

GPU TECHNOLOGY
CONFERENCE

SO088 Point Cloud Library (**PCL**) on **CUDA**

Radu B. Rusu @ Open Perception
Michael Dixon @ Willow Garage

What is PCL?

[Learn more](#)



Point cloud basics



$$\begin{aligned} & \text{It is easy to show for } N=2 \text{ that} \\ & \int_{\Omega} \left(2 \frac{\partial^2 u}{\partial x_1 \partial x_2} \frac{\partial^2 v}{\partial x_1^2} - \frac{\partial^2 u}{\partial x_1^2} \frac{\partial^2 v}{\partial x_2^2} - \frac{\partial^2 u}{\partial x_1^2} \frac{\partial^2 v}{\partial x_2^2} \right) dx_1 dx_2 \\ & = \int_{\Gamma} \left(-\frac{\partial u}{\partial \vec{n}} \frac{\partial v}{\partial \vec{n}} + \frac{\partial u}{\partial \vec{n}} \frac{\partial v}{\partial \vec{n}} \right) ds, \quad u \in H^3(\Omega), v \in H^2(\Omega). \quad (1.2.10) \end{aligned}$$

Let $\tau = (\tau_1, \tau_2)$ be the unit tangent vector along Γ , $\frac{d}{dt}$ the derivative along the tangent direction, and

$$\frac{\partial^2 \pi_i}{\partial \tau^2} = D^2 u \cdot (\tau, \tau) = \sum_{i,j=1}^2 \pi_i \tau_j \frac{\partial^2 u}{\partial x_i \partial x_j},$$

$$\frac{\partial^2 u}{\partial x_i \partial x_j} = D^2 u \cdot (\tau_i, \tau_j) = \sum_{k=1}^2 \tau_i \tau_j \frac{\partial^2 u_k}{\partial x_i \partial x_j},$$

Surface reconstruction

```

 $\hat{b}_{\min}^i, \hat{b}_{\max}^i$  // minimum and maximum bounds for i-th dimension
 $\mathcal{P} = \{p_1, \dots, p_n\}$  // set of all 3D points
 $\mathcal{P}_0 = \{p_i, b_{\min}^i \leq p_i^i \leq b_{\max}^i\}$  // subset of all  $p_i$  in  $\mathcal{P}$ 
estimate( $\vec{n}_i$  from  $\mathcal{P}_0^k$ ) // estimate surface normal
if ( $\mu = \vec{n}_i \times \vec{Z} \approx 0$ ) // check if the
     $\mathcal{P}_z \leftarrow p_i$  // add  $p_i$  to  $\mathcal{P}_z$ 
estimate( $\mathcal{C} = \{p_1^i, \dots, p_n^i\}$ ,  $\mathcal{P}_z^i \subset \mathcal{P}_z$ ) // find the best plane fit using sampling
for all  $c_i = \mathcal{P}_z^i \in \mathcal{C}$  // find the best plane fit using sampling
    estimate( $\{a, b, c, d\}$ ,  $a \cdot p_i^i + b \cdot p_i^j + c \cdot p_i^k + d = 0$ )
    estimate( $b_{\min}, b_{\max}$ ) // find the min and max box
     $\mathcal{M} = F(c_i)$  // add a new table parameter
for all  $p_i \in \mathcal{P}$ 
    if ( $p_i^x \leq \hat{b}_{\min}^i, p_i^y \leq \hat{b}_{\min}^j, p_i^z \leq \hat{b}_{\min}^k \leq p_i^x \leq \hat{b}_{\max}^i, p_i^y \leq \hat{b}_{\max}^j, p_i^z \leq \hat{b}_{\max}^k$ )
         $\mathcal{P}_0 \leftarrow p_i$  // add  $p_i$  to the  $\mathcal{P}_0$  set
estimate( $\mathcal{O} = \{\mathcal{P}_0^1, \dots, \mathcal{P}_0^n\}$ ,  $\mathcal{P}_0^i \subset \mathcal{P}_0$ )
for all  $o_i \in \mathcal{O}$ 
     $\mathcal{M} = F(o_i)$  // add the object parameter

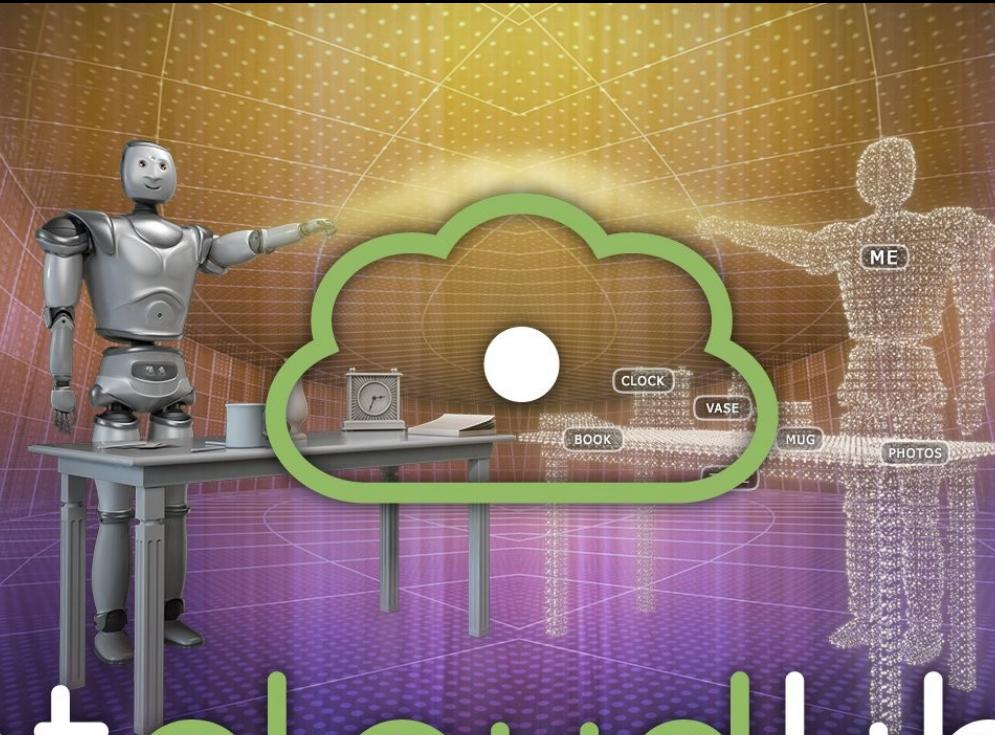
```



- General overview on PCL - Radu
 - CUDA optimizations for KinFu and KinSkel - Michael



pointcloudlibrary



Point Cloud Library (PCL)

- large scale, collaborative, open project for 2D/3D processing
- BSD licensing, free for commercial use
- 350 developers and contributors, many thousands of users



why 3D?

Why 3D?

- Because the world is not 2D



Why 3D?

