Introduction of 3D Vision

Hao Fang

方昊

深圳大学计算机与软件学院

可视计算研究中心（VCC）: https://vcc.tech/index.html

地址：深圳大学沧海校区致真楼812

B.Sc: Beihang University

M.Sc: Beihang University

Engineer Program: Ecole Centrale Paris

Ph.D: Inria Sophia-Antipolis

Research of interest ： 3D Vision

Email: paulfanghao@gmail.com

Personal web: https://vcc.tech/~haofang

Lecture 1

Introduction

Computer Vision

"One picture is worth ten thousand words"

Human beings   

Machines   Algorithm

Computer Graphics

“Using machines to model / animate / render **virtual** objects”

Computer-Aided-Design

Computer Graphics = Modeling / Animating / Rendering “3D data” with “Algorithm”

3D Vision

"Using machines to understand or generate **real-world** models"

Reverse Engineering

3D Visison = Analyzing "3D data" with "Algorithm"

Prerequisites

Mathematics:

- Linear algebra: SVD, eigen values, eigen vectors, PCA...
- Geometry: point, 2D line, 3D plane, 2D polygon, intersections...
- Probabilistic & Statistic: Bayes, Maximum Likelihood, PDF...
- Optimization: convex, non-linear, discrete, IP, MRF, MCMC...

Coding:

- Data structure and algorithm
- C++: class, shared pointer, Polymorphisn...
- Python

Prerequisites

Computer Vision

- Basic Image Processing: SIFT, edge detection, image smoothing...
- High-level tasks: object detection, semantic segmentation...
- OpenCV

Deep Learning:

- Basic knowledge: loss function, networks, data preprocessing...
- Some tricks for training: batch training, regularization...
- PyTorch, Tensorflow

What you can learn from this course

The whole pipeline of 3D Vision

- 3D data representation
- 3D data reconstruction
- 3D data processing
- 3D data understanding
- Real-world applications and methods
- ***Useful tools:***
  - CGAL: mesh processing algorithms (C++)
  - PCL: point cloud processing algorithms (C++)
  - Open3D, PyTorch3D (Python)

Schedule

Lecture: 70% (me)

Paper reading and sharing: 30% (you)

Assignment: to be continued

Methodology

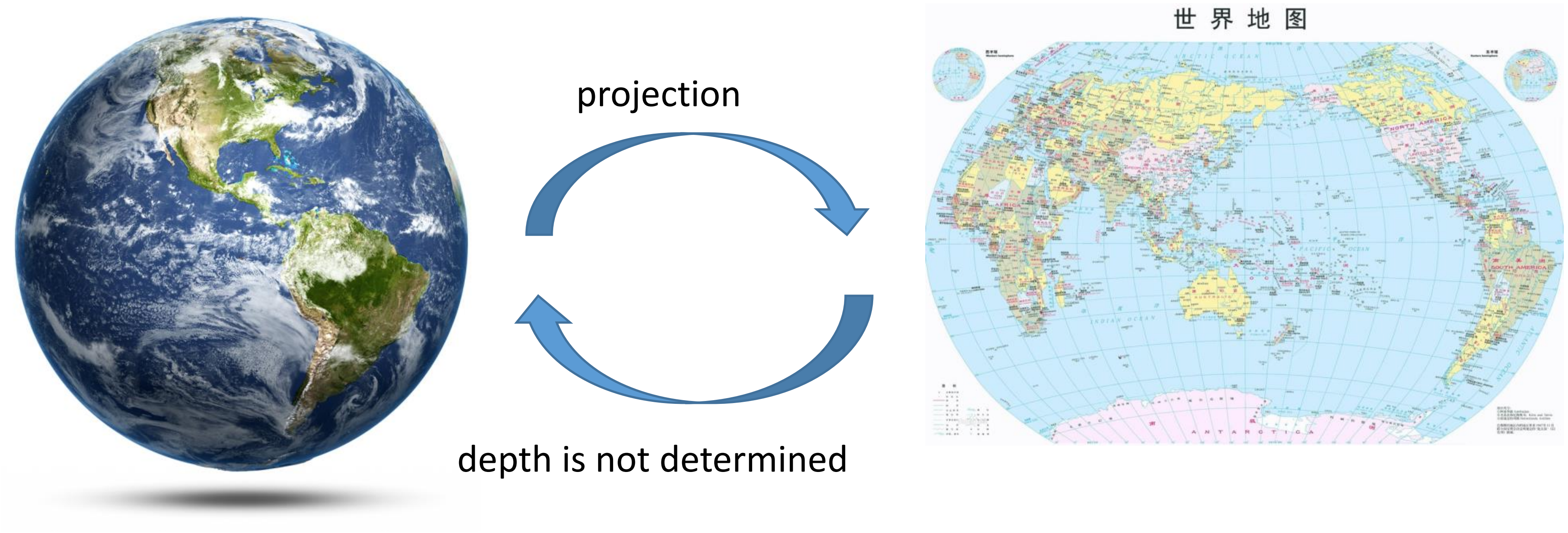
Whatever you learn, ask yourself three questions:

- Why? (the objective, potential applications...)
- What? (input, output, requirements...)
- How? (method, pipeline, technical details...)

Part 1

3D Data and Applications

2D vs 3D



It’s non-trivial to infer 3D infromation from 2D images, so 3D data is more appropriate for real-world applications!

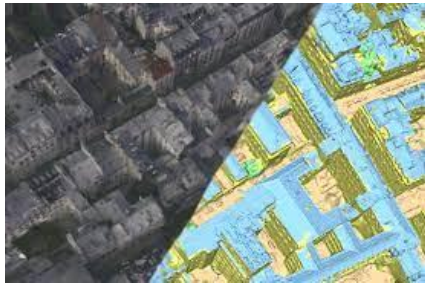
Computer Graphics & 3D Vision

Computer Graphics

Creating "virtual 3D model"

Rendering

3D Visison

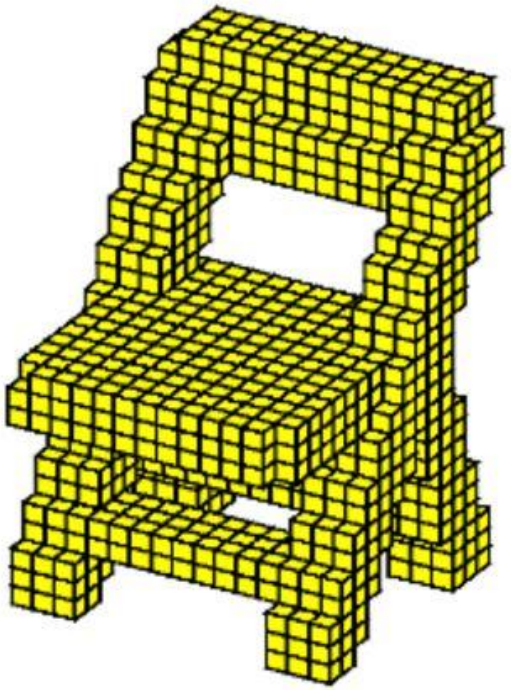


Generating "real-world 3D model"

Understanding

3D Data

❑ Voxel



H * W * D * 1

Pros:

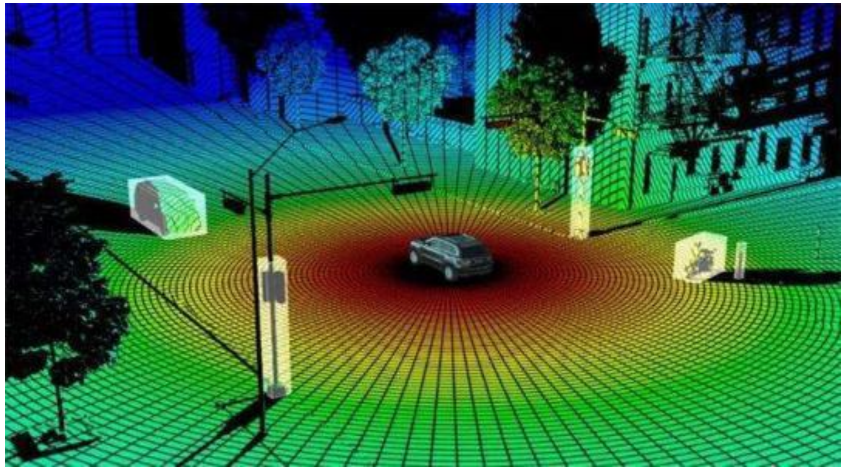
- Regular: 3D grids
- Directly applying 3D CNN

Cons:

- Trade-off between resolution and computational cost

3D Data

❑ Point Cloud



N x C (C=3 or 4)

Pros:

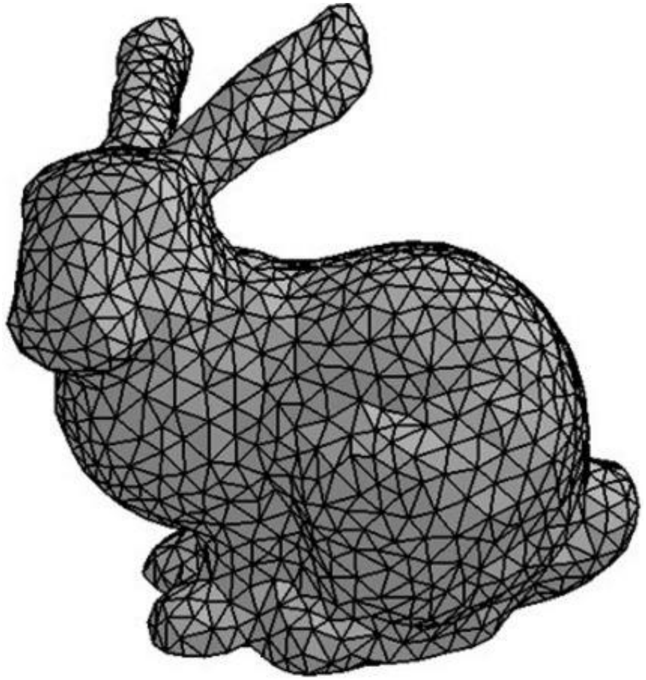
- Easy to collect: 3D scanners
- High accuracy w.r.t the surface of objects

Cons:

- Unordered: no direct deep feature extrators
- No topology between local points
- Artifacts: outliers, noise, holes...
- Cannot be used for rendering
- Difficult for storing and manipulating

3D Data

❑ Polygonal Mesh



Vertices, Eeges, Facets

Pros:

- Explicit topology information
- Used for Rendering
- Easy for manipulating and storing

Cons:

- Cannot be scanned directly： Sketchup,

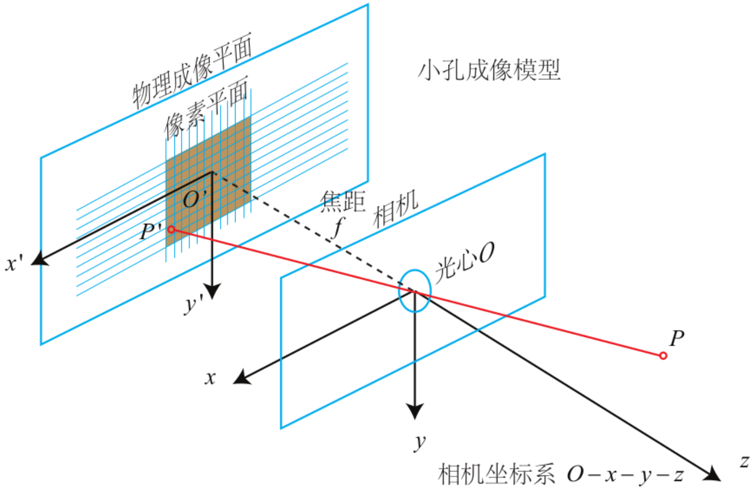
  Reverse Engineering

Sensors

RGB Cameras

- monocular
- stereo
- video
- panoramic

H * W * 3



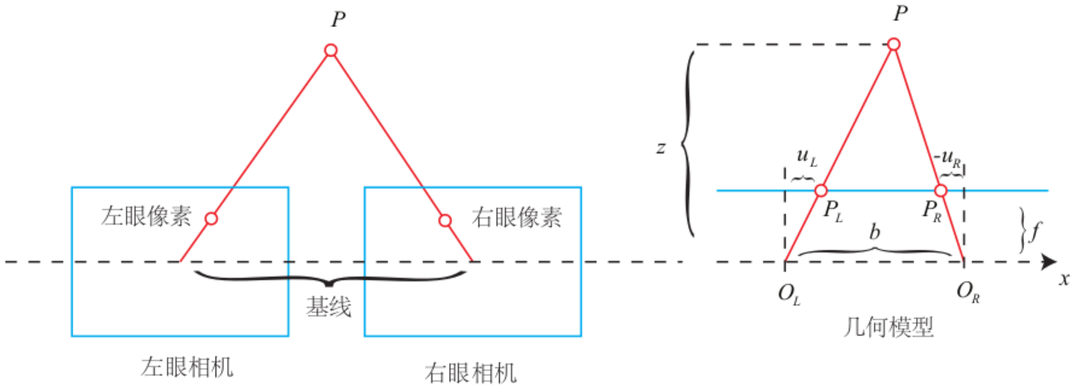
$$Z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Z (depth) is not determined

