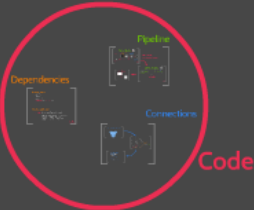


Researcher of the Week:
superquadric-detection module

Goal



Theory



Code



You can find
all the information about superquadric-detection
module on the github repo:
<https://github.com/gulnazov/superquadric-detection>
... What happens next?



Noteworthy ...

Wrong solution
(at first glance)

Possible solutions

Noisy point clouds



Future works

- Superquadric visualization ? VTK/OpenGL
- Grasping application: Good grasping pose, Trajectory planning, Obstacle avoidance



Thank you for your attention!
...any questions or comments?

Then, let's try our code!

Researcher of the Week: superquadric-detection module

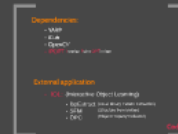
Goal



Theory



Dependencies



Pipeline



Connections



Code

Goal

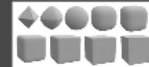
Object detection and modeling

iCub **looks** at the object



the object is **detected**

a **3D model** is computed



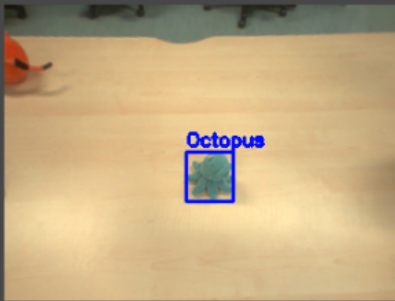
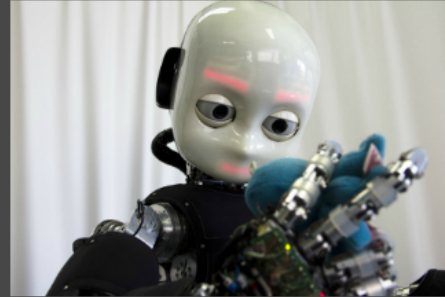
the model is **visualized**

Requests:

- 1 Detection ✓
- 2 Modeling ?
- 3 Visualization ?
- 4 ... and everything robust and in real-time !

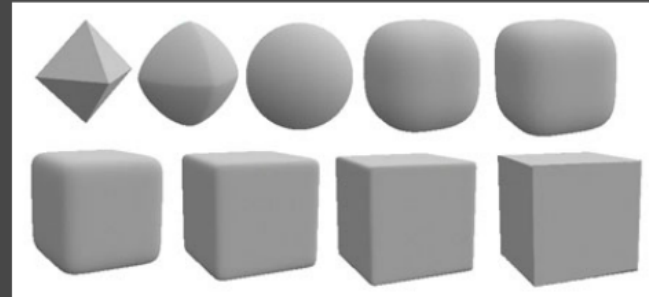
Object detection and modeling

iCub **looks** at the object



the object is **detected**

a **3D model** is computed



the model is **visualized**

Requests:

1 Detection ✓

2 Modeling ?

3 Visualization ?

4 ... and everything robust and in real-time !

Theory

Superquadric fuctions

inside-outside function:

- $F > 1$: point outside
- $F < 1$: point inside
- $F = 1$: point on surface

$$F(x, y, z, \lambda) = \left(\left(\frac{x}{\lambda_1} \right)^{\lambda_1} + \left(\frac{y}{\lambda_2} \right)^{\lambda_2} \right)^{\lambda_3} + \left(\frac{z}{\lambda_3} \right)^{\lambda_4}$$

[5 parameters
for shape]

+

[6 parameters
for pose]

Optimization problem

$$\min_{\lambda} \sum_{i=0}^N \left(\sqrt{\lambda_1 \lambda_2 \lambda_3} F^{\lambda_4}(x_i, y_i, z_i, \lambda) - 1 \right)^2$$

- shape independence
- minimum volume

Superquadric functions

inside-outside function:

- $F > 1$: point outside
- $F < 1$: point inside
- $F = 1$: point on surface

$$F(x, y, z, \lambda) =$$

$$\left(\left(\frac{x}{\lambda_1} \right)^{\frac{2}{\lambda_5}} + \left(\frac{y}{\lambda_2} \right)^{\frac{2}{\lambda_5}} \right)^{\frac{\lambda_5}{\lambda_4}} + \left(\frac{z}{\lambda_3} \right)^{\frac{2}{\lambda_4}}$$

