pcl


# How does a good feature look like?

Federico Tombari, University of Bologna

**PCL TUTORIAL @ICRA’13**

KARLSRUHE
ICRA 2013

# A feature..what?



### Etymology

From Anglo-Norman *feture*, from Old French *faiture*, from Latin *factura*.

### Pronunciation

- (*UK*) IPA: /fiːtʃə(ɹ)/, X-SAMPA: /fi:tS@(r)/
- Audio (US) ▶ 0:00 CC MENU
- Rhymes: -iːtʃə(ɹ)

### Noun

**feature** (*plural* **features**)

1. (*obsolete*) One's structure or make-up; form, shape, bodily proportions. [quotations ▾]
2. An important or main item.
3. (*media*) A long, prominent, article or item in the media, or the department that creates them; frequently used technically to distinguish content from news.
4. Any of the physical constituents of the face (eyes, nose, etc.).
5. (*computing*) A beneficial capability of a piece of software. [quotations ▾]
6. The cast or structure of anything, or of any part of a thing, as of a landscape, a picture, a treaty, or an essay; any marked peculiarity or characteristic; as, one of the features of the landscape. [quotations ▾]
7. (*archaeology*) Something discerned from physical evidence that helps define, identify, characterize, and interpret an archeological site. [quotations ▾]
8. (*engineering*) Characteristic forms or shapes of a part. For example, a hole, boss, slot, cut, chamfer, or fillet.

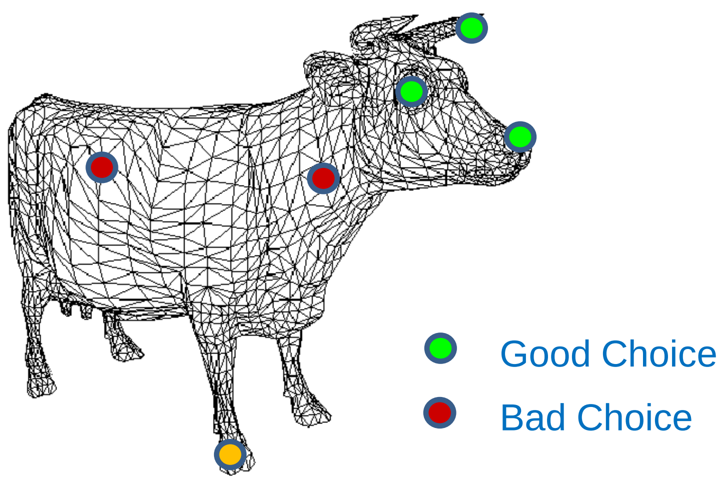
- Feature is a compact – but *rich* – *representation* of our (3D) data
- It is designed to be invariant (or robust) to a specific class of transformations and/or set of disturbances

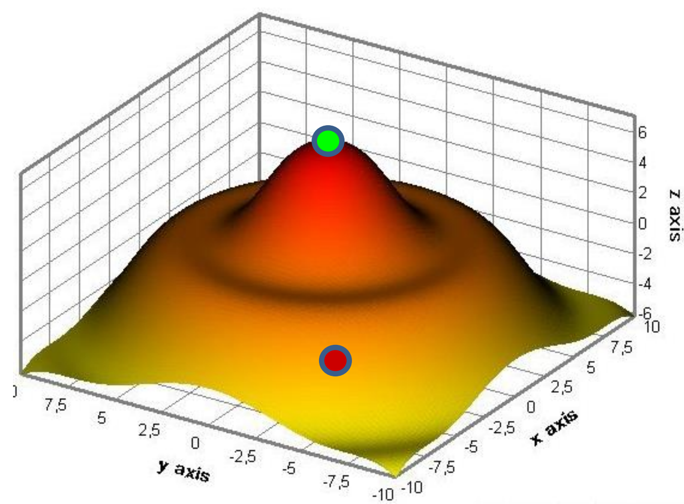
# 3D keypoint detection



**Keypoint Extraction** → Description → Matching

- **3D keypoints** are
  - **Distinctive**, i.e. suitable for effective description and matching (*globally definable*)
  - **Repeatable** with respect to point-of-view variations, noise, etc… (*locally defin*able)
- Usually scale-invariance is not an issue (but better if each feature is extracted together with its characteristic scale)




*Distinctiveness vs. repeatability*

# pcl::Keypoints

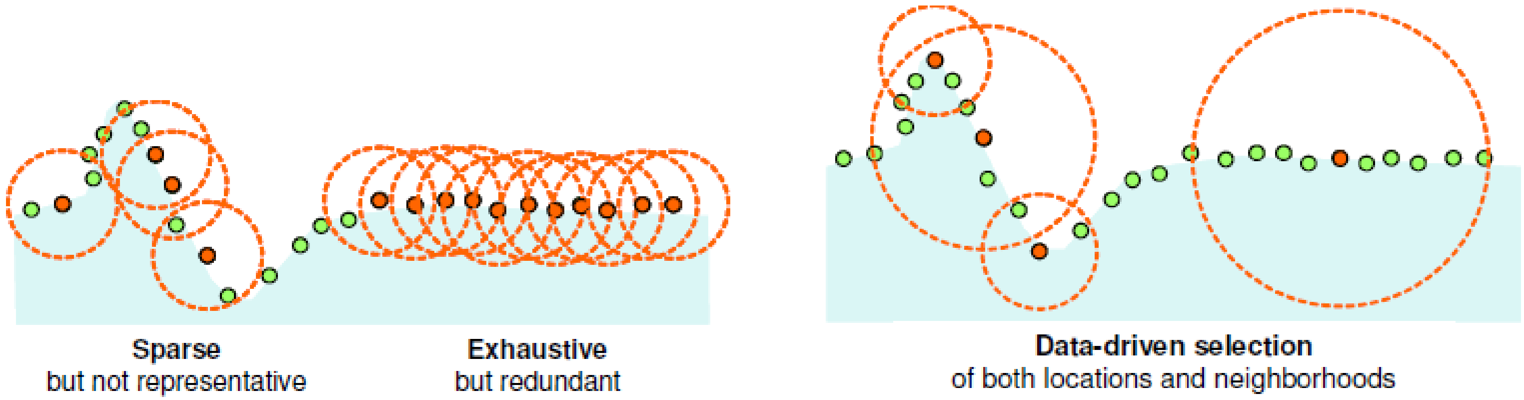


- (for now) a small set of detectors specifically proposed for 3D point clouds and range maps
  - Intrinsic Shape Signatures (ISS) [Zhong ICCVW09]
  - NARF [Steder ICRA11]
  - (Uniform Sampling)