Sensors

RGB Cameras

- monocular
- stereo
- video
- panoramic



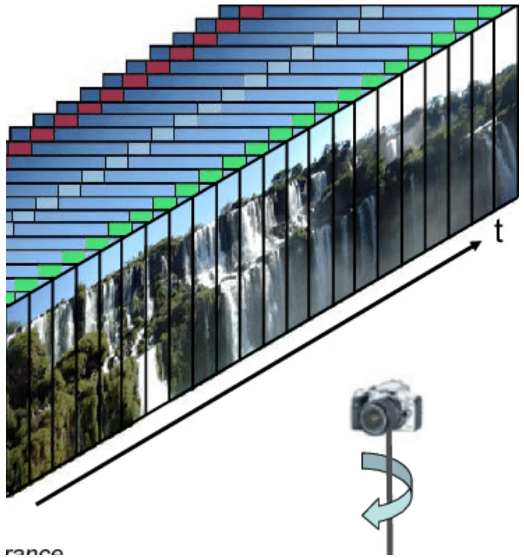
$$z = \frac{fb}{d}, \quad d = u_L - u_R$$

Z (depth) is determined by physical relation

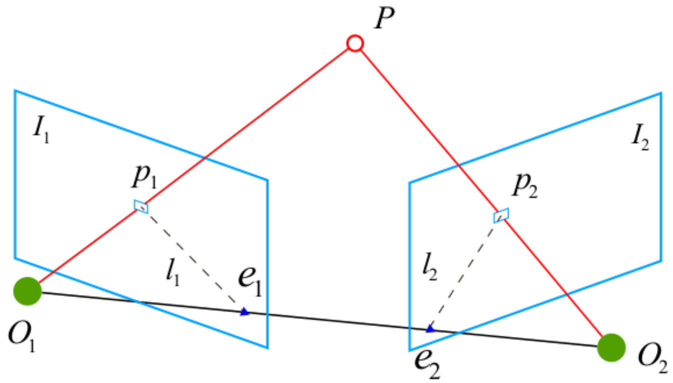
Sensors

RGB Cameras

- monocular
- stereo
- video
- panoramic



T * H * W * 3

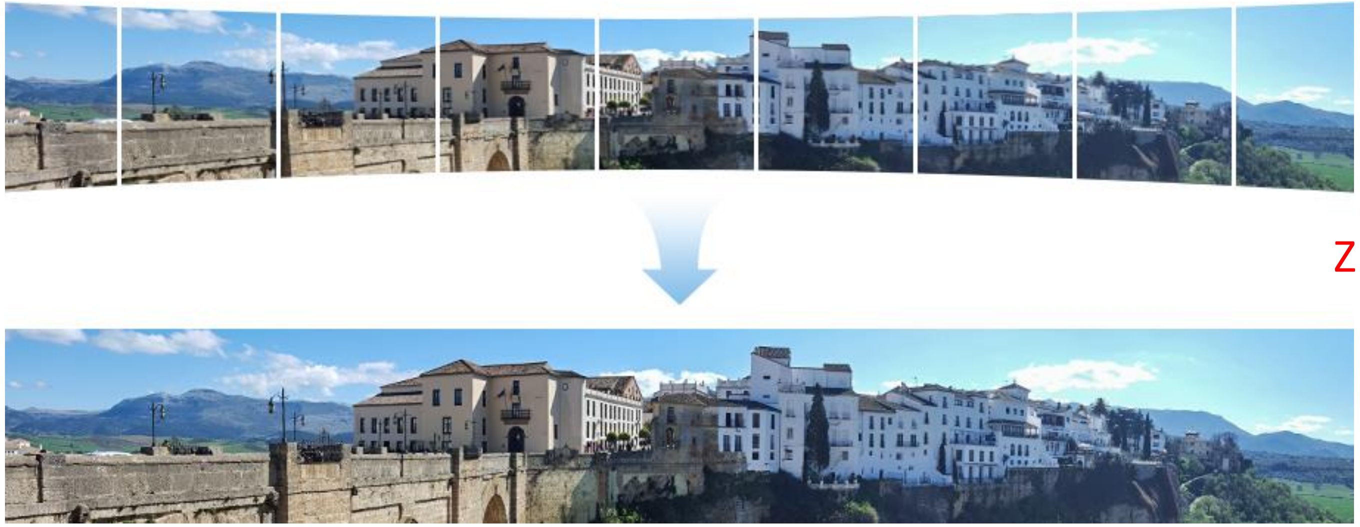


Z (depth) is determined by algorithms:
epipolar geometry and triangulation

Sensors

RGB Cameras

- monocular
- stereo
- video
- panoramic

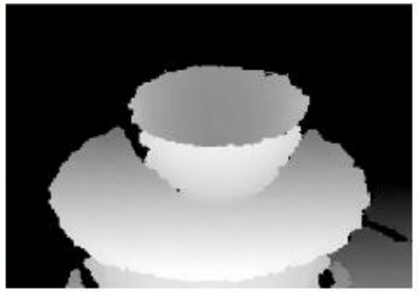


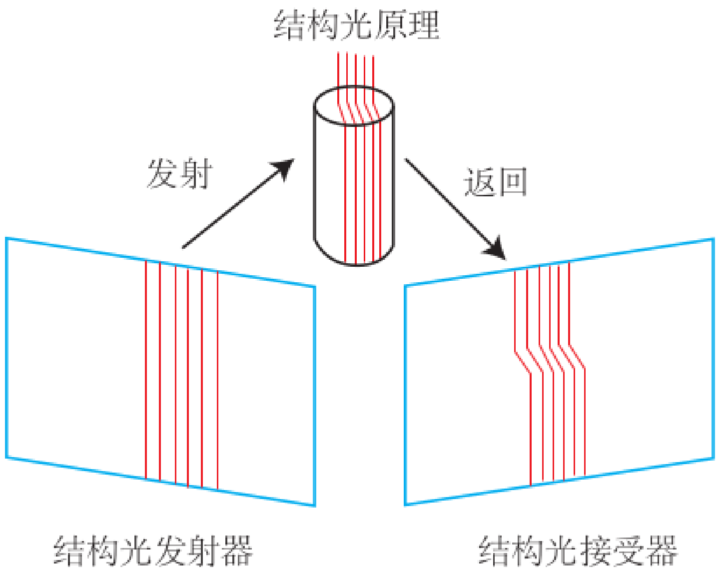
Z (depth) is not determined

Sensors

RGB-D (depth) cameras

- Structured Light: Kinect-1
- Time-of-flight: Kinect-2





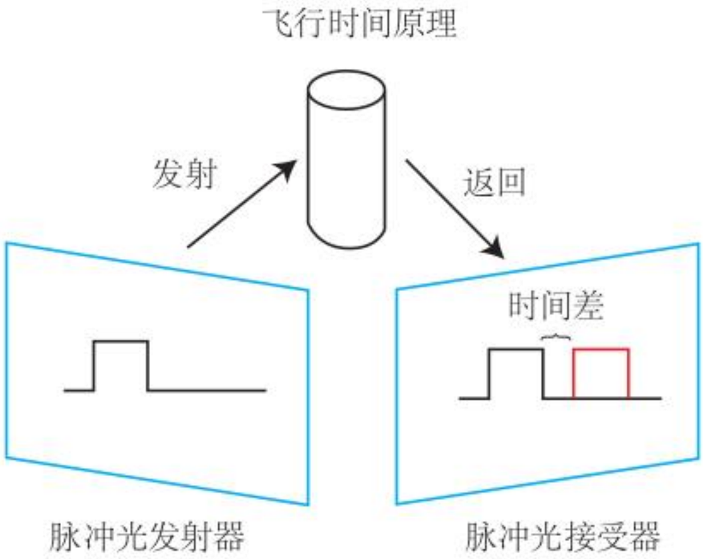
Z (depth) is determined

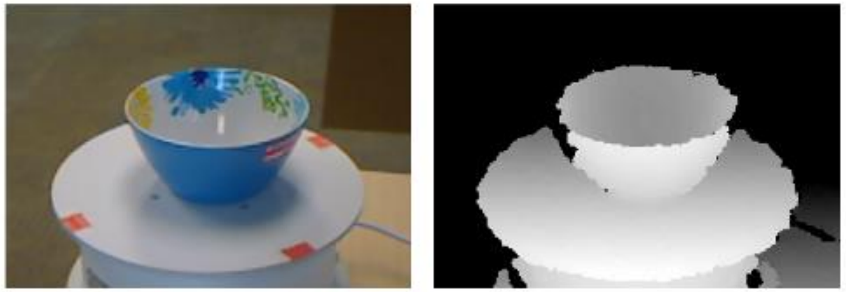
相机根据返回的结构光图案,计算物体离自身的距离

Sensors

RGB-D (depth) cameras

- Structured Light: Kinect-1
- Time-of-flight: Kinect-2





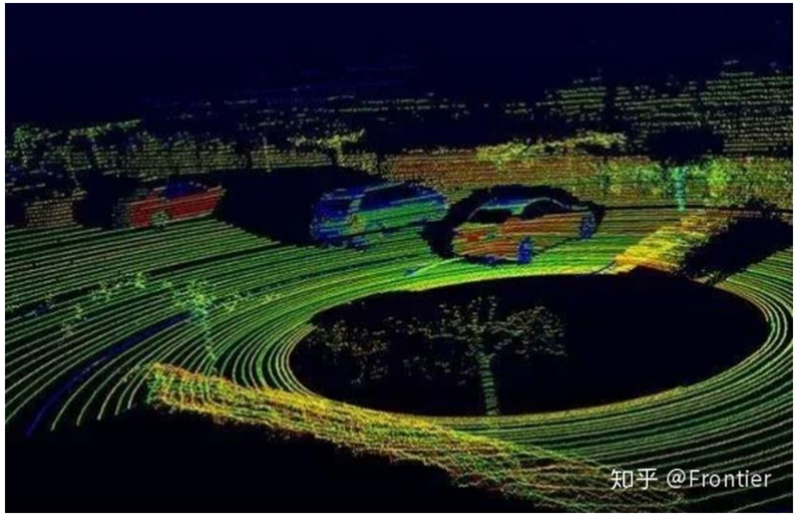
Z (depth) is determined

相机根据发送到返回之间的光束飞行时间,**确定物体离自身的距离**

Sensors

Laser scanners: Lidar

- 机械激光雷达
- 固态激光雷达

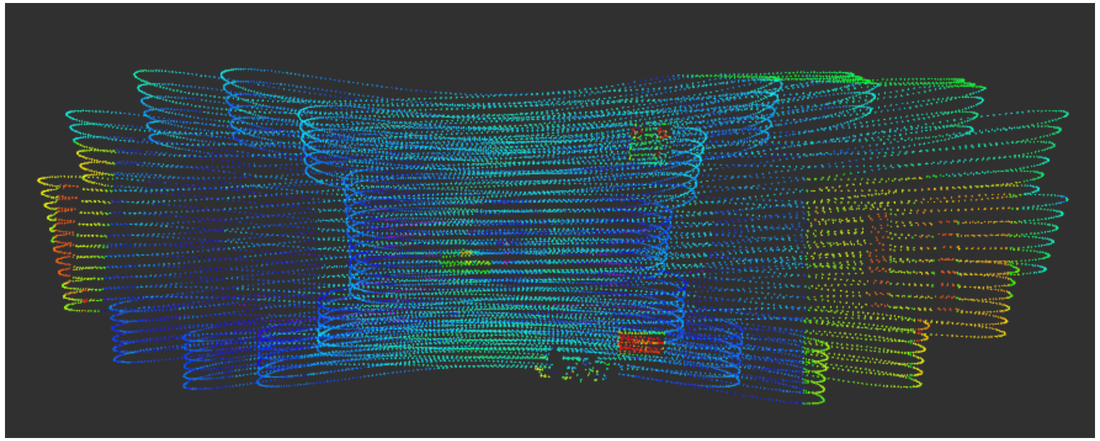


- expensive
- large size
- accurate

Sensors

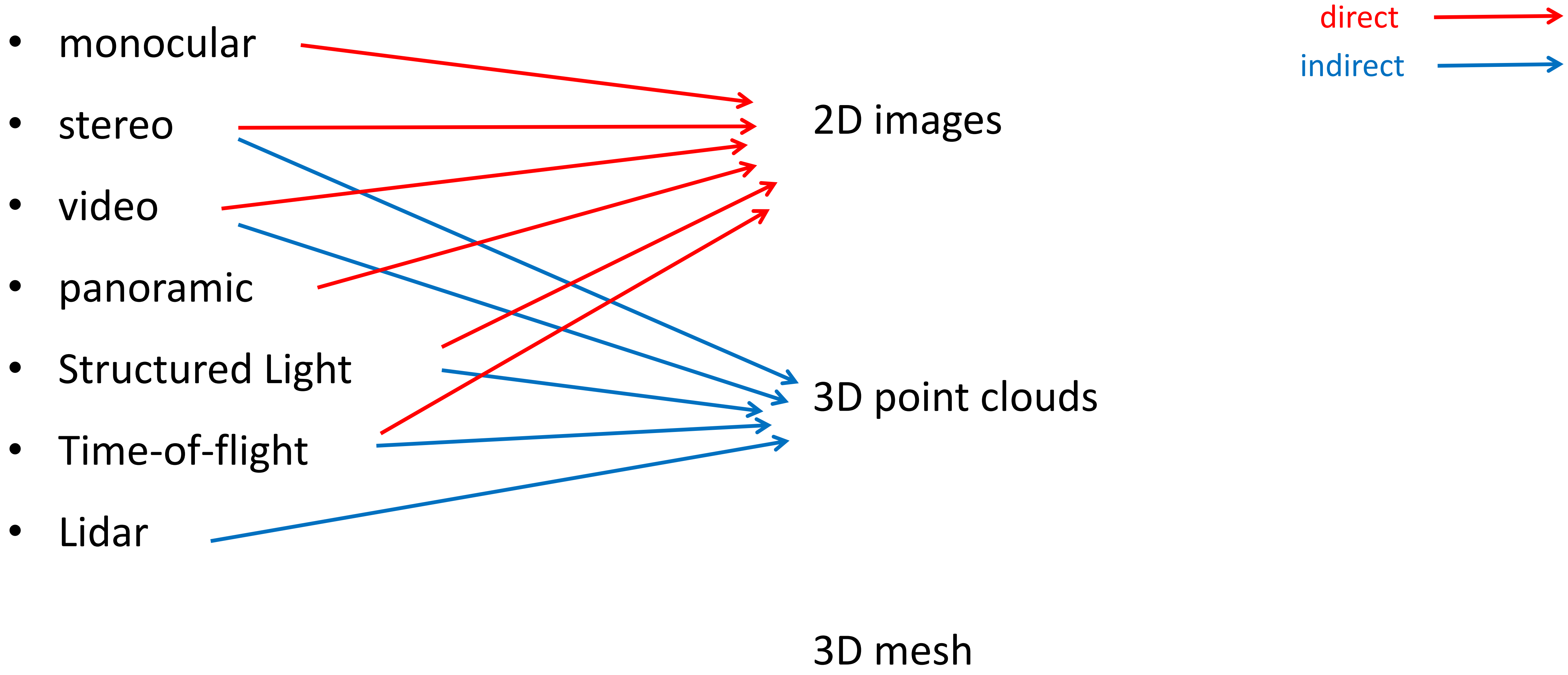
Laser scanners: Lidar

- 机械激光雷达
- 固态激光雷达



- cheap
- small size
- less accurate

Summary



direct
indirect
• monocular
• stereo
• video
• panoramic
• Structured Light
• Time-of-flight
• Lidar
2D images
3D point clouds
3D mesh