5 parameters
for shape

+

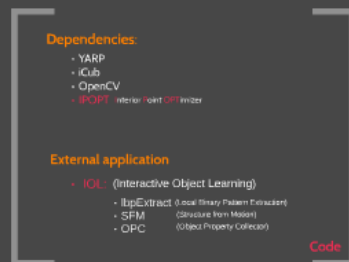
6 parameters
for pose

Optimization problem

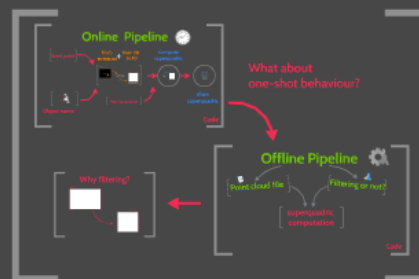
$$\min_{\lambda} \sum_{i=0}^N \left(\sqrt{\lambda_1 \lambda_2 \lambda_3} F^{\lambda_4}(x_i, y_i, z_i, \lambda) - 1 \right)^2$$

- shape independence
- minimum volume

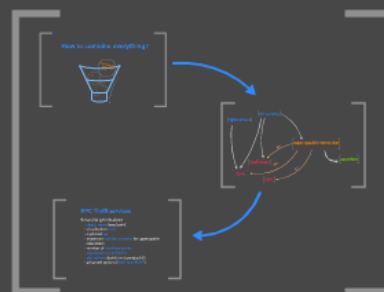
Dependencies



Pipeline



Connections



Code

Dependencies:

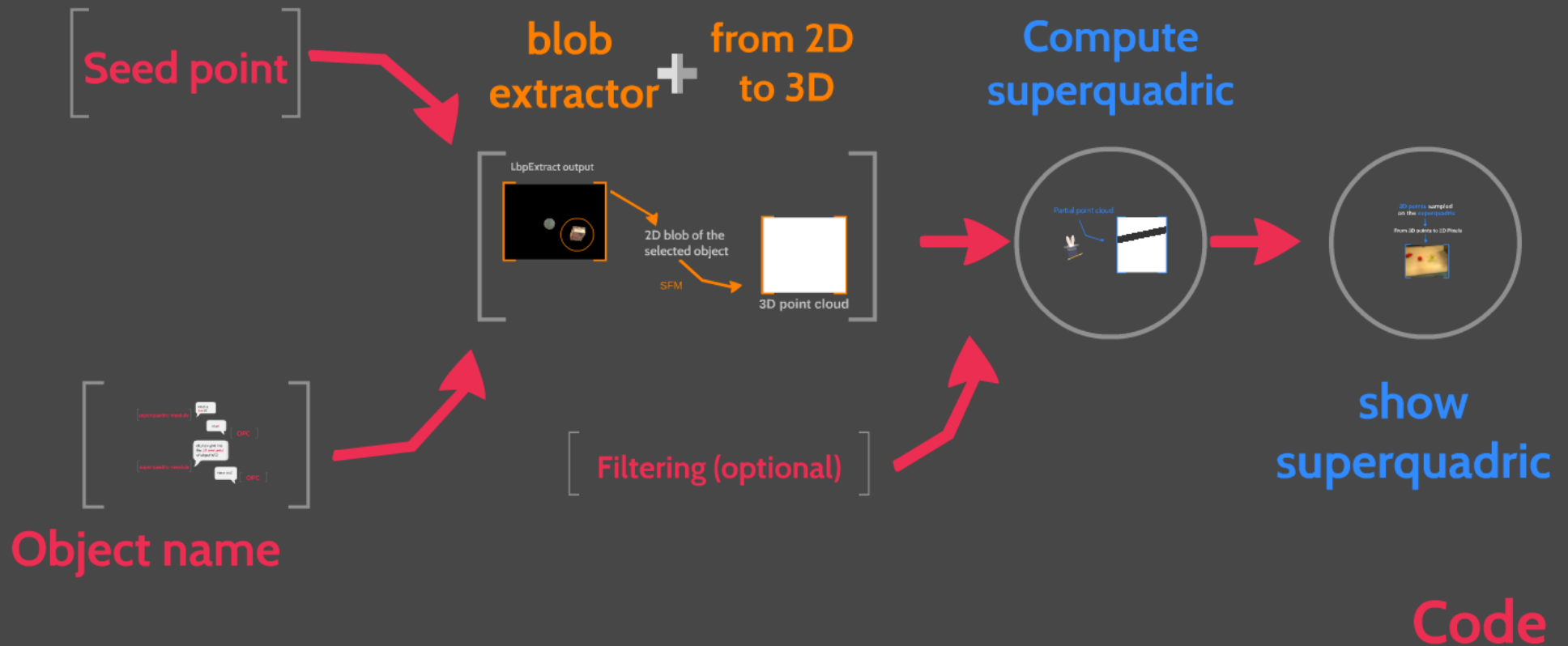
- YARP
- iCub
- OpenCV
- **IPOPT** Interior Point **OPT**imizer