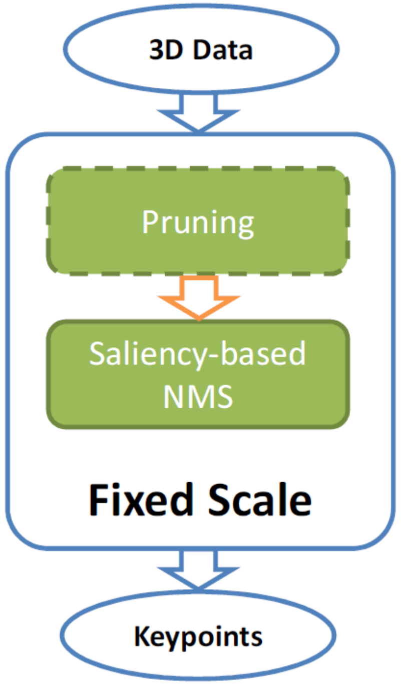
- Several detectors «derived» from 2D interest point detectors
  - Harris (2D, 3D, 6D) [Harris AVC88] - CD
  - SIFT [Lowe IJCV04] - BD
  - SUSAN [Smith IJCV95] - CD
  - AGAST [Mair ECCV10] - CD

# Taxonomy






***Courtesy of Unnikrishnan & Hebert***



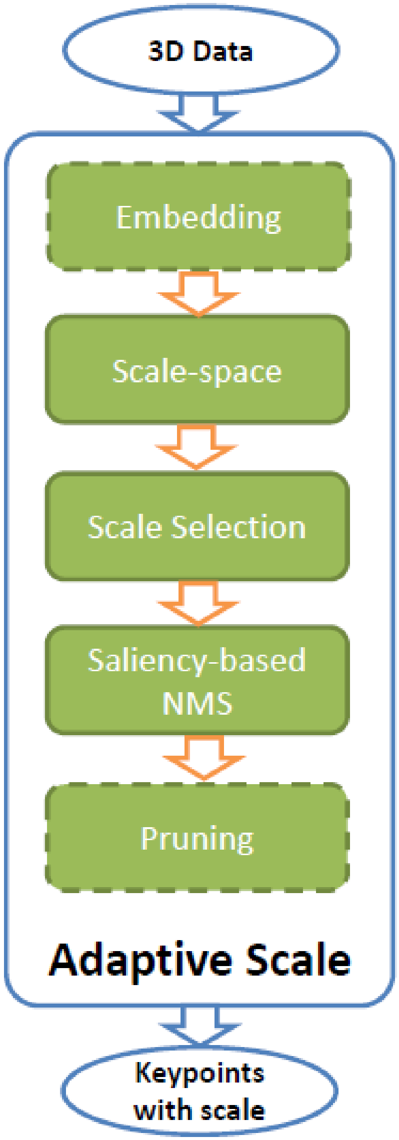

- In 3D scale is (generally) not an issue

  **BUT**
- The characteristic scale is still an important property of a 3D keypoint
- Several recent proposals, two main categories [Tombari IJCV13]
  - **Fixed-scale detectors:** all keypoints are detected at a specific scale (input parameter)
    - Local Surface Patches (LSP) [Chen07]
    - Intrinsic Shape Signatures (ISS) [Zhong09]
    - KeyPoint Quality (KPQ) [Mian10]
    - Heat Kernel Signature (HKS) [Sun09]

# Taxonomy (2)

pointcloudlibrary

- **Adaptive-scale detectors:** specific scale-space analysis to detect salient structures at multiple scales, associating each keypoint a **characteristic scale**
  - Scale space on the cloud/mesh
    - KPQ Adaptive Scale (KPQ-AS) [Mian10]
    - Salient Points (SP) [Castellani08]
    - Laplace-Beltrami Scale-Space (LBSS) [Unnikrishnan08]
    - MeshDoG [Zaharescu12]
  - Scale space on voxel maps
    - 3D-SURF [Knopp10]
  - Scale space on range images
    - Scale-dependent local shape detector [Novatnack08]
    - HK Maps [Akagunduz07])




- Need for performance assessment [Tombari13]
  - Locality repeatability / Quantity
  - Scale repeatability
  - Efficiency
  - ***www.vision.deis.unibo.it/keypoints3d***

# Intrinsic Shape Signatures



- Exploits the covariance matrix $\mathbf{M}(\mathbf{p}_i) = \frac{1}{\sum_{j=1}^{k} \rho_i} \sum_{j=1}^{k} \rho_i (\mathbf{p}_j - \mathbf{p}_i)(\mathbf{p}_j - \mathbf{p}_i)^T$
- Let its eigenvalues, in decreasing magnitude order, be

