Multi-modal Registration of Visual Data

Massimiliano Corsini

Visual Computing Lab, ISTI - CNR - Italy

Overview

- Introduction and Background
- Features Detection and Description (2D case)
- Features Detection and Description (3D case)
- Image-geometry registration
- Recent Advances and Applications

Overview

- Introduction and Background
  - Computer Graphics
    - Introduction
    - 3D models modeling / acquisition
    - Geometric Representations
    - Lighting / material interactions
    - Rendering
  - Computer Vision
    - Applications
    - Interactions with CG
- Features Detection and Description (2D case)
- Features Detection and Description (3D case)
- Image-geometry registration
- Recent Advances and Applications

Introduction

Computer Graphics

- Computer Graphics regards the production of 2D or 3D images starting from data.
- Such data can be the result of acquisition of real models, modeling or data coming from a scientific experiment.

Application Domains

Entertainment Industry (movies / games)

© Crytek Engine

From “Inside Out” movie
Disney-Pixar

Application Domains

Architecture

Application Domains

Engineering / Design

Application Domains

Medicine

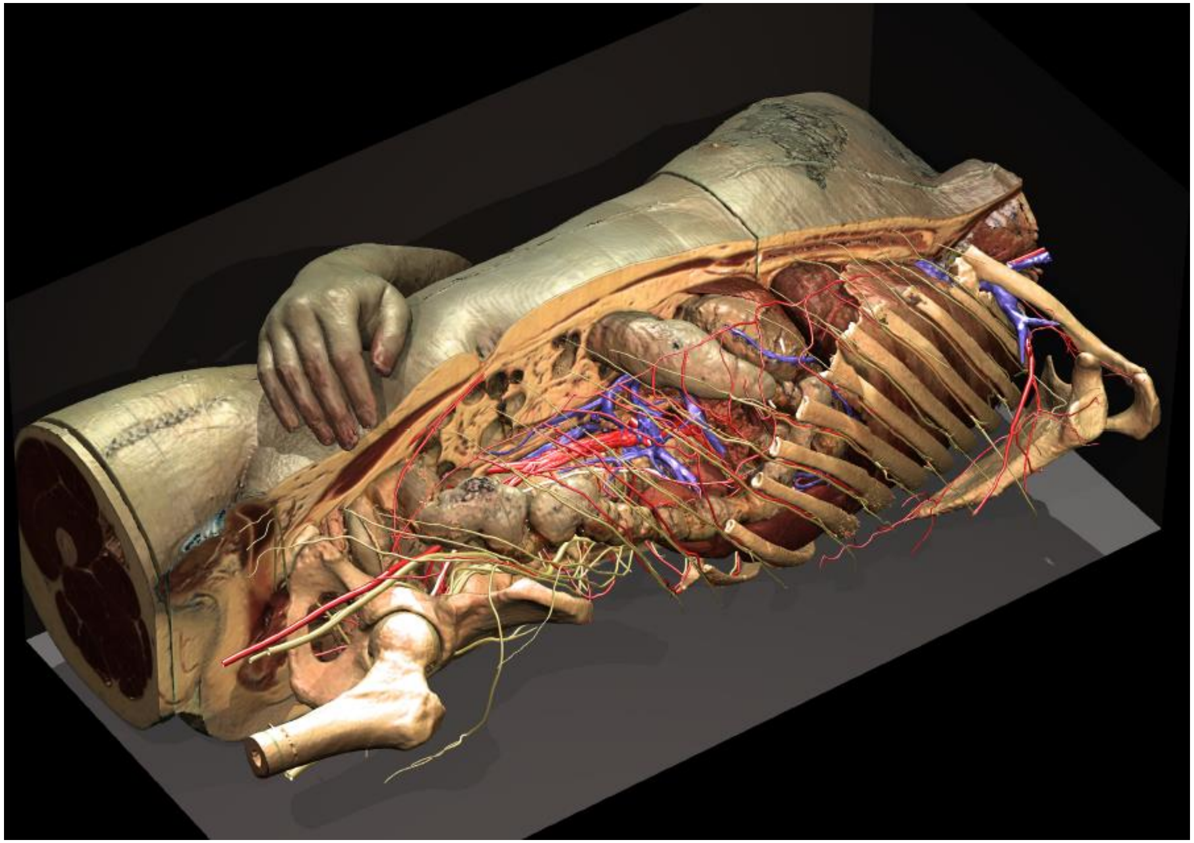
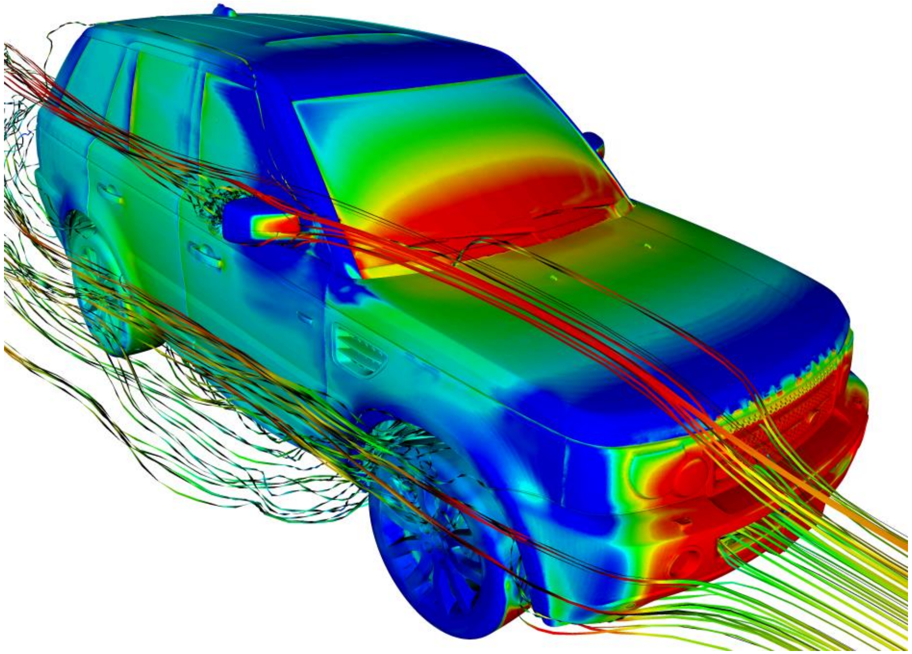


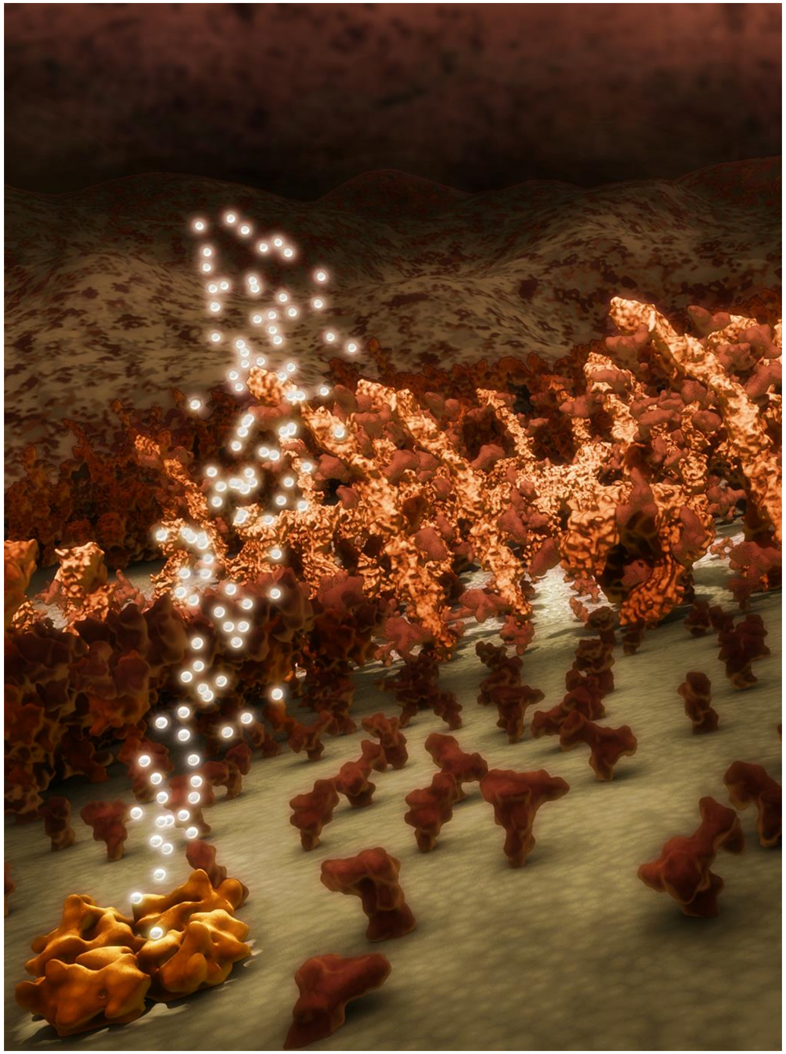
Image from Voxel-Man (http://www.voxel-man.com).

Application Domains

Natural Science



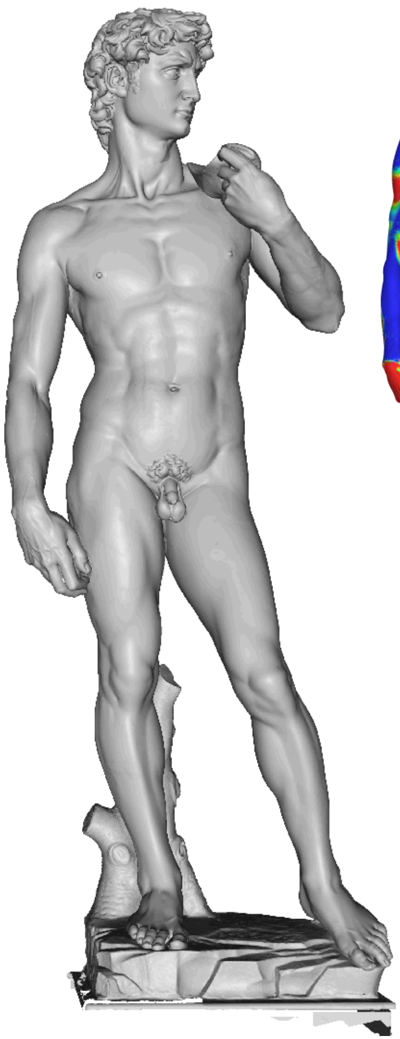
Jaguar Land Rover External Aerodynamic Simulation by Exa's PowerFLOW Software

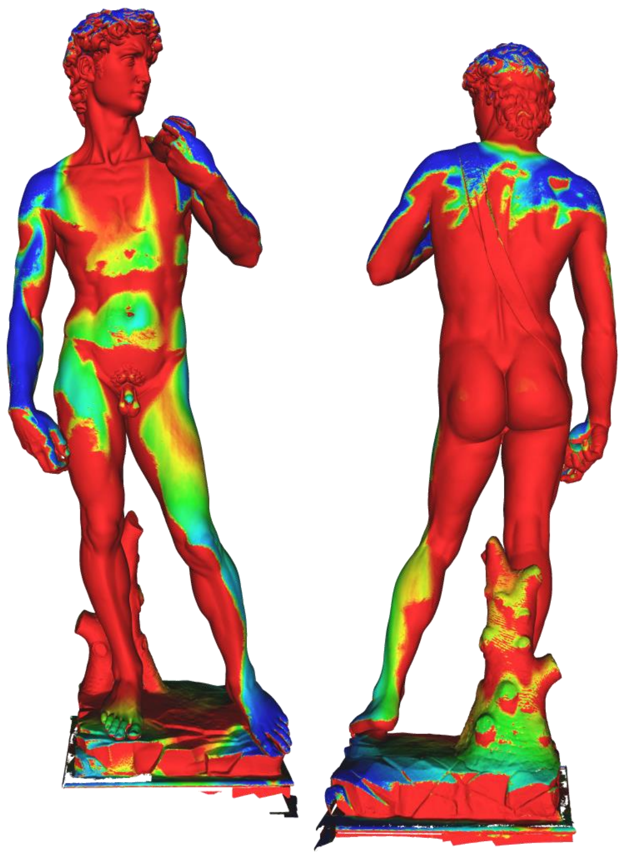


The Raft Dance – by Monica Zoppe' Biophysical Society Image Contest 2012 (Winner)

Application Domains

Cultural Heritage





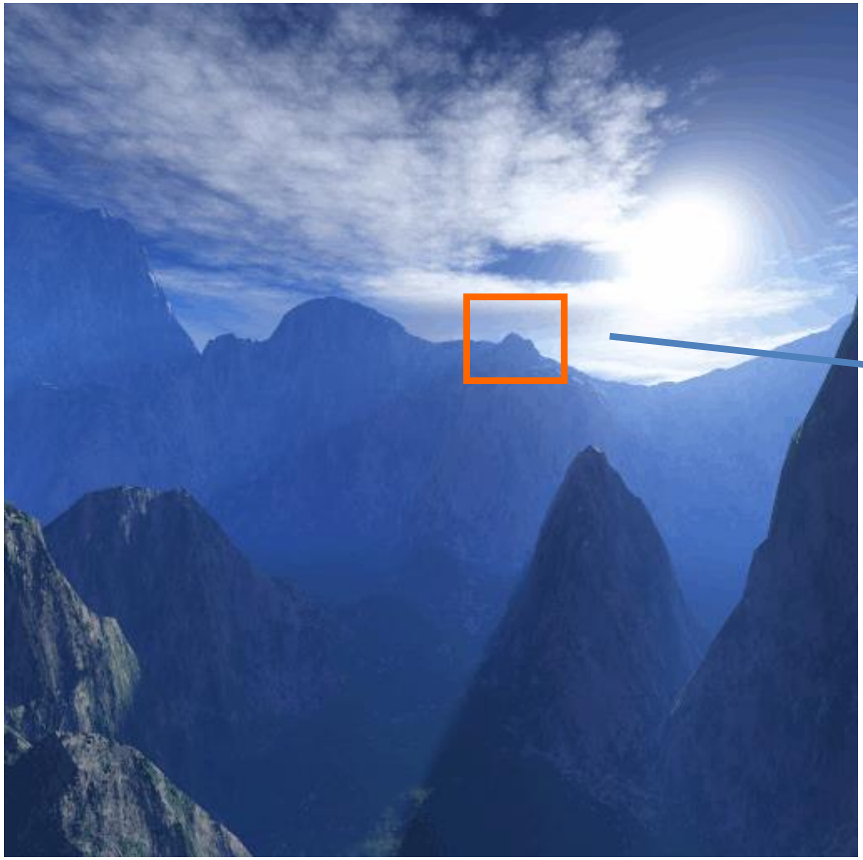
Rendering

- *Rendering* is the process to convert the input data in an image.
- Rendering paradigms:
  - Ray Tracing
  - Path Tracing
  - Rasterization-based Pipeline

Image Representation

Image = a matrix of pixels

Pixel = picture elements
(basic element of the image)



