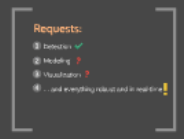
# Researcher of the Week: superquadric-detection module

## Goal


## Theory


## Code

### Pipeline

### Dependencies


### Connections

Giulia Vezzani - 13/05/2016

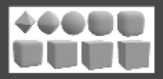
# Goal

## Object detection and modeling

iCub looks at the object

the object is detected

a 3D model is computed



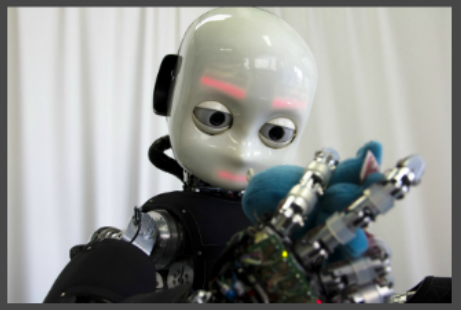
the model is visualized

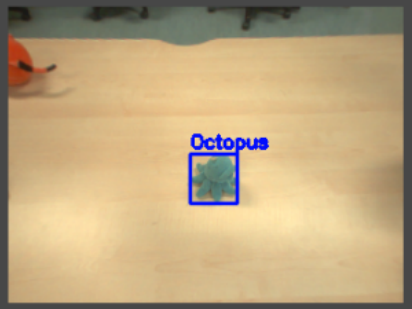
## Requests:

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

# Object detection and modeling

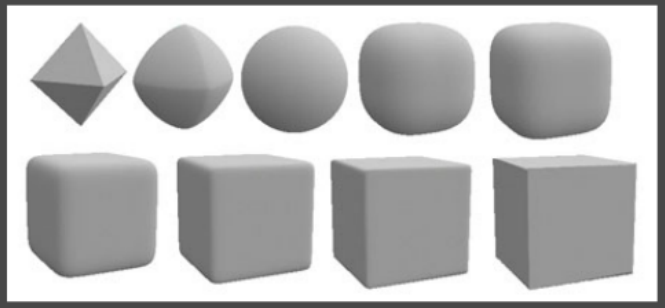
iCub **looks** at the object






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

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

[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 indipendence
- minimum volume

# Superquadric fuctions

inside-outside function:

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

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

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

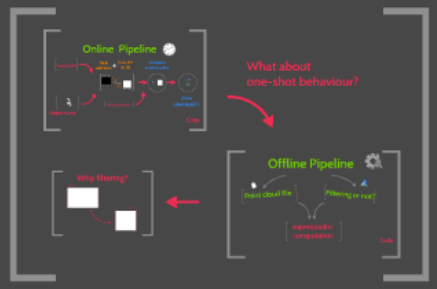
[ 5 parameters for shape ] + [ 6 parameters for pose ]

# Optimization problem

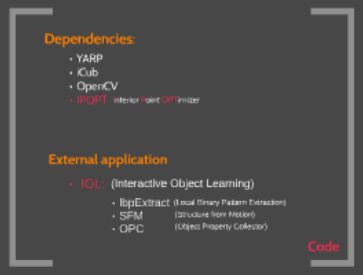
$$\min_{\lambda} \sum_{i=0}^{N} \left( \sqrt{\lambda_1 \lambda_2 \lambda_3} F^{\lambda_4}(x_i, y_i, z_i, \lambda) - 1 \right)^2$$

- shape indipendence
- minimum volume

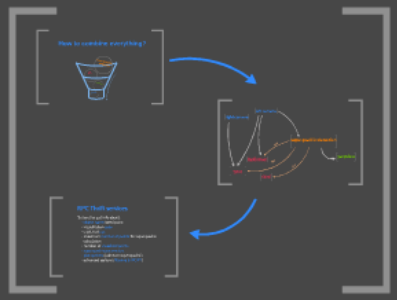
# Pipeline




# Dependencies




# Connections



# Code

## Dependencies:

- YARP
- iCub
- OpenCV
- IPOPT Interior Point OPTimizer

## External application

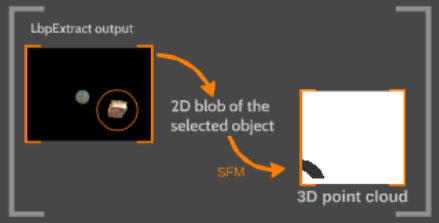
- IOL: (Interactive Object Learning)
  - lbpExtract (Local Binary Pattern Extraction)
  - SFM (Structure from Motion)
  - OPC (Object Property Collector)

