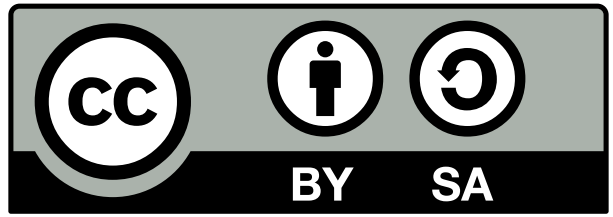
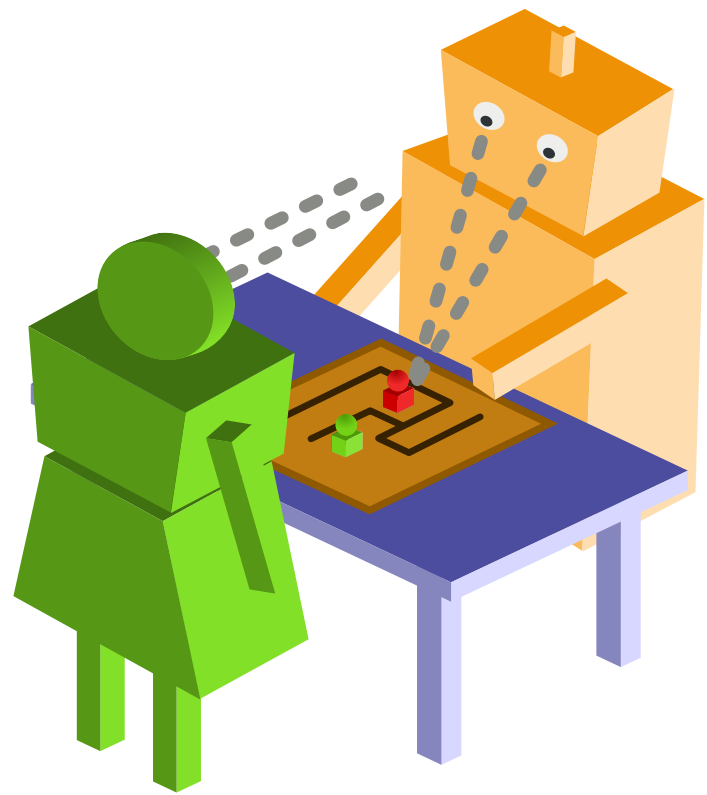


You can download the sources of this presentation here:
github.com/severin-lemaignan/lecture-rgbd-cameras-hri

WITH PLYMOUTH UNIVERSITY

RGB-D Cameras for Human-Robot Interaction

March 16, 2016



Séverin Lemaignan

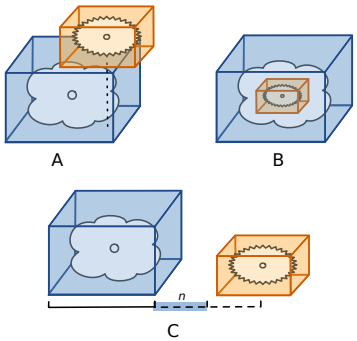
Centre for Neural Systems and Robotics
Plymouth University

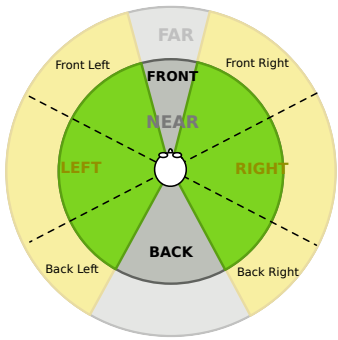
OVERVIEW

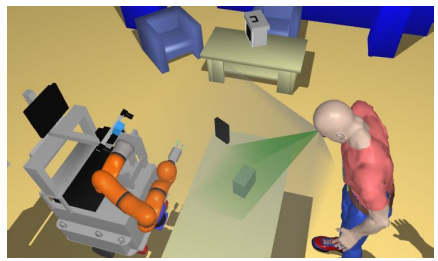
1. 3D Perception for HRI
2. RGB-D cameras
3. Algorithms for point clouds
4. Skeleton Tracking

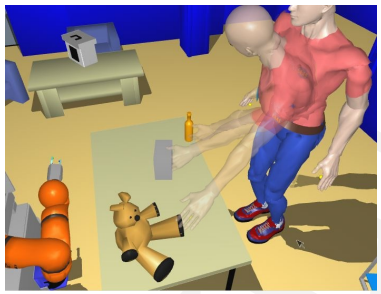
3D PERCEPTION FOR HRI

SITUATION ASSESSMENT





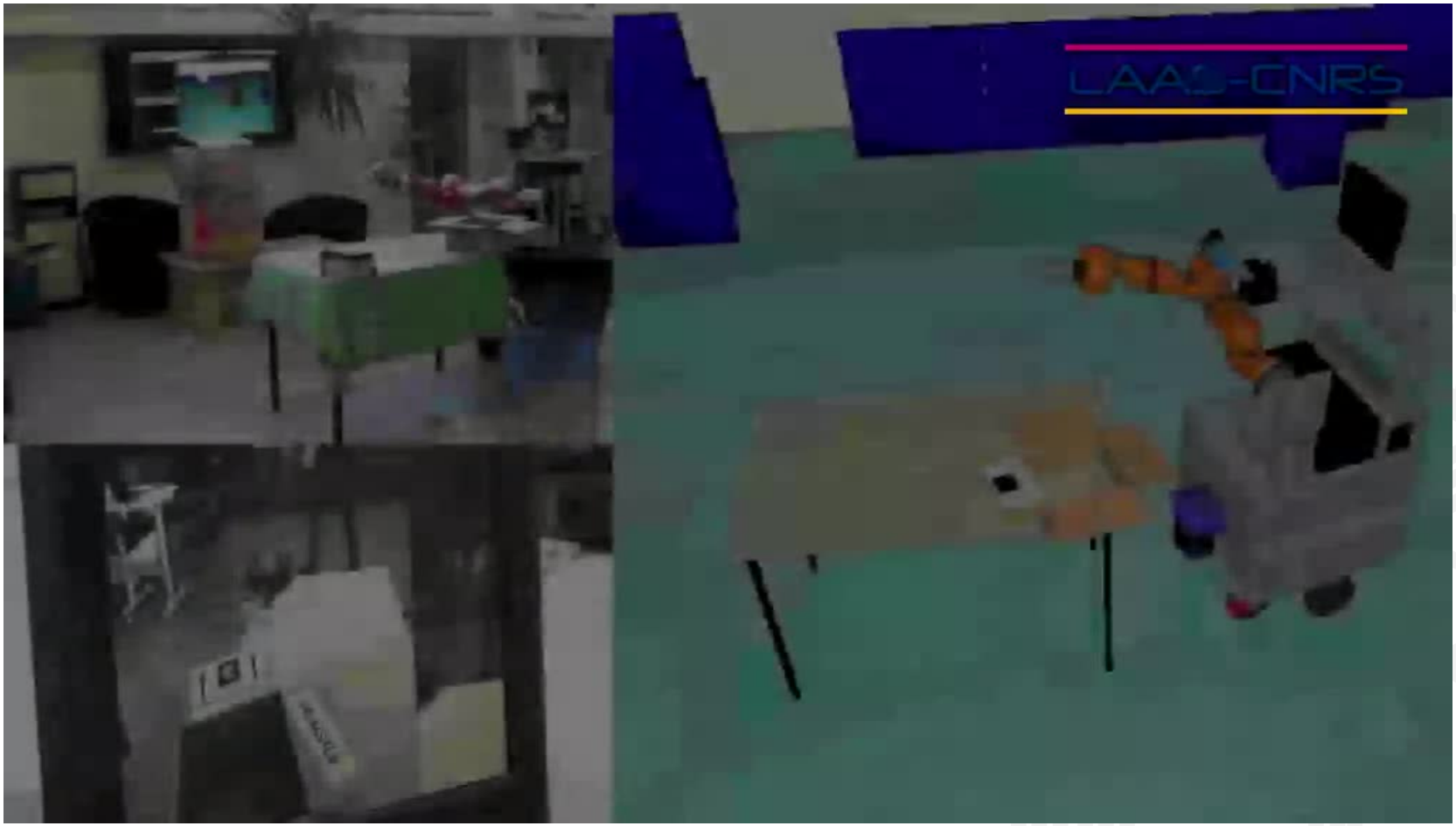




WHY NOT 2D?

3D interpretation of a 2D scene is hard!

Resorts to fiducial markers

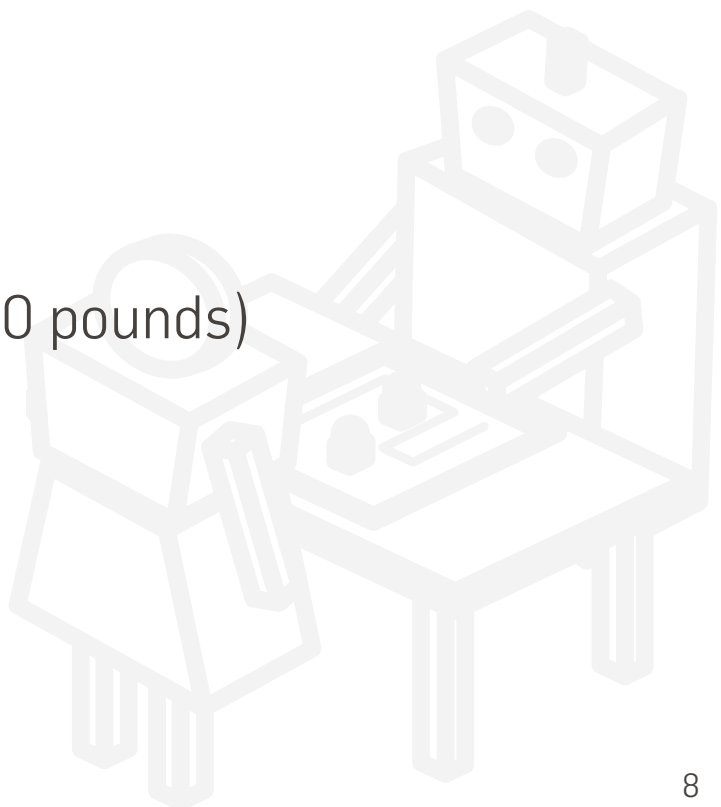


PERCEIVING THE HUMANS

Previously,...

- face detection
- blob detection
- motion capture (£10K+)

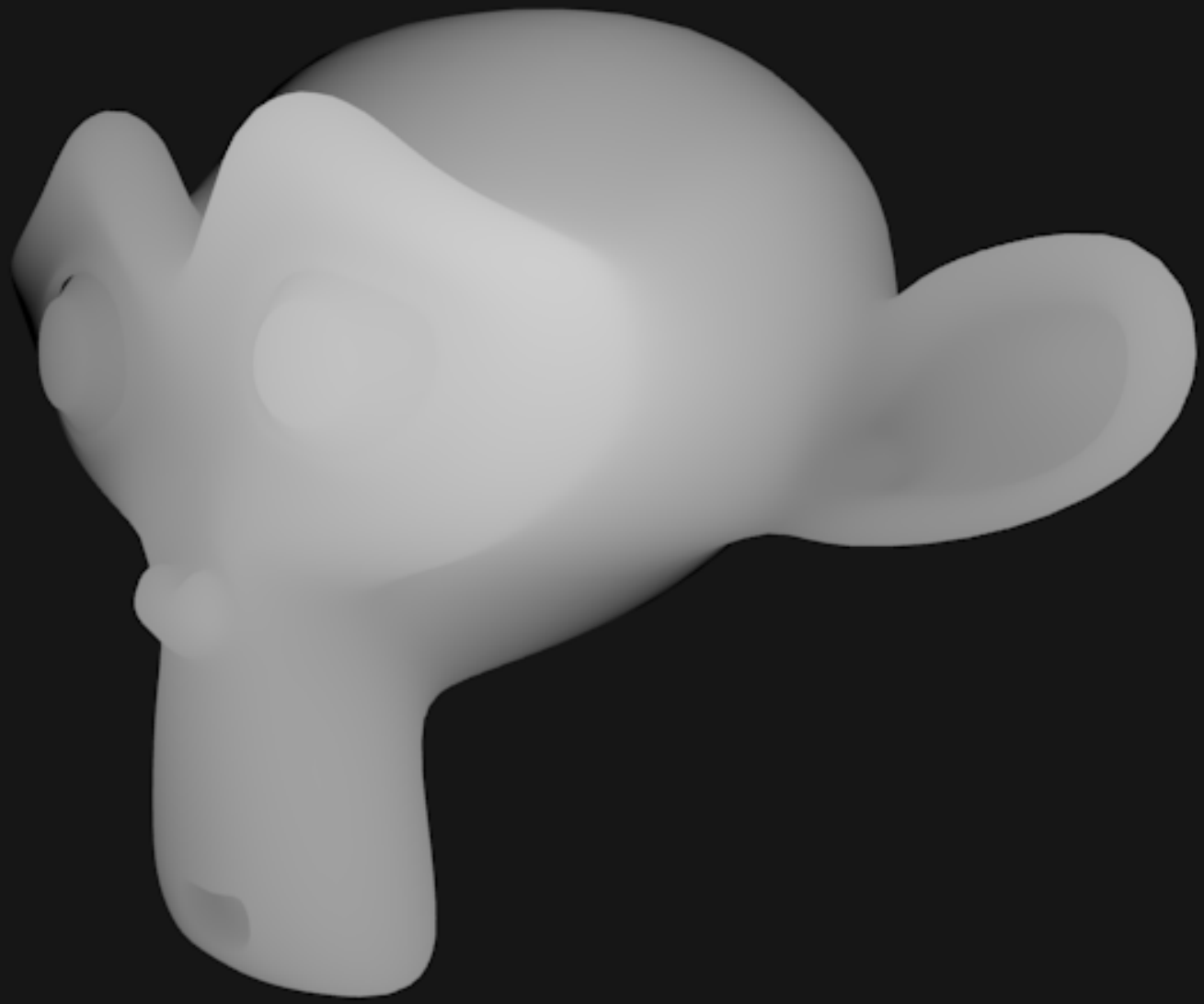
And then, one morning... the Kinect! (≈£120 pounds)



RGB-D CAMERAS

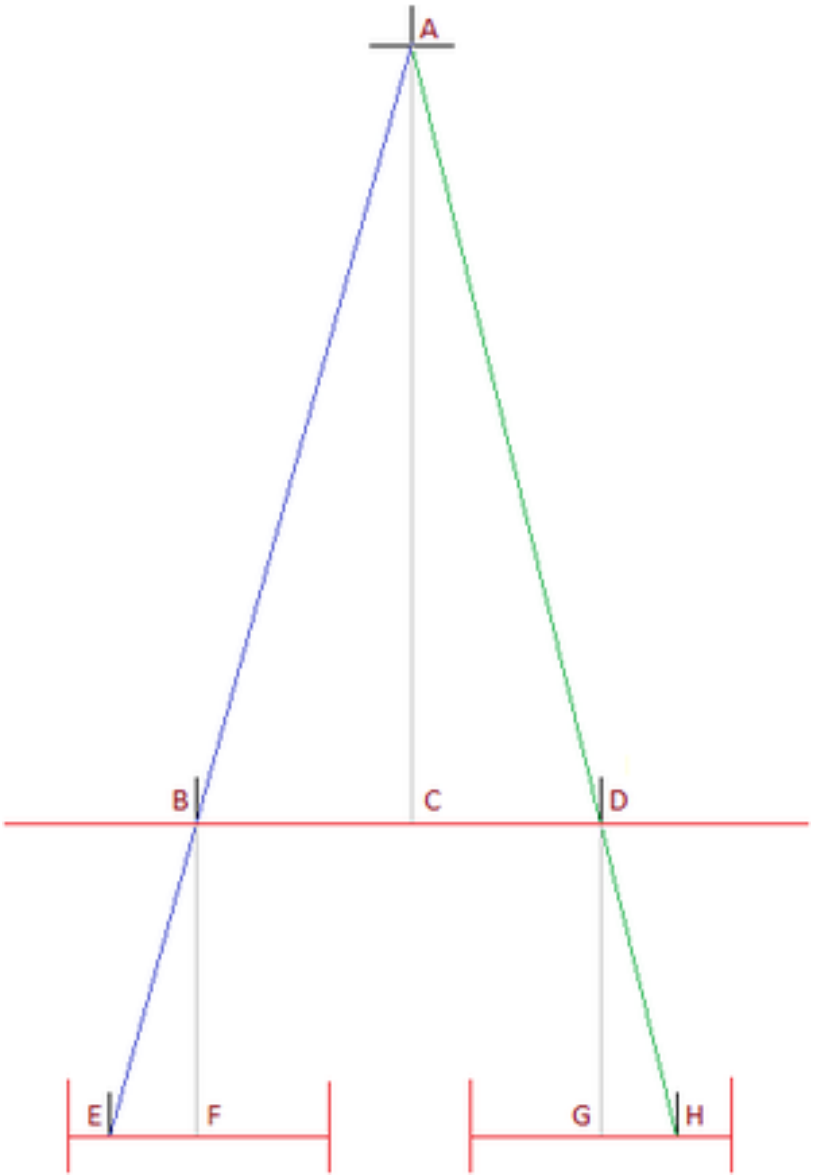
RGB-D CAMERAS: MANY TECHNOLOGIES

- Stereo-vision
- Structured light
- Speckle decorrelation
- Time-of-Flight
- (...several other like coded-aperture)




