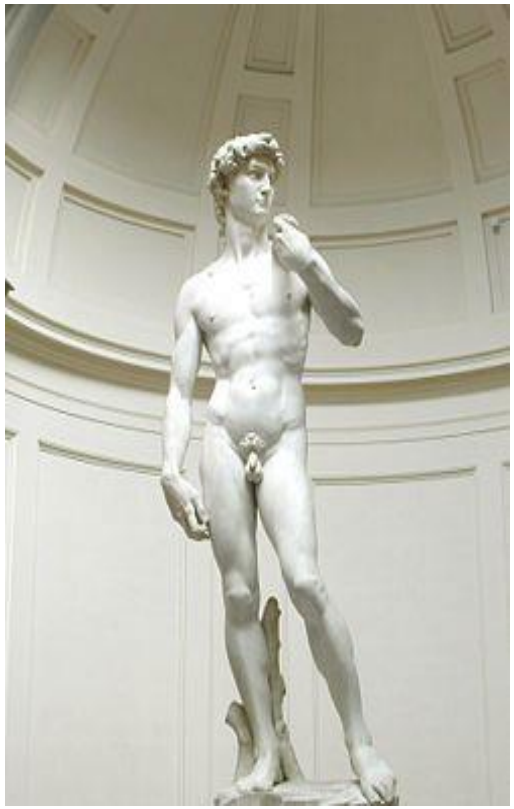
Acquisition

Umberto Castellani

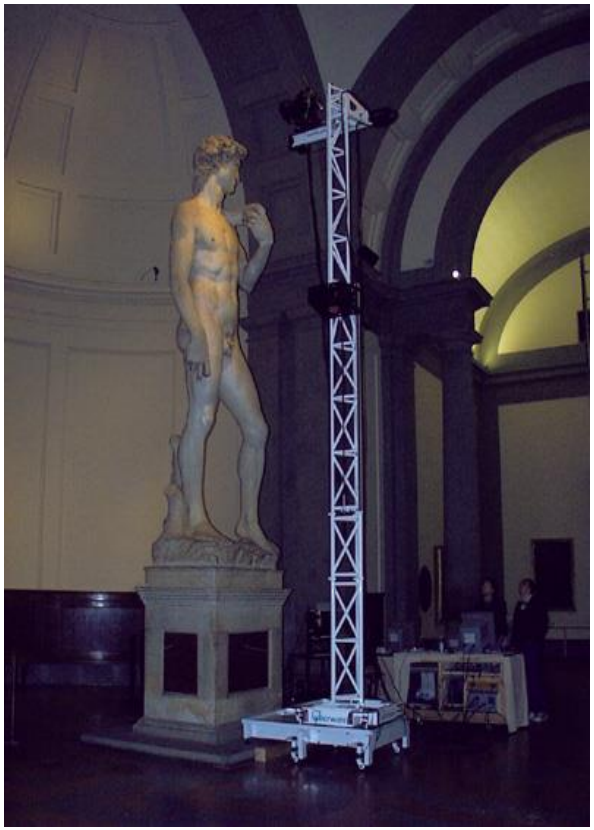
Robotics, vision and control

3D modelling from reality pipeline

Overall aim



Real Object



3D Scanning

Virtual Model

https://youtu.be/r1e088hGee4

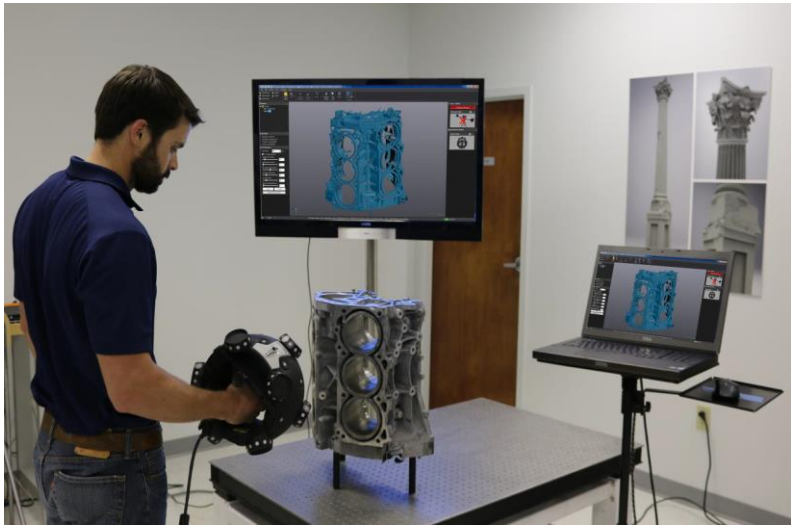
Overall aim

Geometry → Dynamics (Motion, Deformation, etc.) → Rendering (Illumination Model)

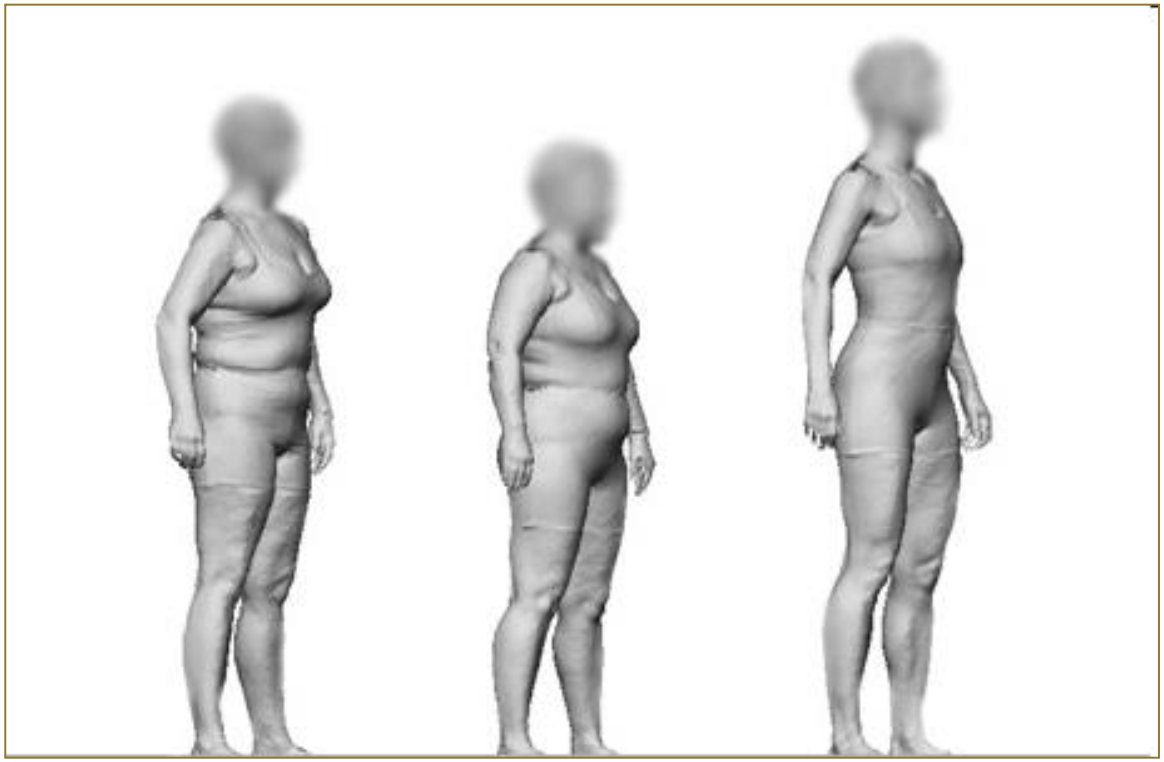
3D Scanning

Overall aim

- Create a **virtual representation** of the real word through the **observation** of the real word
  - Geometric properties,
  - Dynamic properties,
  - Photometric propertis,



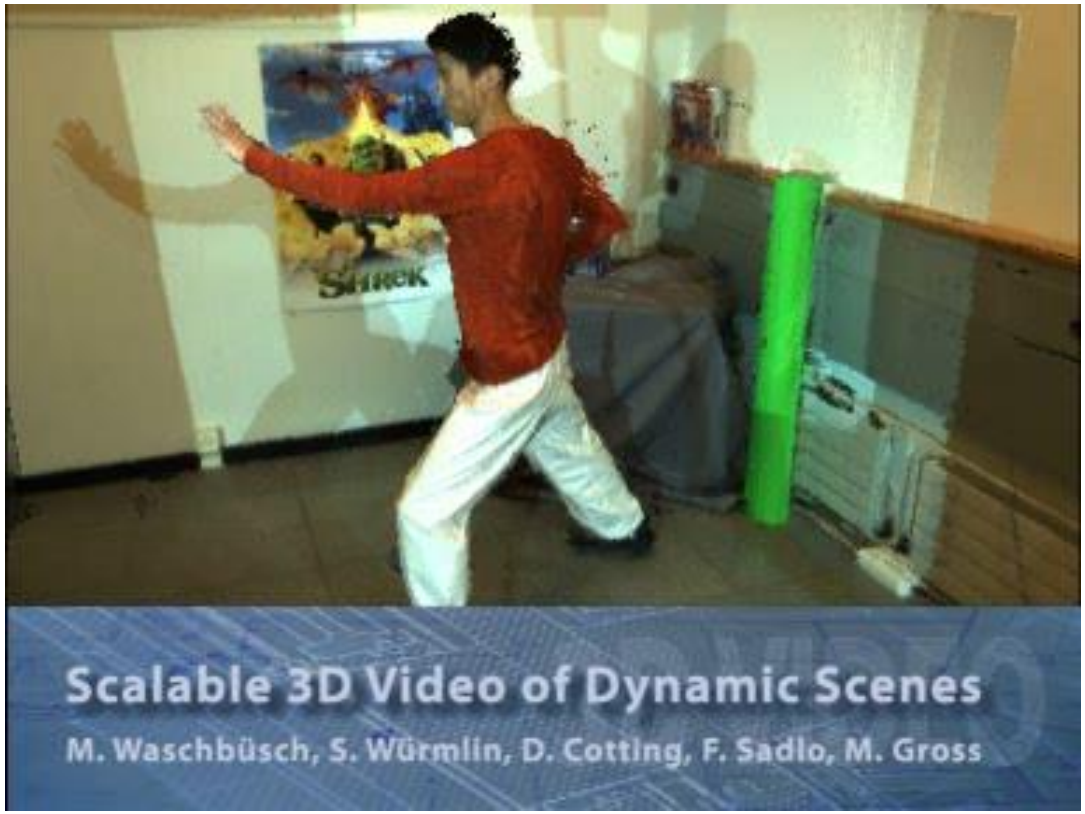
Geometric properties



Fit3D Body Scanner

Body scanner

Dynamic properties



Realtime capture: RGBD Scanning

Photometric properties

Light Dome

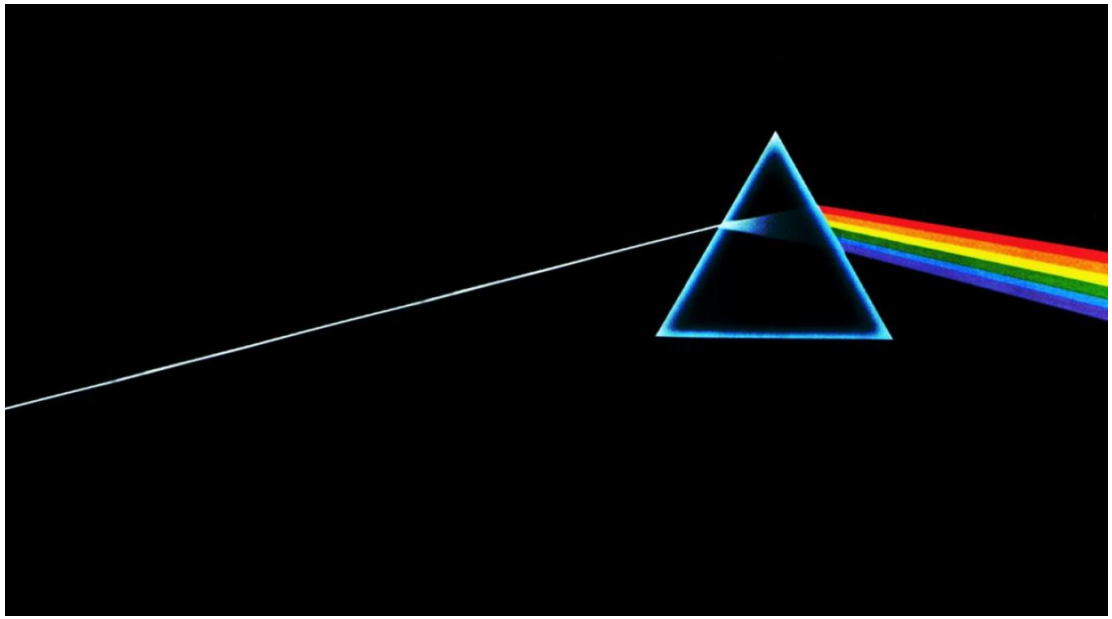
Overal aim

- **Some names**:

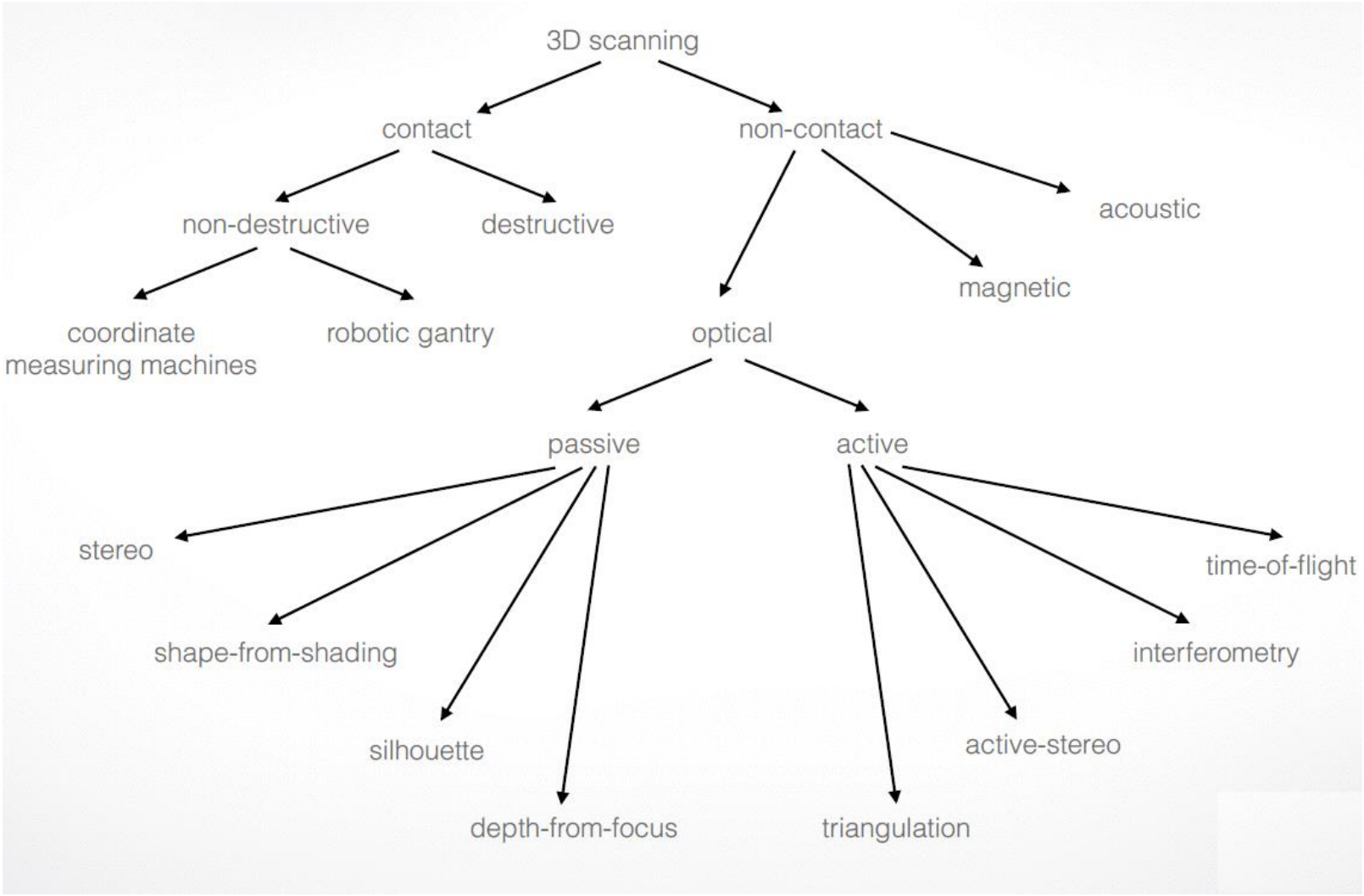
Reality capture, 3D scanning, 3D acquisition, Digital content creation, Range finder, Virtualized reality, RGBD, dynamic scene capture, reverse engineering, photogrammetry, computational photography, 3DTV,...

Principles

- So many acquisition techniques…, physics helps!
  - Each technique is based on a well founded physical principle

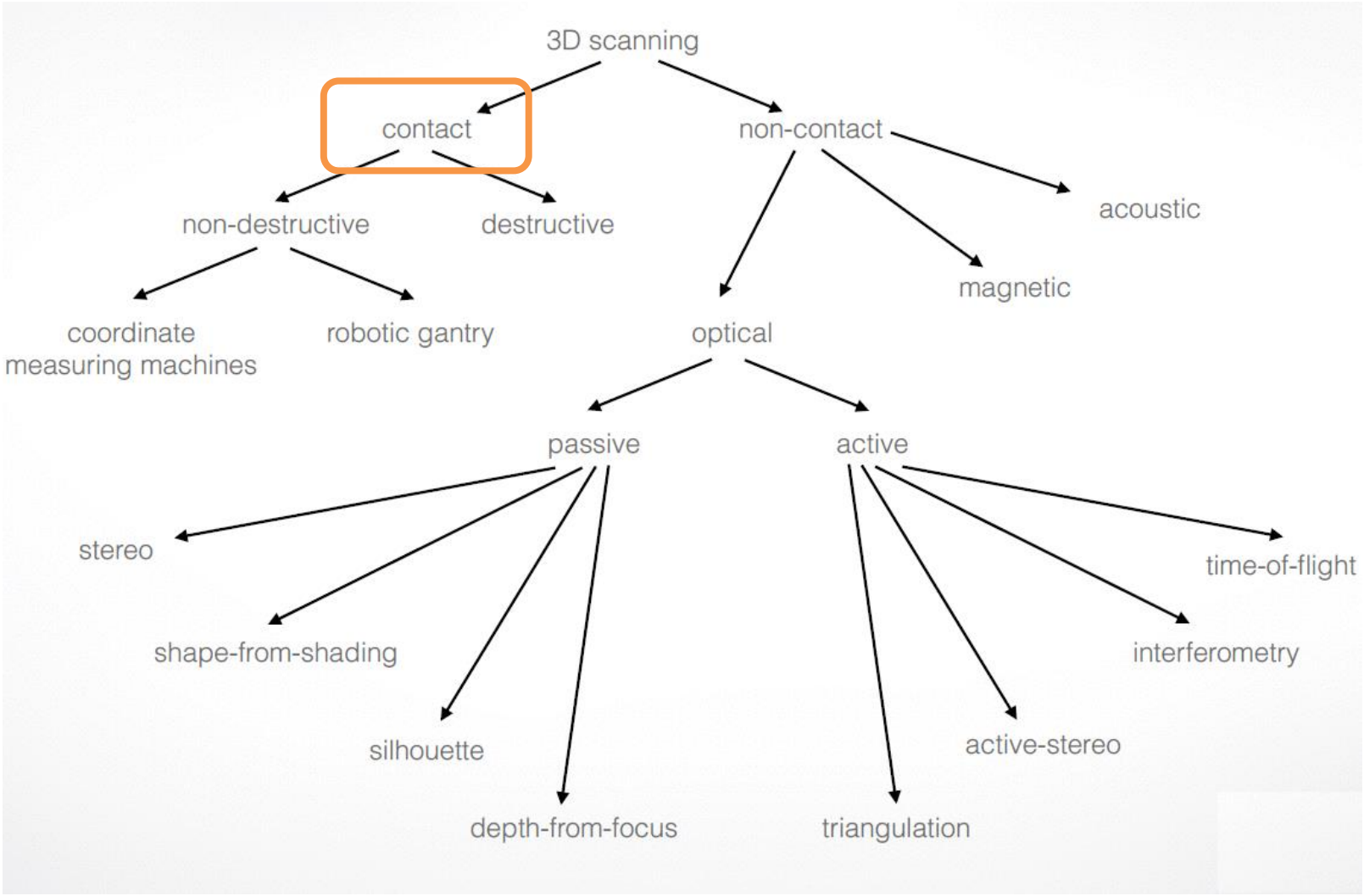


A taxonomy



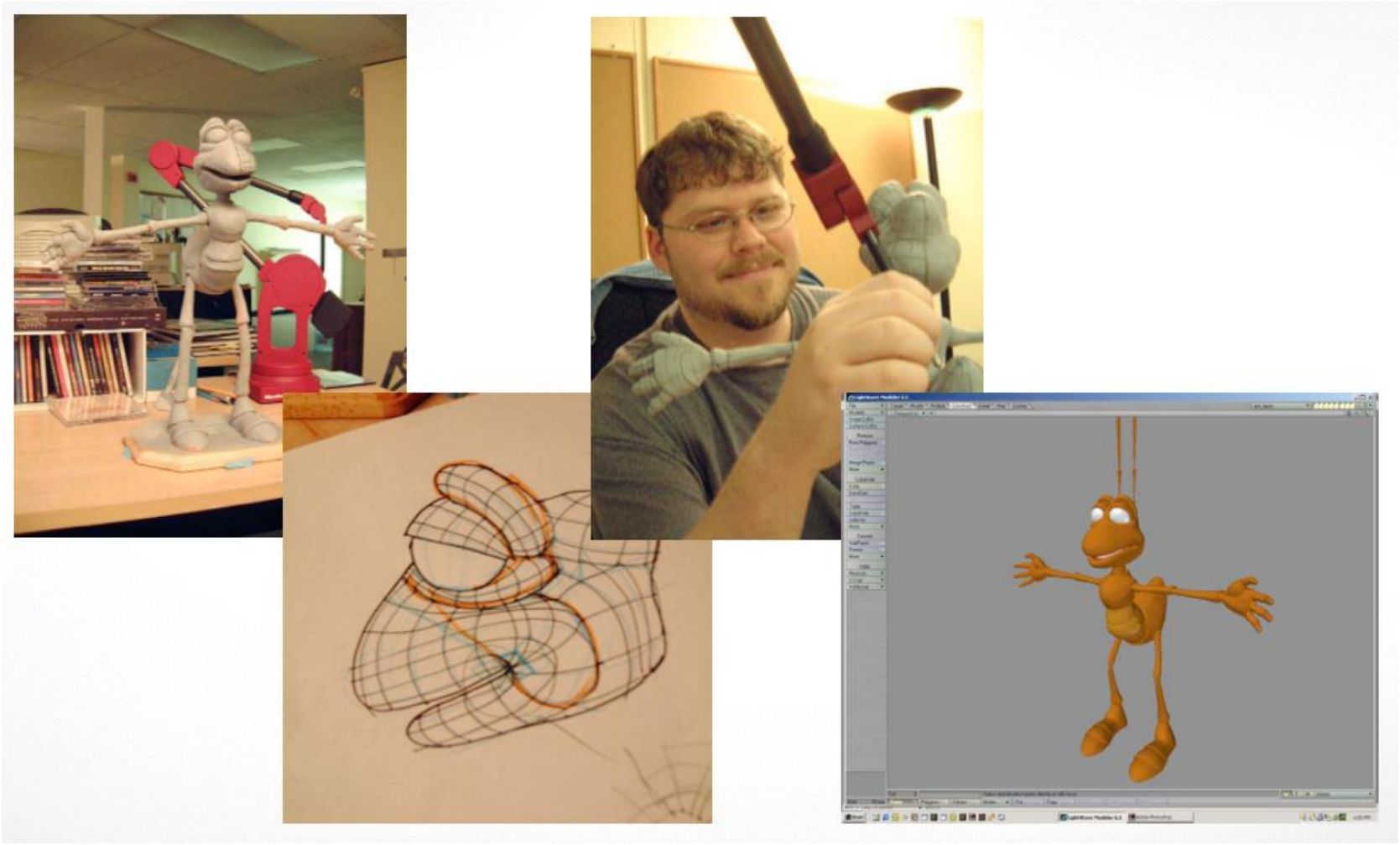
Hao Li http://www.hao-li.com/cs599-ss2015/slides/Lecture04.2.pdf

A taxonomy



Hao Li http://www.hao-li.com/cs599-ss2015/slides/Lecture04.2.pdf

Contact Scanner



[immersion microscribe, magnetic dreams]: http://www.hao-li.com/cs599-ss2015/slides/Lecture04.2.pdf

Contact Scanner

Probe object by physical touch!

- used in manufacturing control,
- highly accurate,
- reflectance independent (transparency!)
- slow scanning, sparse set of samples,
- for rigid and non-fragile objects

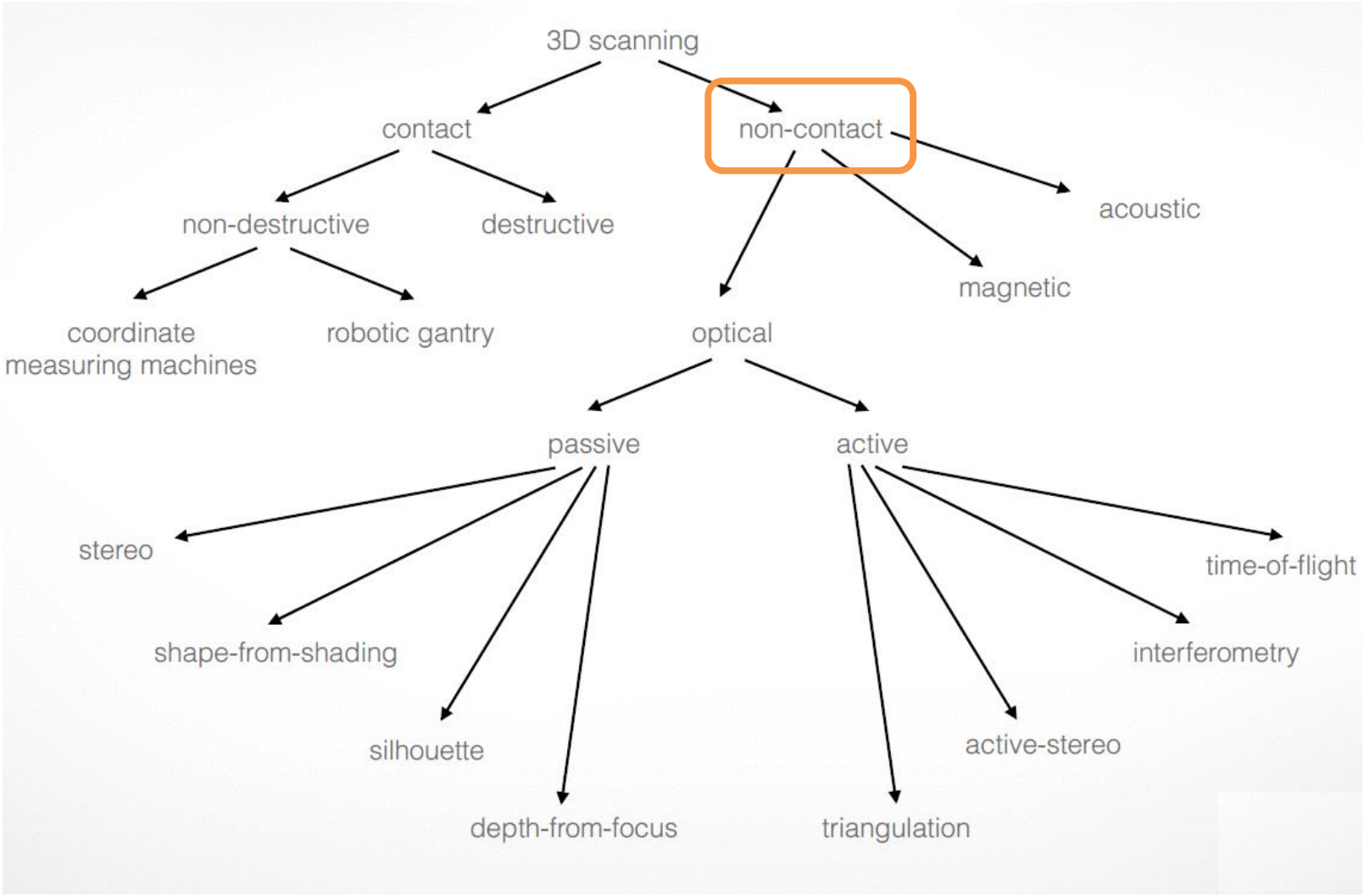
Contact Scanner

Probe object by physical touch!

- hand-held scanners!
- less accurate
- slow scanning, sparse set of samples, it requires experts

http://www.3d-microscribe.com

A taxonomy



Hao Li http://www.hao-li.com/cs599-ss2015/slides/Lecture04.2.pdf

Non-contact Scanner

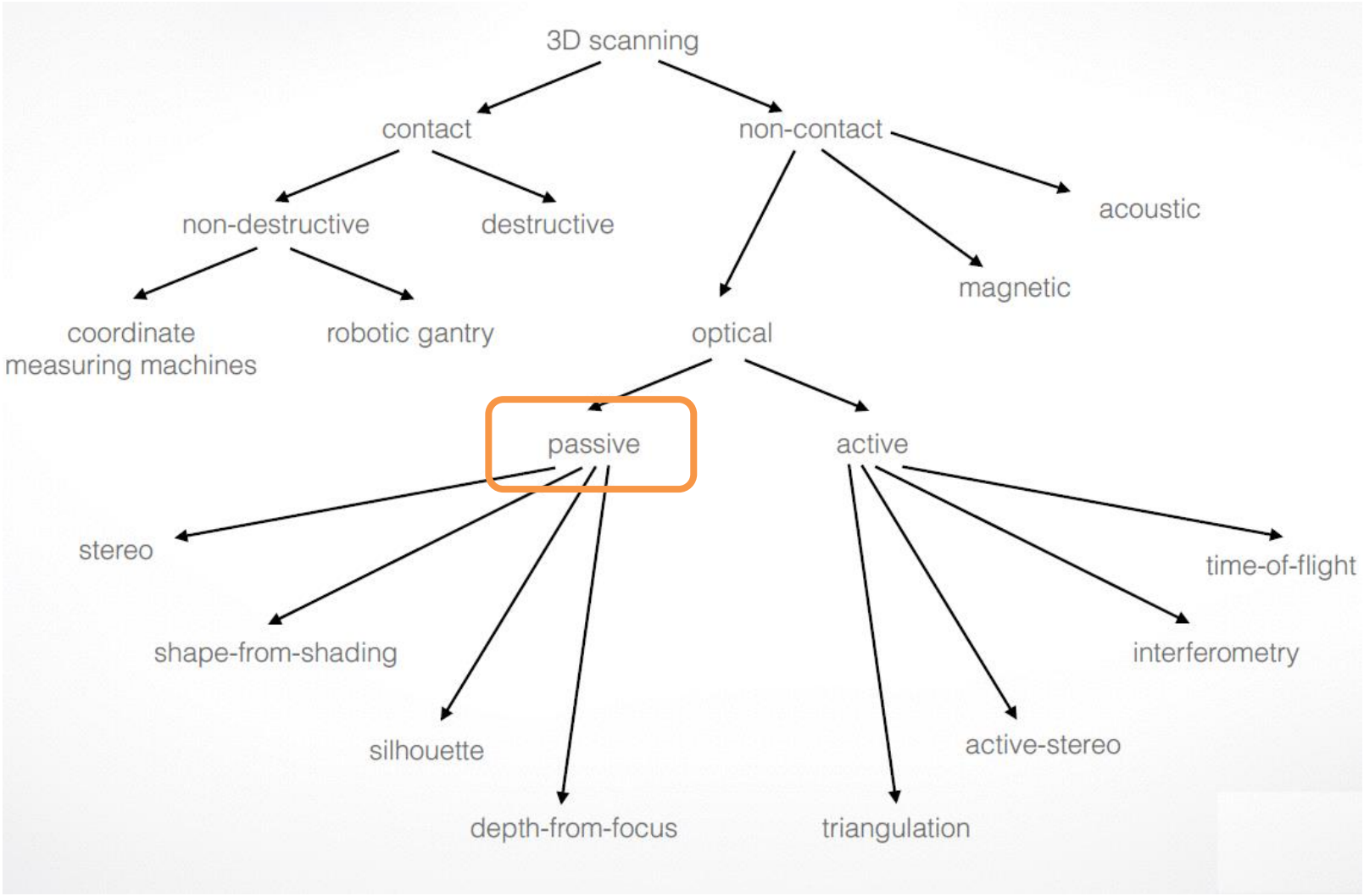
Advantages

- longer and safer distance capture
- potentially faster acquisition
- more automated

Optical Approaches

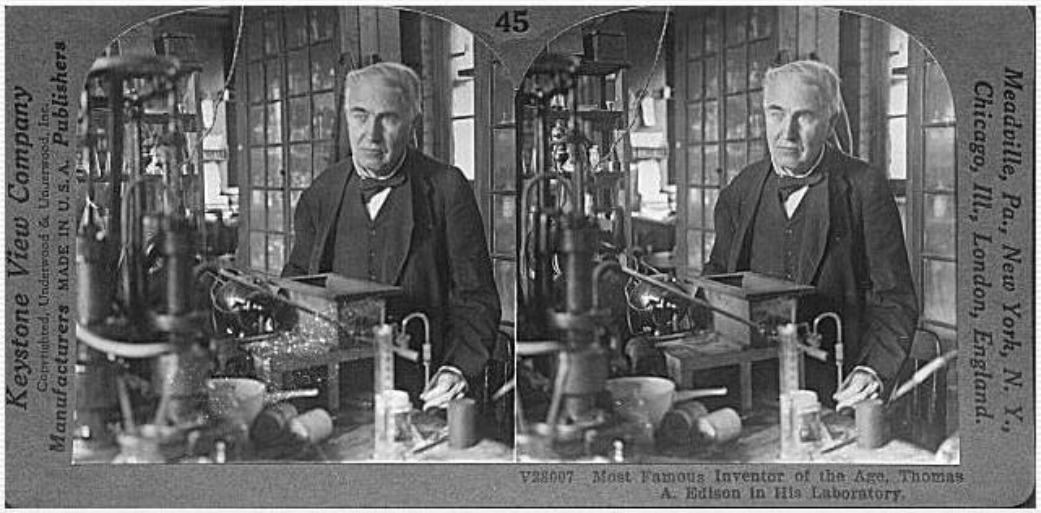
- most relevant and used (no special hardware requirements)
- highly flexible
- most accurate
- **passive** and **active** approaches

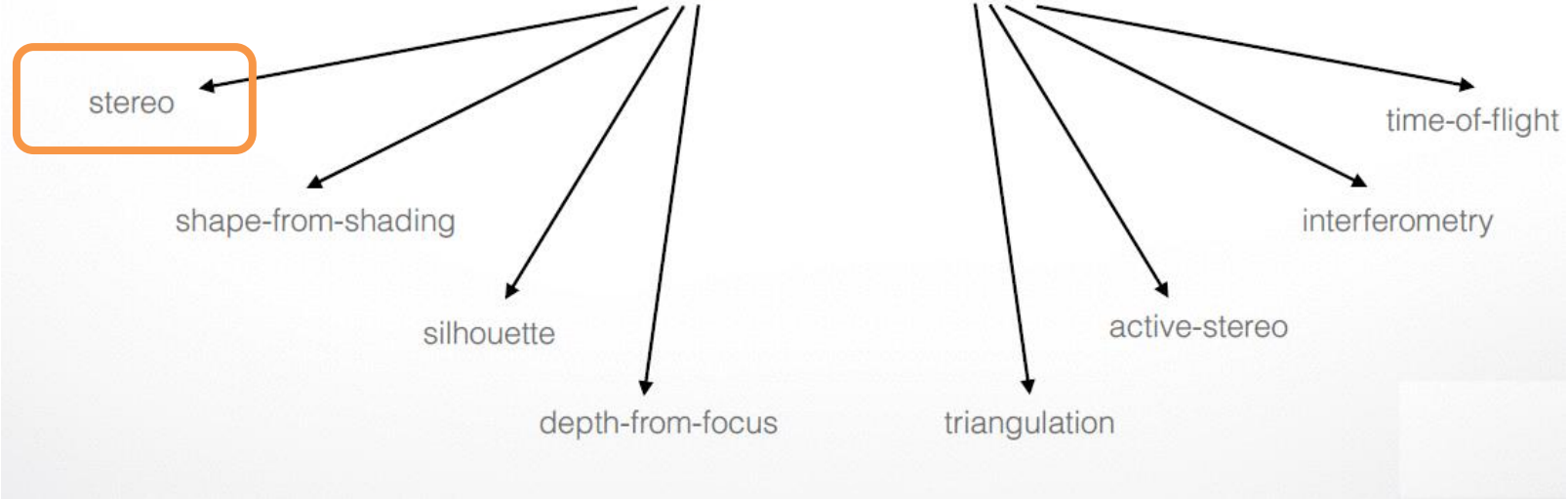
A taxonomy



Hao Li http://www.hao-li.com/cs599-ss2015/slides/Lecture04.2.pdf

A taxonomy



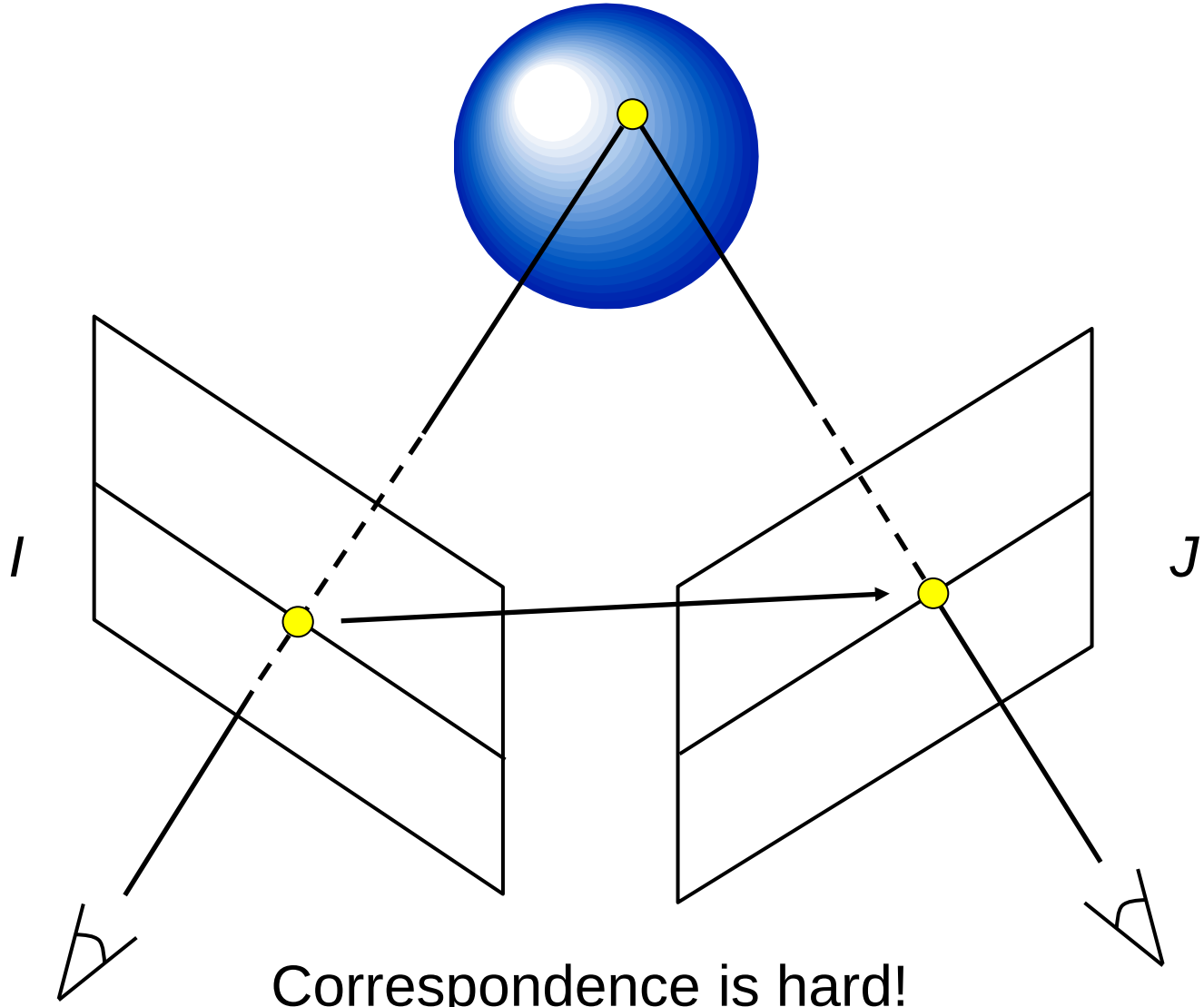


Hao Li http://www.hao-li.com/cs599-ss2015/slides/Lecture04.2.pdf

Passive stereo

- Stereo (two views)
- Multiple views
- Structure and motion

Stereo Triangulation



Correspondence is hard!