Code

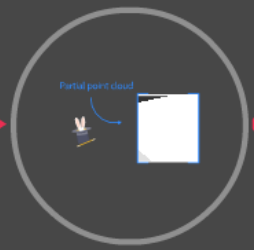
# Online Pipeline

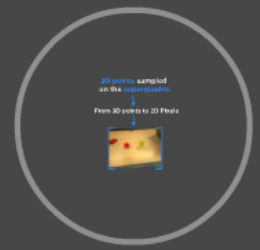
Seed point

blob extractor + from 2D to 3D




Compute superquadric



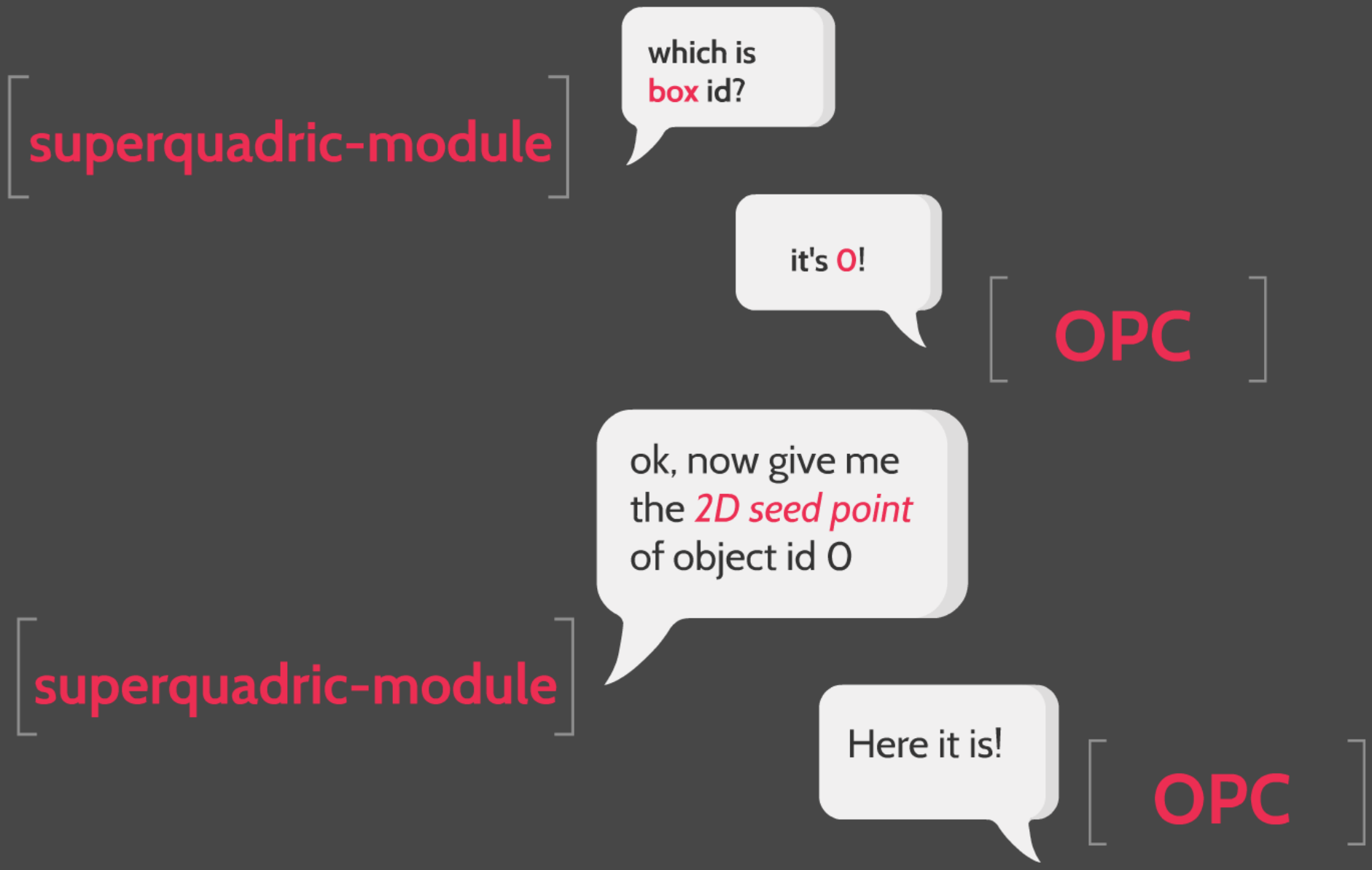



show superquadric