Elphel NC353L
/* source inside */
hama

STEREO-VISION: PRINCIPLE



$$d = EF + GH \quad (1)$$

$$= BF(\frac{EF}{BF} + \frac{GH}{BF}) \quad (2)$$

$$= BF(\frac{EF}{BF} + \frac{GH}{DG}) \quad (3)$$

$$= BF(\frac{BC + CD}{AC}) \quad (4)$$

$$= BF\frac{BD}{AC} \quad (5)$$

$$= \frac{k}{z}, \text{ where} \quad (6)$$

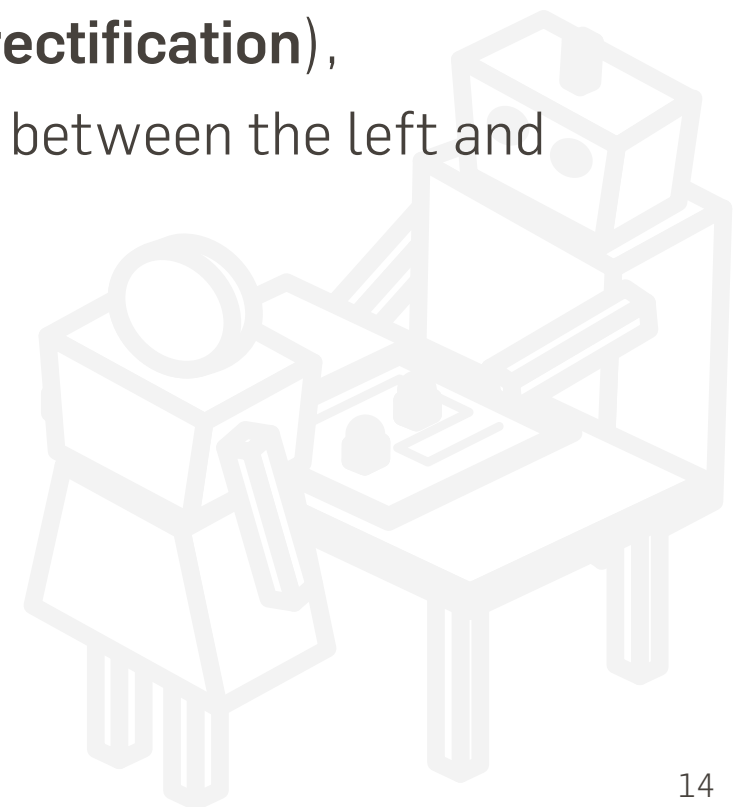
$$k = BD \times BF$$

$$z = AC$$

DISPARITY MAP

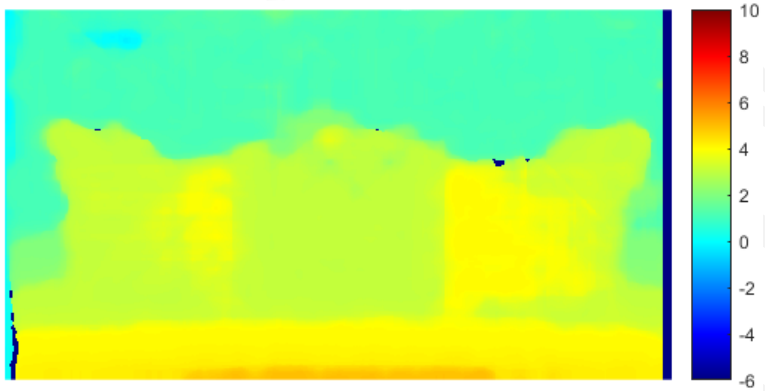
The disparity map is built by:

1. properly aligning the images (**image rectification**),
2. looking for the **disparity** of each pixel between the left and right images.



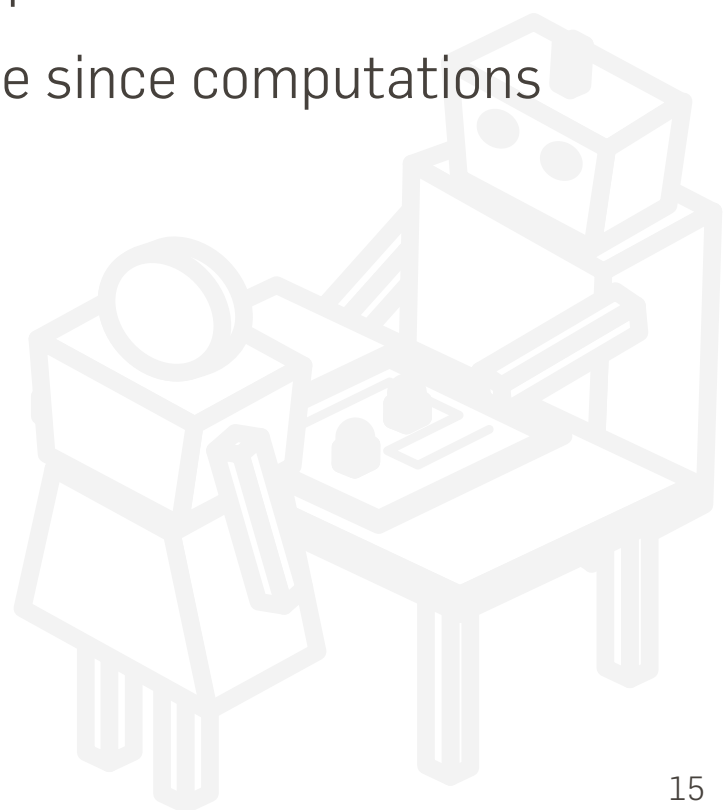
DISPARITY MAP

Pixel disparity: for each **pixel patch** on the left, by how many pixel the same patch is **shifted** on the right?

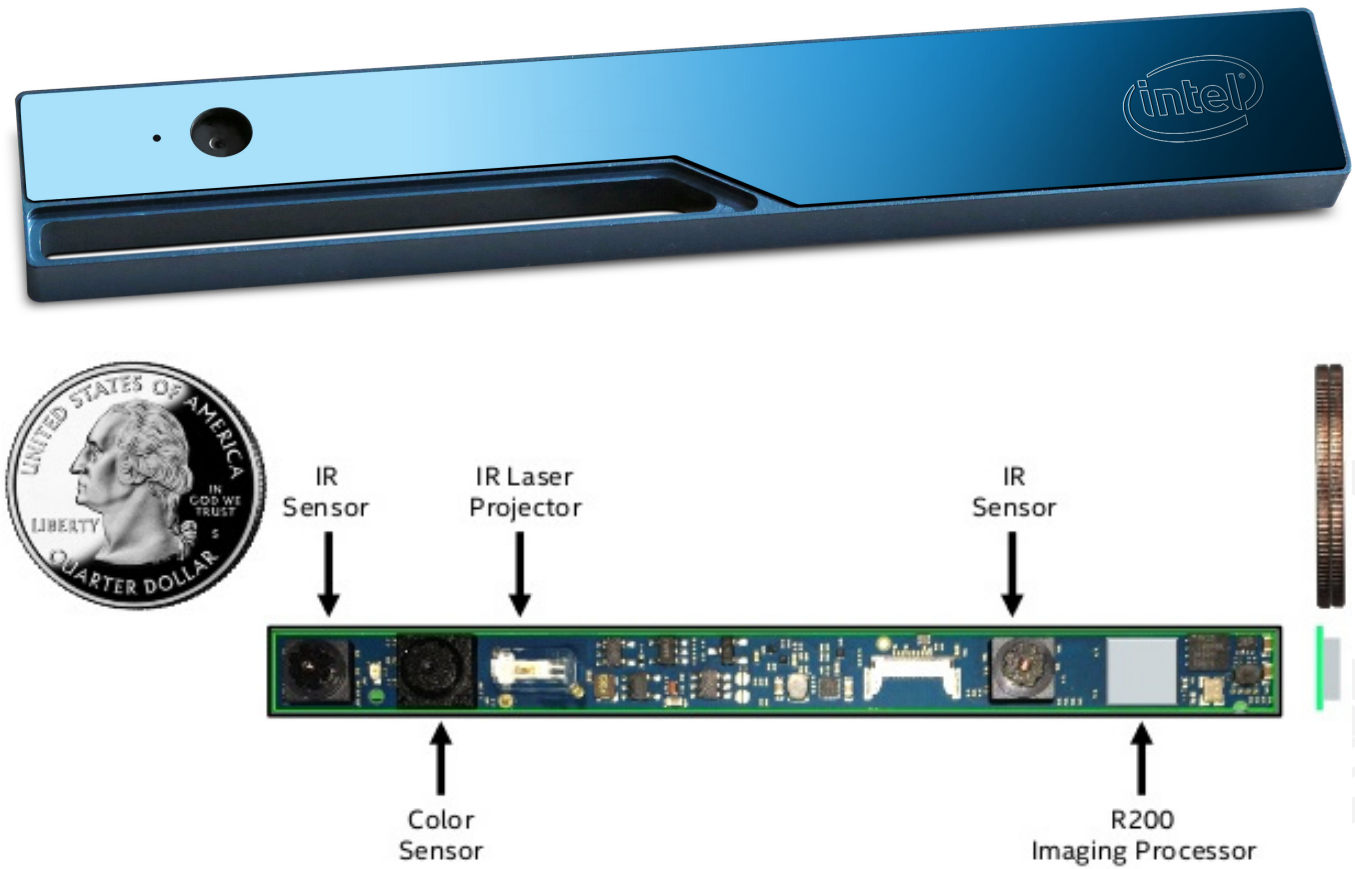


ISSUES WITH STEREOVISION

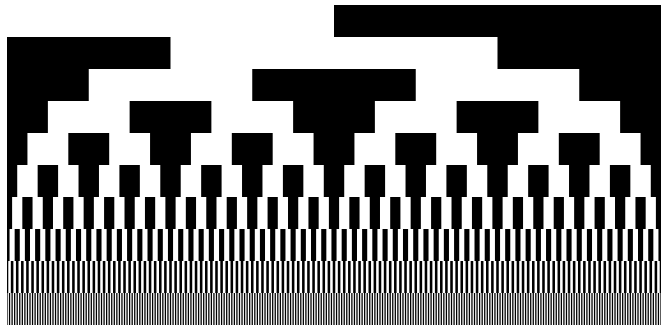
- Requires texture ⇒ active stereovision
- Computation intensive (less of an issue since computations are performed on-board)



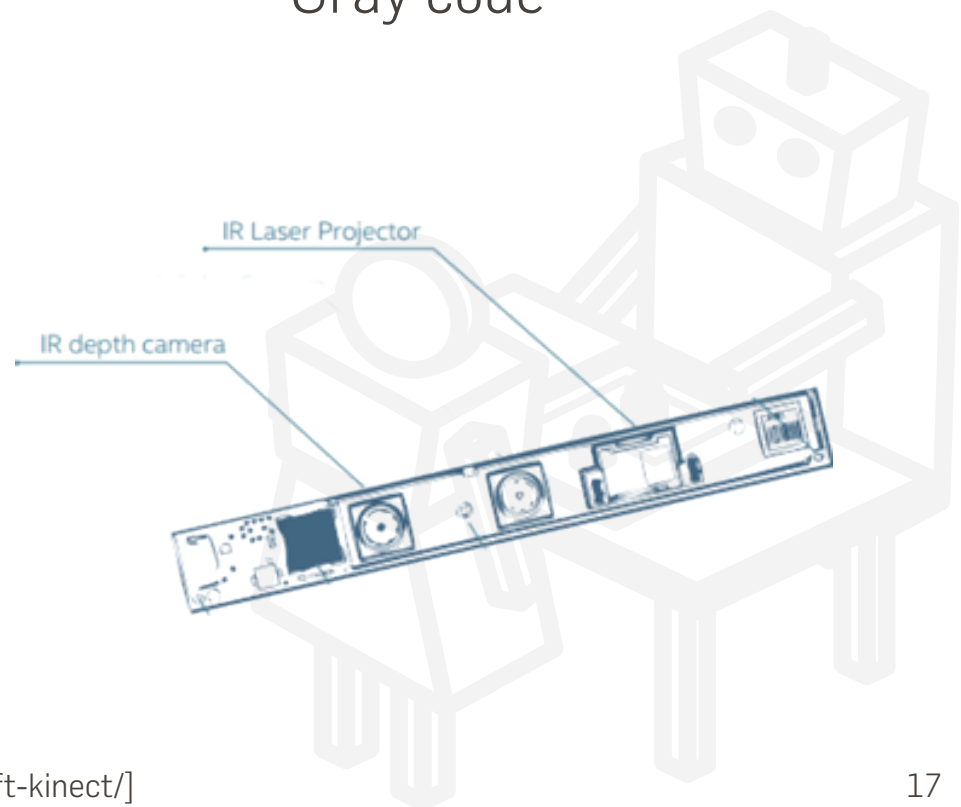
ACTIVE STEREO-VISION



STRUCTURED LIGHT



Gray code

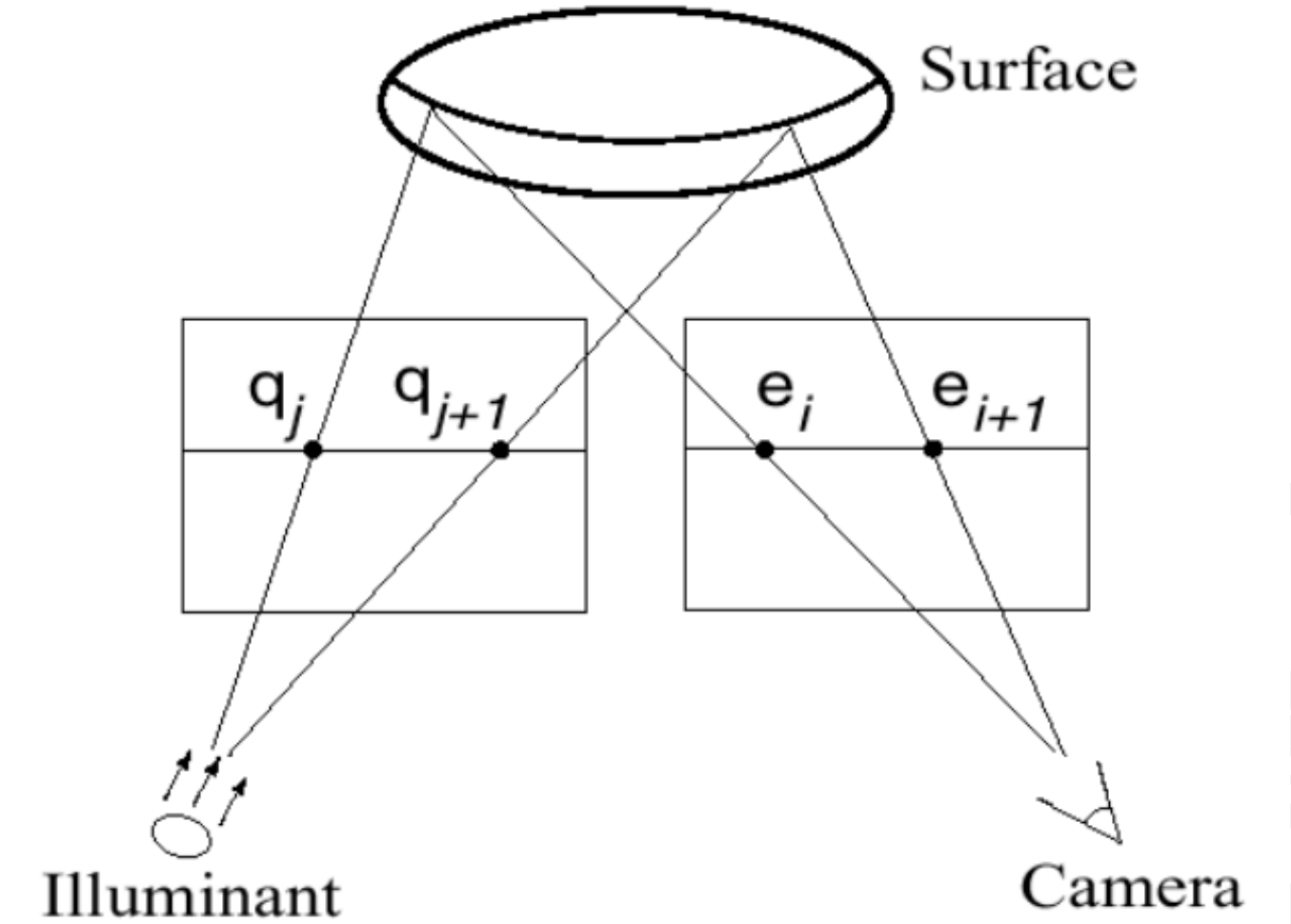


[Source: http://www.hackengineer.com/structured-light-vs-microsoft-kinect/]

STRUCTURED LIGHT

[Source: http://www.hackengineer.com/structured-light-vs-microsoft-kinect/]