Applications

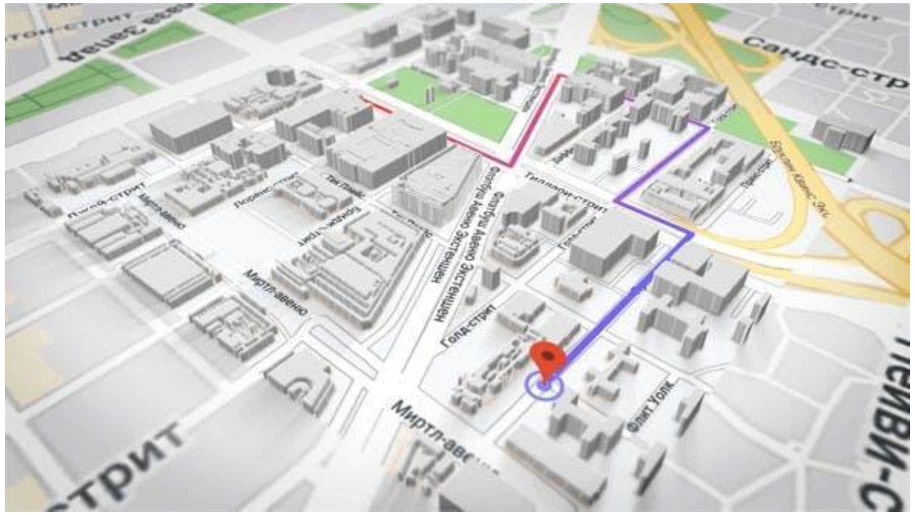
Smart City / Digital Twin

- urban planning
- mapping and navigation
- simulation and modeling
- security

Applications

Smart City / Digital Twin

- urban planning
- mapping and navigation
- simulation and modeling
- security



Applications

Smart City / Digital Twin

- urban planning
- mapping and navigation
- simulation and modeling
- security

Applications

Smart City / Digital Twin

- urban planning
- mapping and navigation
- simulation and modeling
- security

Applications

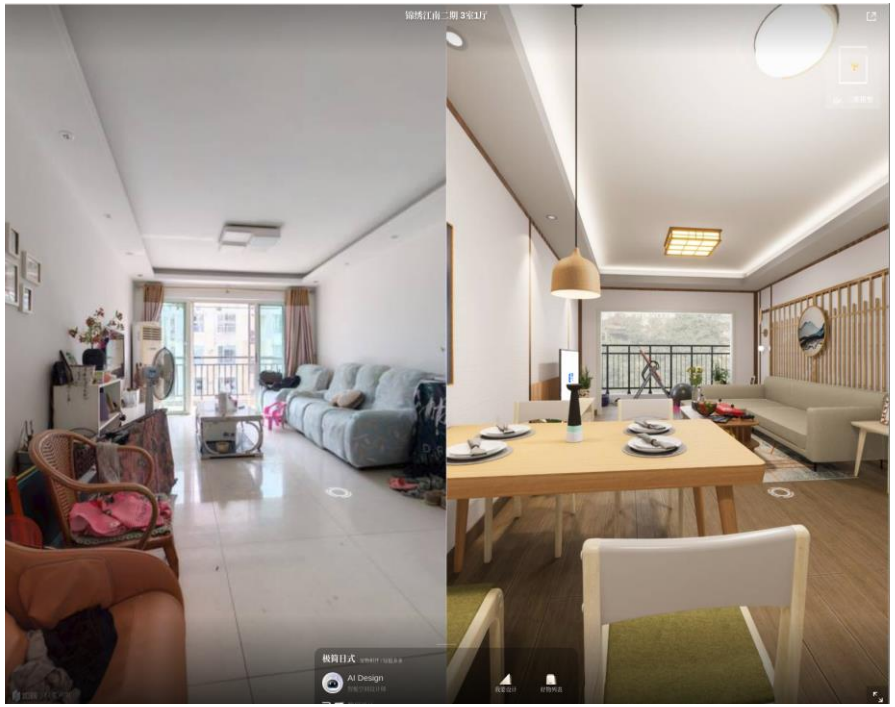
Smart Home

- navigation
- AI furnishing
- VR / AR

Applications

Smart Home

- navigation
- AI furnishing
- VR / AR



Applications

Smart Home

- navigation
- AI furnishing
- VR / AR

Applications

Industry

- autonoumous driving / robotics
- packing & transport
- design

Applications

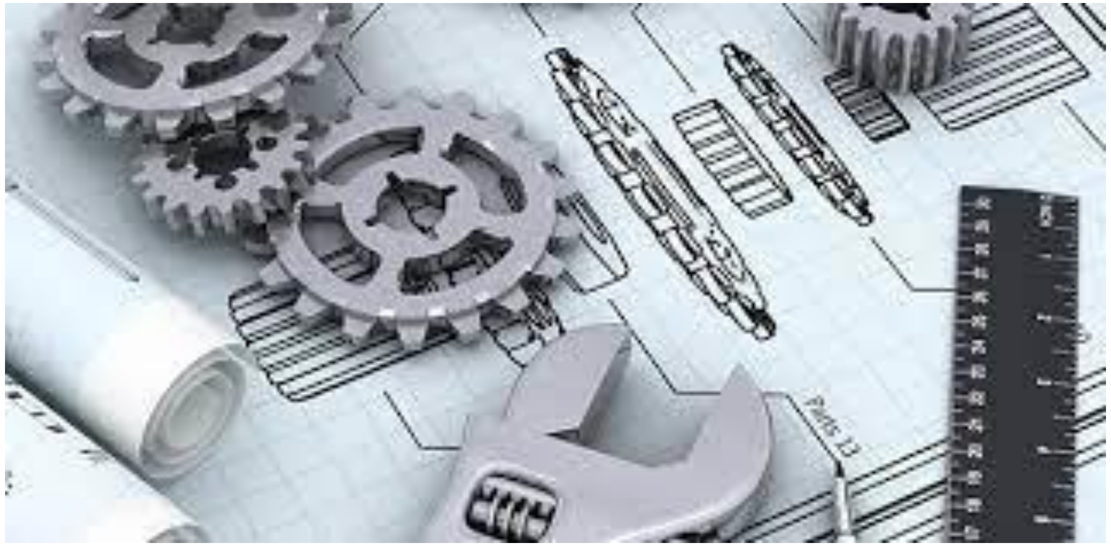
Industry

- autonoumous driving / robotics
- packing & transport
- design

Applications

Industry

- autonoumous driving / robotics
- packing & transport
- design



Applications

Intertainment

- animation
- games
- movie

Applications

Intertainment

- animation
- games
- movie

Applications

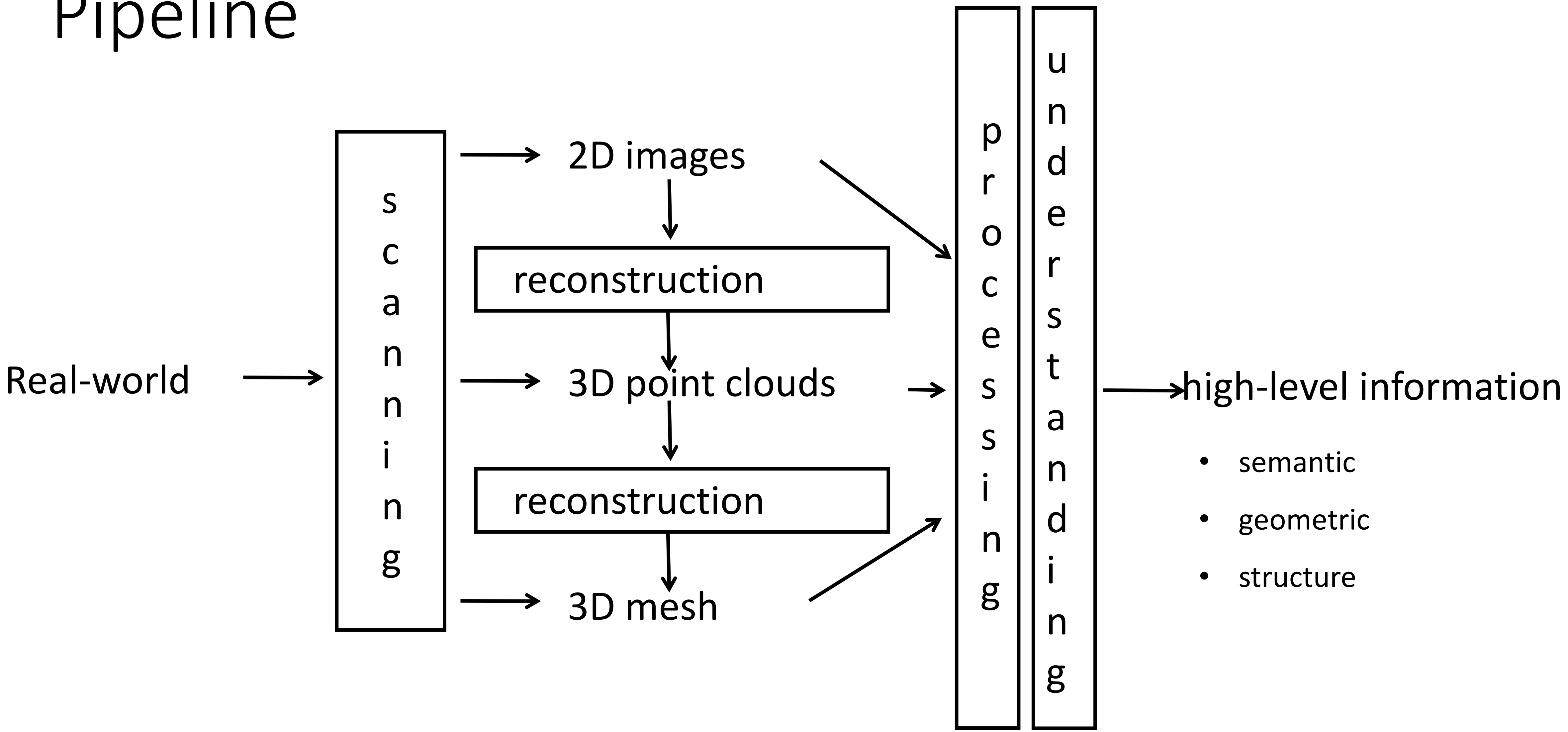
Intertainment

- animation
- games
- movie

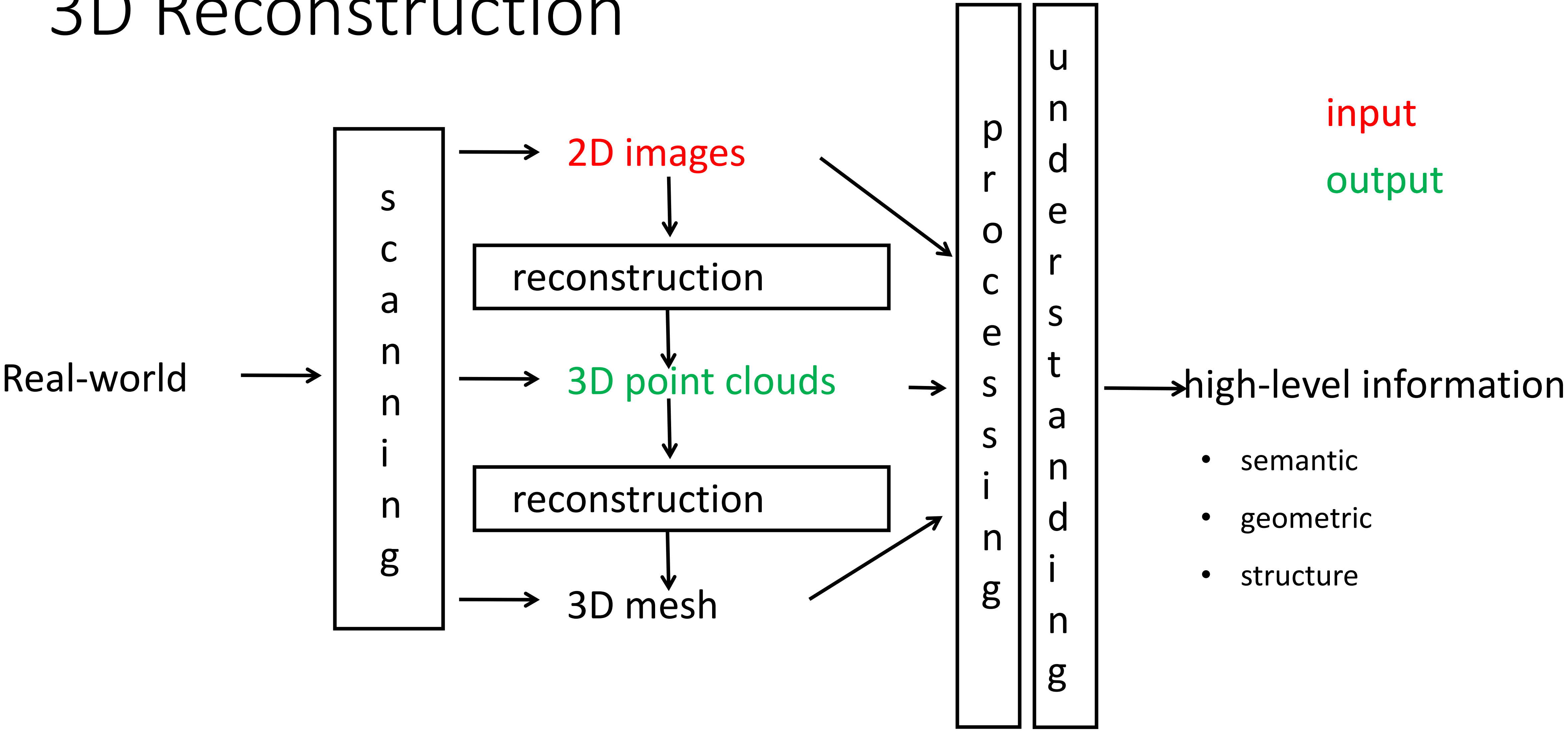
Part 2

3D Vision Pipeline

Pipeline


Real-world
scanning
2D images
reconstruction
3D point clouds
reconstruction
3D mesh
processing
understanding
high-level information
semantic
geometric
structure