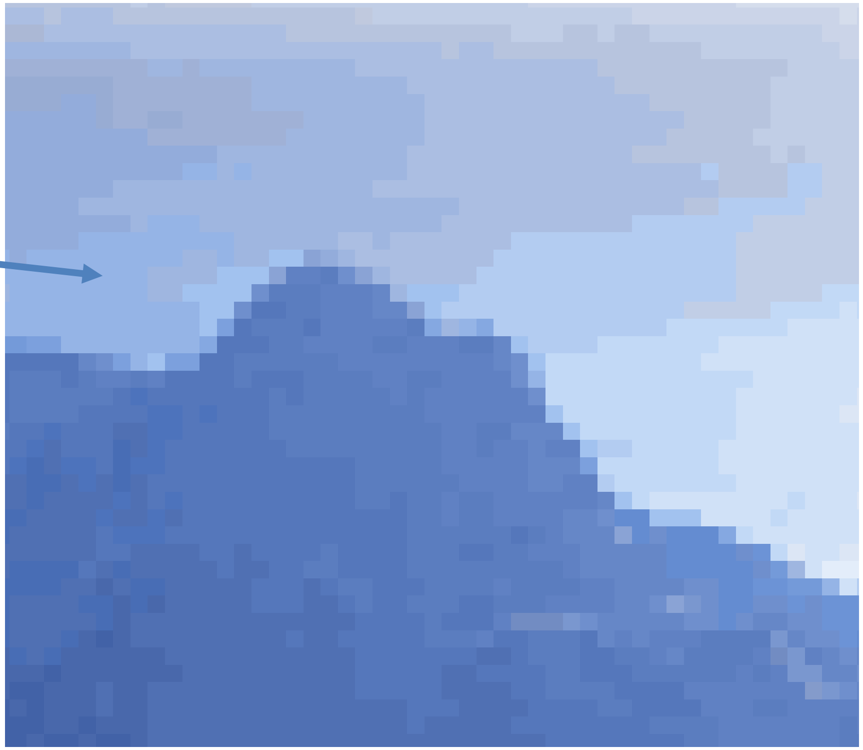


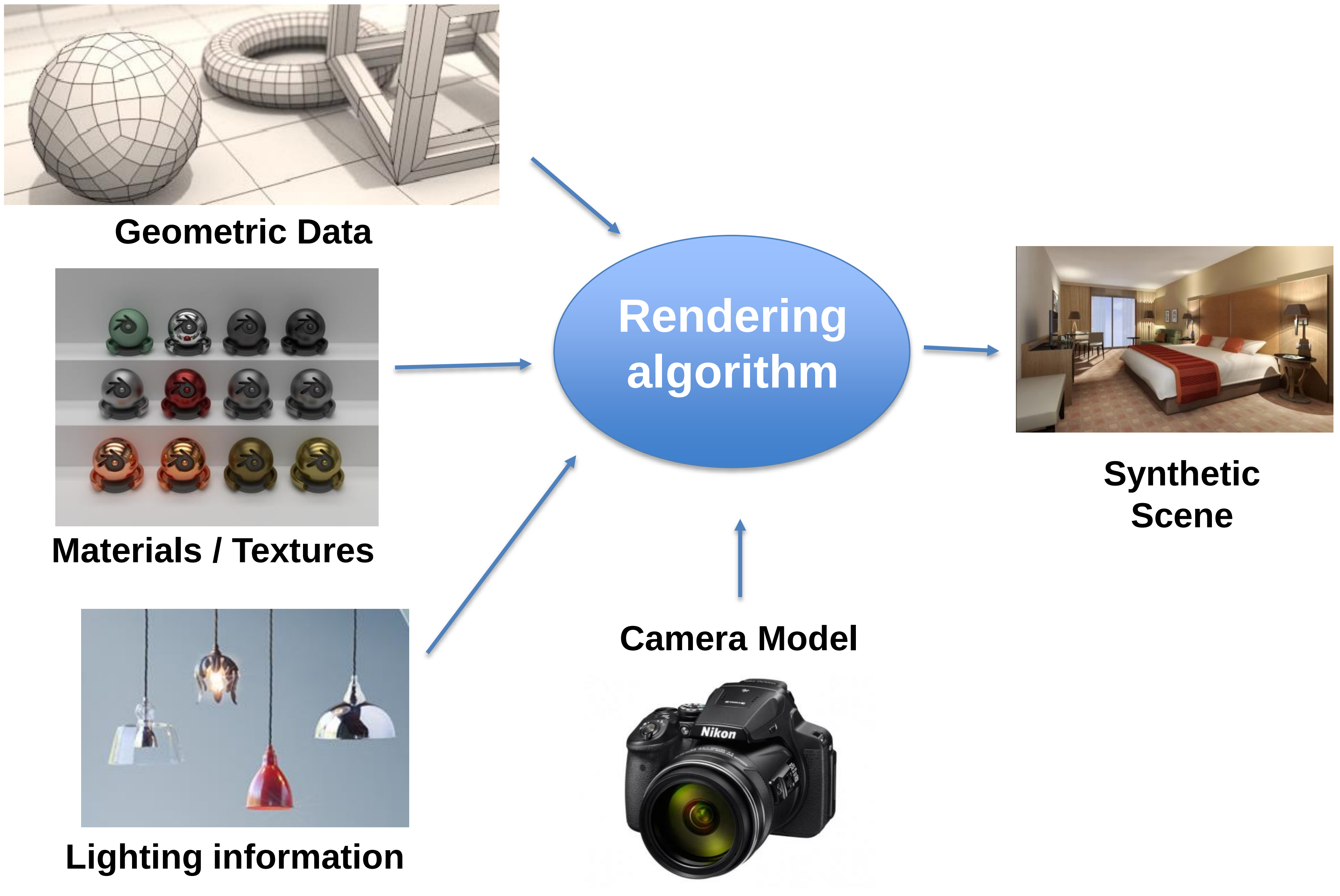
Image Representation

- A pixel can have different components.
- The number of components is the number of *channels* of the image.
- Examples:
  - Greyscale Image
    - One component → the value indicates the brightness of the pixel (e.g. 0.0=black, 1.0=white)
  - Color image (RGB)
    - Three components: Red, Green, Blue → these components can be added to form all the colors

Image Representation

190	187	189	192	192	189	183	172	164	154	139	124	122	122	129	122	111	110	118	112
191	189	191	189	191	190	183	169	158	144	138	136	129	126	136	120	90	81	88	73
194	194	193	190	192	187	180	162	153	138	137	132	103	110	97	71	59	57	55	51
198	198	198	193	193	183	170	151	138	115	100	97	68	70	57	49	50	55	53	56
203	199	197	190	185	173	159	132	118	87	58	62	53	51	54	51	50	52	51	51
203	202	194	188	178	158	128	91	78	59	56	60	49	55	58	55	49	57	62	58
207	203	199	186	156	118	75	61	56	53	59	53	51	52	62	51	52	71	86	91
208	202	189	159	101	56	54	58	51	47	50	55	54	50	57	63	51	86	127	124
207	197	167	112	65	49	50	51	45	46	42	49	59	58	81	113	92	75	157	150
204	181	122	81	54	50	49	44	43	49	44	44	55	56	91	148	128	69	161	164
193	143	92	67	50	50	53	60	52	48	43	45	61	57	77	143	137	76	150	176
158	111	86	69	56	50	65	67	63	54	45	40	60	60	68	99	106	70	143	179
129	101	78	79	78	51	60	84	80	63	42	46	72	85	71	83	76	69	149	178
124	90	85	103	97	61	61	94	100	89	75	67	87	105	85	75	58	87	162	176
110	93	105	120	120	93	58	85	97	100	93	86	88	89	87	70	64	123	174	173
104	105	108	128	134	118	81	65	85	103	100	96	98	82	69	71	111	161	179	173
99	111	114	129	144	148	111	90	77	75	89	88	86	75	76	115	158	180	182	177
100	97	107	125	137	150	139	114	105	88	79	79	86	100	123	156	183	181	177	172

Rendering a scene



Rendering a scene

Frog **by Manuel Peter**

3D Objects Creation

- Modeling
- 3D Scanning
- Image-based 3D reconstruction

3D Scanning Technologies

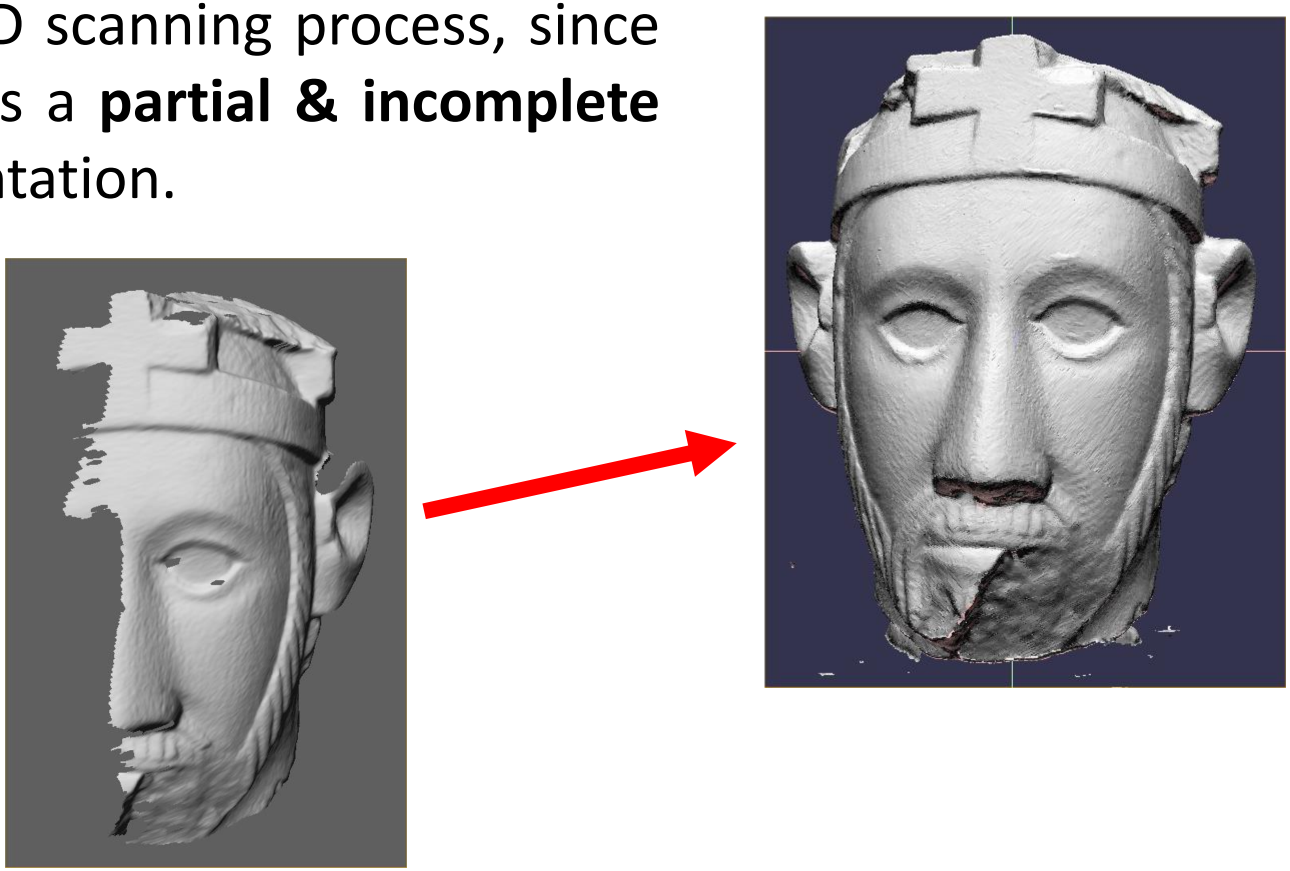
Many different technologies, just two examples:

- **Laser or structured light, Triangulation**
  - Small/medium scale artifacts (statues)
  - Small/medium workspace 20x20 -> 100x100 cm, distance from artifact ~1 m
  - High accuracy (>0.05 mm)
  - High sampling density (0.2 mm)
  - Fast (1 shot in ~1-2 sec)

- **Laser, Time of flight**
  - Large scale (architectures)
  - Wide workspace (many meters)
  - Medium accuracy (~4-10 mm)
  - Medium sampling density (10 mm)
  - Slow (1 shot in ~20 min)

Scanning Process

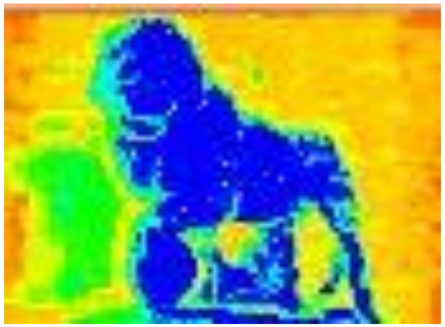
The acquisition of a **single shot** (*range map*) is only a single step in the 3D scanning process, since it returns a **partial & incomplete** representation.



Depth Maps and Range Maps

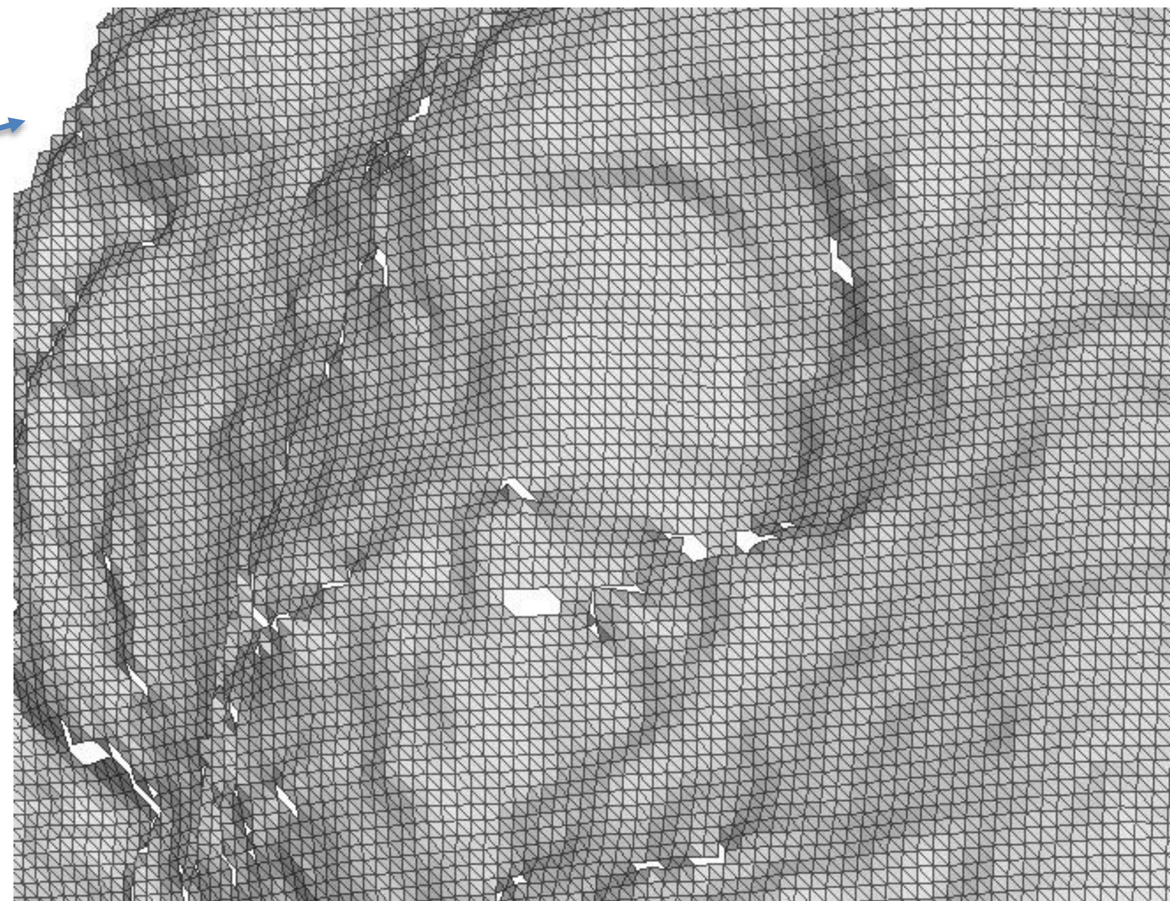
- A *depth map* is a particular type of image where the pixel value indicates a value of depth.
- A *range map* is the triangulated version of a depth map.
- The output of a 3D scanner is typically a range map.

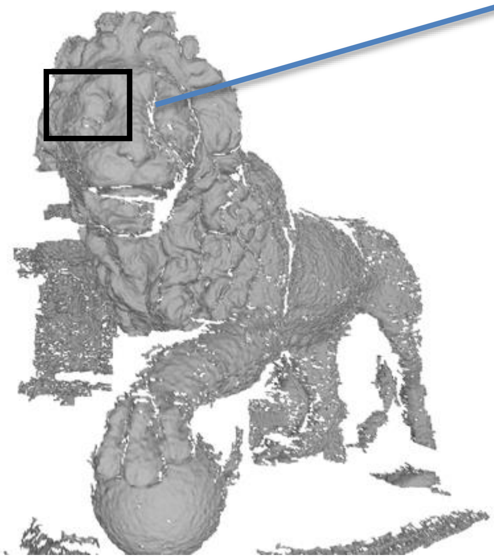
Depth Maps and Range Maps



Depth Map

Range Map (detail)

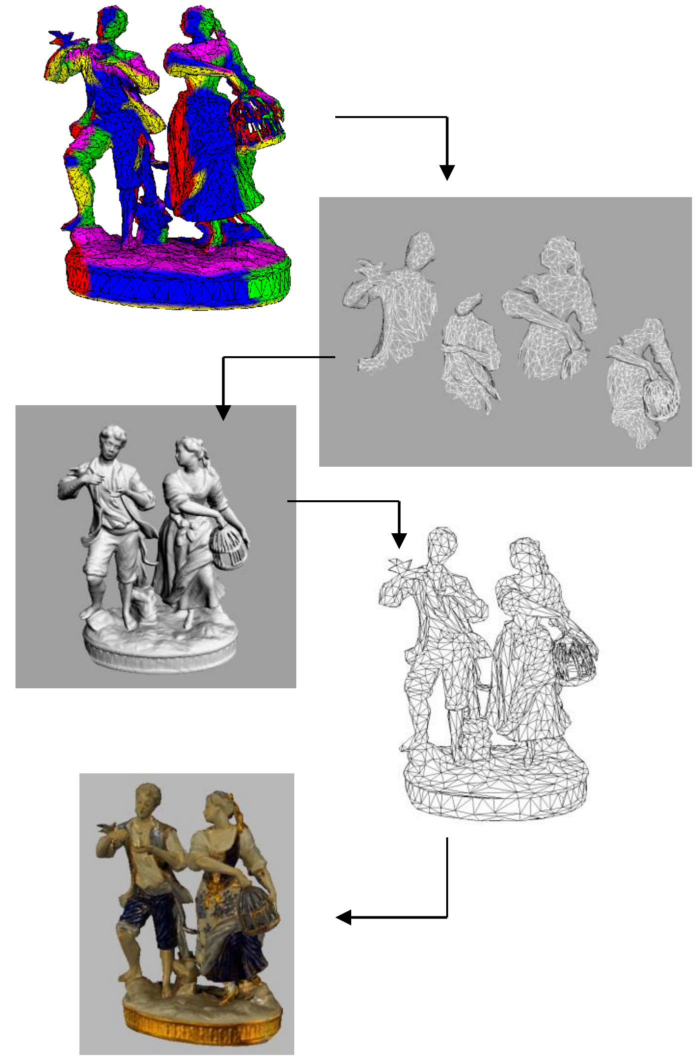




Corresponding Range Map

3D Scanning pipeline

- **3D Scanning**:
  - *[**Acquisition planning**]*
  - **Acquisition** of multiple range maps
  - Range map **filtering**
  - **Registration** of range maps
  - **Merge** of range maps
  - Mesh **Editing**
  - Geometry **simplification**
  - **Capturing** of **visual appearance** (**color/reflectance properties**)