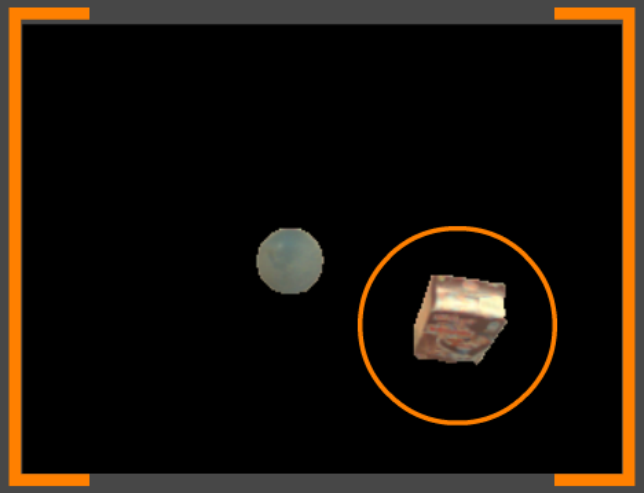
Object name

Filtering (optional)

Code

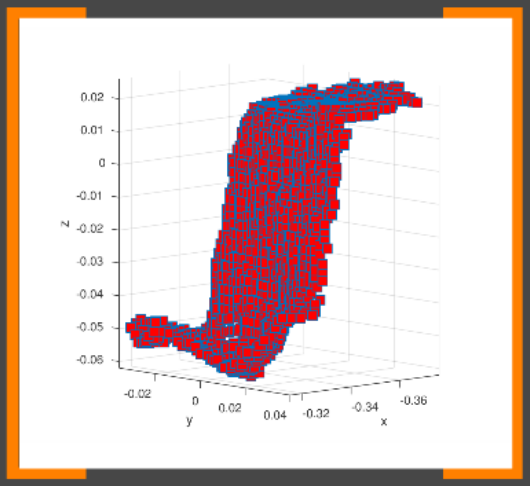

superquadric-module
which is box id?
it's 0!
OPC
ok, now give me the 2D seed point of object id 0
superquadric-module
Here it is!
OPC