3D Reconstruction

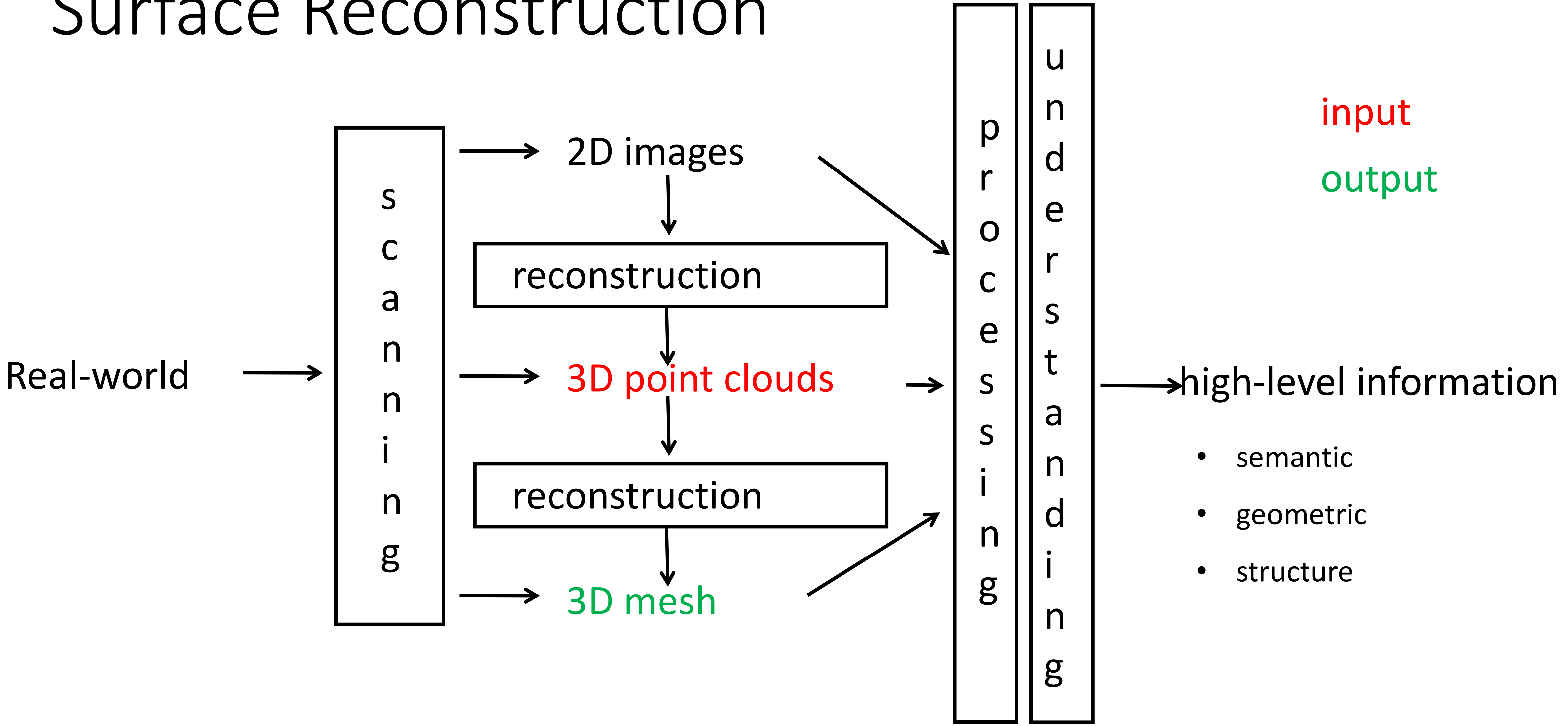


3D Reconstruction

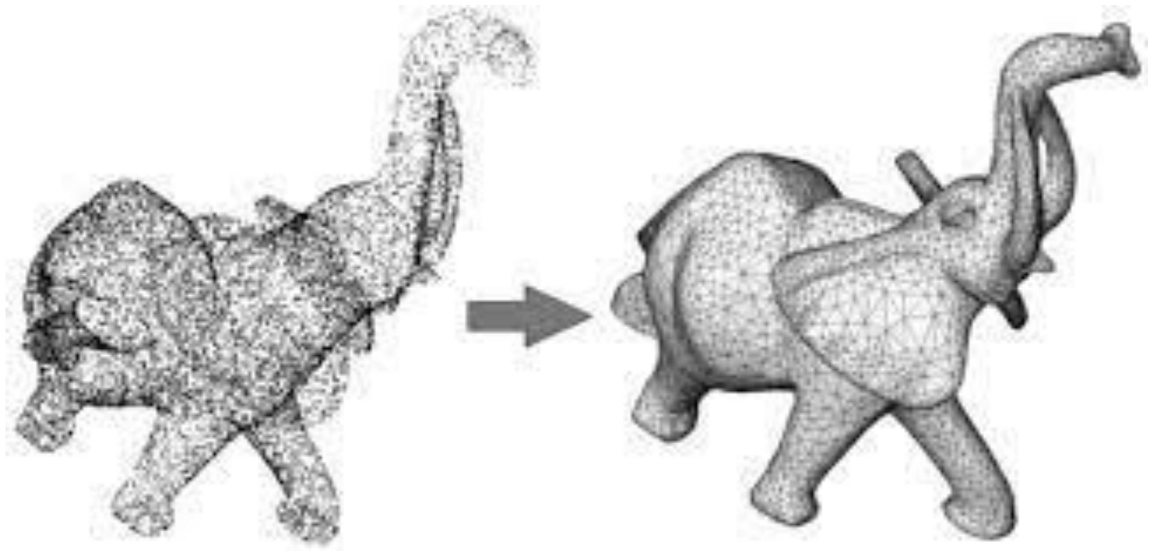
input: 2D images

output: 3D point clouds

Surface Reconstruction



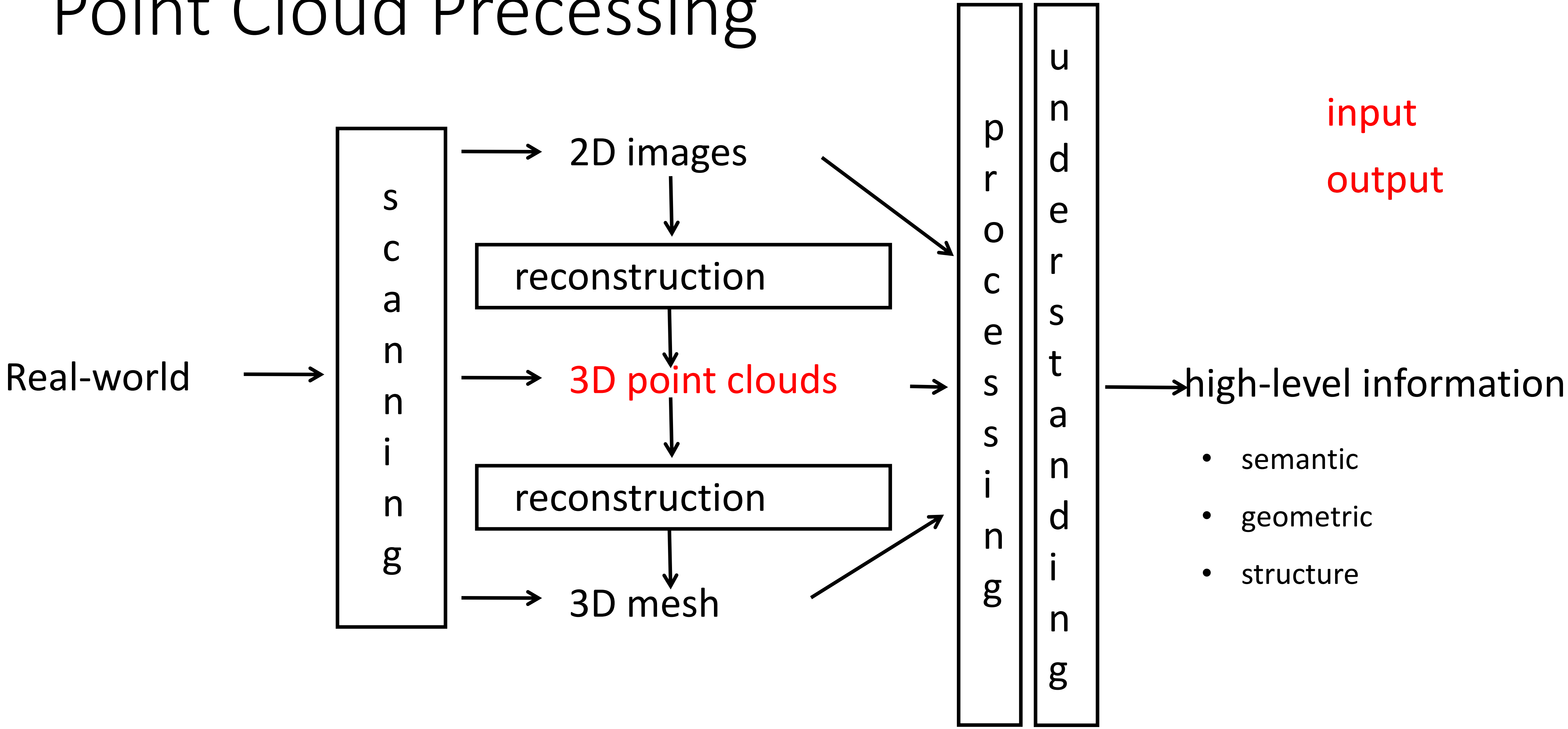
Surface Reconstruction



input: 3D point clouds

output: 3D mesh

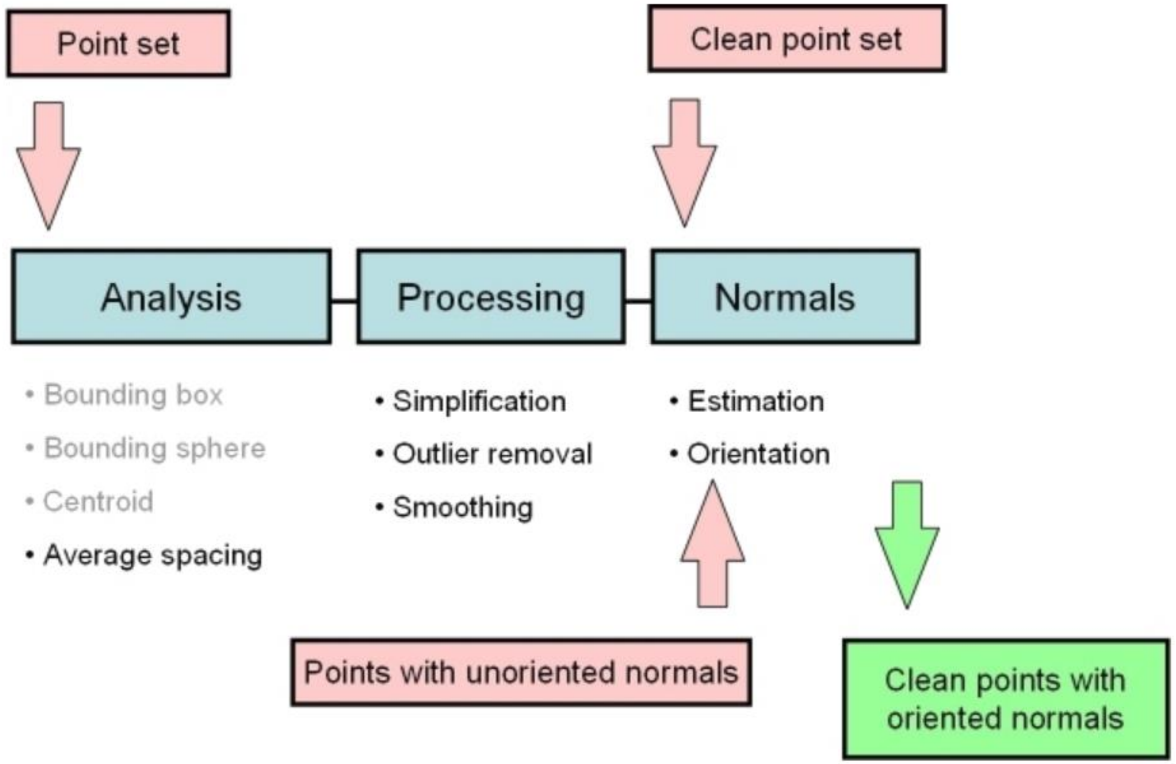
Point Cloud Precessing



input

output

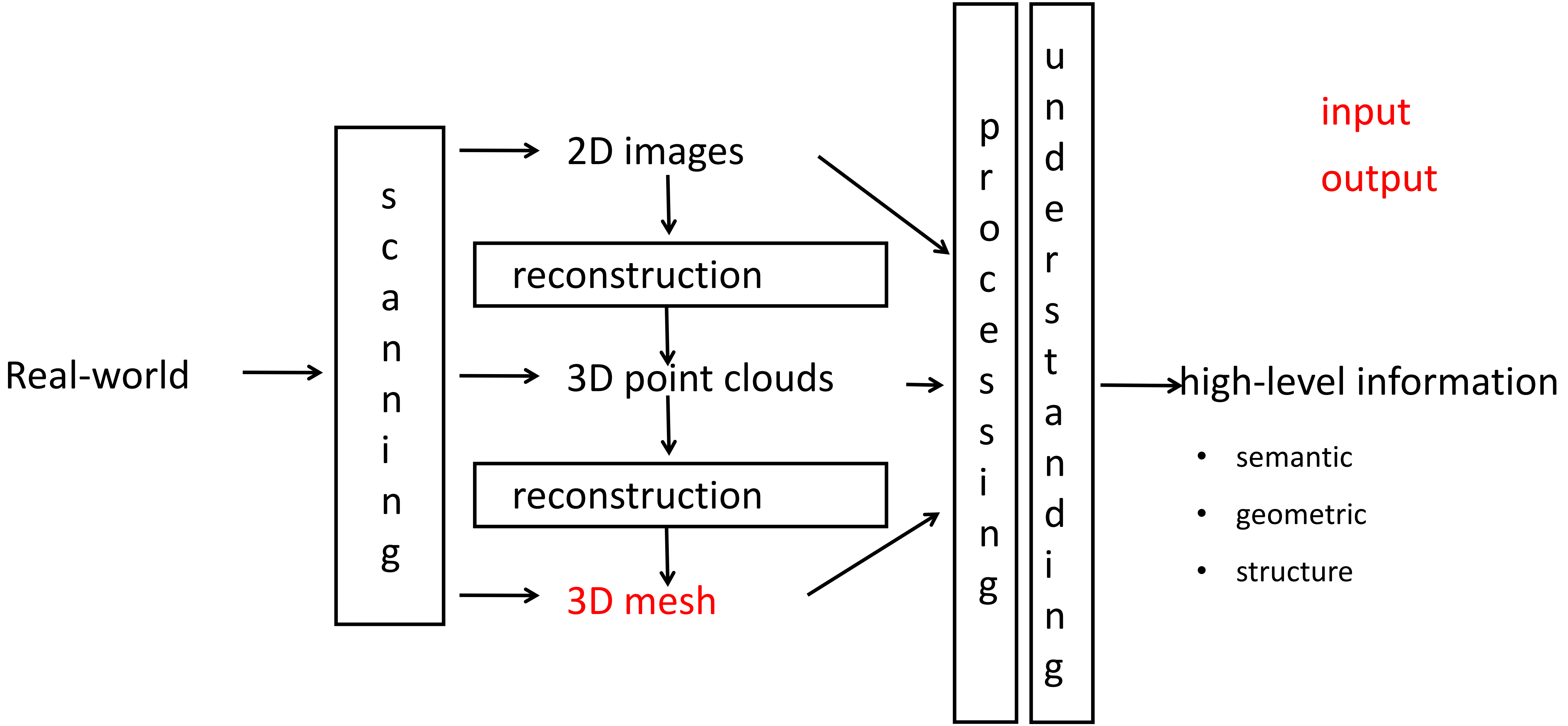
Point Cloud Precessing



input: raw point clouds

output: clean point clouds with normals