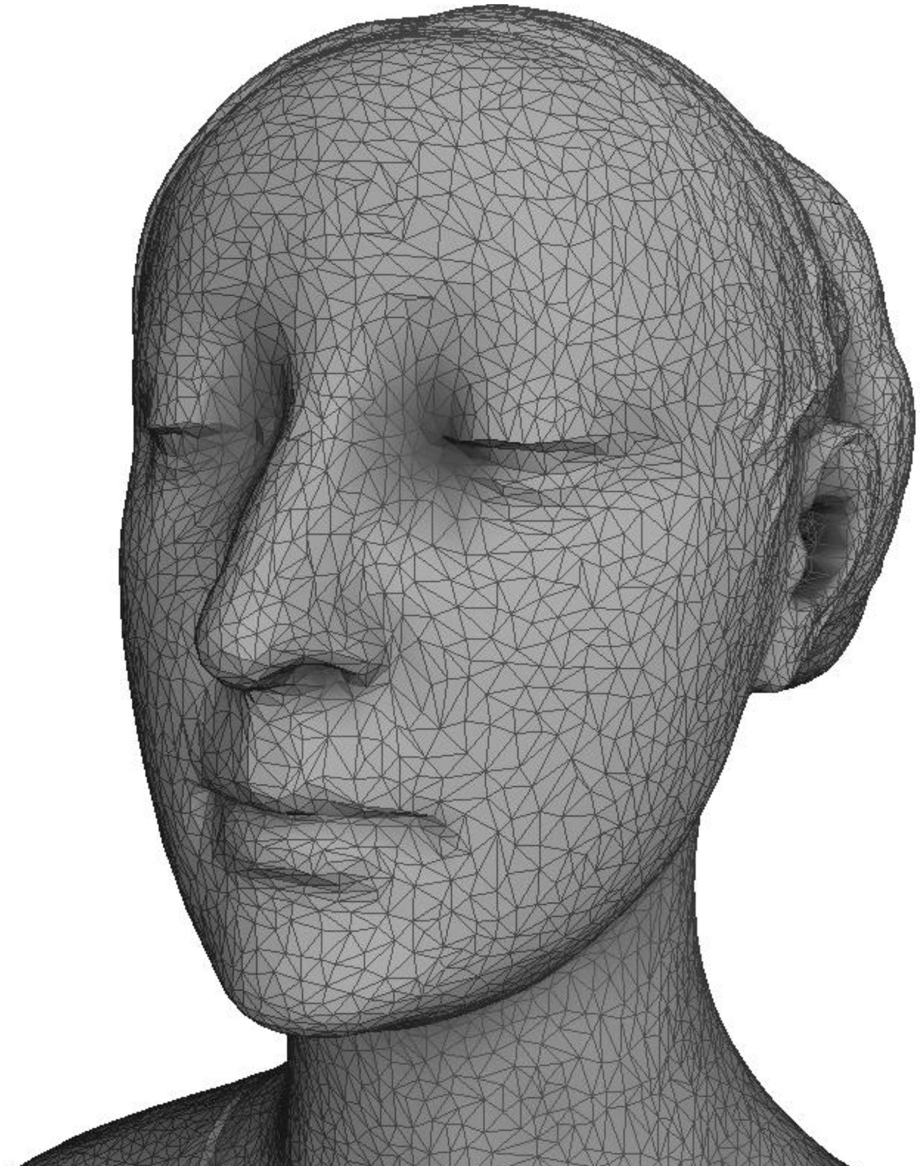
Scanning Process

Many range maps registered together

Filtering
Fine registration
Fusion

Final 3D model (geometry only)

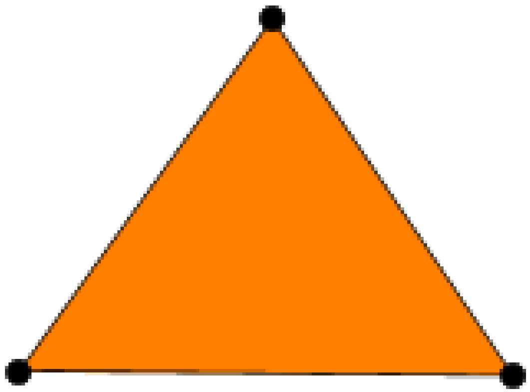
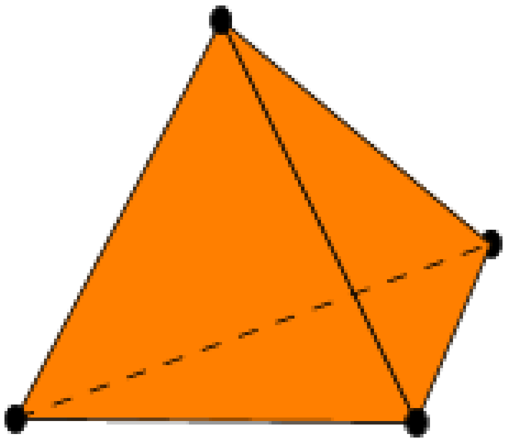
Polygonal Mesh



- A *polygonal mesh* is the partition of a continuos surface in polygons (e.g. triangles, quadrilaterals).
- A mesh *M* can be defined as a tuple *(V,K)* where *V* is the set of the vertices (i.e. points in R^3) and *K* is the set of the simplicial complexes, i.e. the set which contains the topology information.

Simplicial Complexes

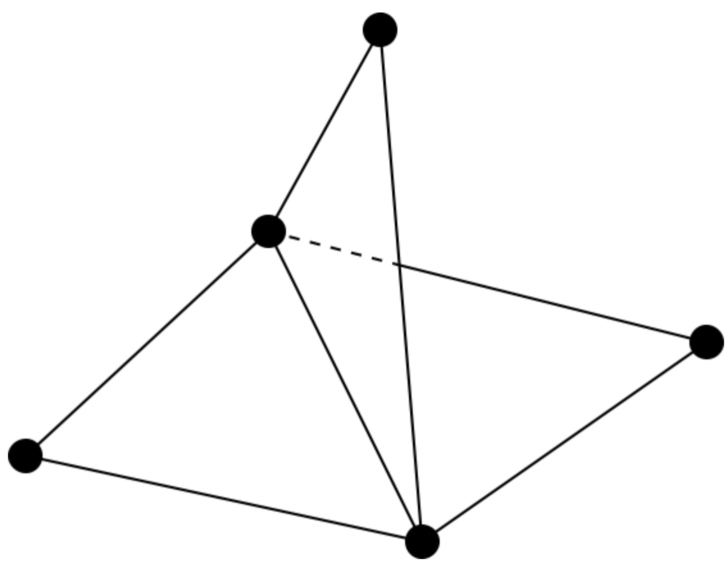
- A *simplex* of order *k* is the *convex envelope* of the k+1 points that belong to it:
  - Simplex of order 1 → segment
  - Simplex of order 2 → triangle
  - Simplex of order 3 → tetrahedron

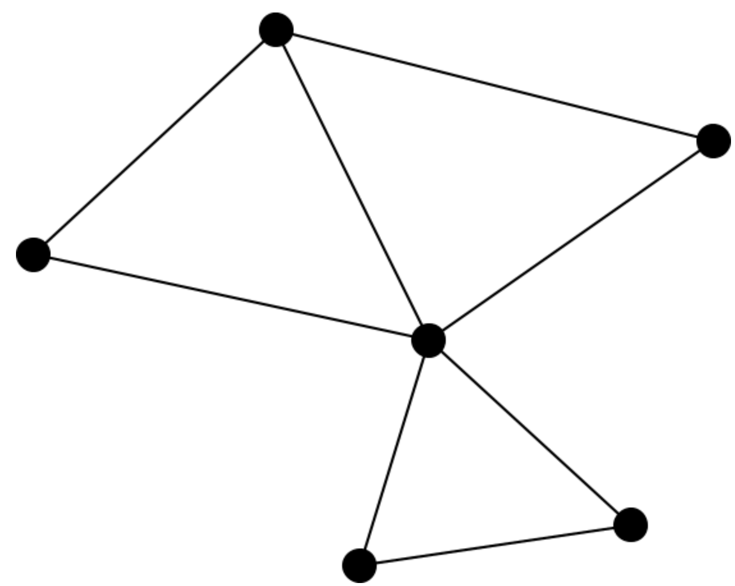




Two-manifold meshes

- A mesh is said *two-manifold* if it is homeomorphic to a disk.
- Many algorithms assume that the mesh is two-manifold.
- Properties:
  - Every edge has two and only two incident vertices
  - Every edge has one (border edge) or two incident faces
  - No disconnected elements (e.g. isolated vertex)

Non-manifold examples



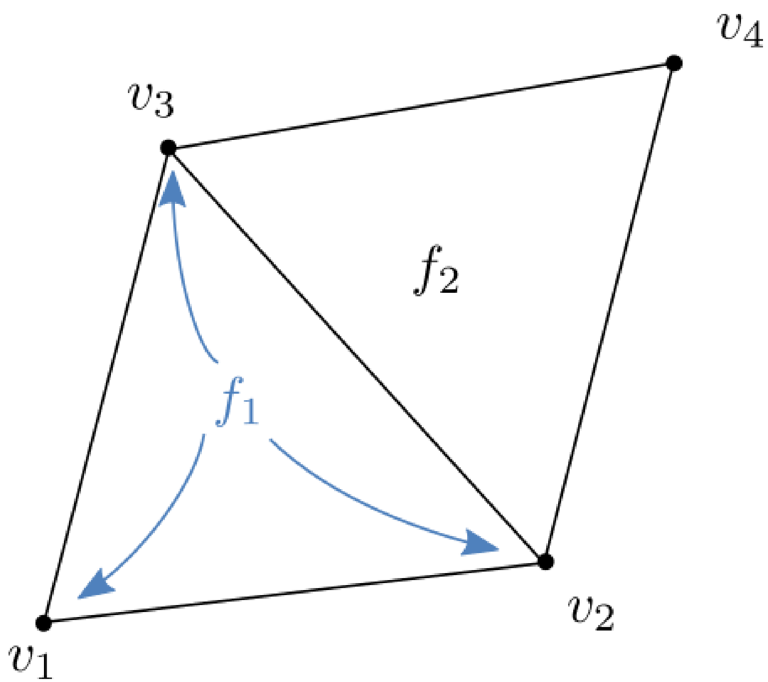


Mesh Query

- Often we need to query a mesh in some way.
- Examples:
  - Which are the faces incident on a given vertex *v* ?
  - Which are the faces incident on a given edge *e* ?
  - Which vertices are connected to a given vertex *v* ? *(1-ring)*?

Indexed Data Structure

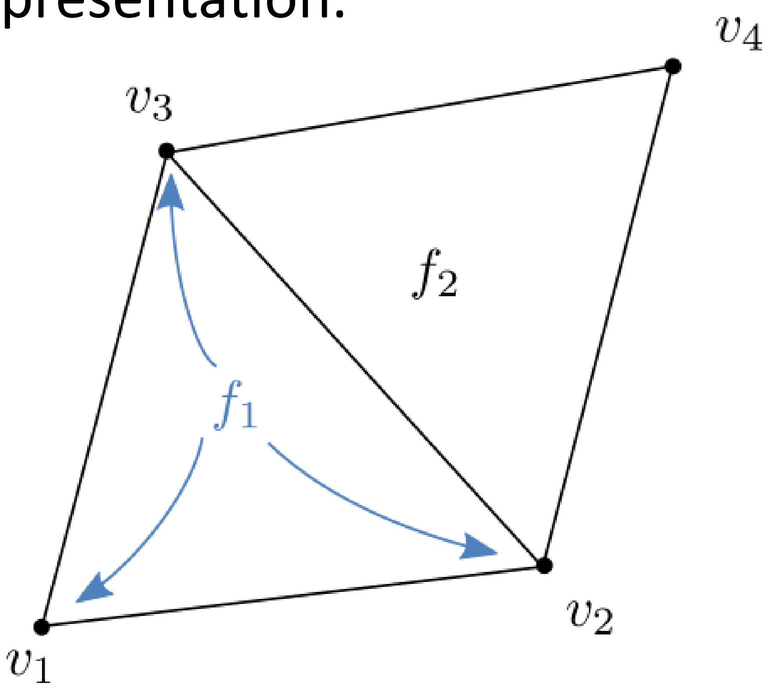
- One of the most intuitive representation.
- Each face is represented by its vertices. For example, the triangle T is $T = \{(v_{1x}, v_{1y}, v_{1z}), (v_{2x}, v_{2y}, v_{2z}), (v_{3x}, v_{3y}, v_{3z})\}$.
- Non-efficient (vertices are duplicated).
- Many queries are difficult to achieve.




```
struct
{
  float v1[3];
  float v2[3];
  float v3[3];
} face;
```

Vertex List

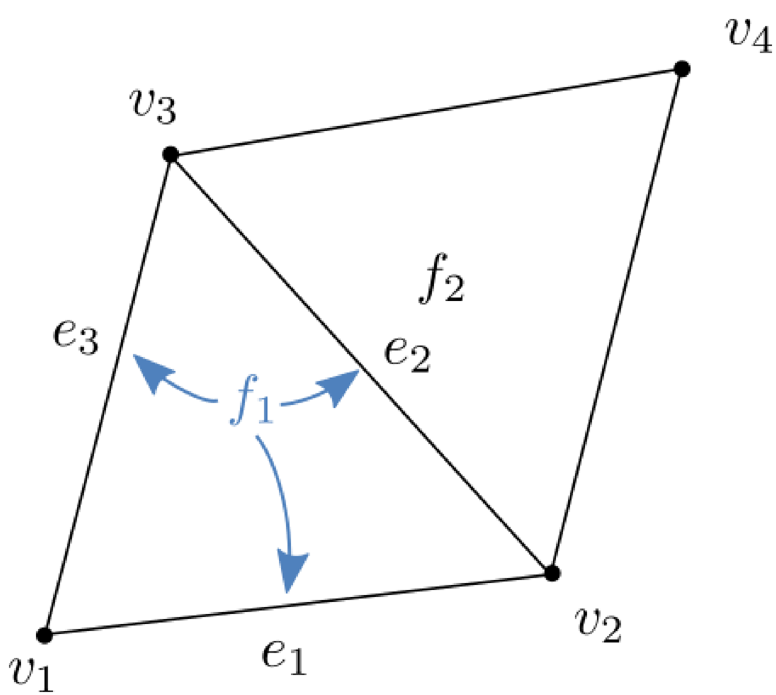
- We can increase the efficiency by using a vertex list (of non-duplicated vertices).
- The face *T* refers to its vertices.
- Edges are still duplicated.
- Many queries are difficult to implement also using this representation.




```
struct {
  float x,y,z;
} vertex;

struct {
  vertex *v1, *v2, *v3;
} face;
```

Edge List

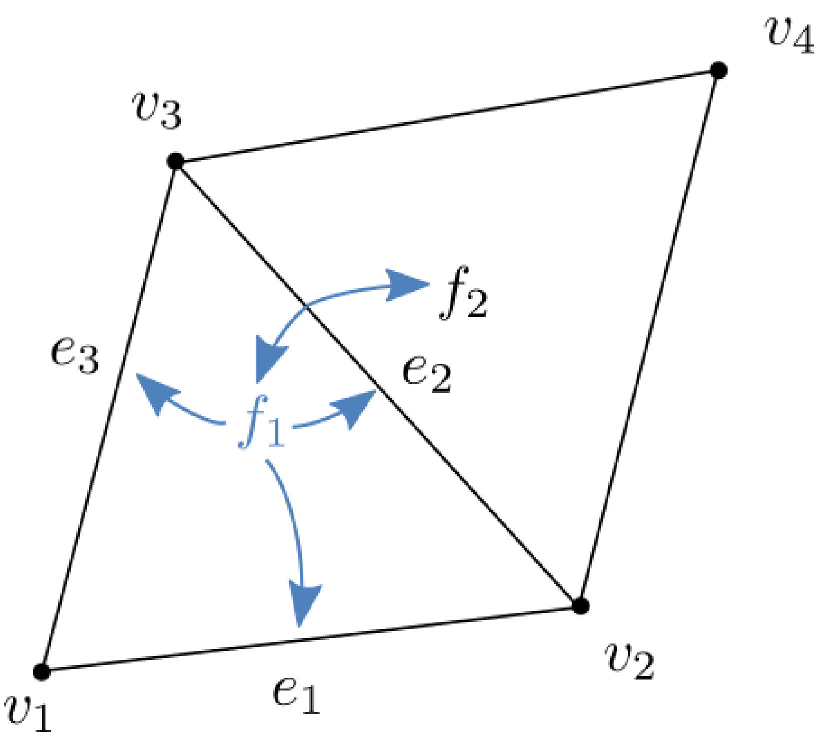
- We add a list of edges (without duplication).
- A face refers to the edges that belong to it.
- Some queries become more simpler to implement.




```
struct {
  float x,y,z;
} vertex;

struct {
  vertice *v1, *v2;
} edge;

struct {
  edge *e1, *e2, *e3;
} face;
```