Passive stereo

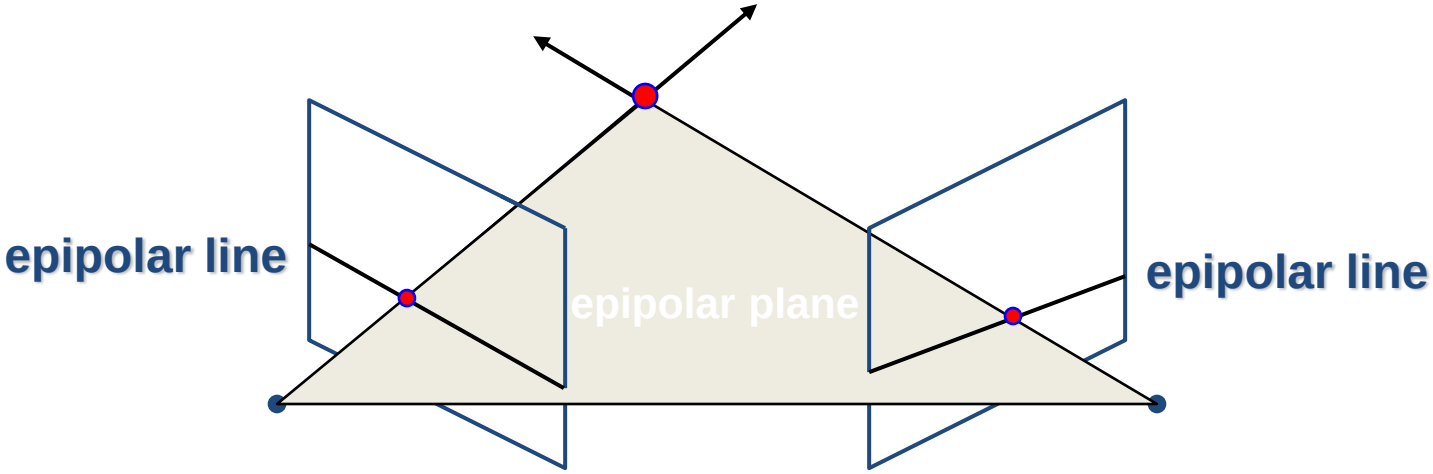
image 1

image 2

Dense depth (or disparity) map

Stereo correspondence

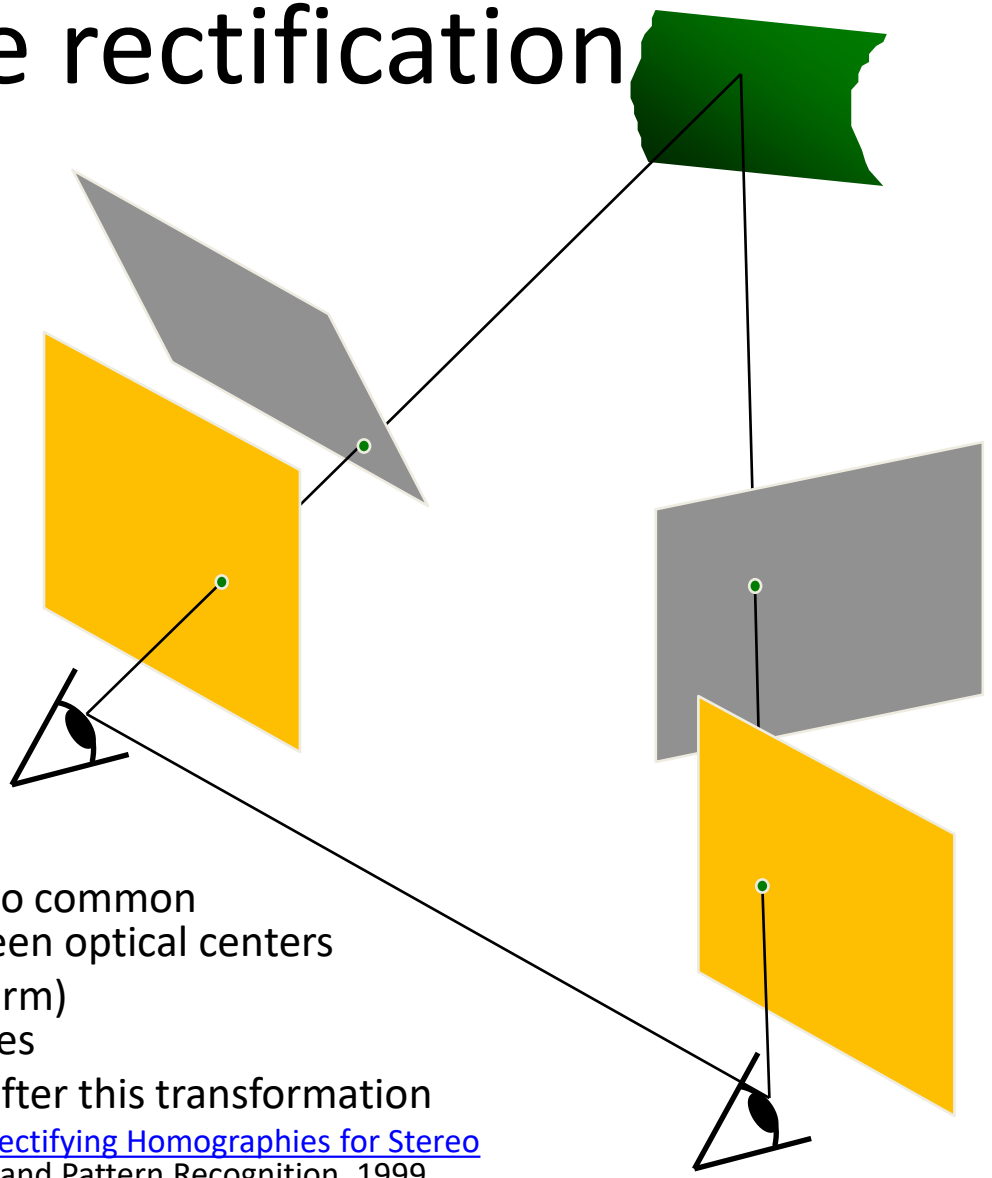
Pairs of points that correspond to same scene point



Epipolar Constraint

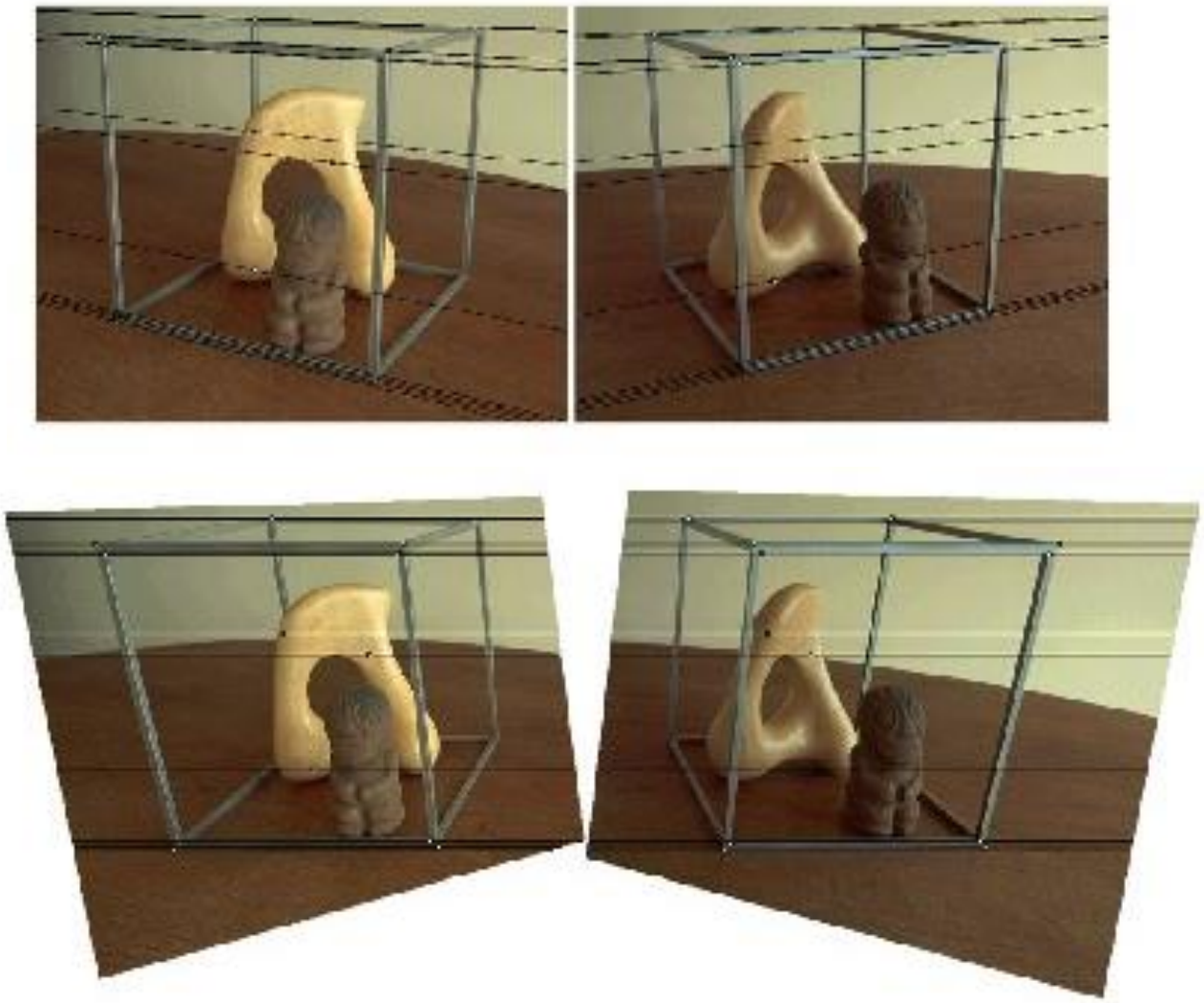
- Reduces correspondence problem to 1D search along *conjugate epipolar lines*

Image rectification

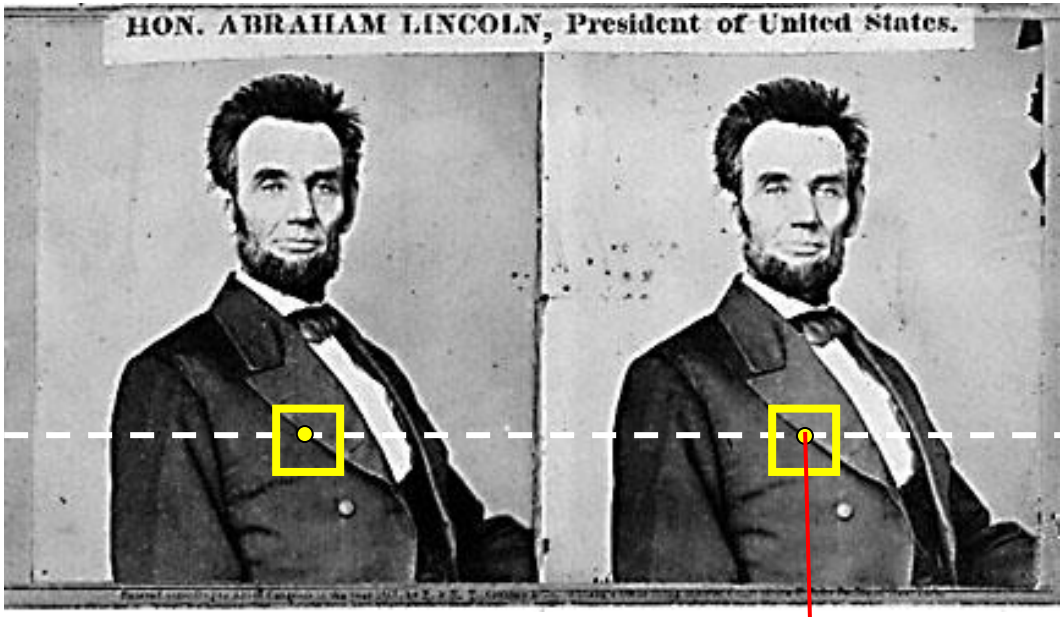


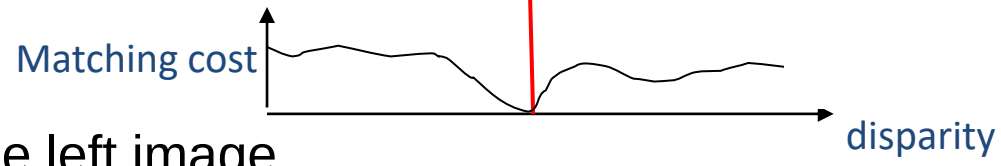
- Image Reprojection
  - reproject image planes onto common plane parallel to line between optical centers
  - a homography (3x3 transform) applied to both input images
  - pixel motion is horizontal after this transformation
  - C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. IEEE Conf. Computer Vision and Pattern Recognition, 1999.

Image rectification



Your basic stereo algorithm





For each epipolar line

For each pixel in the left image

- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

Improvement: match ***windows***

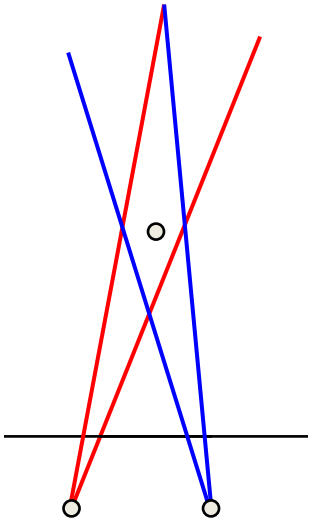
Stereo camera

https://www.stereolabs.com/

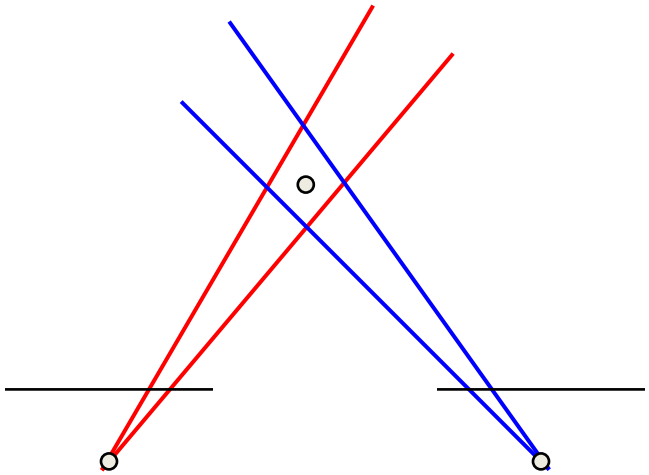
Multiview passive stereo

Larger baseline (i.e., distance between cameras)

The role of the baseline



Small Baseline

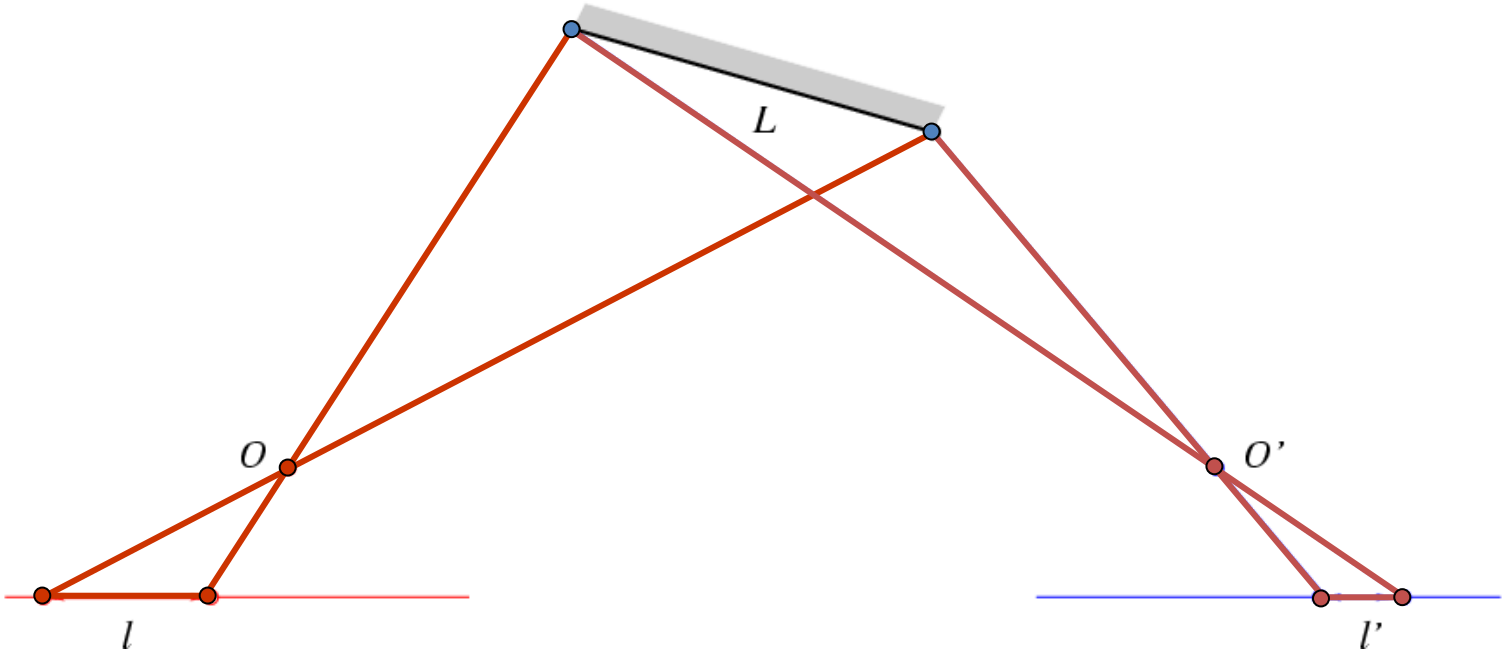


Large Baseline

- Small baseline: large depth error
- Large baseline: difficult search problem

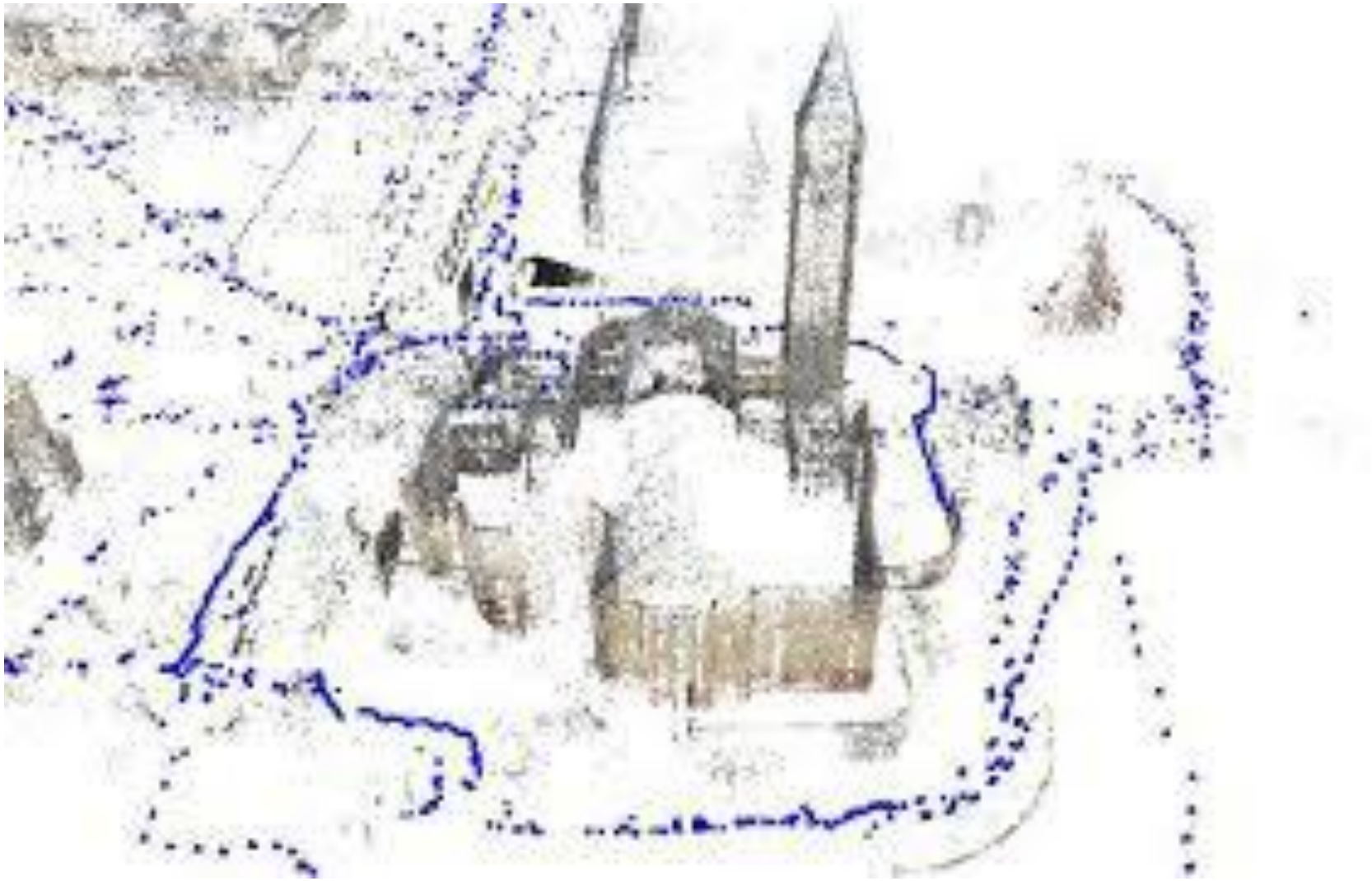
Source: S. Seitz

Problem for wide baselines: Foreshortening

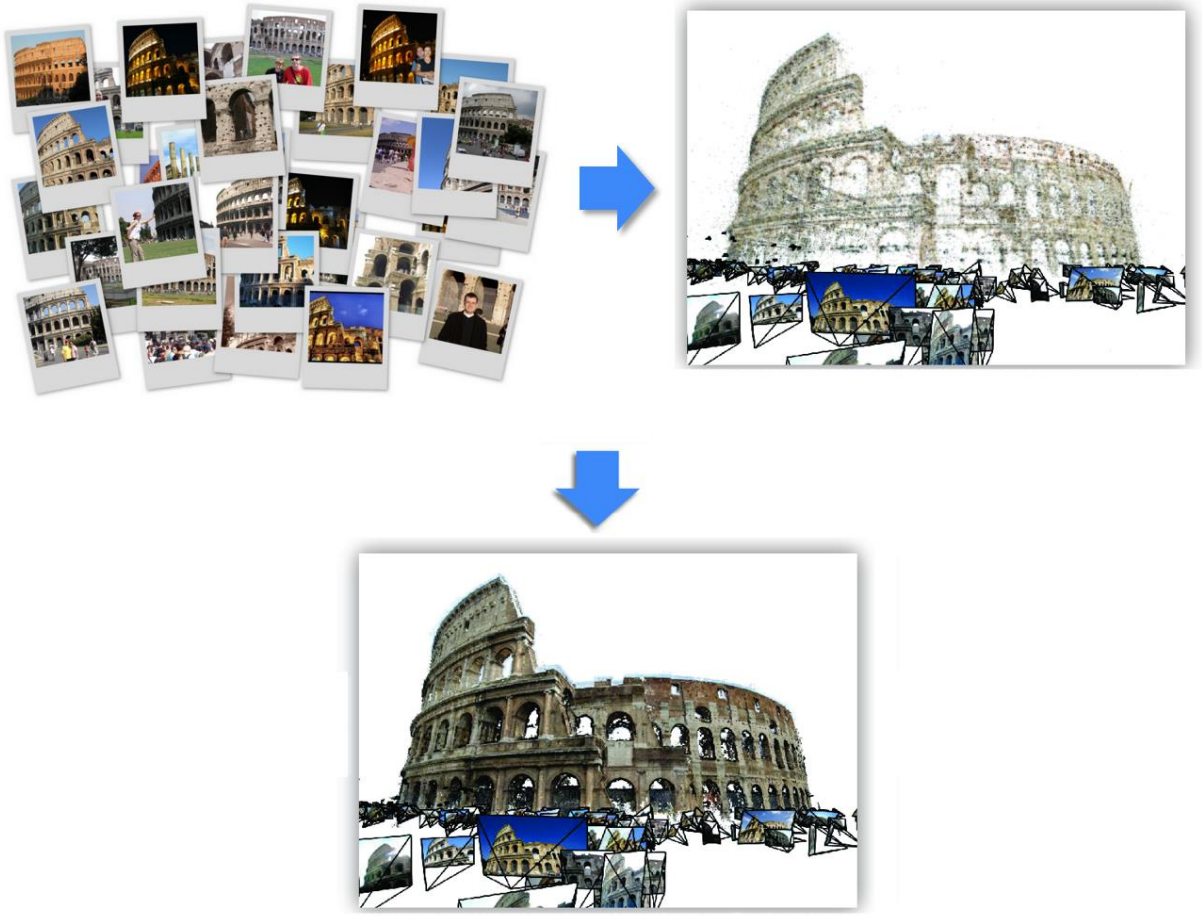


- Matching with fixed-size windows will fail!
- Possible solution: adaptively vary window size

Structure and motion (S&M)

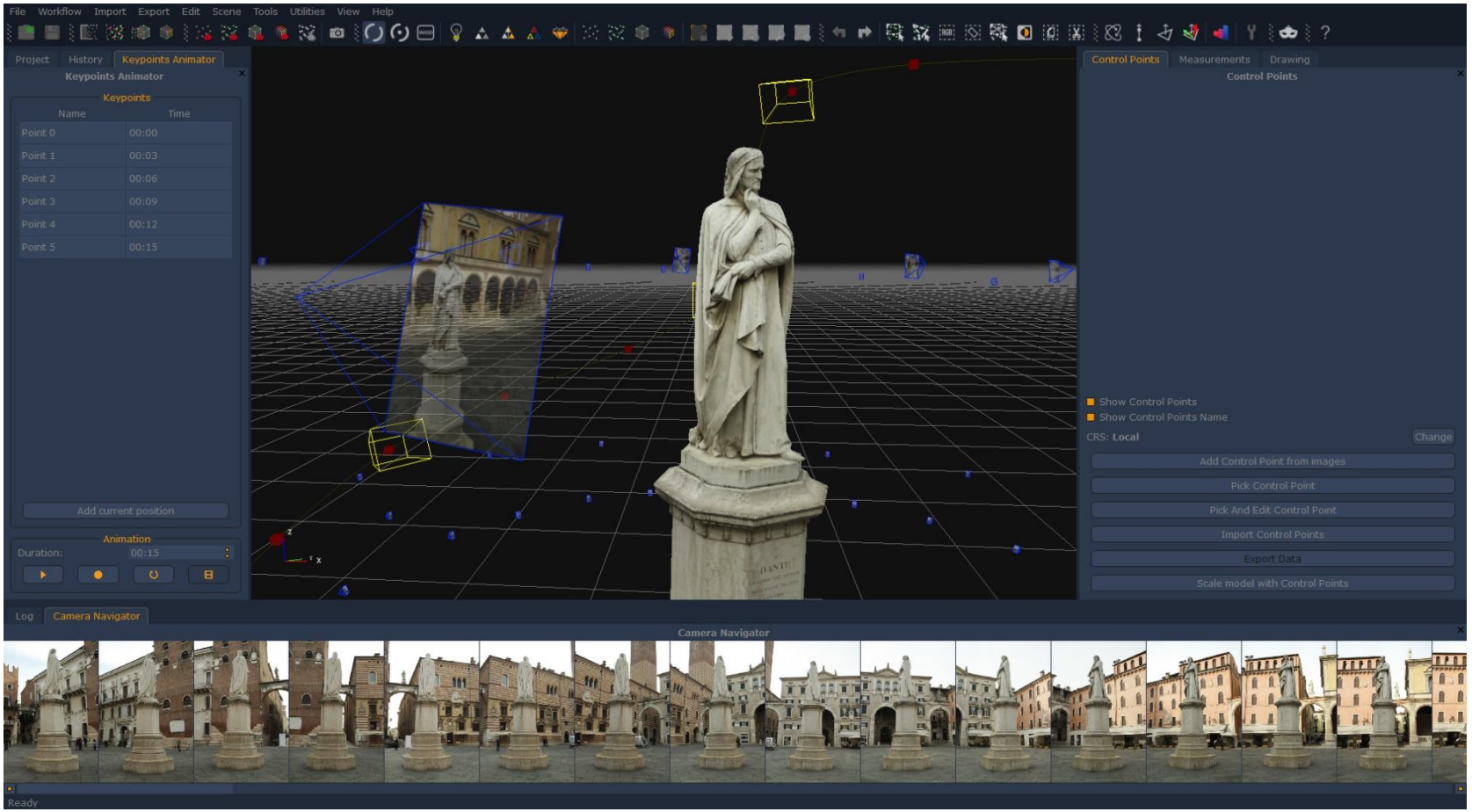


Large scale S&M



Building Rome in a day: https://grail.cs.washington.edu/rome/

Structure and motion



3DF Zephyr Pro

Structure and motion

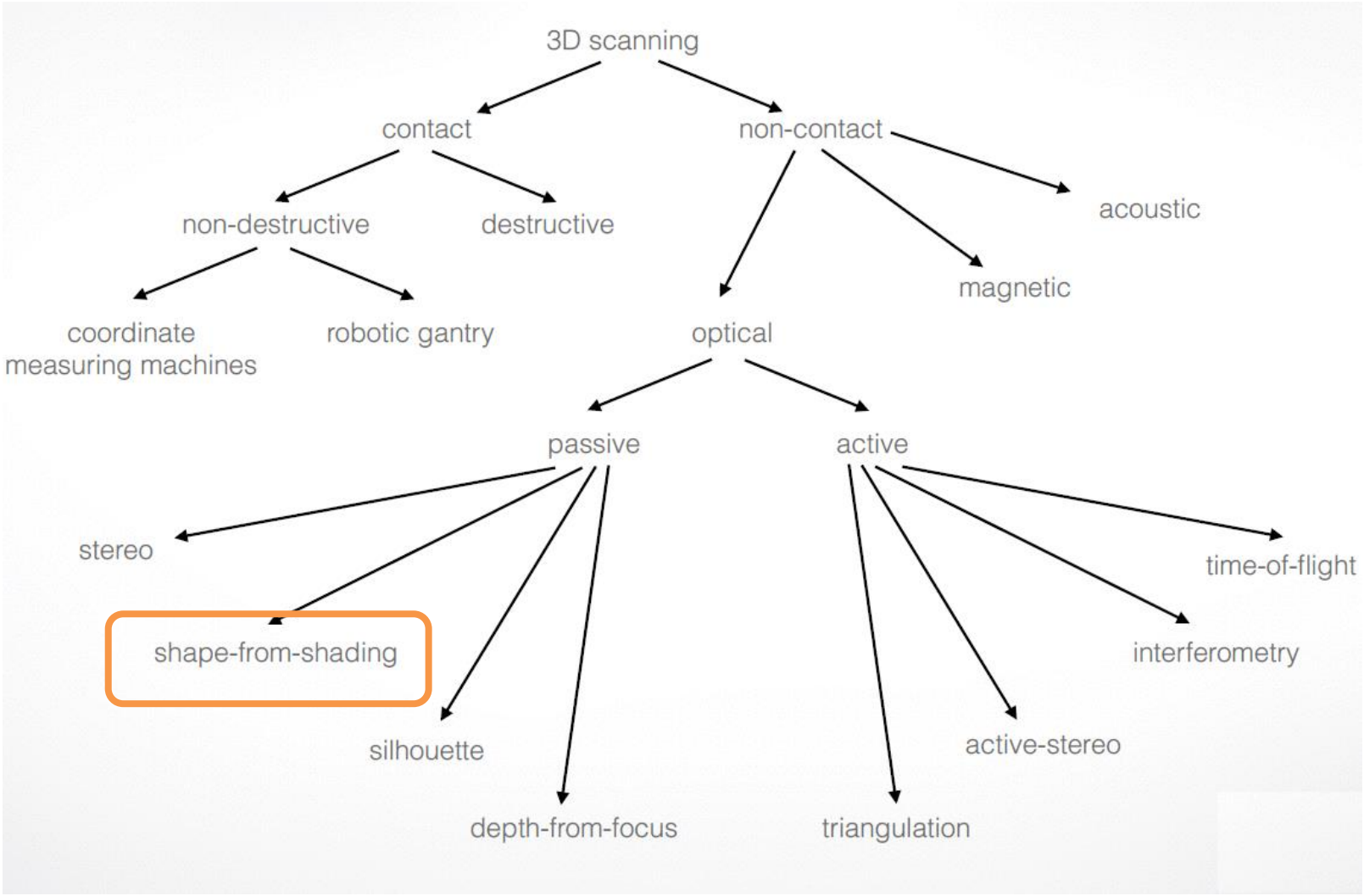
https://www.3dflow.net/3df-zephyr-photogrammetry-software//

3DF Zephyr Pro

From aerial images

3DF Zephyr Pro

A taxonomy



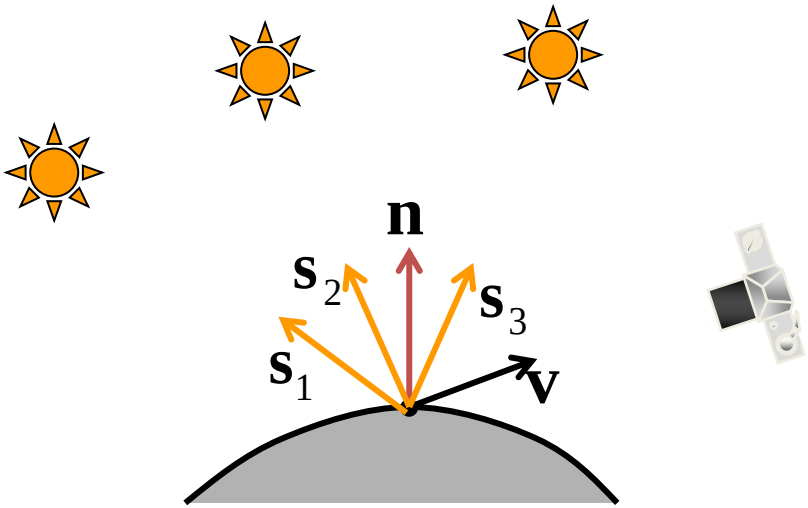
Hao Li http://www.hao-li.com/cs599-ss2015/slides/Lecture04.2.pdf

Shape from shading

- Image with a controlled illumination set-up
  - Not a proper ‘passive’ system…,
- Only surface normals are estimated,
- Better performance when more images with different light sources are used

Photometric stereo!