LbpExtract output



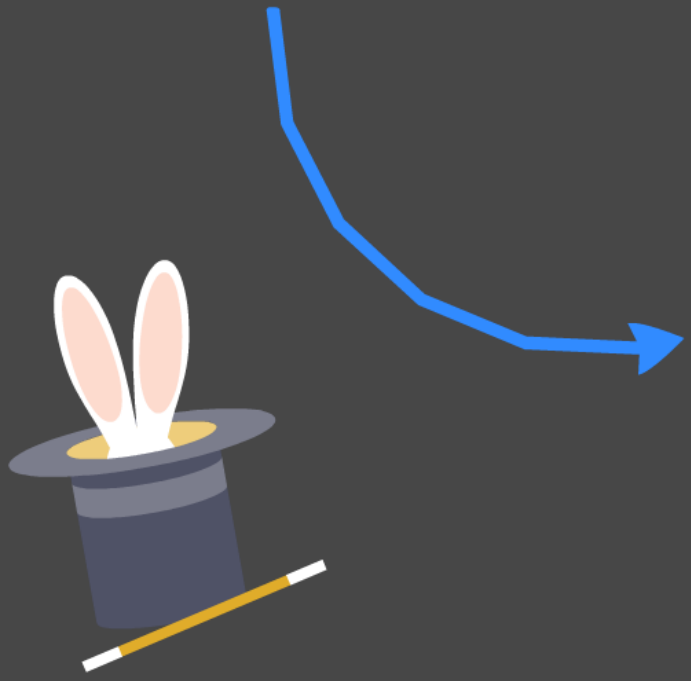
2D blob of the selected object

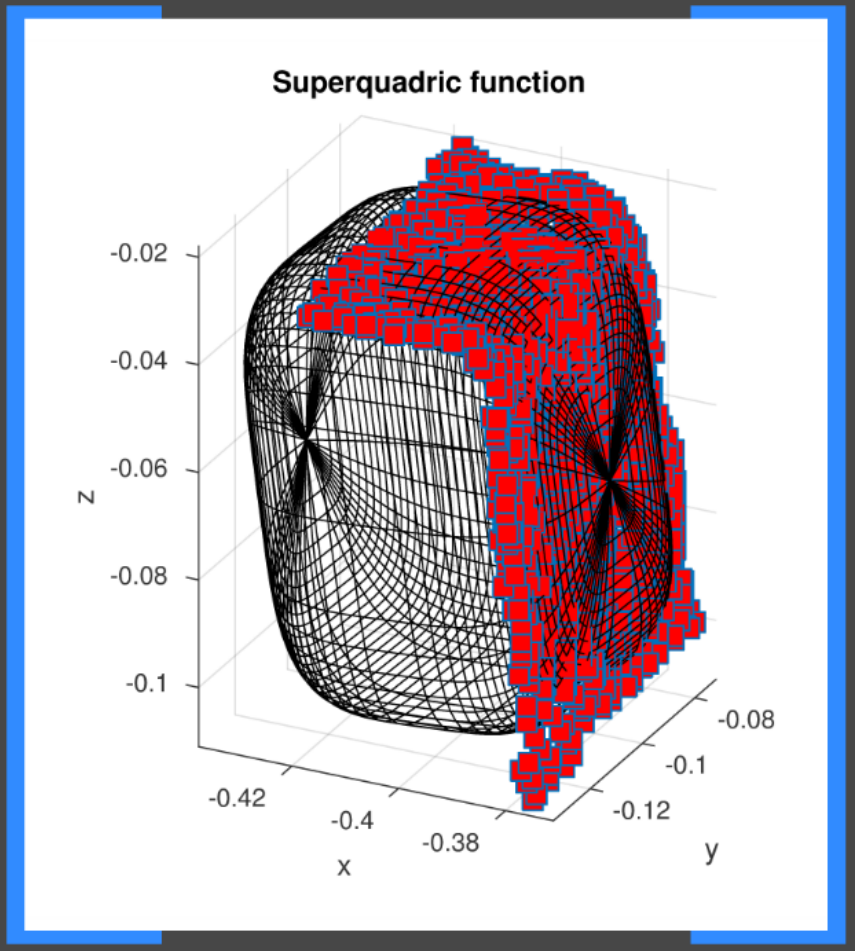
SFM




3D point cloud

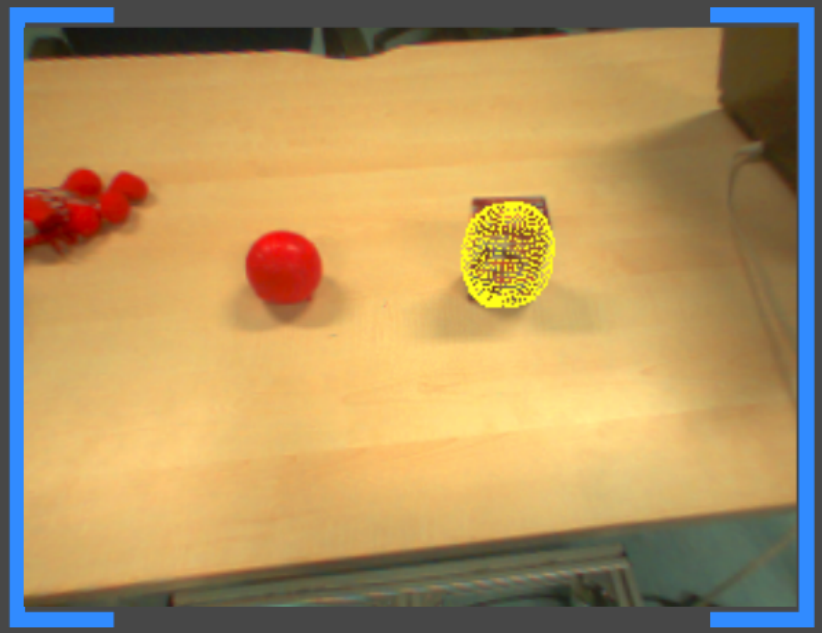
# Partial point cloud





3D points sampled
on the superquadric

From 3D points to 2D Pixels