STRUCTURED LIGHT



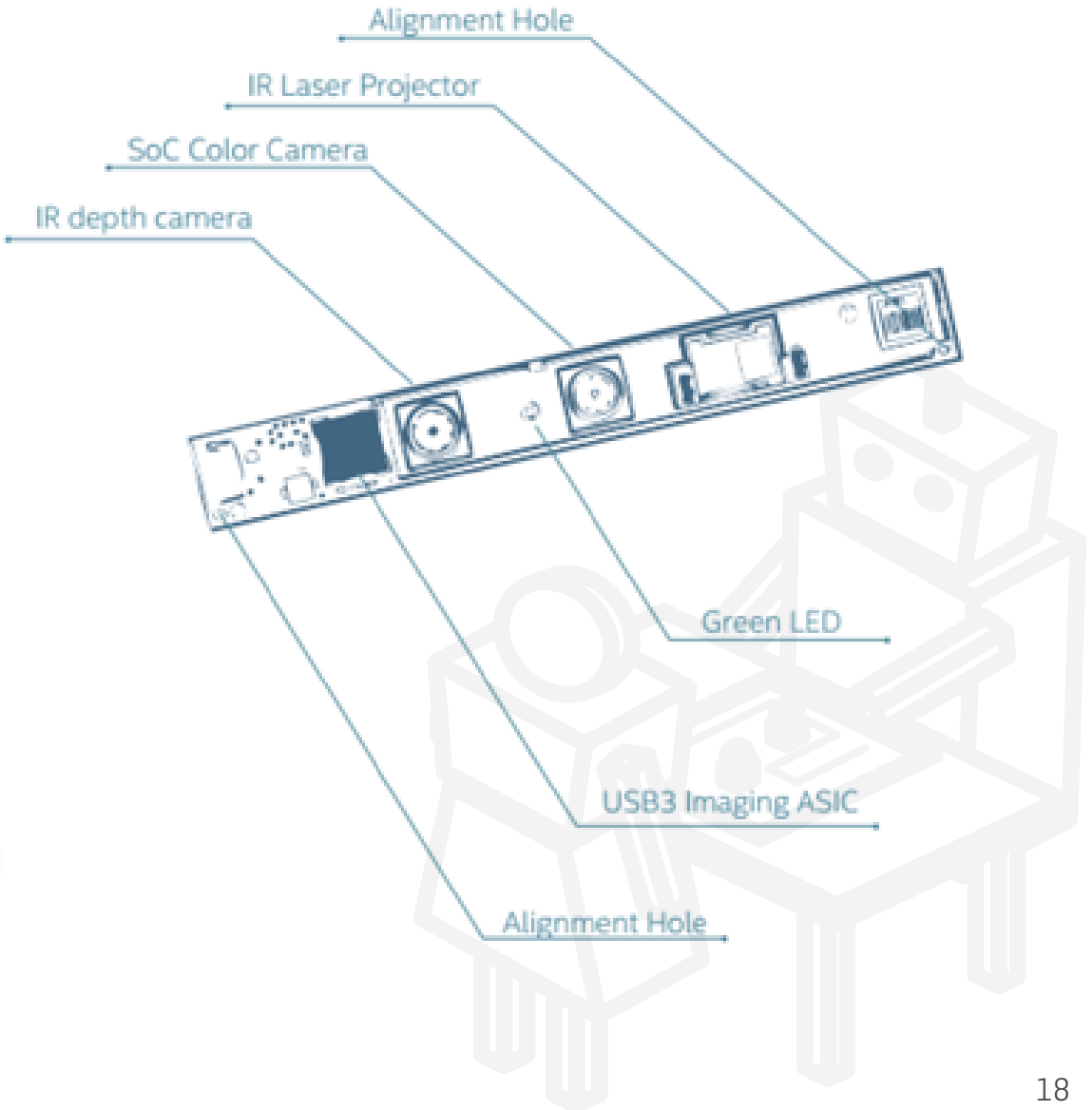
[Source: http://www.hackengineer.com/structured-light-vs-microsoft-kinect/]

STRUCTURED LIGHT

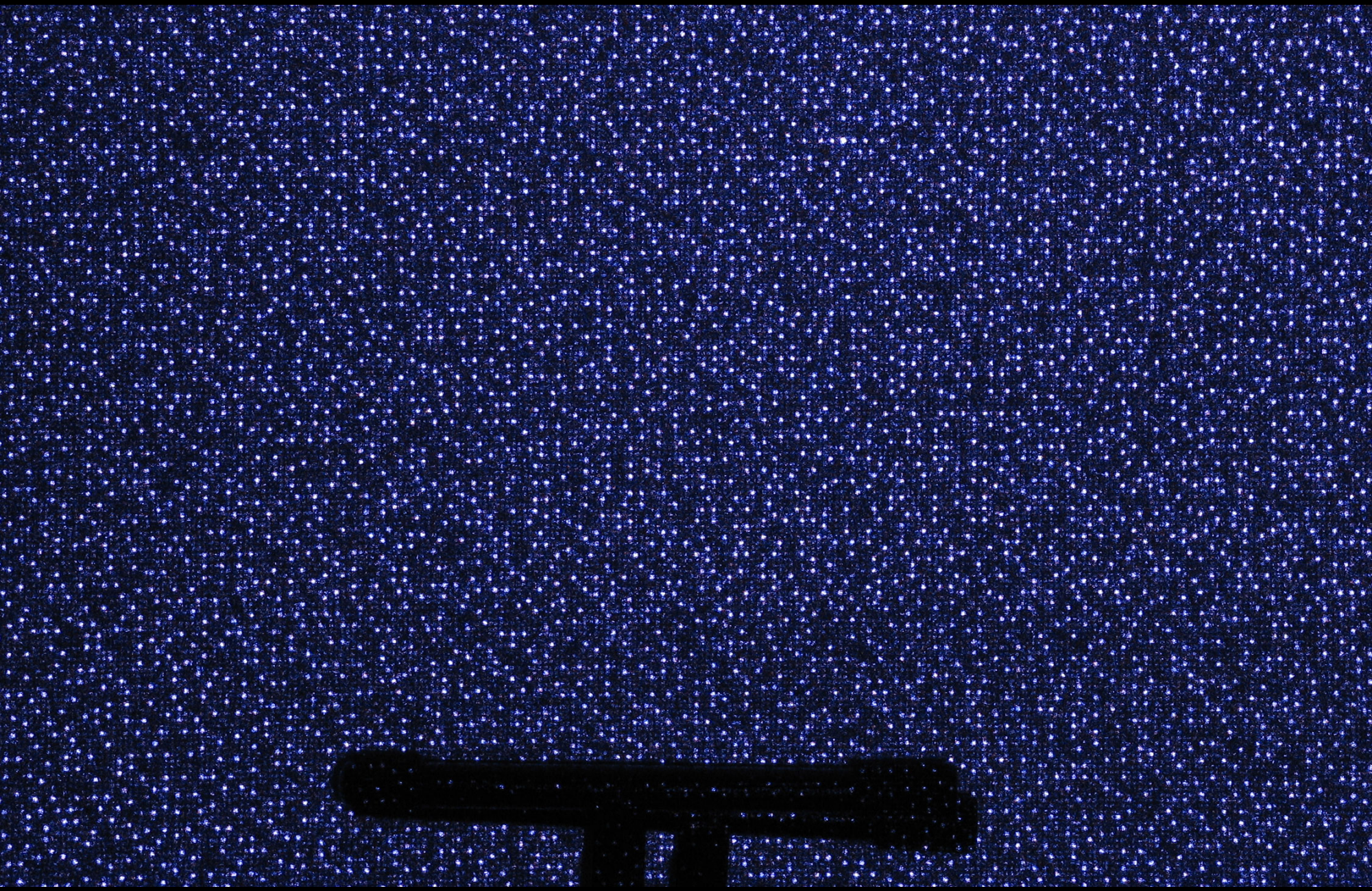
	Yellow	**Red**
Gray code	110100111	010000111
Projector col.	314	250
Camera col.	335	392
Disparity	335-314=21	392-250=142

[Source: http://www.hackengineer.com/structured-light-vs-microsoft-kinect/]

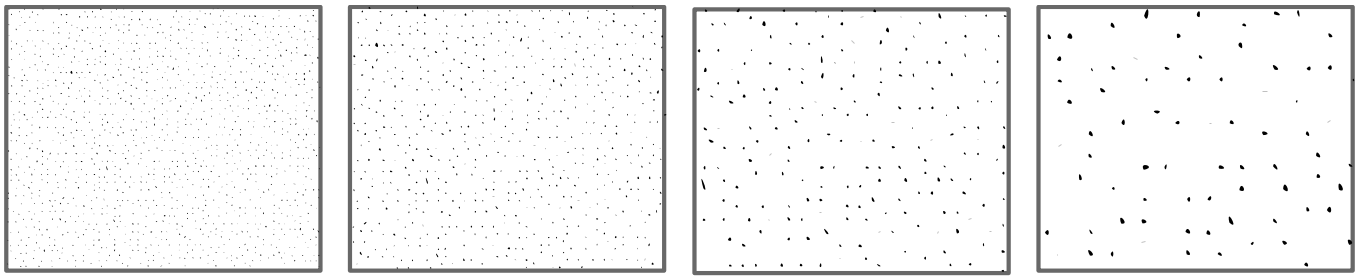
STRUCTURED LIGHT CAMERAS



SPECKLE DECORRELATION



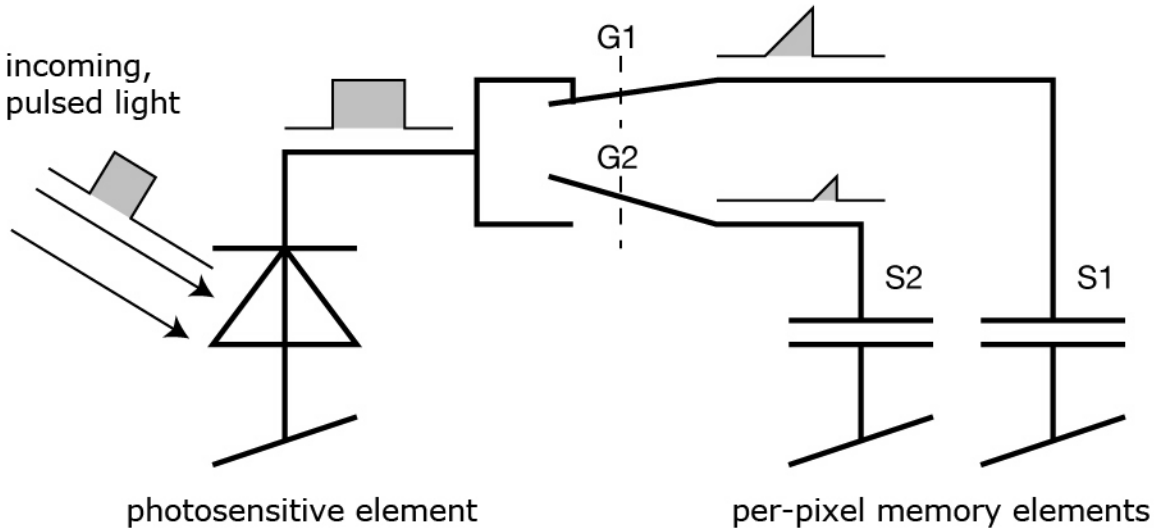
MULTISCALE PATTERN

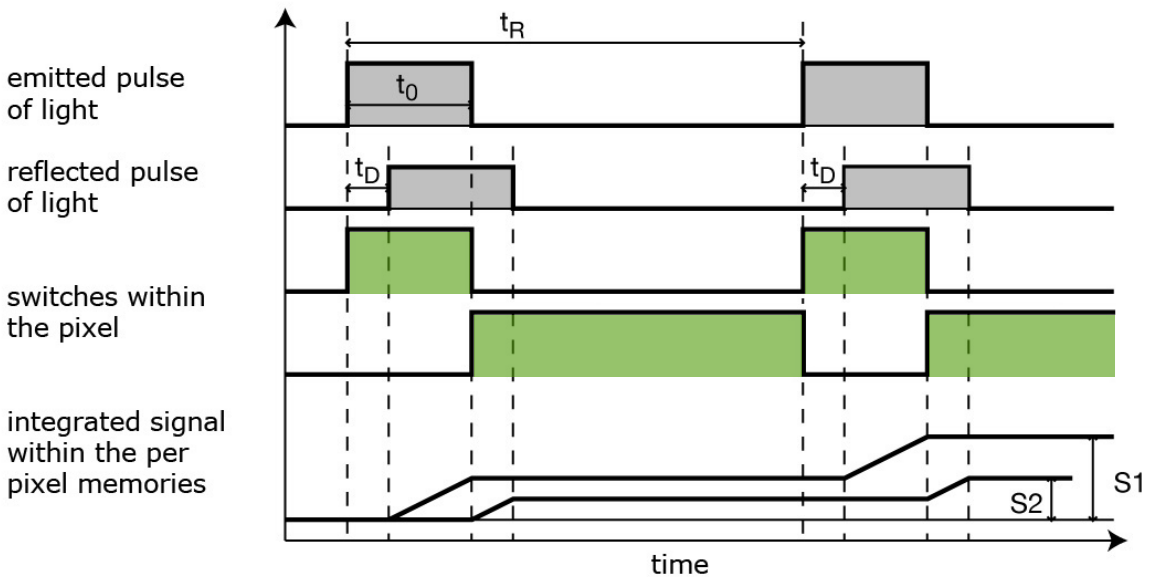


TIME-OF-FLIGHT CAMERAS

[Image source: Wikipedia]

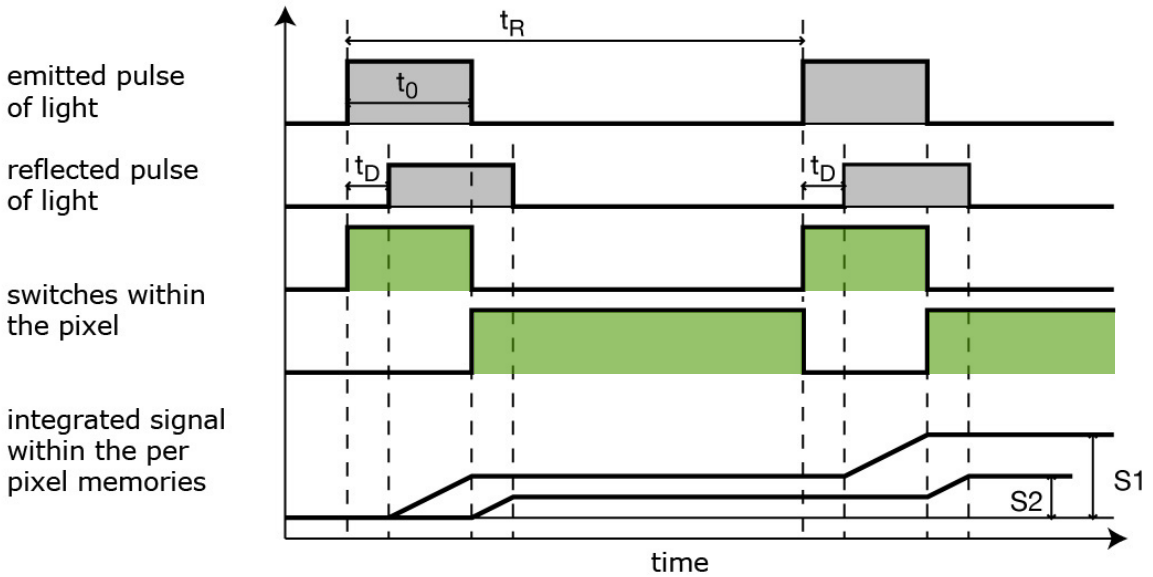
TIME-OF-FLIGHT CAMERAS





[Image source: Wikipedia]

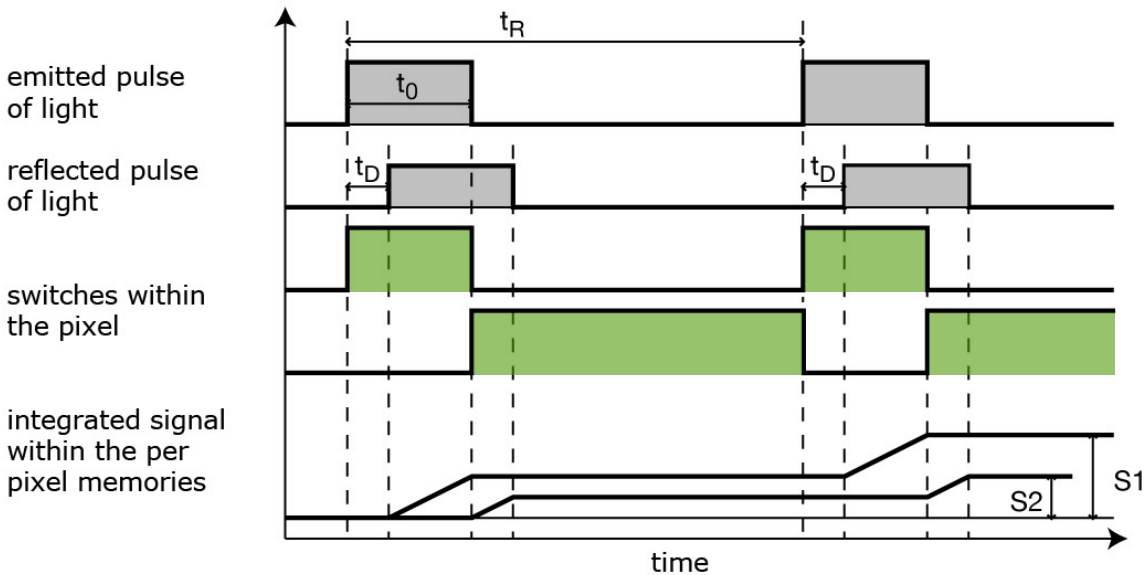
TIME-OF-FLIGHT CAMERAS



$$t_D = 2 \cdot \frac{D}{c} = 2 \cdot \frac{2.5m}{300000000 \frac{m}{s}} = 0.00000001666s = 16.66ns$$

[Image source: Wikipedia]

TIME-OF-FLIGHT CAMERAS



$$t_D = 2 \cdot \frac{D}{c} = 2 \cdot \frac{2.5m}{300000000\frac{m}{s}} = 0.00000001666s = 16.66ns$$

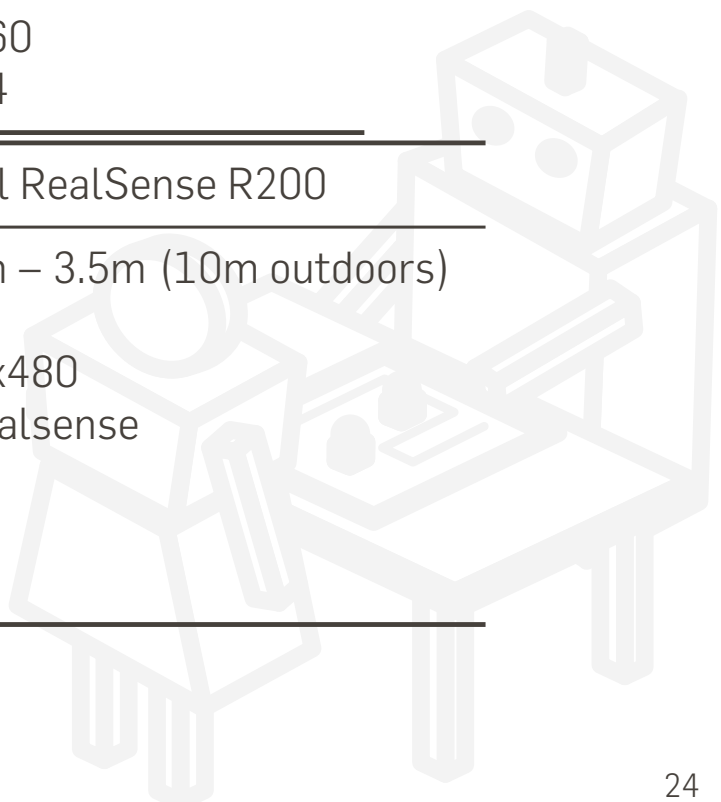
$$D = \frac{1}{2} \cdot c \cdot t_0 \cdot \frac{S2}{S1+S2}$$