Extended Edge List

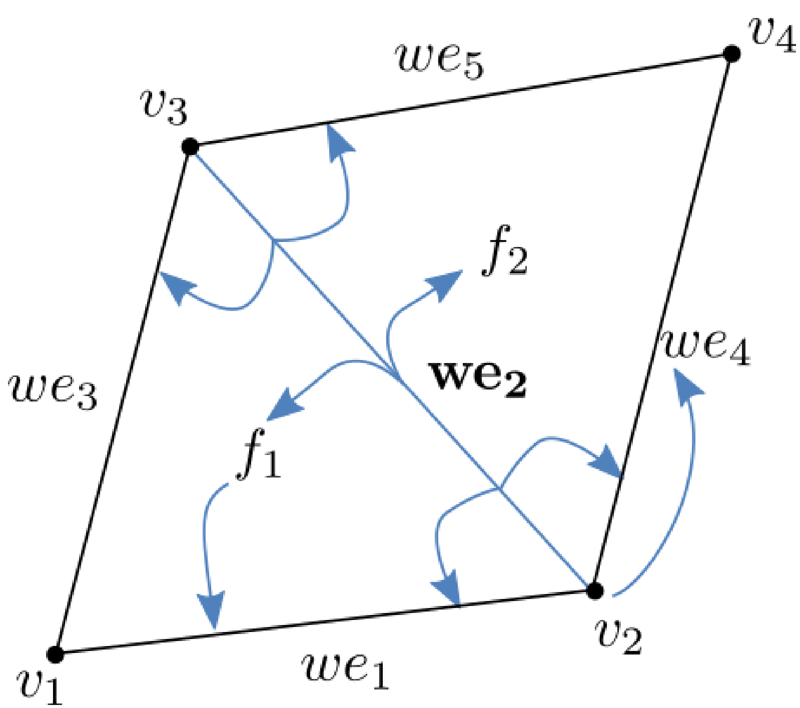
- We insert explicitly the edge-faces adjacency information.
- All the queries that use this type of adjacency benefit from this representation.




```
struct {
  float x,y,z;
} vertex;

struct {
  vertex *v1, *v2;
  face *f1, *f2;
} edge;

struct {
  vertex *e1, *e2, *e3;
} face;
```

Winged-Edge

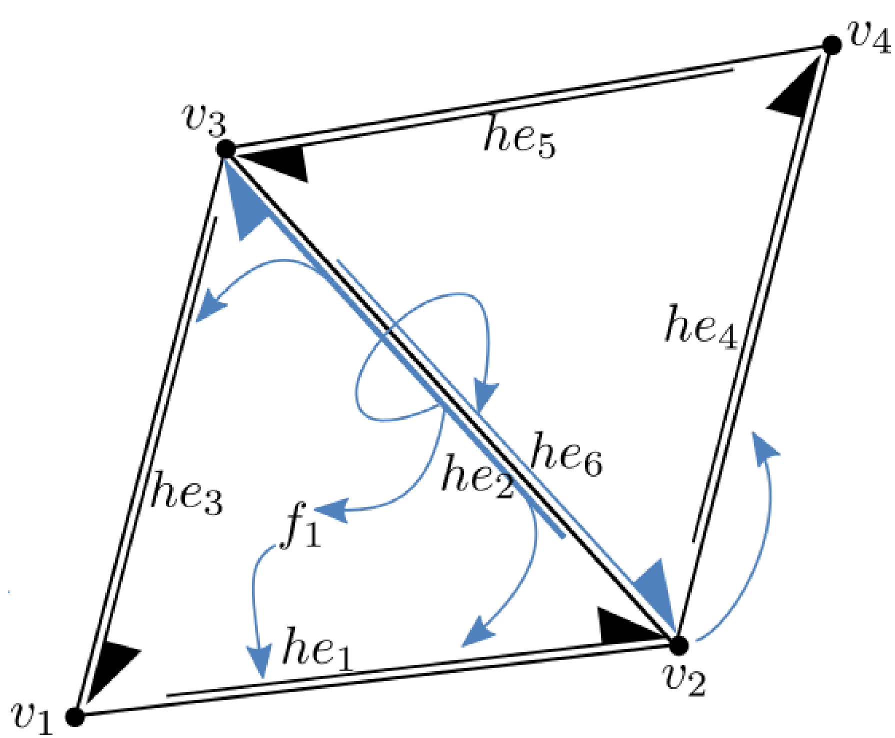
- Each winged-edge contains: its vertices, the references to its incident faces plus the references to all the incident winged-edges.
- Each vertex contains a reference to one of its incident winged-edge.
- A face is defined by a reference to one of its edge.




```
Typedef struct {
  float x,y,z;
  edge_we *edge;
} vertex_we;

struct {
  vertex_we *v1, *v2;
  edge_we *l1sin, *l1des;
  edge_we *l2sin, *l2des;
  face_we *f1, *f2;
} edge_we;

struct {
  edge_we *edge;
} face_we;
```

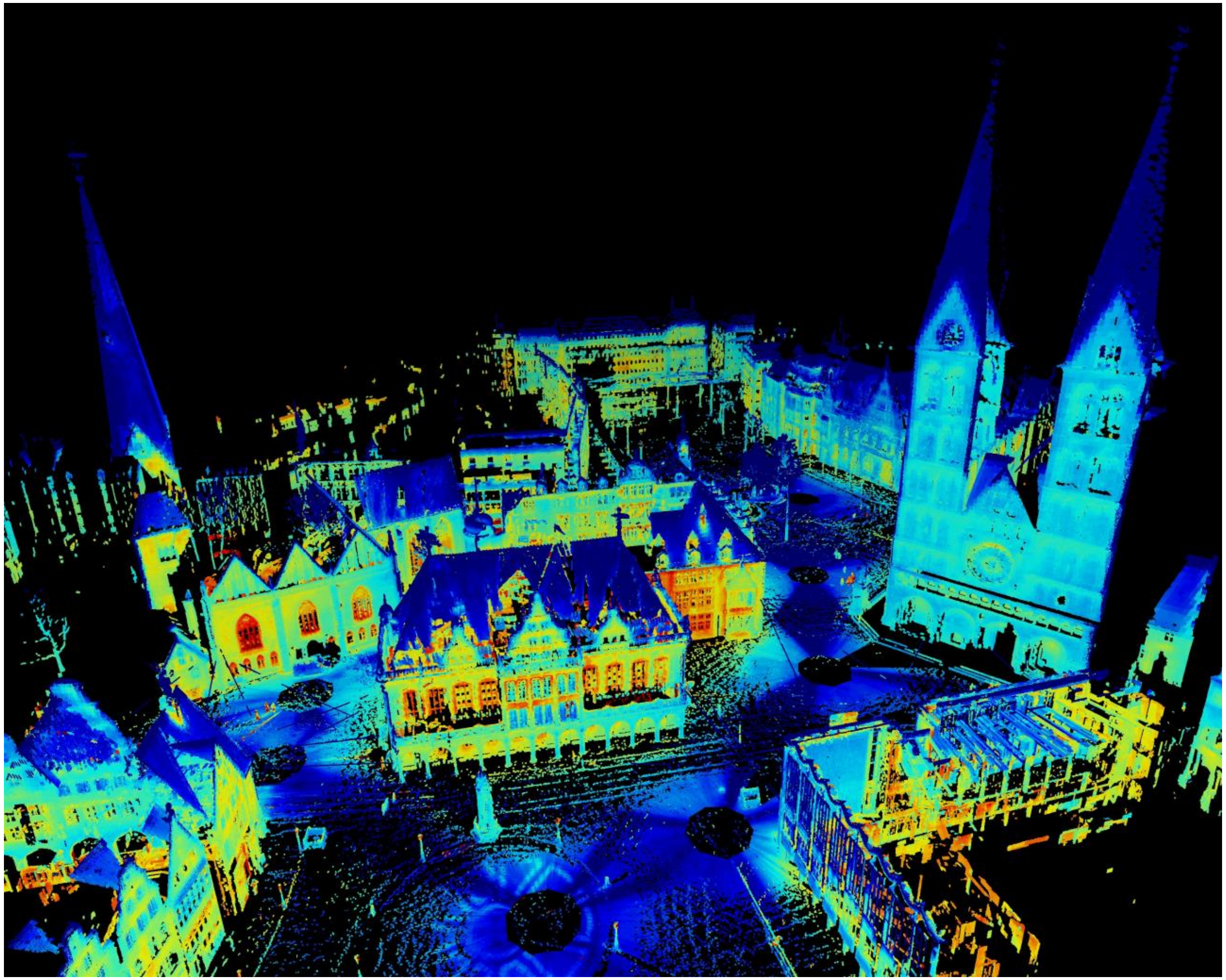
Half-Edge

- Each edge is subdivided into two half-edges.
- Each *half-edge* contains one pointer to its initial vertex and another pointer to its twin half-edge and to the half-edge.
- Each vertex contains one pointer to one of its half-edge.
- Each face contain a pointer to one of its half-edge.




```
struct {
  float x,y,z;
  edge_he *edge;
} vertex_he;

struct {
  vertex_he *origin;
  edge_he *twin;
  edge_he *prevedge;
  edge_he *nextedge;
  face_he *face;
} edge_he;

struct {
  edge_he *edge;
} face_he;
```

Point Cloud



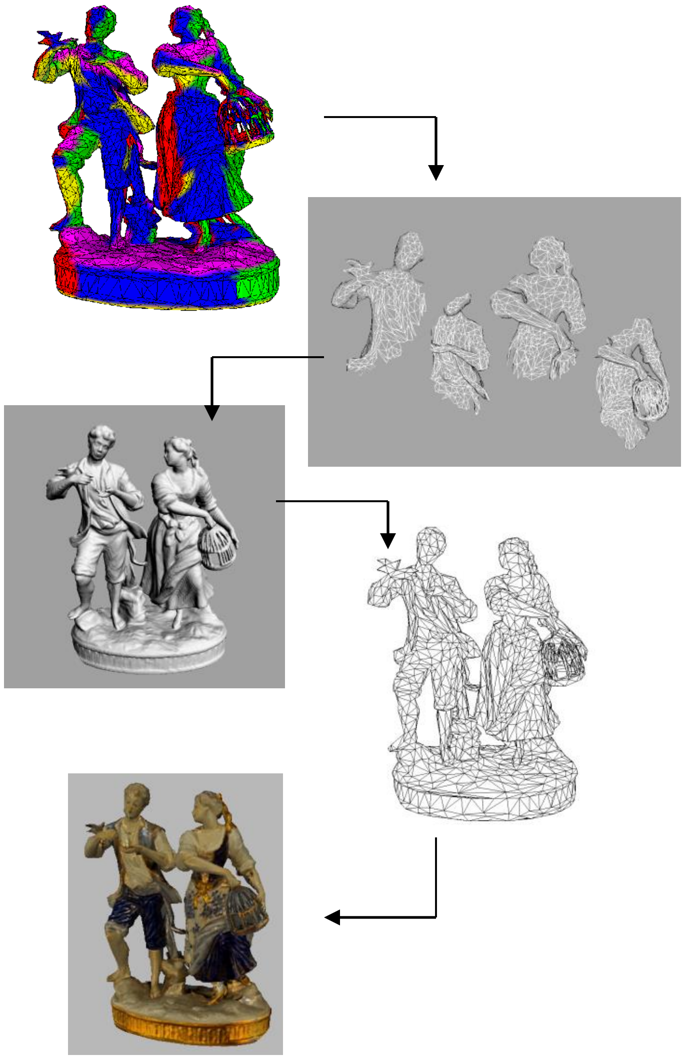
Figures by Dorit Borrmann and Andreas Nüchter from Jacobs University Bremen gGmbH, Germany.

Point Cloud

- Point cloud can be transformed into polygonal mesh through surface reconstruction methods
  - Poisson surface reconstruction
  - Moving least square
  - Ball-Pivoting

3D Scanning pipeline

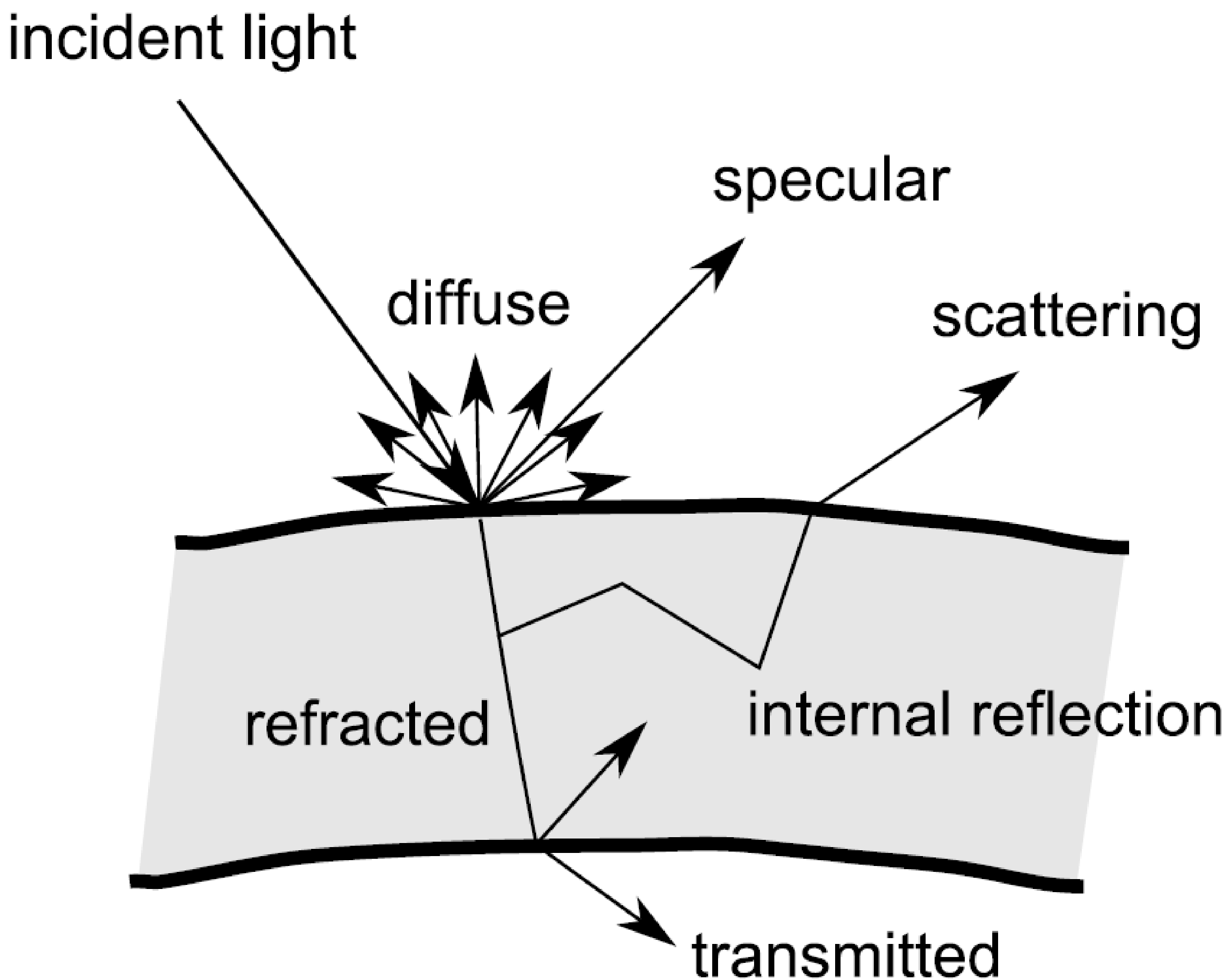
- **3D Scanning**:
  - *[**Acquisition planning**]*
  - **Acquisition** of multiple range maps
  - Range map **filtering**
  - **Registration** of range maps
  - **Merge** of range maps
  - Mesh **Editing**
  - Geometry **simplification**
  - **Capturing** of **visual appearance (color/reflectance properties)**



Light and matter

- Light is the flow of radiant energy. A light source emits such energy. This energy travels from the source to the objects and interacts with them bouncing until an equilibrium is reached.
- We see all stable because this process is very fast due to the speed of the light.

Light-matter interaction

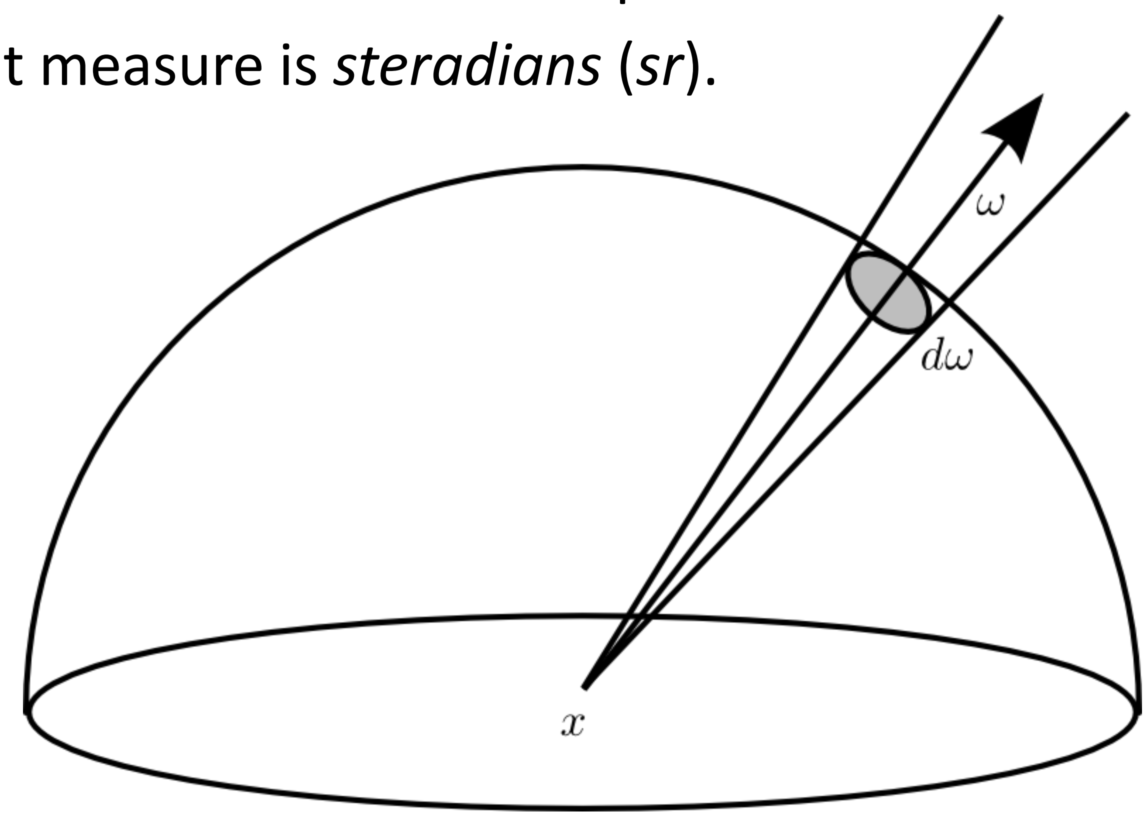


Radiometry in a nutshell

- Concepts:
  - Solid angle
  - Radiant flux
  - Ingoing and outgoing Radiance
  - Irradiance

Solid angle

- It represents the angular dimension of an infinitesimal conoid along a given direction.
- It can be seen as the conjunction representation of a direction and an infinitesimal area on a sphere.
- The unit measure is *steradians* (*sr*).