Photometric Stereo



Lambertian case:

$$I = \frac{\rho}{\pi} kc \cos\theta_i = \rho \mathbf{n} \cdot \mathbf{s} \qquad \left(\frac{kc}{\pi} = 1\right)$$

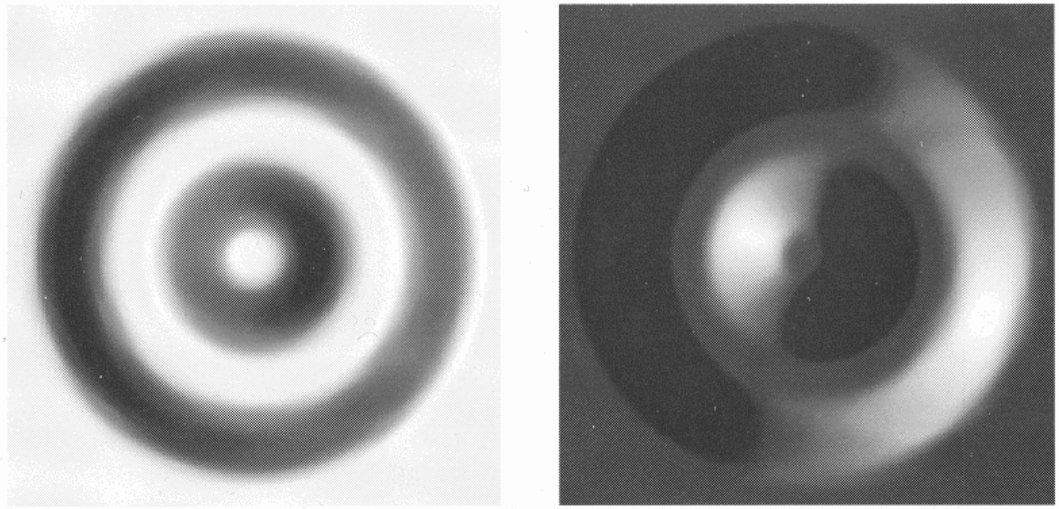
Image irradiance:

$$I_1 = \rho \mathbf{n} \cdot \mathbf{s}_1$$
$$I_2 = \rho \mathbf{n} \cdot \mathbf{s}_2$$
$$I_3 = \rho \mathbf{n} \cdot \mathbf{s}_3$$

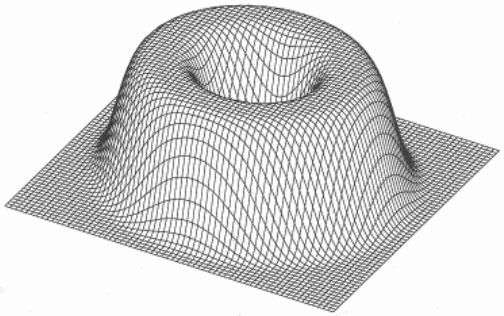
- We can write this in matrix form:

$$\begin{bmatrix} I_1 \\ I_2 \\ I_2 \end{bmatrix} = \rho \begin{bmatrix} \mathbf{s}_1^T \\ \mathbf{s}_2^T \\ \mathbf{s}_3^T \end{bmatrix} \mathbf{n}$$

Photometric Stereo



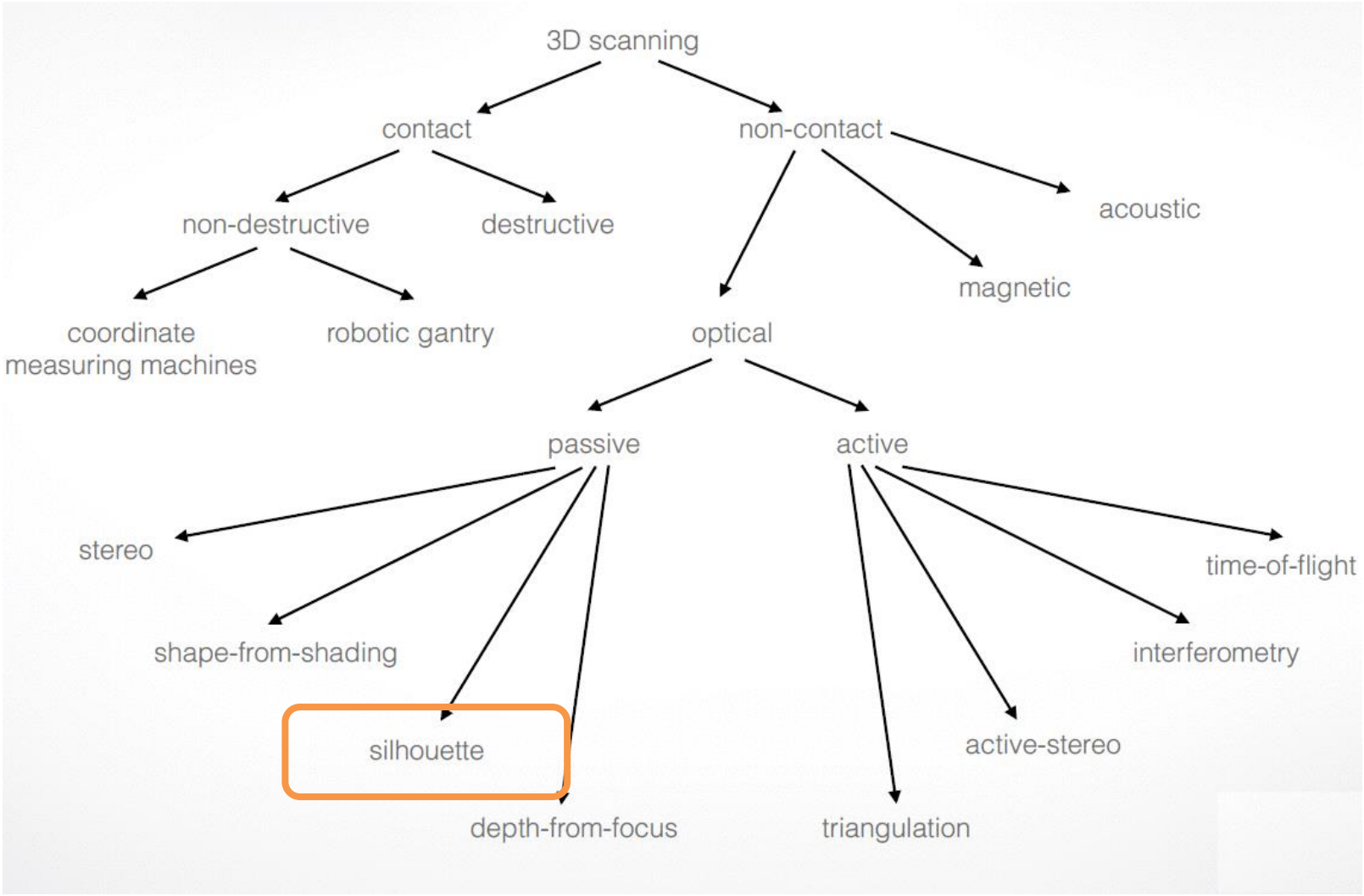
(b)



(a)

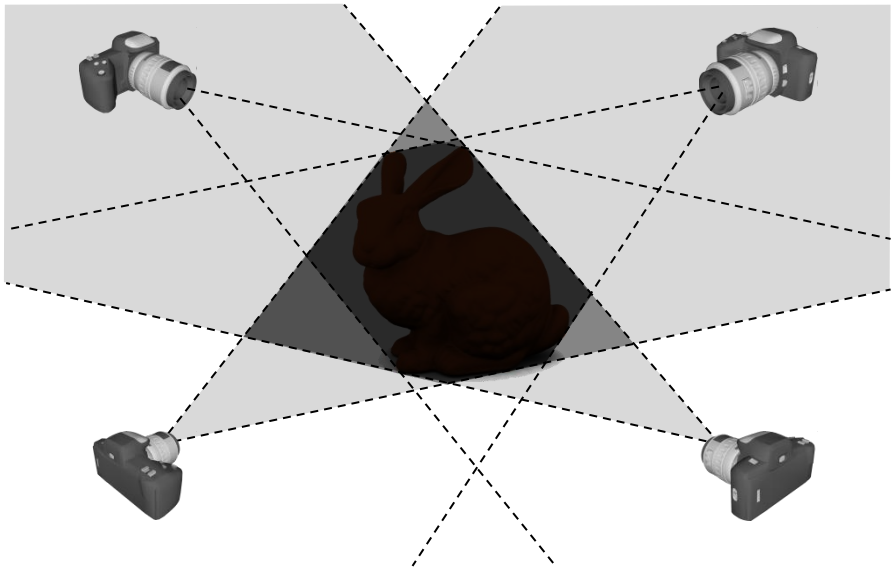
Figure 9.2 Two images of the same Lambertian surface seen from above but illuminated from different directions and 3-D rendering of the surface. Practically all the points in the top left image receive direct illumination ($\mathbf{i} = [0.20, 0, 0.98]^\top$); some regions of the top right image are in the dark due to self-shadowing effects ($\mathbf{i} = [0.94, 0.31, 0.16]^\top$).

A taxonomy



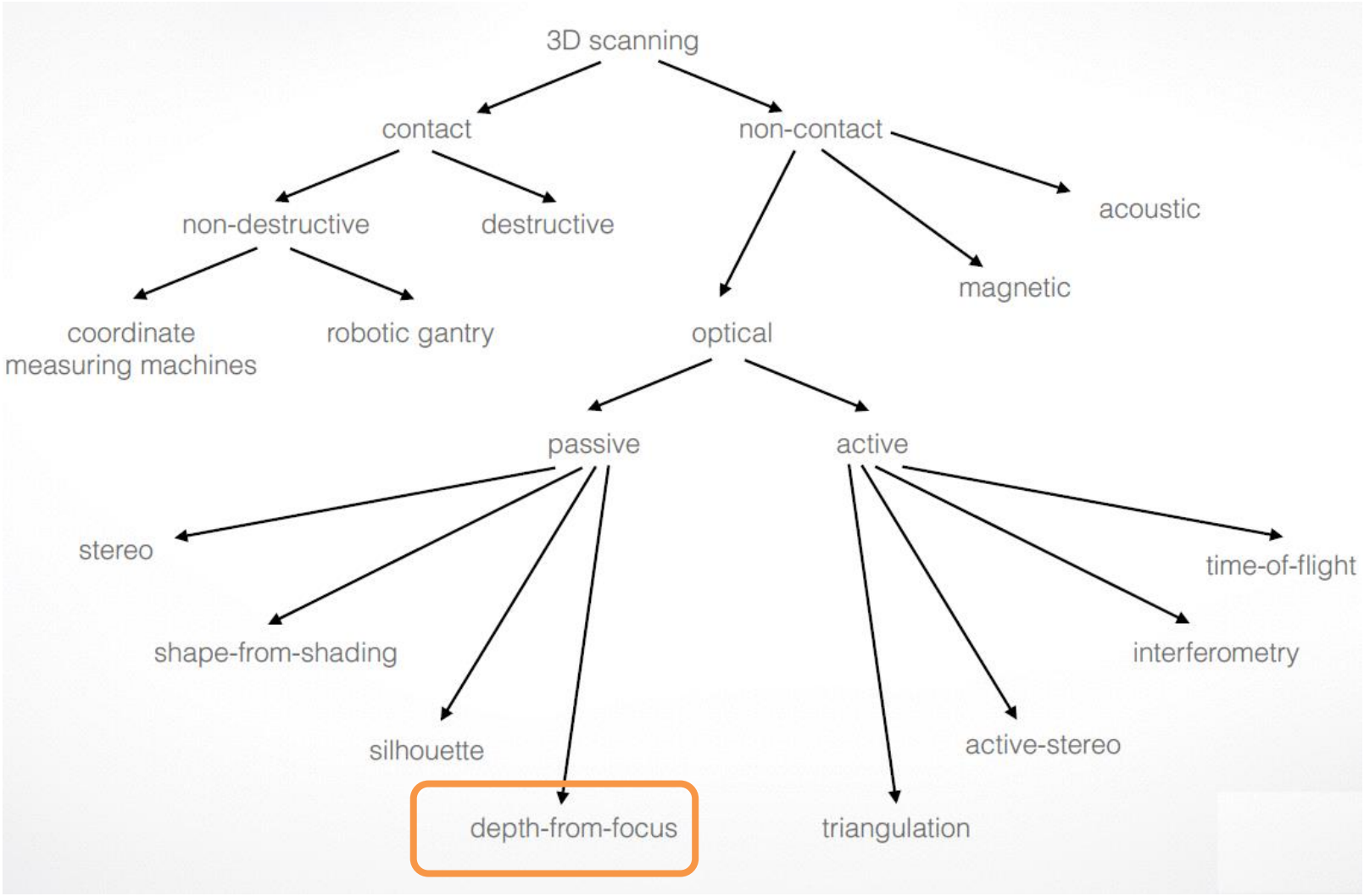
Hao Li http://www.hao-li.com/cs599-ss2015/slides/Lecture04.2.pdf

Shape from Silhouette



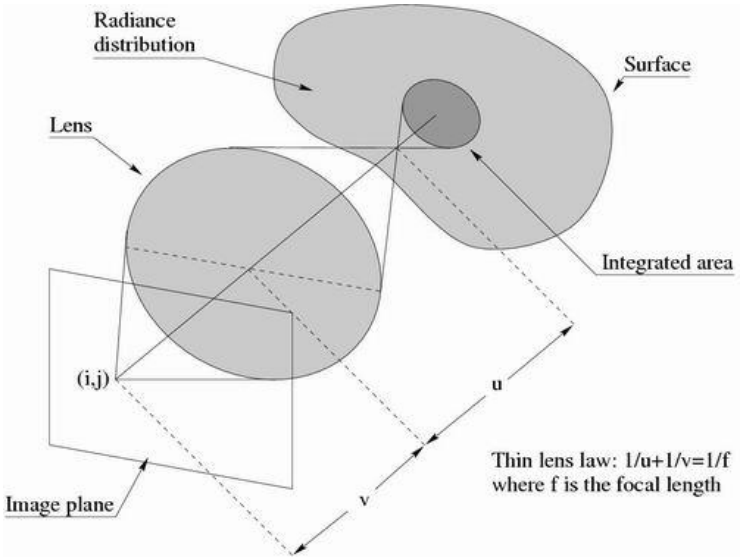
J. Starck and A. Hilton. Surface Capture for Performance-Based Animation. *IEEE Computer Graphics and Applications*, 2007

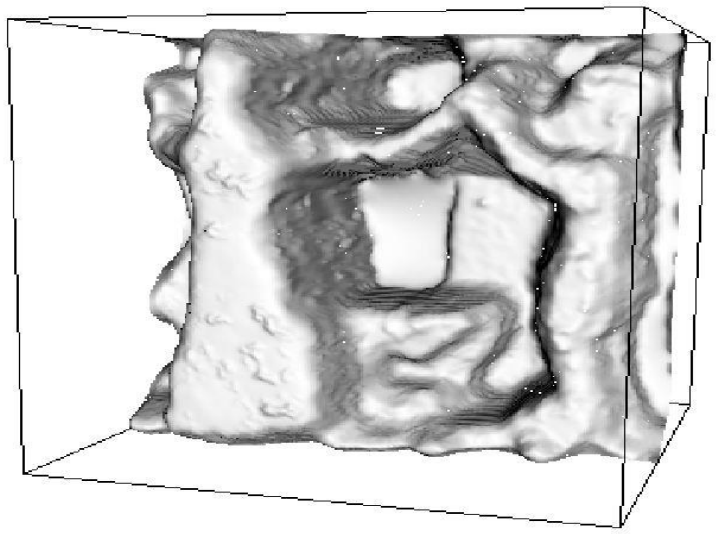
A taxonomy



Hao Li http://www.hao-li.com/cs599-ss2015/slides/Lecture04.2.pdf

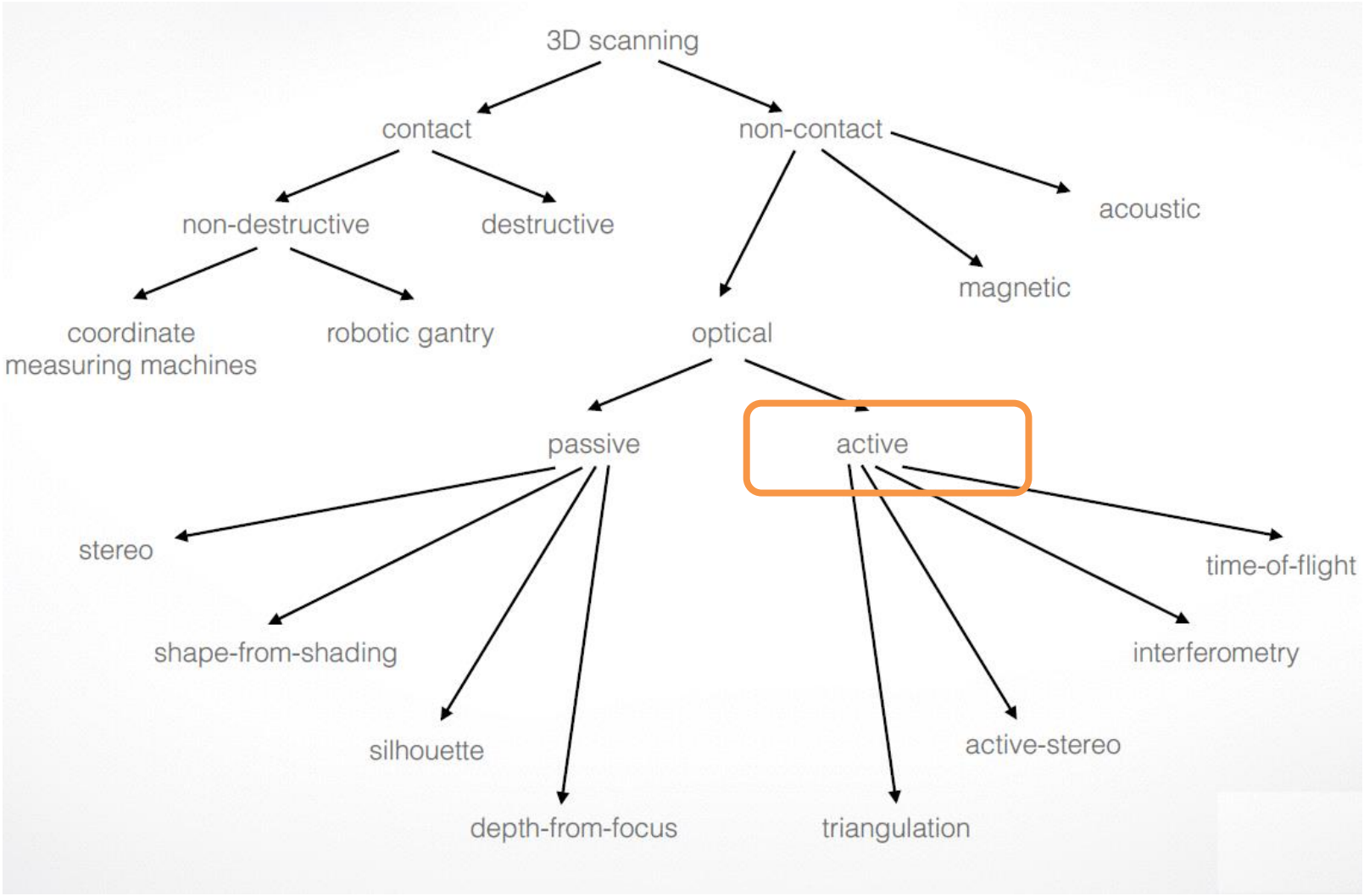
Depth from focus





M. Watanabe and S. Nayar. Rational filters for passive depth from defocus. *Intl. J. of Comp. Vision*, 27(3):203-225, 1998

A taxonomy

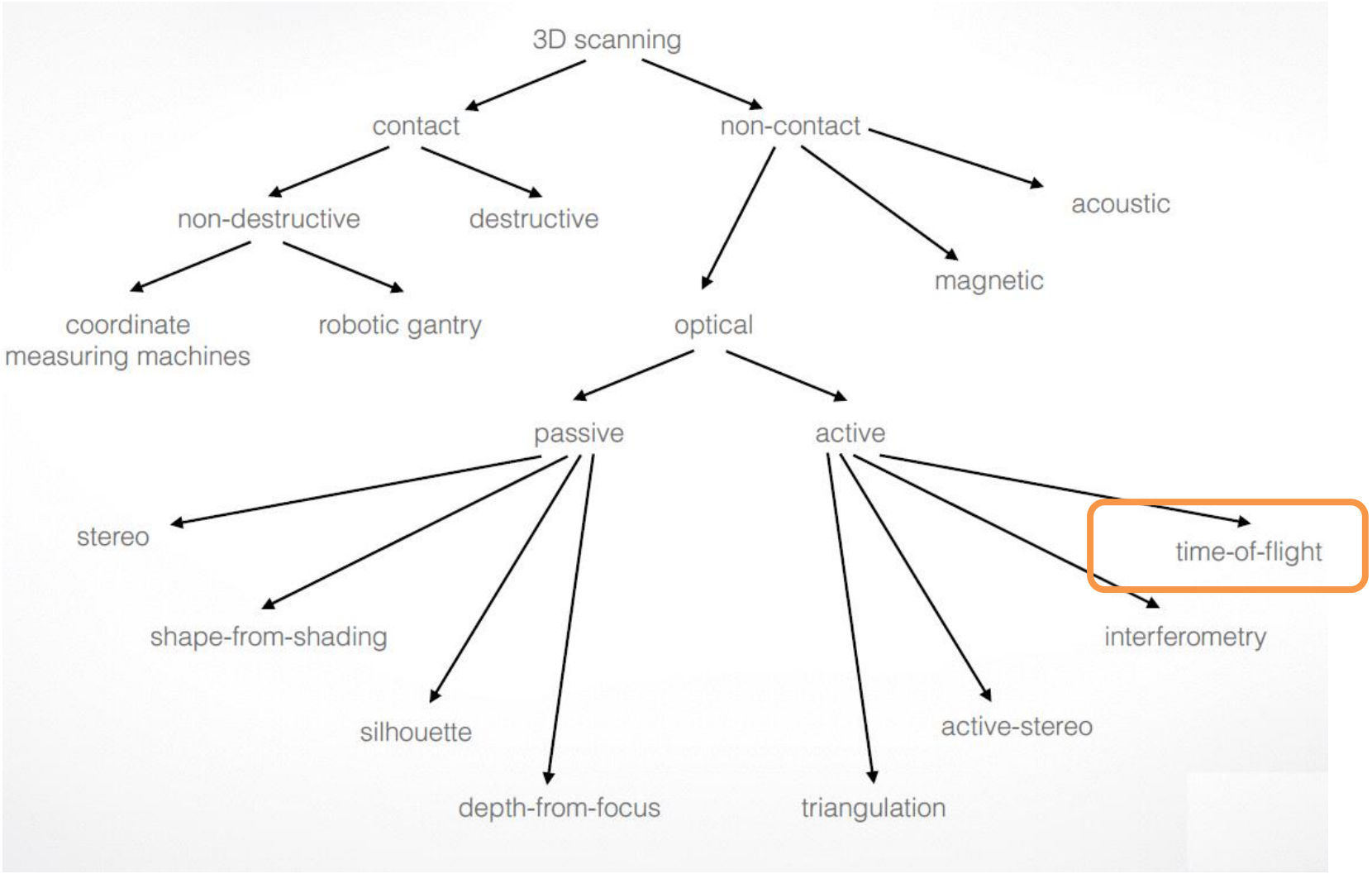


Hao Li http://www.hao-li.com/cs599-ss2015/slides/Lecture04.2.pdf

Active acquisition systems

- based on **sensor** and **emitter** (controlled EM wave)
- influence of surface reflectance to emitted signal
- **correspondence** problem simplified (via known signal) → les computation (realtime?)
- examples (laser, structured light)
- high resolution and dense capture possible, even for texture poor regions
- more sensitive to surface reflection properties (mirrors?)

A taxonomy

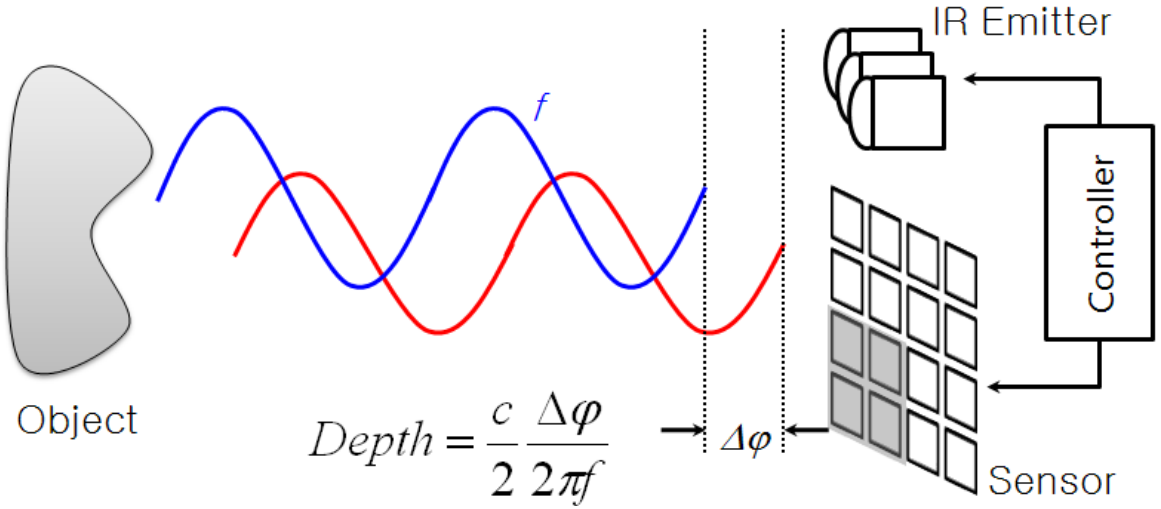


Hao Li http://www.hao-li.com/cs599-ss2015/slides/Lecture04.2.pdf

Time of flight

Probe object by laser or infrared light!

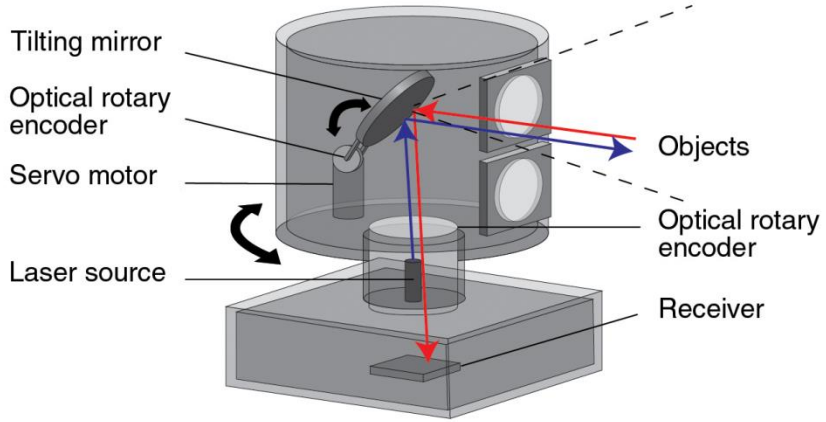
- Emit pulse of light, measure time till reflection from surface is seen by a detector
- Known speed of light & round-trip time allows to compute distance to surface



Time of flight

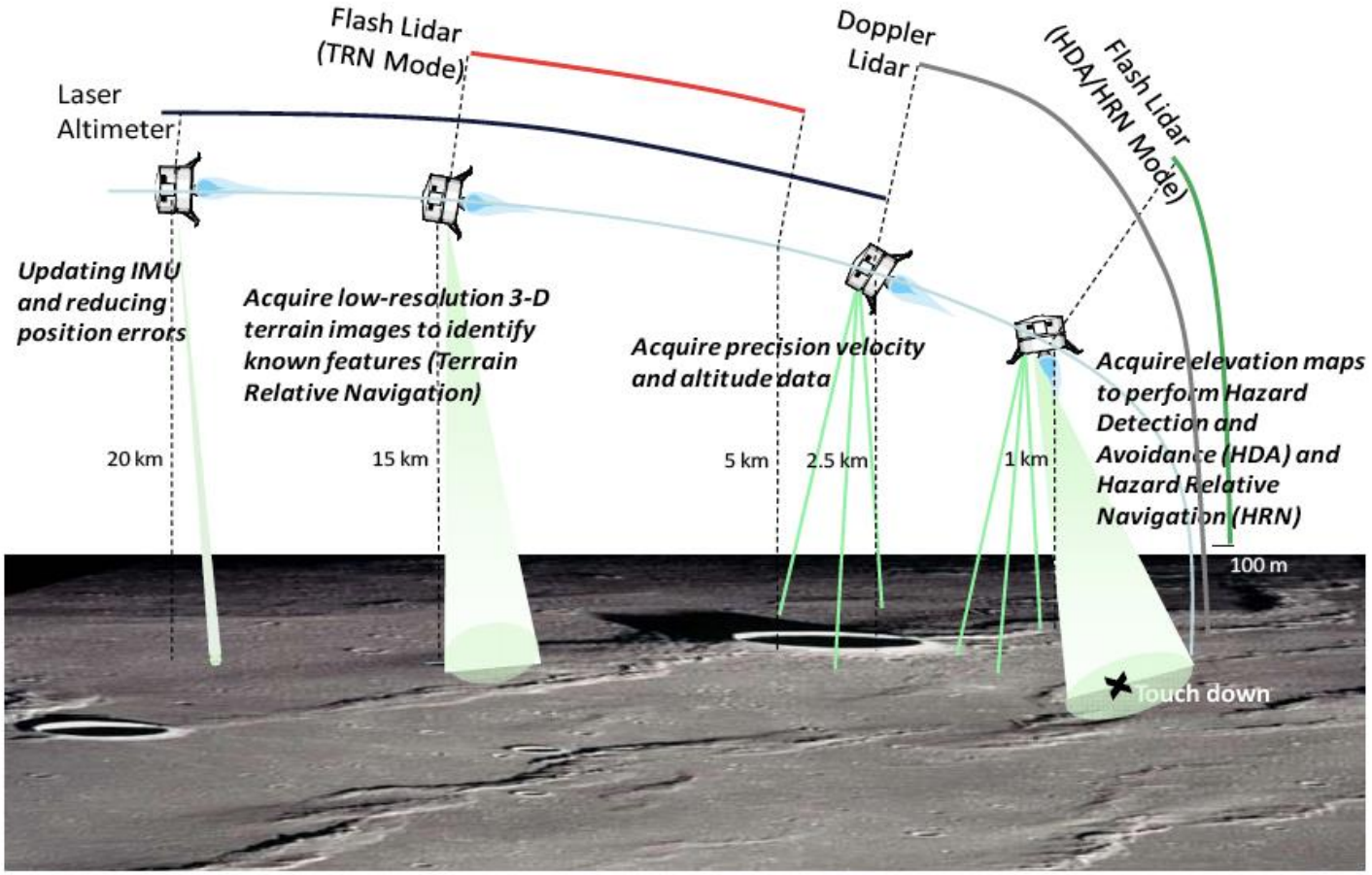
Laser LIDAR!

- **Li**ght **D**ectection **a**nd **R**anging
- Good for long distance scans
- 6mm accuracy at 50 m distance



Time of flight

Laser LIDAR for planetory exploration



Time of flight

Laser LIDAR for planetory exploration

Sensor	Function	Operational Altitude Range	Precision/Resolution
Flash Lidar	HDA/HRN	1000 m – 100 m	5 cm/40 cm
	TRN	15 km – 5 km	20 cm/6 m
	Altimetry[1]	20 km – 100 m	20 cm
Doppler Lidar	Velocimetry	2500 m – 10 m	1 cm/sec
	Altimetry	2500 m – 10 m	10 cm
Laser Altimeter	Altimetry	20 km – 100 m	20 cm
	TRN[1]	15 km – 5 km	20 cm

Time of flight

Infrared light!

- 176x144 pixels, up to 50 fps
- 30 cm to 5 m distance, 1 cm accuracy
- technology is improving drastically

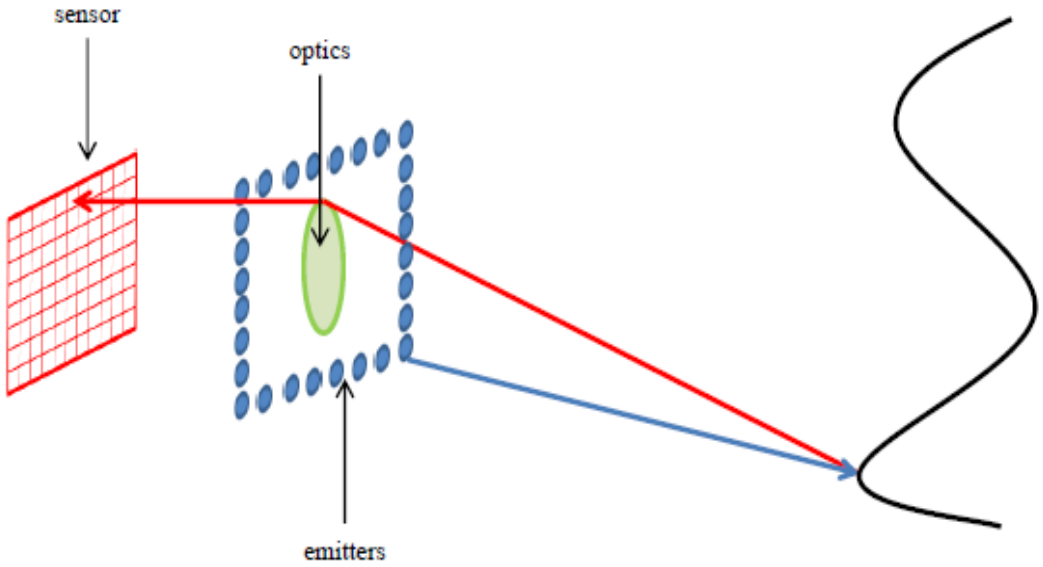


Fig. 2.7 ToF camera structure and signaling: propagation towards the scene (blue arrow), reflection (from the black surface on the right), back-propagation (red arrow) towards the camera through the optics (green) and reception (red sensor).

Matricial Time-of-Flight

Time of flight

Kinect One (= second gen Kinect)!