$$\lambda_1, \lambda_2, \lambda_3$$

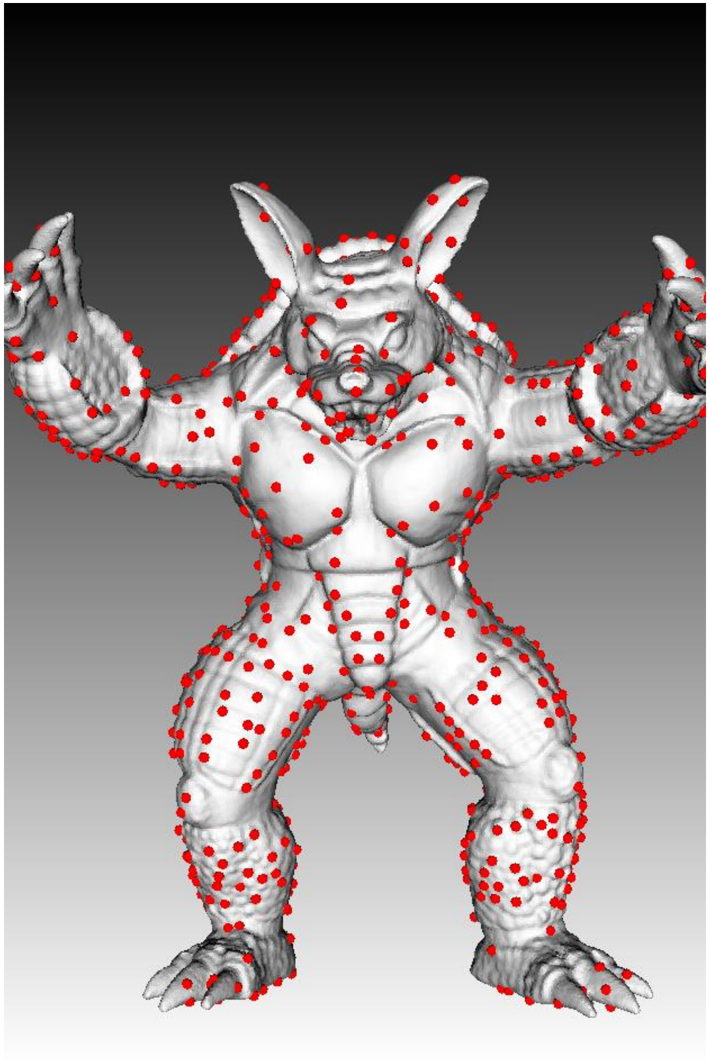
- The pruning step discards points with similar spreads along the principal directions, where a repeatable LRF cannot be defined

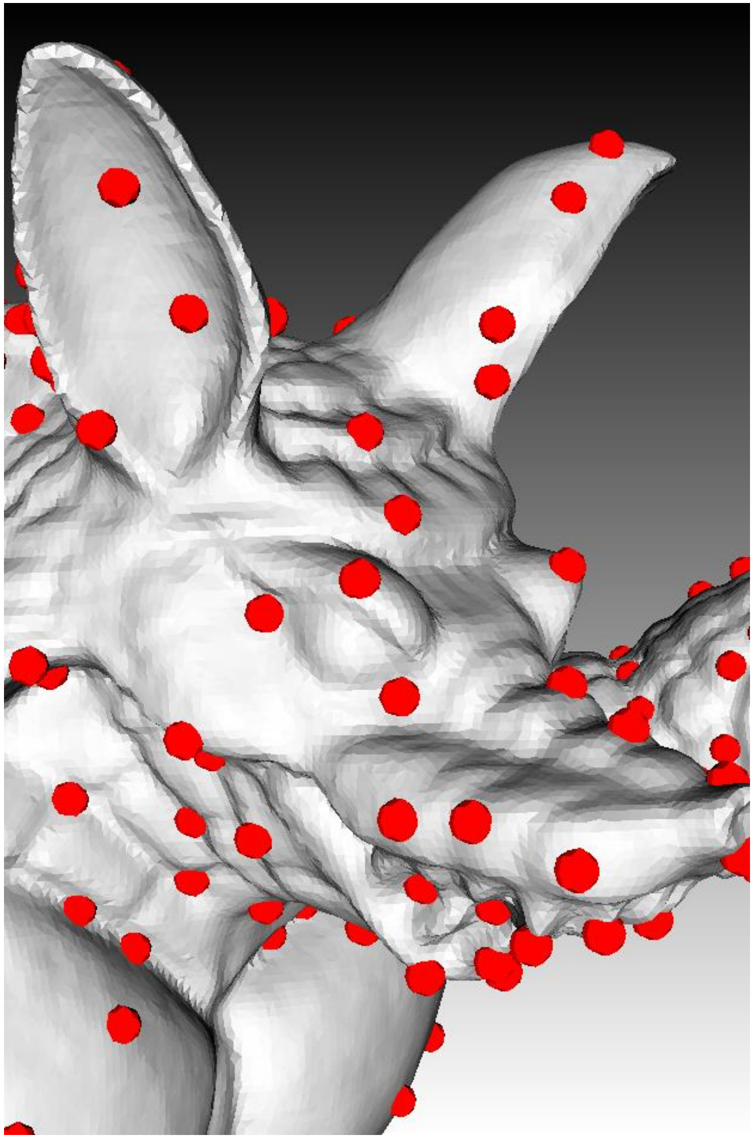
$$\frac{\lambda_2(\mathbf{p})}{\lambda_1(\mathbf{p})} < Th_{12} \wedge \frac{\lambda_3(\mathbf{p})}{\lambda_2(\mathbf{p})} < Th_{23}$$

- Saliency is the magnitude of the third eigenvalue $\rho(\mathbf{p}) \doteq \lambda_3(\mathbf{p})$
- It includes only points with large variations along each principal direction
- “Winner” of PCL 3D detector evaluation in [Filipe 2013]

# Intrinsic Shape Signatures

pointcloudlibrary



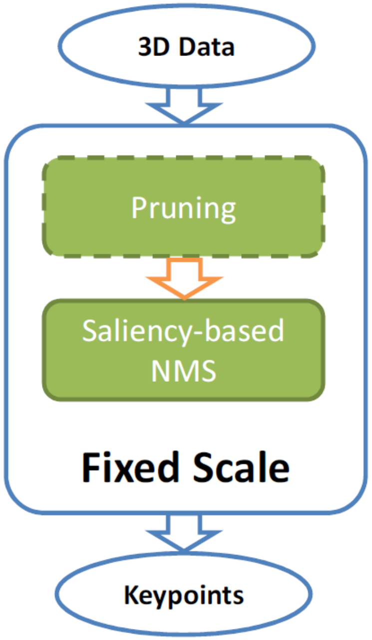


# Example

pointcloudlibrary

```
pcl::PointCloud<int> indices;
pcl::UniformSampling<pcl::PointXYZ> uniform_sampling;
uniform_sampling.setInputCloud (cloud);
uniform_sampling.setRadiusSearch (0.05f);
uniform_sampling.compute (indices);
```