Solid angle

- It can be seen as the extension of the concept of angle at the 3D space.
- In fact, Θ is measured (in *radians*) as the ratio ***s*** / ***r*** where ***s*** is the length of the arc or ray ***r*** under Θ.
- Analougously, the solid angle Ω is measured (in *steradians*, *sr*) as the ratio ***A*** / $\boldsymbol{r^2}$ where ***A*** is the area of the spherical surface of ray ***r*** under Ω.
- Examples:
  - Square angle: $2 \pi r / 4 r = \pi / 2$ radians
  - Solid angle of an hemisphere: $4 \pi r^2 / 2 r^2 = 2 \pi$ steradians

Radiant flux

- *Radiant flux* (watt) is the amount of light passing through (leaving or reaching) an area or a volume in the unit of time:

$$\Phi = \frac{dQ}{dt}$$

Irradiance

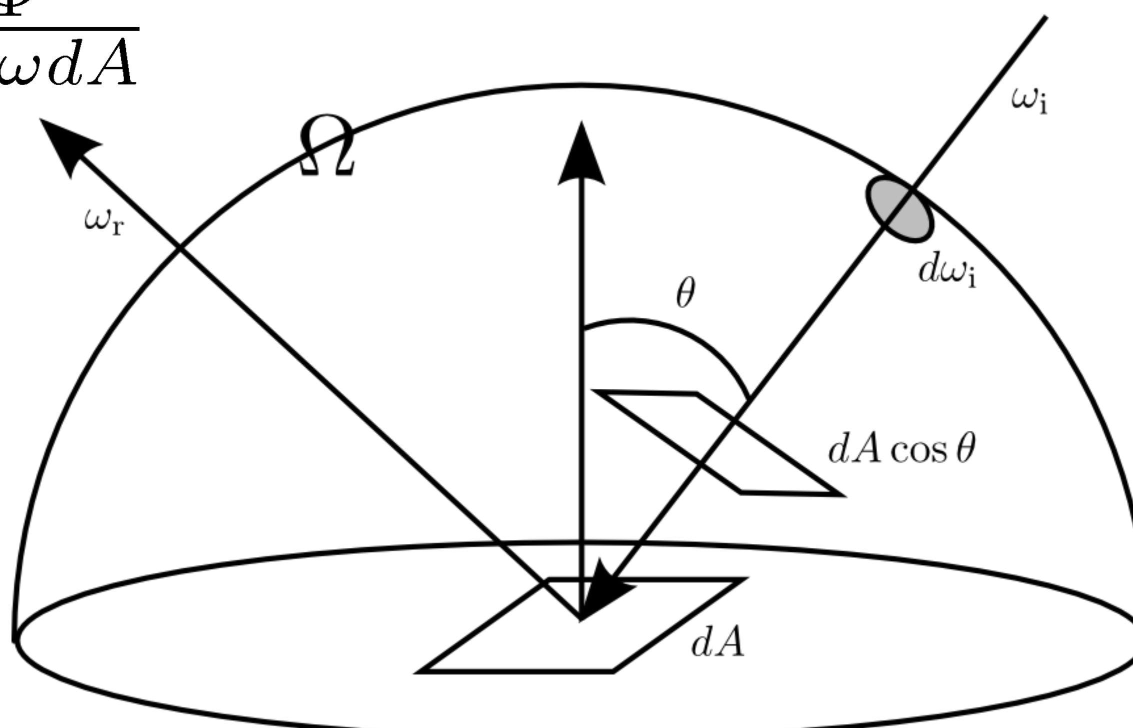
- *Irradiance* (*E*) (measured in *watt/m*^{2}) is the radiant flux incident on a surface element per unit area:
- *Exitance* (indicated also with *E*) is the flux leaving a surface per unit area:

$$E(x) = \frac{d\Phi}{dA}$$

Radiance

- *Radiance* (*L*) (*watt/m²sr*) represents the flow of radiant energy from (or to) a surface per unit projected area and per unit solid angle:

$$L(x, \vec{\omega}) = \frac{d^2\Phi}{\cos\theta \, d\omega \, dA}$$



Radiance and Irradiance

- The integral of the incoming radiance along all the possible directions corresponds to the irradiance:

$$E(x) = \int_{\Omega} L(x, \vec{\omega}')(\vec{\omega}' \cdot n) d\vec{\omega}'$$

Rendering a 3D scene

- ***Rendering*** is the processing to generate a synthetic image starting from some data.
- Rendering of realistic 3D scenes is obtained by defining the light sources, the geometry of the scene and the reflection properties of each surface of the scene.
- A common way to specify the reflection property of a surface is the *Bidirectional Reflectance Distribution Function (BRDF).*

Bidirectional Reflectance Density Function (BRDF)

- The BRDF function is defined as the ratio between the radiance and the (directional) irradiance:

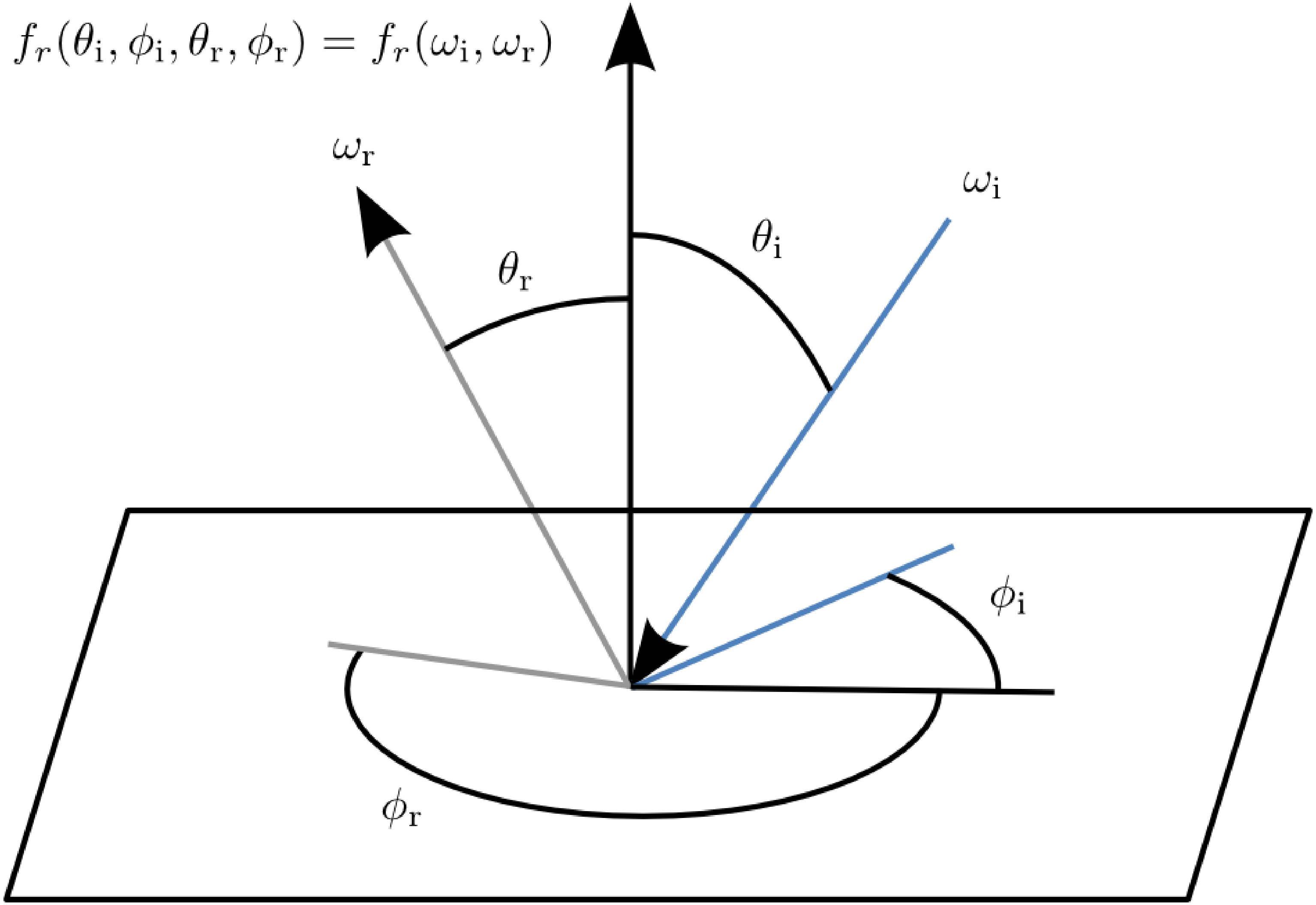
$$f_r\left(\omega_{\mathrm{i}}, \omega_{\mathrm{r}}\right) = \frac{dL\left(\omega_{\mathrm{r}}\right)}{dE_{\mathrm{i}}\left(\omega_{\mathrm{i}}\right)}$$

Bidirectional Reflectance Density Function (BRDF)

- Taking into account the relation between radiance and irradiance it is possible to write the BRDF as a function of ingoing and outgoing radiance only

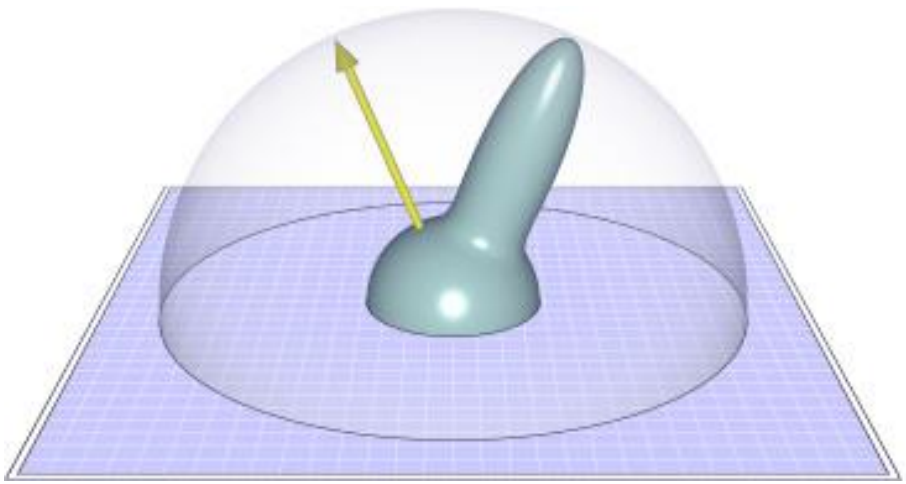
$$f_r(\omega_{\mathrm{i}}, \omega_{\mathrm{r}}) = \frac{dL(\omega_{\mathrm{r}})}{L(\omega_{\mathrm{i}}) \cos\theta_i d\omega}$$

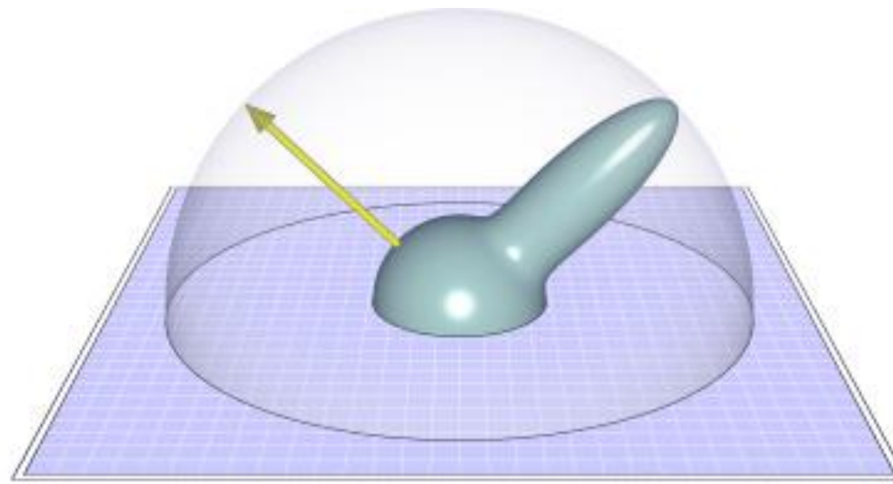
BRDF parameterization



BRDF example

- BRDF is a 4D function
- Here, a visualization of a 2D slice:





BRDF Properties

- Energy conservation:

$$\int_{\omega_{\mathrm{r}} \in \Omega} f_r\left(\omega_{\mathrm{i}}, \omega_{\mathrm{r}}\right) \cos \theta_{\mathrm{r}} d\omega \leq 1$$

- ***Helmholtz reciprocity*** (simmetry of the light transport):

$$f_r\left(\omega_{\mathrm{i}}, \omega_{\mathrm{o}}\right) = f_r\left(\omega_{\mathrm{o}}, \omega_{\mathrm{i}}\right)$$

Isotropic vs Anisotropic BRDF

- Many real world surfaces exhibit invariance with respect to the rotation of the surface around the normal vector at the incident point.

Isotropic $f_r\left(\theta_{\mathrm{i}}, \phi_{\mathrm{i}}, \theta_{\mathrm{r}}, \phi_{\mathrm{r}}\right)$

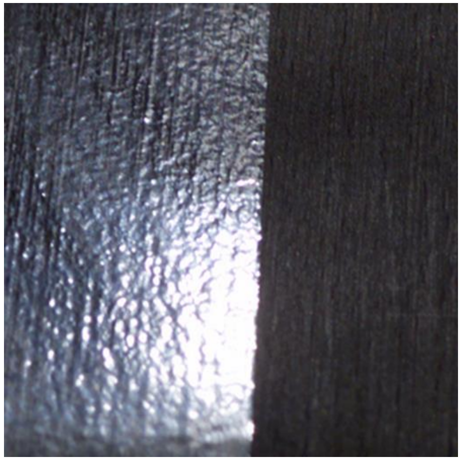
Anisotropic
$$\begin{cases} f_r\left(\theta_{\mathrm{i}}, \phi_{\mathrm{i}}, \theta_{\mathrm{r}}, \phi_{\mathrm{r}}\right) = f_r\left(\theta_{\mathrm{i}}, \theta_{\mathrm{r}}, \phi\right) \\ \phi = \phi_{\mathrm{i}} - \phi_{\mathrm{r}} \end{cases}$$

Materials

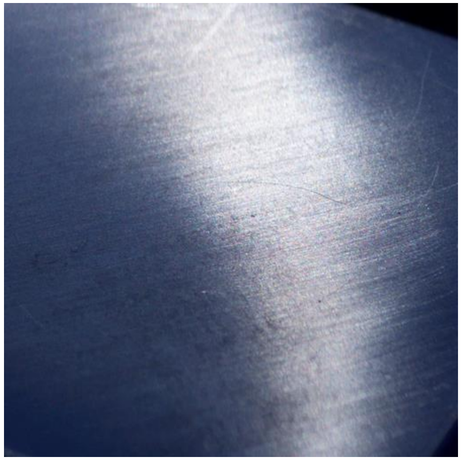
Diffuse reflection

Subsurface scattering

Mirror reflection



Layered-material



Anisotropic reflection

The rendering equation

- A rendering algorithm, to generate the synthetic image must compute radiance contributions from each surface point to the virtual camera.

$$L(\omega_{\mathrm{r}}) = \int_{\omega_{\mathrm{i}} \in \Omega} dL(\omega_{\mathrm{r}}) = \int_{\omega_{\mathrm{i}} \in \Omega} f_r(\omega_{\mathrm{i}}, \omega_{\mathrm{r}}) \, L(\omega_{\mathrm{i}}) \cos\theta_{\mathrm{i}} \, d\omega$$