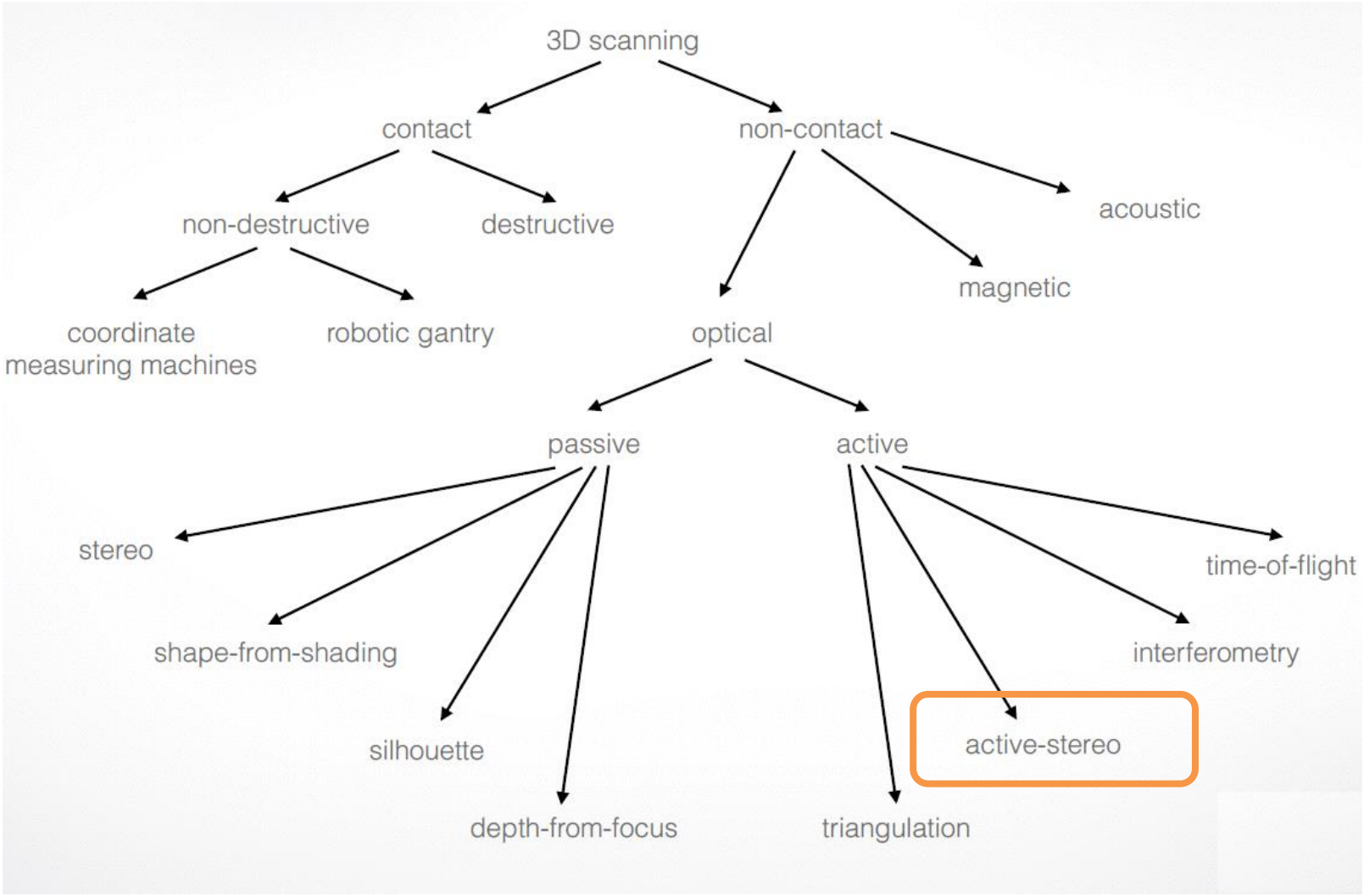
- 30 fps
- Depth map x/y resolution: 512 x 424
- z-resolution 1 mm & accuracy:
- <1.5 mm (depth < 50 cm)
- < 3.9 mm (depth < 180 cm)
- < 17.6 mm (depth < 450 cm)
- 1080 HD for RGB input
- uses Kinect2 SDK

Time of flight

Intel RealSense L514

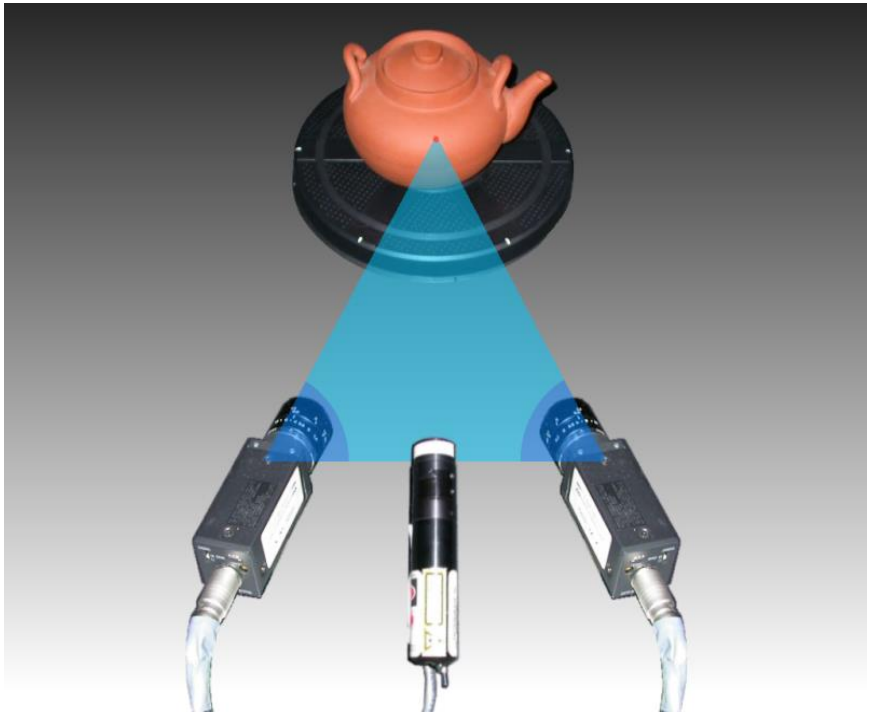
- 30 fps
- Depth map x/y resolution: 1024x768
- **Depth Accuracy:** ~5 mm to ~14 mm thru 9 m^2
- 1920x1080 RGB input

A taxonomy



Hao Li http://www.hao-li.com/cs599-ss2015/slides/Lecture04.2.pdf

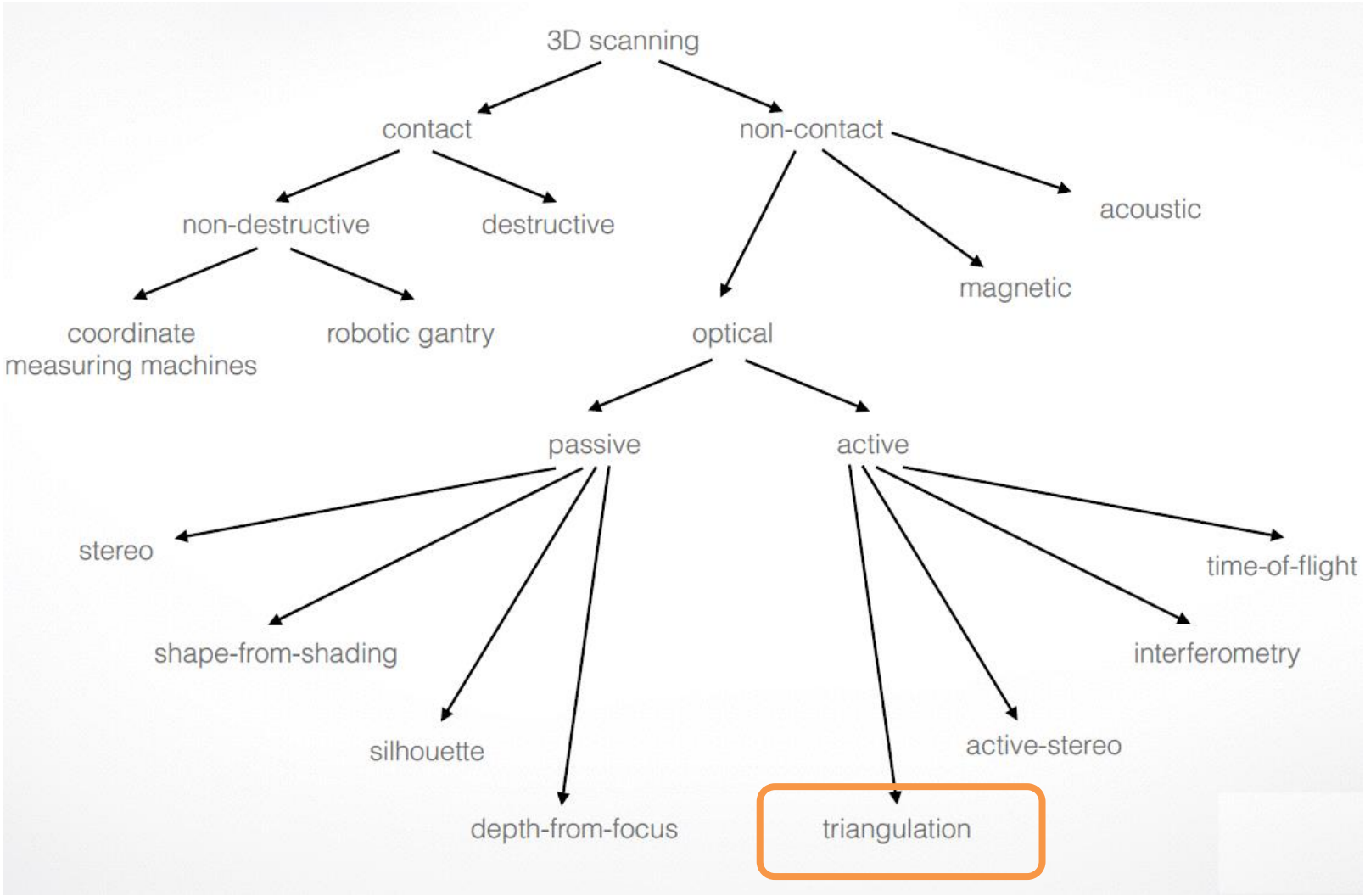
Active stereo



Active stereo

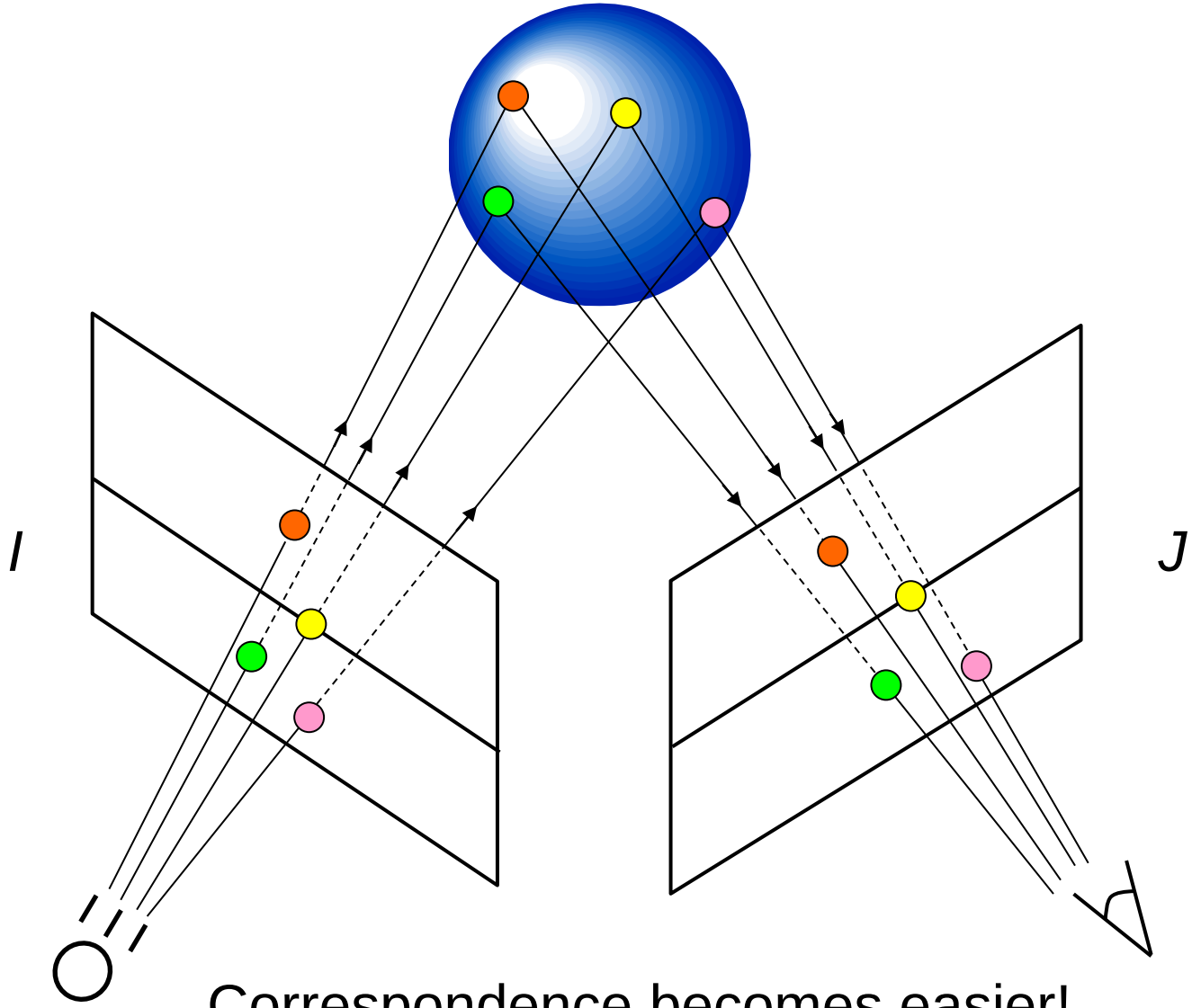
M. Waschbüsch et al. Scalable 3D Video of Dynamic Scenes. *The Visual Computer*, pp. 629-638, 2005

A taxonomy



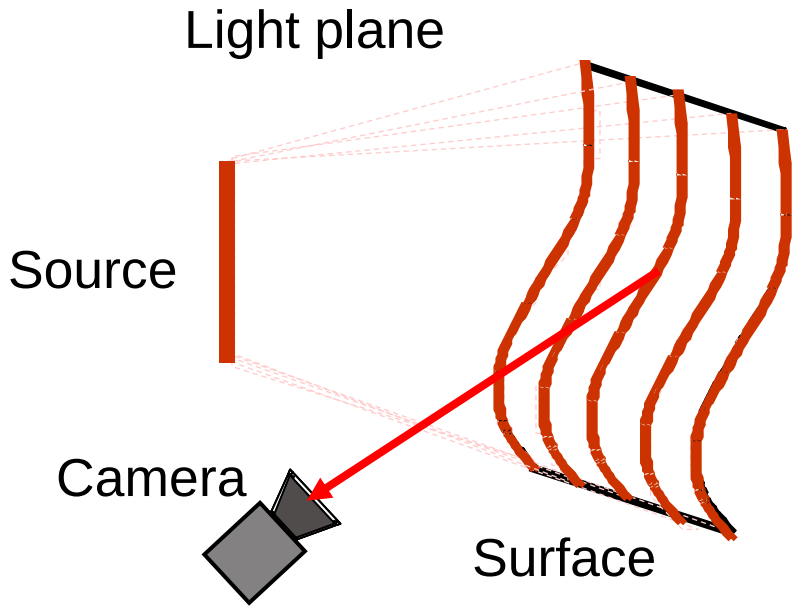
Hao Li http://www.hao-li.com/cs599-ss2015/slides/Lecture04.2.pdf

Structured Light Triangulation



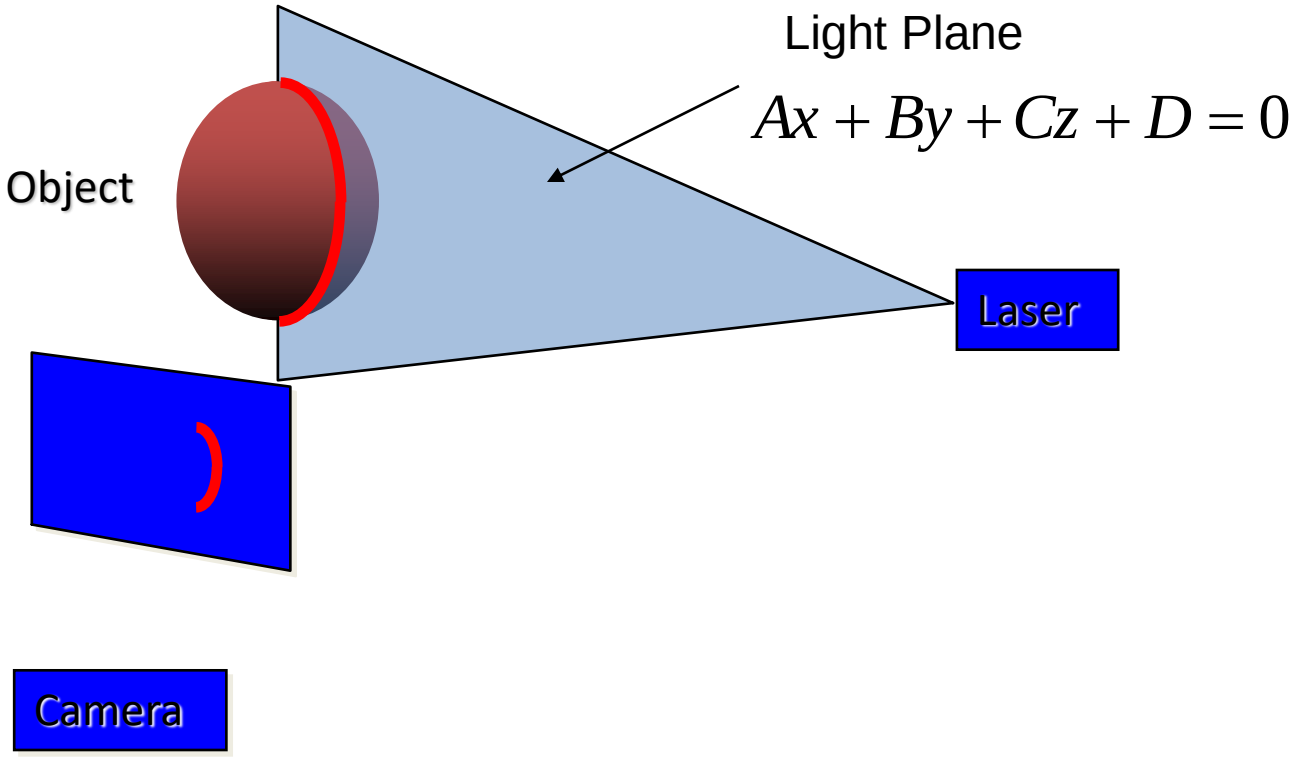
Correspondence becomes easier!

Light Stripe Scanning – Single Stripe



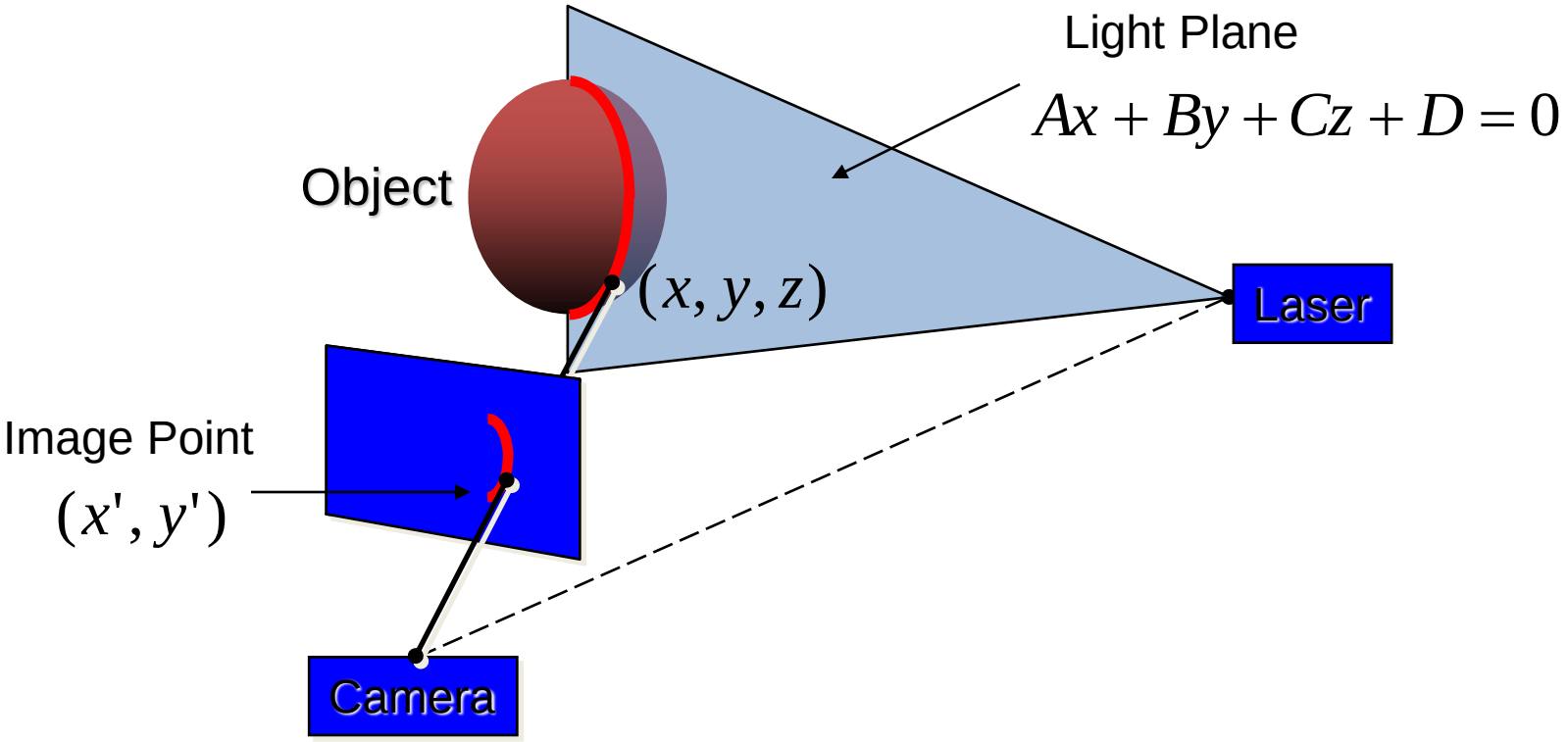
- Optical triangulation
  - Project a single stripe of laser light
  - Scan it across the surface of the object
  - This is a very precise version of structured light scanning
  - Good for high resolution 3D, but needs many images and takes time

Triangulation



- Project laser stripe onto object

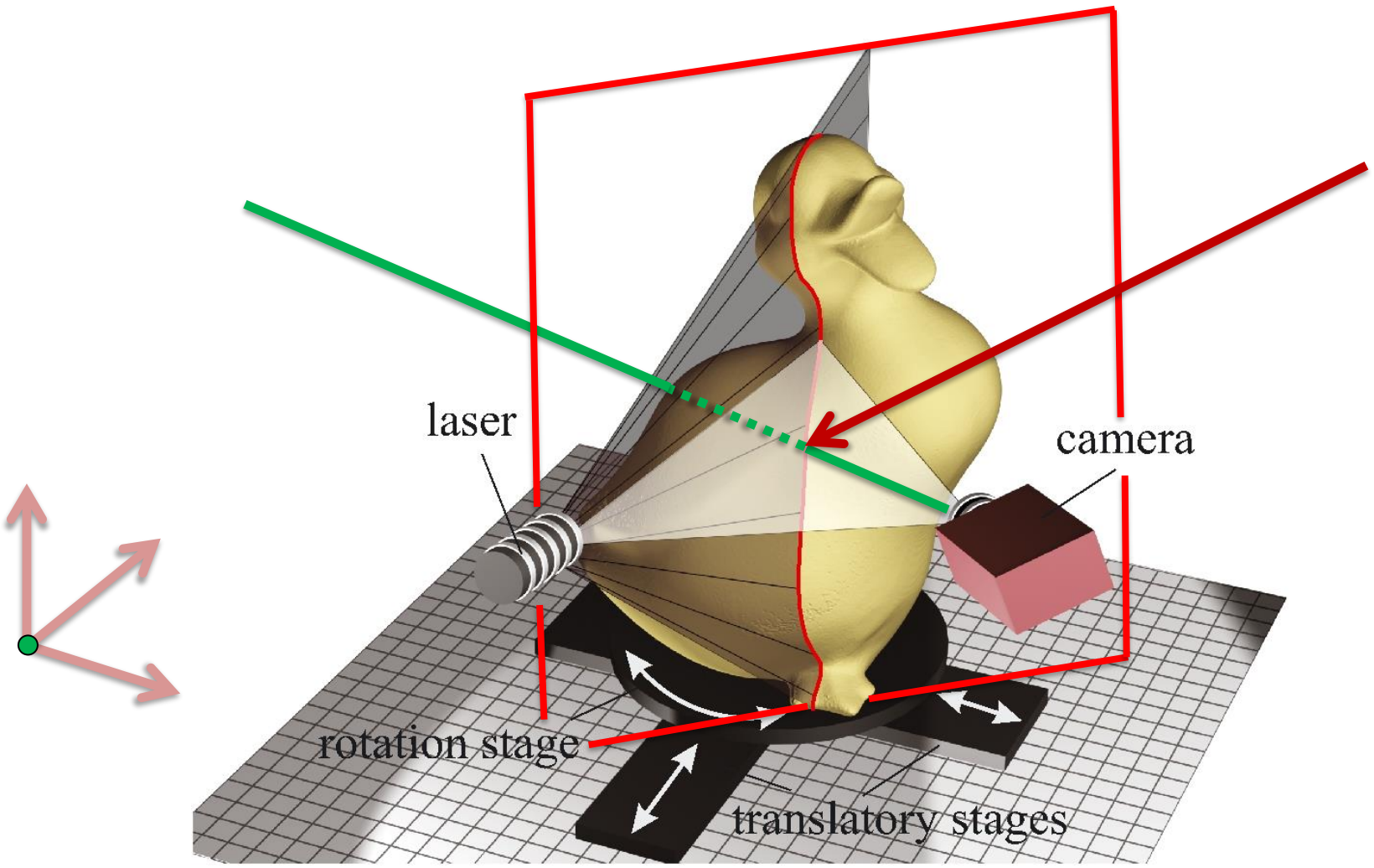
Triangulation



- Depth from ray-plane triangulation:
  - Intersect camera ray with light plane

$$x = x'z / f \qquad y = y'z / f \qquad z = \frac{-Df}{Ax' + By' + Cf}$$

3D Triangulation: plane-ray intersection

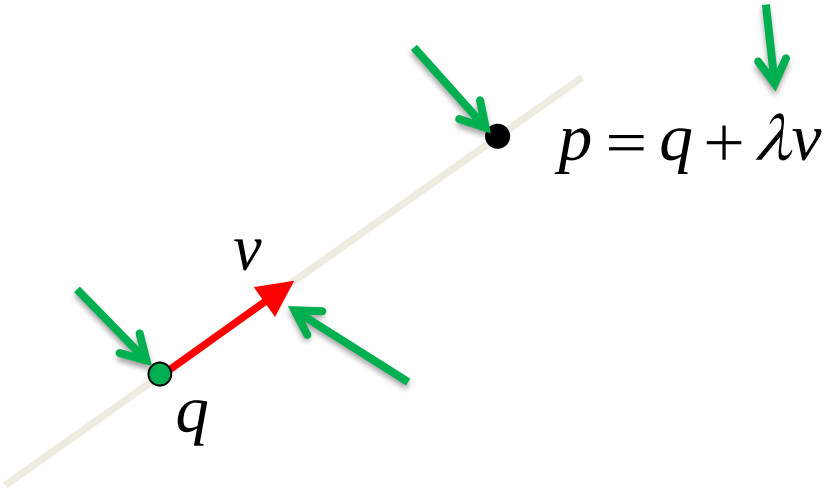


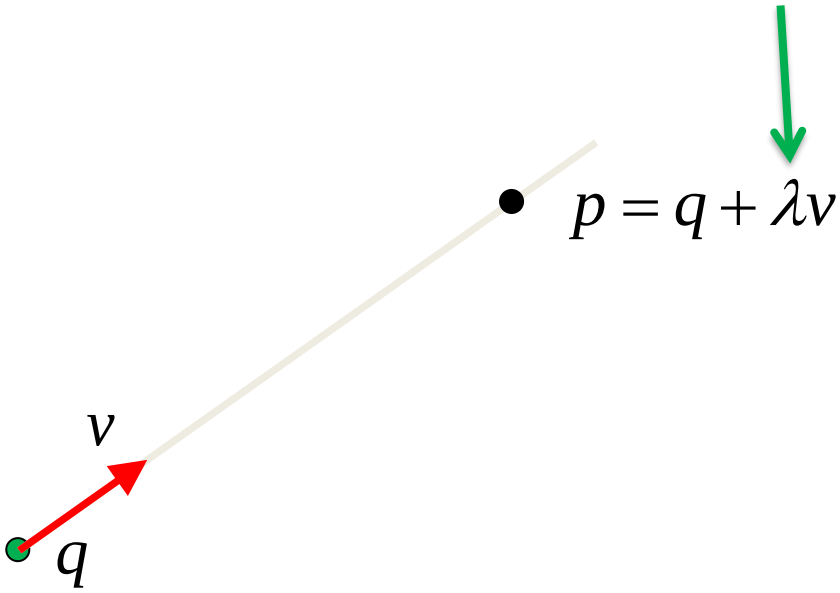
Representation of lines and rays

COMPUTER VISION

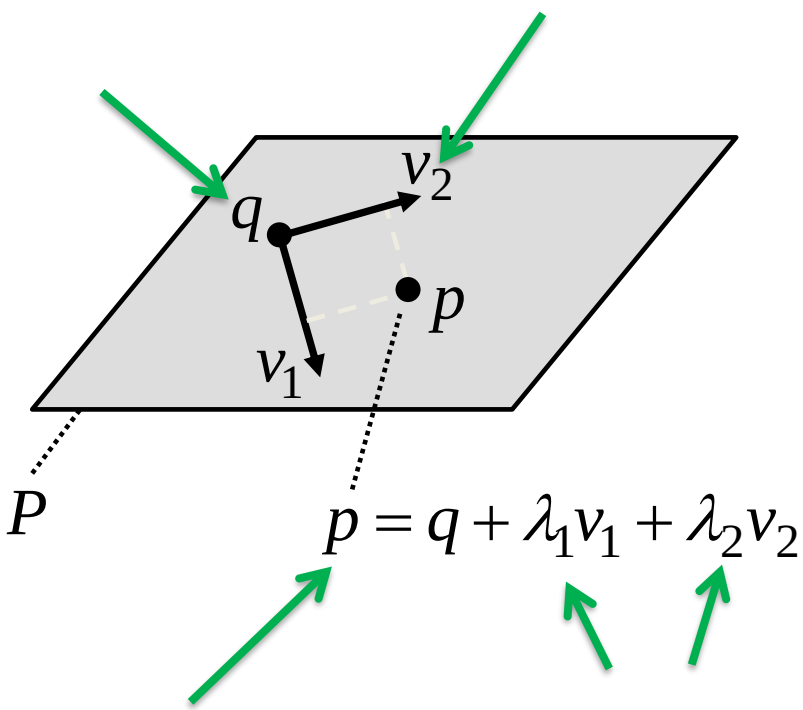
$M = C + \lambda \begin{bmatrix} Q^{-1}m \\ 0 \end{bmatrix}$

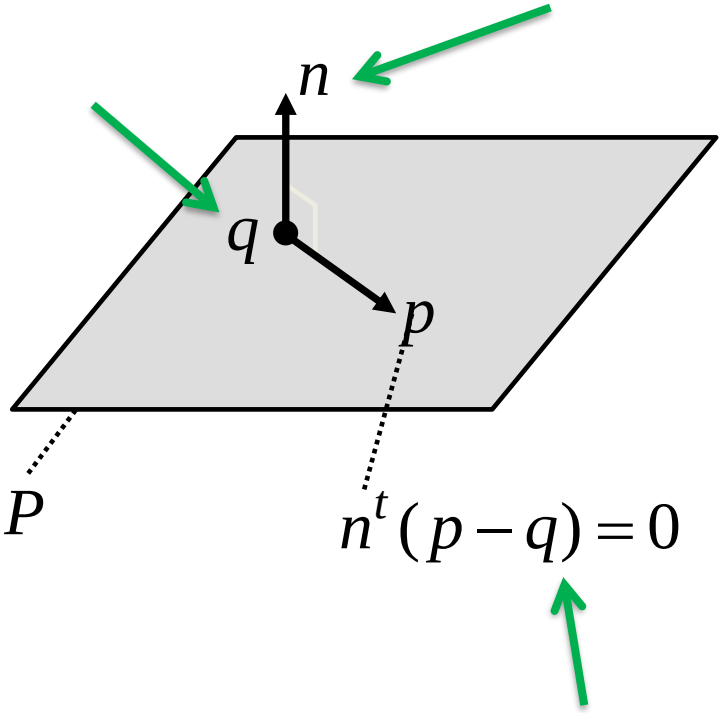
$P = [Q \mid q]$



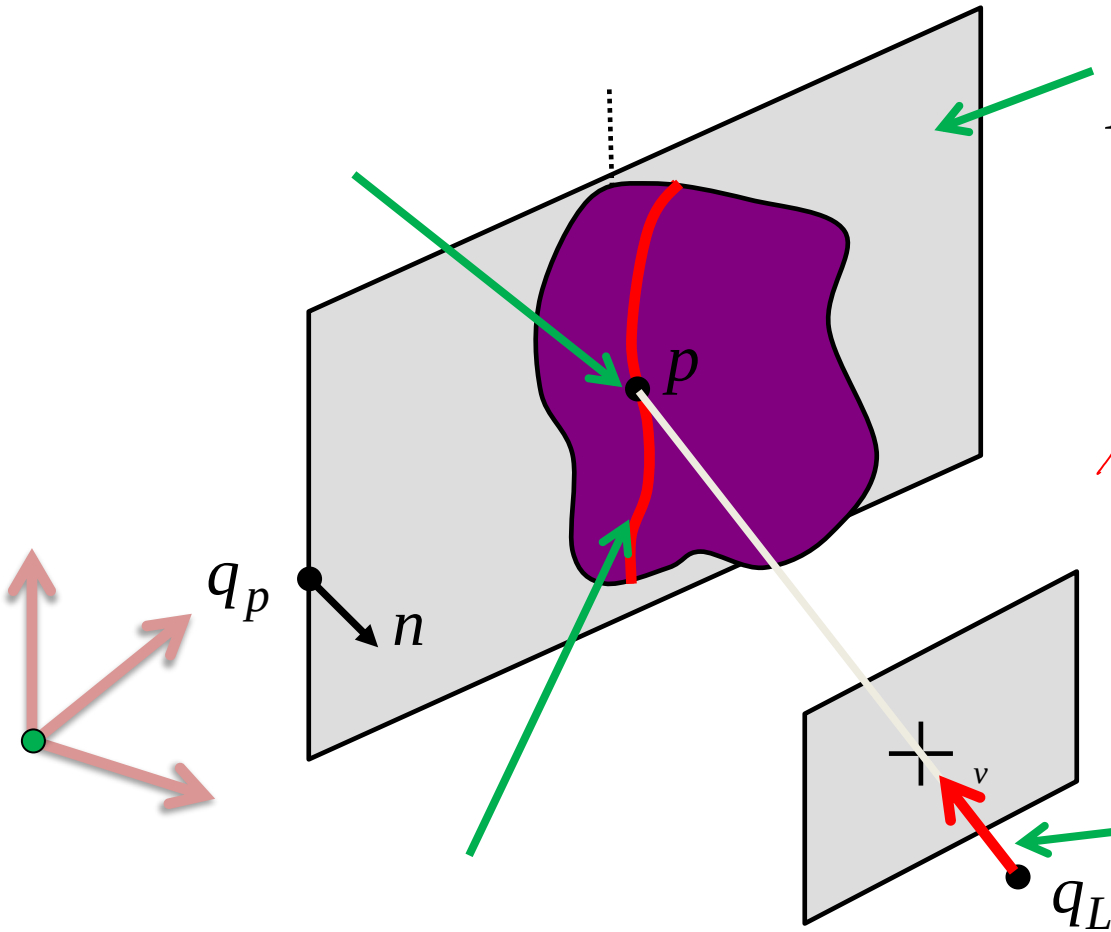


Representation of planes





Triangulation by line-plane intersection



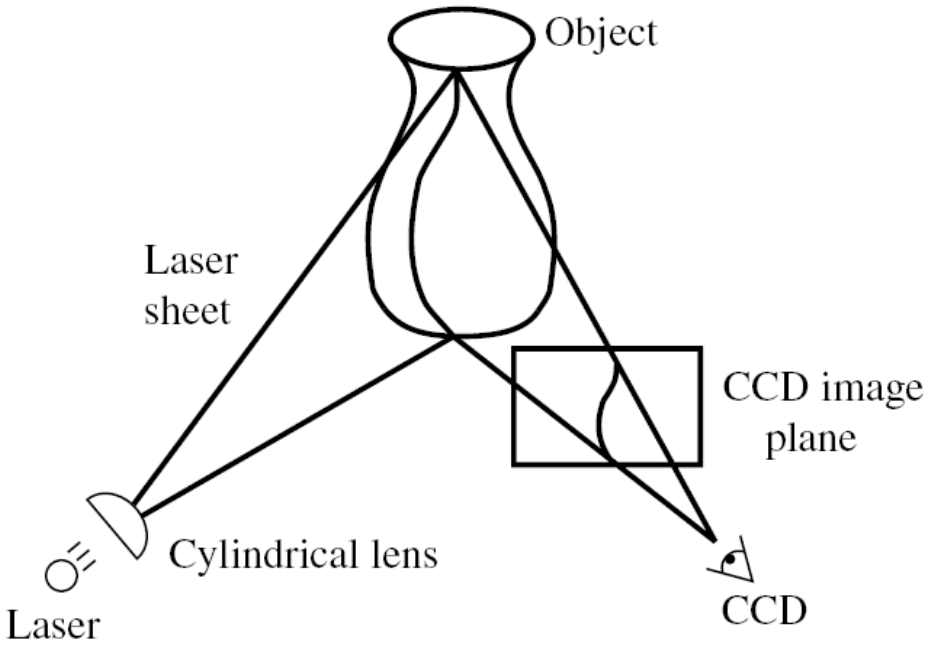
$$P = \{p : n^t(p - q_p) = 0\}$$

$$n^t(\lambda v + q_L - q_p) = 0$$

$$\lambda = \frac{n^t(q_p - q_L)}{n^t v}$$

$$L = \{p = q_L + \lambda v\}$$

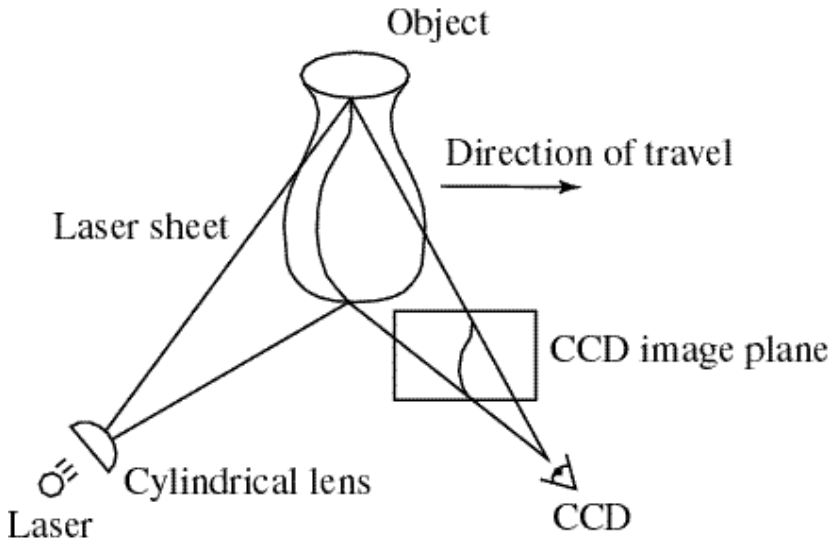
Example: Laser scanner

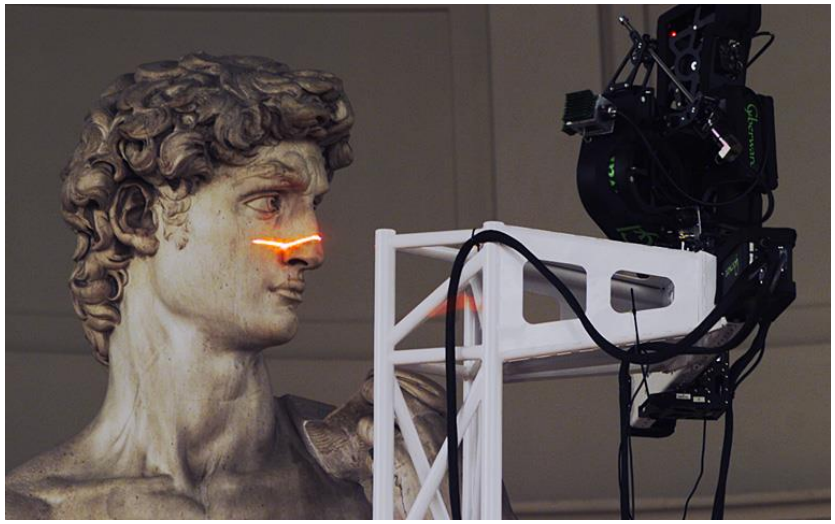


Cyberware® face and head scanner

+ very accurate < 0.01 mm
− more than 10sec per scan

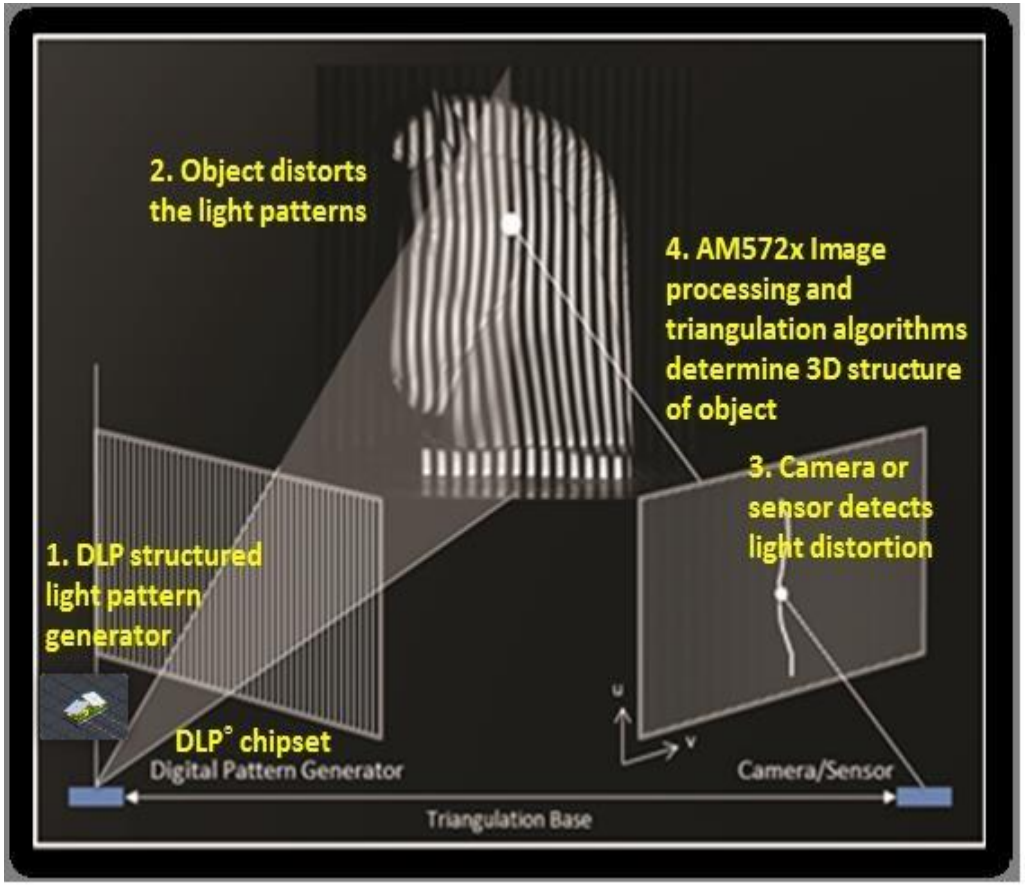
Example: Laser scanner





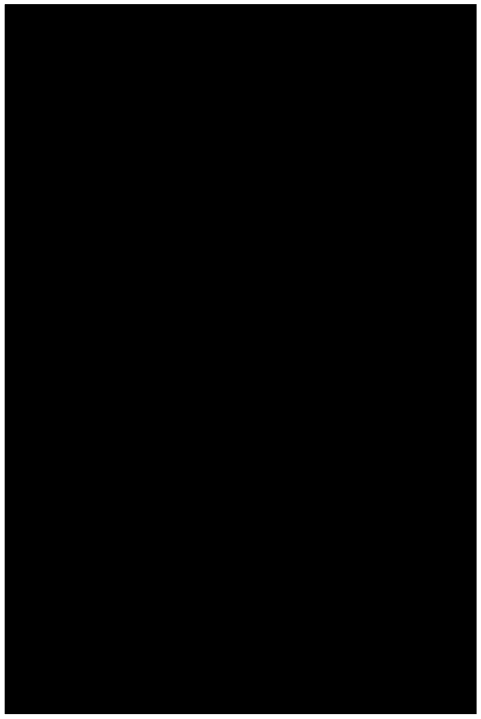
Digital Michelangelo Project
http://graphics.stanford.edu/projects/mich/