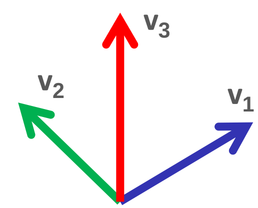
```
pcl::PointCloud<pcl::PointXYZ>::Ptr keypoints (new
pcl::PointCloud<pcl::PointXYZ>());
pcl::ISSKeypoint3D<pcl::PointXYZ, pcl::PointXYZ> iss_detector;
iss_detector.setSalientRadius (support_radius);
iss_detector.setNonMaxRadius (nms_radius);
iss_detector.setInputCloud (cloud);
iss_detector.compute (*keypoints);
```



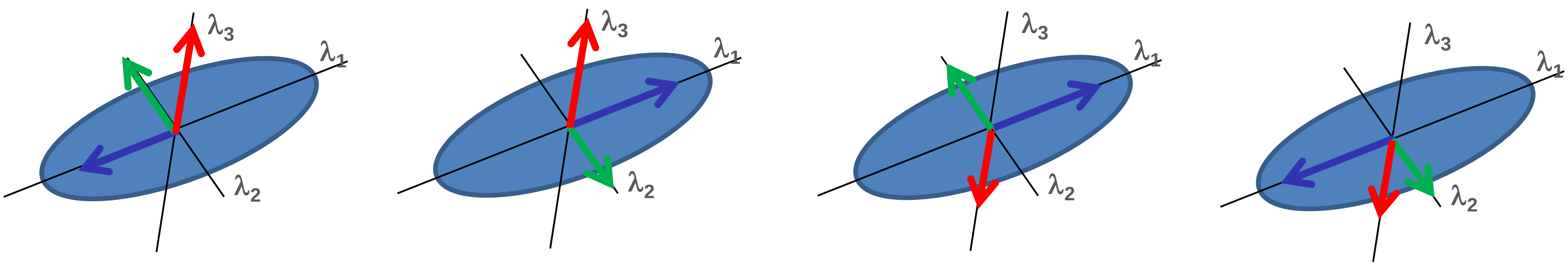
# Local Reference Frame

pointcloudlibrary

- 3 orthogonal unit vectors defined upon a local support
- Goal:
  - invariant to rotations and translations
  - robust to noise and clutter




- Common approach to deal with ambiguities in the LRF definition
  - Define **multiple LRFs** at each keypoint, providing multiple descriptions of the same keypoint
  - Cons:
    - more descriptors to be computed and matched (less efficient)
    - ambiguity pushed to the matching stage
  - Eg. EVD of the scatter matrix computed over the support as used in [Mian10] [Novatnack08] [Zhong09], provides 3 repeatable directions but **no repeatable sign** [Tombari10]
  - 4 different RFs can be obtained by enforcing the right-hand rule



# LRF: example

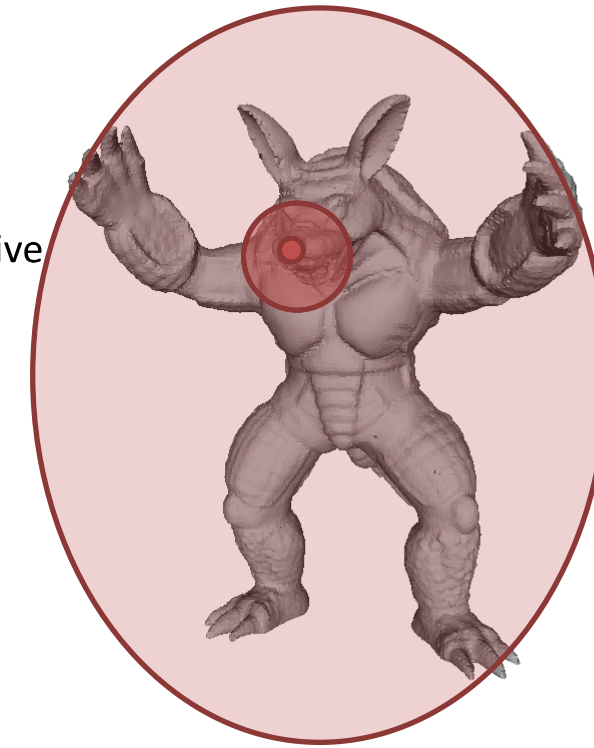
pointcloudlibrary

```
pcl::PointCloud< pcl::ReferenceFrame >::Ptr lrfs(new pcl::PointCloud<
pcl::ReferenceFrame> ());

pcl::BOARDLocalReferenceFrameEstimation<pcl::PointXYZ, pcl::Normal,
pcl::ReferenceFrame> lrf_est;
lrf_est.setRadiusSearch (0.5f);
lrf_est.setInputCloud (keypoints);
lrf_est.setInputNormals (cloud_normals);
lrf_est.setSearchSurface (cloud);

lrf_est.compute (*lrfs);
```