[Image source: Wikipedia]

COMPARISON OF CAMERAS

	Kinect 360 (v1)	Kinect One (v2)
Range	0.6m – >5m	1.37m – 8m?
Price (new)	£230	£118
Depth resolution	640x480px, 11bits	512x424px, 13bits
Libraries	libfreenect/OpenNI1	libfreenect2
horizontal FoV	57	70
vertical FoV	43	60
Microphones?	4	4

	Intel RealSense SR300	Intel RealSense R200
Range	0.2m – 1.6m	0.6m – 3.5m (10m outdoors)
Price (new)	£66	£66
Depth resolution	640x480	640x480
Libraries	librealsense	librealsense
horizontal FoV	77	77
vertical FoV	43	47
Microphones?	2	0



POINT CLOUDS

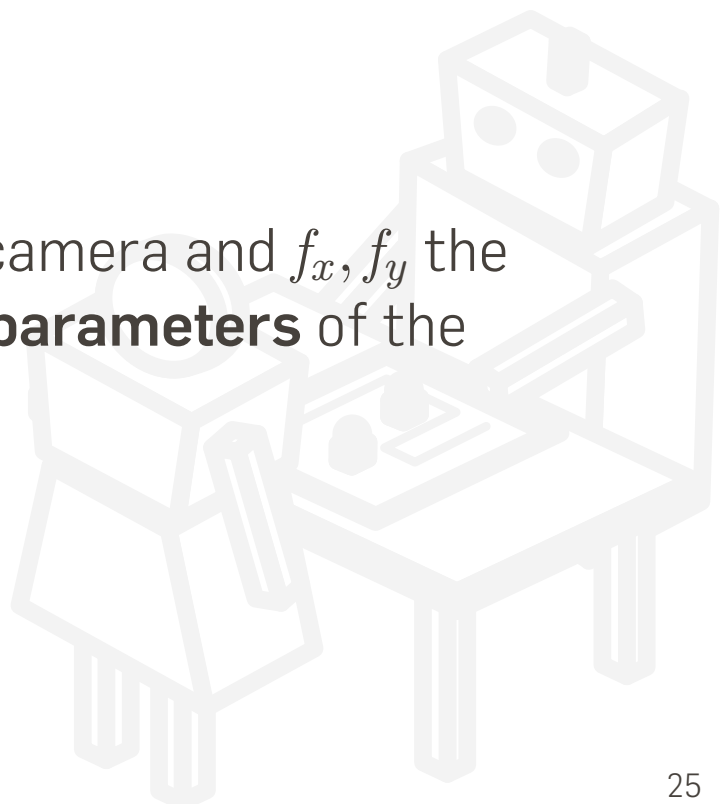
The (2D) depth image is usually converted to a (3D) point cloud:

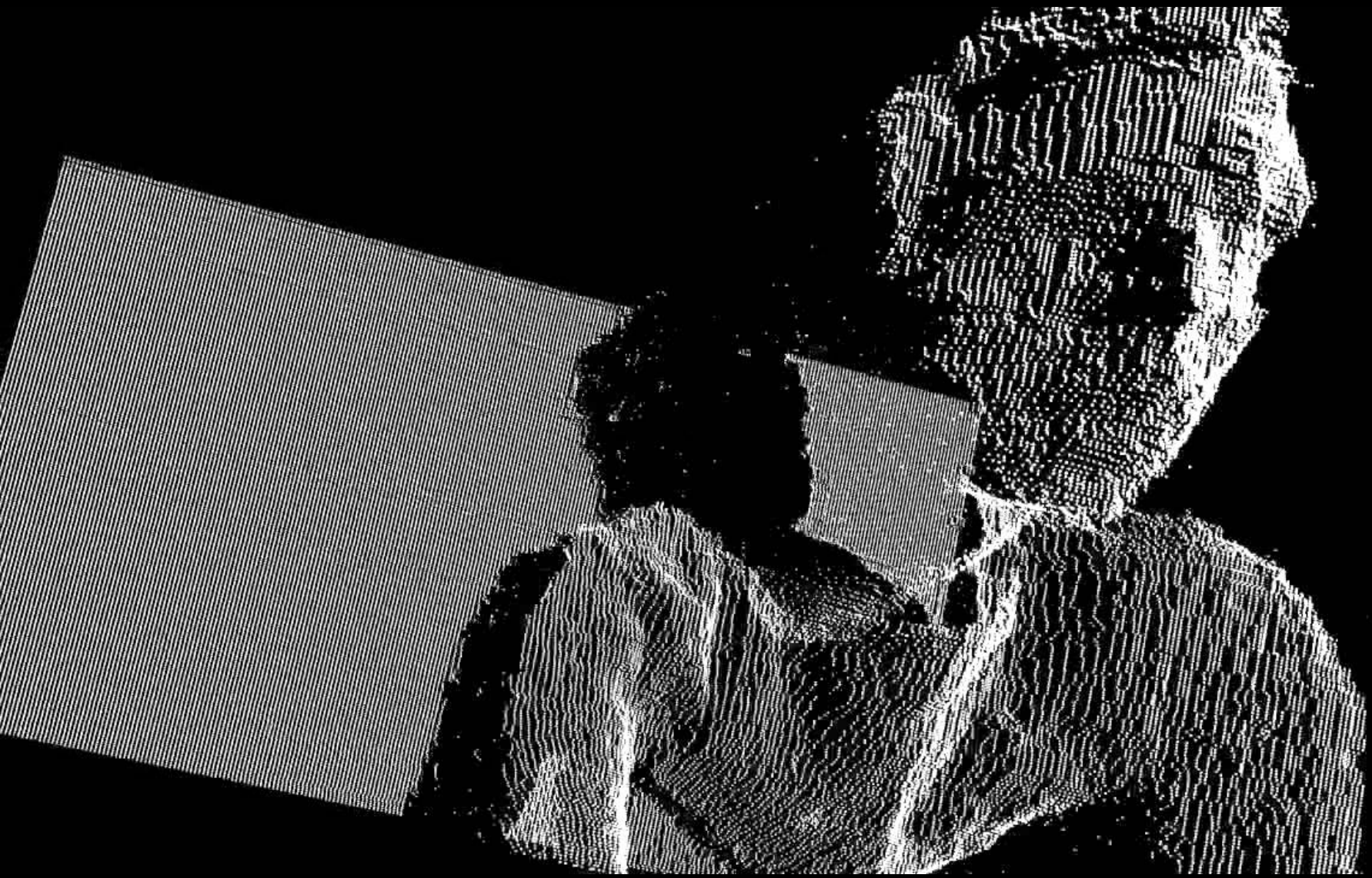
$$Z = depthmap(x, y)$$

$$X = \frac{x - c_x}{Z - f_x}$$

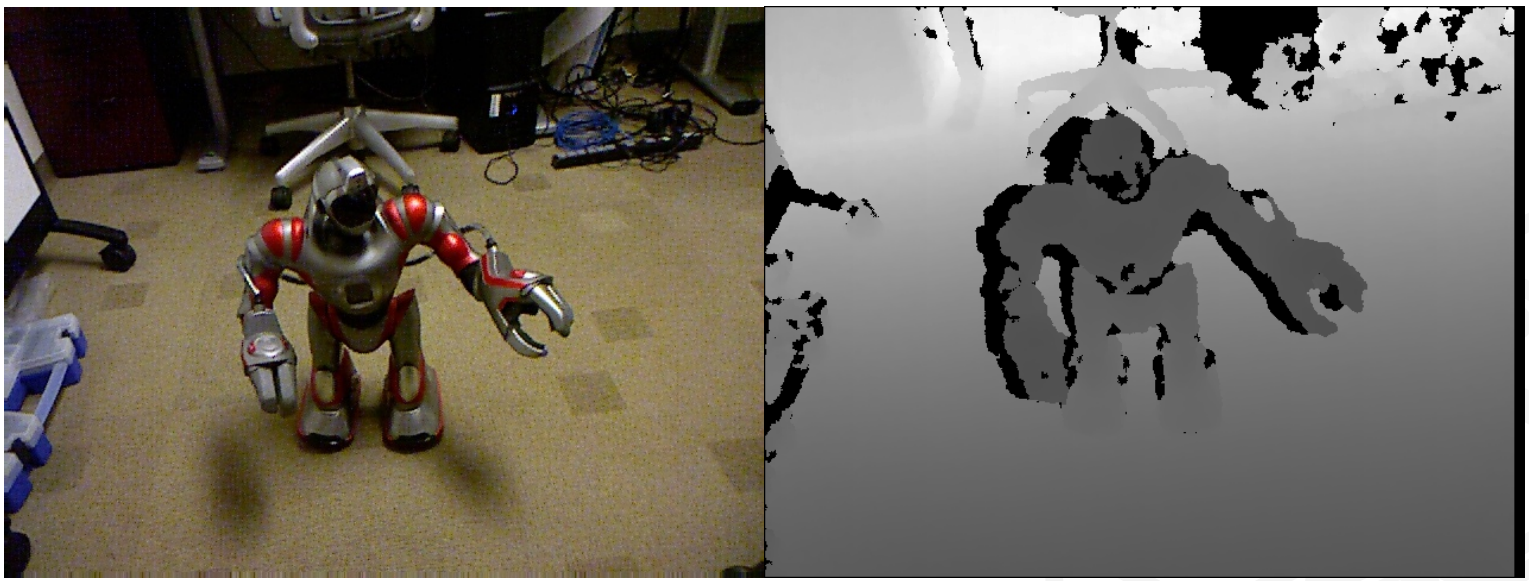
$$Y = \frac{x - c_y}{Z - f_y}$$

with c_x, c_y the optical center of the depth camera and f_x, f_y the focal lenght of the depth camera (**intrisic parameters** of the camera).