# What about one-shot behaviour?

# Offline Pipeline

Point cloud file

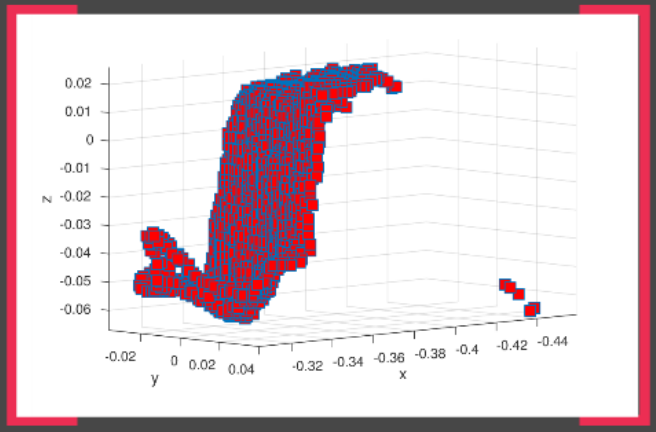
Filtering or not?

superquadric computation

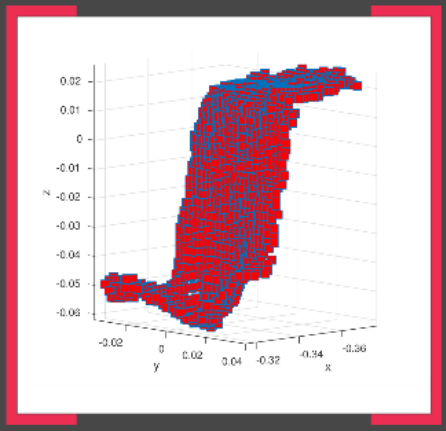
Code

# Why filtering?

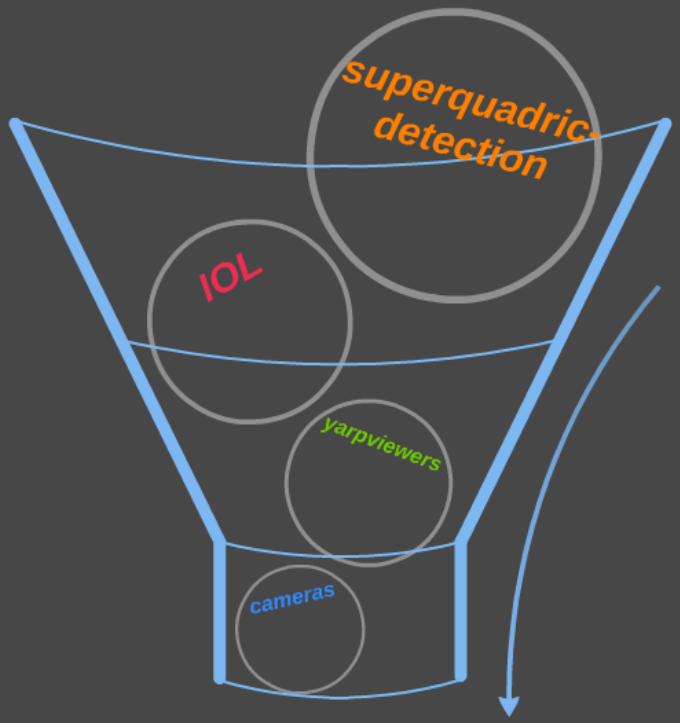
Before ...