→ **All these lighting contributions are local (!)**

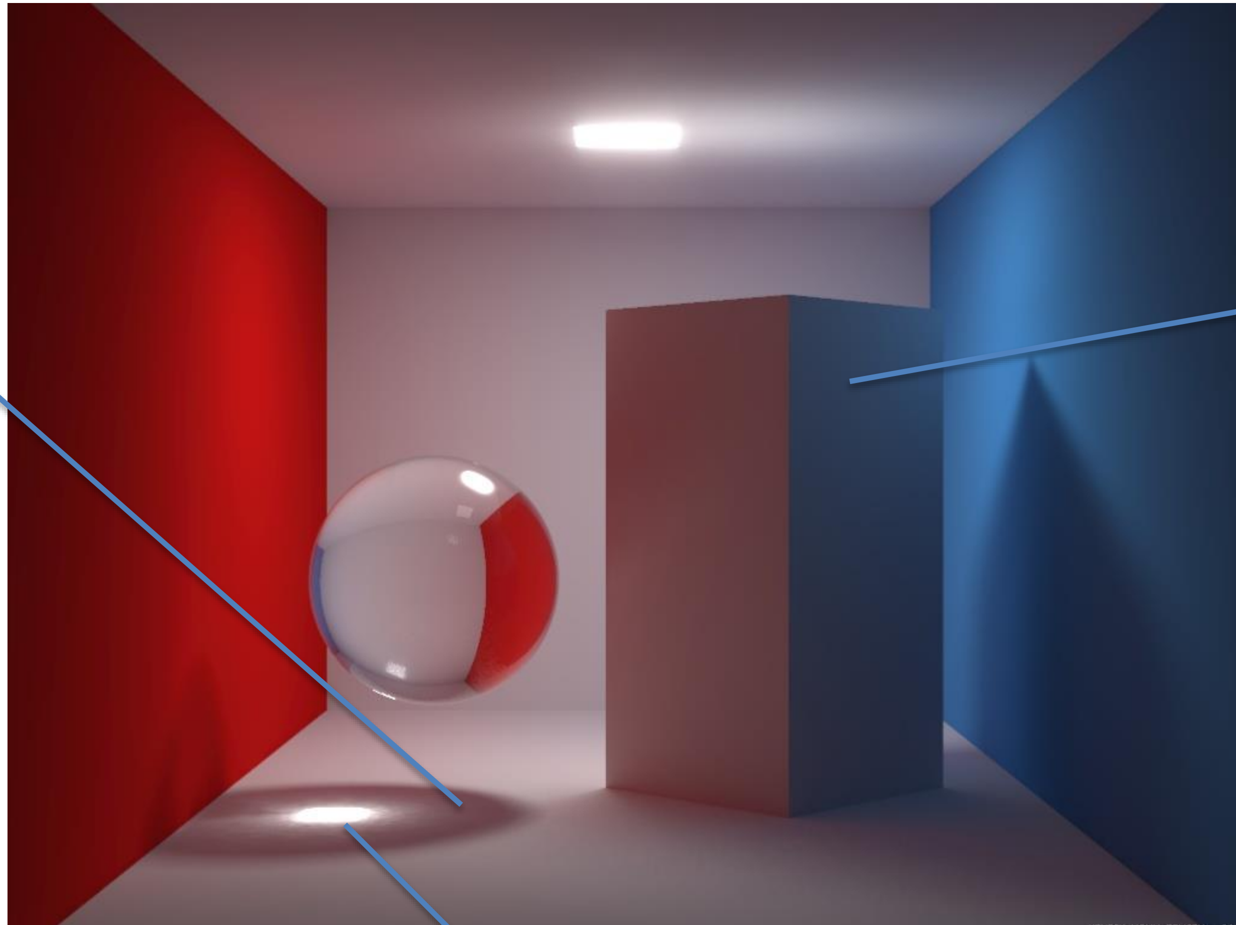
Generalized Rendering equation

Emitted light

Reflected light

$$\begin{aligned} L(\omega_\mathrm{o}) &= L_\mathrm{e}(\omega_\mathrm{o}) + L_\mathrm{r}(\omega_\mathrm{o}) \\ &= L_\mathrm{e}(\omega_\mathrm{o}) + \int_{\omega_\mathrm{i}\in\Omega} f_r(\omega_\mathrm{i}, \omega_\mathrm{o})\, L(\omega_\mathrm{i}) \cos\theta_\mathrm{i} d\omega \end{aligned}$$

Global and local effects



shadows

Color bleeding

caustics

Images from Henrik Wann Jensen

Generalized Rendering equation

Visibility Term

$$L\left(x,\omega_{\mathrm{o}}\right)=L_{\mathrm{e}}\left(\omega_{\mathrm{o}}\right)+\int_{\omega_{\mathrm{i}}\in\Omega}f_{r}\left(\omega_{\mathrm{i}},\omega_{\mathrm{o}}\right)L\left(\omega_{\mathrm{i}}\right)V\left(x,\omega_{\mathrm{i}}\right)\cos\theta_{\mathrm{i}}d\omega$$

Bidirectional Scattering Surface Reflectance Distribution Function (BSSRDF)

$$f_r(\omega_\mathrm{i}, \omega_\mathrm{o}) \longrightarrow f_r(x_\mathrm{i}, x_\mathrm{o}, \omega_\mathrm{i}, \omega_\mathrm{o})$$

- BSSRDF is an 8D function, BRDF is a 4D function.
- It can be plugged in the rendering equation.

Acquisition through Gonioreflectometer

- A gonioreflectometer consists of a moving light source and a photometric sensor.

Moving light source **Orientable material sample**

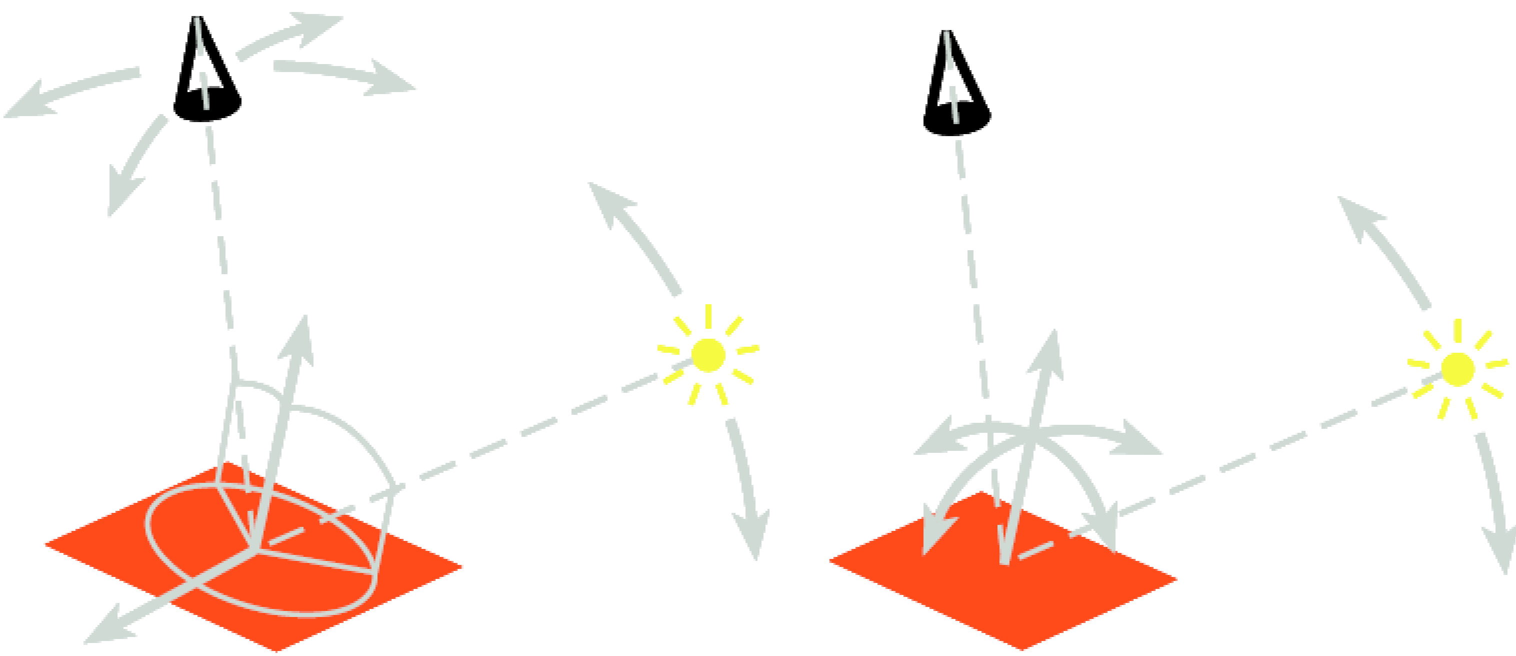


Figure from Hendrik P.A. Lensch, "Efficient, Image-Based Appearance Acquisition of Real-World Objects. PhD Thesis, 2003.

Limitation

- Gonioreflectometers are relatively slow since the sensor and the light source have to be repositioned for every pair of incident and outgoing direction (one sample at time).
- BRDF measured “a single material” at a time → *spatially-varying materials* require a lot of effort to be captured.

Image-based Acquisition

- Using images, instead of using other sensors, permits to capture more BRDF samples at time.

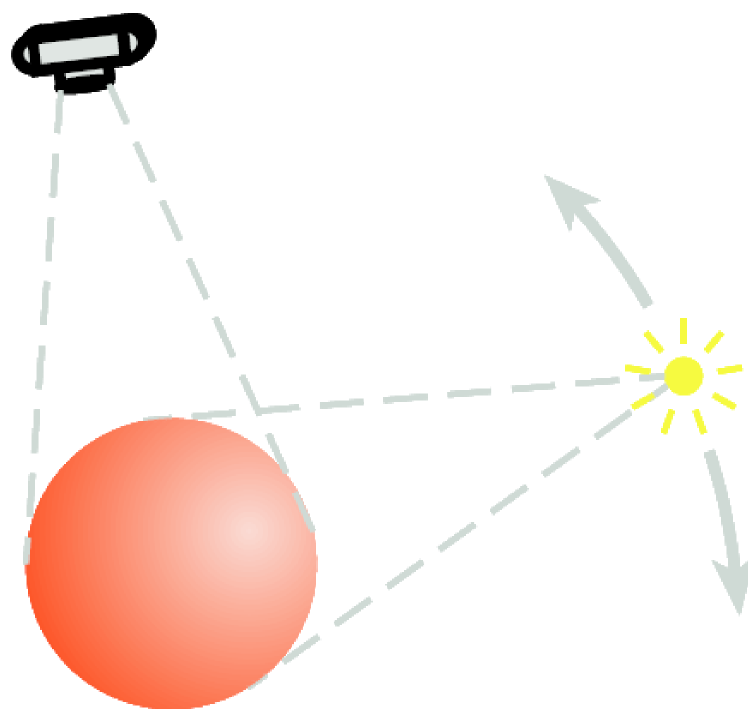


Figure from Hendrik P.A. Lensch, “Efficient, Image-Based Appearance Acquisition of Real-World Objects. PhD Thesis, 2003.

Image-based Acquisition

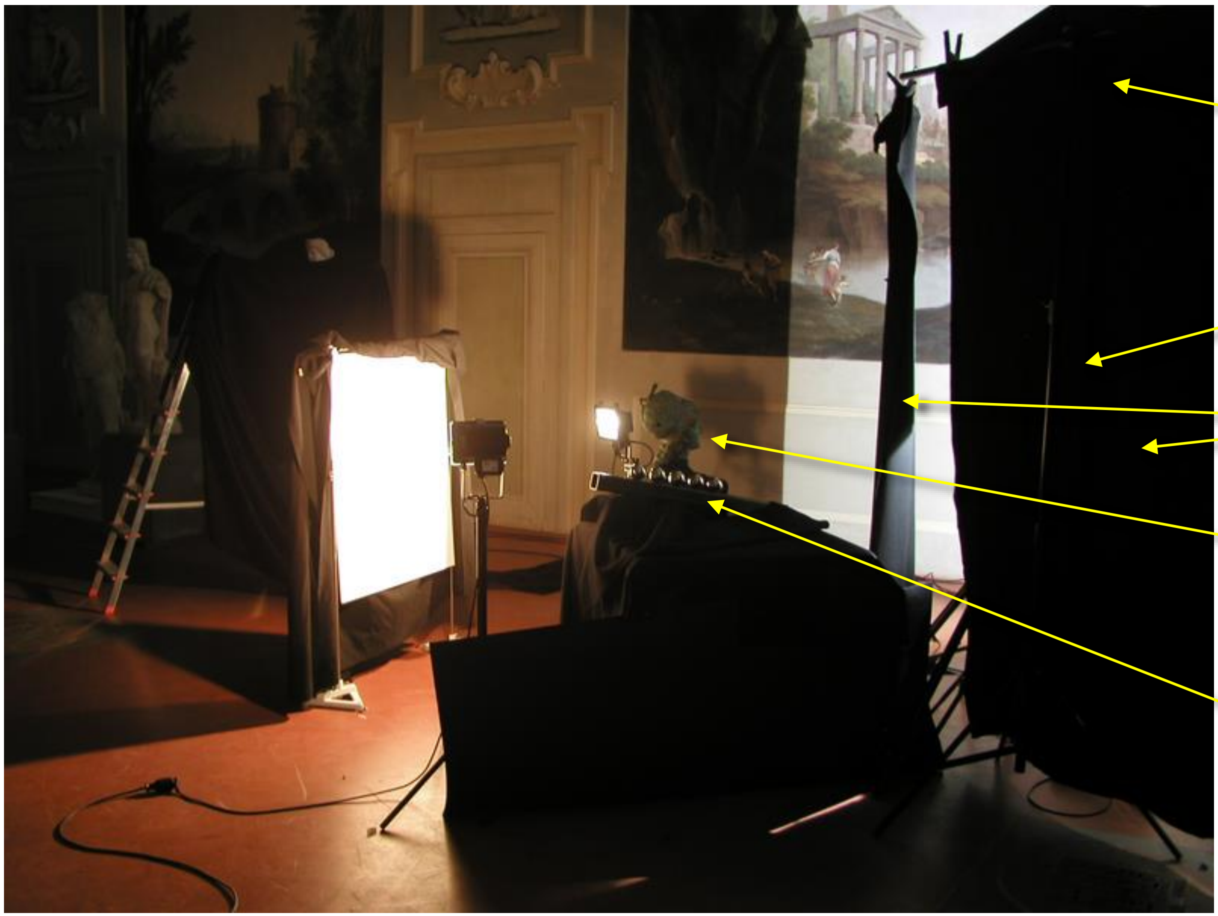
- High Dynamic Range (HDR) images may be preferably used.
- Images should be aligned with the geometry to obtain the surface normal → **image-geometry registration (**also called **2D/3D registration).**
- Intensive data processing.

Image-based Acquisition

Stanford Spherical Gantry

Image-based Acquisition

On-site acquisition setup (MPI Saarbucken, by M. Goesele et al.)



BRDF Representation

- The most common representation is to store the data acquired in ***tabulated form*** (discretized directions) and interpolate.
- Enormous amount of storage to get high accuracy (!)
- <u>Undersampling problems.</u>

Limitations

- Flexibility
  - The moving setup is always very cumbersome.
  - How to cope with large objects ?
- Time required (several hundreds images to process)
- Controlled lighting conditions ?
  - Field conditions have a lot of difficulties (museums, archeological sites)

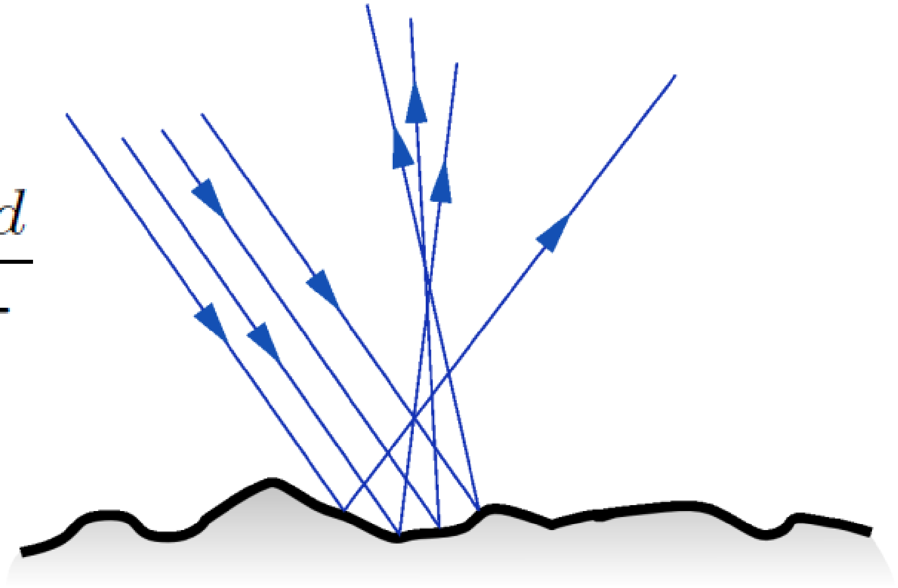
Parametric Acquisition/Representation

- Parametric acquisition consists in assume a model for the BRDF and estimate its parameters (nonlinear system) → *sparse vs dense* acquisition.
- If a particular model is not assumed, the BRDF can be factorized with different basis functions and its coefficients estimated.
- Parametric representation/factorization can be used also to obtain a compact form starting from a dense acquisition.

BRDF Models

- **DIFFUSE REFLECTION:** light is equally diffuse in every direction due to the uniform random microfacet of the surface
- Lambertian materials.