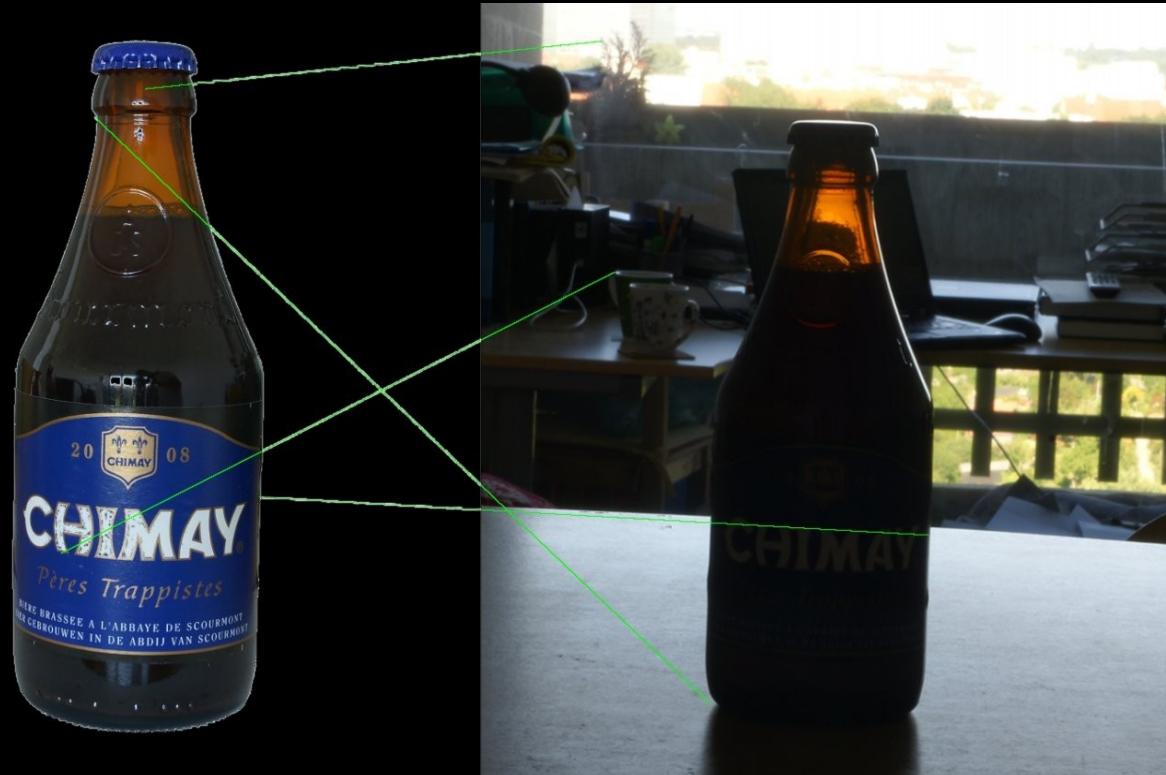
- Because the world is not 2D



© Rex

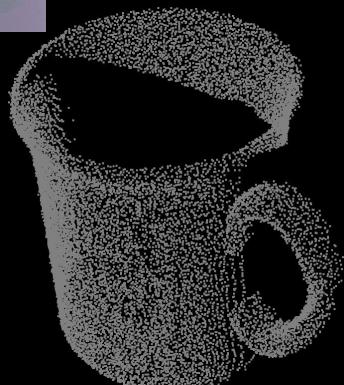
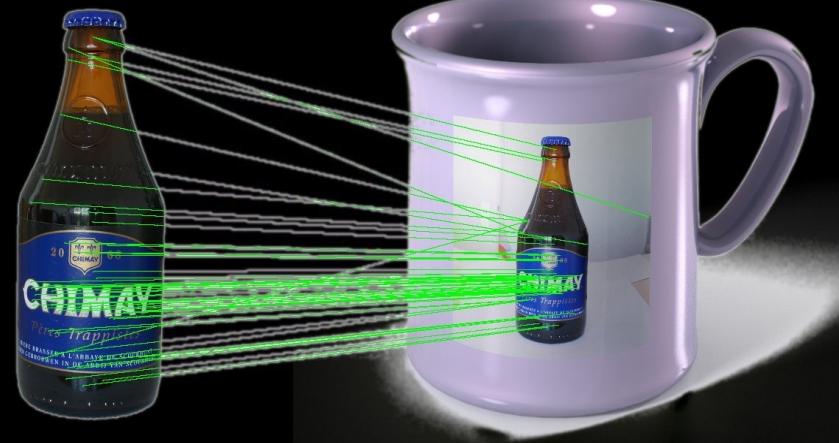
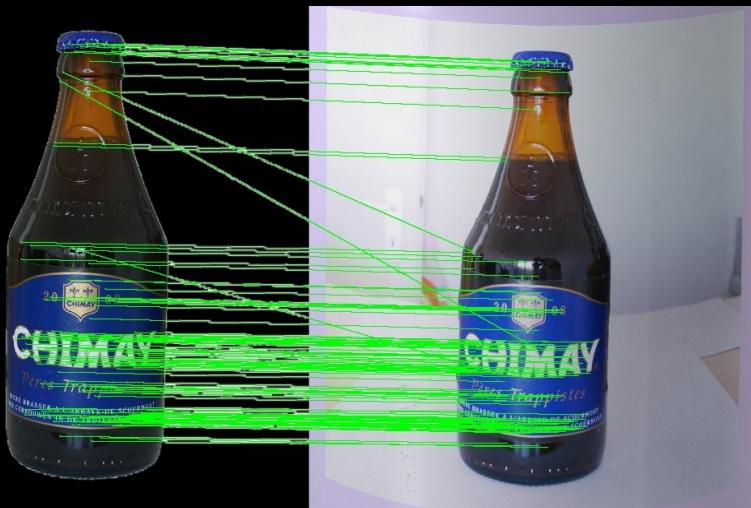
Why 3D?

- Because 2D imagery is **useless** in certain conditions



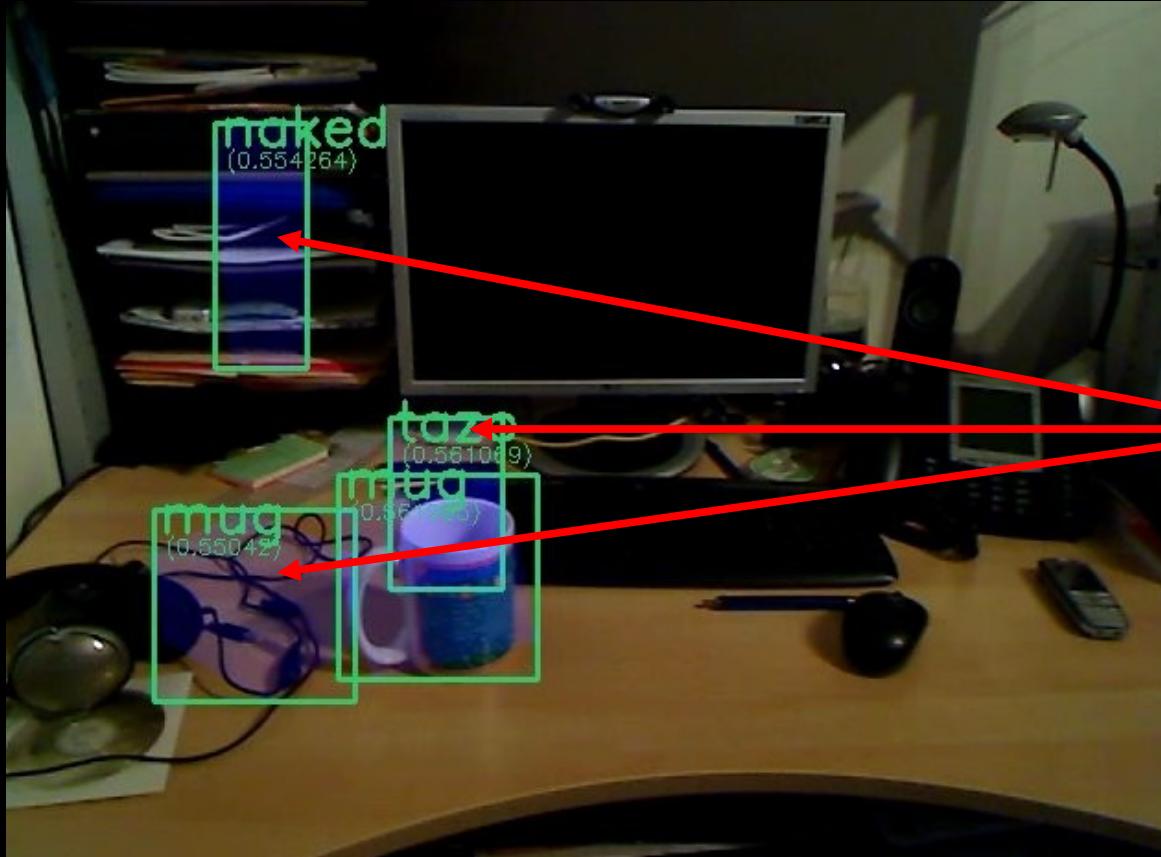
Why 3D?

- Because 2D imagery doesn't always infer good semantics



Why 3D?

- Because 2D imagery is just a ... 2D projection



2D matching failures

3D

50% better than 2D

Three-dimensional data

- Point Cloud = collection of 3D points
 - 3D is really more like nD



What can we do with Point Clouds?