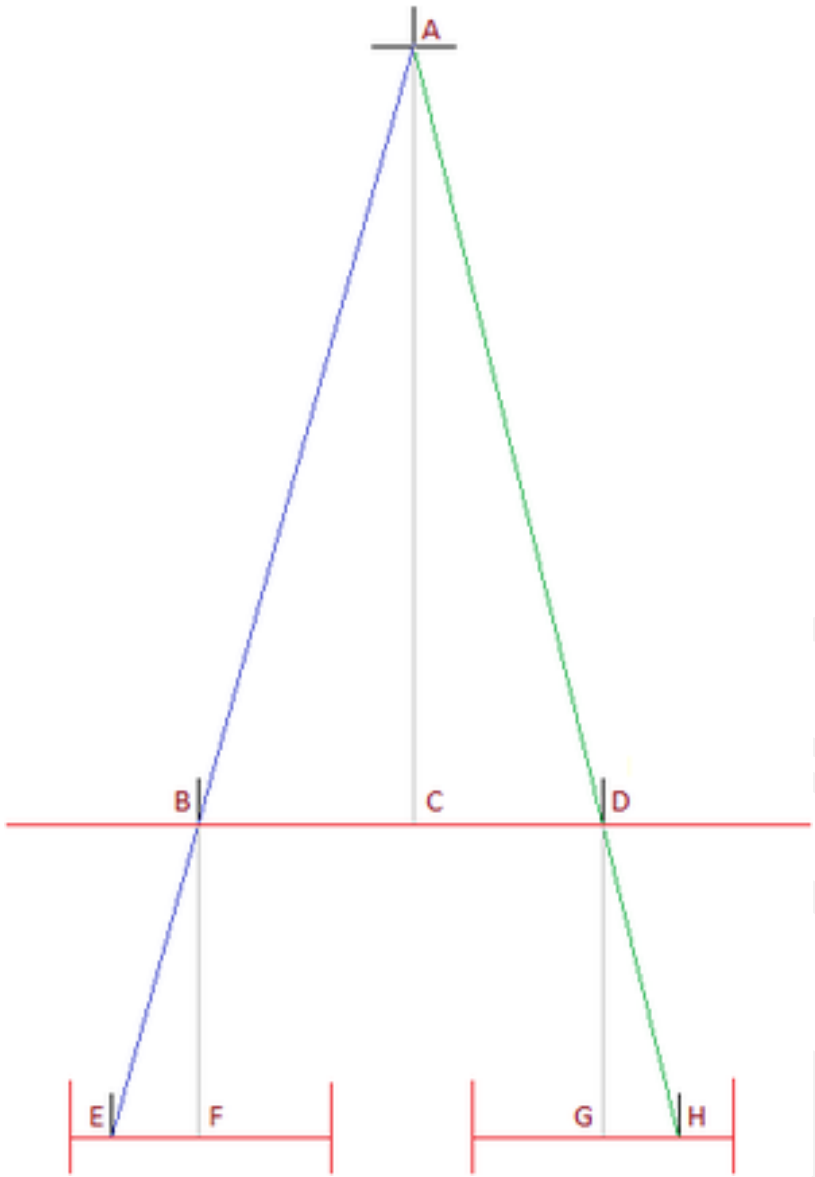




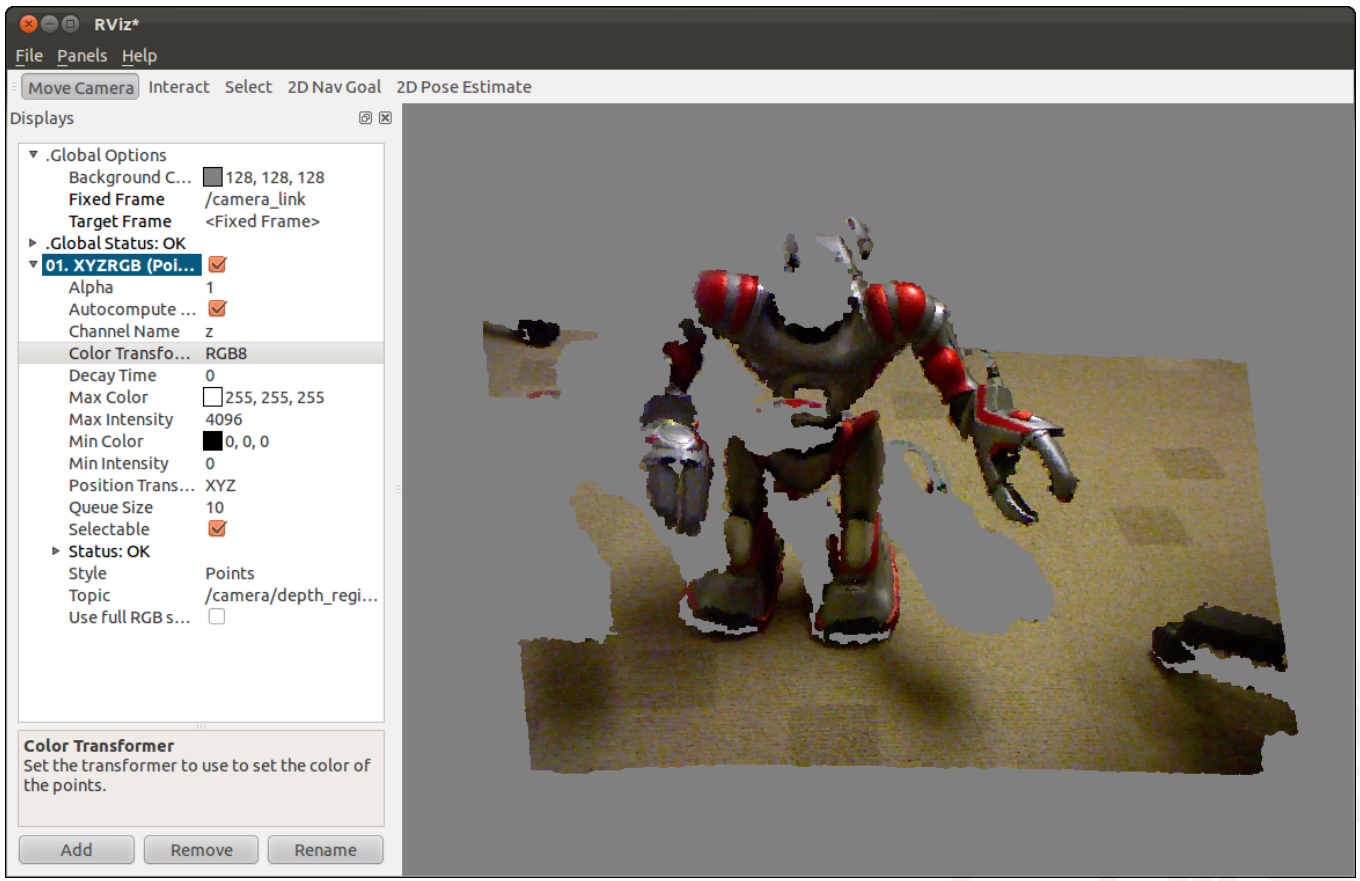
RGB-D REGISTRATION



RGB-D REGISTRATION



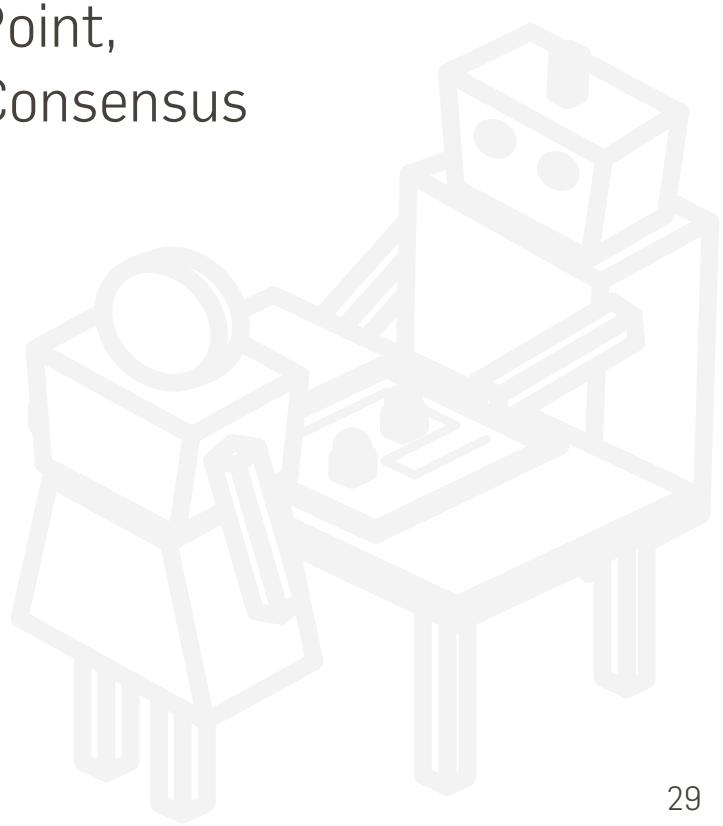
RGB-D REGISTRATION



ALGORITHMS FOR POINT CLOUDS

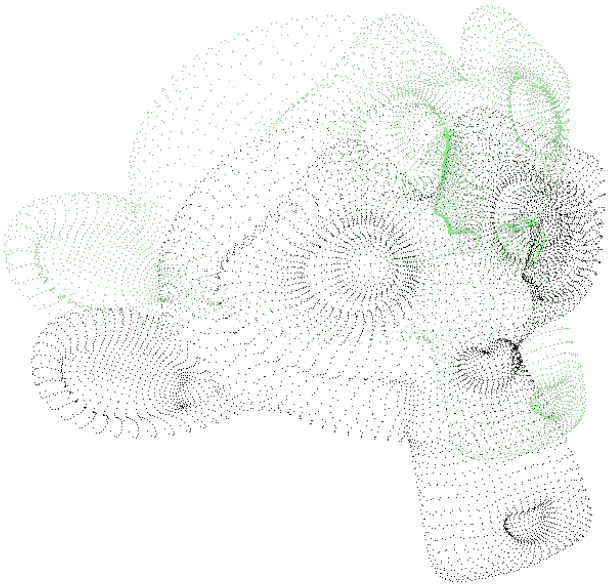
TWO FUNDAMENTAL ALGORITHMS

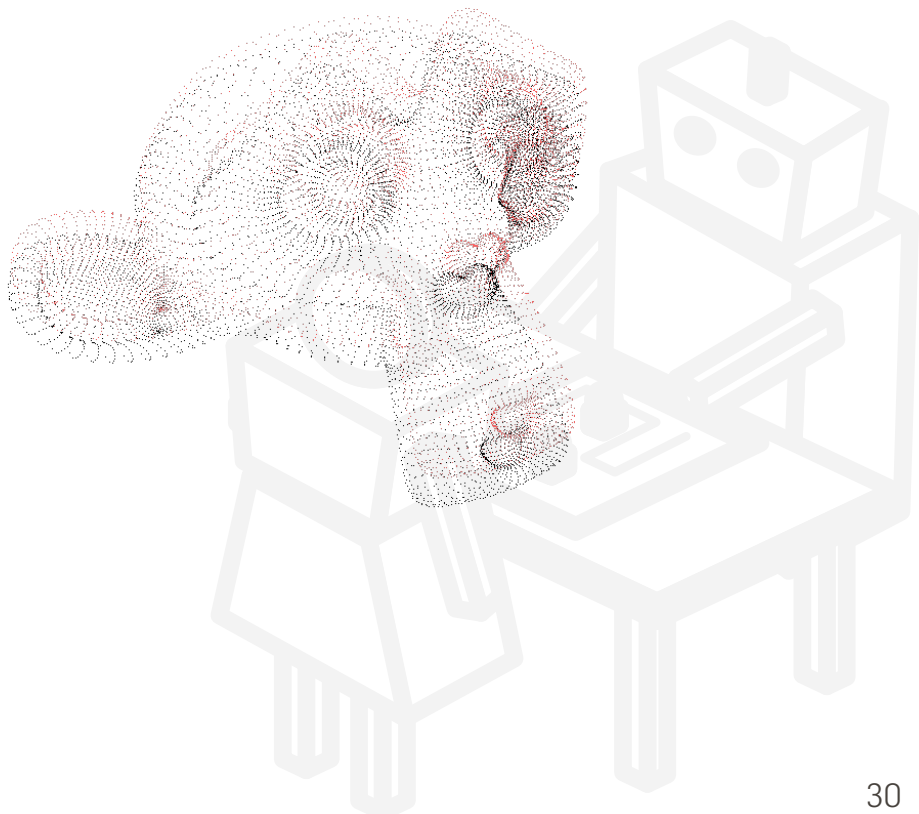
ICP: Iterative Closest Point,
RANSAC: RANdom SAmple Consensus



ITERATIVE CLOSEST POINT (ICP)

Purpose: align to sets of points – more precisely: **iteratively** estimate the **rigid transformation** between two point clouds.

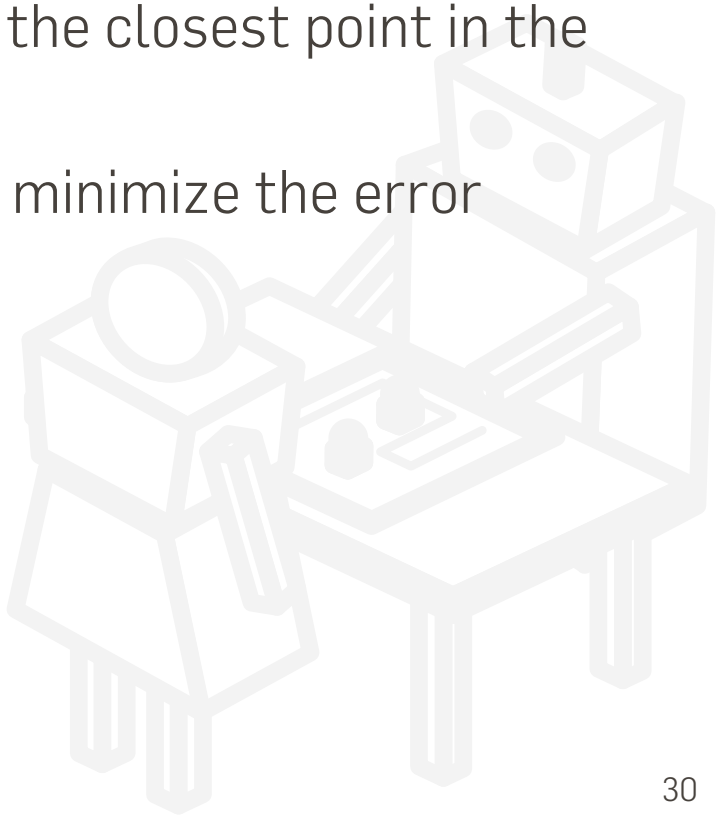




ITERATIVE CLOSEST POINT (ICP)

Purpose: align to sets of points – more precisely: **iteratively** estimate the **rigid transformation** between two point clouds.

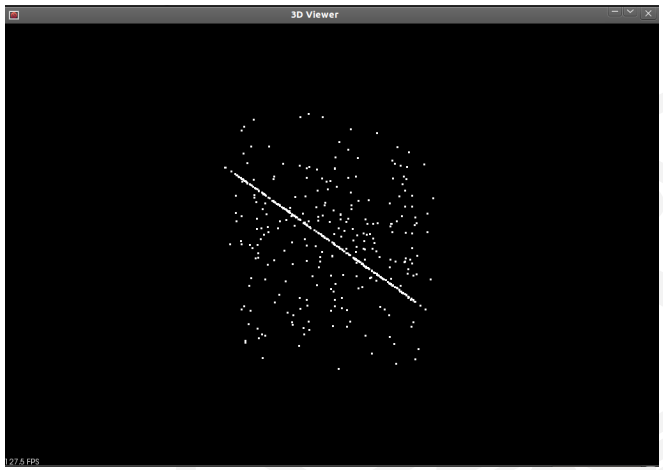
1. for each point in the source cloud, find the closest point in the target cloud
2. estimate the transformation, trying to minimize the error
3. transform the source cloud
4. iterate



RANDOM SAMPLE CONSENSUS (RANSAC)

Iterative algorithm that tries to **fit a parametric model** of a point distribution to observations. The result is an **estimation of the model's free parameters**.

1. take a random subset of the data; assume they are **inliers**; fit the model;
2. test the other observations against the fitted model;
3. if enough points are classified as inliers, return the model; otherwise go to 1.

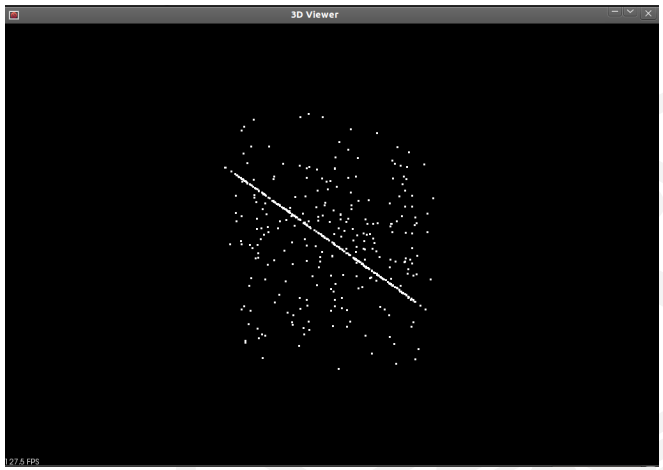


[Source: PCL documentation]

RANDOM SAMPLE CONSENSUS (RANSAC)

Iterative algorithm that tries to **fit a parametric model** of a point distribution to observations. The result is an **estimation of the model's free parameters**.

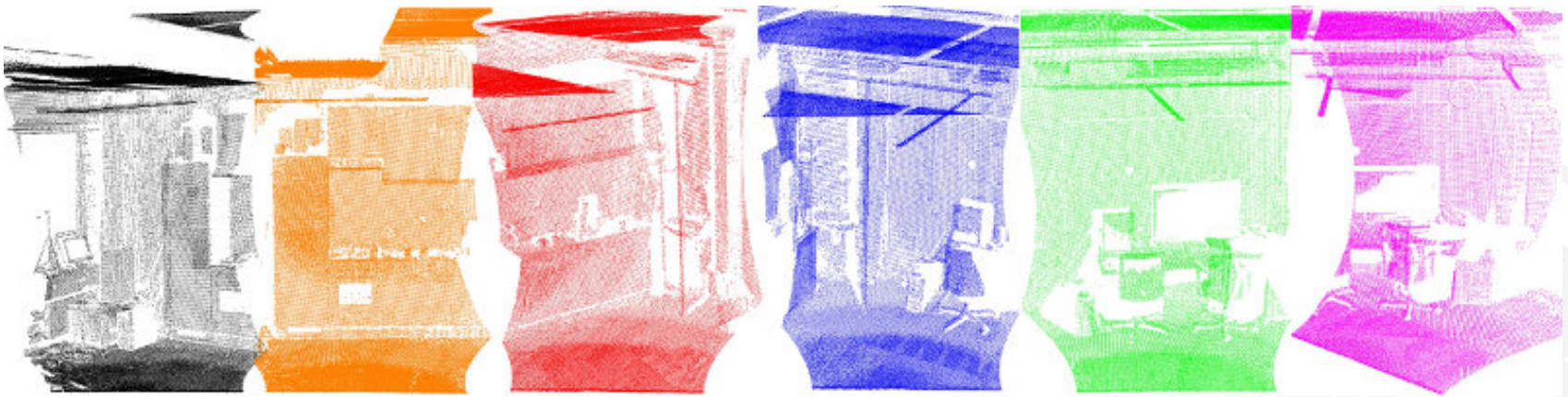
1. take a random subset of the data; assume they are **inliers**; fit the model;
2. test the other observations against the fitted model;
3. if enough points are classified as inliers, return the model; otherwise go to 1.



Robust to **outliers**, but no guarantee of optimality

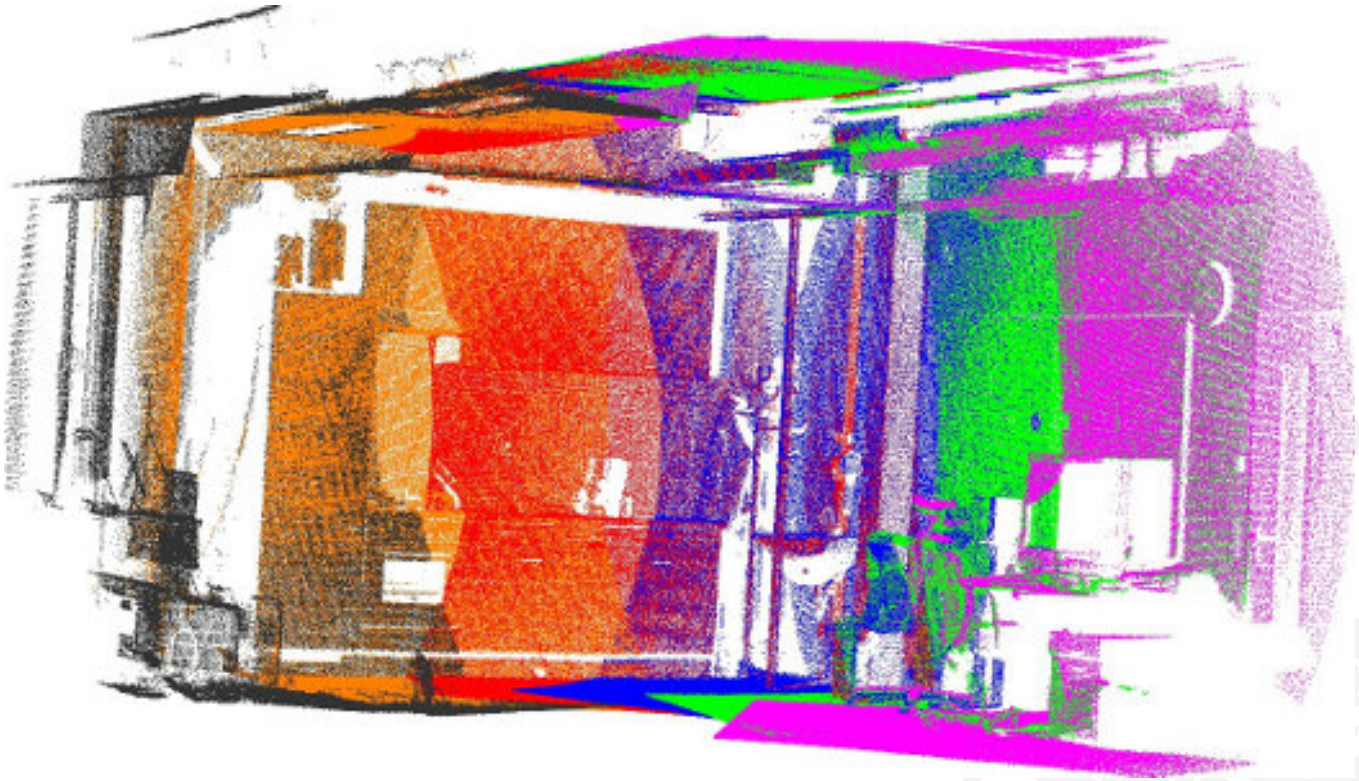
[Source: PCL documentation]

POINT CLOUDS REGISTRATION



[Source: PCL documentation]

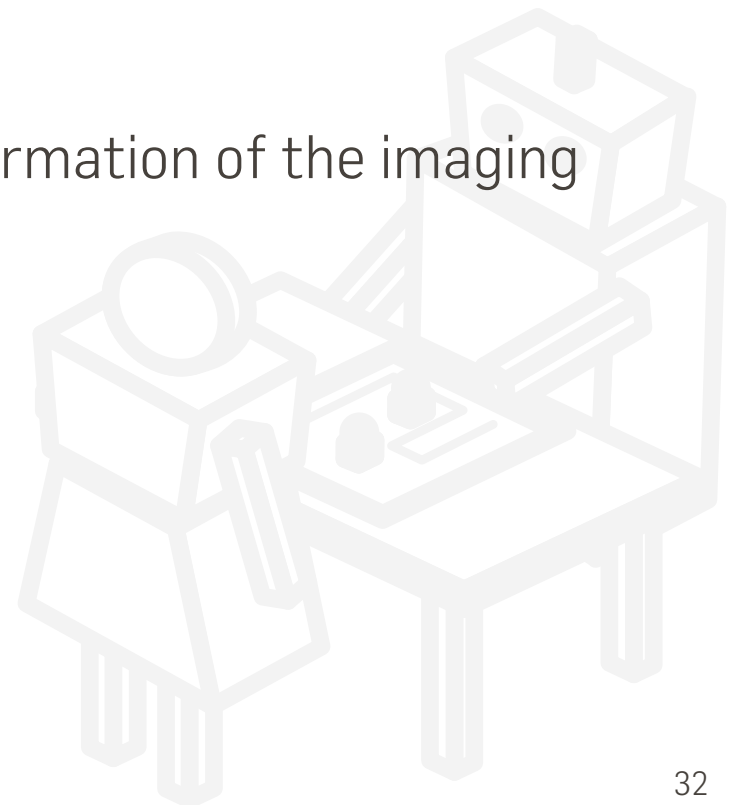
POINT CLOUDS REGISTRATION



[Source: PCL documentation]