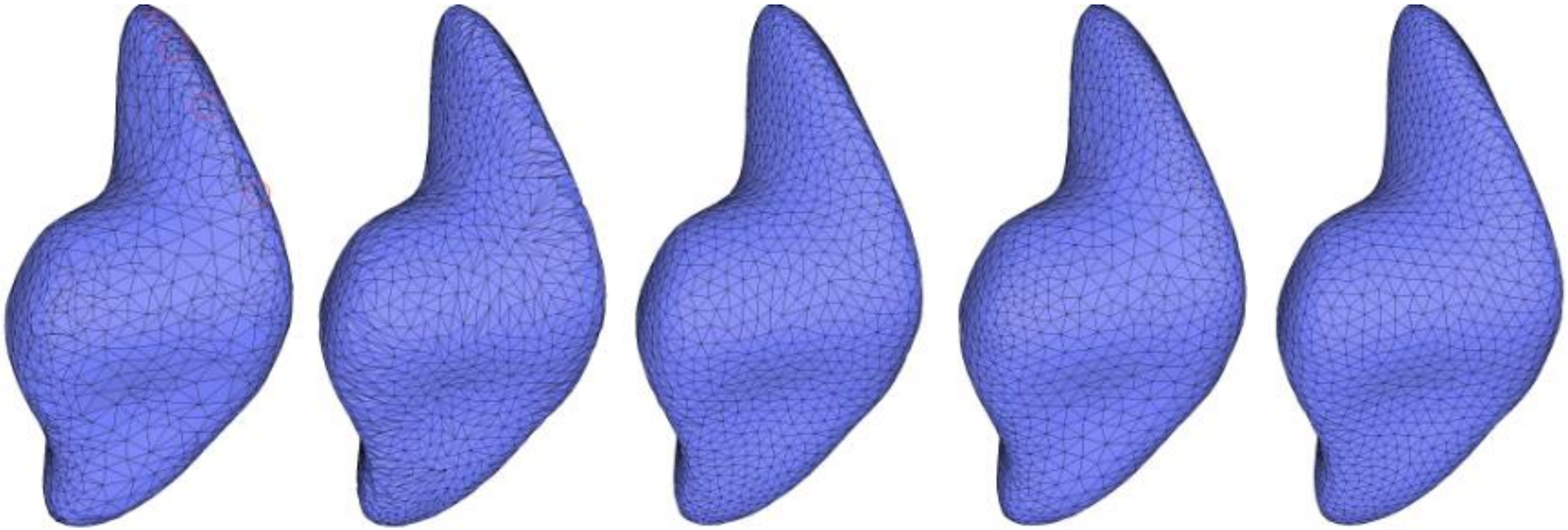
Polygonal Mesh Precessing



- semantic
- geometric
- structure

Polygonal Mesh Precessing

Smoothing



input: raw triangular mesh

output: smooth mesh

Polygonal Mesh Precessing

Hole filling

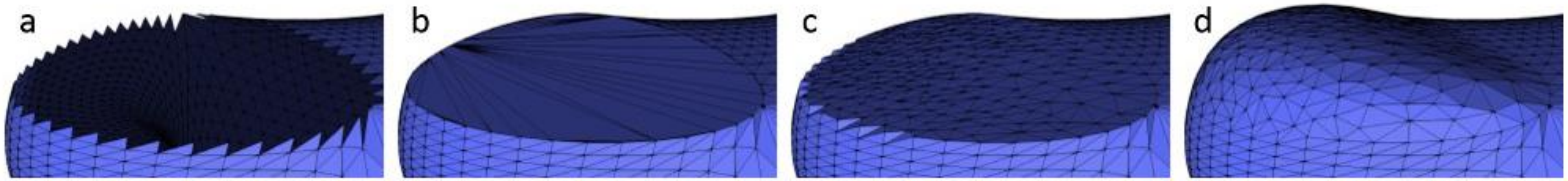


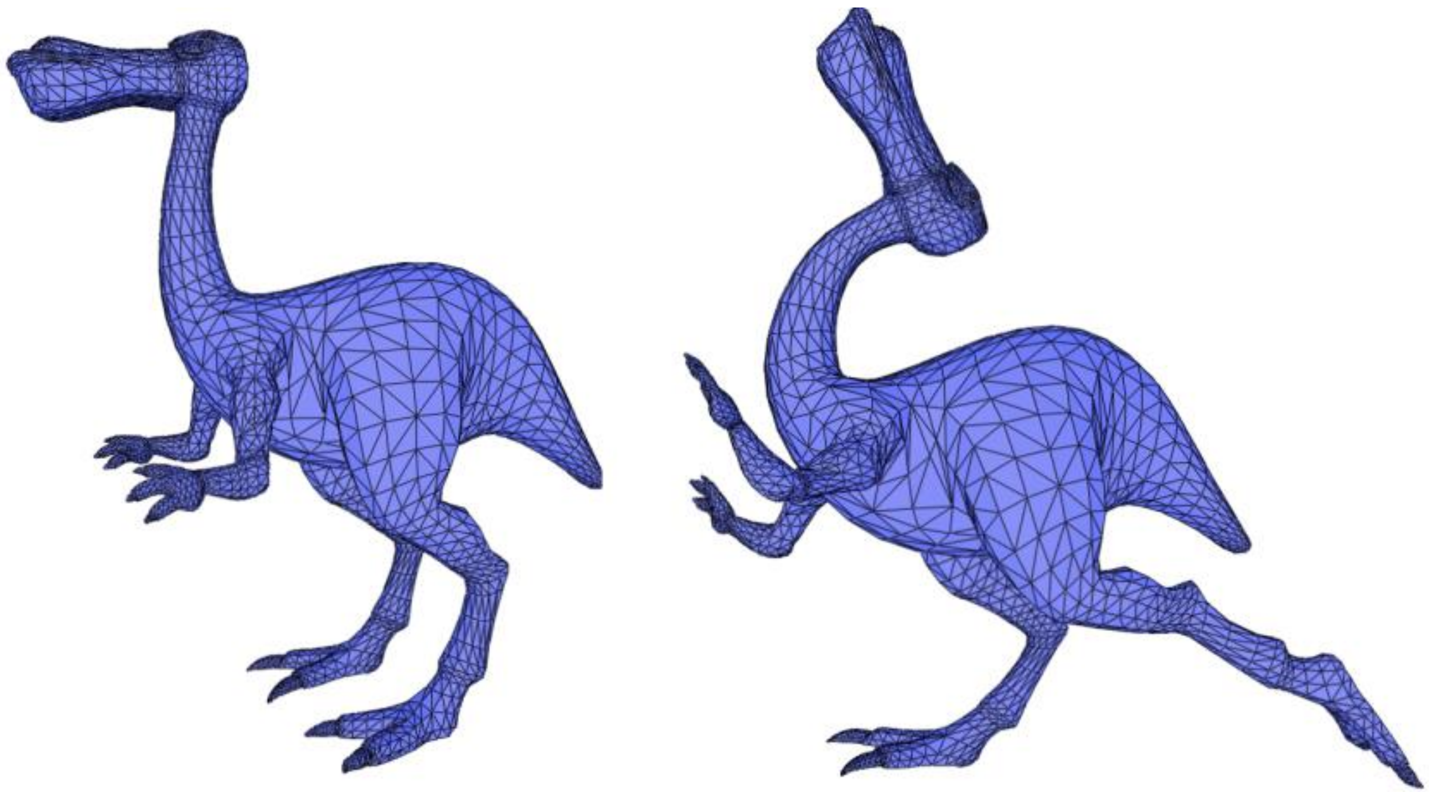
Figure 64.10 Results of the main steps of the algorithm. From left to right: (a) the hole, (b) the hole after its triangulation, (c) after triangulation and refinement, (d) after triangulation, refinement and fairing.