External application

- **IOL**: (Interactive Object Learning)
 - lbpExtract (Local Binary Pattern Extraction)
 - SFM (Structure from Motion)
 - OPC (Object Property Collector)

Code

Online Pipeline



[superquadric-module]

which is
box id?

it's 0!

[OPC]

[superquadric-module]

ok, now give me
the *2D seed point*
of object id 0

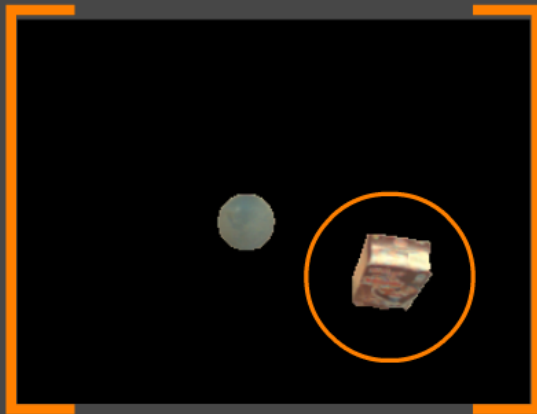
Here it is!

[OPC]

extractor

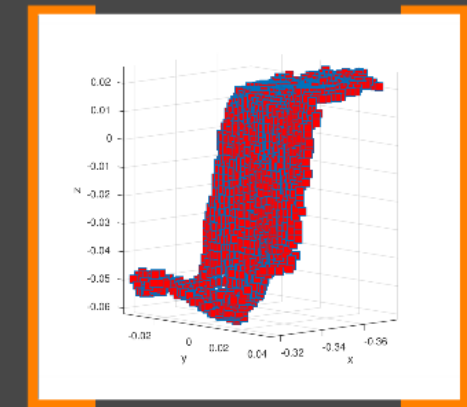