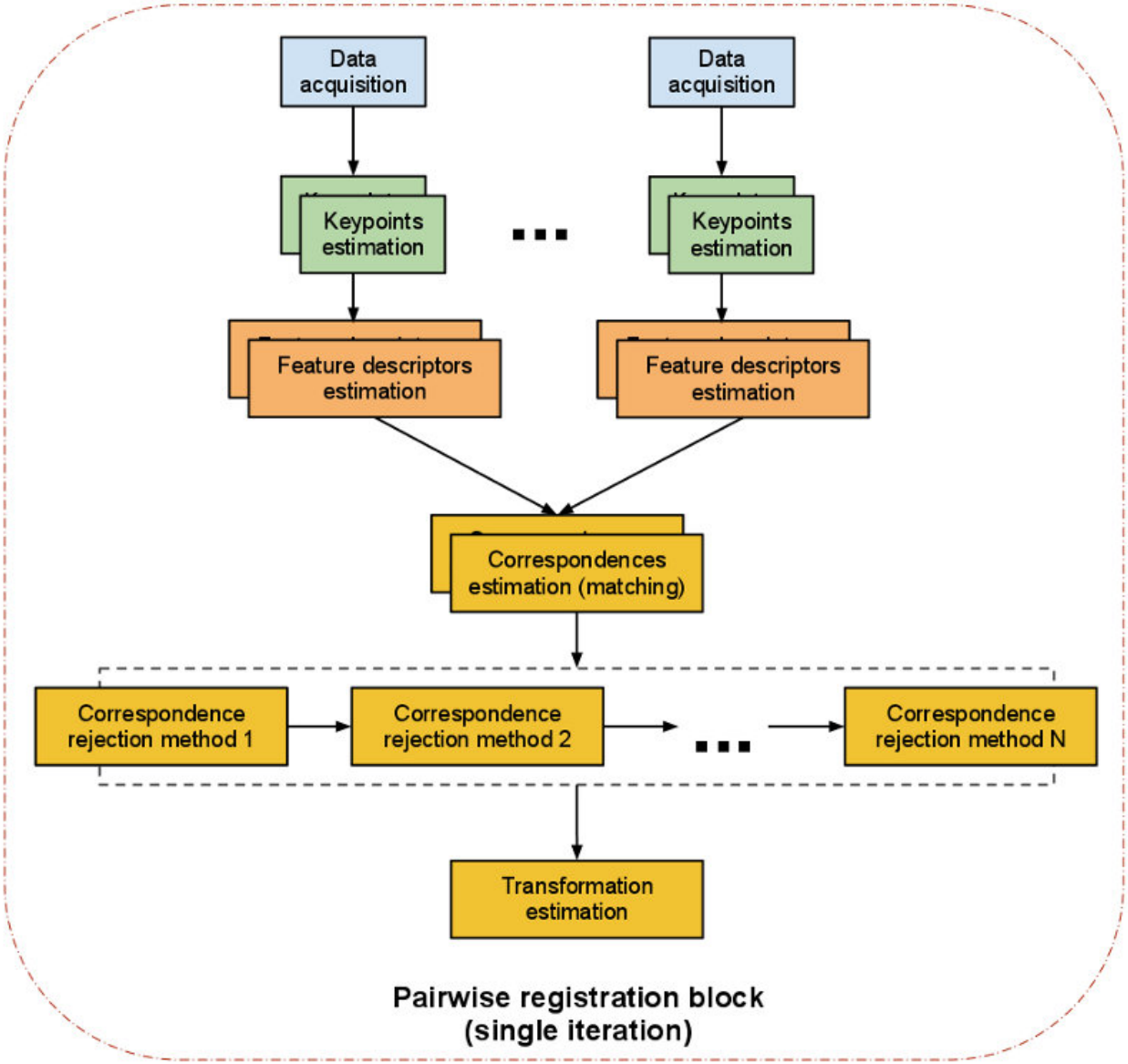
POINT CLOUDS REGISTRATION

Registration $\equiv$ estimating the rigid transformation of the imaging sensor between scans.



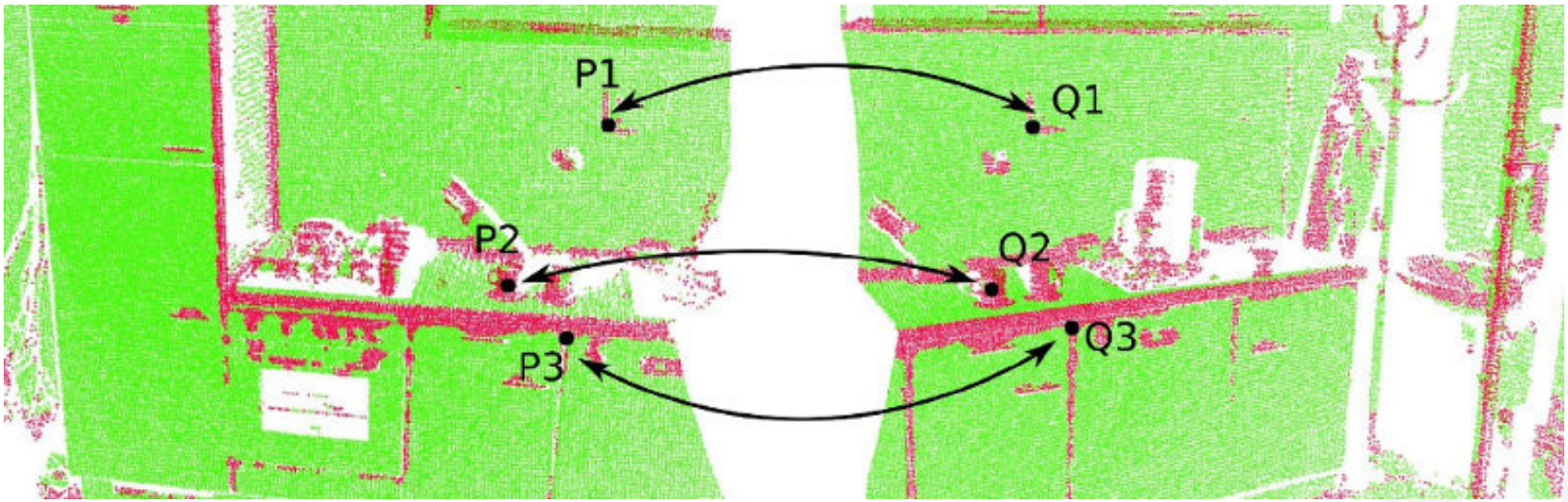
[Source: PCL documentation]

REGISTRATION ALGORITHM



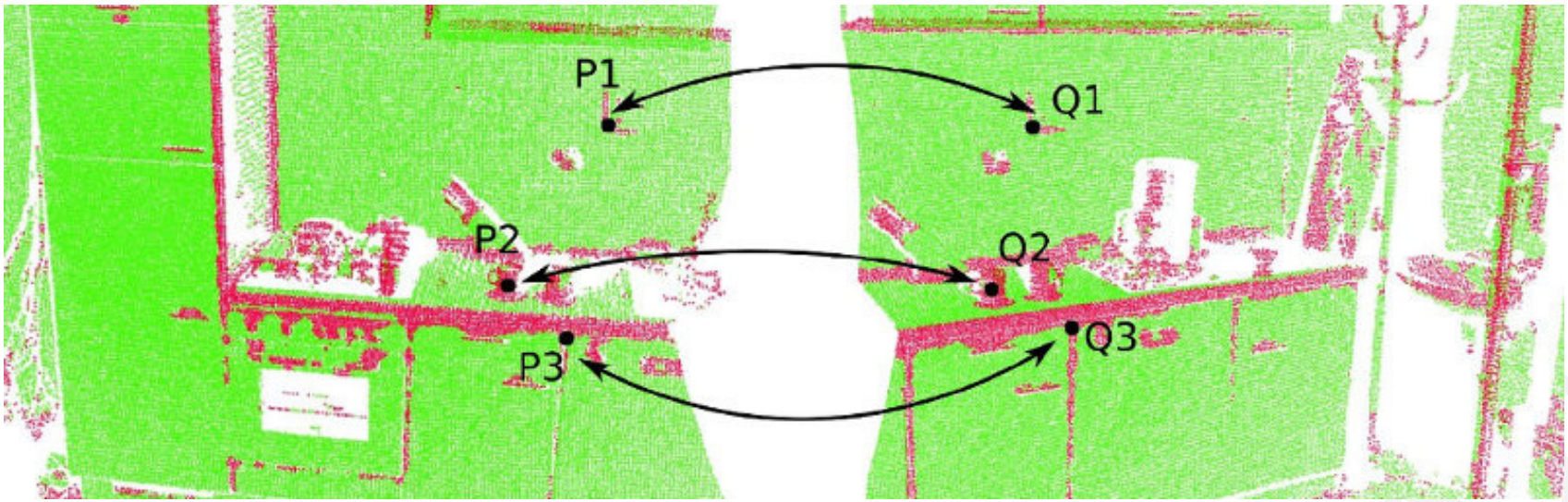
[Source: PCL documentation]

FEATURES?



[Source: PCL documentation]

FEATURES?

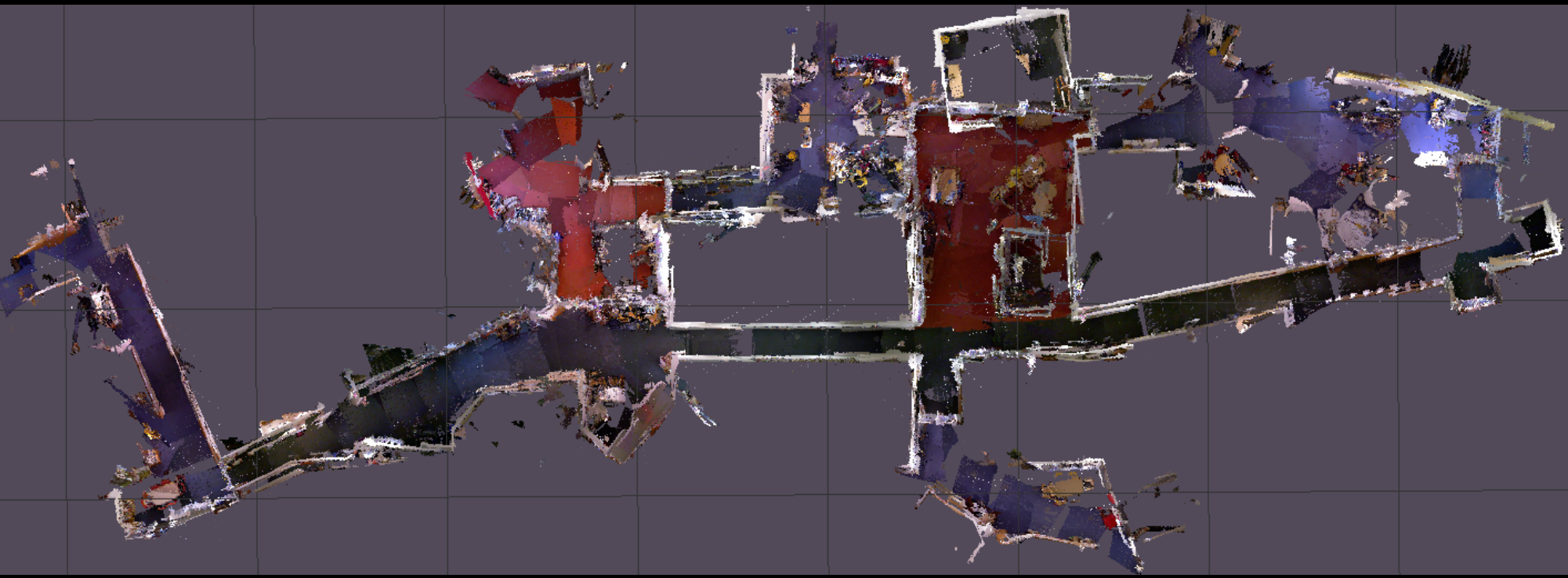


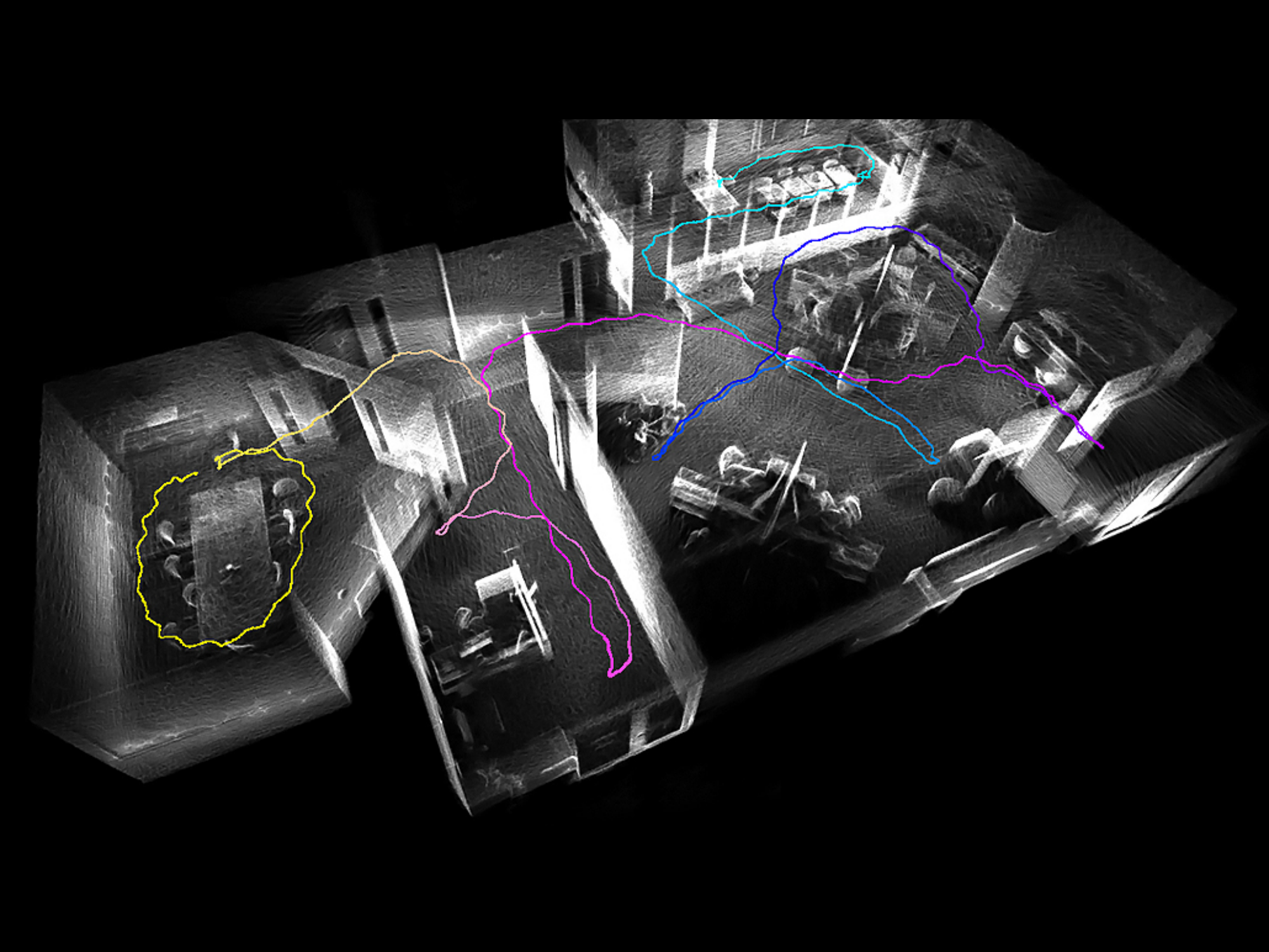
Good features should be:

- invariant over transformations,
- invariant over sampling densing,
- robust to noise

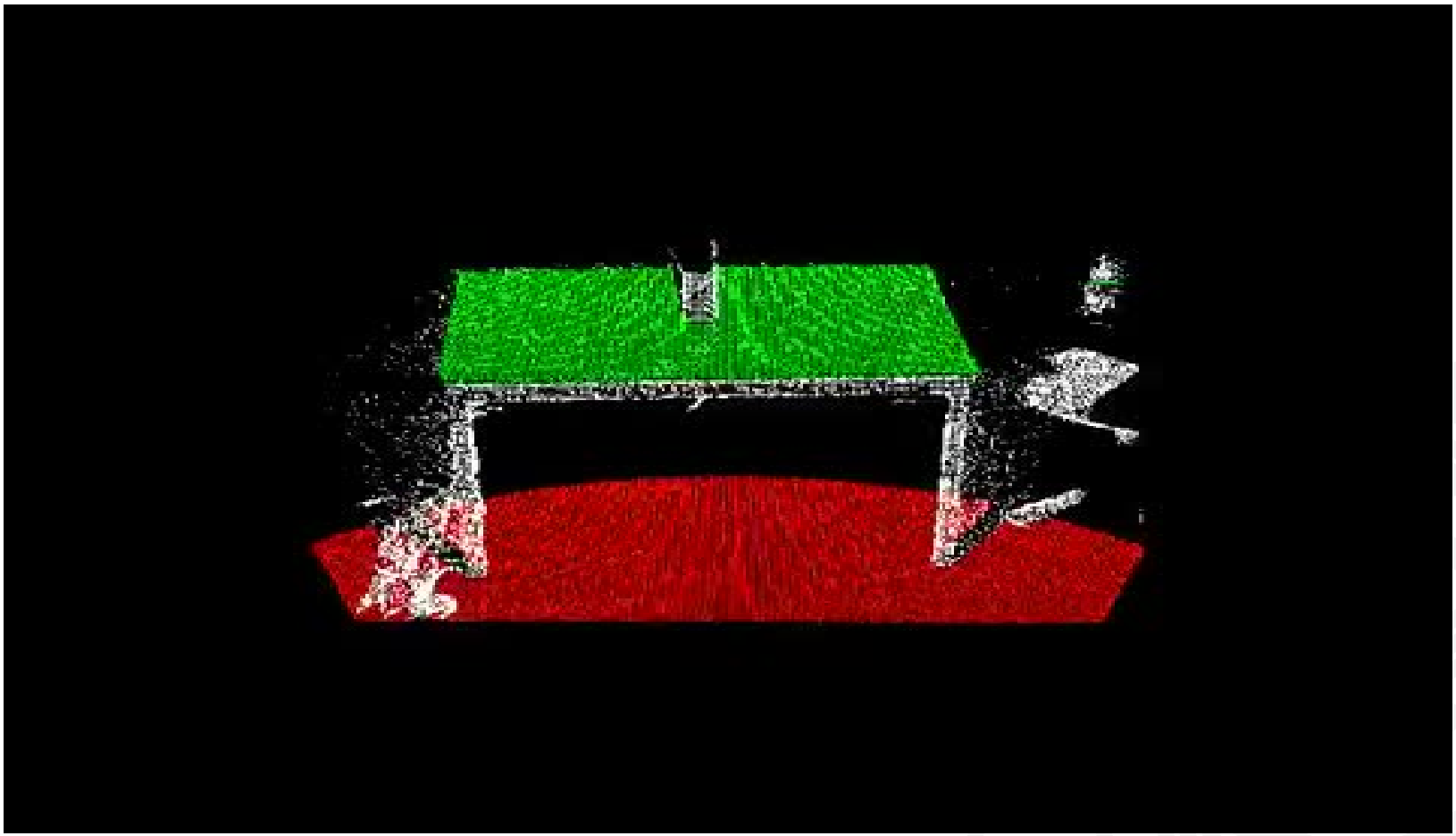
[Source: PCL documentation]

