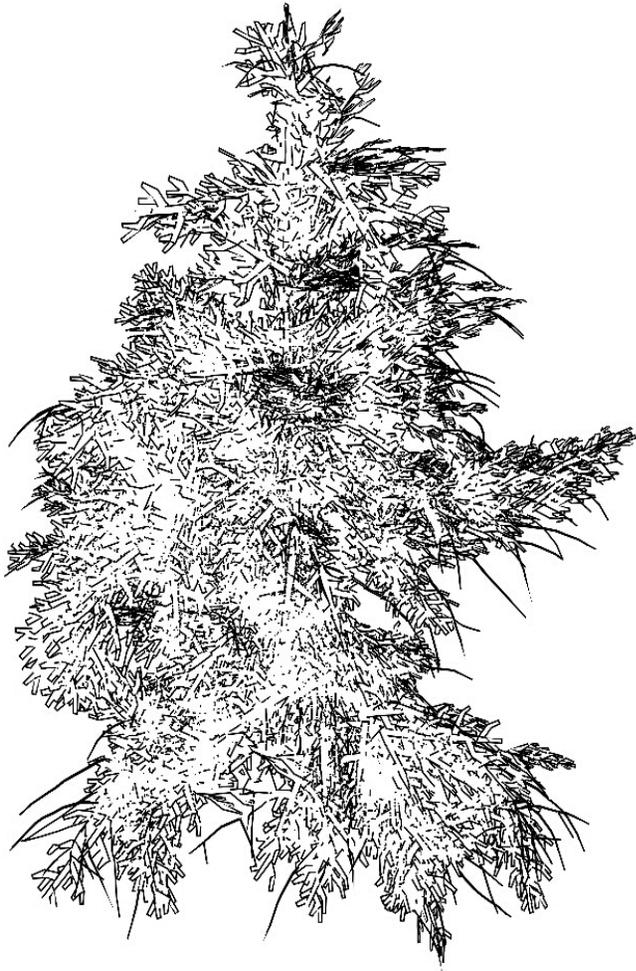
- * unsolved problem
 - * occlusion problems vs detail
 - * intuitiveness
 - * are there good glyphs?
 - * coherency under time changes
 - * inspection techniques
- * possible good solutions
 - * curl/divergence decomposition?
 - * lagrangian coherent structures?
 - * hair or simplified line drawing techniques?