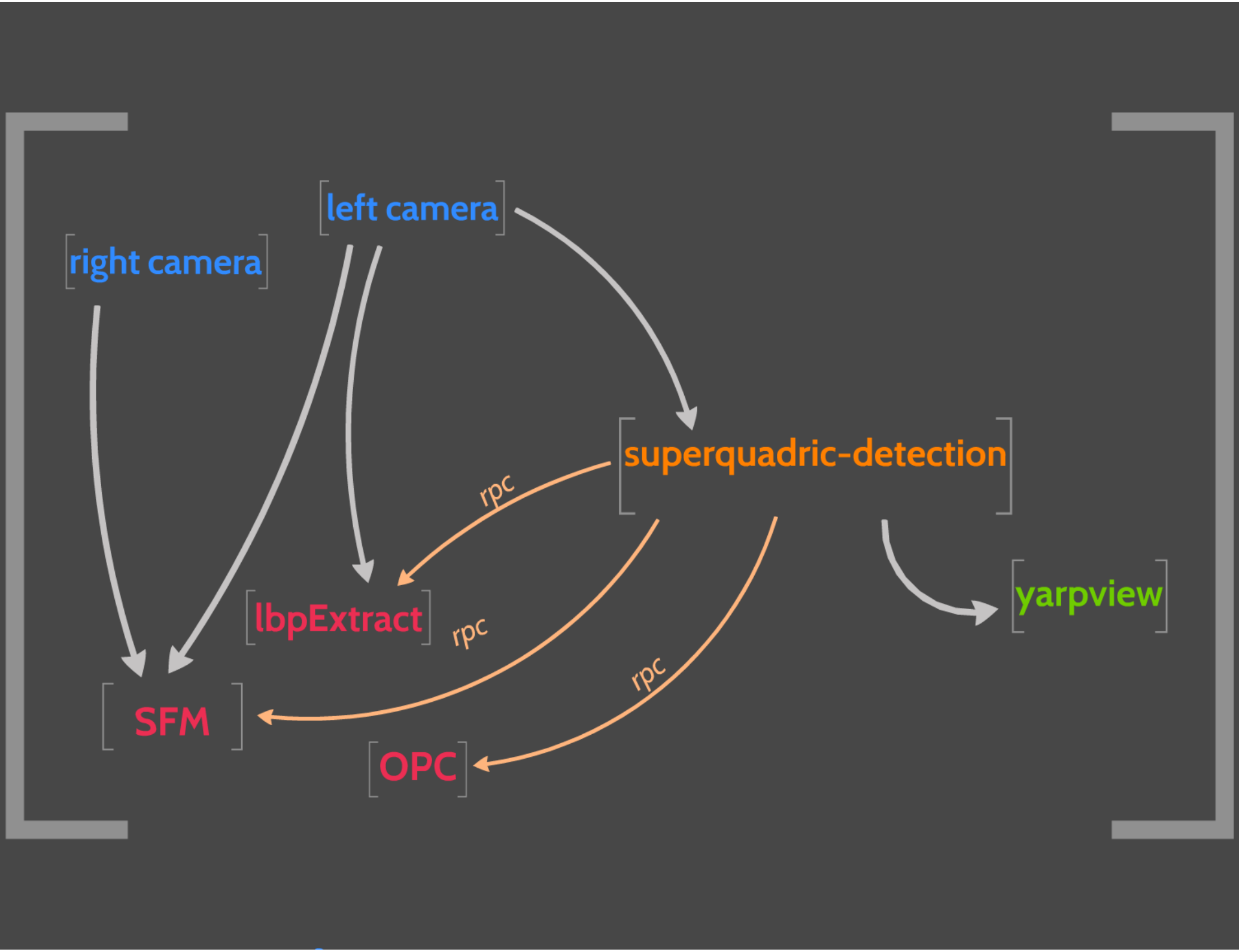



... after!



# How to combine everything?




left camera
right camera
superquadric-detection
rpc
lbpExtract
rpc
yarpview
rpc
SFM
OPC

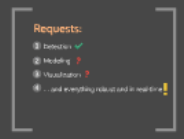
# RPC Thrift services

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


## Code

### Pipeline

### Dependencies


### Connections

Giulia Vezzani - 13/05/2016

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

*https://github.com/giuliavezzani/superquadric-detection*

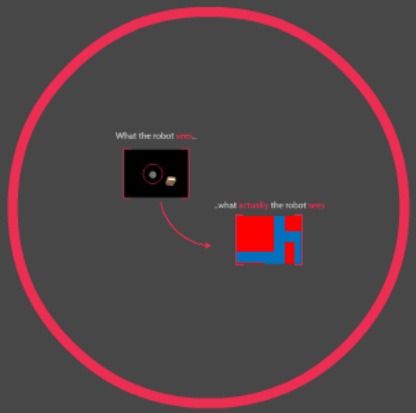
# ... What happens next?

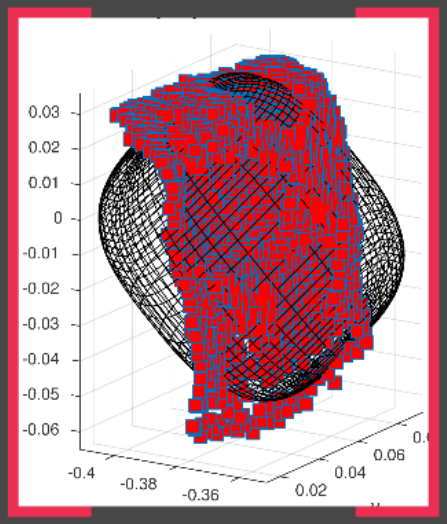
# Noteworthy ...