Applications of Point Clouds

- Point clouds enable very cool applications
- We are trying to solve extremely hard problems



Applications of Point Clouds

- Point clouds enable very cool applications
- We are trying to solve extremely hard problems



Applications of Point Clouds

- Point clouds enable very cool applications
- We are trying to solve extremely hard problems



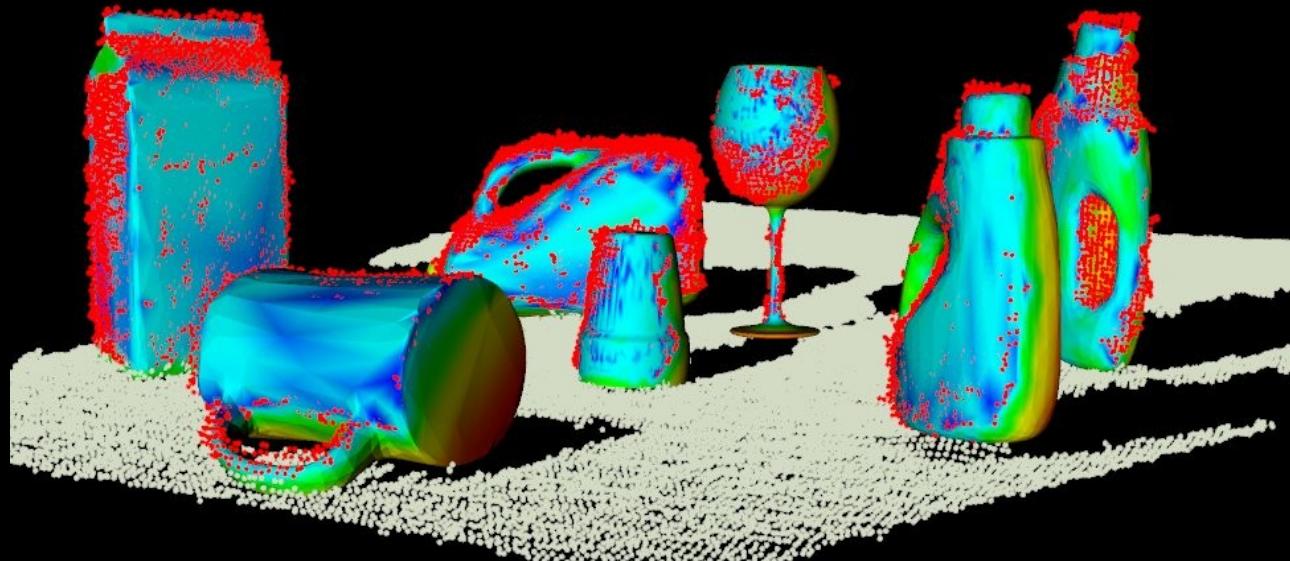
Applications of Point Clouds

- Point clouds enable very cool applications
- We are trying to solve extremely hard problems



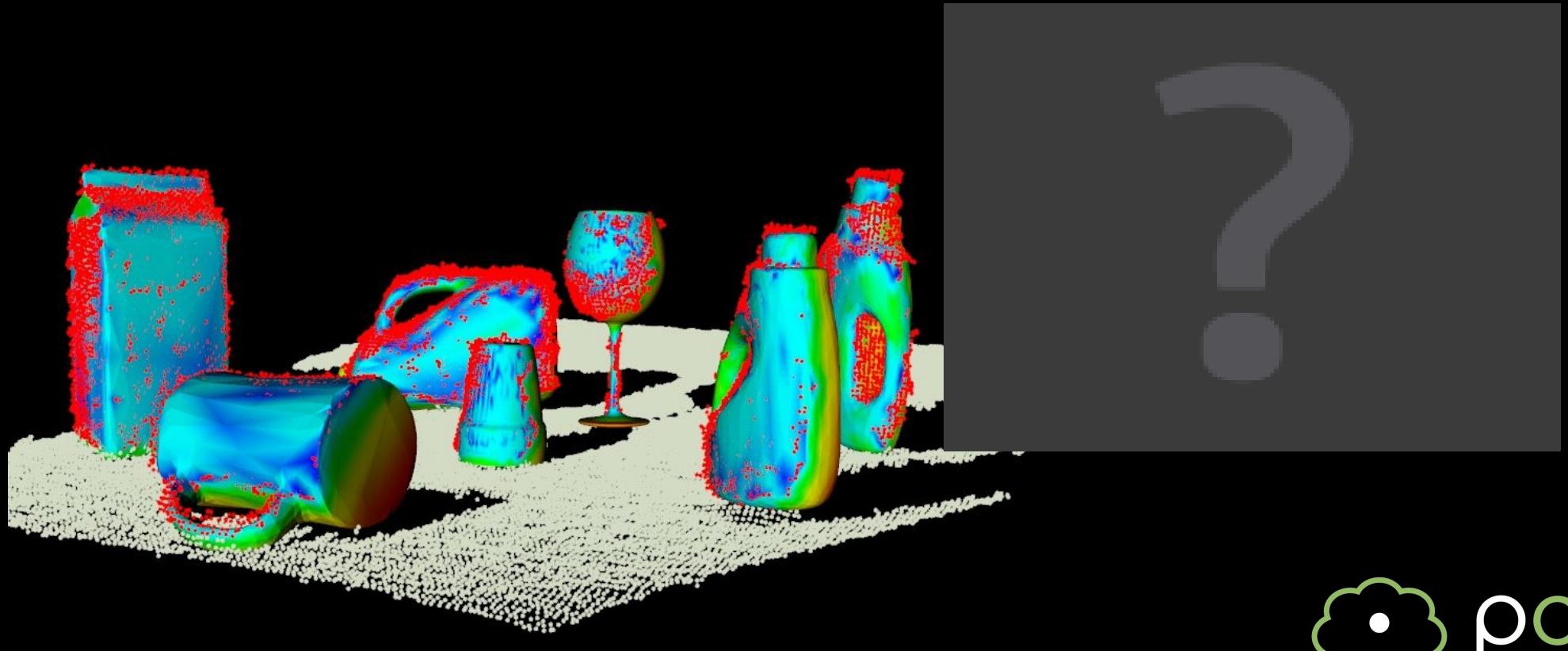
Applications of Point Clouds

- Point clouds enable very cool applications
- We are trying to solve extremely hard problems



Applications of Point Clouds

- Point clouds enable very cool applications
- We are trying to solve extremely hard problems



Applications of Point Clouds

- Point clouds enable very cool applications
- We are trying to solve extremely hard problems



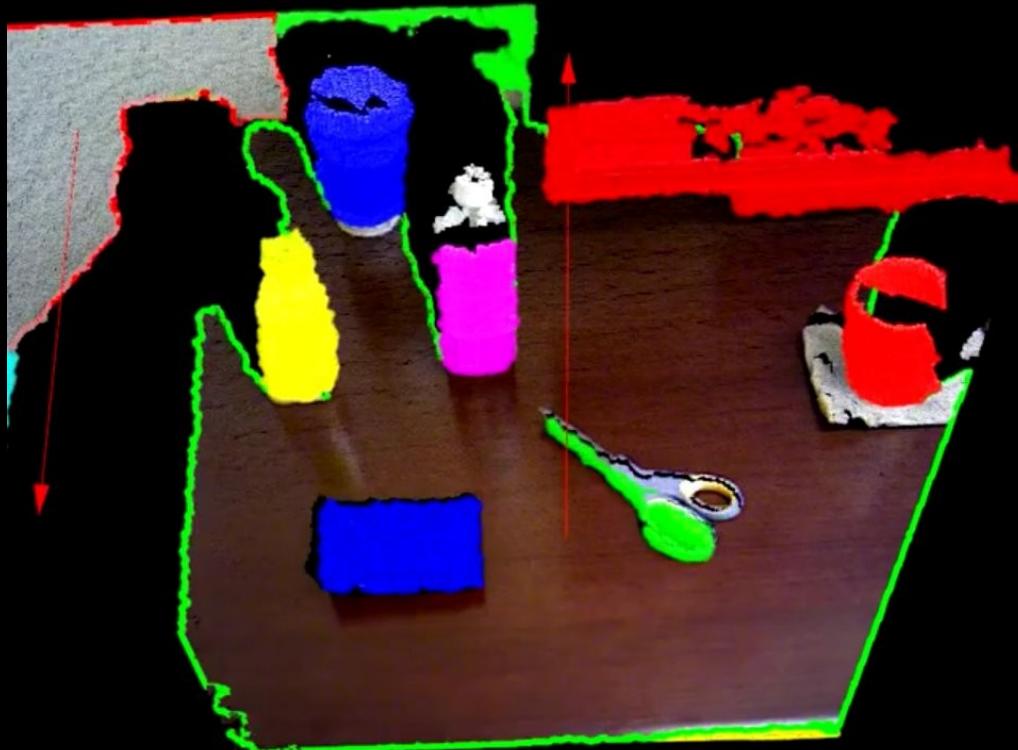
Applications of Point Clouds

- Point clouds enable very cool applications
- We are trying to solve extremely hard problems