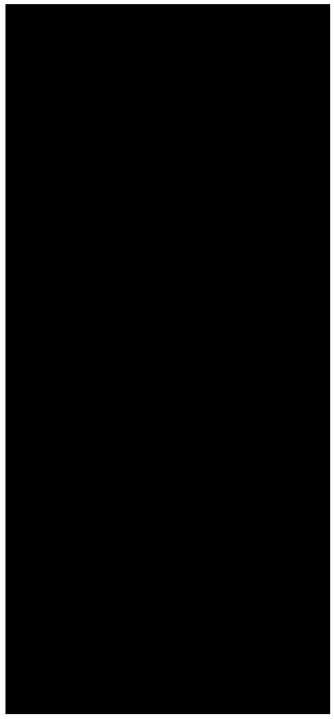
Lighting Patterns



- Any spatio-temporal pattern of light projected on a surface using a standard light projector.
- Cleverly illuminate the scene to extract correspondences
- Avoids problems of mechanical motion
- Allows faster acquisition projecting multiple patterns simultaneously

Structured Lighting: Swept-Planes Revisited





- Swept-plane scanning recovers 3D depth using ray-plane intersection
- Use a data projector to replace manually-swept laser/shadow planes
- How to assign correspondence from projector planes to camera pixels?
- which stripe is which?
- Solution: Project a spatially- and temporally-encoded image sequence
- **What is the optimal image sequence to project?**

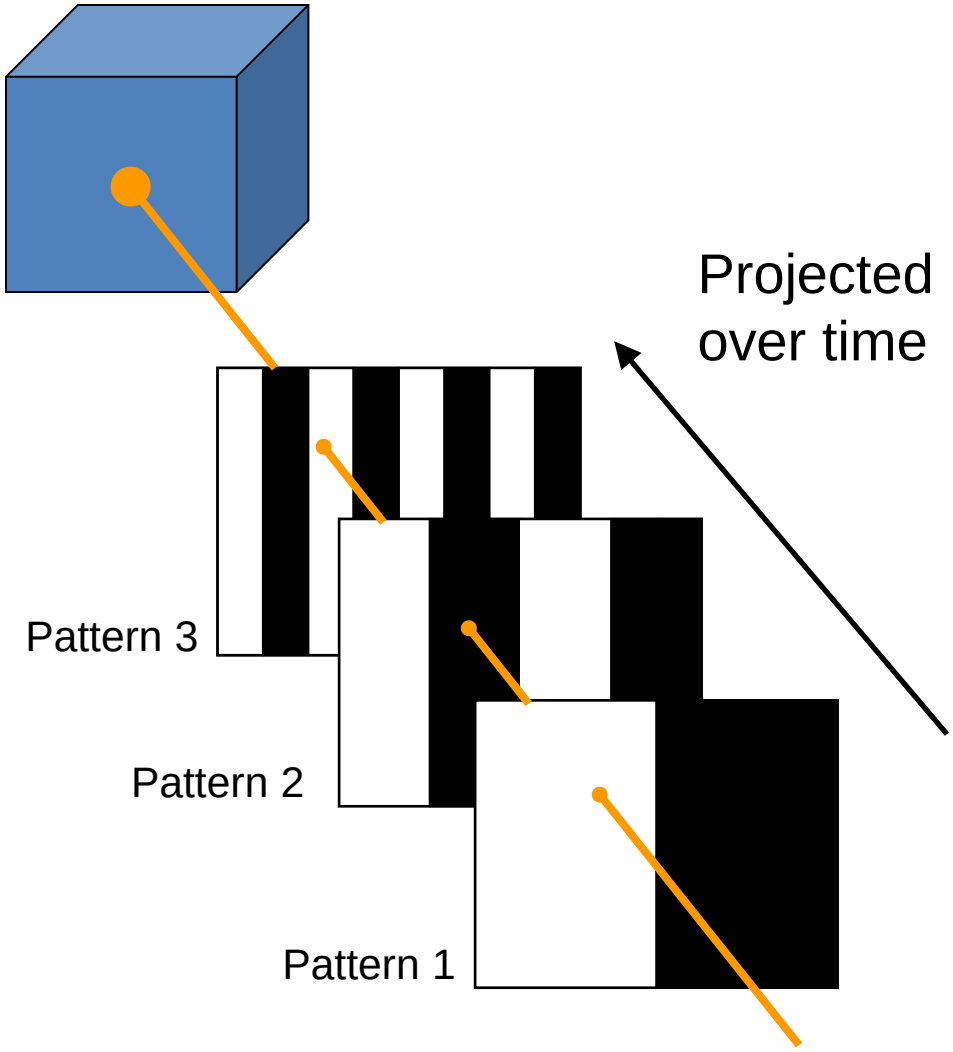
Binary Coding

Faster:

$2^n - 1$ stripes in n images.

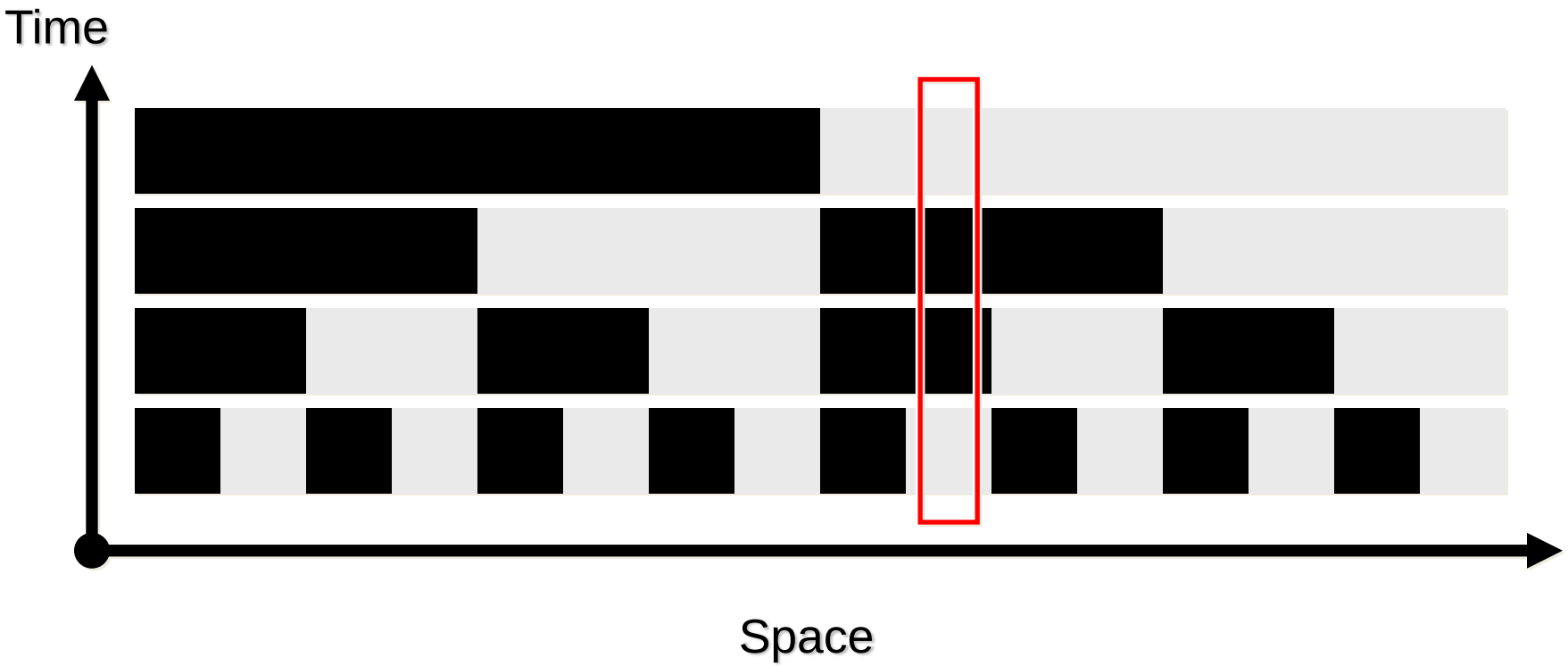
Example:

3 binary-encoded patterns which allows the measuring surface to be divided in 8 sub-regions

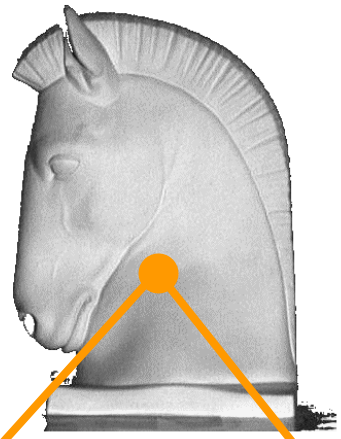


Binary Coding

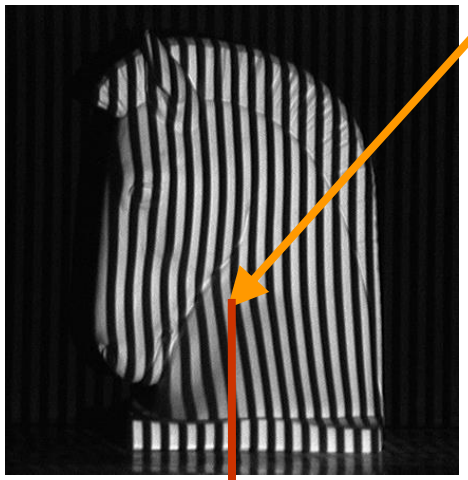
- Assign each stripe a unique illumination code over time [Posdamer 82]



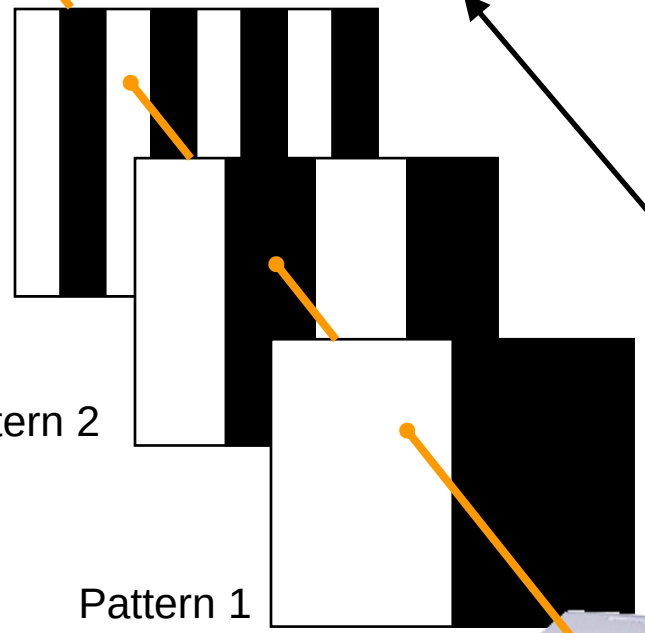
Binary Coding



Example: 7 binary patterns proposed by Posdamer & Altschuler

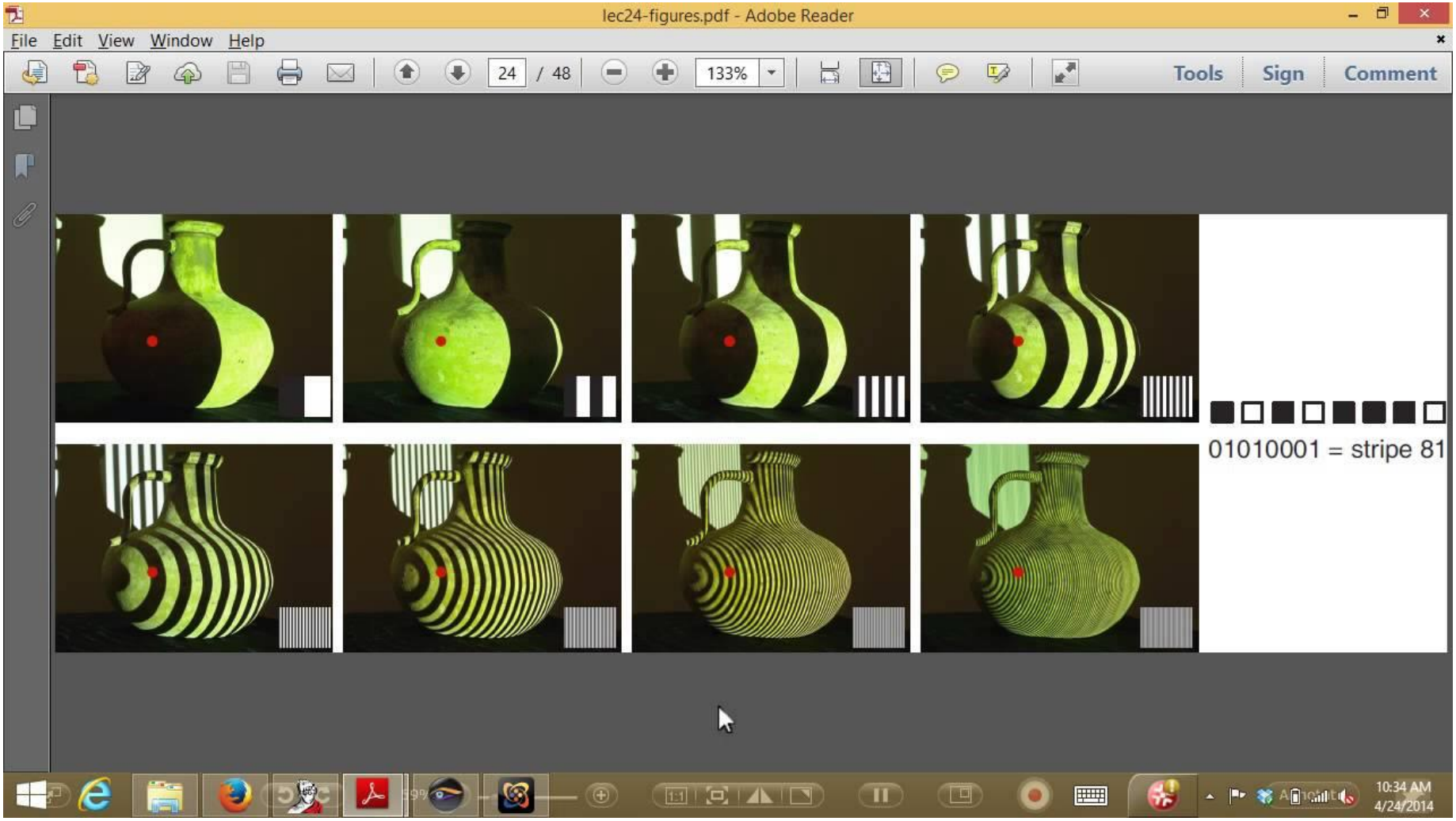


. . .

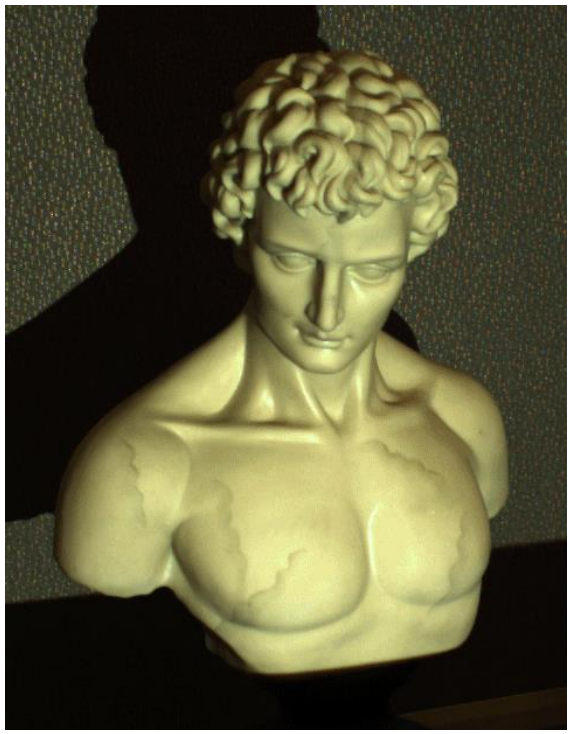


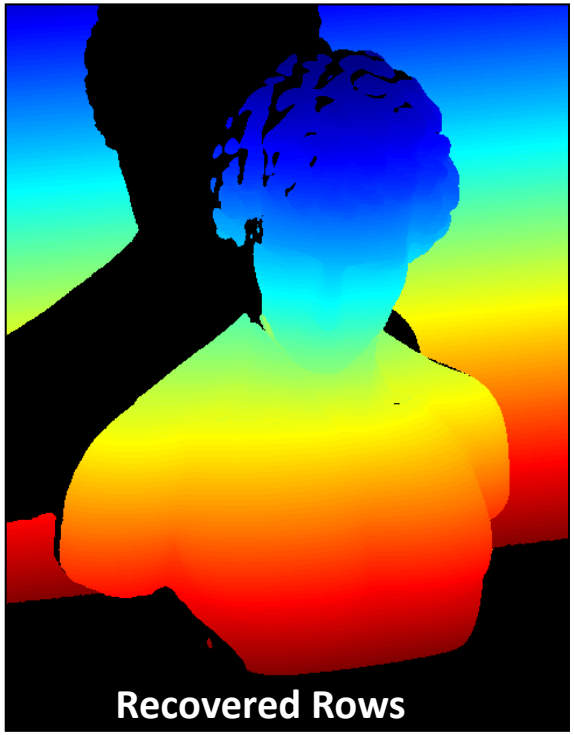
Projected over time

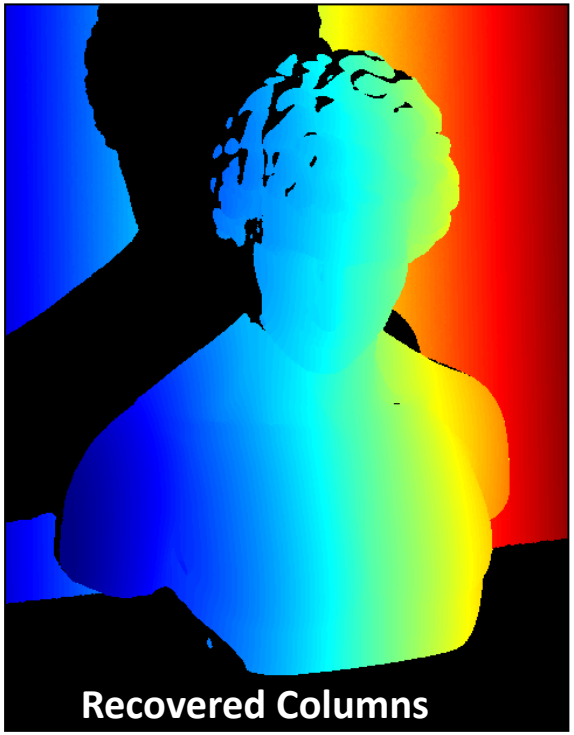
Codeword of this píxel: 1010010 → identifies the corresponding pattern stripe


lec24-figures.pdf - Adobe Reader
File Edit View Window Help
24 / 48
133%
Tools Sign Comment
01010001 = stripe 81
10:34 AM
4/24/2014

Gray Codes: Decoding Performance





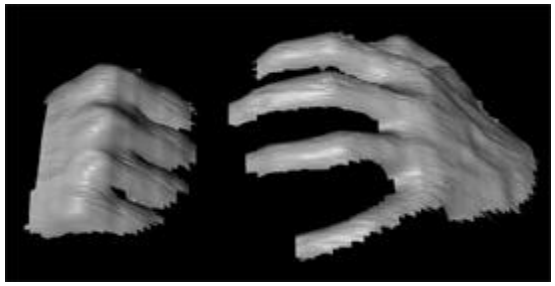


3D Reconstruction using Structured Light [Inokuchi 1984]

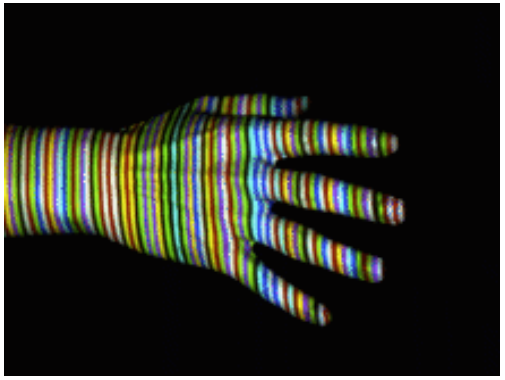
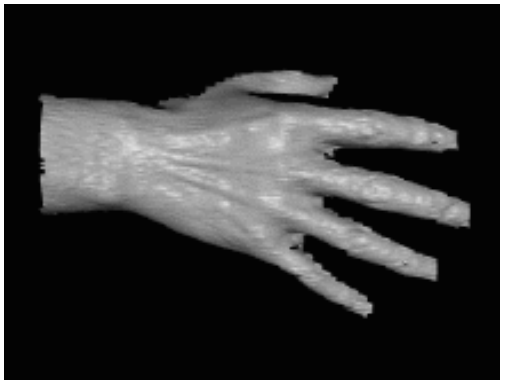
- Implemented using a total of 42 images (2 to measure dynamic range, 20 to encode rows, 20 to encode columns)
- Individual bits assigned by detecting if bit-plane (or its inverse) is brighter
- Decoding algorithm: Gray code → binary code → integer row/column index

More complex patterns





Works despite complex appearances

Works in real-time and on dynamic scenes

- Need very few images (one or two).
- But needs a more complex correspondence algorithm

Zhang et al

Output of acquisition

- Range image (or depth image),
- Disparity map,
- Cloud of point,
- Distribution of normals,
- Camera positions,
- Color/Texture (i.e., Uv-map),
- Poligonal mesh,

Range image

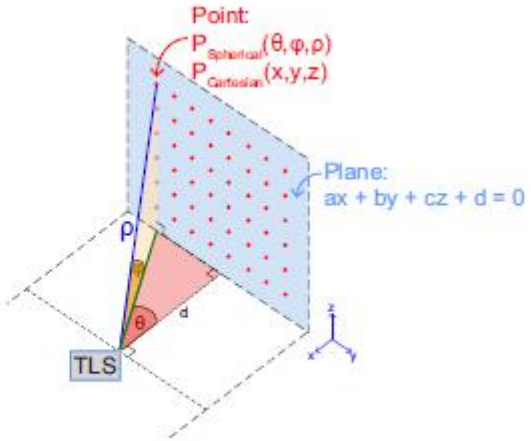
Intensity map

Range (of depth) map

Range map: each pixel encodes a depth value (i.e., the distance from sensor to the object in the scene). Lighter pixels corresponds to closer objects (and viceversa).

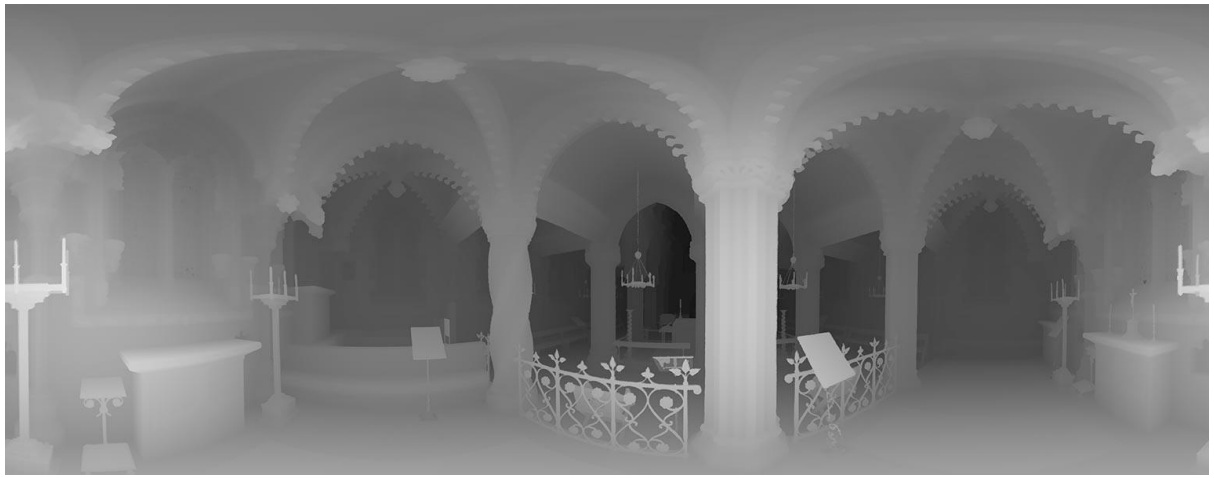
Range image

(a) FARO LS880 Phase based Terrestrial Laser Scanner

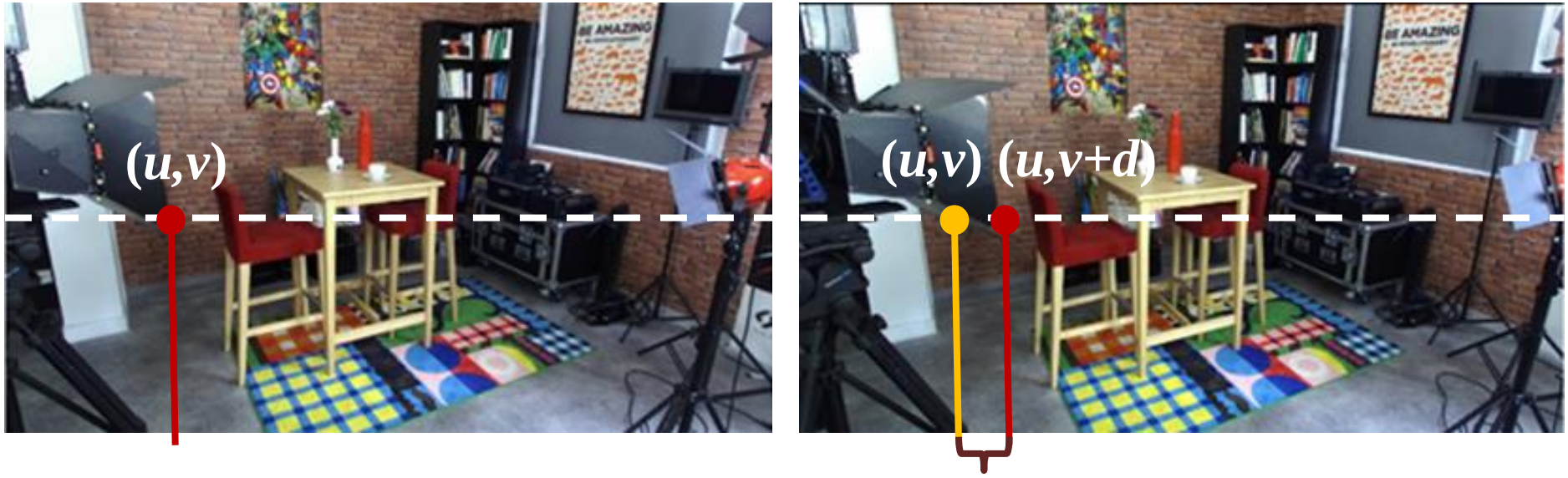


(b) Schematic representation of a point cloud acquisition. A point P is measured at the angular positions (θ,φ), with a range ρ.

Spherical Scanning

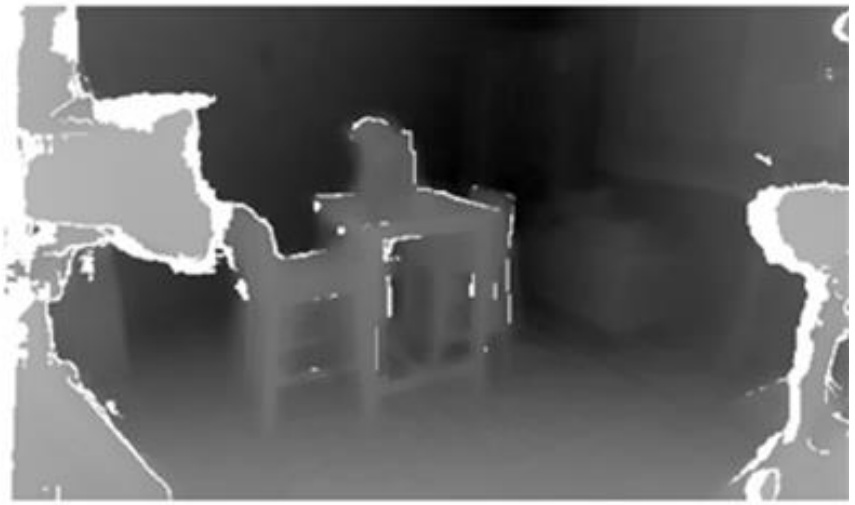


Disparity map



Disparity map

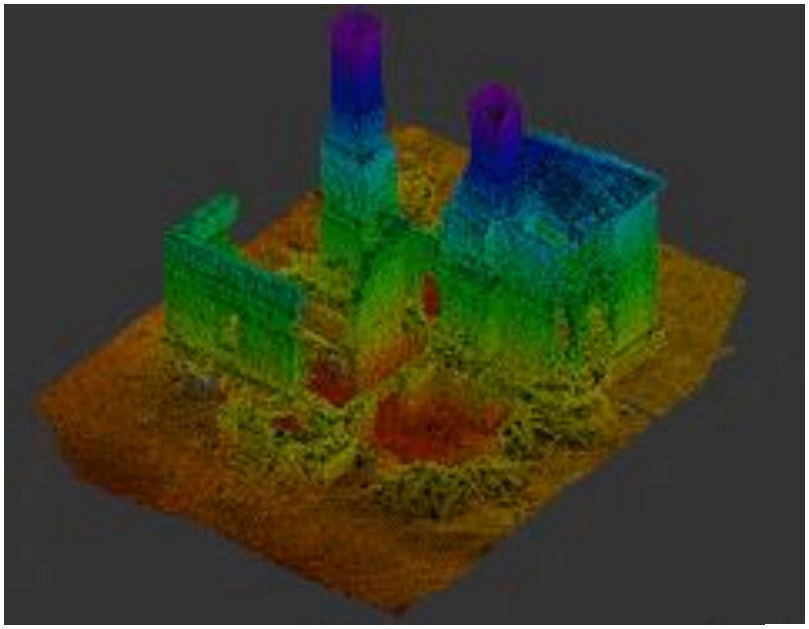
Intensity map

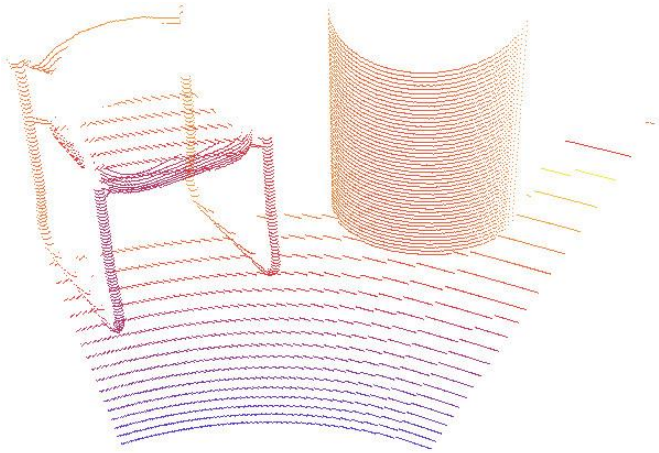


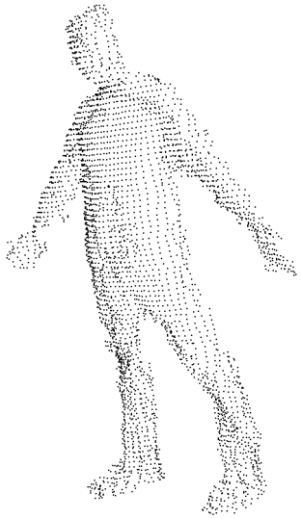
Disparity map

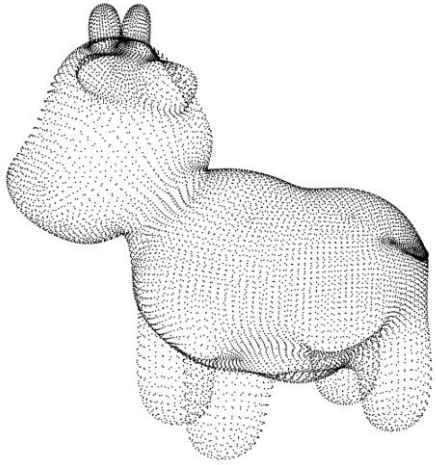
Disparity map: each pixel encodes a disparity value. Lighter pixels corresponds to closer objects (and viceversa).