DECOMPOSED VECTOR FIELD EDITING



STREAMLINES ON DIVERGENCE FREE FIELD

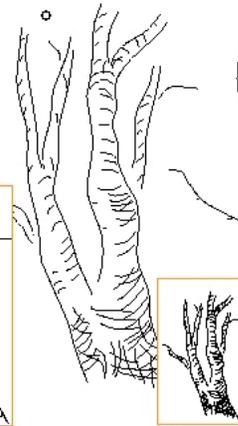




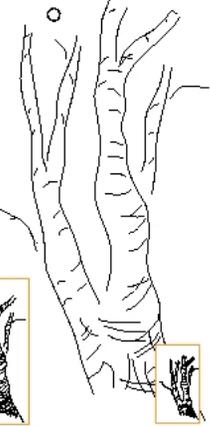
376 input lines



269 clusters



134 clusters

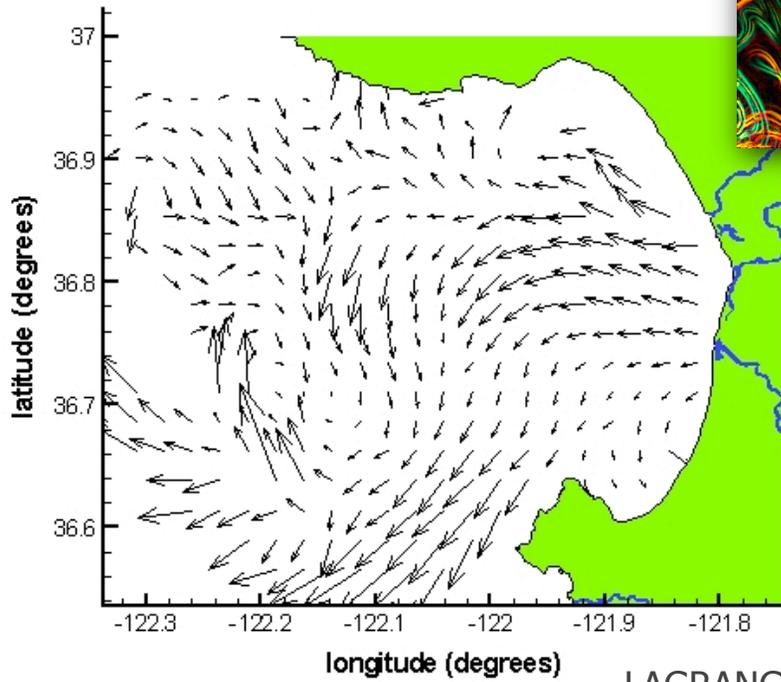


81 clusters

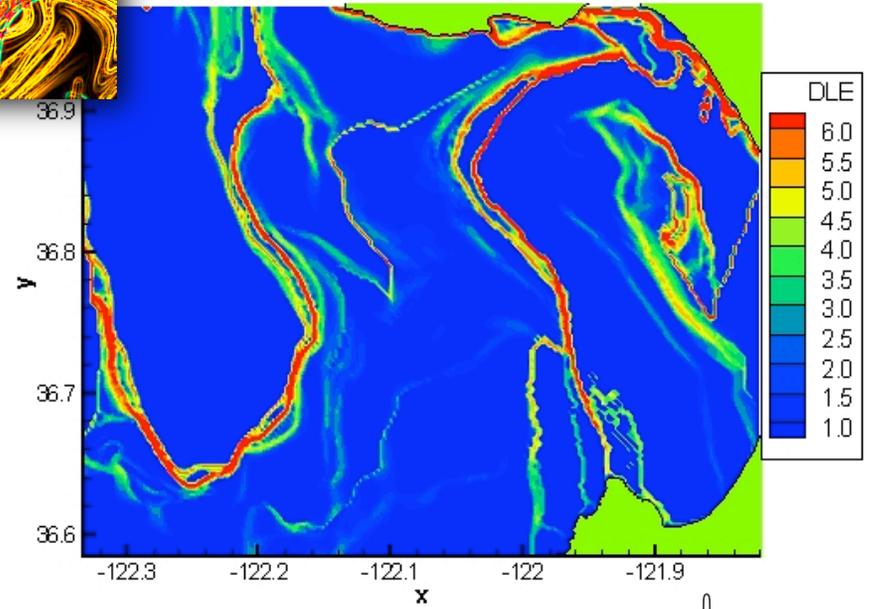
↑ LINEDRAWING SIMPLIFICATION ↓



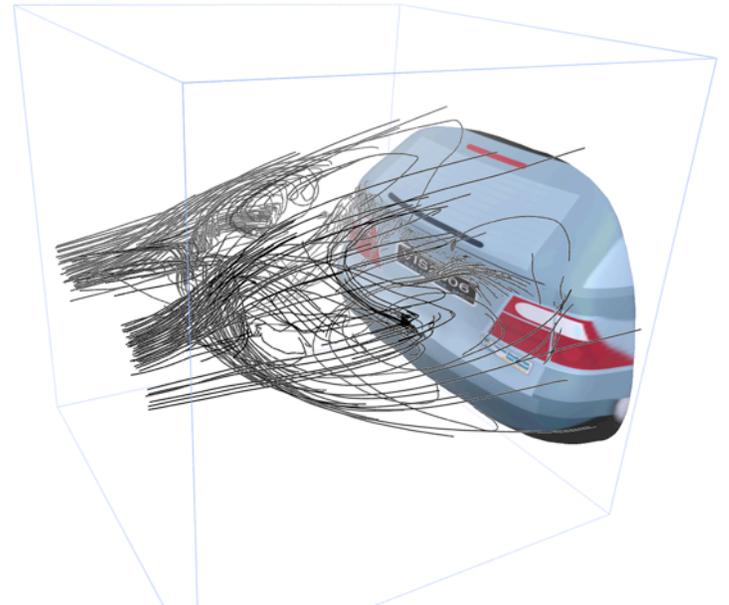
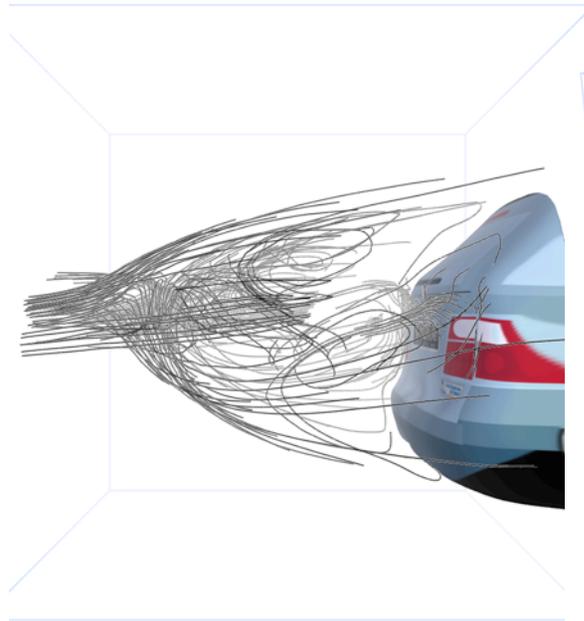
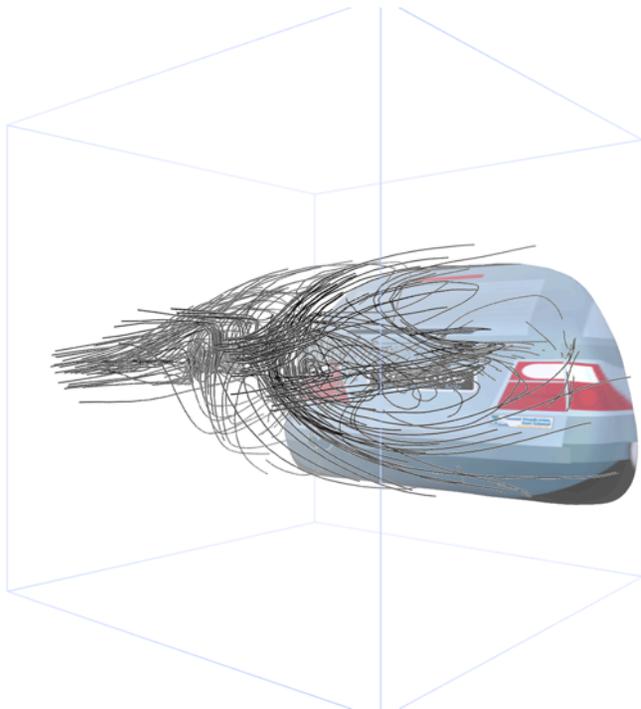
08-18-2003 11:00:00 GMT



08-12-2003 19:00:00 GMT



LAGRANGIAN COHERENT STRUCTURES ON COASTAL EDDY 



usual visualization "engine"



we be done!

interaction

exercises

- * dig through *ieee xplore/ acm* library
 - * look for **one** interesting paper in
 - * **siggraph** (conference on graphics)
 - * **vis conference**
 - * **infoviz contest**
 - * **vast challenge**
- * consider:
 - * is a similar result doable with off-the-shelf tools?
 - * could you apply it to “your” data?
- * post the result on the blog in the form:

check out this paper [\[link\]](#). i was wondering, wouldn't it be interesting to do something like that with my [...] data? huh? huh?

 - * *post a image from the paper if possible...*

thanks!

data mining coming next... stayed tuned

avyakta.caltech.edu:8888/esci101

methods of computational science

visualization

day iii – case studies + advanced viz

santiago v lombeyda

center for advanced computing research

caltech