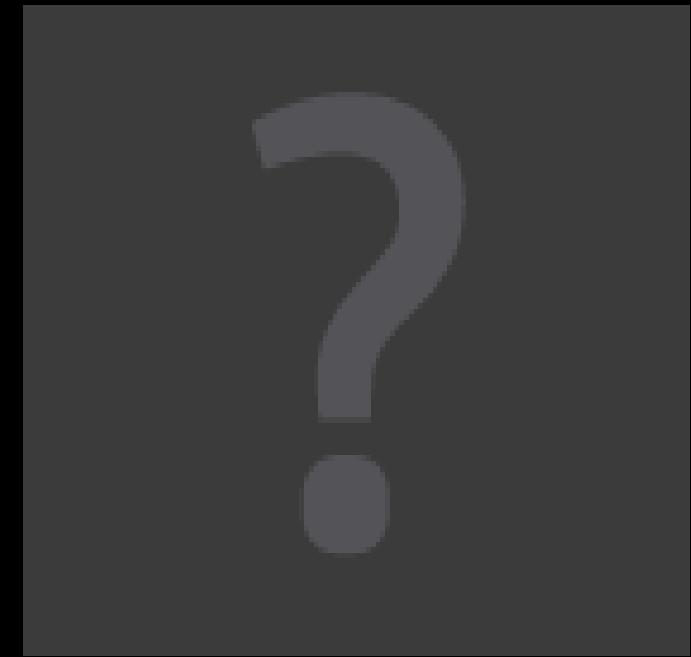
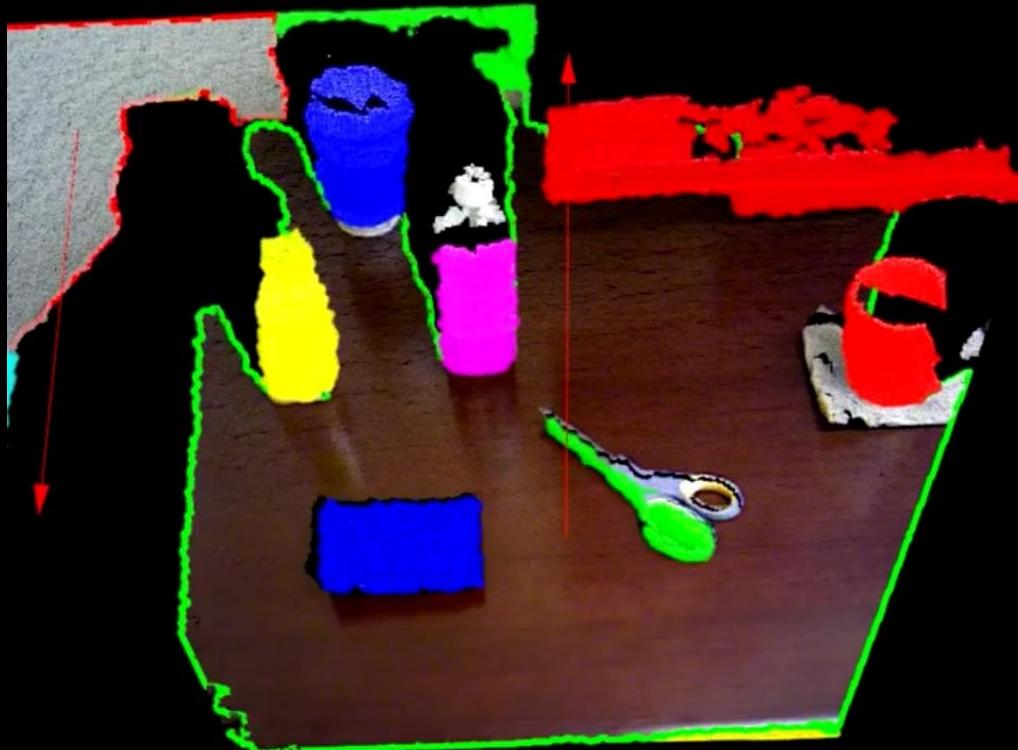
Applications of Point Clouds

- Point clouds enable very cool applications
- We are trying to solve extremely hard problems



Applications of Point Clouds

- Point clouds enable very cool applications
- We are trying to solve extremely hard problems



Applications of Point Clouds

- Point clouds enable very cool applications
- We are trying to solve extremely hard problems



Applications of Point Clouds

- Point clouds enable very cool applications
- We are trying to solve extremely hard problems

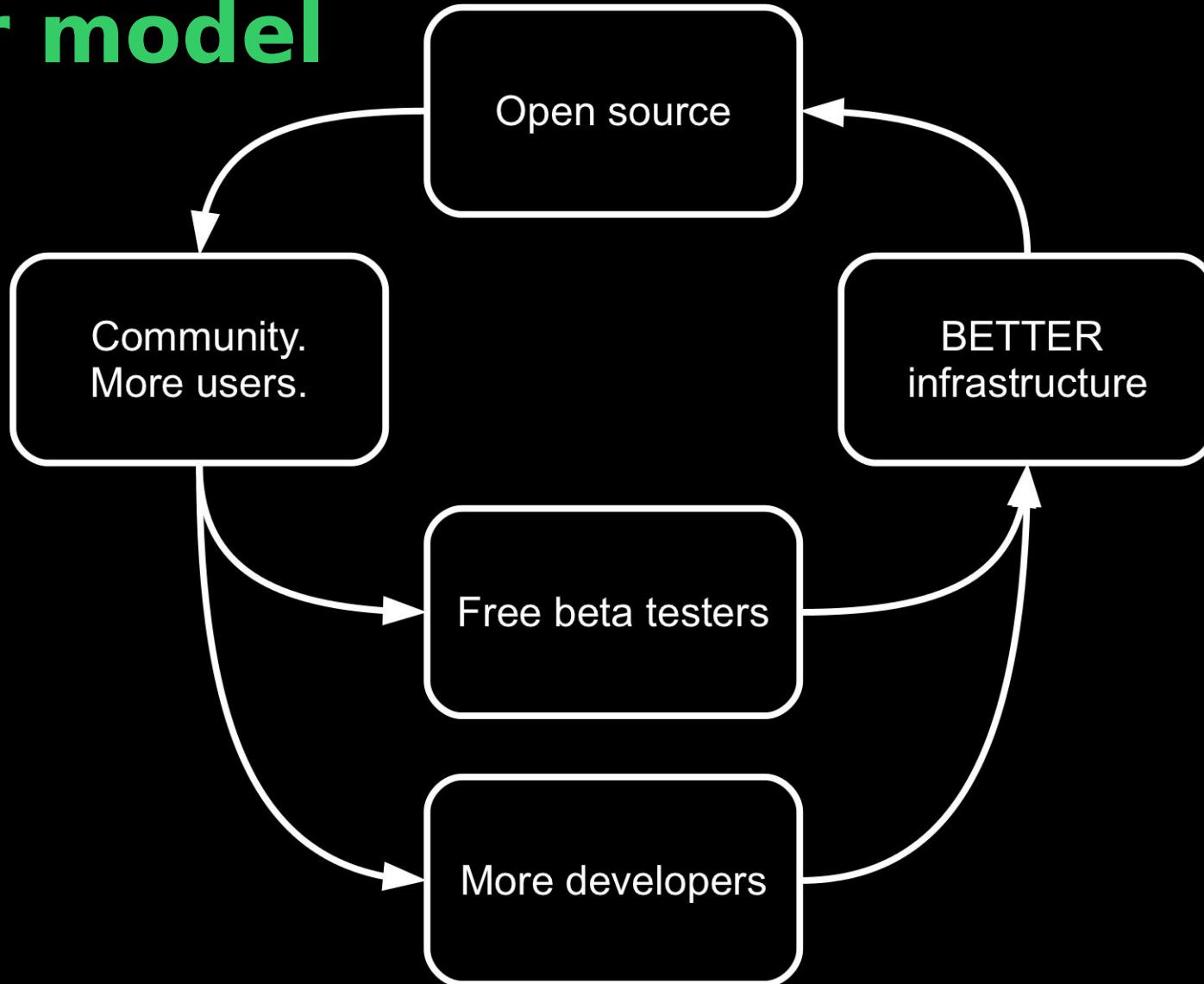


Applications of Point Clouds

- Point clouds enable very cool applications
- We are trying to solve extremely hard problems