Cloud of points

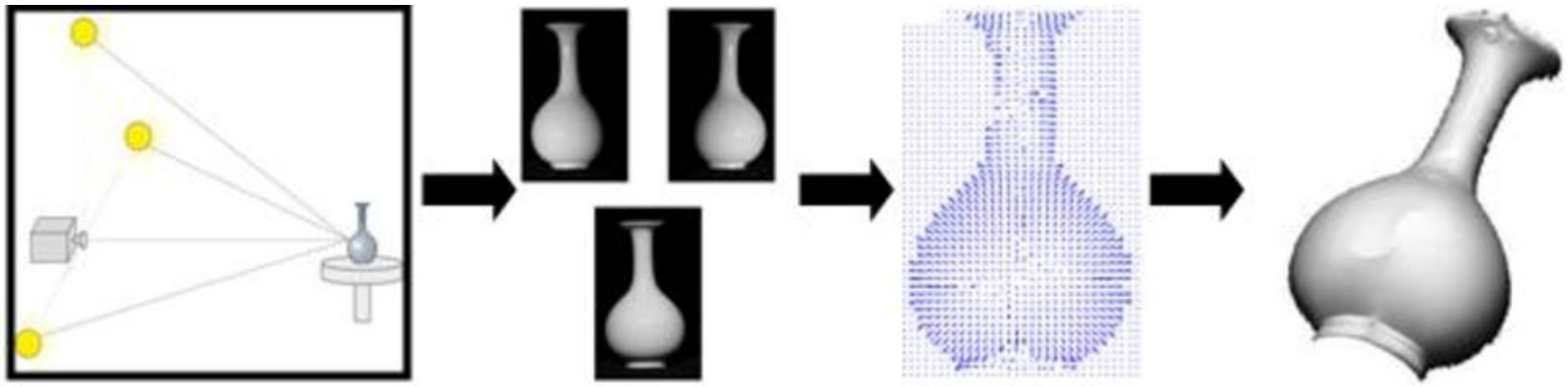


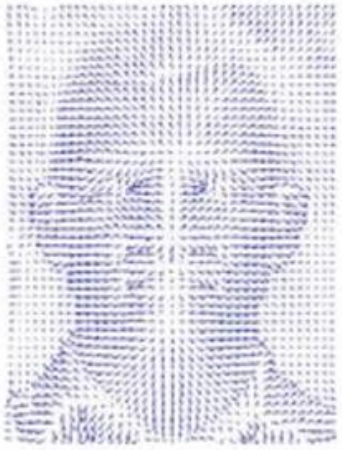






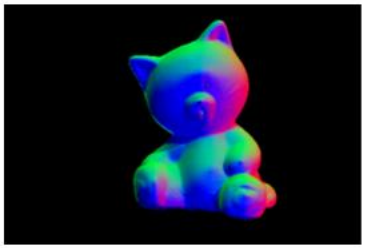
Distribution of normals





(a) Normal map

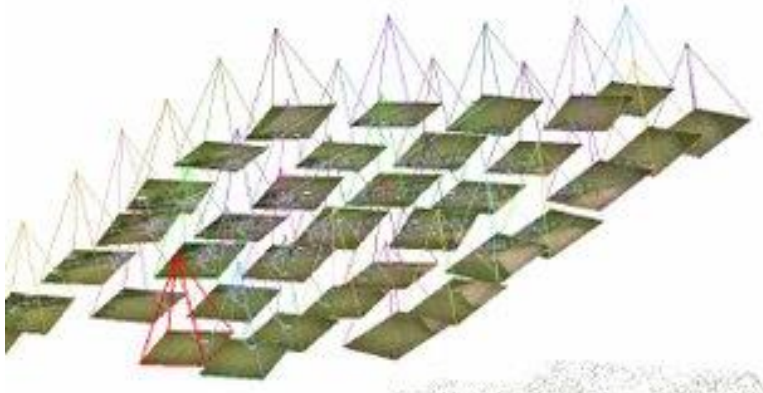
(b) Normal vectors



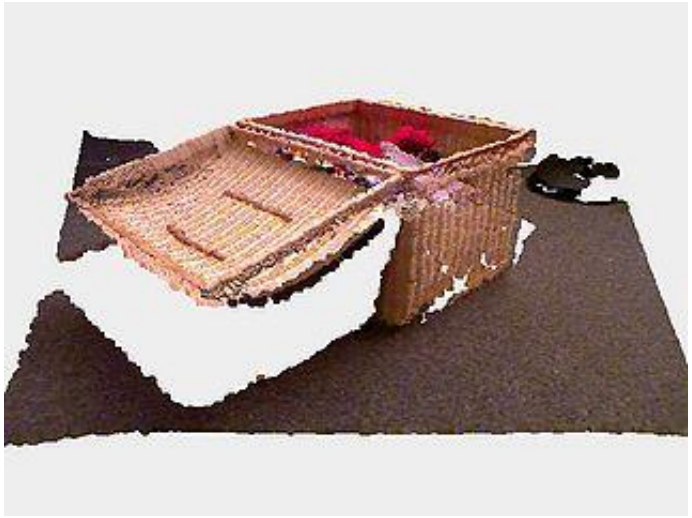
(a) Normal map

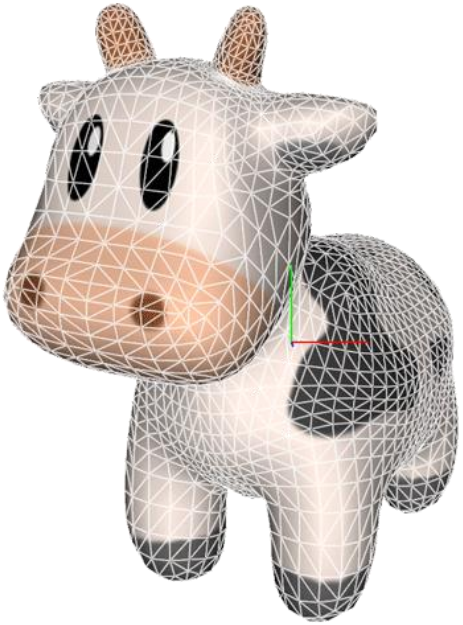
(b) Normal vectors

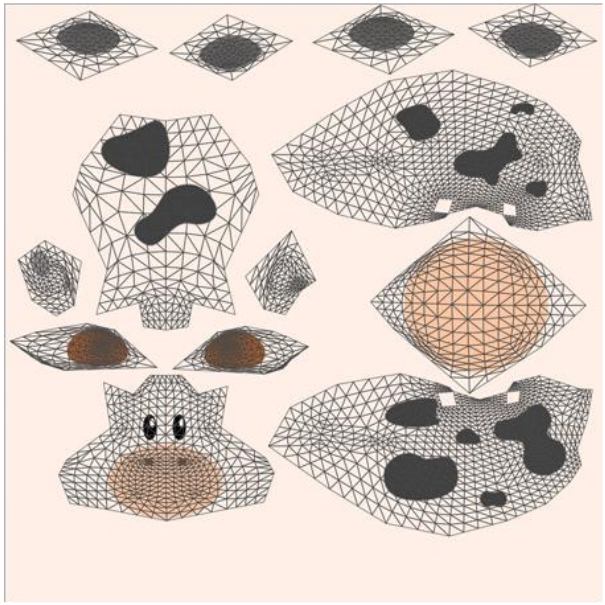
Camera positions



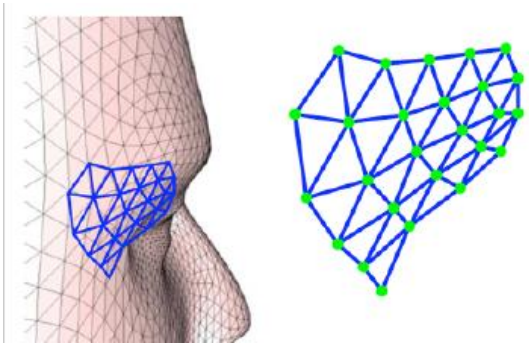
Color and texture







Polygonal meshes



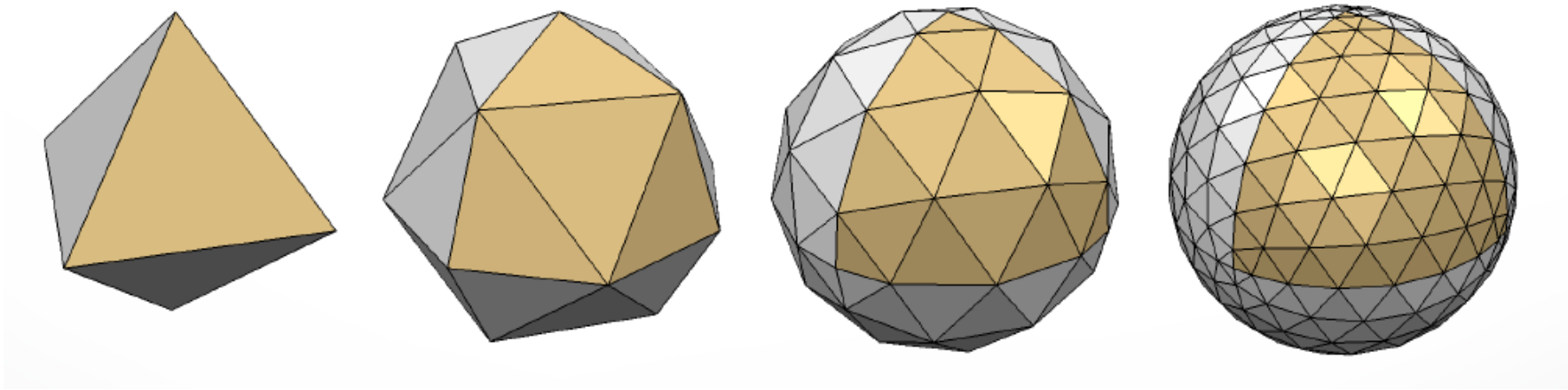
surface mesh: set of vertices, edges and faces (polygons) defining a polyhedral surface in embedded in 3D (discrete approximation of a shape)

- Polygonal meshes are the most used representation for 3D shapes

Polygonal meshes

- Polygonal meshes are the most used representation for 3D shapes:

  "Piecewise linear approximation --> error is $O(h^2)$"

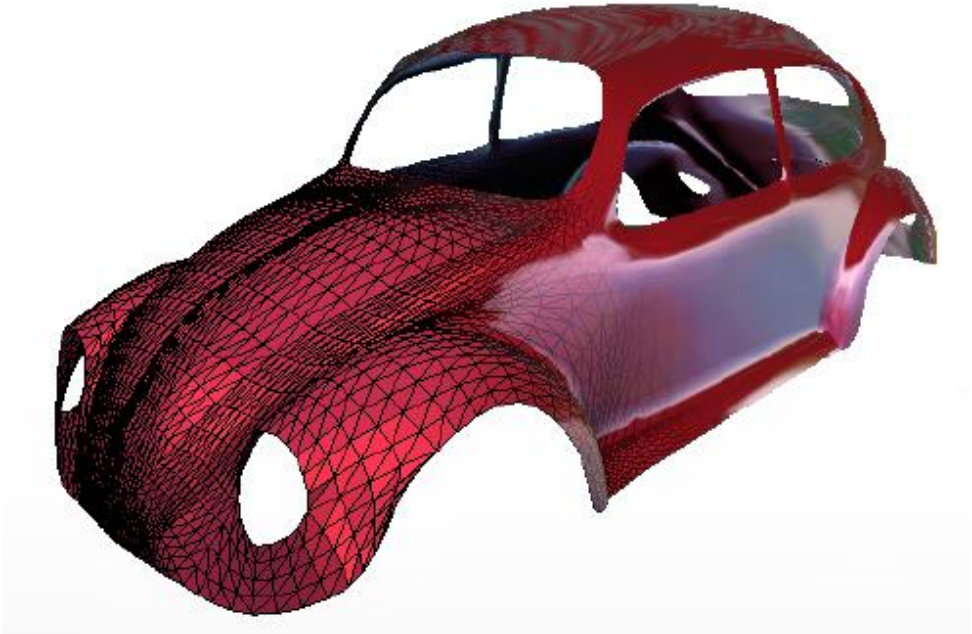


Error inversely proportional to #faces

Polygonal meshes

- Polygonal meshes are the most used representation for 3D shapes:

"Arbitrary topology surfaces"

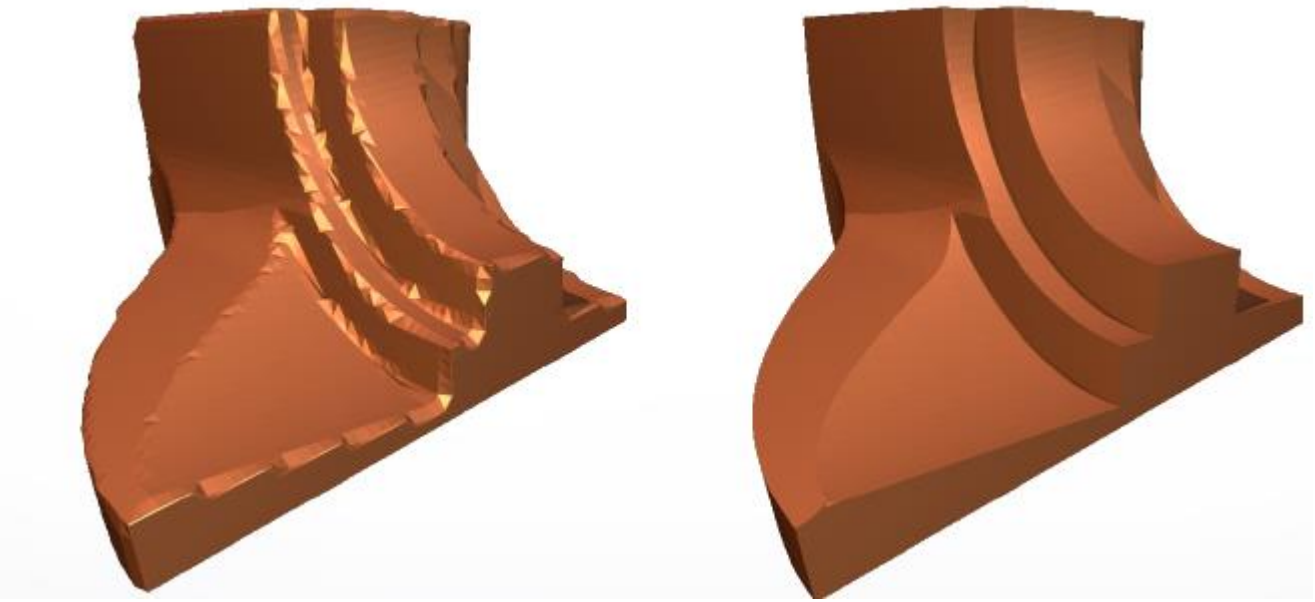


Allow subdivision for smoothness (today)

Polygonal meshes

- Polygonal meshes are the most used representation for 3D shapes:

"Piecewise smooth surfaces"

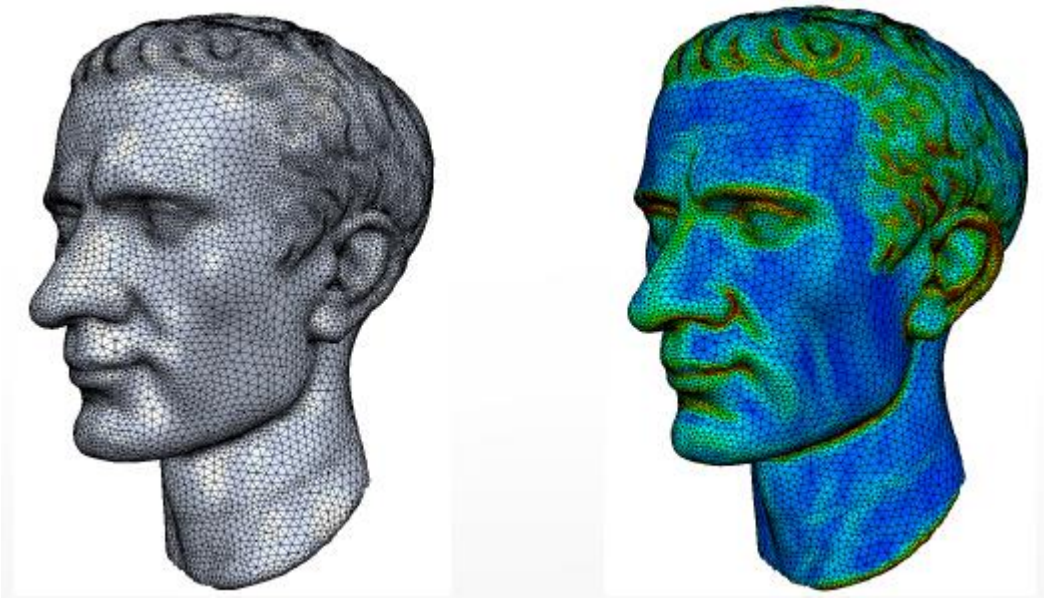


No need to have special rules for joining different patches

Polygonal meshes

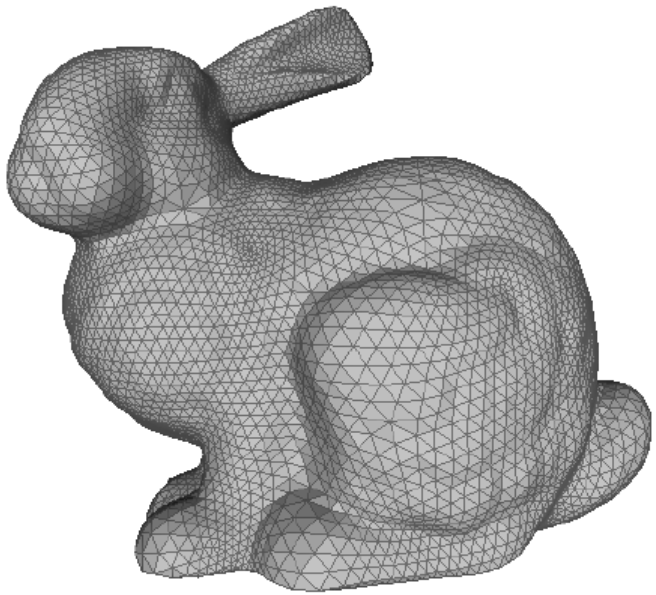
- Polygonal meshes are the most used representation for 3D shapes:

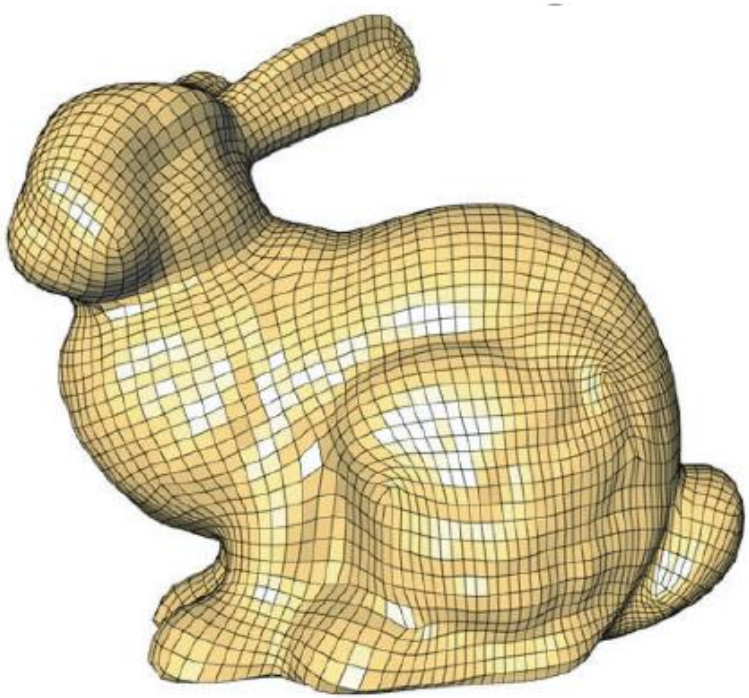
"Adaptive sampling"



Can add resolution only where necessary

Triangular meshes





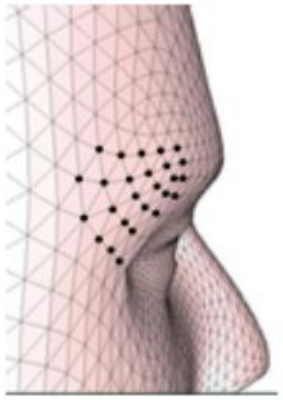
- Simplest piecewise linear element
- Easy to reconstruct from point clouds
- Easy to represent

- Quad meshes often used in animation
- Typically require some handtuning in reconstruction
- Can provide more flexibility for deformation

Triangular mesh

- Two main **components**:

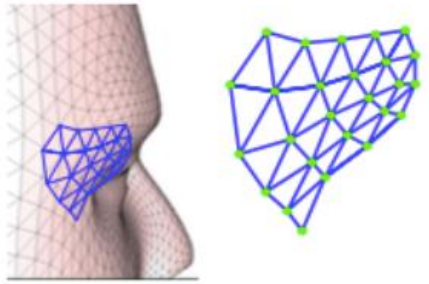
Geometry



vertex coordinates

geometric structure

"Connectivity": the underlying triangulation



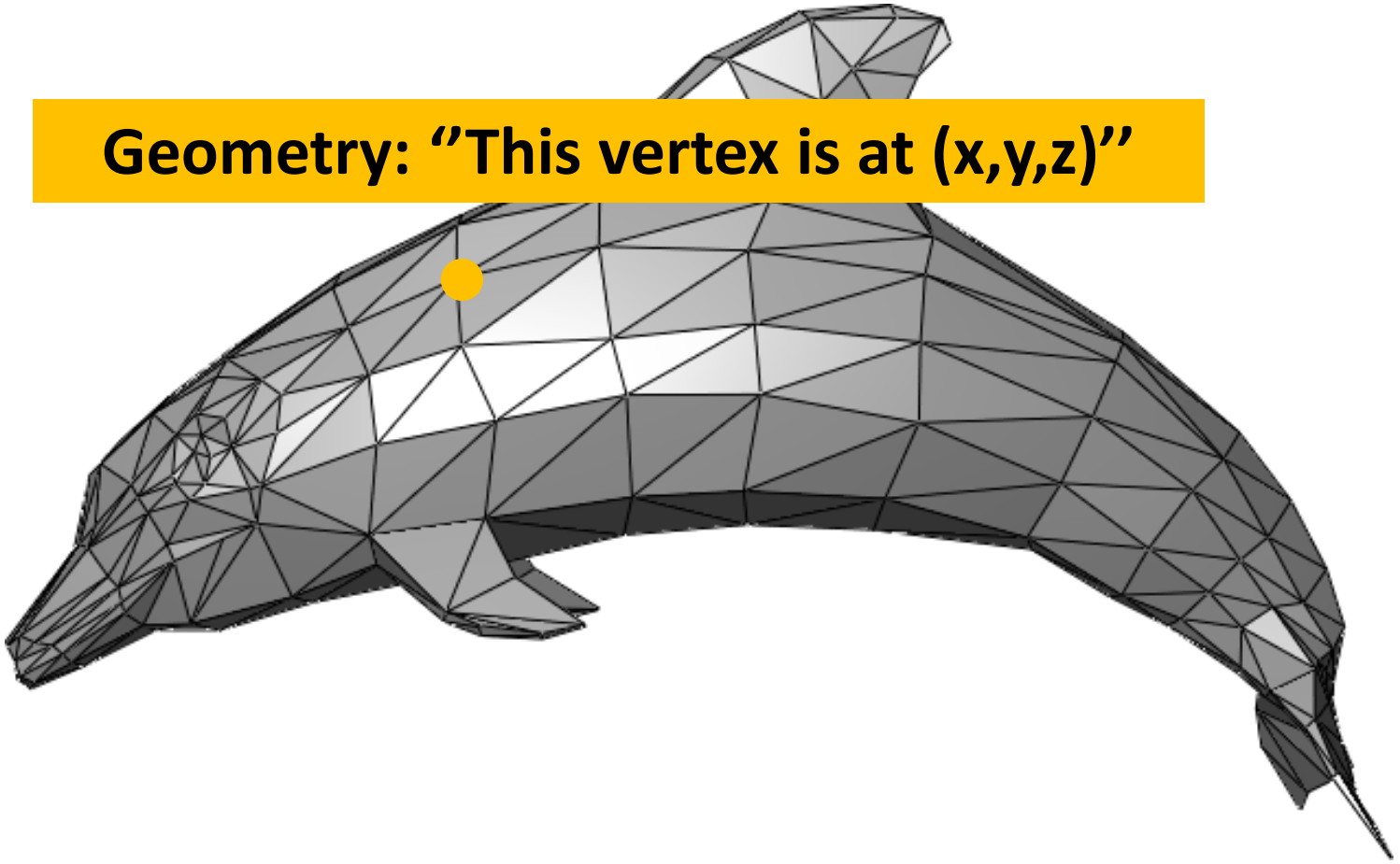
incidence relations between triangles, vertices and edges

combinatorial structure

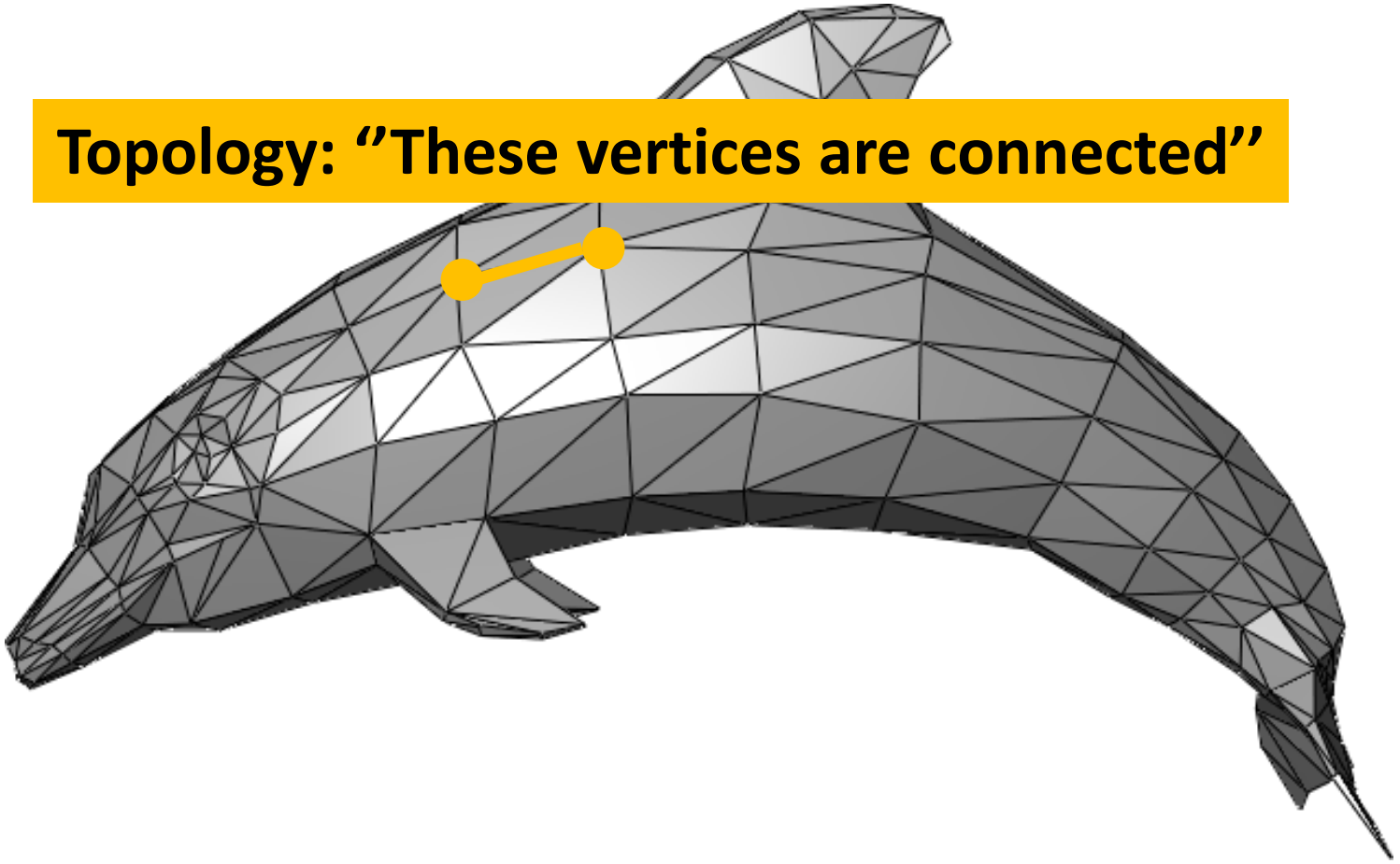
Topology [t*uh*-**pol**-*uh*-jee]:
The study of geometric properties that remain invariant under certain transformations

http://dictionary.reference.com/browse/topology?s=t

Geometry vs. Topology



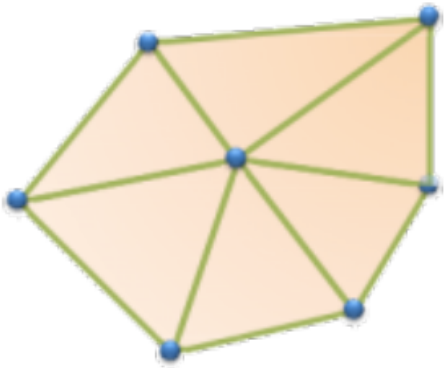
Geometry vs. Topology



Two components

- Geometry: vertex positions

$$\mathcal{P} = \{p_1, p_2, ..., p_n\}, \quad p_i \in \mathbb{R}^3$$

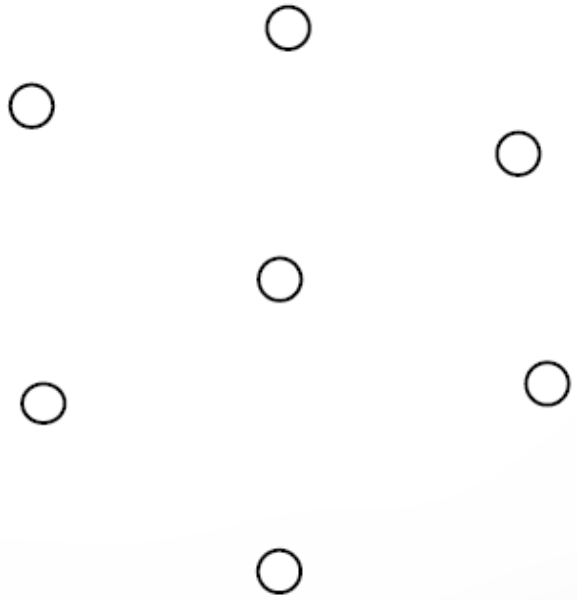


- Connectivity:
  - Vertices: $\mathcal{V} = \{v_1, v_2, ..., v_n\}$
  - Edges: $\mathcal{E} = \{e_1, e_2, ..., e_m\}, \quad e_i \in \mathcal{V} \times \mathcal{V}$
  - Faces: $\mathcal{F} = \{f_1, f_2, ..., f_k\}, \quad f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$

Two components

$$\mathcal{M} = (\{\mathbf{v}_i\}, \{e_j\}, \{f_k\})$$

geometry $\mathbf{v}_i \in \mathbb{R}^3$

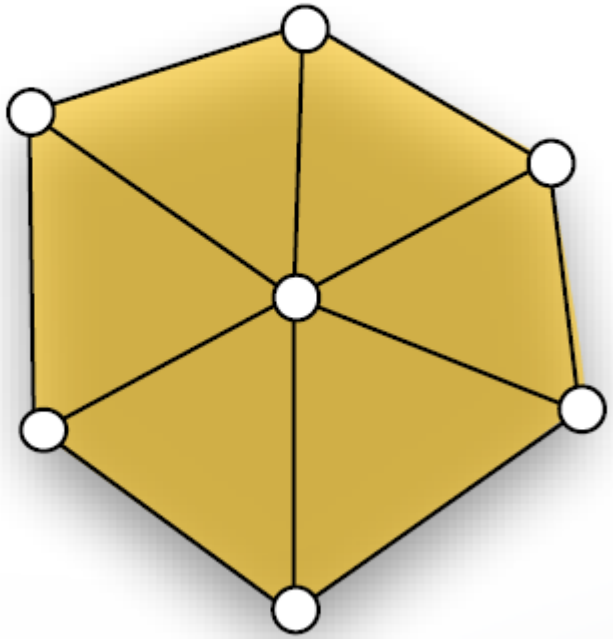


Two components

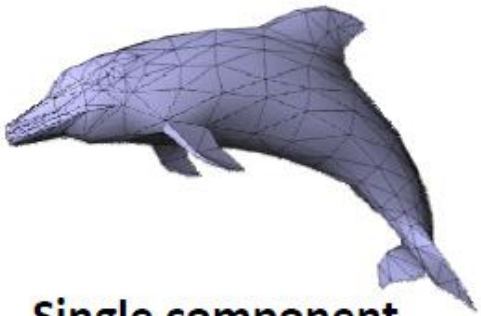
$$\mathcal{M} = (\{\mathbf{v}_i\}, \{e_j\}, \{f_k\})$$

geometry $\mathbf{v}_i \in \mathbb{R}^3$

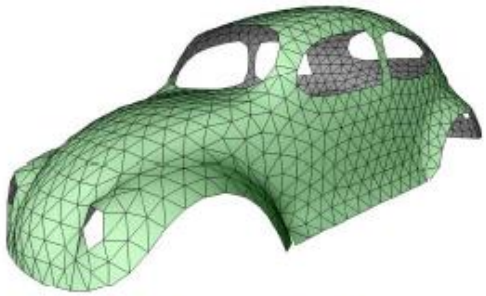
topology $e_i, f_i \subset \mathbb{R}^3$



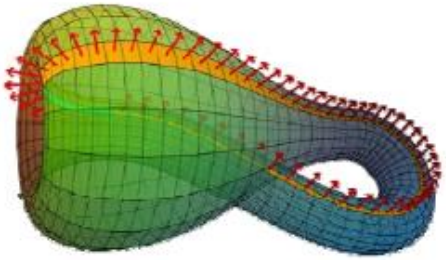
Mesh Zoo



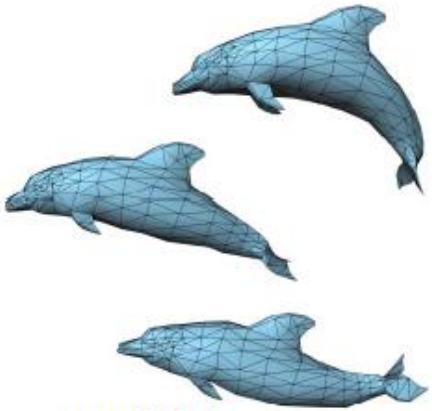
Single component, closed, triangular, orientable manifold



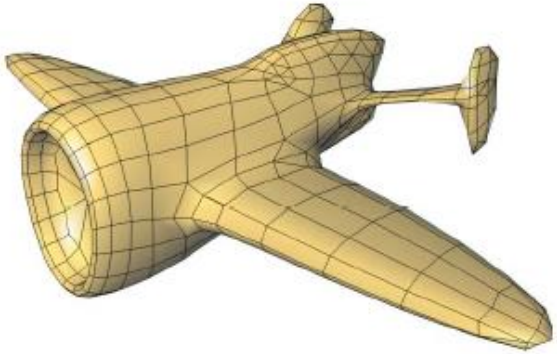
With boundaries



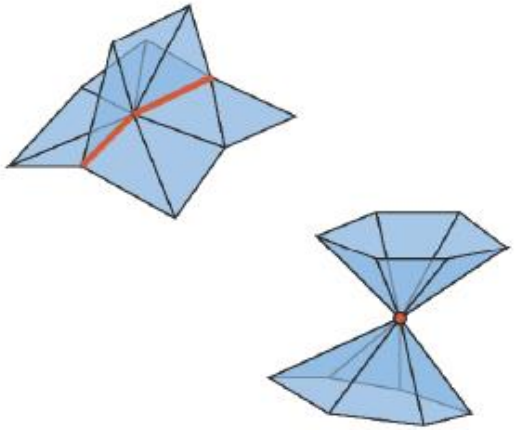
Not orientable



Multiple components



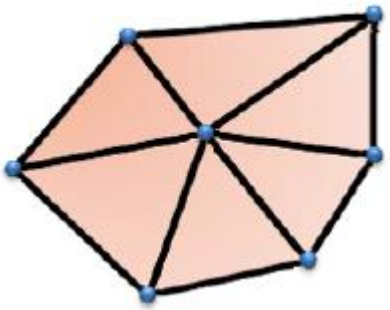
Not only triangles

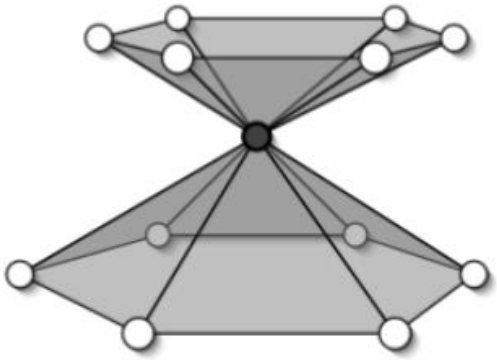


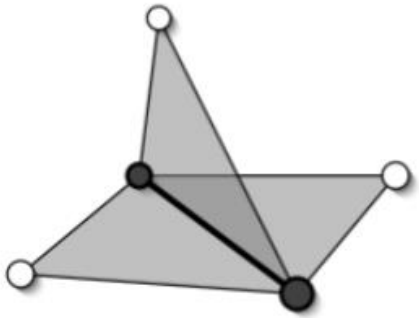
Non manifold

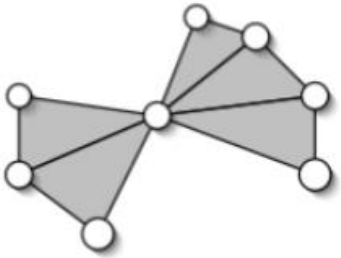
Triangular manifold mesh

- Each Edge is adjacent to at most 2 faces
- Each vertex has a disk-shaped neighborhood



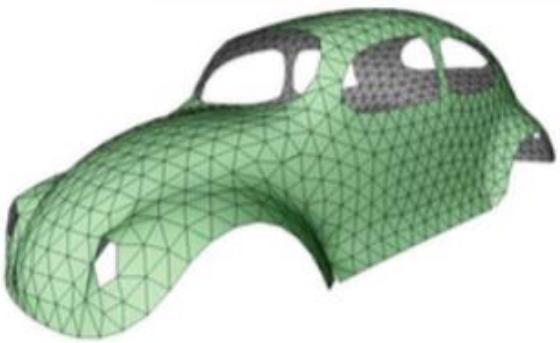






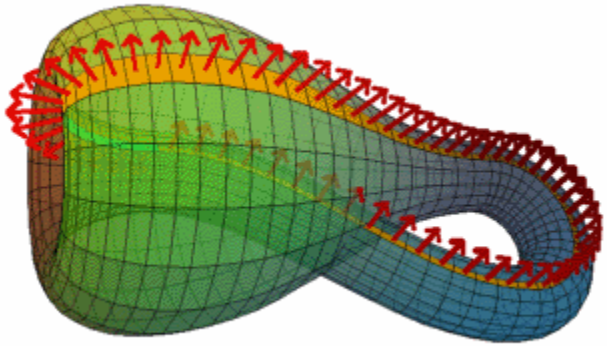
Non-manifold

With boundaries, Not orientable



With boundaries

Boundary edge: adjacent to exactly one face

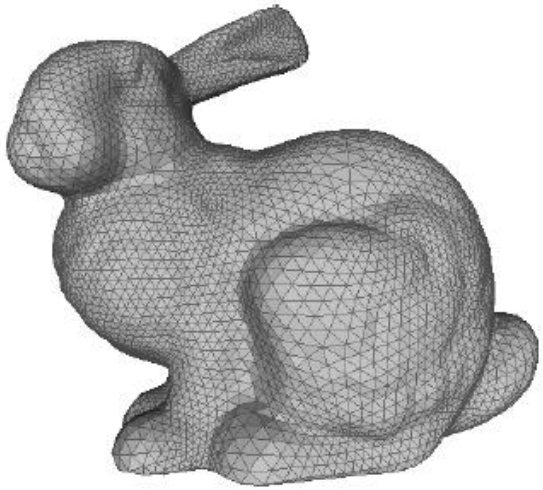


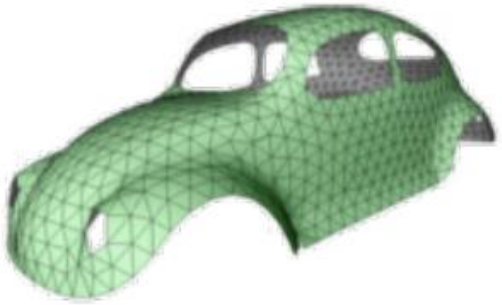
Not orientable

Orientable surface: possible to assign a consistent normal orientation (e.g. outward)

Single component

Single component, closed, triangular, orientable manifold

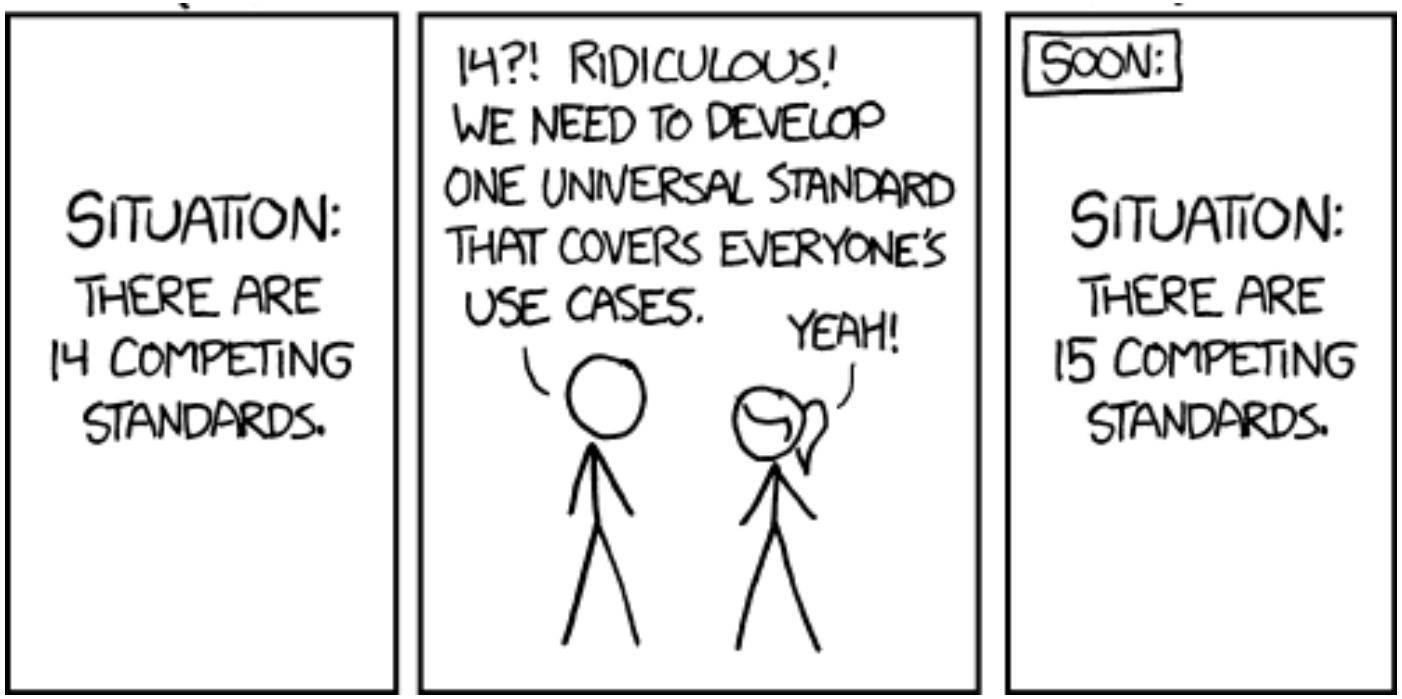




Possibly with boundaries

Closed (without boundaries), Single component, 2D-manifold, is often called Watertight mesh

File Formats



(xkcd.com)

File Formats

- 3DS - 3D Studio Max file format
- OBJ - Another file format for 3D objects
- MA, MB - Maya file formats
- 3DX - Rinoceros file format
- BLEND - Blender file format
- DAE - COLLADA file format (Khornos)
- FBX - Autodesk interchange file format
- X - Direct X object
- SMD - good for animations (by Valve)
- MD3 - quake 3 vertex animations
- DEM - Digital Elevation Models
- DXF - (exchange format, Autodesk's AutoCAD)
- FIG - Used by REND386/AVRIL
- FLT - MulitGen Inc.'s OpenFlight format
- HDF - Hierarchical Data Format
- IGES - Initial Graphics Exchange Specification
- IV - Open Inventor File Format Info
- LWO, LWB & LWS - Lightwave 3D file formats
- MAZ - Used by Division's dVS/dVISE
- MGF - Materials and Geometry Format
- MSDL - Manchester Scene Description Language
- 3DML - by Flatland inc.
- C4D – Cinema 4D file format
- SLDPTR - SolidWork "part"
- WINGS - Wings3D object
- NFF - Used by Sense8's WorldToolKit
- SKP - Google sketch up
- KMZ - Google Earth model
- OFF - A general 3D mesh Object File Format
- OOGL - Object Oriented Graphics Library
- PLG - Used by REND386/AVRIL
- POV - “persistence of vision” ray-tracer
- QD3D - Apple's QuickDraw 3D Metafile format
- TDDD - for Imagine & Turbo Silver ray-tracers
- NFF & ENFF - (Extended) Neutral File Format
- VIZ - Used by Division's dVS/dVISE
- VRML, VRML97 - Virtual Reality Modeling Language (RIP)
- X3D - tentato successore di VRML
- PLY - introdotto by Cyberware – tipic. dati range scan
- DICOM - Dalla casa omonima – tipic. dati volum, es CAT scan
- Renderman - per l'omonimo visualizzatore
- RWX - RenderWare Object
- Z3D - ZModeler File format
- etc, etc, etc...