input: raw triangular mesh with holes

output: watertight mesh

Polygonal Mesh Precessing

Deformation

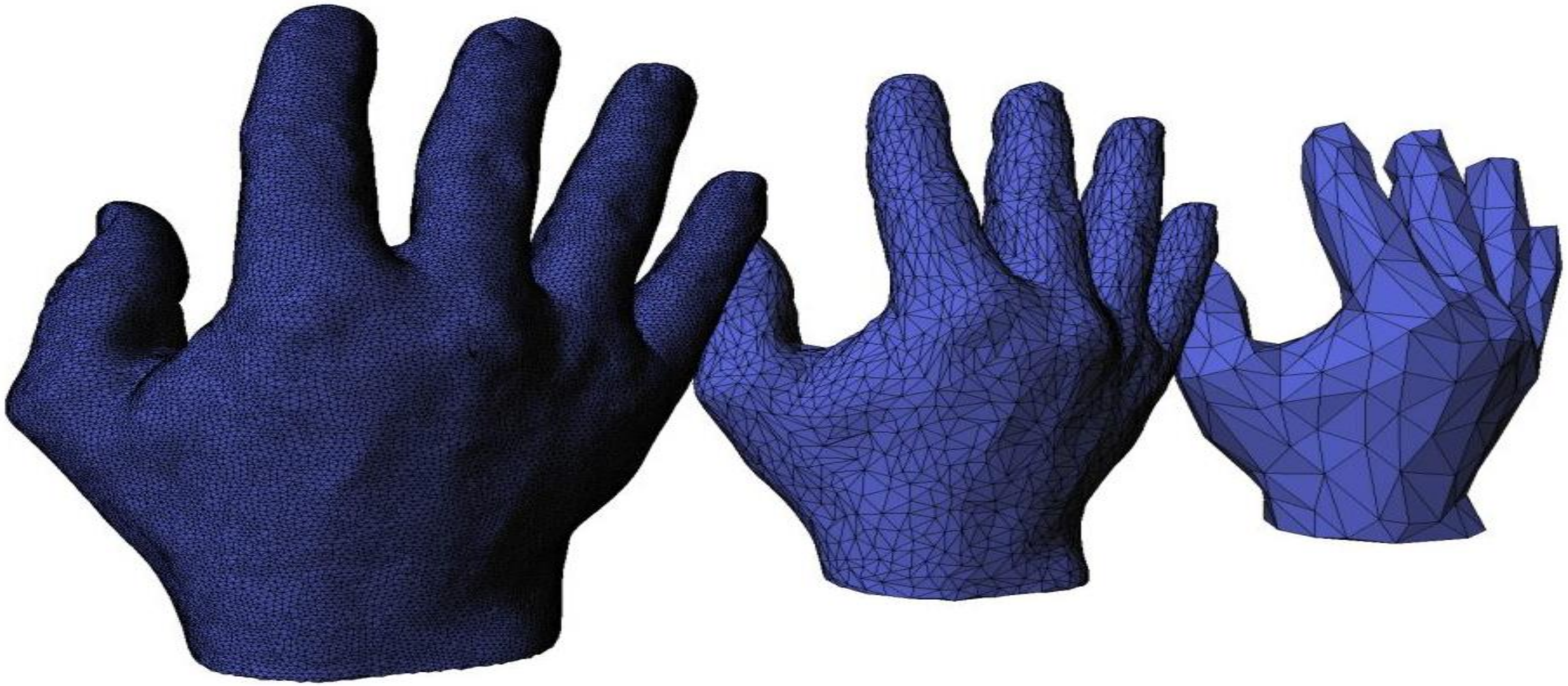


input: raw triangular mesh

output: defromed mesh

Polygonal Mesh Precessing

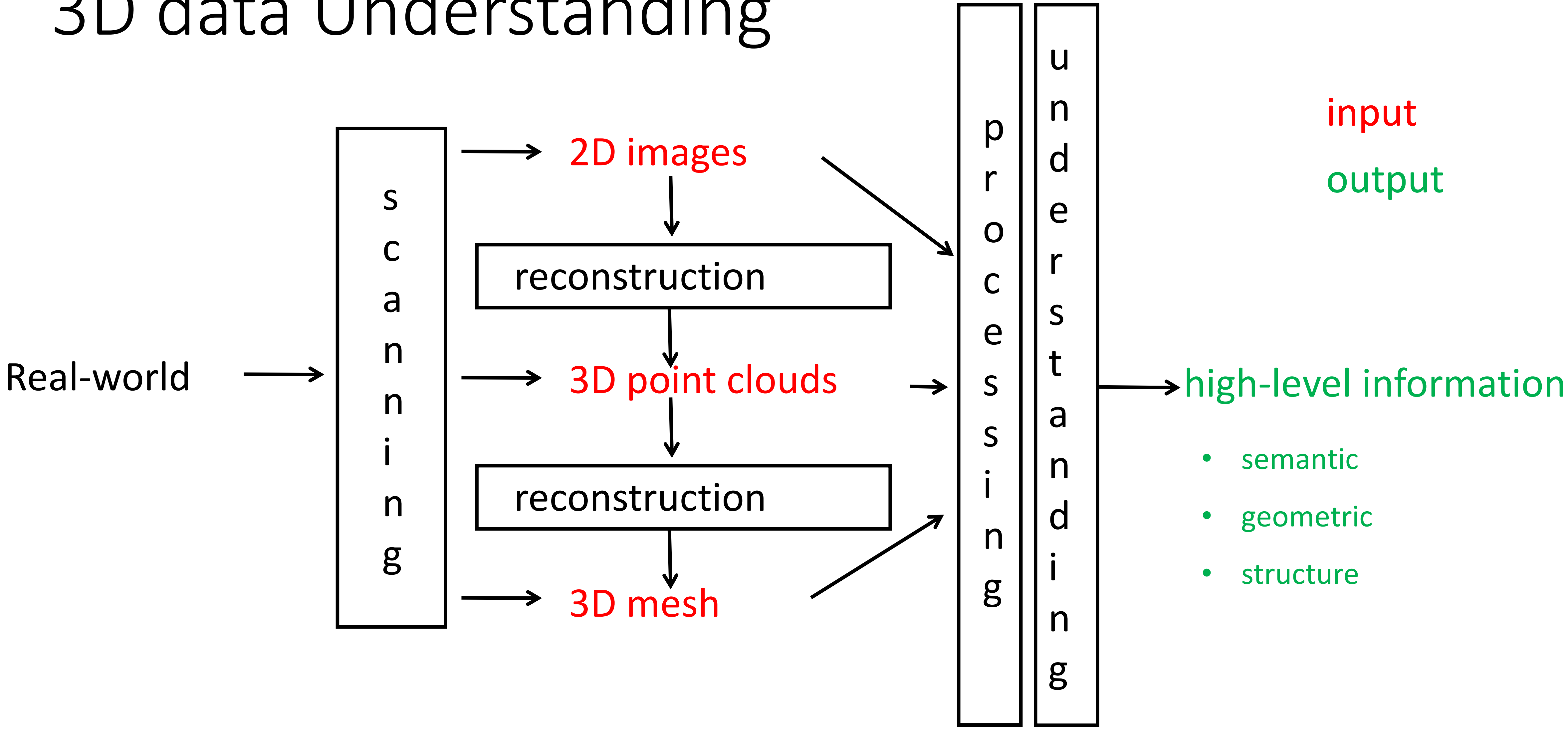
Simplifiaction



input: raw triangular mesh

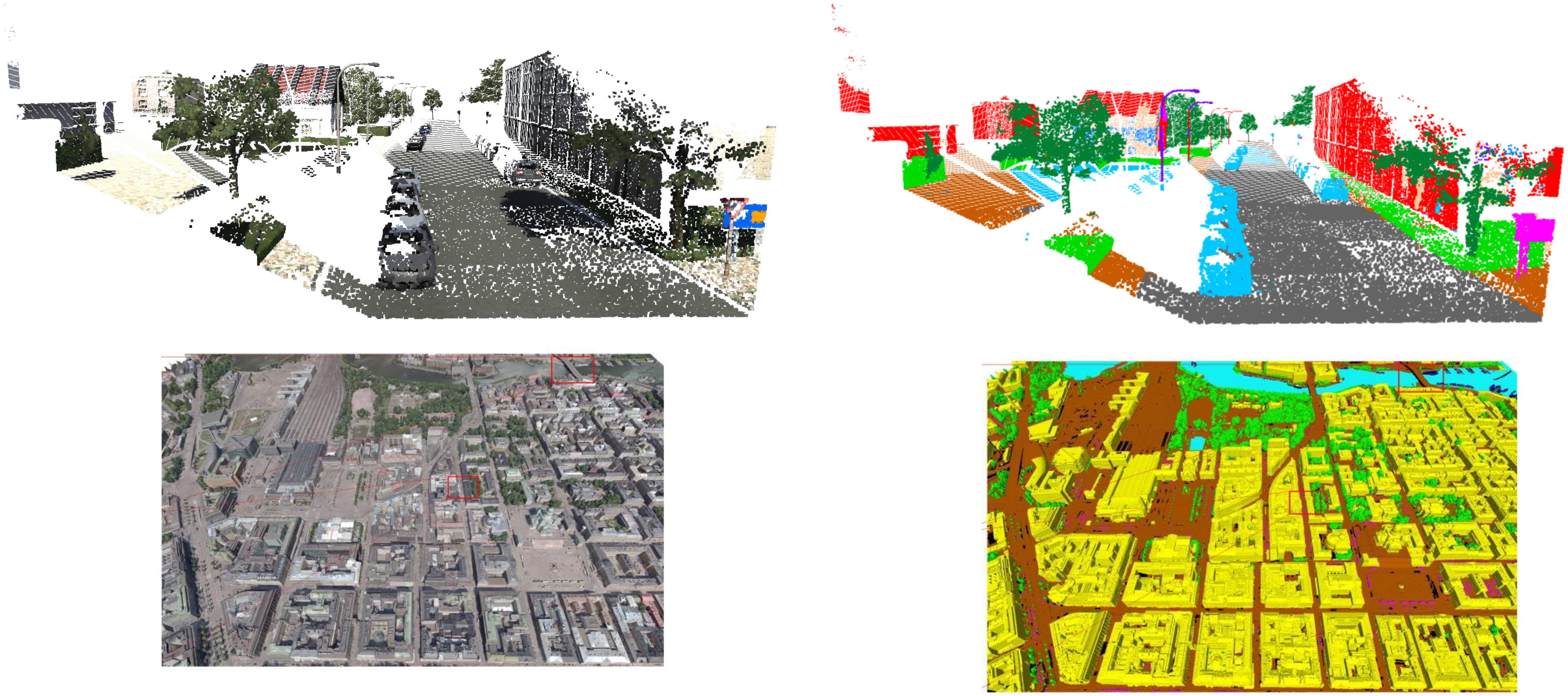
output: simplified mesh

3D data Understanding



3D data Understanding

3D semantic segmentation

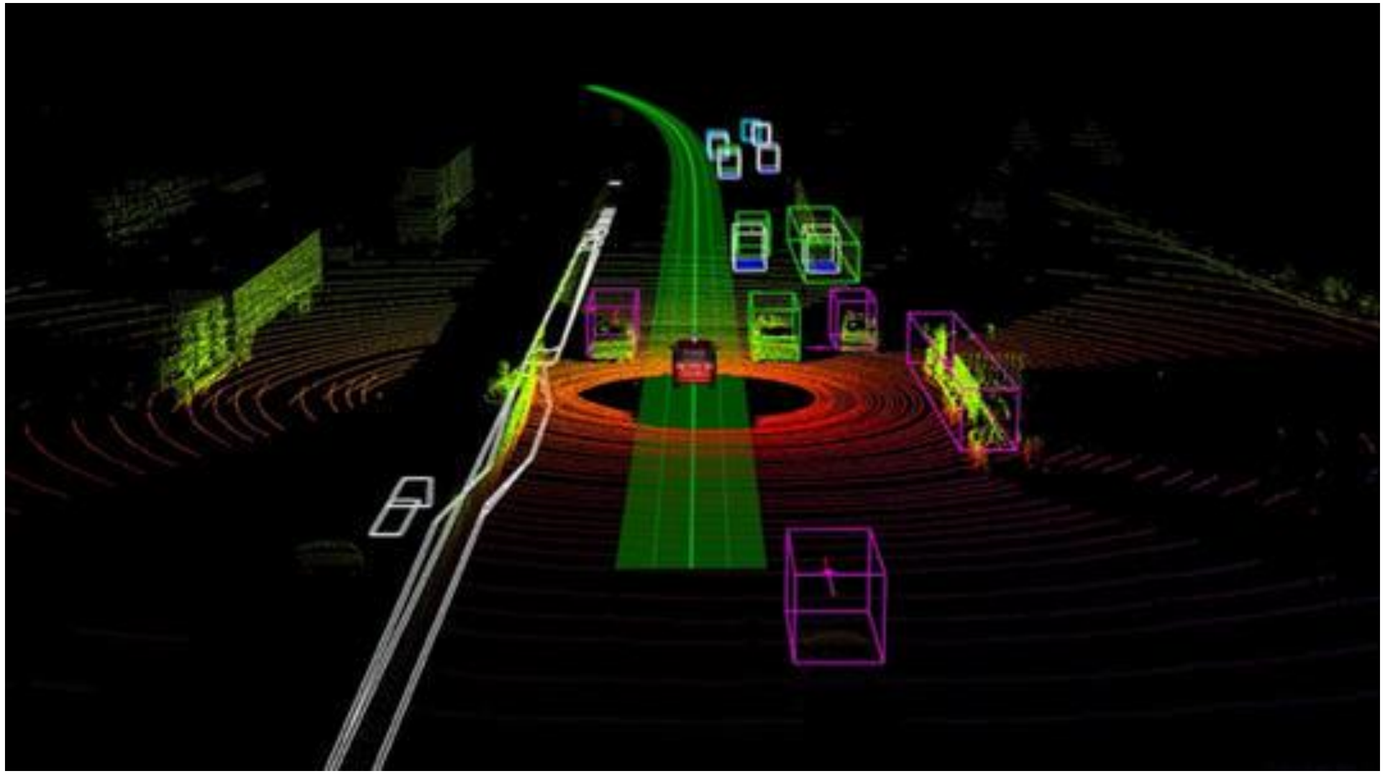


raw point cloud or triangular mesh

semantic label

3D data Understanding

3D object detection

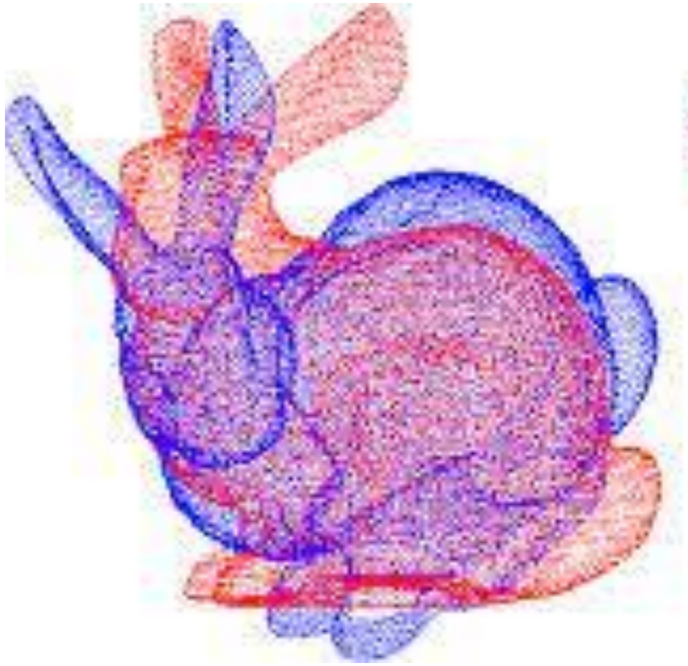


input: point clouds

output: 3D bounding box

3D data Understanding

3D registration

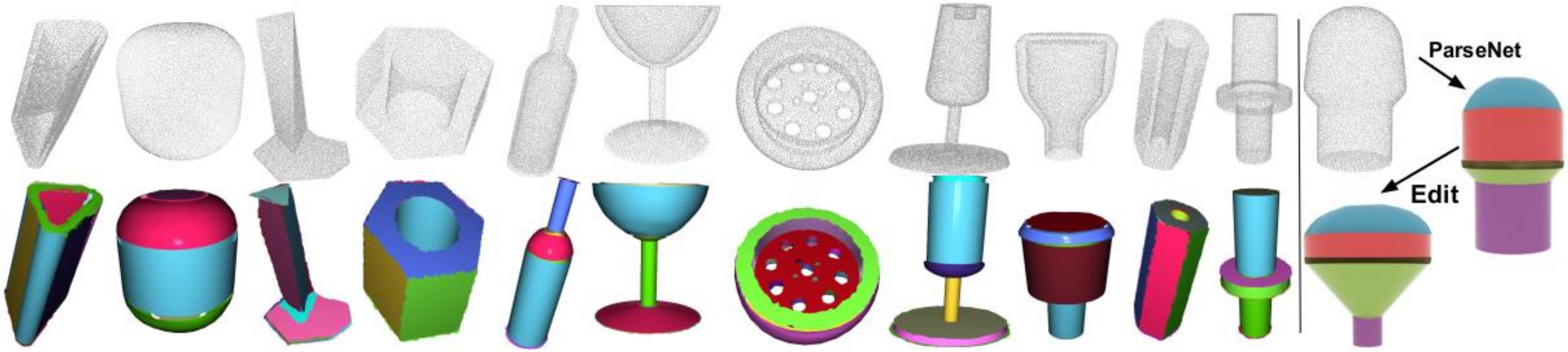


input: two point clouds

output: transformation matrix

3D data Understanding

3D shape detection



input: point clouds or mesh

output: geoemtric shapes

3D data Understanding

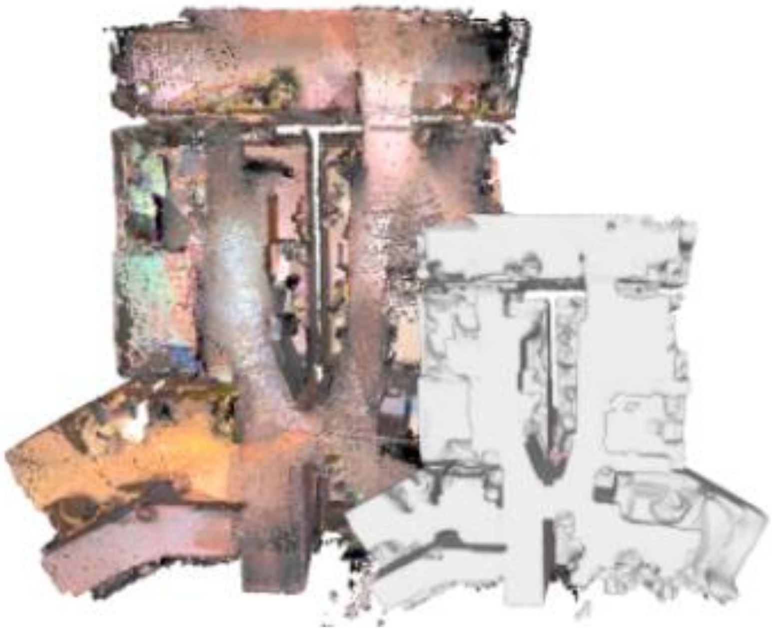
scene / model completion

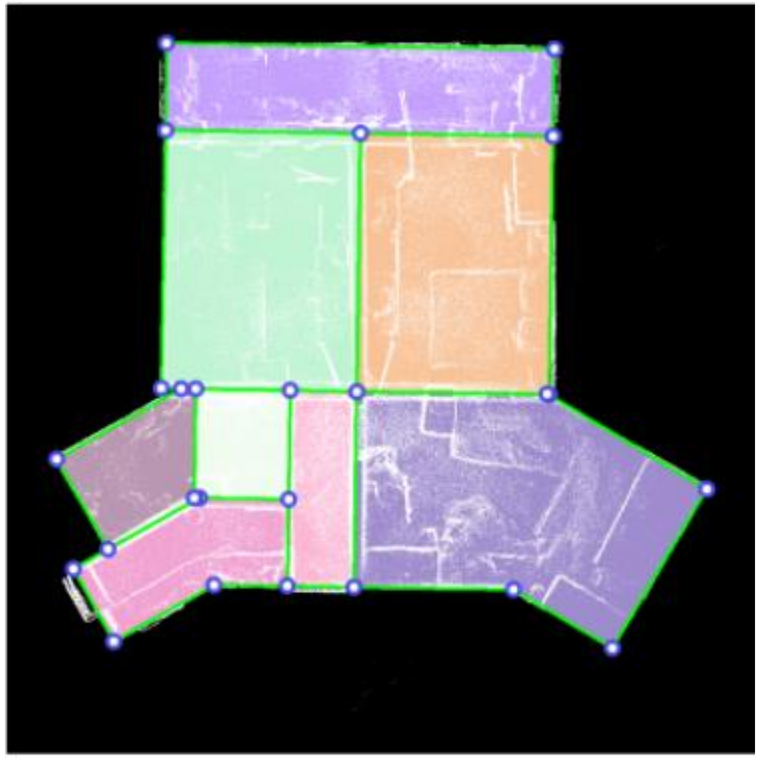
input: partial 3D data

output: complete 3D data

3D data Understanding

Floorplan generation



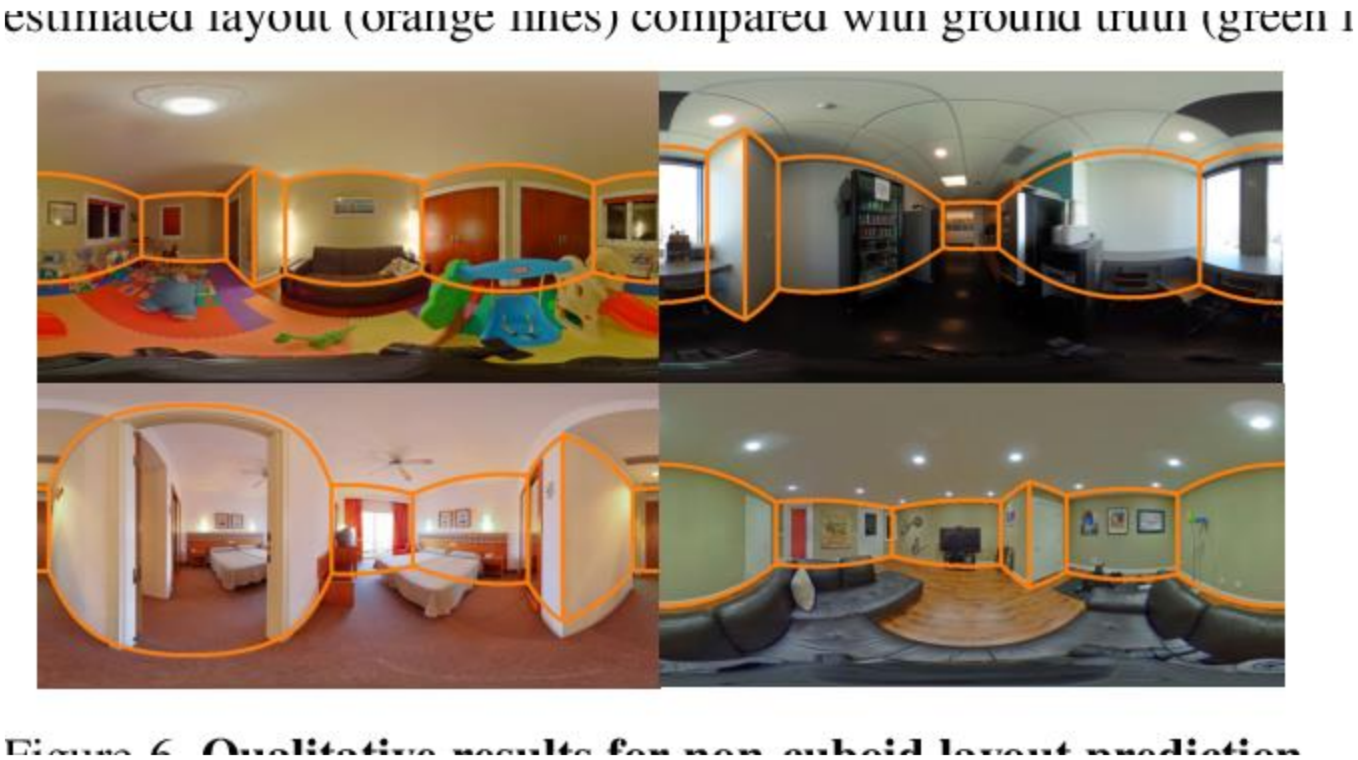


input: RGB images