# Global vs local representations

pointcloudlibrary

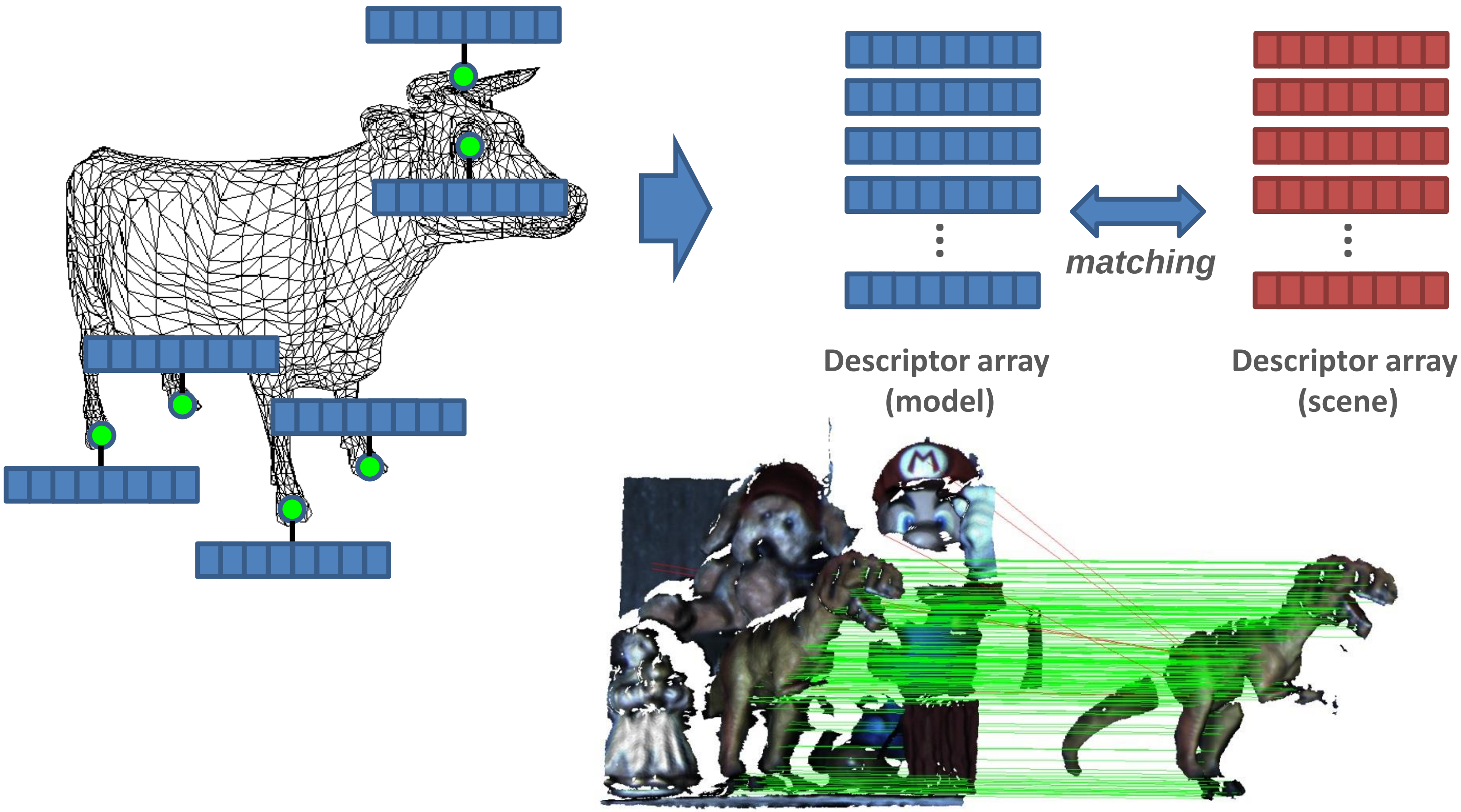

- **compact** representations aimed at detecting similarities between surfaces (*surface matching*)
- based on the support size
  - **Pointwise descriptors**
    - Simple, efficient, but not robust to noise, often not descriptive enough
  - **Local/Regional descriptors**
    - Well suited to handle clutter and occlusions
    - Can be vector quantized in codebooks
    - Segmentation, registration, recognition in clutter, 3D SLAM
  - **Global descriptors**
    - Complete information concerning the surface is needed (no occlusions and clutter, unless pre-processing)
    - Higher invariance, well suited for **retrieval and categorization**
    - More descriptive on objects with poor geometric structure (household objects..)



# Local descriptors



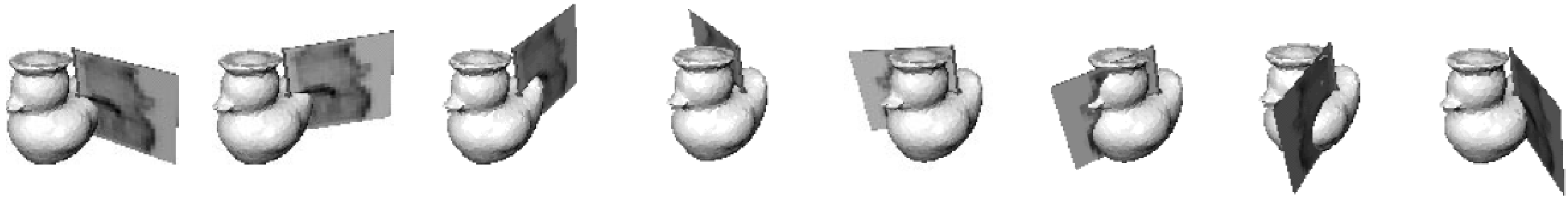
- Descriptive representation of the local neighborhood (***support***) of a point
- Local descriptors can embed also intensity/color information (***RGB-D descriptors***)
- Matching descriptions yields point-to-point correspondences between two surfaces



# Spin Images

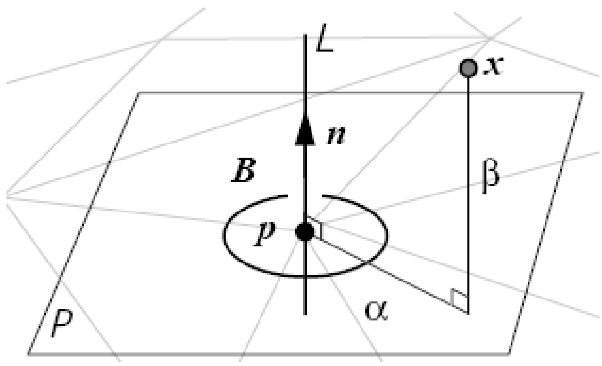
pointcloudlibrary

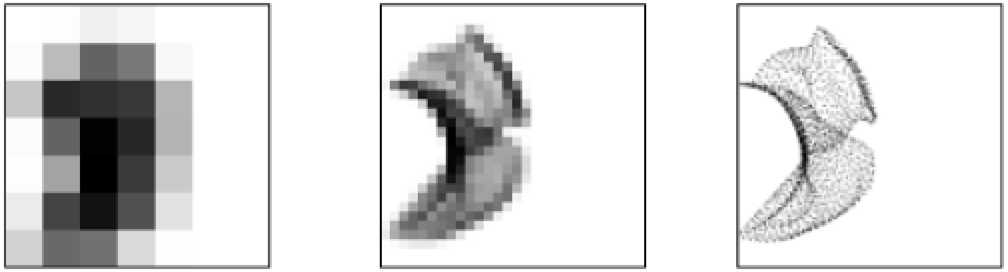
- Spin Image descriptor [Johnson99] is arguably the most popular 3D local descriptor
- 2D histograms accumulating points by spinning around a repeatable axis (*normal*)