File formats for 3D meshes

(Tower of Babel!)

- Different Types
  - Usually triangular or polygonal meshes
  - Sometimes contains also scene description
    - Cameras, model-view matrices, hierarchies…
  - Even with Polygonal mesh many variations
    - Direct, Indexed, Half edges
    - Per face, per vertex attributes
- Some are open standards, others are commercial soft
  - Some are easy to parse, others very difficoulr
  - Some of them pretend to be ver*y generic*: «3D esperanto»
- Format
  - ASCII o binary
  - Sometimes also compressed

File formats for 3D meshes

more commons

less commons

.OBJ (wavefront)

- ☺ Most common
- ☺ indexed, normals , uv-mapping
- ☹ no colors (only per face material index)
- ☹ no skinning o ani

.SMD (VALVE

- ☺ Skeletal animations + skinning
- ☺ normals , uv-mapping
- ☹ no indexed
- ☹ no colors

.MD3 (Quake, IDsoft)

- ☺ vertex animations, normals
- ☹ no colors

.PLY (cyberware)

- ☺ extendable
- ☹ “diffuse in acamic world”

.3DS (AUTODESK.

- ☺ color on faces/vertices, uv-mapping, indexed, materials, textures…
- ☹ no: normals
- ☹ Maximum number of vertices(64K)

.COLLADA (KHRONOS GROUP

- ☺ super complete!
- ☺ designed to be extendible
- ☺ open standard
- ☹ difficoult to parse

.FBX (AUTODESK.

- ☺ complete with animation
- ☹ difficoult to parse

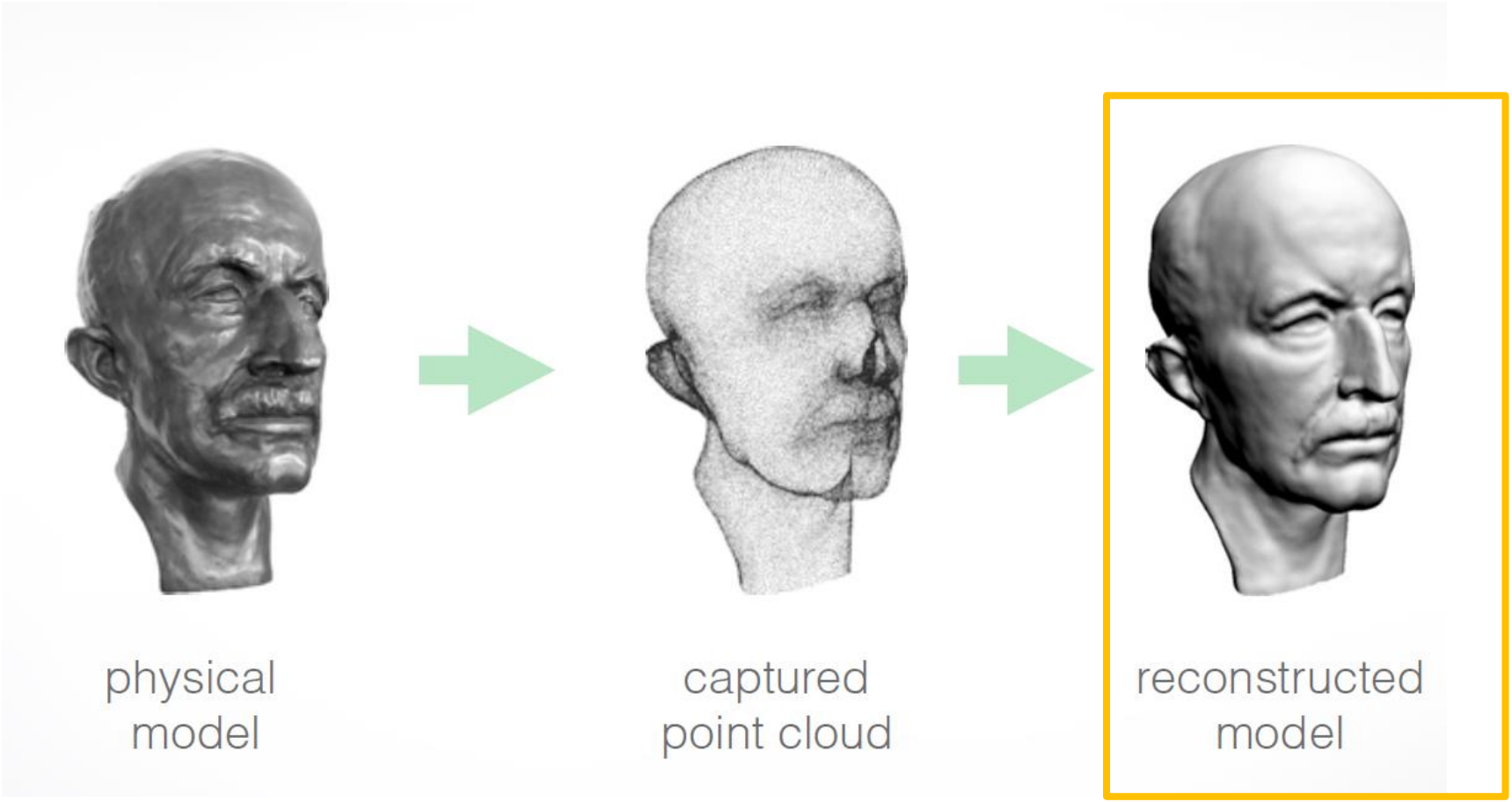
.MA / **.MB** (AUTODESK.

- ☺ complete with animation
- ☹ difficoult to parse

simple

complex

Surface reconstruction



Polygonal mesh

Input

Set of irregular sample points

- with or without normals
- examples: multi-view stereo, union of range scan vertices

Set of range scans

- each scan is a regular quad or tri-mesh
- normal vectors can be obtained through local connectivity

Surface reconstruction

- **Two approaches:**

Explicit

Local surface connectivity estimation

Point interpolation

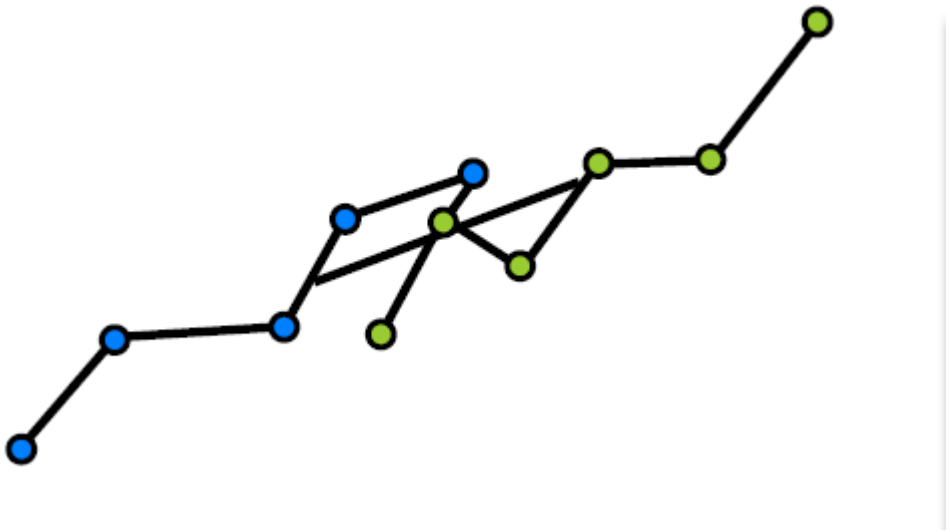
Implicit

Signed distance function estimation

Mesh approximation

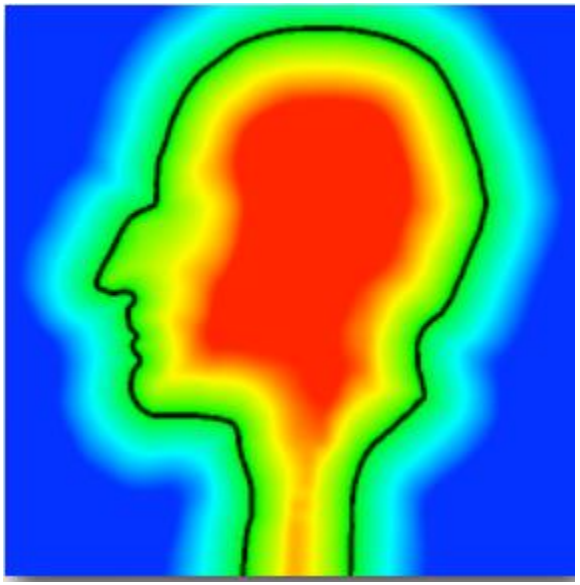
Explicit methods

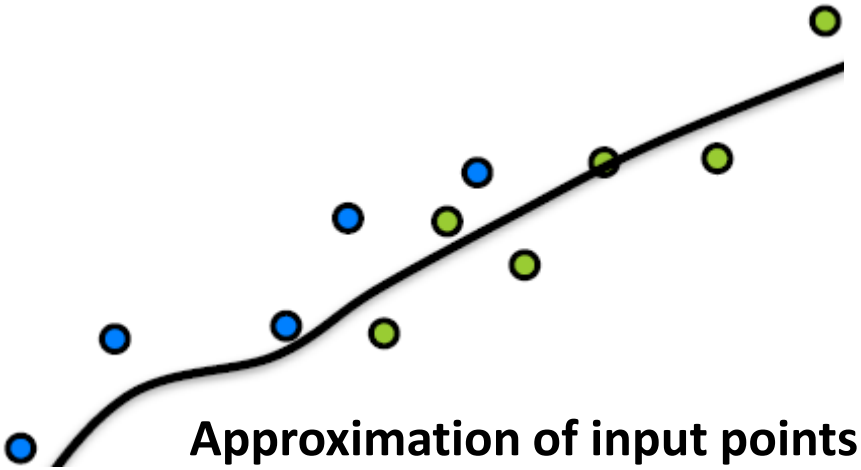
- Connect sample points by triangles
- Exact interpolation of sample points
- Bad for noisy or misaligned data
- Can lead to holes or non-manifold situations



Implicit methods

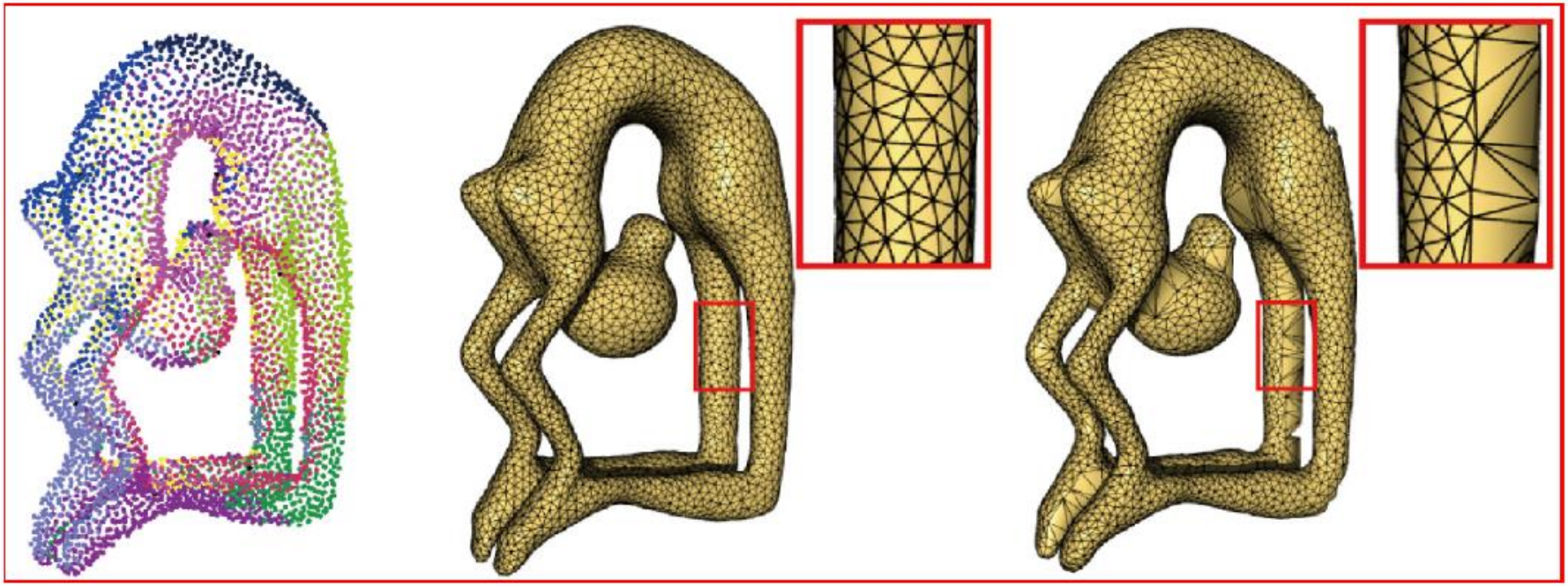
1. estimate a signed distance function(SDF);
2. extract 0-level set mesh using Marching Cubes





Ouput is a **Watertight manifold** by construction !

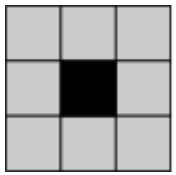
Explicit vs Implicit



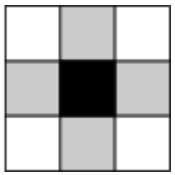
Input Implicit Explicit

Esplicit method

- Mesh reconstruction from **range image**,
  - **Idea**: points are on a regular grid where the connectivity can be inherited from the pixel neighbourhood



8-neigh



4-neigh

- Zippering range scans,
  - **Idea**: “Zipper” several scans to one single model

Mesh from range image

- Points are on a regular grid where the connectivity can be inherited from the pixel neighbourhood,

  BUT…

  1) Not all the pixels on the range image are the projection of a point on the 3D space!

  ➡ a binary mask can be used to define valid points,

  2) Nearby pixels should not correspond to nearby points on the 3D space!

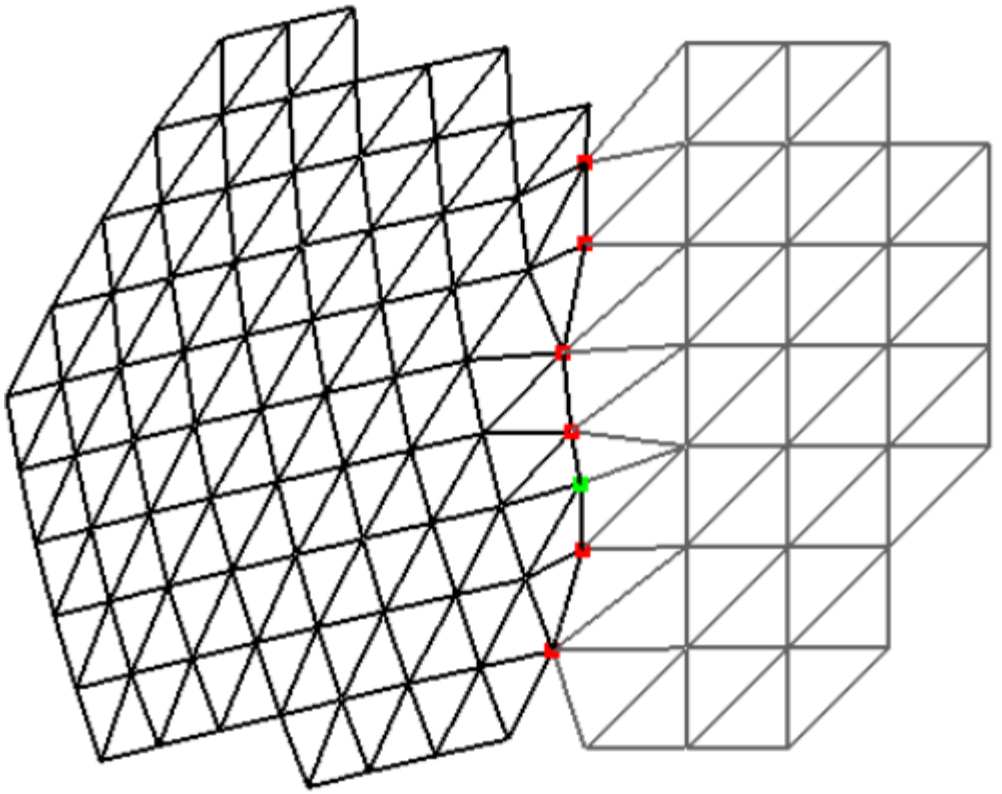
  ➡ a robust strategy can be used to remove long edges.

See Matlab script available from lab section!

Zippering range scans

- “Zipper” several scans to one single model

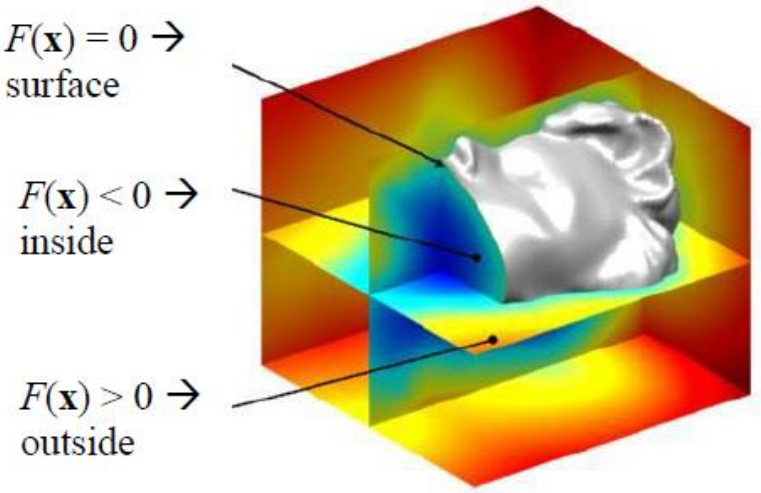
1. Project & insert boundary vertices
2. Intersect boundary edges
3. Discard overlap region
4. Locally optimize triangulation

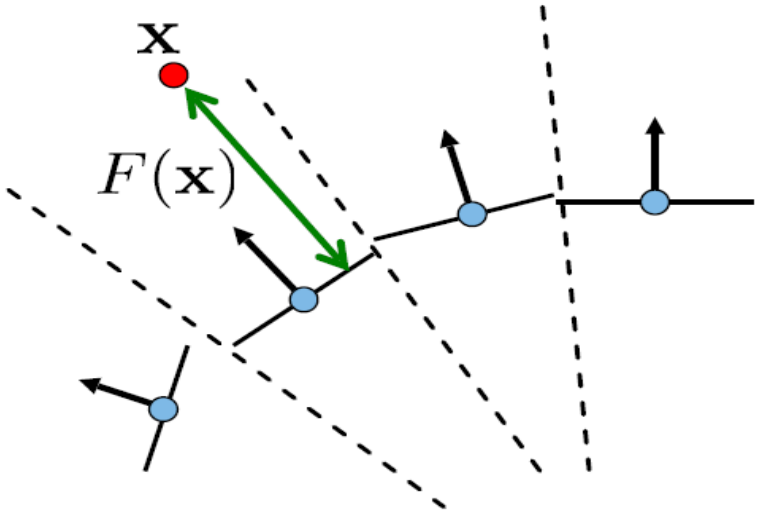


Not much used in practice!

Implicit methods

- Several methods for signed distance computation (SDF)
  - **Marching Cube**: classical method
  - **Poisson method**: the currently most used method

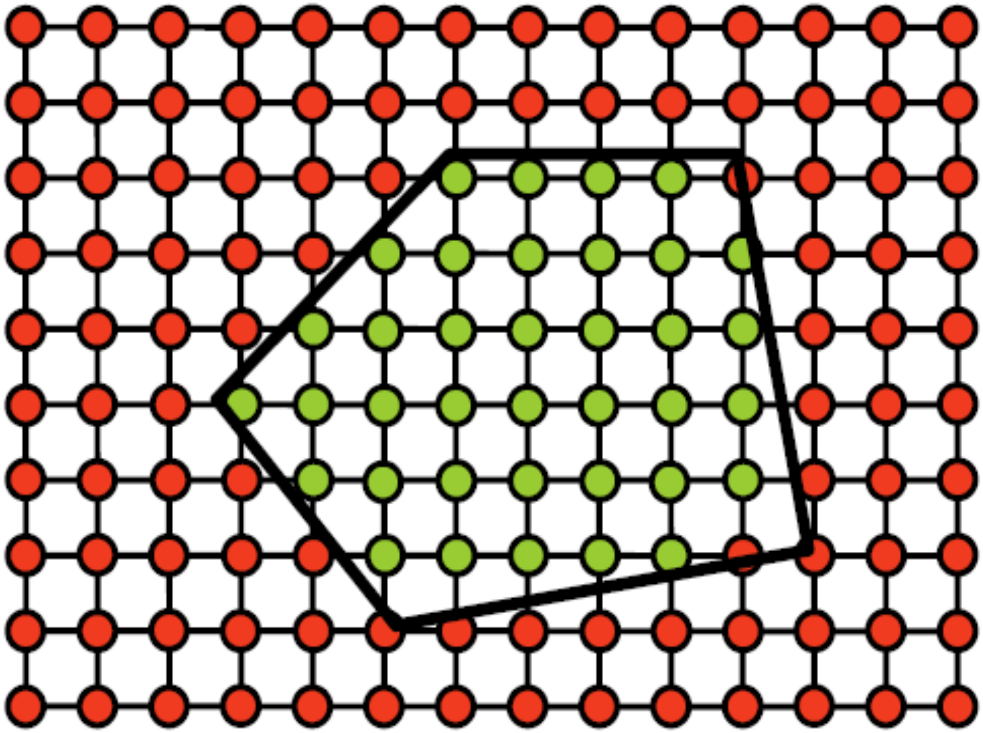




E.g.: signed distance to the tangent plane of the closest point

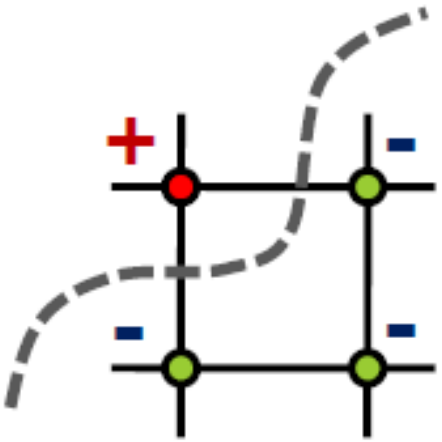
Marching cube

- **Idea**: sample the SDF



Marching cube

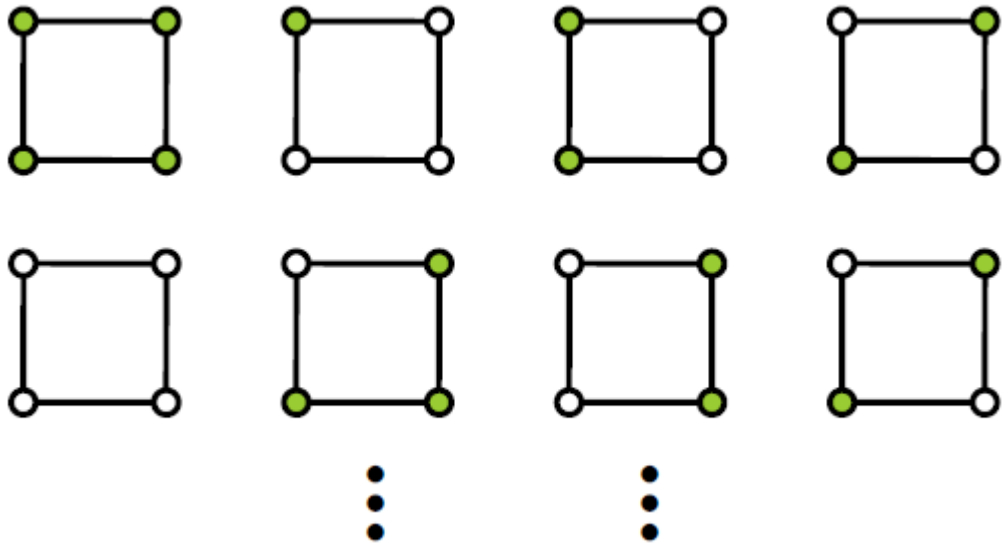
- **Idea**: sample the SDF



$\bullet F(\mathbf{x}) < 0$

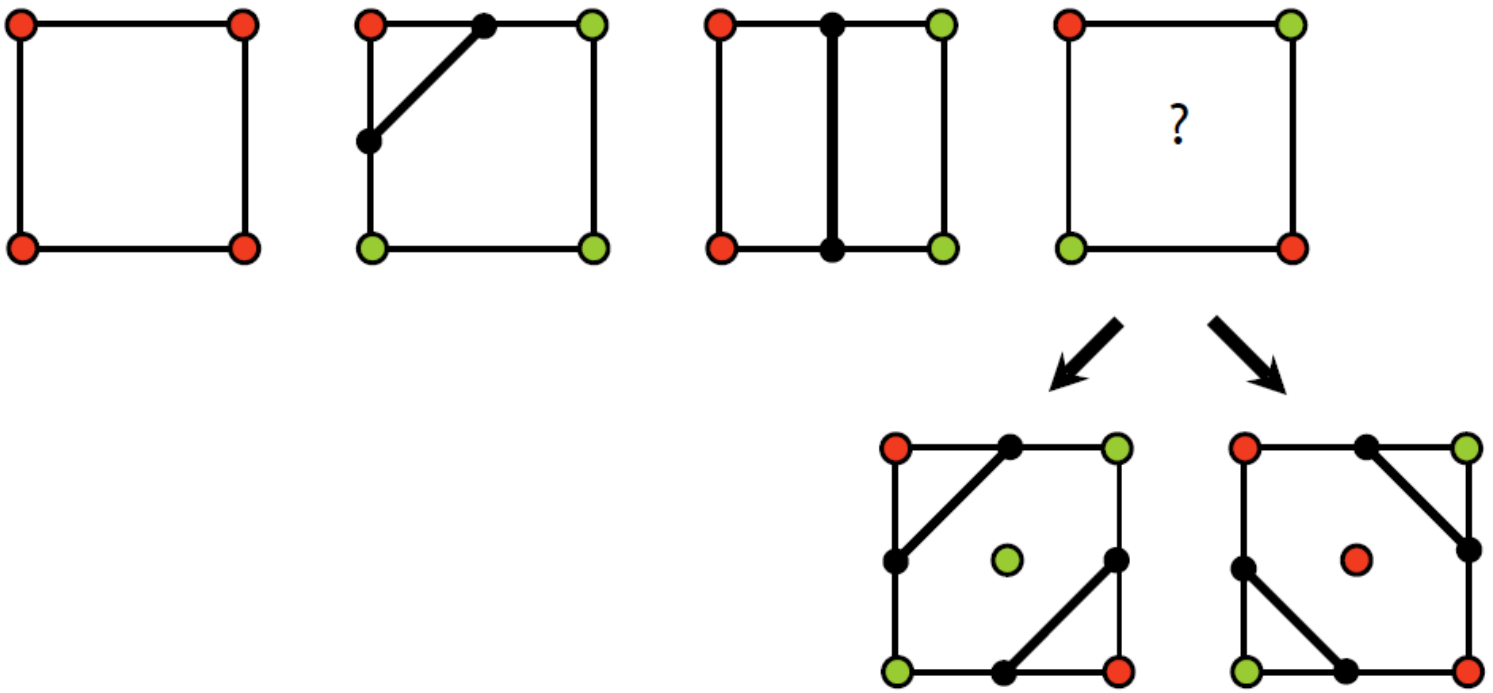
Marching square

- 2^4=16 different configurations in 2D
- 4 equivalence classes (up to rotational and reflection symmetry + complement)

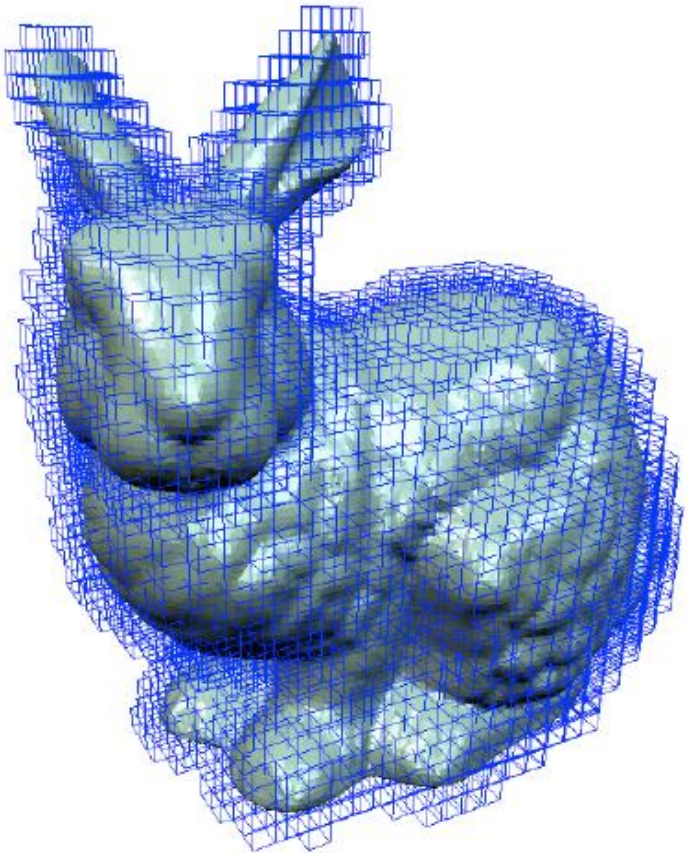


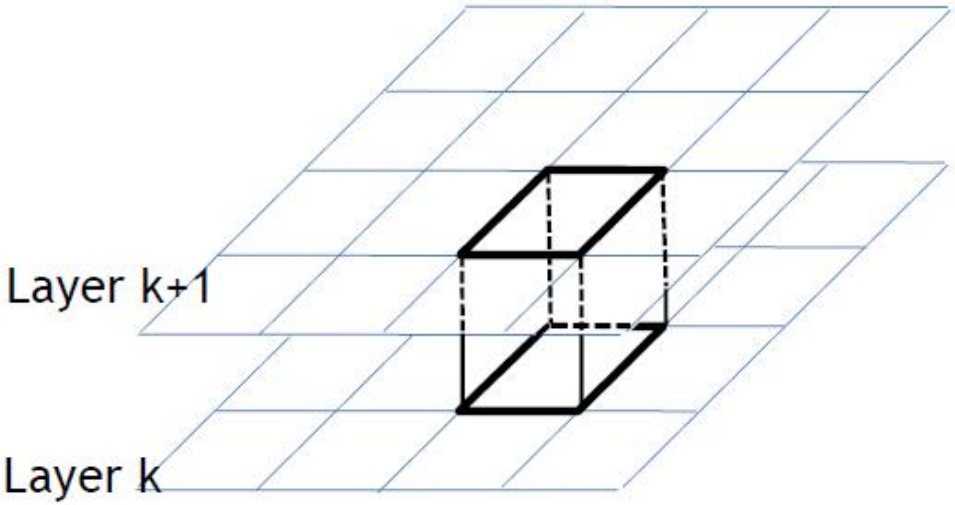
Marching square

- 4 equivalence classes (up to rotational and reflection symmetry + complement)



Marching cube



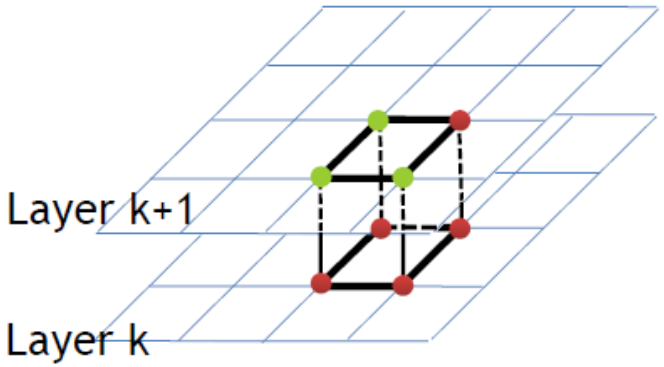


3D case!

Marching cube

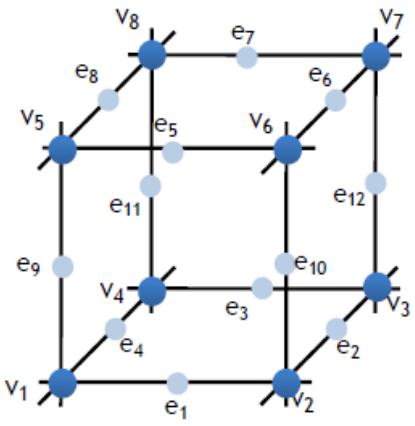
Marching Cubes (Lorensen and Cline 1987)

1. Load 4 layers of the grid into memory
2. Create a cube whose vertices lie on the two middle layers
3. Classify the vertices of the cube according to the implicit function (inside, outside or on the surface)

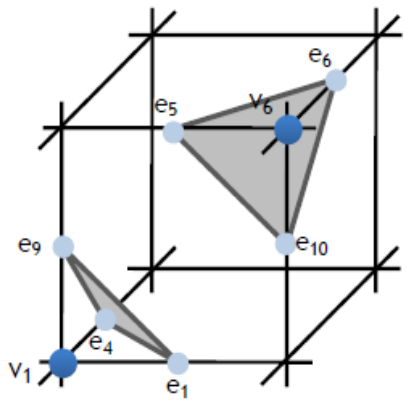


Marching cube

Compute case index. We have $2^8 = 256$ cases (0/1 for each of the eight vertices) – can store as 8 bit (1 byte) index.