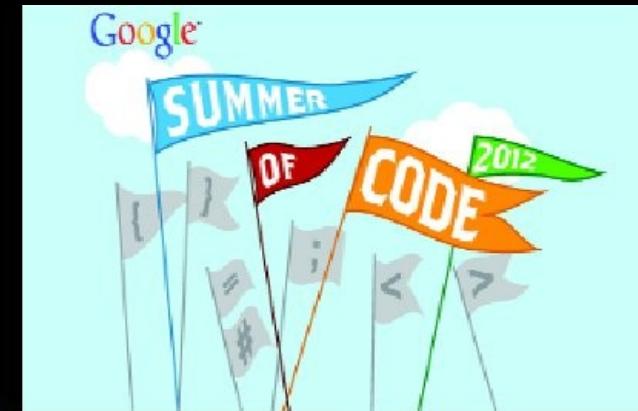
Our model



Commercial partnerships



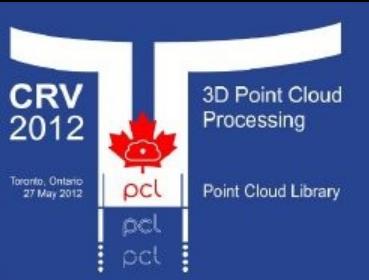
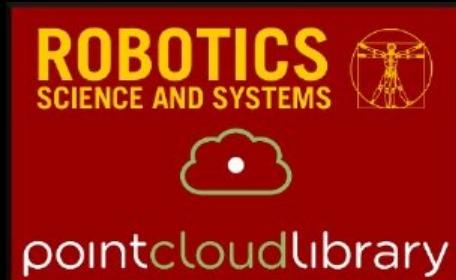
Current sponsors



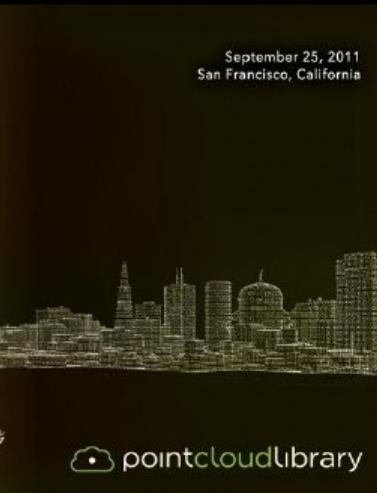
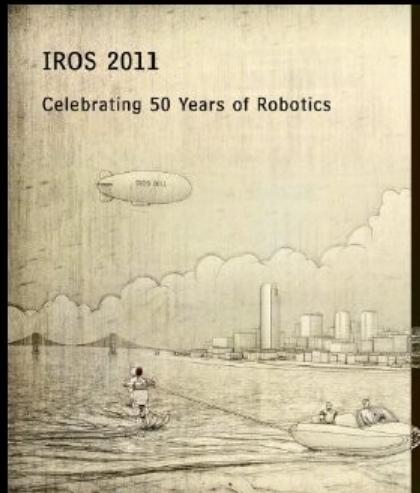
GPU

TECHNOLOGY
CONFERENCE

Tutorials



Advanced 3D Image Processing
with Point Cloud Library



Advanced 3D Point Cloud Processing
with Point Cloud Library (PCL)



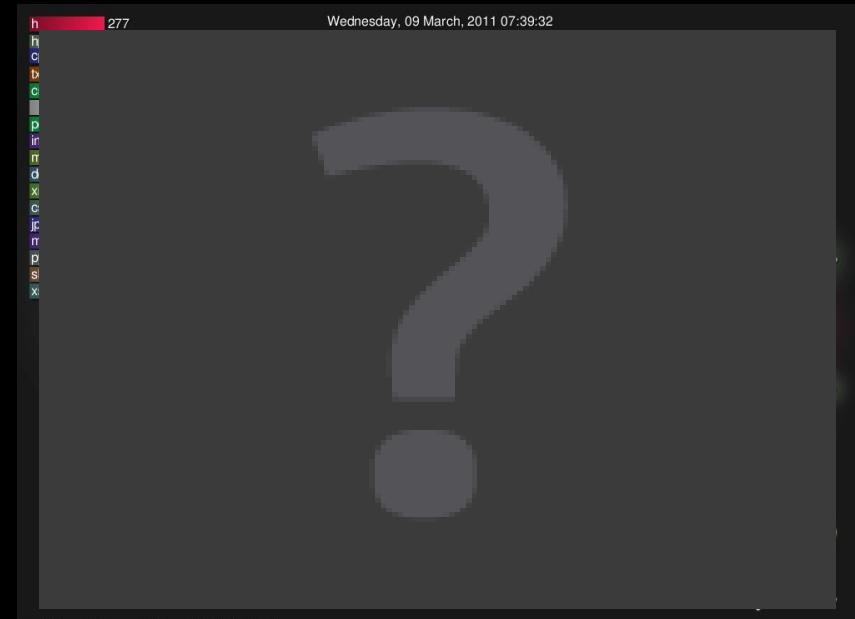
May 16, 2012: Saint Paul, Minnesota, USA





OSS 2011 challenge first prize

24/7 development



What is PCL?

[Learn more](#)



Point cloud basics

A point cloud is a set of points in three-dimensional coordinate system. These points are ordered by x, y, and z coordinates, and typically are intended to be representative of the external surface of an object.



It is easy to show for $N = 2$ that

$$\int_{\Omega} \left(\frac{\partial^2 u}{\partial x_1 \partial x_2} \frac{\partial^2 v}{\partial x_1 \partial x_2} - \frac{\partial^2 u}{\partial x_1^2} \frac{\partial^2 v}{\partial x_2^2} - \frac{\partial^2 u}{\partial x_2^2} \frac{\partial^2 v}{\partial x_1^2} \right) dx_1 dx_2$$

$$= \int_{\Gamma} \left(\frac{\partial^2 u}{\partial \tau^2} \frac{\partial v}{\partial \tau} + \frac{\partial^2 u}{\partial \tau \partial \nu} \frac{\partial v}{\partial \nu} \right) d\sigma, \quad \forall u \in H^3(\Omega), v \in H^2(\Omega). \quad (1.2.10)$$

Let $\tau = (\tau_1, \tau_2)$ be the unit tangent vector along Γ , $\frac{\partial}{\partial \tau}$ the derivative along the tangential direction, and

$$\frac{\partial^2 u}{\partial \tau^2} = D^2 u \cdot (\tau, \tau) = \sum_{i,j=1}^2 \tau_i \tau_j \frac{\partial^2 u}{\partial x_i \partial x_j},$$

$$\frac{\partial^2 u}{\partial \tau \partial \nu} = D^2 u \cdot (\tau, \nu) = \sum_{i,j=1}^2 \tau_i \tau_j \frac{\partial^2 u}{\partial x_i \partial x_j}.$$