SLAM: SIMULTANEOUS LOCALIZATION AND MAPPING

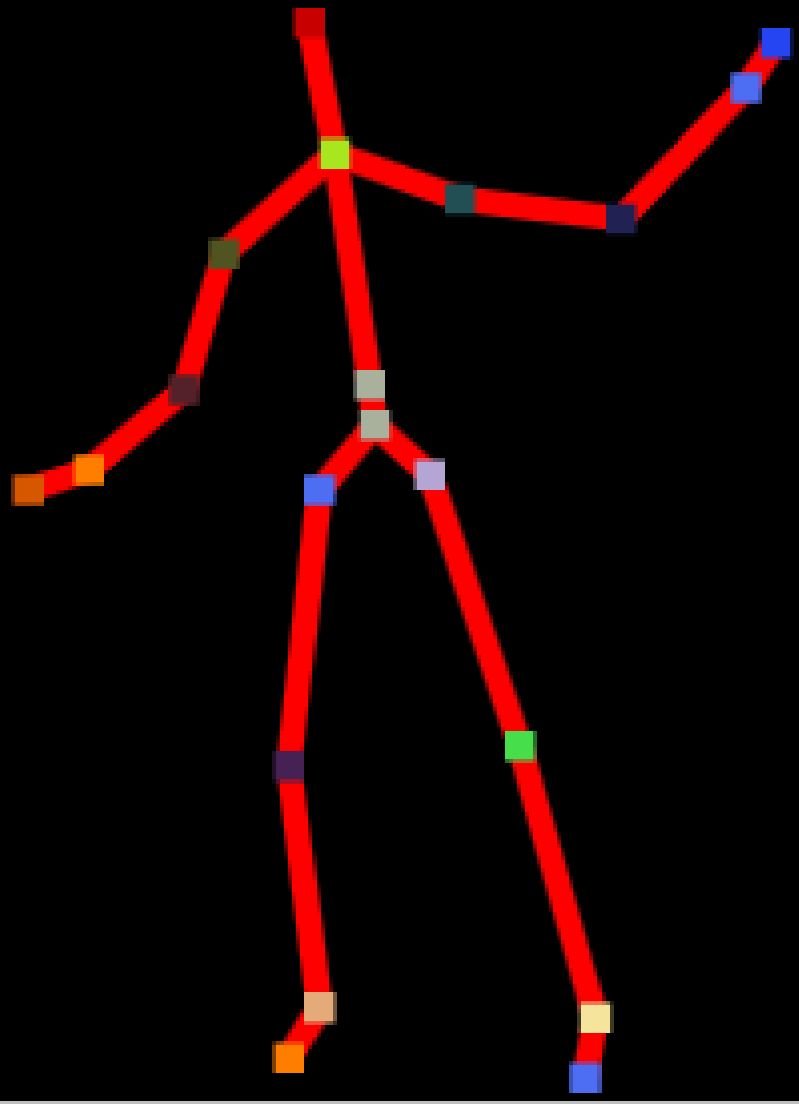




PLANE SEGMENTATION



SKELETON TRACKING



SKELETON TRACKING

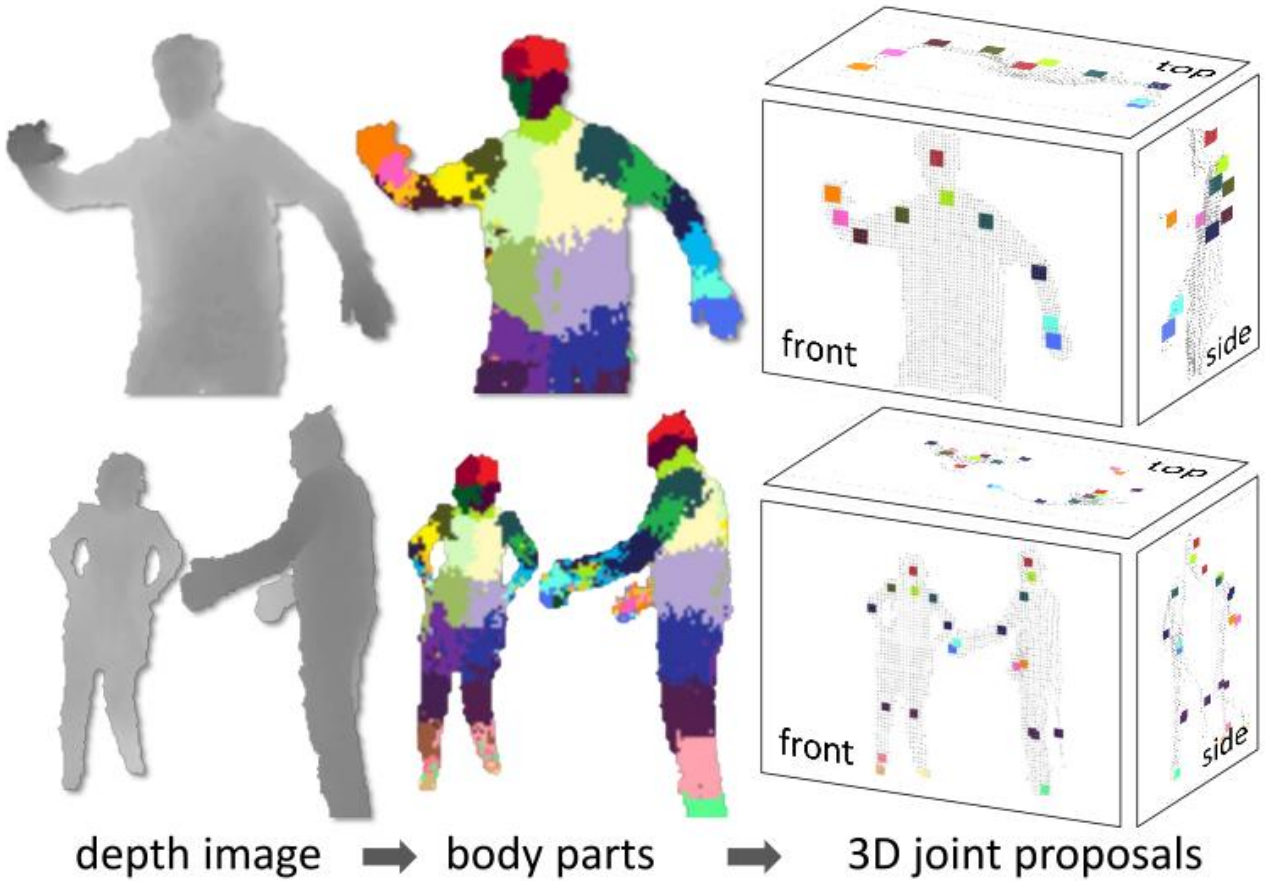
[Source: John MacCormick, Shotton et al., CVPR 2011]

SKELETON TRACKING

[Source: John MacCormick, Shotton et al., CVPR 2011]

SKELETON ESTIMATION

1. Estimate body parts using a randomized decision forest
2. Estimate the skeleton



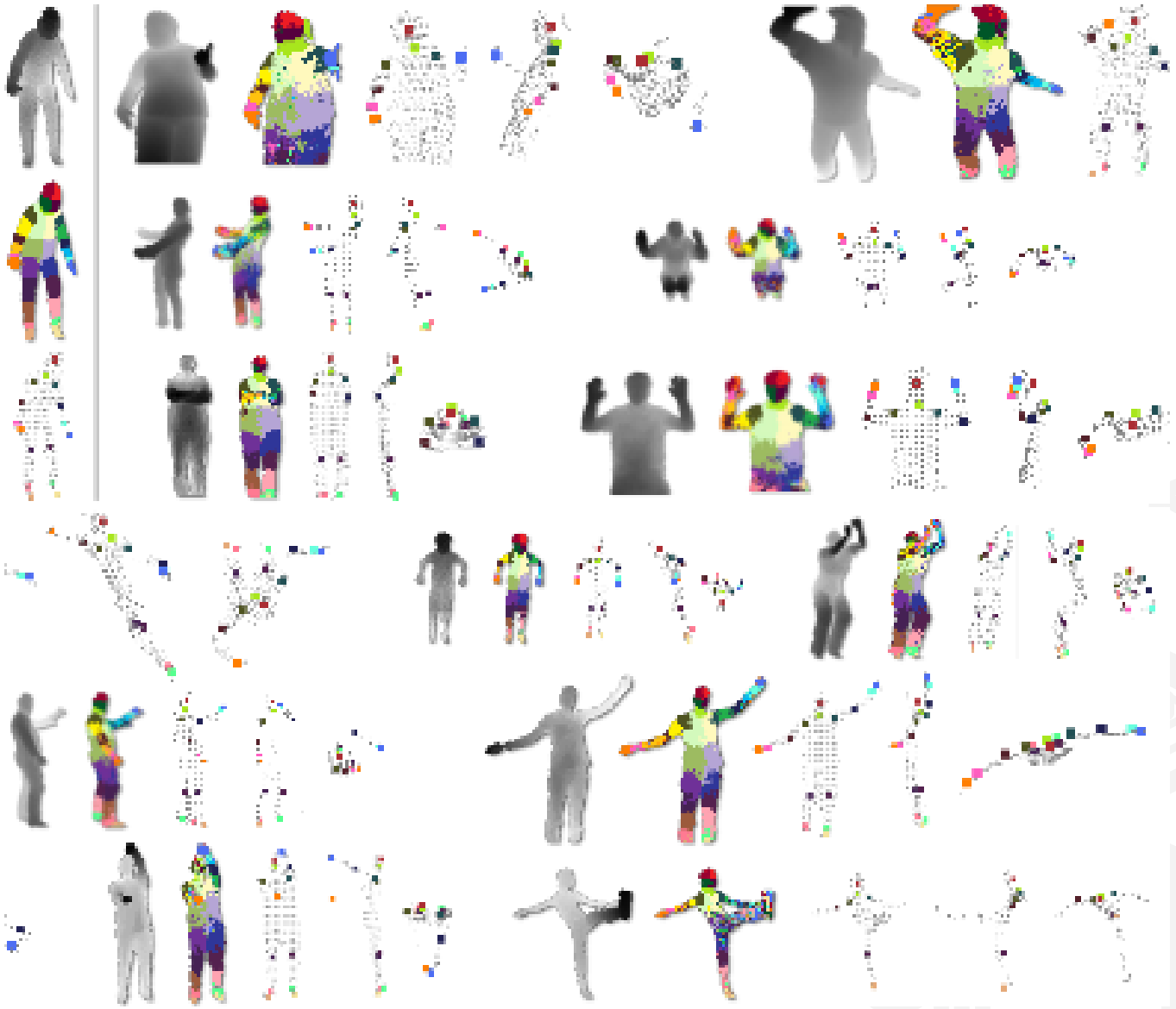
[Source: John MacCormick, Shotton et al., CVPR 2011]

BODY PARTS

Train a decision tree from ≈1M samples, computer-generated from ≈100K acquired pairs (depth image, motion capture). This allow **tagging** of body parts in the depth image.

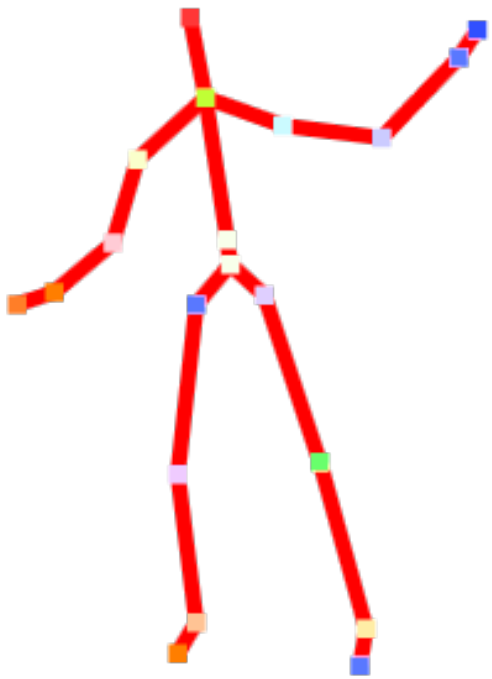
[Source: John MacCormick, Shotton et al., CVPR 2011]

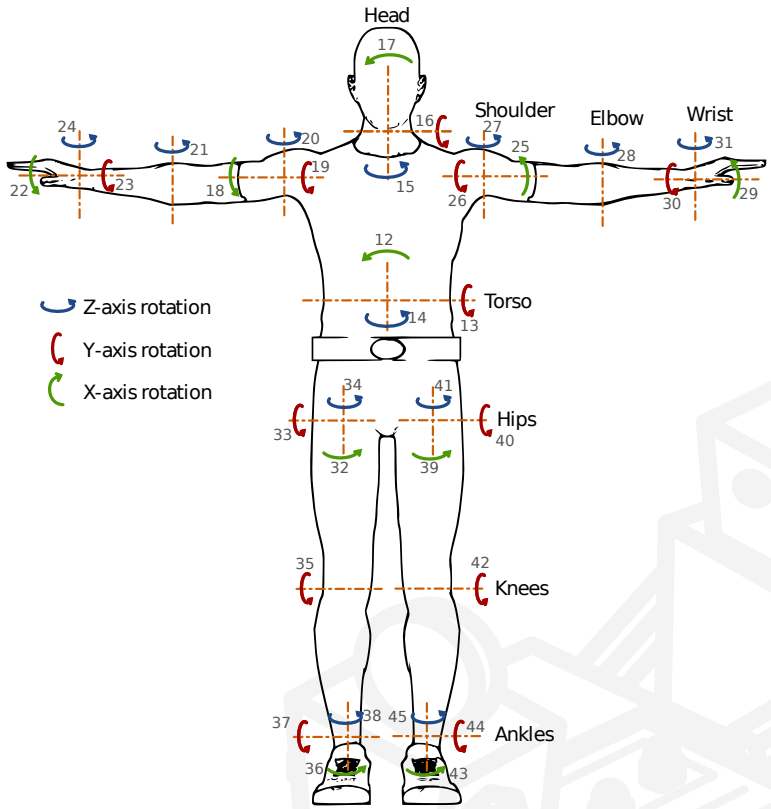
SKELETON



What to do with a skeleton?