***(courtesy of Johnson & Hebert)***

- Rotation and translation invariant, not scale invariant
- Appreciates uniform surface sampling
- Variants: compressed-SI (PCA)
- ***pcl::SpinImageEstimation***



Effect of bin size (courtesy of Johnson & Hebert)

# Point Feature Histogram

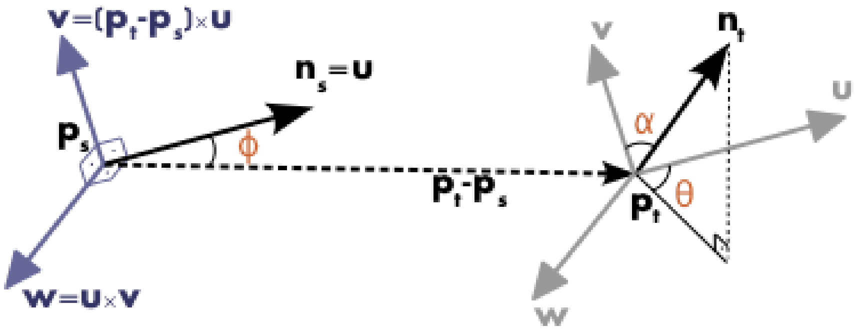
pointcloudlibrary

- PFH [Rusu08] computes 3 values for each pair in the neighbourhood
  - Complexity O(k²), extremely slow.
- ***pcl::PFHEstimation***
- For each pair, it computes a LRF *u-v-w* centred on one point $p_s$ as
  - The normal $u = n_s$
  - The cross product between ns and the vector (pt-ps) $v = n_s \times (p_t - p_s)$
  - The cross product between the previous vectors $w = u \times v$
- Then, it computes and accumulates

$$\alpha = \arccos(v \cdot n_t)$$

$$\phi = \arccos\left( u \cdot \frac{(p_t - p_s)}{\|p_t - p_s\|_2} \right)$$

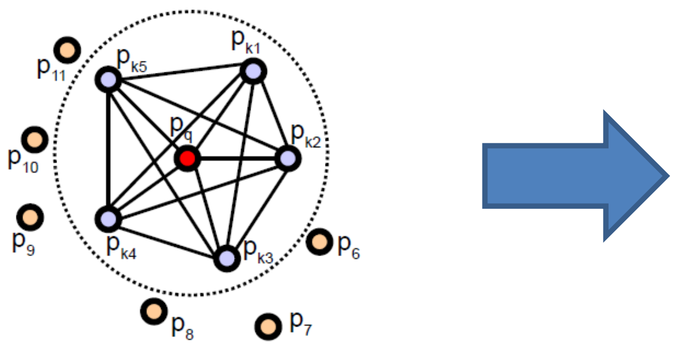
$$\theta = \arctan(w \cdot n_t, u \cdot n_t)$$

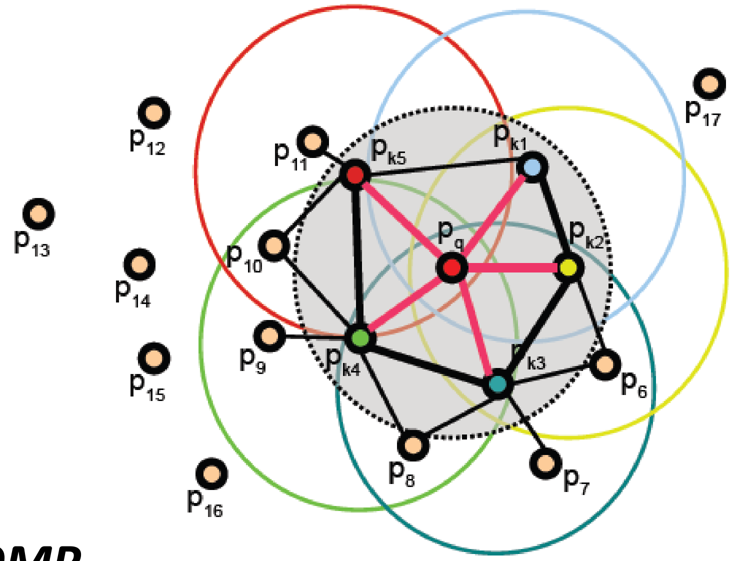


# Fast PFH

- FPFH [Rusu09]: approximation of PFH with linear complexity in the number of neighbors
  - Compute SPFH (Simplified PFH) between the keypoint and every neighbor
  - Combine the weighted SPFHs to form the final Fast PFH

$$FPFH(p_i) = SPFH(p_i) + \frac{1}{k}\sum_{j=1}^{k}\frac{1}{\omega_j} SPFH(p_j)$$