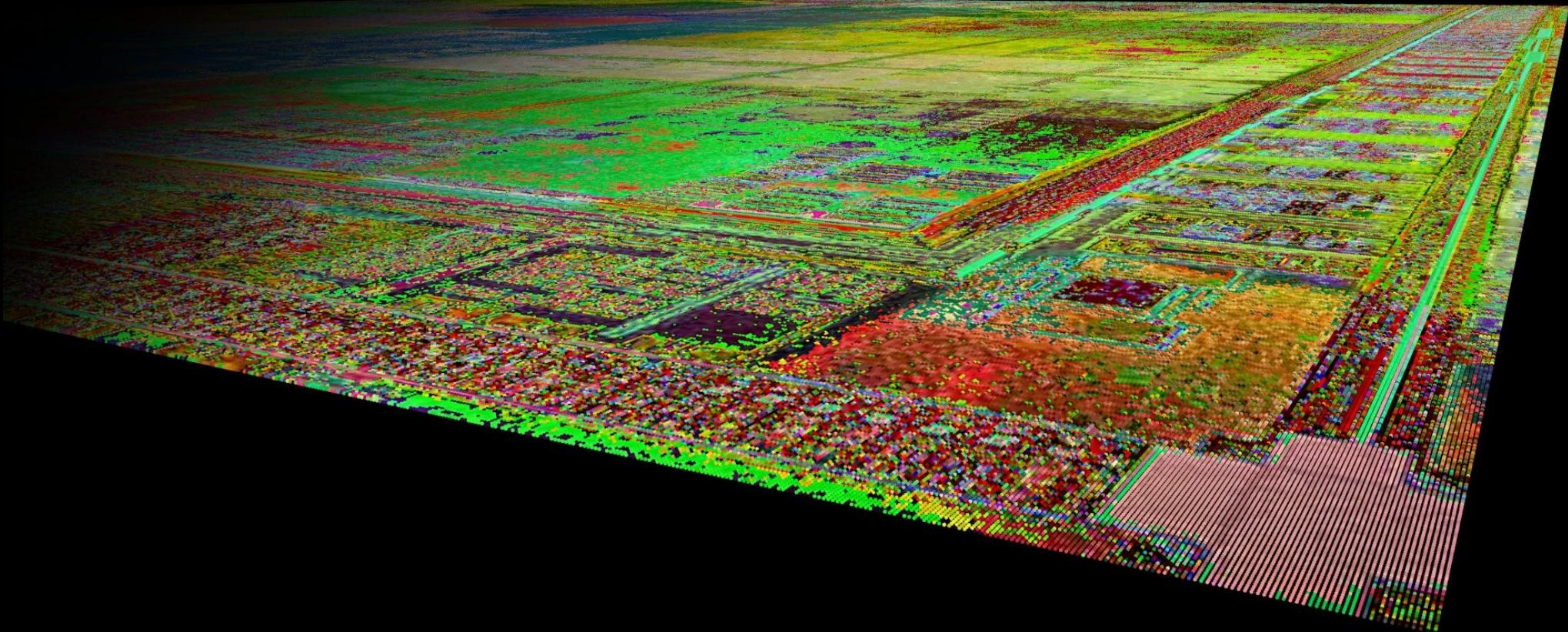
Surface reconstruction

```

bminu, bmaxu // min/max reachable p
P = {p1, ..., pn} // set of 3D points
P0 = {pi, bminu ≤ piu ≤ bmaxu} // subset
for all pi ∈ P0
    estimate (ni) from Pk // estimate surface normal
    if (ni = n̄  $\times$  Z ≈ 0) // check if the point is flat
        Pz ← pi // add pi to the Pz set
    estimate (C = {Pz1, ..., Pzm}), Pzl ⊂ C
    for all cl = Pzl ∈ C
        if find the best plane fit using sample points
            estimate ([a, b, c, d]), a · pzl + b · x + c · y + d · z
            estimate (amin, amax) // find the a min/max box
            M ← F(cl) // add the fitable parameter
        for all pi ∈ P
            if (aminu ≤ piu ≤ amaxu, aminv ≤ piv ≤ amaxv)
                Pc ← pi // add pi to the Pc set
        estimate (O = {Pc1, ..., Pcn}), Pcl ⊂ Pc
        for all ol ∈ O
            M ← F(ol) // add the object parameter
    
```



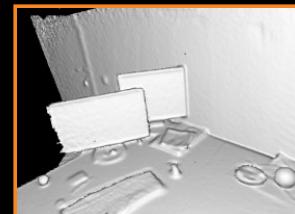
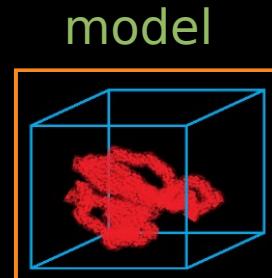
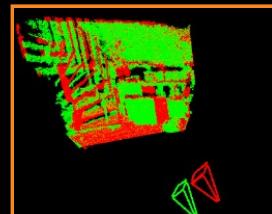
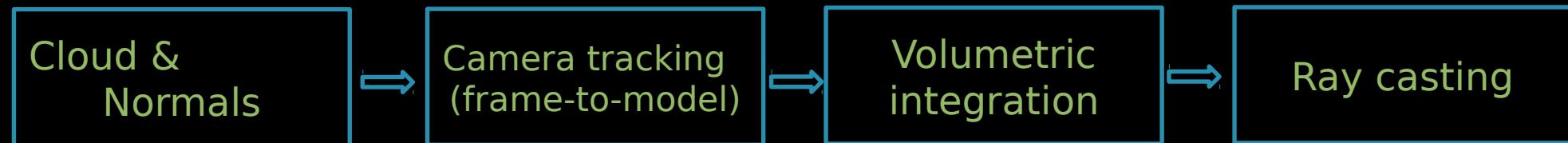
- General overview on PCL - Radu
- CUDA optimizations for KinFu and KinSkel - Michael



- Extremely large speedups with CUDA processing!
 - depth to cloud assembly: 10x
 - 3D feature estimation: 50-500x
- We are enabling research otherwise not possible!

Kinect Fusion

first open source implementation



References:

- R.Newcombe at all, “Kinectfusion: Real-time dense surface mapping and tracking”. In ISMAR, IEEE, 2011.

Kinect Fusion - camera tracking ICP

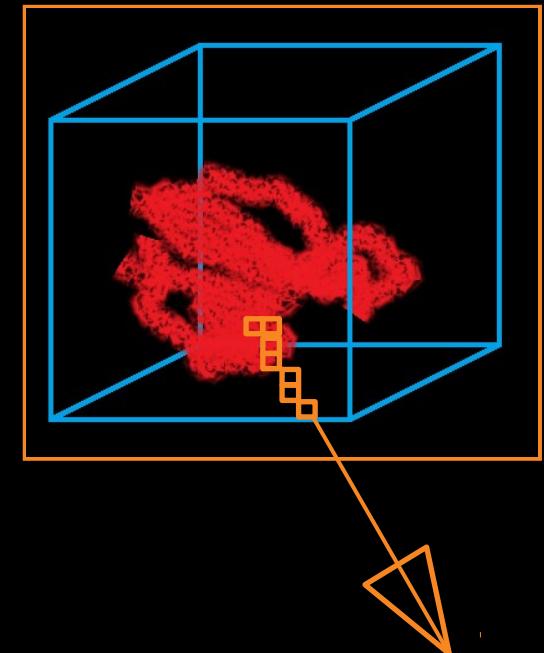
- Want to compute: $\alpha, \beta, Y, tx, ty, tz$
- At each iteration
 - Find matching points between two frames **in parallel**
 - Cost: Sum of point-to-plane distances for all matching pixels
 - Least squares problem $Ax = b$ (6×6)
 - Coefficients are computed using **standard GPU reduction** over 640×480 pixels, next linear system is solved on CPU

Kinect Fusion - volume integration

- Voxel grid 512x512x512 (voxels contain avg. distance to surface)

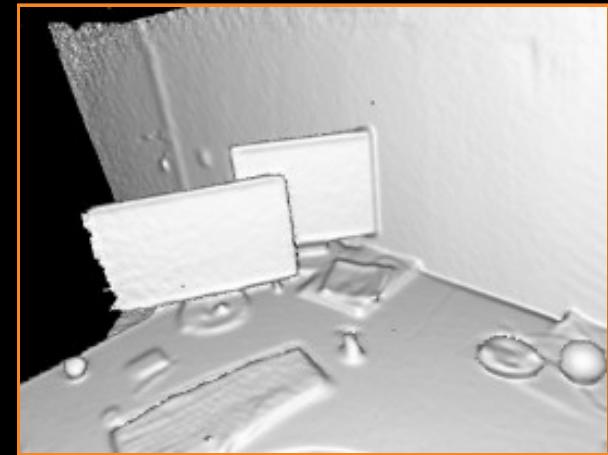
Know current distance measurements

- for all voxels in camera frustum
- For each voxel **in parallel** on GPU
 - Project voxel to Kinect image
 - Get depth from pixel and update voxel value
(using running average)