CARTESIAN SPACE, JOINT SPACE



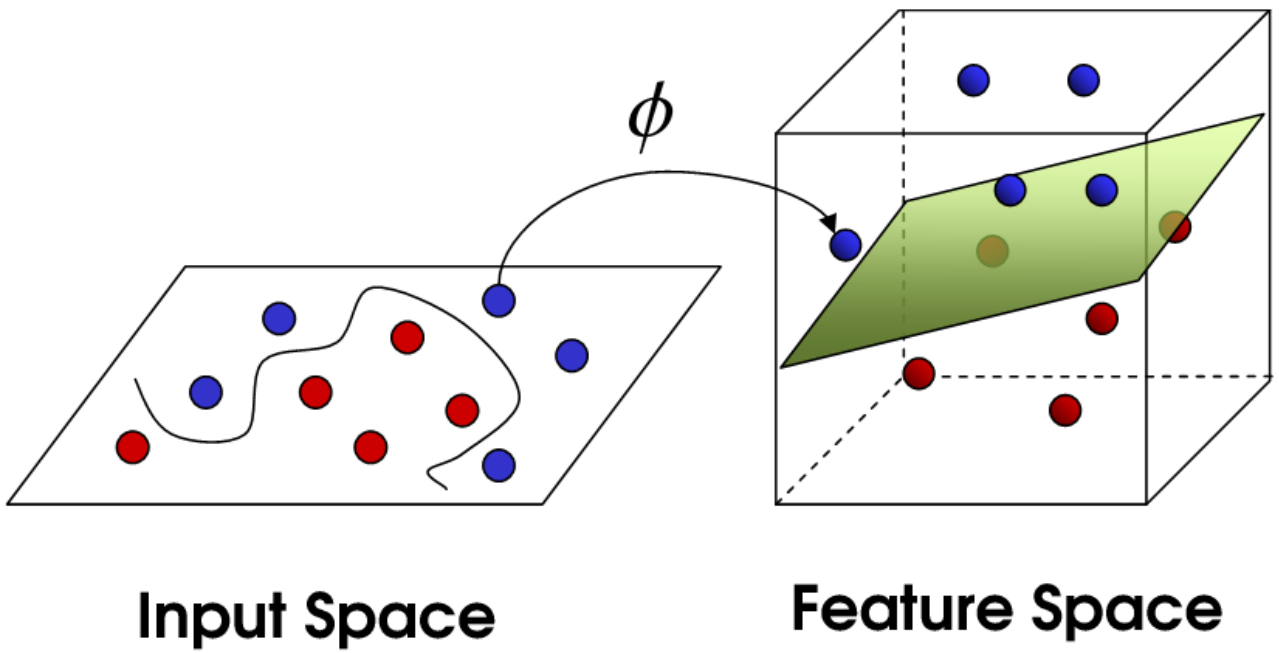


Joints values to cartesian positions: **forward kinematics**

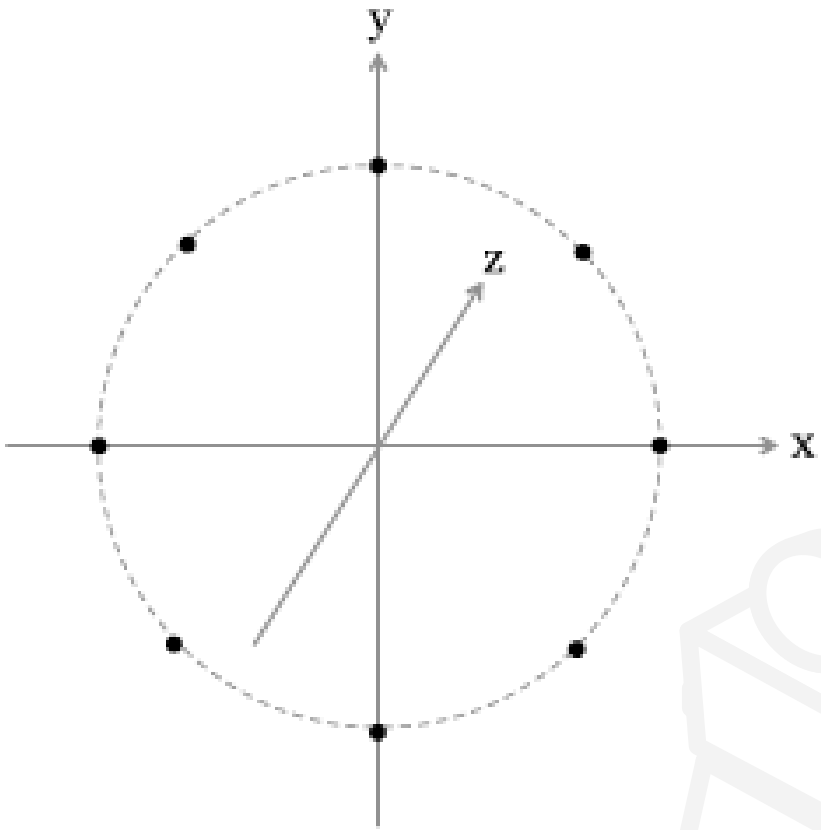
Cartesian positions to joints: **inverse kinematics**

RECOGNIZING POSTURES

Support Vector Machine (SVM) in joint space ⇒ supervised training

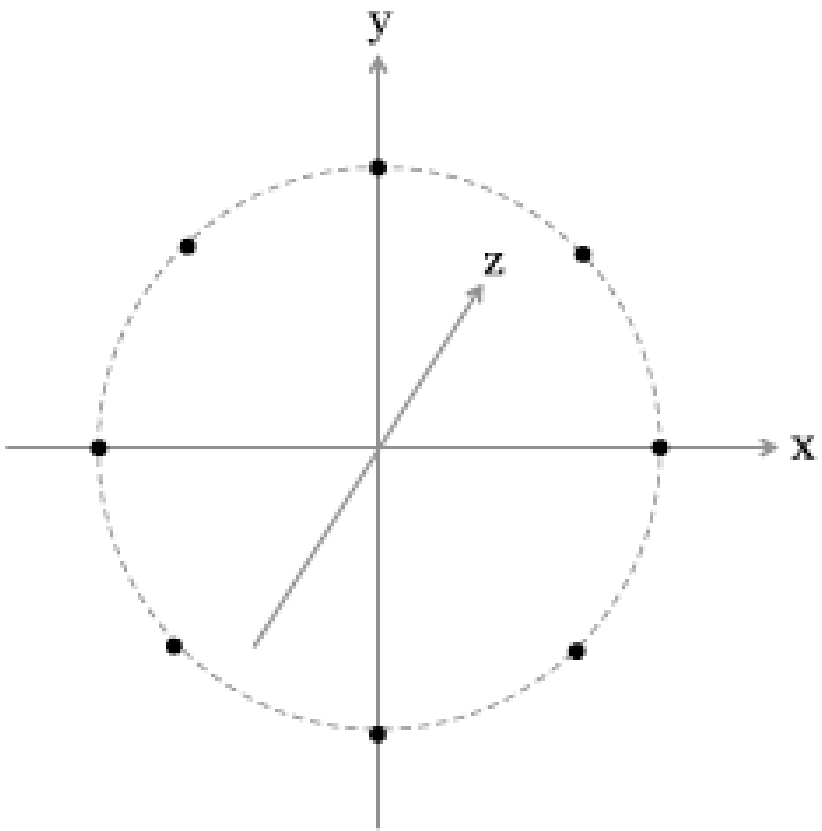


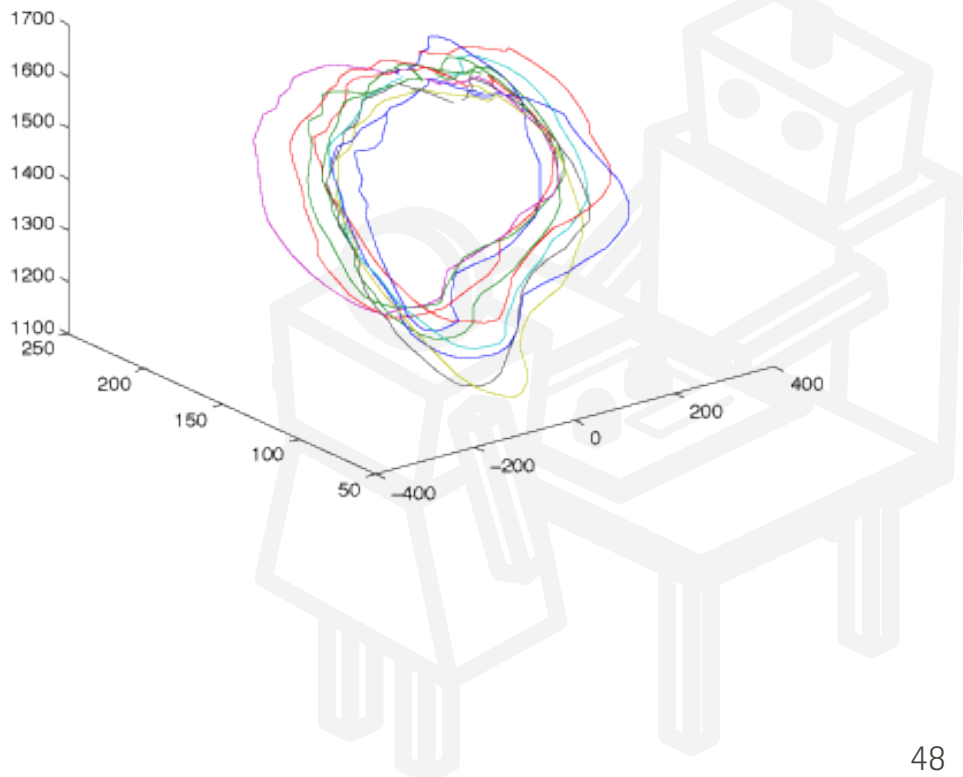
GESTURES



[Images source: http://www.creativedistraction.com]

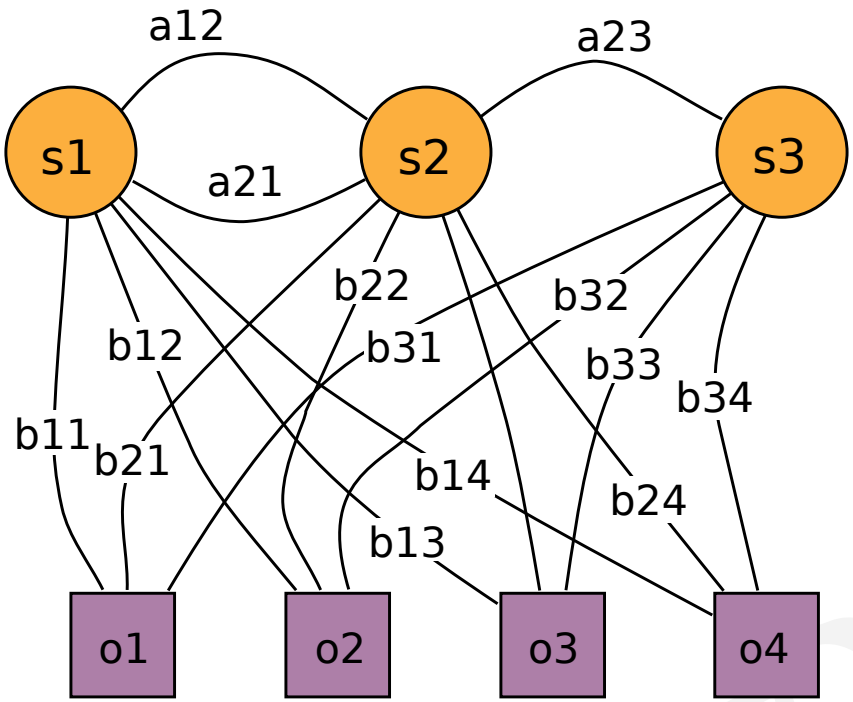
GESTURES





[Images source: http://www.creativedistraction.com]

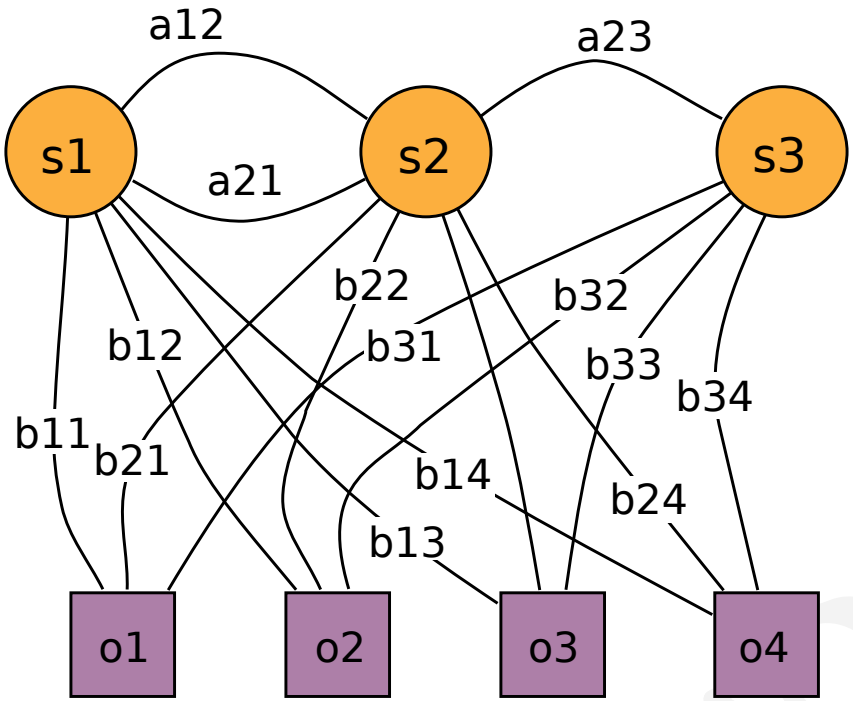
HIDDEN MARKOV MODELS



States: $s_1, s_2, \ldots s_n$ (eg, step of the gesture)
Observations: $o_1, o_2, \ldots o_m$ (eg, hand position)
State transition probabilities: a_{ij}
Output probabilities: b_{ij}

[Image source: Tdunning, CC-BY 3.0]

HIDDEN MARKOV MODELS



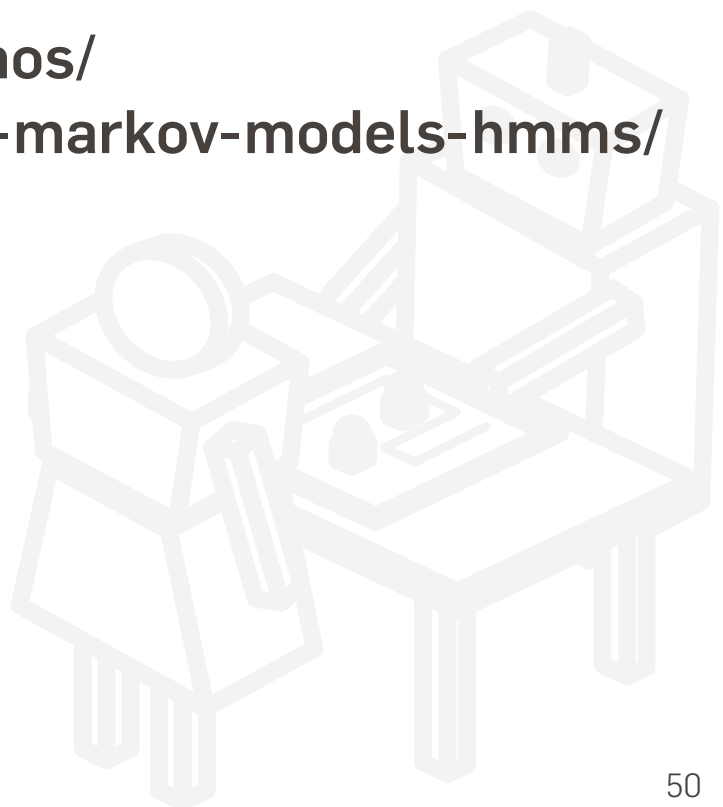
Probabilities are learnt from examples: **supervised training**

Very common for **temporal pattern recognition** (speech, handwriting, gestures...)

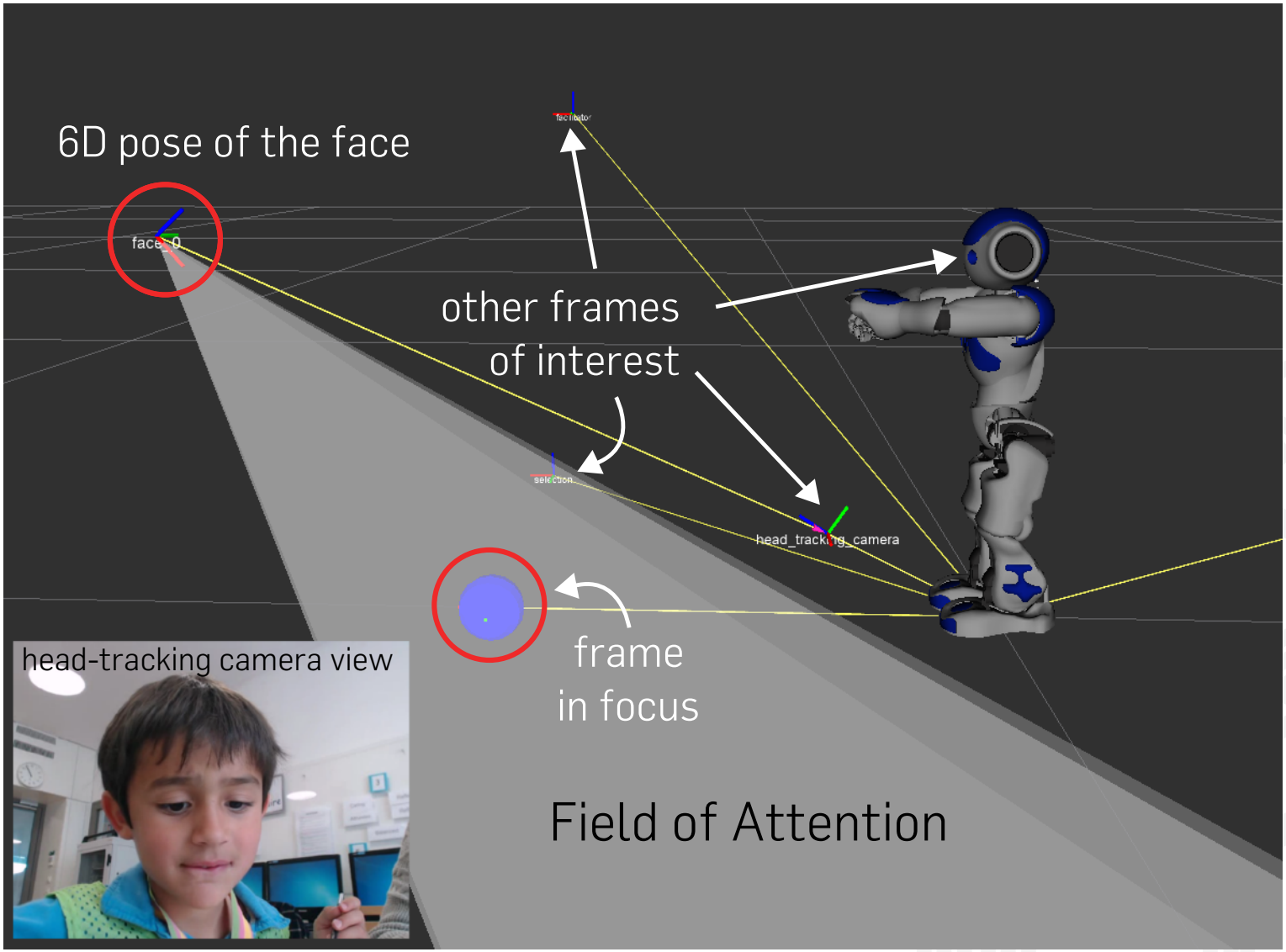
[Image source: Tdunning, CC-BY 3.0]

HIDDEN MARKOV MODELS

We'll leave it here, but for the curious:
http://www.creativedistraction.com/demos/gesture-recognition-kinect-with-hidden-markov-models-hmms/



ATTENTION


6D pose of the face
face_0
other frames
of interest
head_tracking_camera
frame
in focus
head-tracking camera view
Field of Attention

That's all, folk!

Questions:
Portland Square A216 or **severin.lemaignan@plymouth.ac.uk**

Slides:
github.com/severin-lemaignan/lecture-rgbd-cameras-hri