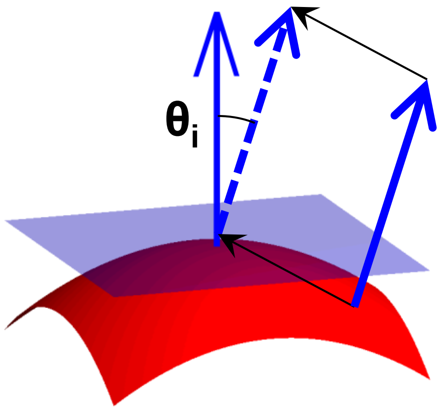



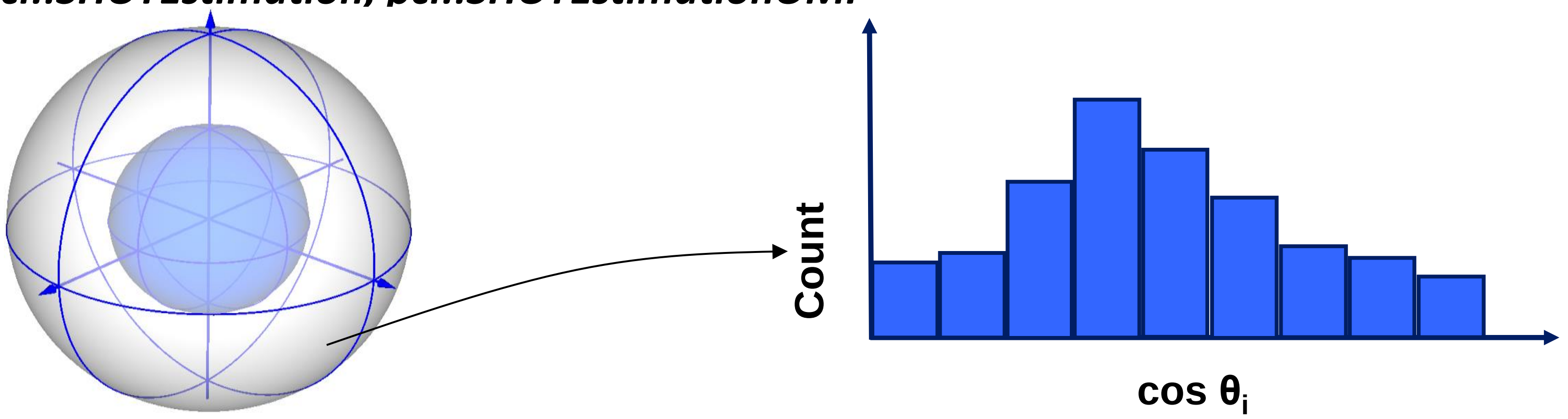

- ***pcl::FPFHEstimation, pcl::FPFHEstimationOMP***

# SHOT descriptor



- Signatures of Histograms of OrienTations [Tombari10]
- Inspired by SIFT: computation of a geometric coarsely localized local set of histograms of first-order derivatives.
- The local support is partitioned by means of a spherical grid
- For each volume of the grid, an histogram of the cosines of the angle θi between the normal at each point and the normal at the feature point is computed.
- Quadrilinear interpolation to smooth out quantization distortions
- Normalization of the descriptor for robustness towards point density variations
- ***pcl::SHOTEstimation, pcl::SHOTEstimationOMP***

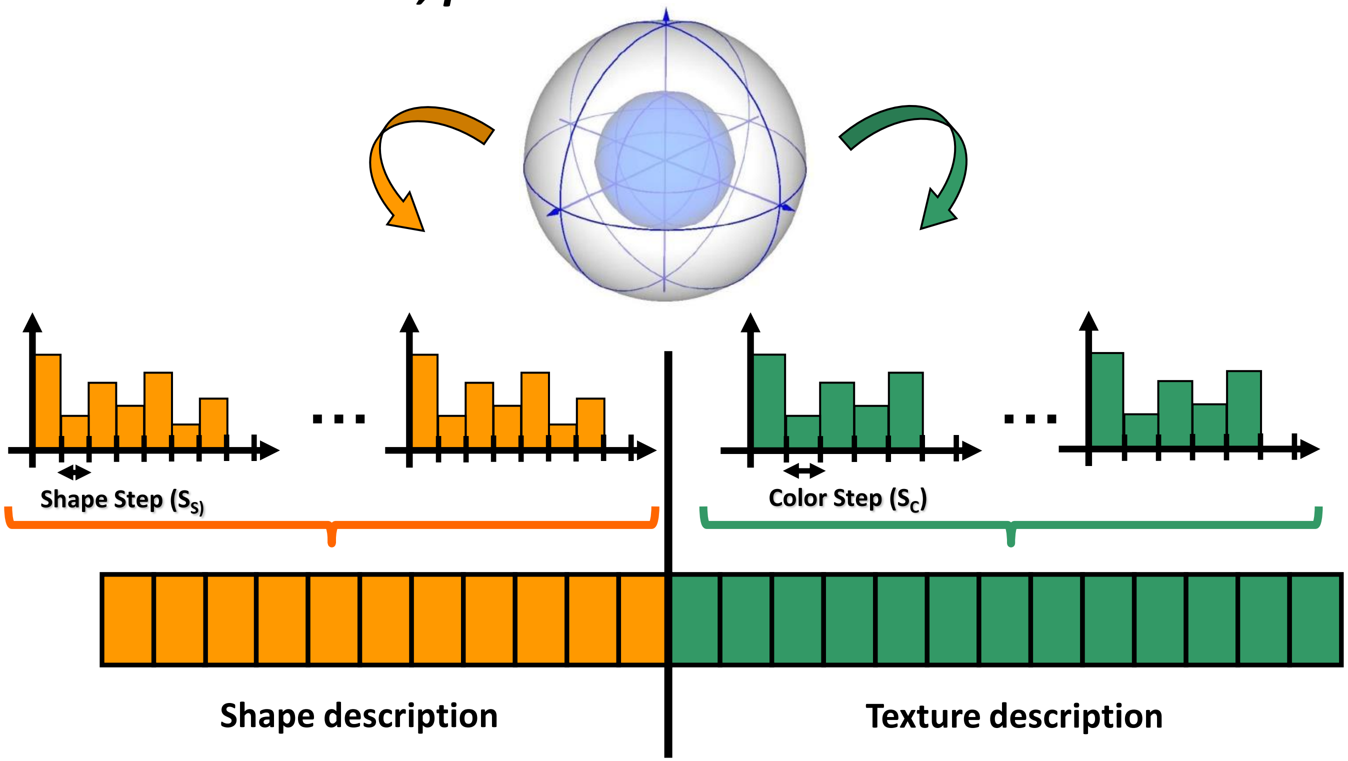


# SHOT for RGB-D data



- SHOT for RGB-D data [Tombari11] deploys
  - Shape, as the SHOT descriptor
  - Texture, as histograms in the *Lab* space
  - Pairs of *Lab* triplets (center point and its neighbor) can be compared using specific metrics (CIE94, CIE2000, ..), although the L1-norm proved to be a good trade-off
- ***pcl::SHOTColorEstimation, pcl::SHOTColorEstimationOMP***