output: 2D planar graph

3D data Understanding

Layout prediction



input: panoramic image

output: scene layout

3D data Understanding

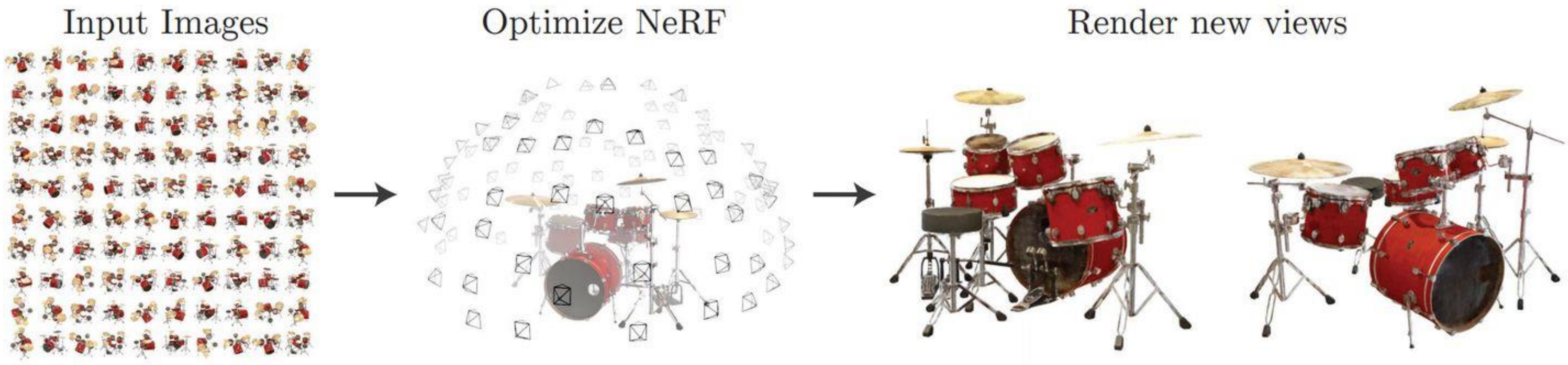
Depth prediction

input: video images

output: depth map

3D data Understanding

View Synthesis



input: RGB images

output: radiance at (x,y,z) from view direction (θ, φ)

Part 3

Important Tools

CGAL

Computational Geometry Algorithms Library

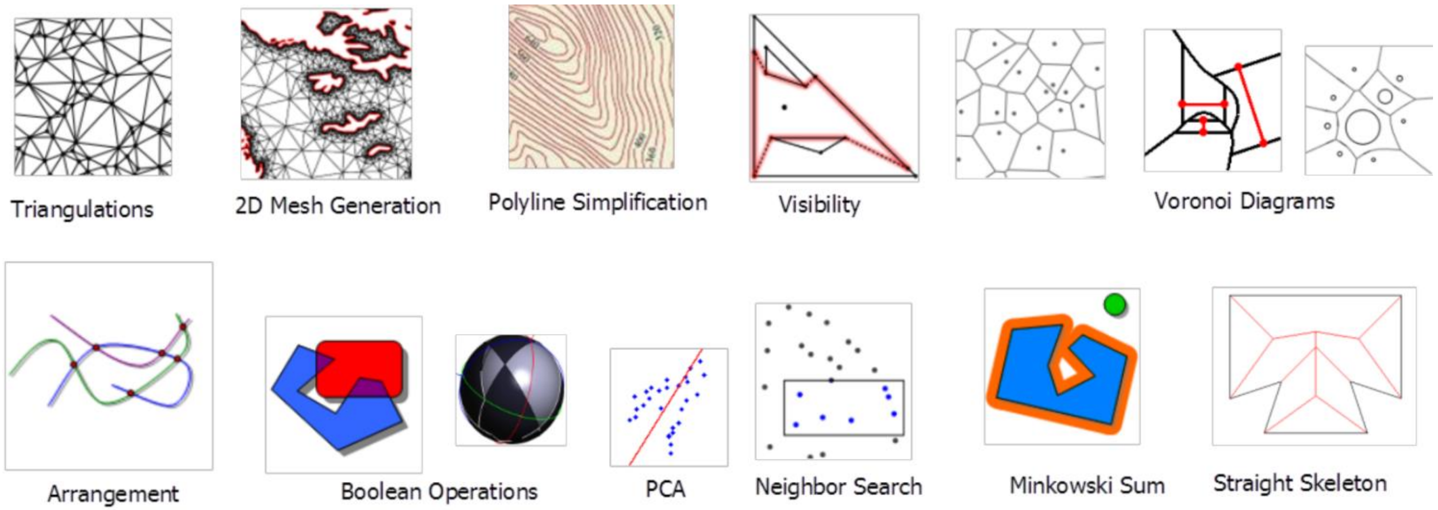
- **基本的几何数据**结构：点，线，面，向量，多边形，多面体，点云，**mesh等**
- **基本的几何元素之**间关系的算法：查询是否相交，计算距离
- **基本的点云，mesh数据**处理算法
- **基本的**优化算法

CGAL

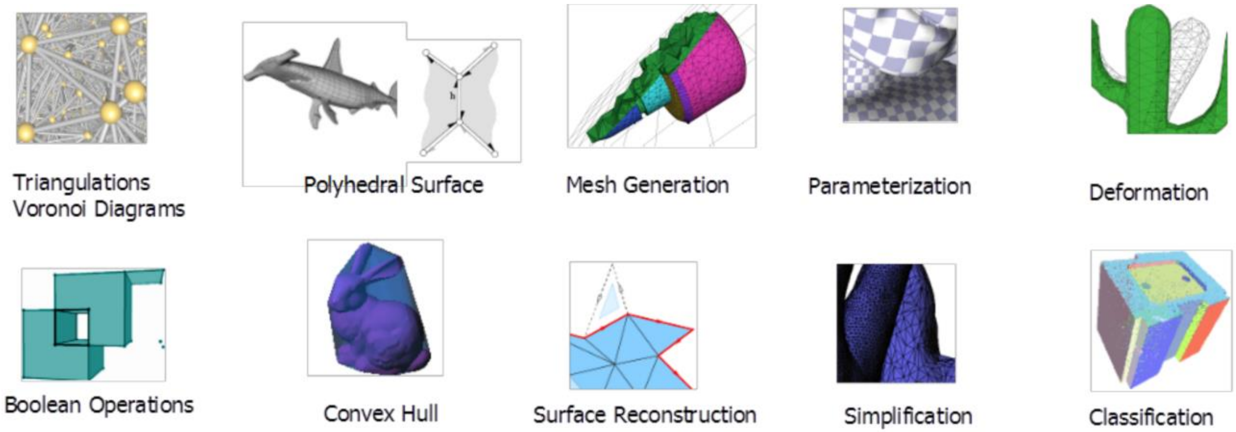
- **可移植性高**
- generic programming：template， STL， boost
- 处理浮点数计算的鲁棒性高
- **准确性高**
- **效率高**

CGAL

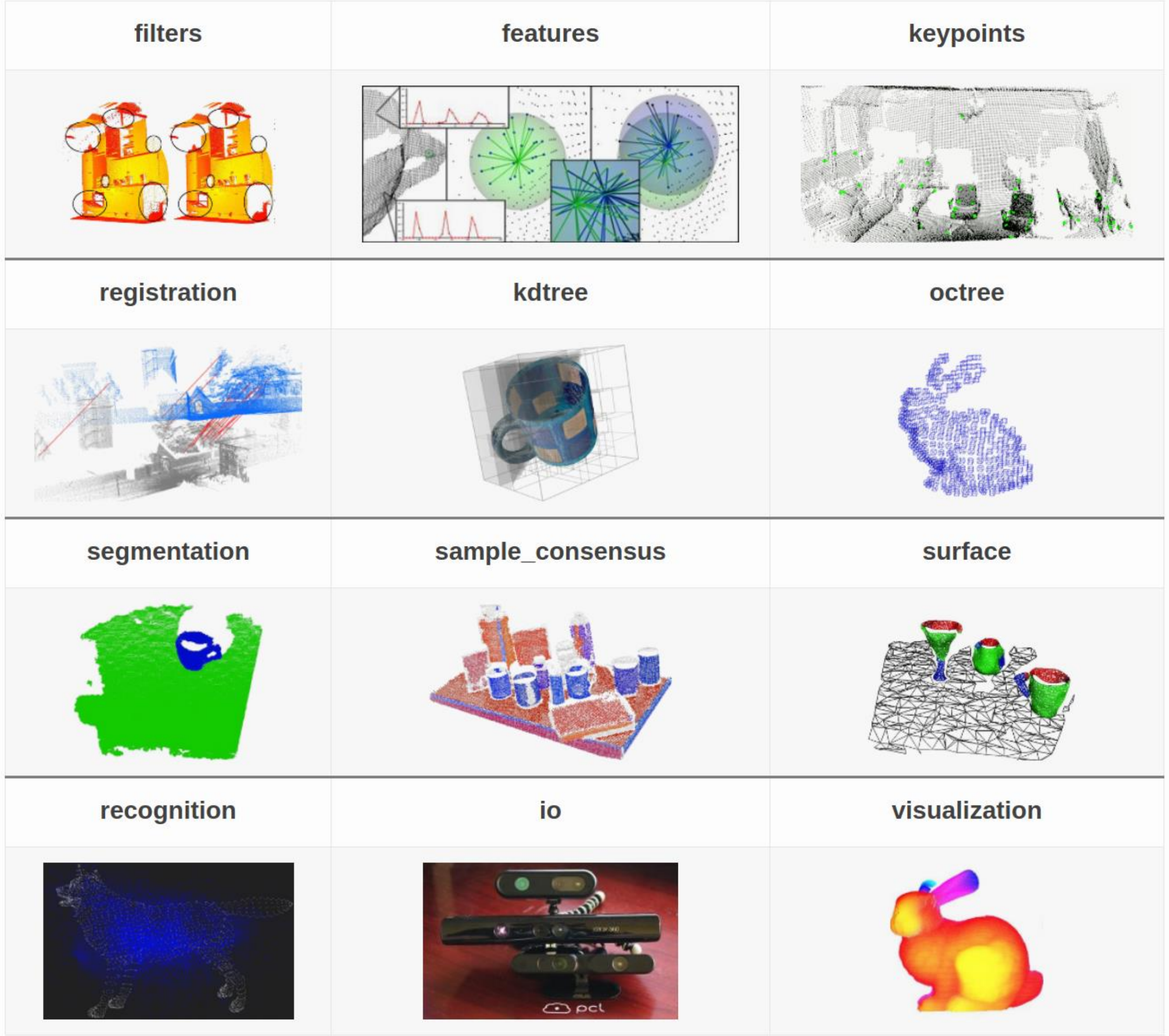
- **丰富的几何相关数据**结构和算法：
  - 2D：



  - 3D：



PCL



pointcloudlibrary

PyTorch

- Dataset and dataloader are easy to use
- Rich deep learning operators (differential->end-to-end), i.e. CNN, FC, dropout, cross entropy loss, L2 loss...
- Famous network architectures, i.e. ResNet50, LSTM
- C++\CUDA based operators extentions

Paper & Code & Data

- Structured3D, S3DIS, ScanNet, KITTI...
- arXiv: https://arxiv.org/
- Conference: CVPR, ICCV, ECCV, SIGGRAPH， SIGGRAPH Asia