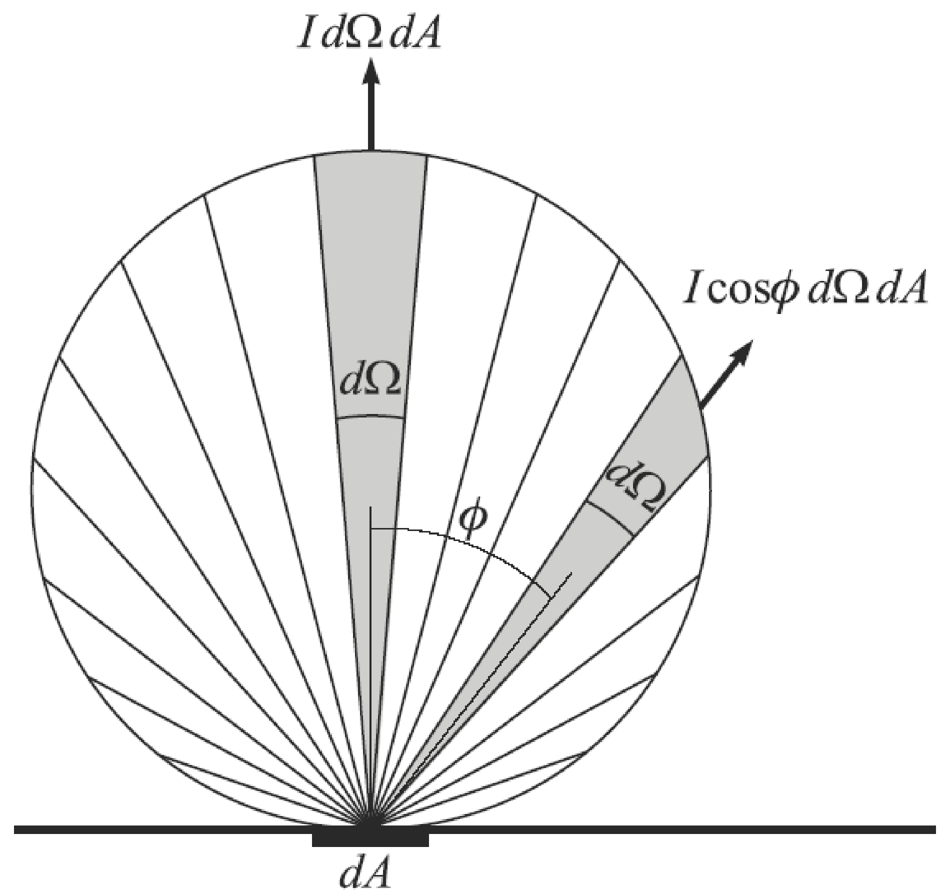
$$f_r(\omega_\mathrm{i}, \omega_\mathrm{o}) = k_D = \frac{\rho_d}{\pi}$$



Diffuse reflection

$$L_{\mathrm{diffuse}} = L_{\mathrm{incident}} k_{\mathrm{diffuse}} \cos\theta$$

- ***Lambertian Law:*** a diffusive surface reflects an amount of light which depends on the incident light direction. Maximum when perpendicular, then it reduces as cosine of the angle between the normal and the direction of incidence.



BRDF Models

- ***LAFORTUNE REFLECTION MODEL:*** It consists of a diffuse terms and a set of specular lobes.

$$f_r\left(\omega_{\mathrm{i}}, \omega_{\mathrm{o}}\right) = k_D + \sum_j \left[C_{x,j}\omega_{\mathrm{i},x}\omega_{\mathrm{o},x} + C_{y,j}\omega_{\mathrm{i},y}\omega_{\mathrm{o},y} + C_{z,j}\omega_{\mathrm{i},z}\omega_{\mathrm{o},z}\right]^{n_j}$$

- Depending on the coefficients, it can exhibit different reflection behaviours like *retro-reflection* and *anisotropic reflection*.

Local Reflection Models

- Solving the Rendering equation is complex (!)
- To make the computation easier the visibility is not taken into account (only direct light sources).
- One of the most used local illumination model is the Bui Tuong Phong (empirical):

$$L_{\text{reflected}} = k_A L_{\text{ambient}} + k_D L_{\text{diffuse}} + k_S L_{\text{specular}}$$

Phong Illumination Model

$$L_{\text{diffuse}} = L_{\text{incident}}(\boldsymbol{L} \cdot \boldsymbol{N})$$

$$L_{\text{specular}} = L_{\text{incident}} \cos(\alpha)^{n_s}$$

Blinn-Phong variant

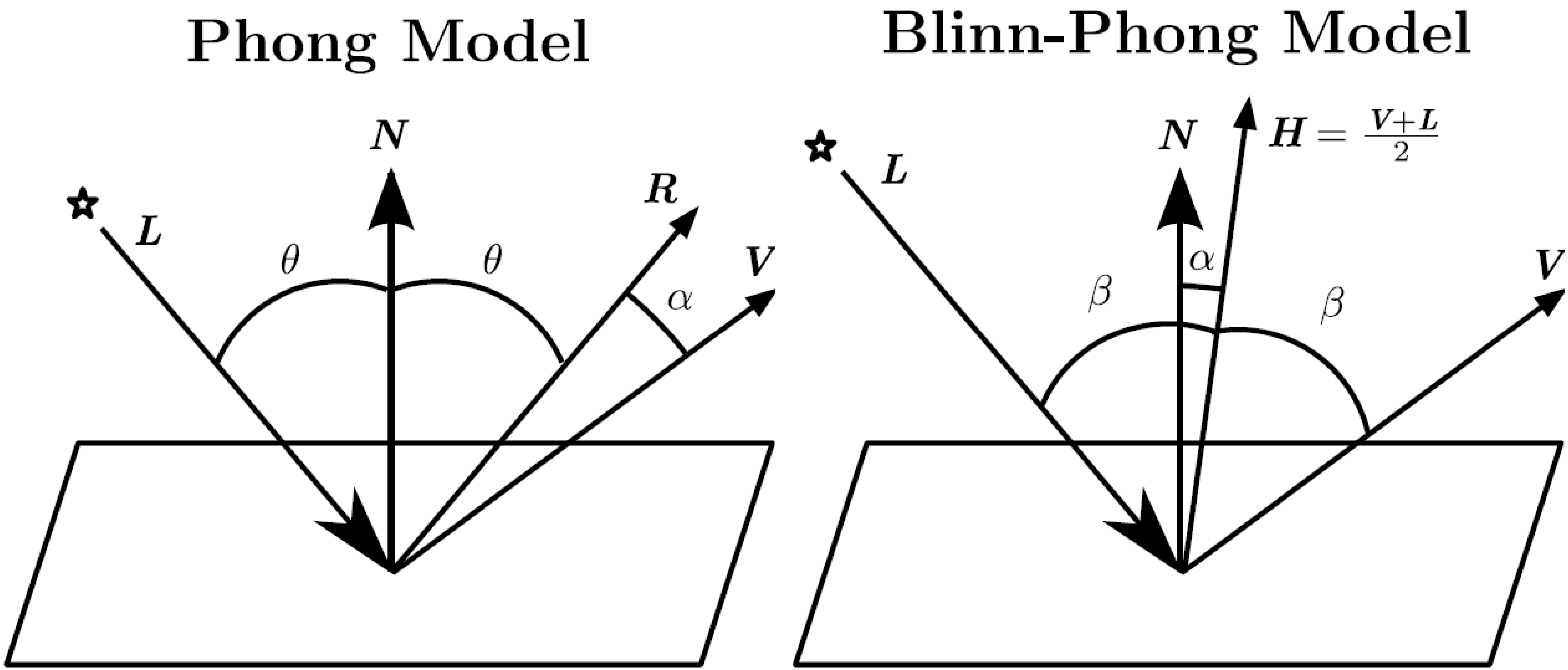
$$\cos\alpha = \boldsymbol{R} \cdot \boldsymbol{V}$$

$$\boldsymbol{R} = \boldsymbol{L} - 2(\boldsymbol{L} \cdot \boldsymbol{N})\boldsymbol{N}$$

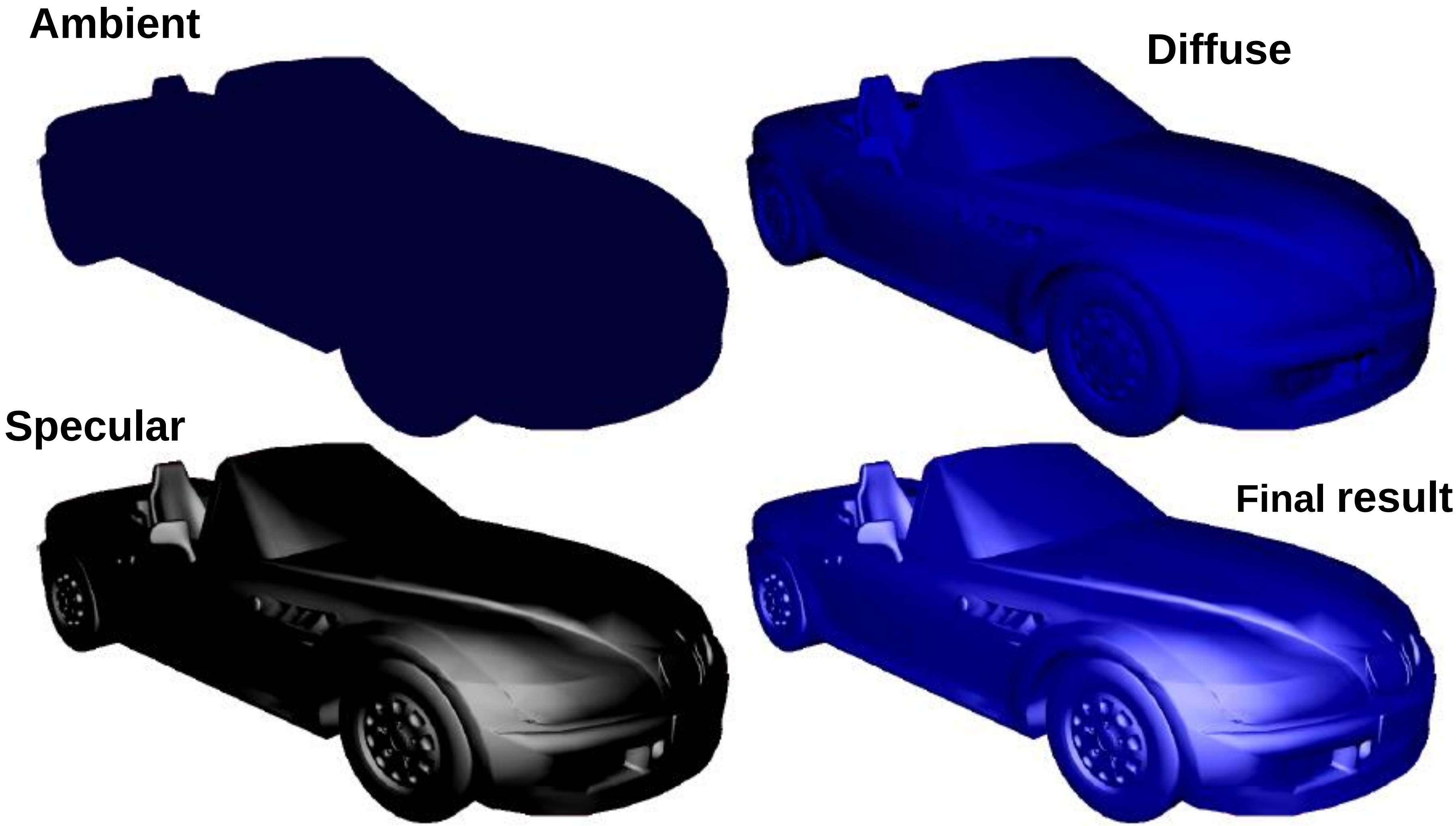
$$\cos\alpha = \boldsymbol{N} \cdot \boldsymbol{H}$$

$$\boldsymbol{H} = \frac{\boldsymbol{L} + \boldsymbol{V}}{2}$$

Blinn-Phong Model – Half Vector



Phong Illumination Model



Local Reflection Models

Phong

Cook-Torrance

Minnaert

Oren-Nayar

Rendering Paradigms

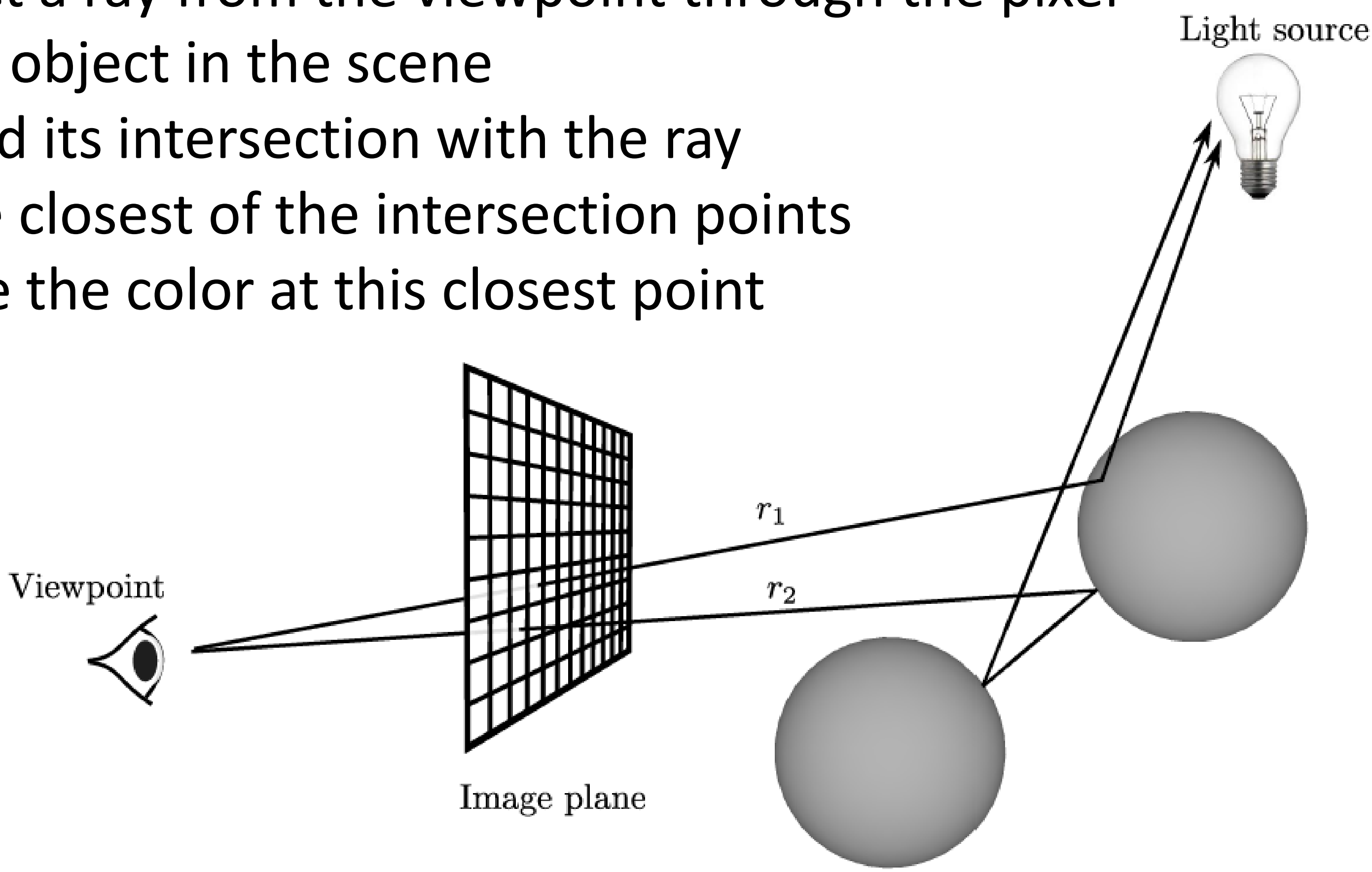
- Ray tracing/Path tracing → they solve the Rendering equation by integration.
- Rasterization-based Pipeline

Ray Tracing

BASIC RAY TRACING

For each pixel of the image :

1. Construct a ray from the viewpoint through the pixel
2. For each object in the scene
    2.1. Find its intersection with the ray
3. Keep the closest of the intersection points
4. Compute the color at this closest point



Ray Tracing

```
INPUT camera, scene
OUTPUT image
for row = 1 to rows
    for col = 1 to cols
        ray = getRay(row,col,camera)
        image[row][col] = getColor(ray, scene)
    endfor
endfor
```

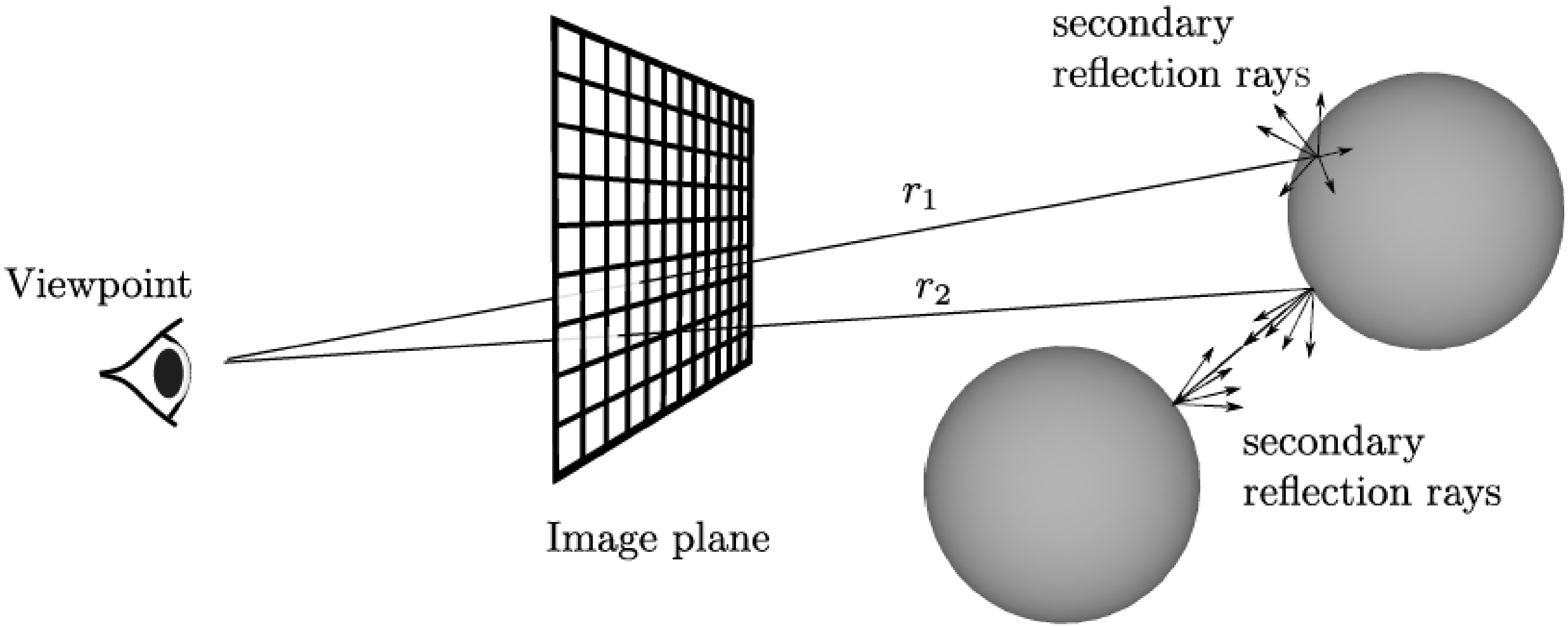
Compute color

Generate a ray

Classic Ray Tracing

```
function getColor()
    INPUT ray, scene
(t, object, intersectFlag) = raySceneIntersect(ray,scene)
if (intersectFlag == FALSE) return backgroundColor
for i=1 to #Lights
    shadowRay = computeShadowRay(ray,t,object,scene,lights[i])
    if (underShadow(t,ray,scene) == FALSE)
        color += computeDirectLight(t,ray,scene.lights[i])
    endif
endfor
if (isReflective(object)) // Inter-reflection support
    newRay = reflect(ray,t,object)
    color += object.specularReflectance * getColor(newRay,scene)
endif
if (isRefractive(object)) // transparency support
    newRay = refract(ray,t,object)
    color += object.transparency * getColor(newRay,scene)
endif
return color
```