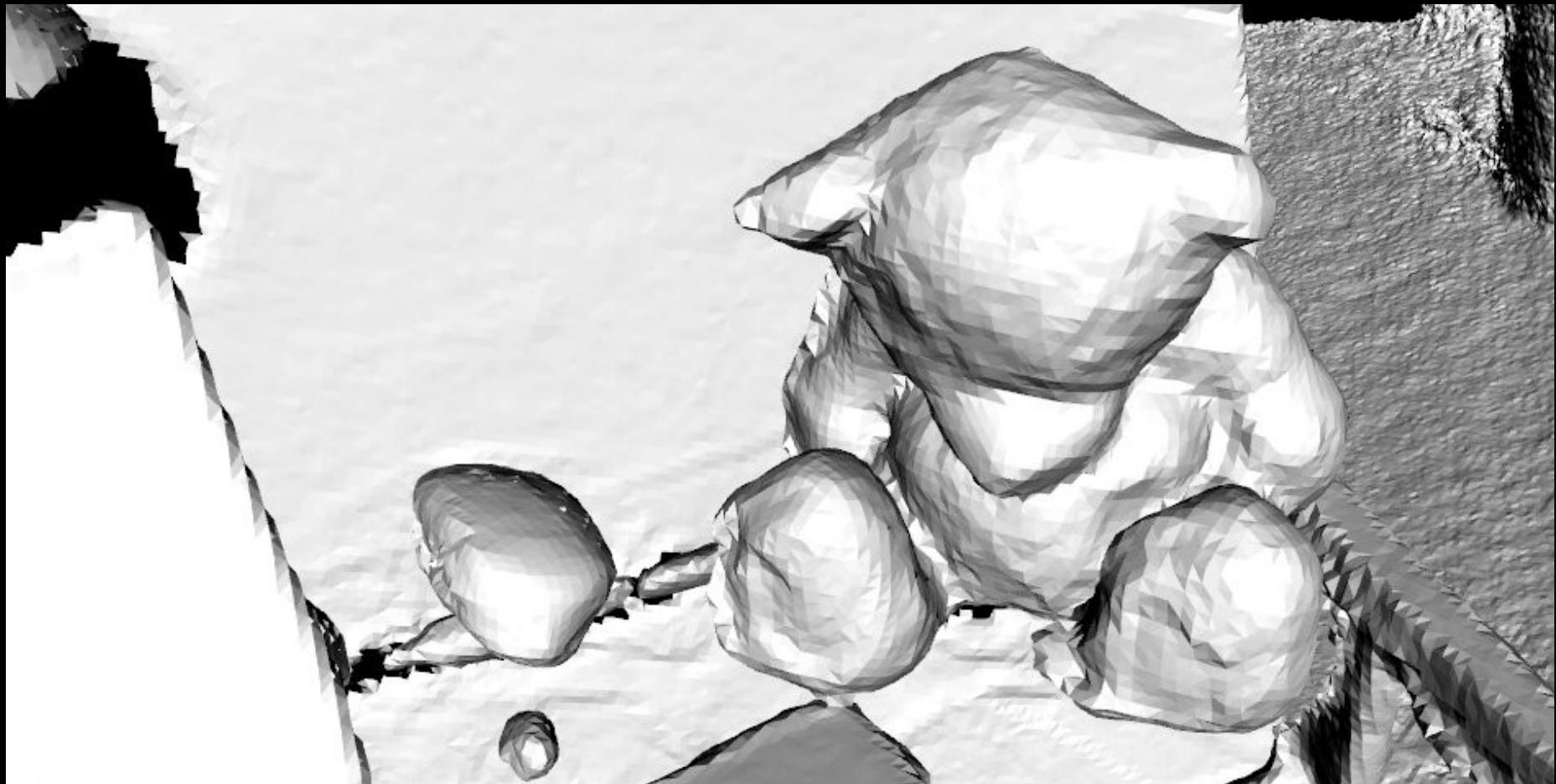
Kinect Fusion - parallel raycasting

- Want to measure exact distance to surface for each pixel from current camera position.
- For each pixel **in parallel** on GPU
- Ray is traced through integration volume
 - until surface intersection



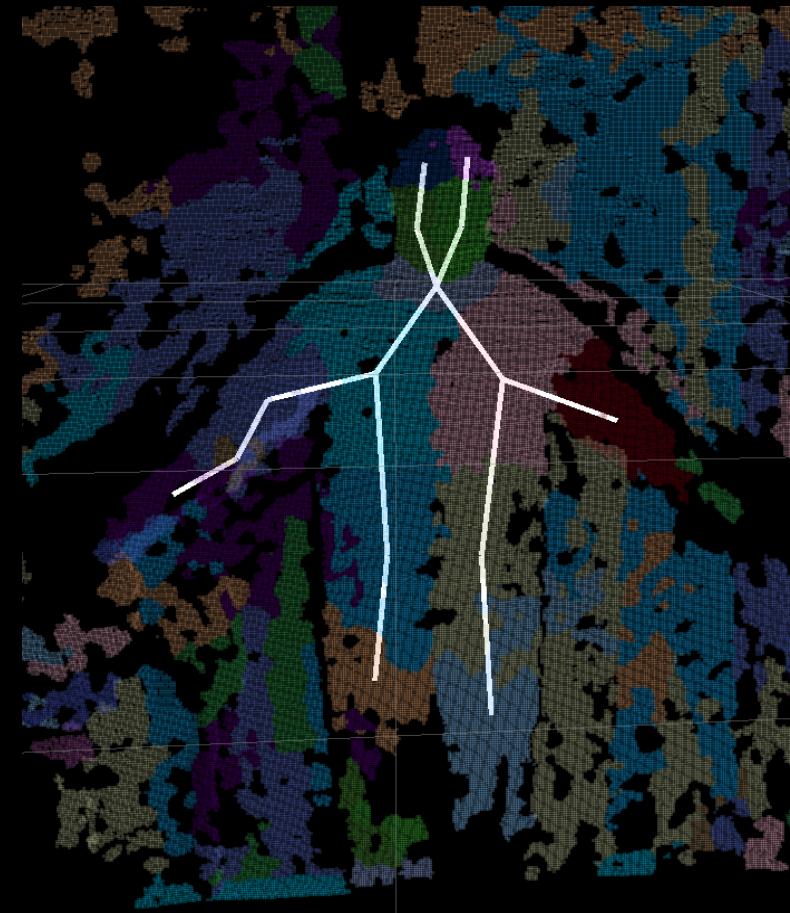
Output is noise-less and much more precise than frame from Kinect (because volume contains averaged data from different positions)

Kinect Fusion - demo



Kinect Skeleton Tracking

first open source implementation



Algorithm overview

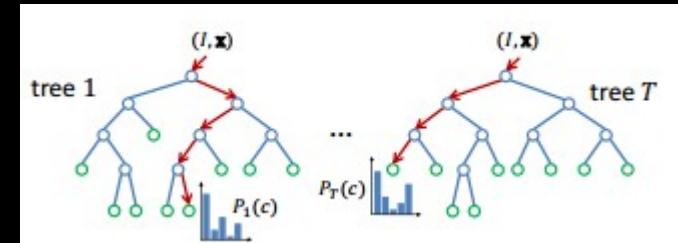
Given a depth image...