- Journal: PAMI, IJCV, ISPRS Journal, TOG, TVCG
- Webpage of famous researchers or laboratories
- Github: realeased code

3D Vision Lab and Researchers

- The Stanford Geometric Computation Group： https://geometry.stanford.edu/
- Facebook AI Research: https://ai.facebook.com/research/
- Hao Su (UCSD): https://cseweb.ucsd.edu/~haosu/
- Computer Vision and Geometry lab (CVG) , ETH, http://www.cvg.ethz.ch/index.php
- Angela Dai: https://www.3dunderstanding.org/
- Andreas Geiger: http://www.cvlibs.net/
- Vladlen Koltun: http://vladlen.info/
- Thomas A. Funkhouser: https://www.cs.princeton.edu/~funk/
- Matthias Nießner: https://niessnerlab.org/publications.html

3D Vision Lab and Researchers

- Visual Computing Research Center (VCC) , SZU: http://vcc.szu.edu.cn/index.html
- Kun Zhou: http://kunzhou.net/
- Ligang Liu: http://staff.ustc.edu.cn/~lgliu/
- Baoquan Chen: http://cfcs.pku.edu.cn/baoquan/
- Kai Xu: https://kevinkaixu.net/
- Xiaowei Zhou: http://xzhou.me/
- Jingyi Yu: http://www.yu-jingyi.com/cv/

Part 4

Homework

Assignment 1

- 0. Learn CMake
- 1. Install PCL
- 2. Install CGAL
- 3. Compile two projects on BB and run the test code
- 4. Upload the results to BB

Questions?