to 3D

LbpExtract output



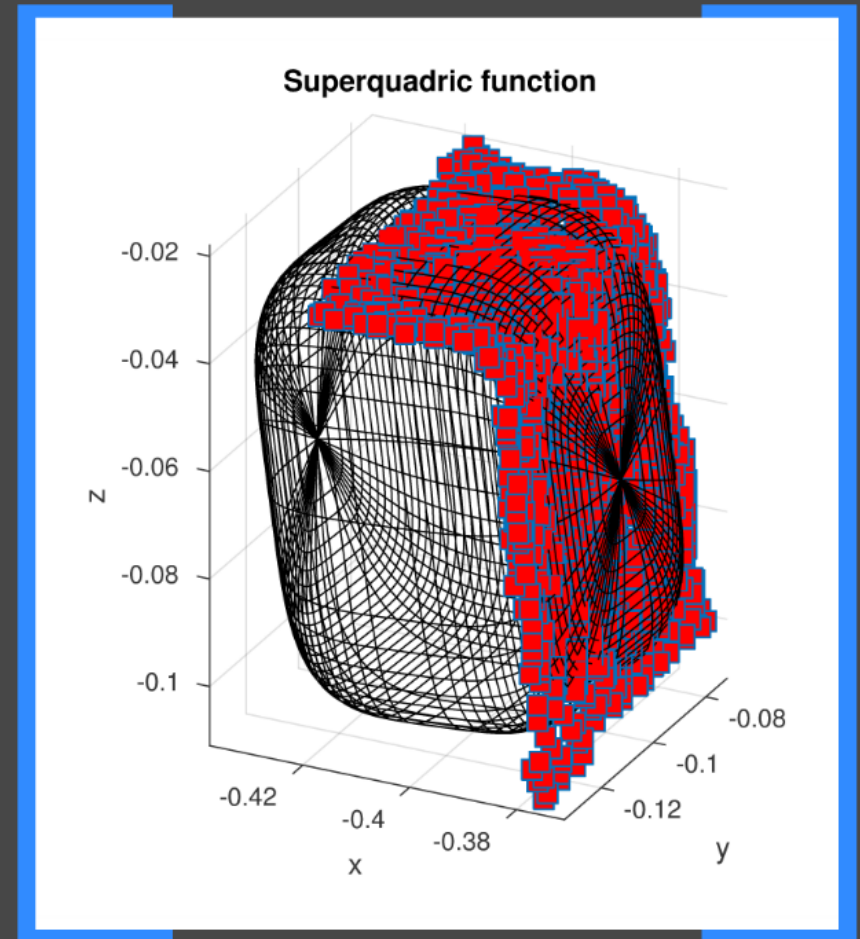
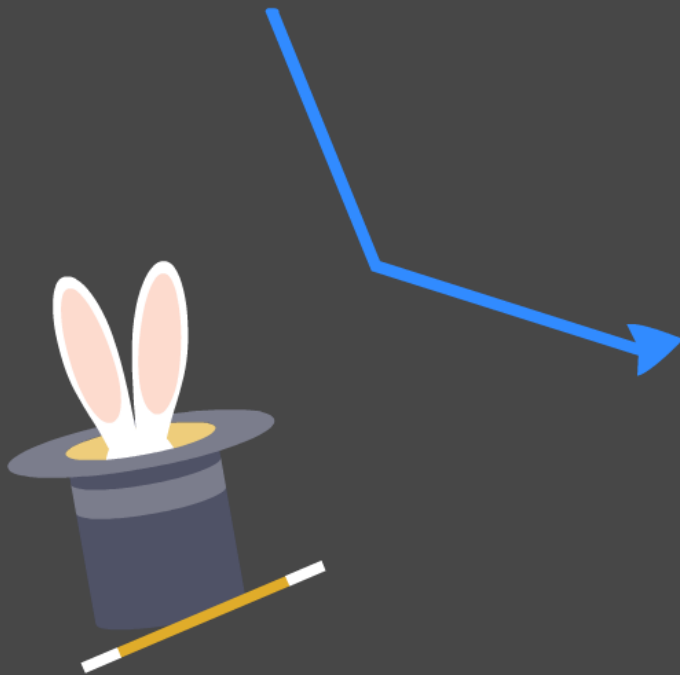
2D blob of the
selected object

SFM



3D point cloud

Partial point cloud



3D points sampled
on the **superquadric**



From 3D points to 2D Pixels



**What about
one-shot behaviour?**





Offline Pipeline



[Point cloud file]



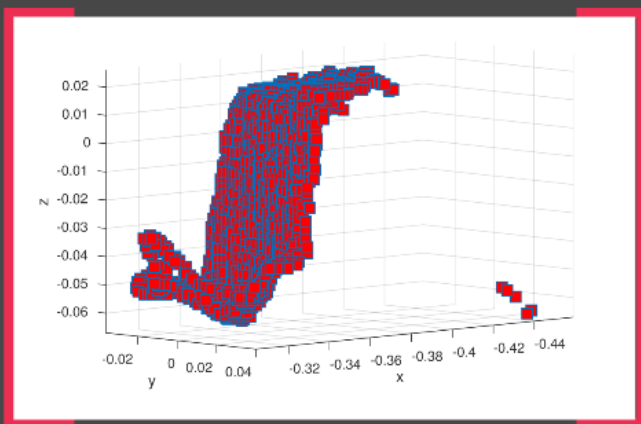
[Filtering or not?]