Path Tracing

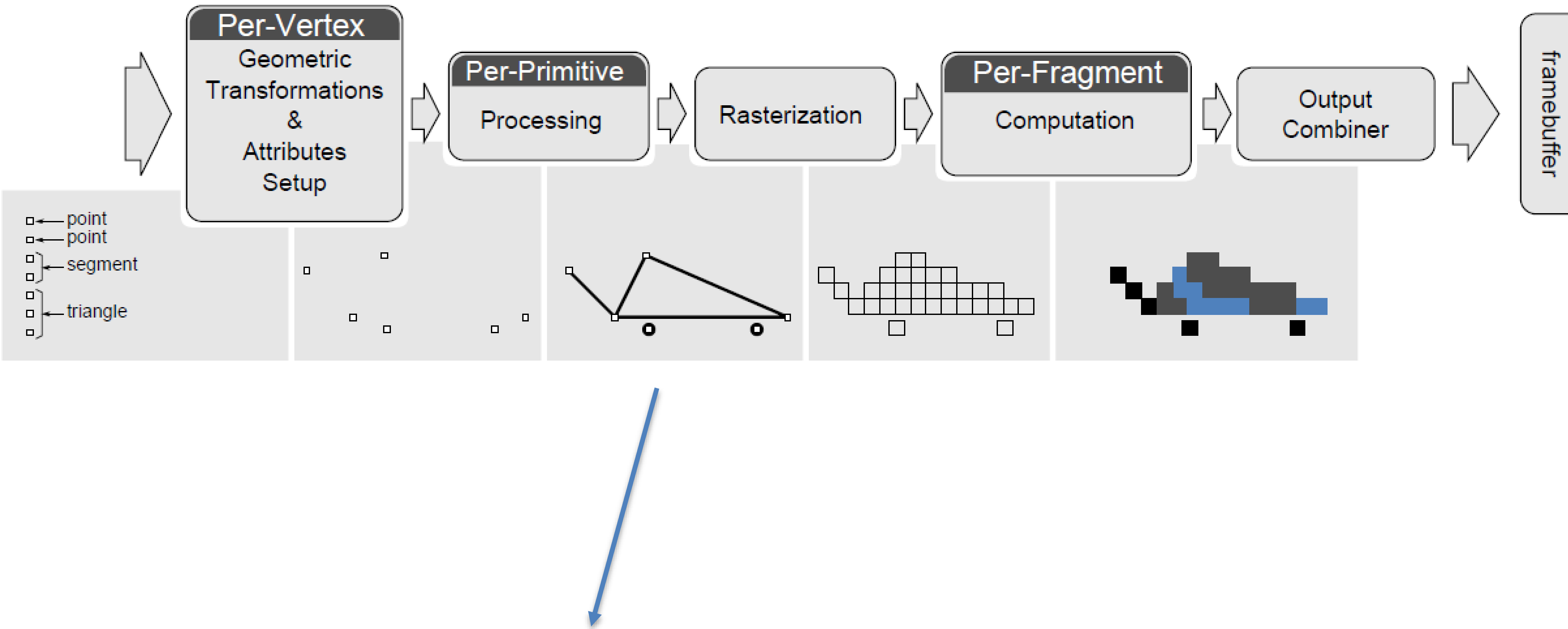


Path Tracing is a *Monte Carlo* based ray tracing method that renders global illumination effects in scenes with arbitrary surface properties.
The rays are recursively generated from the primary rays (paths are formed).
The difference with Classic Ray Tracing is that the secondary reflection rays are not restricted to mirror like surfaces, and refracted rays are not restricted to transparent only surfaces.

Path Tracing

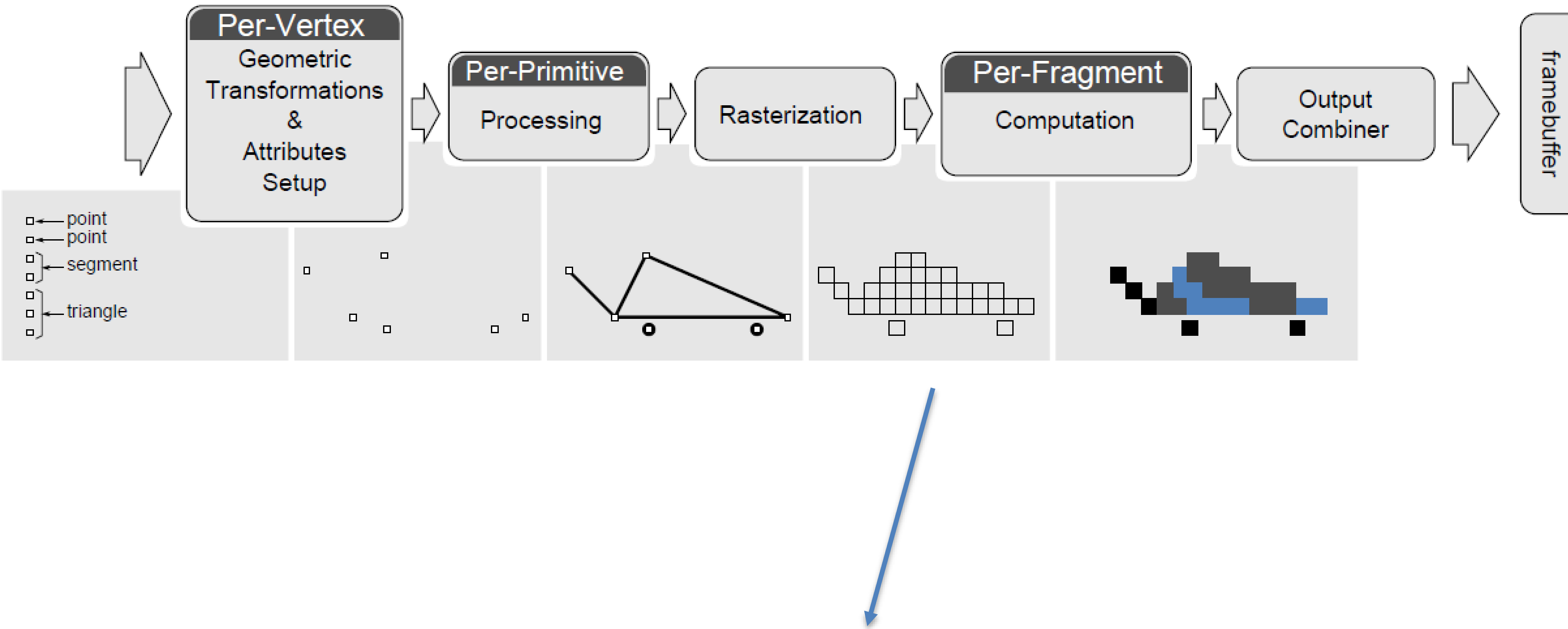
```
function getPathColor()
INPUT ray, scene

(t, object, intersectFlag) = raySceneIntersect(ray,scene)
if (intersectFlag==FALSE) return backgroundColor
color = black
for i=1 to #Lights
      shadowRay = computeShadowRay(ray, t,object,scene,lights[i])
      if (underShadow(t,ray,scene) == FALSE)
            color += computeDirectLight(t,ray,scene.lights[i])
      endif
endfor
if (isReflective(object)) // Inter-reflection support
      (newRay, factor) = sampleHemisphere(ray, t, object)
      color += factor * getPathColor(newRay, scene)
endif
return color
```

Rasterization-based Pipeline



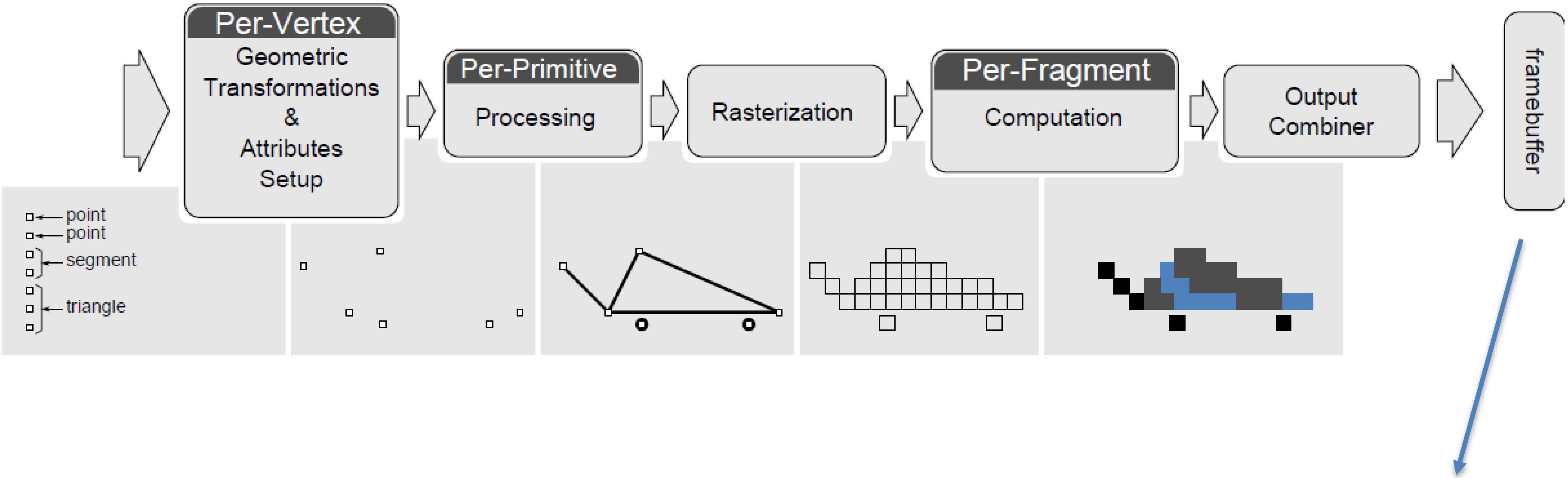
Geometric primitives are projected to the screen through geometric transformation.

Rasterization-based Pipeline



The *Rasterization stage* converts points, lines and triangles to their raster representation and interpolates the value of the vertex attributes of the primitive being rasterized (i.e. from polygons/lines/points to pixels).

Rasterization-based Pipeline



The *framebuffer* is the data buffer that stores the image during its formation.

Computer Vision

- It can be seen the inverse of Computer Graphics → from the image(s) to the data that generate this image(s).
- For example:
  - Estimate camera parameters
  - Estimate reflection properties
  - Estimate lighting environment
- But Computer Vision is more than this..

Computer Vision

- Segmentation
- Motion estimation
- 3D reconstruction
- Visual tracking
- Object recognition
- Human activity recognition
- Computational photography
- Etc..

CV-CG Interactions

- Strong interactions
- Share common knowledge (e.g. geometric transformation, image formation models)
- Many state-of-the-art Computer Vision algorithms are employed in Computer Graphics tools/applications

CV-CG Interactions

- Priors can be used in CG (e.g. synthesis)
- Concepts/algorithms developed to process images/video can be extended to 3D objects
- CV can benefit from CG knowledge (e.g. reflection models)

CV-CG Interactions

- Appearance acquisition of 3D objects → improve realism → one of the main motivation of 2D/3D registration.
- Image-based rendering.
- Computational Photography.

Image-based 3D Reconstruction

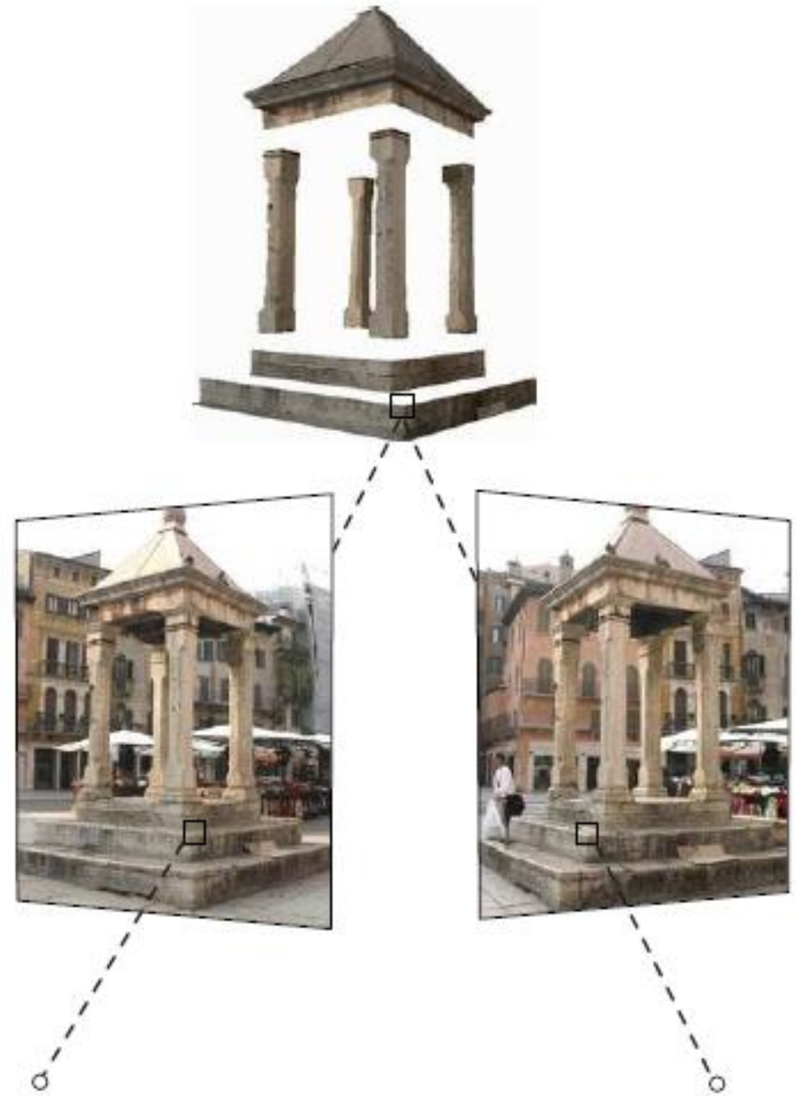
- We see that a real 3D object can be digitized/acquired
- This is possible using digital photographs only
- This involves typically two steps
  - Structure-from-Motion (SfM) – sparse reconstruction
  - Multi-View Stereo reconstruction – dense reconstruction

Structure From Motion (SFM)

From: Noah Snavely, Steven M. Seitz, Richard Szeliski, "Photo Tourism: Exploring image collections in 3D", ***SIGGRAPH 2006*****, 2006.**

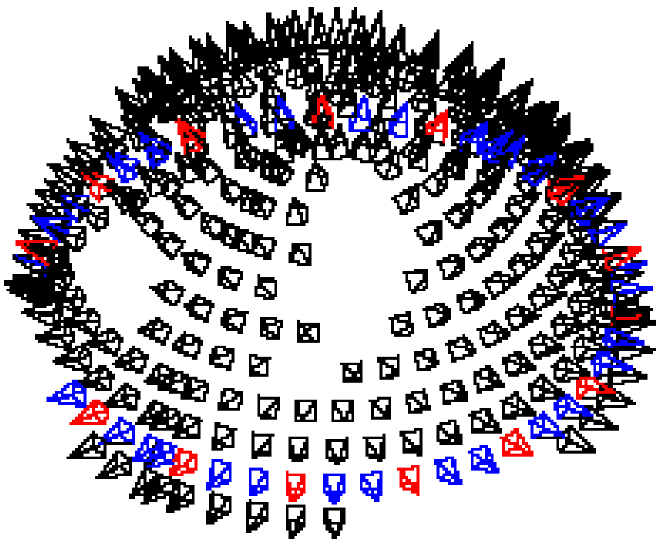
Structure From Motion (SFM)

- The idea is to find *correspondences* between images and use such information to calibrate the cameras. Also 3D points of the object are obtained as a result.



Figures from:
Andrea Fusiello, “Appunti Visione Computazionale”, Università di Udine.

Multi-View Stereo Reconstruction



Middlebury Multi-view Stereo Benchmark
http://vision.middlebury.edu/mview/

Image-based 3D Reconstruction

- Let’s me show some videos..

Importance of Features Extraction

Input Image(s)/Video

Features extraction

Corners, edges, keypoints, regions, ..

Specific Task

Stitching
Reconstruction
Recognition
…

Recap

- Some notions about rendering/3D acquisition.
- Registration is a basic step for model acquisition (registration of range maps, registration of point cloud).
- Registration is a basic step for appearance acquisition (2D/3D registration).
- Features extraction is at the base of registration and many other Computer Vision tasks.

Questions ?