- **Segment** the person from the background
- **Label** each pixel with a body part
- **Cluster** part labels to locate skeletal joints

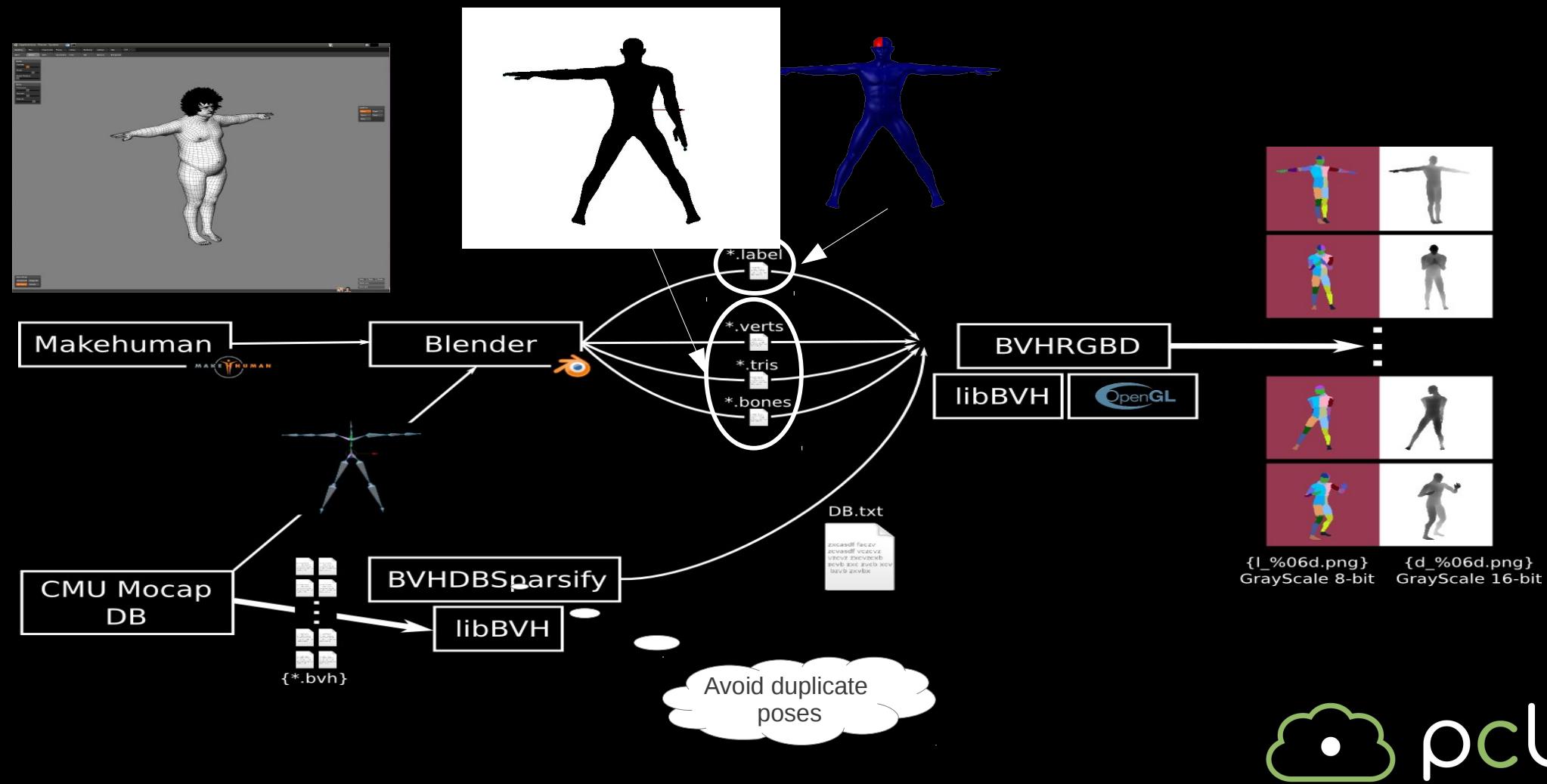


Part labeling

- Use a “forest” of decision trees to classify each pixel
- Train these decision trees using **lots of synthetic data**



Synthesizing training data



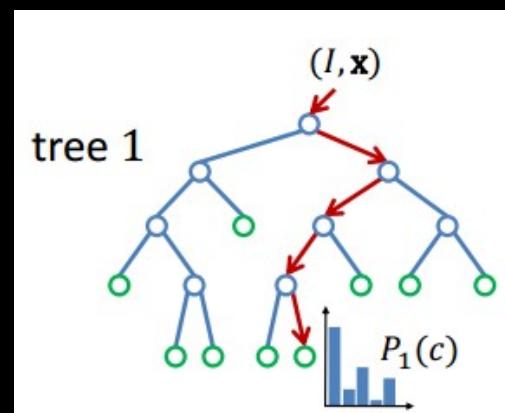
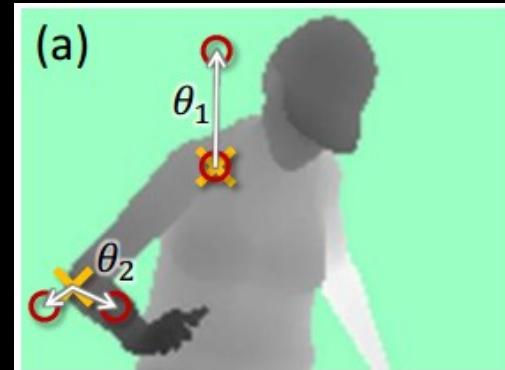
Training data

- Use virtual body to render lots of depth maps and labels
 - Sample lots of small patches from each image
 - Train decision trees to predict the label for each patch



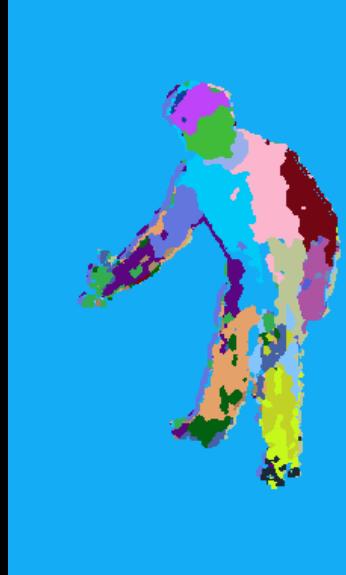
Decision tree structure

- Each node evaluates a simple local feature based on two pixel locations and a threshold (chosen during training)
- To evaluate: subtract the depth values at the given pair of pixels and compare to the difference to the given threshold
- If less than the threshold branch left, otherwise branch right
- Recurse until it reaches a leaf node
 - Leaf node contains the probability distribution of each body part label



Skeleton tracking

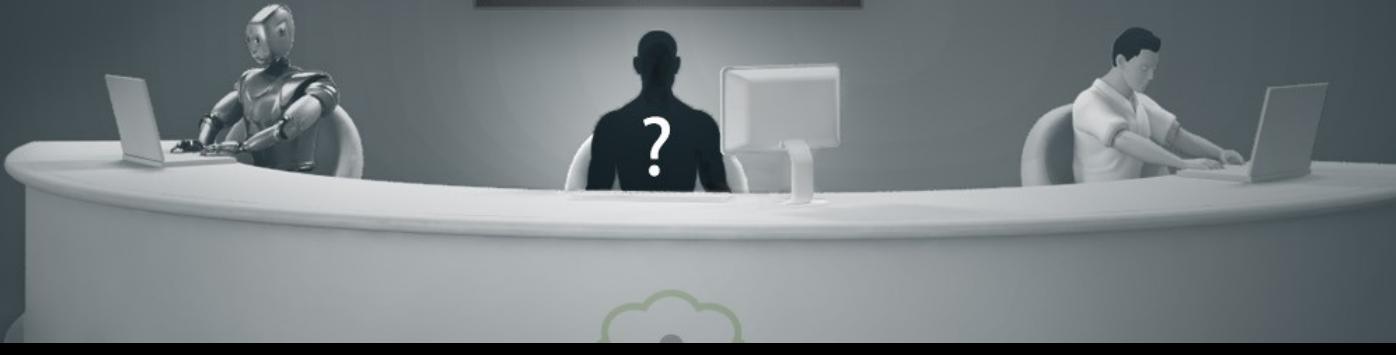
- Remove the background
- Use each tree in the forest to label each pixel
 - Highly parallel
- Average the outputs from each tree into one label image
- Cluster each part label to find the joint position



Developers needed

Help us build the Point Cloud Library.

Join us



Thanks to our amazing developer community!

(Aitor, Alex², Anatoly, Caroline, Francisco, Julius, Koen, Pat, Raphael, Raymond, Ryohei, Suat, Walter)