[superquadric
computation]

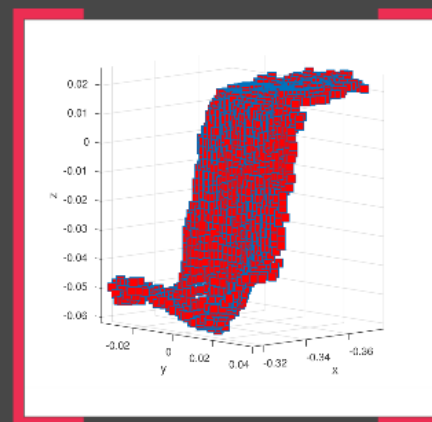
Code

Why filtering?

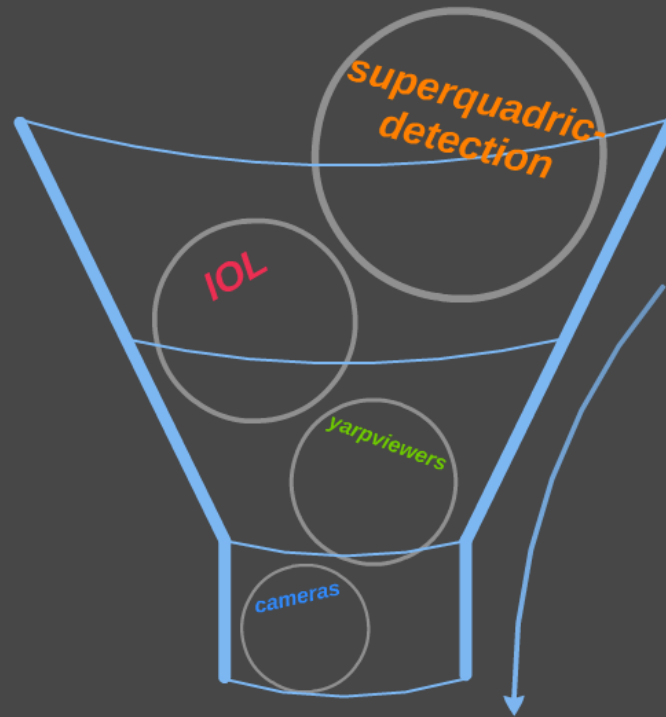
Before ...