# Code Example: descriptors

pointcloudlibrary

```
pcl::PointCloud<pcl::SHOT352>::Ptr descriptors (new
           pcl::PointCloud<pcl::SHOT352>());

pcl::SHOTEstimationOMP<PointType, NormalType, DescriptorType> describer;

describer.setRadiusSearch (support_radius);
describer.setInputCloud (keypoints);
describer.setInputNormals (normals);
describer.setSearchSurface (cloud);

describer.compute (*descriptors);
```

# Summing up..

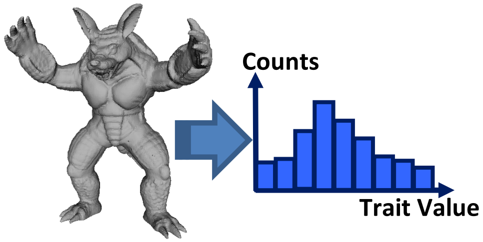
pointcloudlibrary

| Method | Category | Unique LRF | Texture |
|---|---|---|---|
| Struct. Indexing [Stein92] | Signature | No | No |
| PS [Chua97] | Signature | No | No |
| 3DPF [Sun01] | Signature | No | No |
| 3DGSS [Novatnack08] | Signature | No | No |
| KPQ [Mian10] | Signature | No | No |
| 3D-SURF [Knopp10] | Signature | Yes | No |
| SI [Johnson99] | Histogram | RA | No |
| LSP [Chen07] | Histogram | RA | No |
| 3DSC [Frome04] | Histogram | No | No |
| ISS [Zhong09] | Histogram | No | No |
| USC [Tombari10] | Histogram | Yes | No |
| PFH [Rusu08] | Histogram | RA | No |
| FPFH [Rusu09] | Histogram | RA | No |
| Tensor [Mian06] | Histogram | No | No |
| RSD [Marton11] | Histogram | RA | No |
| HKS [Sun09] | Other | - | No |
| MeshHoG [Zaharescu09] | Hybrid | Yes | Yes |
| SHOT [Tombari10] | Hybrid | Yes | Yes |

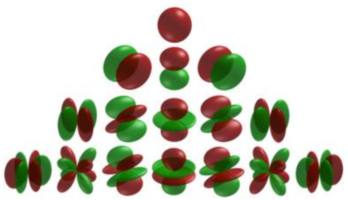
# Global descriptor taxonomy

pointcloudlibrary

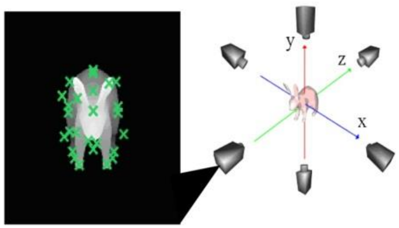
- Taxonomy for global descriptors [Akgul09]
- **Histogram-based:** accumulators of local or global features
  - Robustness, paid off with less descriptivness
  - **Shape Distributions [Osada02],** 3D Shape Histograms [Ankerst99], Orientation Histograms [Horn84], **Viewpoint Feature Histogram (VFH) [Rusu10]**, **Clustered-VFH** [Aldoma11], OUR-CVFH [Aldoma12]



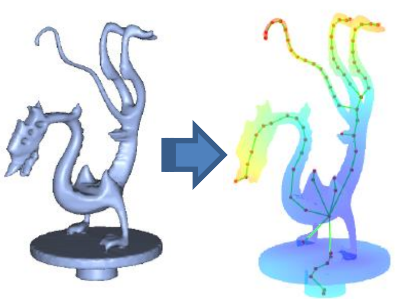

- **Transform-based:** Transform geometric information in a domain where representation is compact and invariant
  - Compact descriptors by retaining only a subset of (eg. the first) coefficients
  - 3D Fourier Transform [Dutagaci05], Angular Radial Tr. [Ricard05], 3D Radon Tr. [Daras04], **Spherical Harmonics** [Kazhdan03], wavelets [Laga06]



- **2D view-based:** 3D surface is transformed into a set of 2D projections (range maps)
  - 2D image descriptors are computed on each 2D view
  - Fourier descriptors [Vranic 04], Zernike moments [Chen03], SIFT [Ohbuchi08], SURF, ..

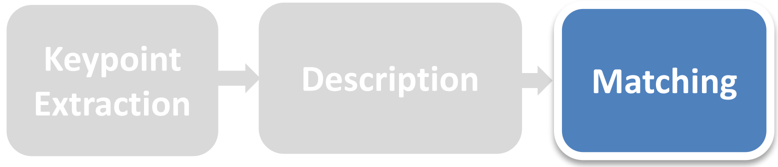



- **Graph-based:** A graph is built out of the surface
  - Transform the graph into a vector-based numerical description
  - topology-based[Hilaga01], Reeb graph[Tung05], skeleton-based[Sundar03]



# Descriptor matching






- Problem: find the kNN of a n-dimensional query vector **q** within a set of m candidates (same size)
  - Variant: find all neighbors within an hypersphere of radius **r** centered on **q**
- To speed up the brute force, fast indexing schemes
  - Kd-tree [Freidman77]
  - Hierarchical k-means tree [Fukunaga75]
  - Locality Sensitive Hashing (LSH) [Andoni06]
- Kd-tree slows down at high dimensions (too many nodes, long exploration time), need for approximate kd-tree search
  - Best Bin First [Beis97]
  - Randomized kd-tree [Silpa-Anan08]
  - FLANN [Muja09]
- Example: pcl::KdTreeFLANN<pcl::SHOT352> matcher; (in *pcl_kdtree* module)

  (also have a look at pcl::search::FlannSearch)

# Acknowledgements




- Thanks to: Samuele Salti, Aitor Aldoma