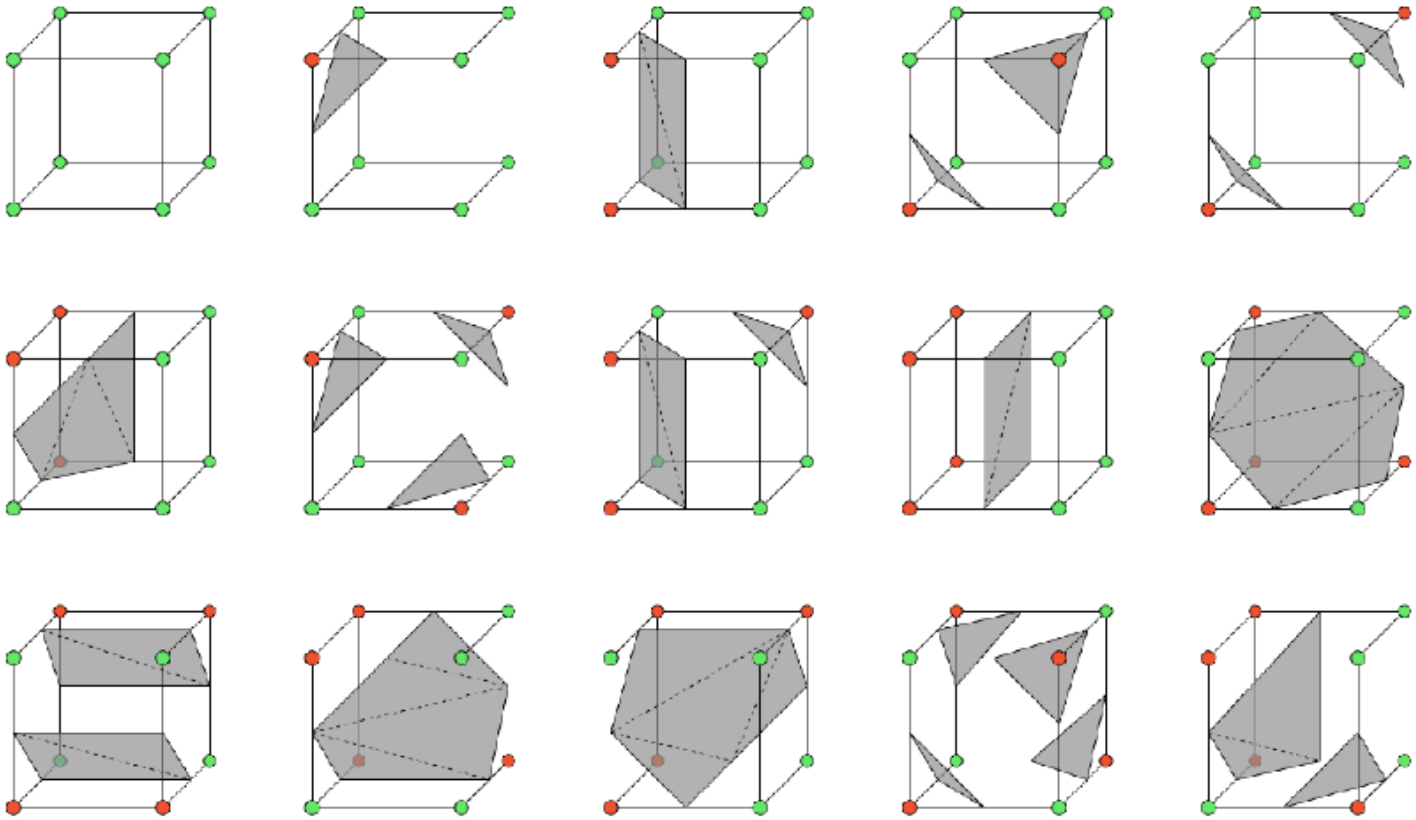
index = | v_1 | v_2 | v_3 | v_4 | v_5 | v_6 | v_7 | v_8 |



index = | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | = 33

Marching cube

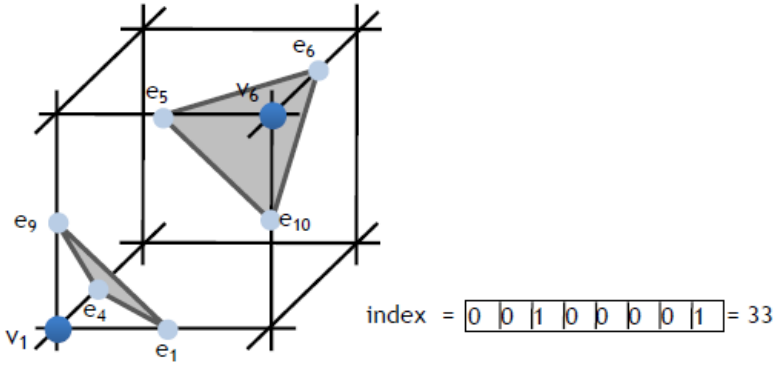
Unique cases (by rotation, reflection and complement)



Marching cube

Using the case index, retrieve the connectivity in the look-up table

- Example: the entry for index 33 in the look-up table indicates that the cut edges are e_1; e_4; e_5; e_6; e_9 and e_{10} ; the output triangles are (e_1; e_9; e_4) and (e_5; e_{10}; e_6).

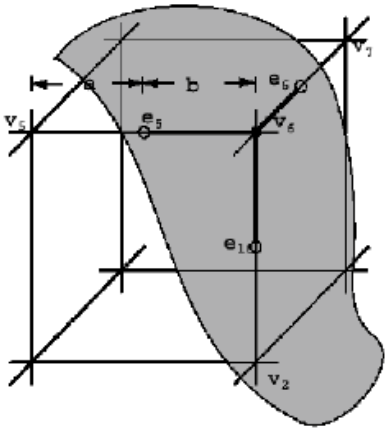


Marching cube

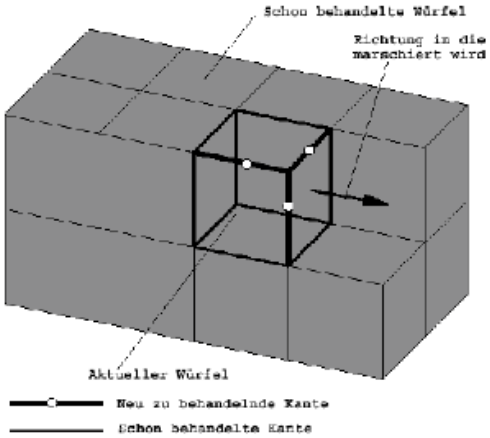
Compute the position of the cut vertices by linear interpolation:

$$\mathbf{v}_s = t\mathbf{v}_a + (1-t)\mathbf{v}_b$$

$$t = \frac{F(\mathbf{v}_b)}{F(\mathbf{v}_b) - F(\mathbf{v}_a)}$$



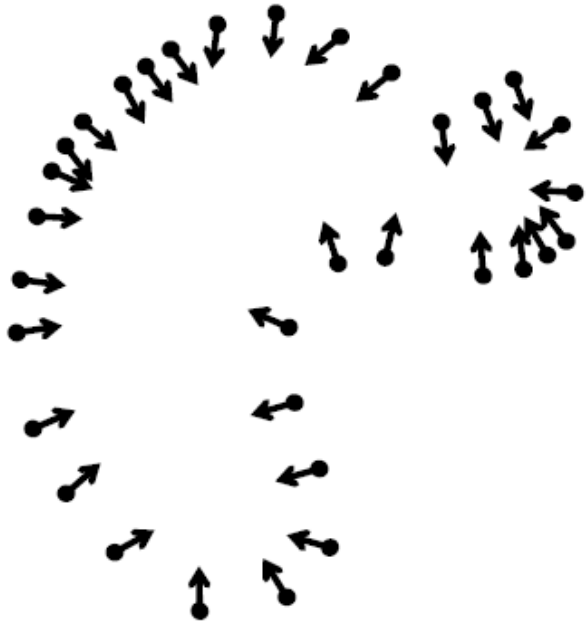
Move to the next cube



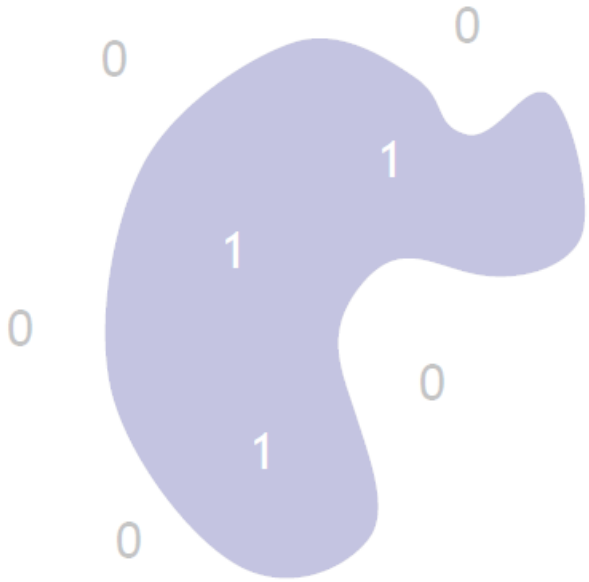
Poisson method

- Global fitting of an **indicator function** using Partial Differential Equation (PDE),

Poisson method



Oriented points

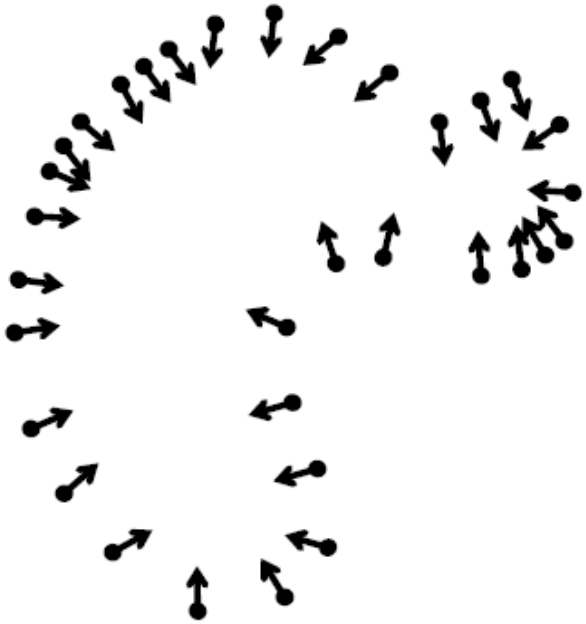


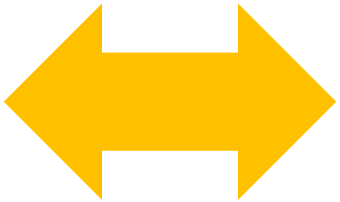
Indicator function

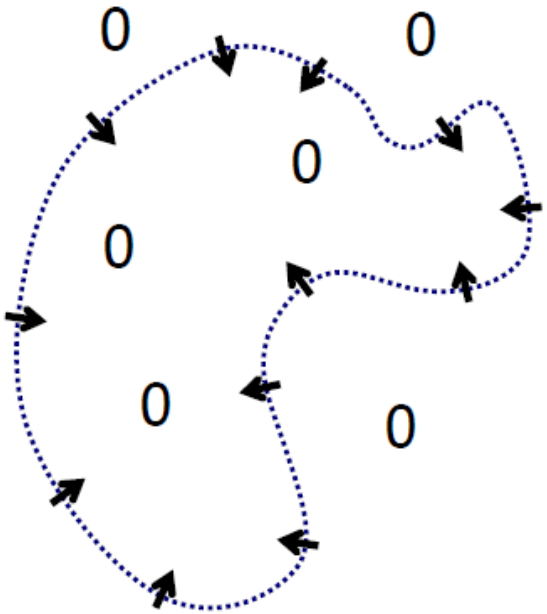
$\chi_{\mathcal{M}}$

We don't know the indicator function ☹

Poisson method







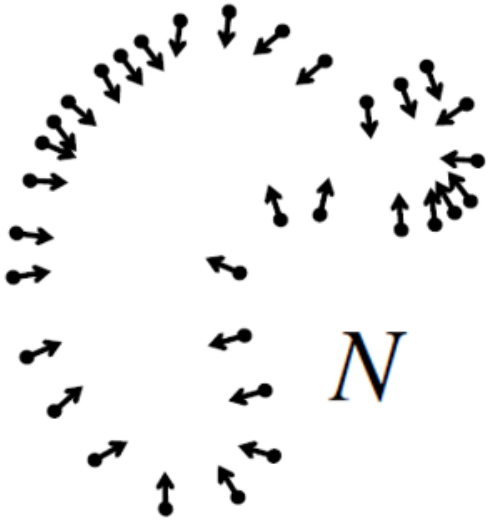
Oriented points

Indicator gradient

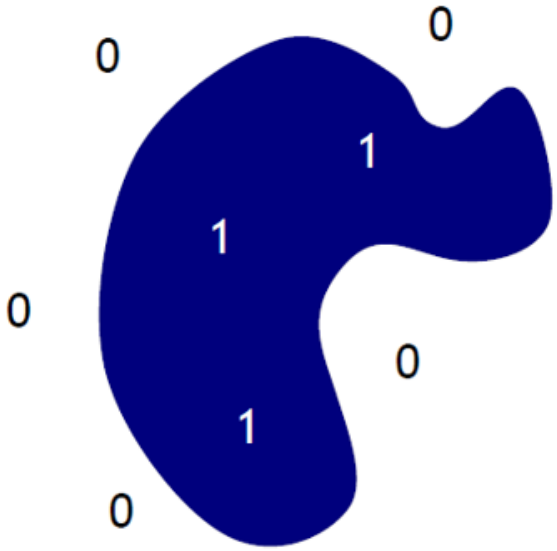
$\nabla \chi_{\mathcal{M}}$

But we can estimate its gradient! ☺

Poisson method

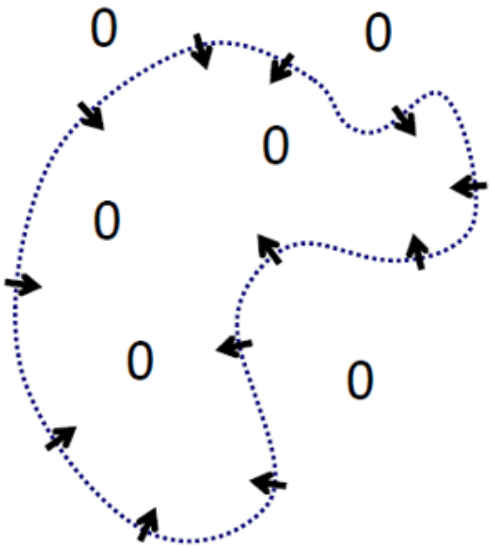


Oriented points



Indicator function

$\chi_{\mathcal{M}}$



Indicator gradient

$\nabla\chi_{\mathcal{M}}$

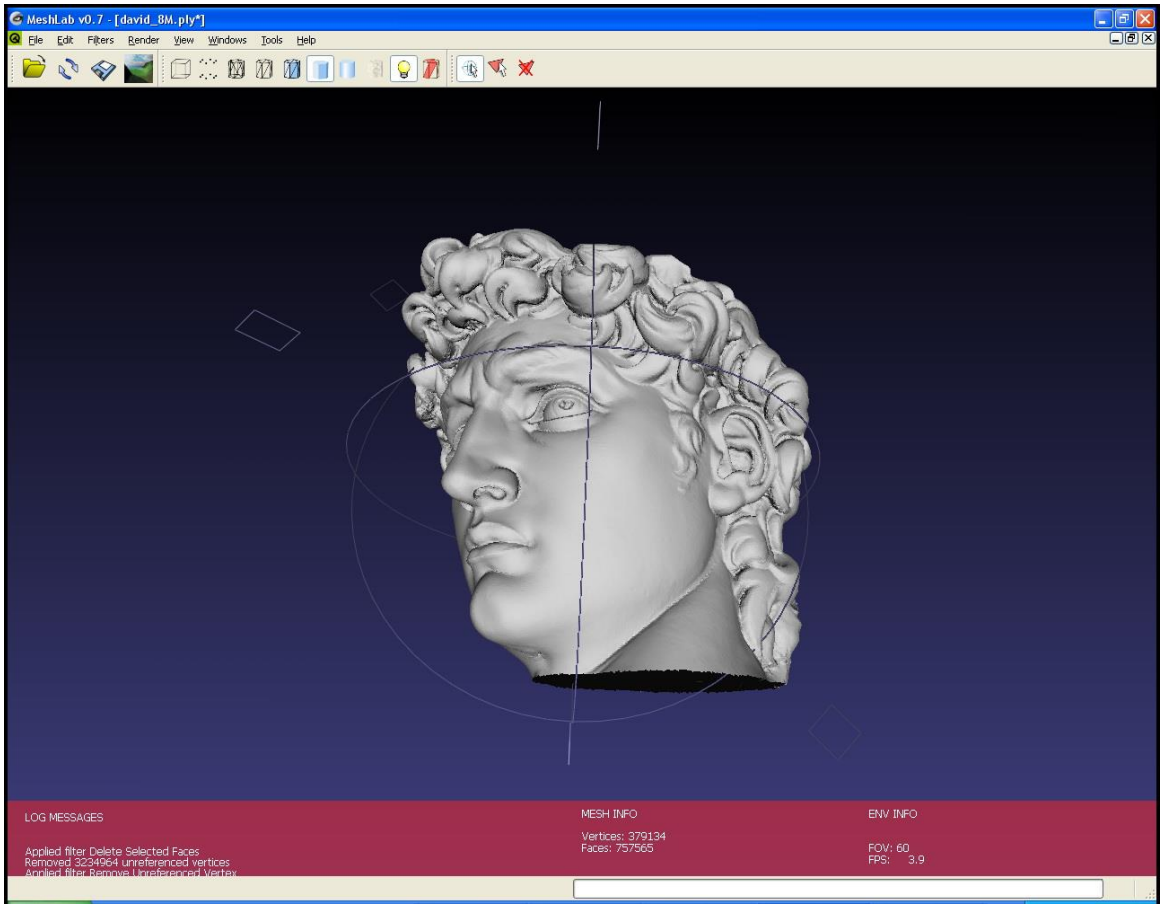
Reconstruct χ by solving the Poisson equation

$$\Delta \mathrm{X}_M = \nabla \cdot (\nabla \mathrm{X}_M)$$

$$\nabla \cdot N$$

Poisson method

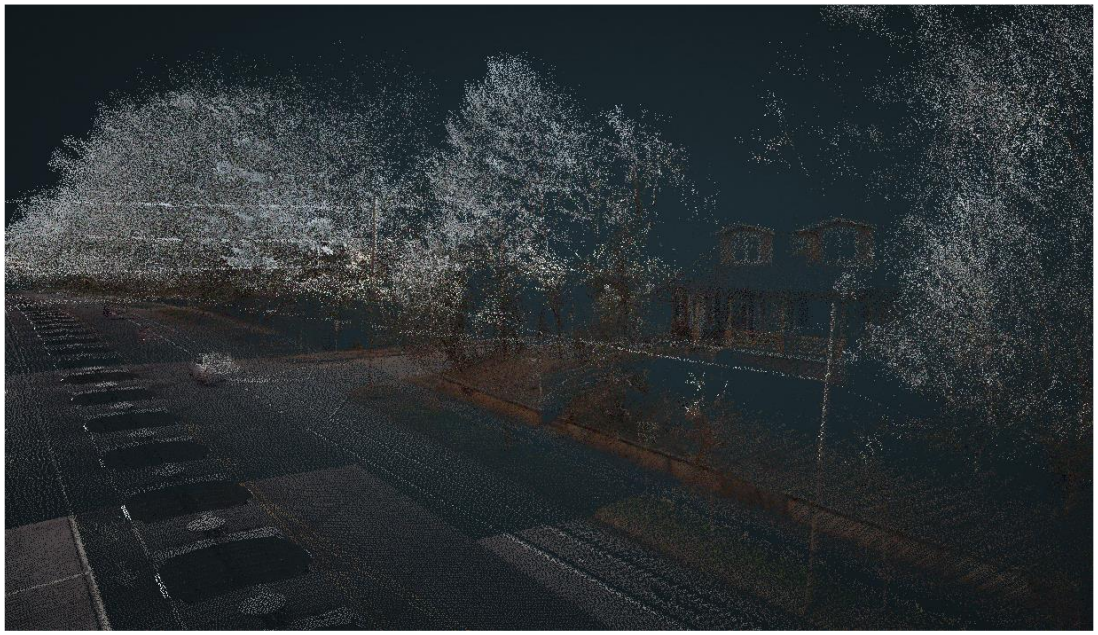
Meshlab



https://www.meshlab.net/

Homework1

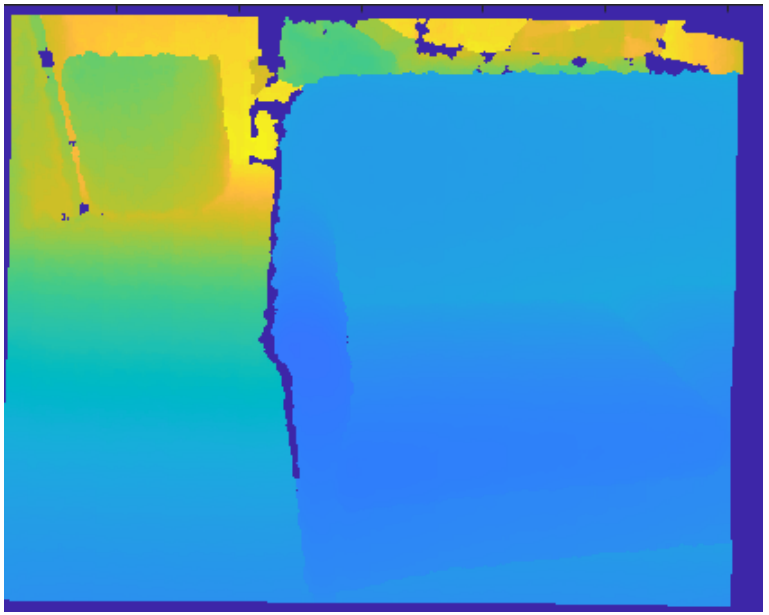
1. Find public dataset of range images from different acquisition systems
   – E.g. https://www.lidarusa.com/sample-data.html#



Homework2

2. Create a 3D cloud of points from a range image

Color image

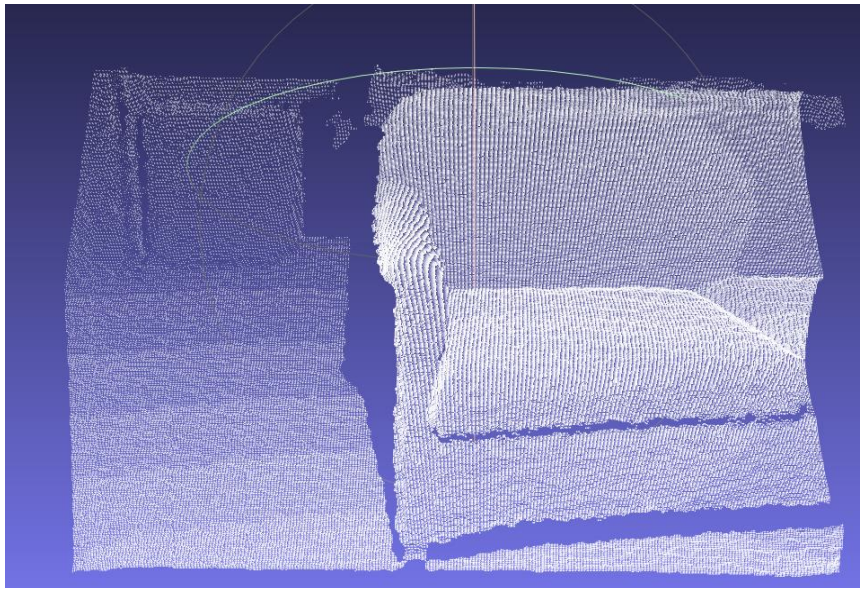


Range image

Homework2

2. Create a 3D cloud of points from a range image

Color image

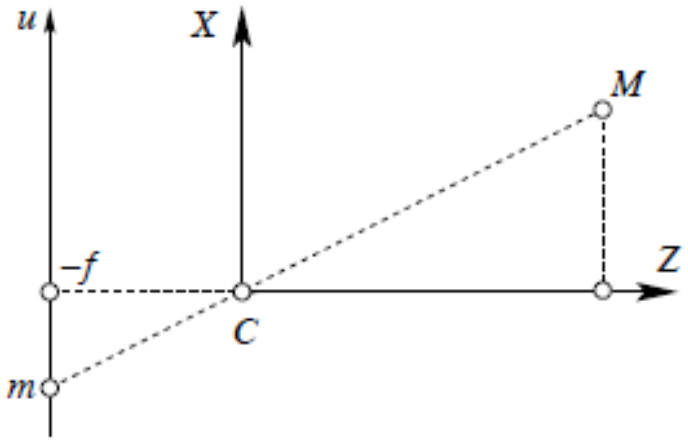


3D point cloud

Homework2

2. Create a 3 coud of points from a range image

Suggestions: z is encoded in the range image, find the x and y coordinates from the projection equations



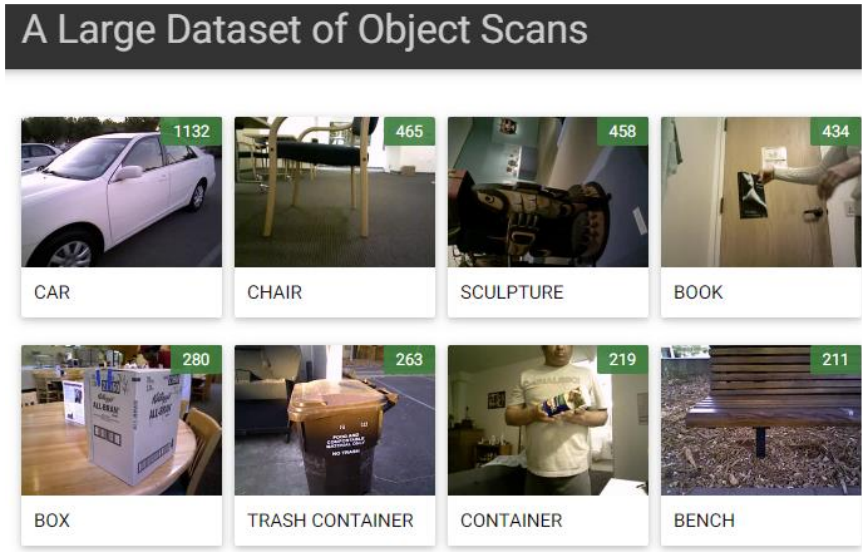
$$\begin{cases} u = k_u \dfrac{-f}{z} x + u_0 \\ v = k_v \dfrac{-f}{z} y + v_0 \end{cases},$$

The global reference system is on the camera, therefore only the main intrinsic parameters are required: (u_0, v_0) and $f_u=fk_u$, $f_v=fk_v$

Homework2

2. Create a 3 coud of points from a range image

E.g.: take some samples from http://redwood-data.org/3dscan/



The RGB-D sequences were acquired with **PrimeSense Carmine** cameras
The focal length is 525 for both axes (f_u=525, f_v=525) and the principal point is (u_0 =319.5, v_0=239.5).

Homework2

Export the cloud of point with this script and visualize it with Meshlab

```
function exportMeshToPly(vertices, faces, vertex_color, name)
if(max(max(vertex_color))<= 1.0)
   vertex_color = vertex_color.*256;
end
if(size(vertex_color,2) == 1)
    vertex_color = repmat(vertex_color,1,3);
end
vertex_color = uint8(vertex_color);
fidply = fopen([name '.ply'],'w');

fprintf(fidply, 'ply\n');
fprintf(fidply, 'format ascii 1.0\n');
fprintf(fidply, 'element vertex %d\n', size(vertices,1));
fprintf(fidply, 'property float x\n');
fprintf(fidply, 'property float y\n');
fprintf(fidply, 'property float z\n');
fprintf(fidply, 'property uchar red\n');
fprintf(fidply, 'property uchar green\n');
fprintf(fidply, 'property uchar blue\n');
fprintf(fidply, 'element face %d\n', size(faces,1));
fprintf(fidply, 'property list uchar int vertex_index\n');
fprintf(fidply, 'end_header\n');

for i=1:size(vertices,1)
      fprintf(fidply, '%f %f %f %d %d %d\n',vertices(i,1), vertices(i,2), vertices(i,3), vertex_color(i,1), vertex_color(i,2), vertex_color(i,3));
end

for i=1:size(faces,1)
    fprintf(fidply, '3 %d %d %d\n',faces(i,1)-1, faces(i,2)-1, faces(i,3)-1);
end

fclose(fidply);


end
```

Homework2

- Try other public available datasets like:

`https://rgbd-dataset.cs.washington.edu/index.html`

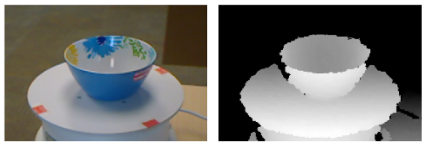
RGB-D Object Dataset

Home | People | Dataset | Results | Software | Demos

News

- April 5, 2014 - The RGB-D Scenes Dataset v.2 is now available here! It contains 14 new scenes reconstructed from RGB-D videos with furniture and tabletop objects, as well as Trimble 3D Warehouse objects use
- July 14, 2013 - Code for HMP features now available here. It achieves state-of-the-art results on the RGB-D Object Dataset!
- December 13, 2012 - Software and data for detection-based object labeling in Kinect videos now available here.
- October 3, 2012 - The dataset is now available for download directly from the website! No more sending emails necessary (questions and suggestions are, of course, still welcomed!).
- April 6, 2012 - RGB-D kernel descriptors are now available.
- March 22, 2012 - 3D reconstructions created by aligning video frames of all 8 scenes in the RGB-D Scenes Dataset are now available.
- June 20, 2011 - Pose annotations for all 300 objects in the RGB-D Object Dataset are now available.

Overview



The RGB-D Object Dataset is a large dataset of 300 common household objects. The objects are organized into 51 categories arranged using WordNet hypernym-hyponym relationships (similar to ImageNet). This datas and aligned 640x480 RGB and depth images at 30 Hz. Each object was placed on a turntable and video sequences were captured for one whole rotation. For each object, there are 3 video sequences, each recorded with t different angles with the horizon.

Unlike many existing datasets,such as Caltech 101 and ImageNet, objects in this dataset are organized into both *categories* and *instances*. In these datasets, the class *dog* contains images from many different dogs and th RGB-D Object Dataset the category *soda can* is divided into physically unique instances like *Pepsi Can* and *Mountain Dew Can*. The dataset also provides ground truth pose information for all 300 objects.

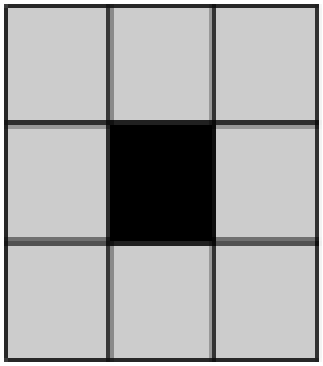
Here are some example objects that have been segmented from the background.

Use (and understand) the available code !

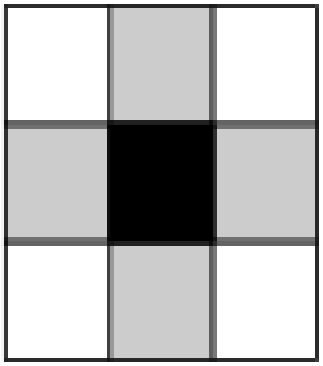
Homework 3

Mesh reconstruction from **range image**

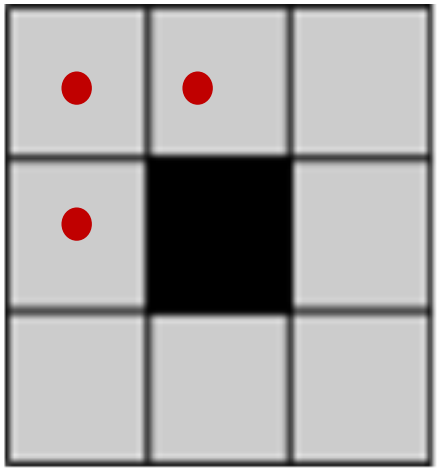
- **Idea**: points are on a regular grid where the connectivity can be inherited from the pixel neighbourhood



8-neigh



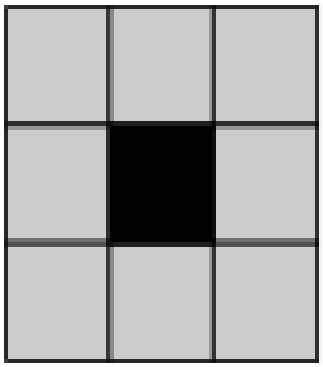
4-neigh



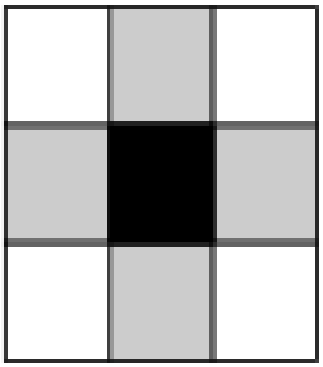
Homework 3

Mesh reconstruction from **range image**

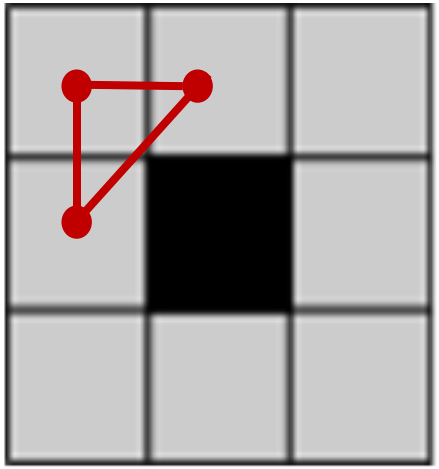
- **Idea**: points are on a regular grid where the connectivity can be inherited from the pixel neighbourhood



8-neigh



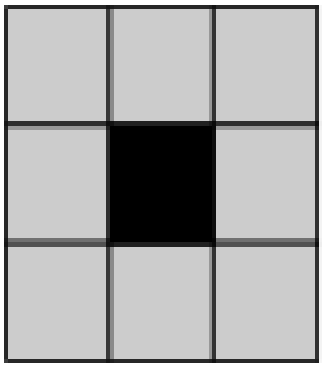
4-neigh



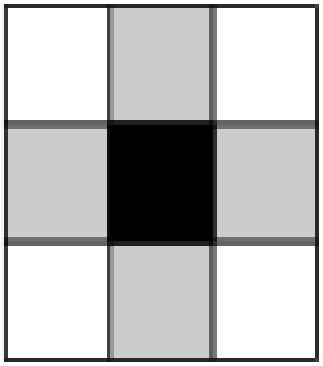
Homework 3

Mesh reconstruction from **range image**

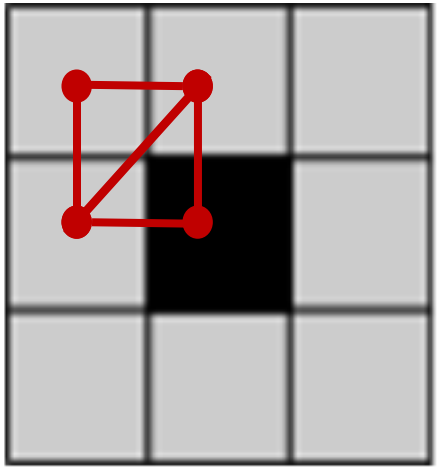
- **Idea**: points are on a regular grid where the connectivity can be inherited from the pixel neighbourhood



8-neigh



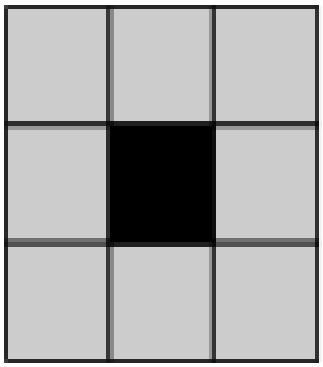
4-neigh



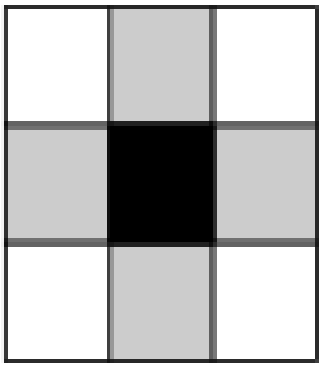
Homework 3

Mesh reconstruction from **range image**

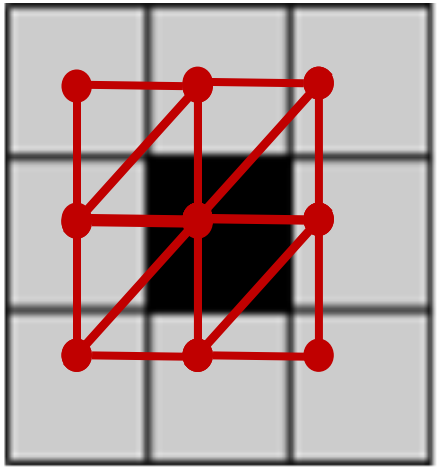
- **Idea**: points are on a regular grid where the connectivity can be inherited from the pixel neighbourhood



8-neigh



4-neigh



Homework 3

- Points are on a regular grid where the connectivity can be inherited from the pixel neighbourhood,

  BUT…

  1) Not all the pixels on the range image are the projection of a point on the 3D space!

  ➡ a binary mask can be used to define valid points,

  2) (optional) Nearby pixels should not correspond to nearby points on the 3D space!

  ➡ a robust strategy can be used to remove long edges.