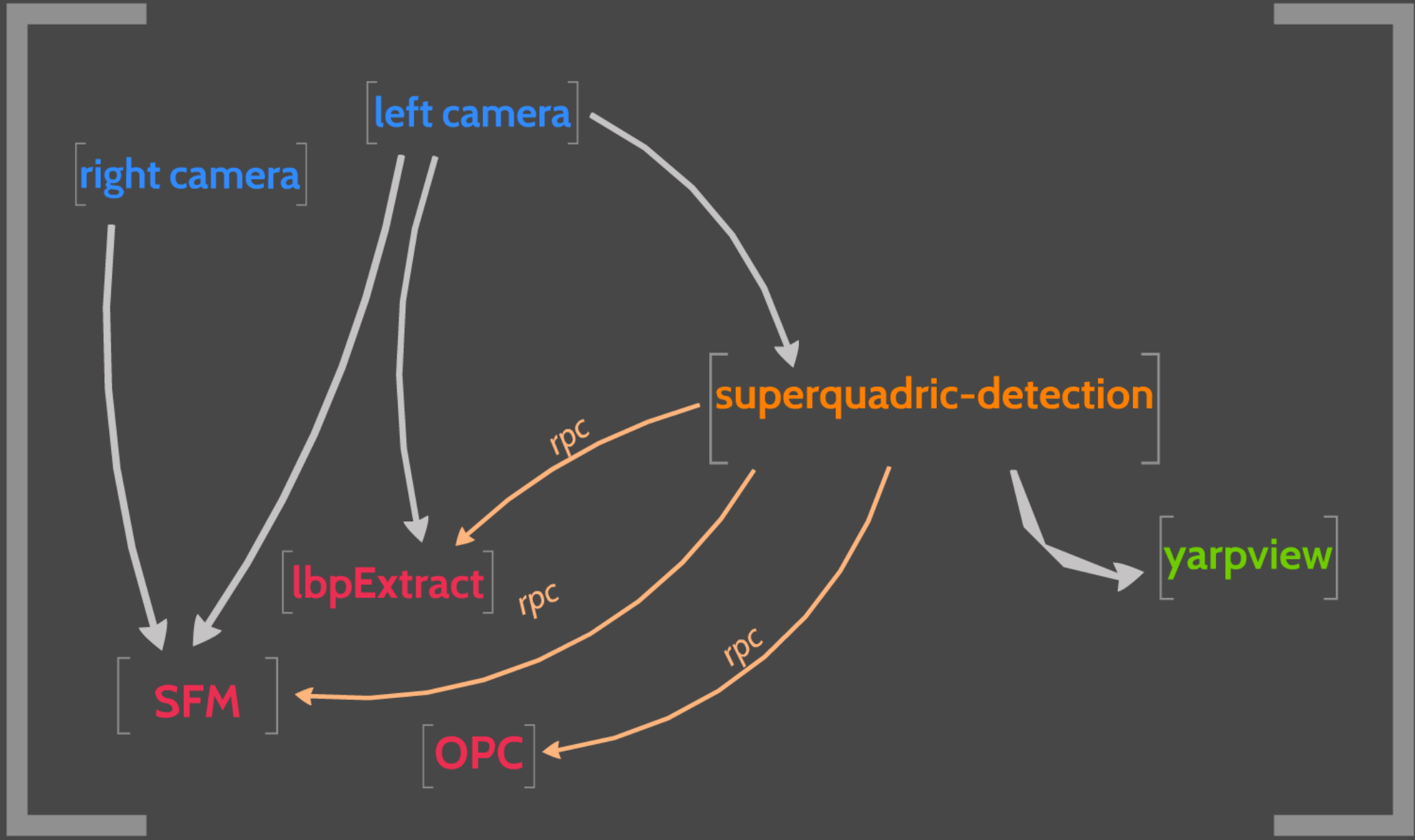


... after!



How to combine everything?





RPC Thrift services

Set and/or get info about:

- object name/seed point
- visualization color
- exploited eye
- maximum number of points for superquadric calculation
- number of visualized points
- superquadric parameters
- plot options (points or superquadric)
- advanced options (filtering & IPOPT)

Researcher of the Week: superquadric-detection module

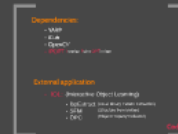
Goal



Theory



Dependencies



Pipeline



Connections



Code

You can find
all the information about superquadric-detection
module on the github repo:

<https://github.com/giuliavezzani/superquadric-detection>

... What happens next?

Noteworthy ...

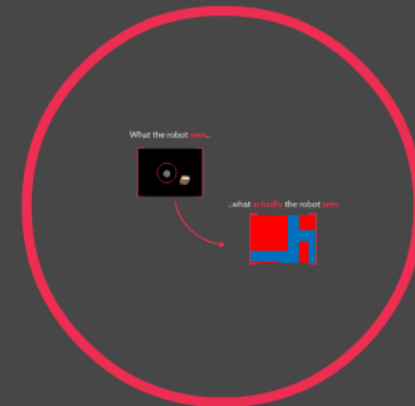
Wrong solution
(at first glance)

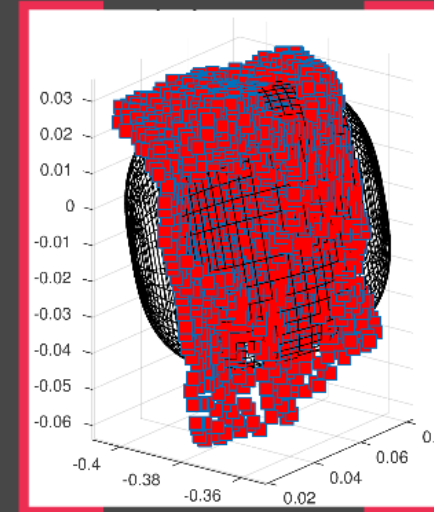
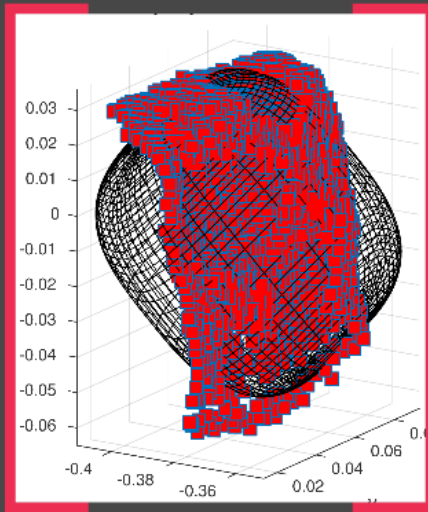