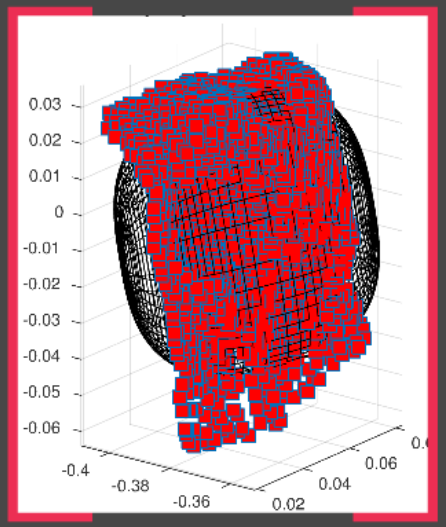
## Wrong solution (at first glance)


## Possible solutions


## Noisy point clouds

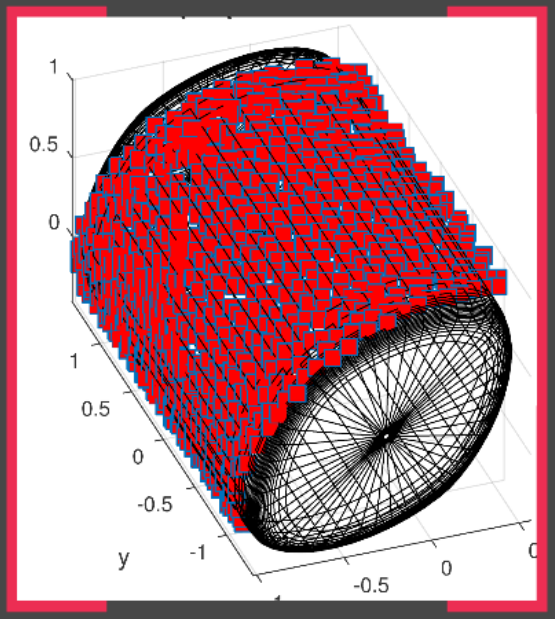




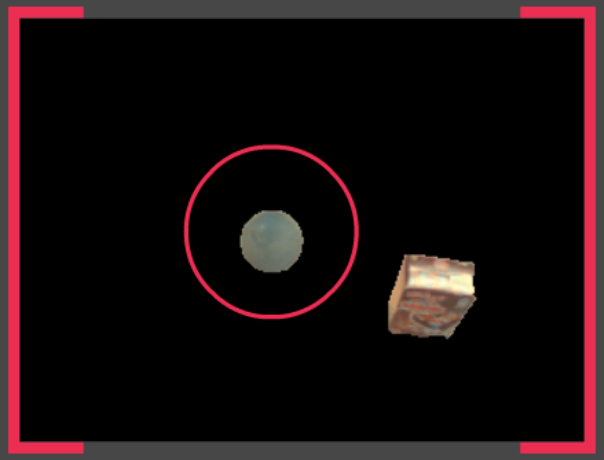




**Same cost function value!**

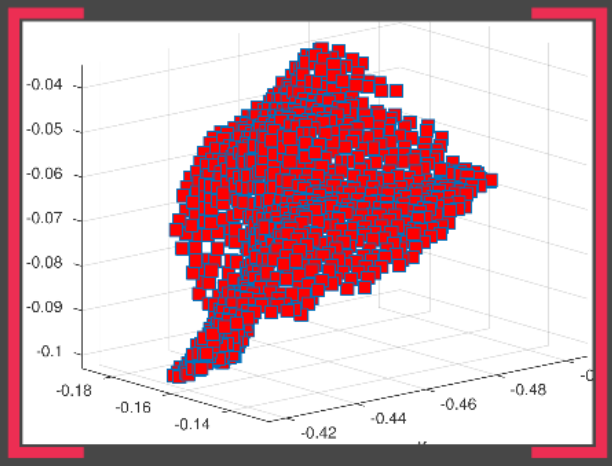
Partial non-noisy clouds:



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

$$\min_{\lambda} \forall(\lambda)$$

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

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

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

# 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!