Possible solutions



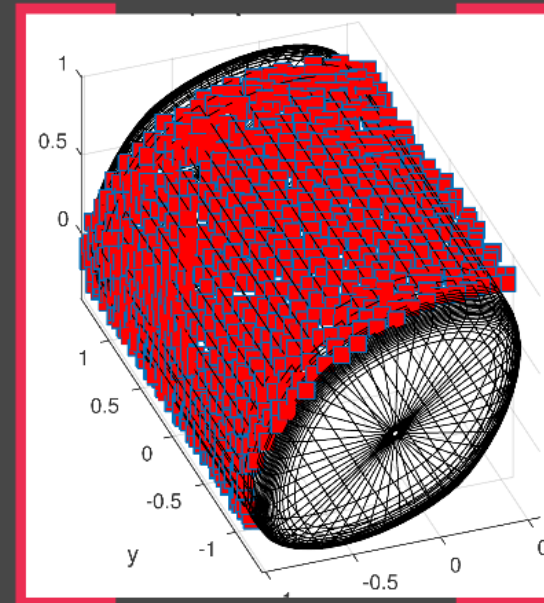
Noisy point clouds



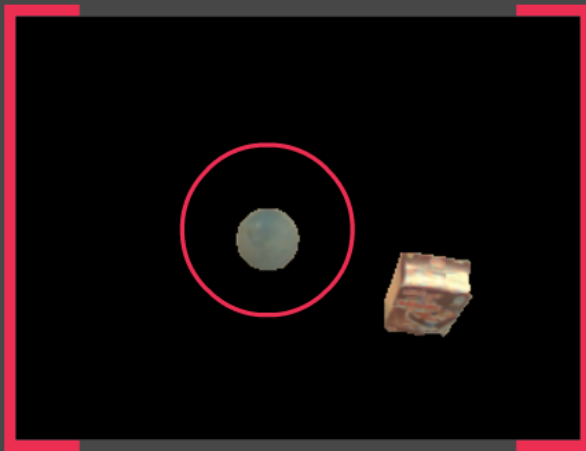


Same cost function value!

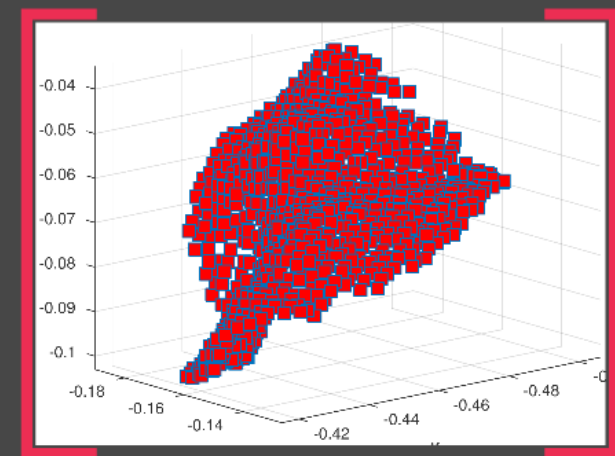
Partial **non-noisy** clouds:



What the robot **sees**..



..what **actually** the robot **sees**





Ideas

- 1 Merged point clouds (ICP)
- 2 Point cloud time refinement and filtering (Voxelization)
- 3 2D surface overlapping
- 4 Different optimization problem formulation

$$\min_{\lambda} \bar{V}(\lambda) = \sum_{i=0}^N (F(x_i, y_i, z_i, \lambda))$$

$$\min_{\lambda} \mathbb{V}(\lambda)$$

$$\sum_{i=0}^N (F(x_i, y_i, z_i, \lambda) - 1)^2 = 0$$

Future works

[coding & applications]

- Superquadric visualization ?

VTK/OpenGL

- Grasping application 

Good grasping pose



Trajectory planning

Obstacle avoidance



Thank you for your attention!

..any questions or comments?



Then, let's try our code!

