Keypoints and Features

Federico Tombari
CGLibs, Pisa
June 4, 2013
A feature..what?

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 keypoints are

- **Distinctive**, i.e. suitable for effective description and matching (*globally definable*)
- **Repeatable** with respect to point-of-view variations, noise, etc... (*locally definable*)

Usually scale-invariance is not an issue (but better if each feature is extracted together with its characteristic scale)

*Distinctiveness vs. repeatability*
• (for now) a small set of detectors specifically proposed for 3D point clouds and range maps
  • Intrinsic Shape Signatures (ISS) [Zhong ICCV09]
  • NARF [Steder ICRA11]
  • Uniform Sampling (basically a voxelGrid, where selected points are a subset of the input cloud)

• 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
In the context of most PCL applications scale is 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)
- **Adaptive-scale detectors**: specific scale-space analysis to detect salient structures at multiple scales, associating each keypoint a **characteristic scale**

Need for performance assessment

- Locality repeatability / Quantity, Scale repeatability, Efficiency
- **www.vision.deis.unibo.it/keypoints3d**
Exploits the covariance matrix

\[
M(p_i) = \frac{1}{k} \sum_{j=1}^{k} \rho_i (p_j - p_i)^T (p_j - p_i)
\]

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

\[
\frac{\lambda_2(p)}{\lambda_1(p)} < T h_{12} \land \frac{\lambda_3(p)}{\lambda_2(p)} < T h_{23}
\]

Saliency is the magnitude of the third eigenvalue \( \rho(p) \equiv \lambda_3(p) \)

Non-Maxima Suppression (NMS) over saliency

It includes only points with large variations along each principal direction

“Winner” of PCL 3D detector evaluation in [Filipe 2013]
Intrinsic Shape Signatures

Federico Tombari

Keypoints and Features
Example

UNIFORM SAMPLING

pcl::PointCloud<int> indices;
pcl::UniformSampling<pcl::PointXYZ> uniform_sampling;
uniform_sampling.setInputCloud (cloud);
uniform_sampling.setRadiusSearch (0.05f); //the 3D grid leaf size
uniform_sampling.compute (indices);

ISS

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);
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

```cpp
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);
```
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..)
Spin Image descriptor [Johnson99] is arguably the most popular 3D local descriptor.

2D histograms accumulating points by spinning around a repeatable axis (*normal*).

- 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).
3D/Unique Shape Contexts

- **3DSC** [Frome ECCV04]: extension of the Shape Contexts approach [Belongie et al. PAMI02] to the 3D domain *(pcl::ShapeContext3DEstimation)*
- Each point is accumulated in the 3D bin it falls in, being weighted proportionally to the local point cloud density around the bin and to the bin volume
- No unique local Reference Frame -> L descriptions for each feature (*L*: number of azimuth bins)

![2D Log-polar histogram of 2D points](image)

- **Unique Shape Context (USC)** [Tombari 3DOR10]: a unique local RF is plugged in to orient univocally the 3D grid *(pcl::UniqueShapeContext)*
- Hence, only one description is needed for each feature point, decreasing the number of possible mismatches (spurious correspondences) during the matching stage.
PFH [Rusu08] computes 3 values for each pair in the neighbourhood
- Complexity $O(k^2)$, 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 $n_s$ and the vector $(p_t - p_s)$ $v = n_s \times (p_t - p_s)$
  - 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

\[
FPFH\left(p_i\right) = SPFH\left(p_i\right) + \frac{1}{k} \sum_{j=1}^{k} \frac{1}{\omega_j} SPFH\left(p_j\right)
\]

- \textit{pcl::FPFHEstimation, pcl::FPFHEstimationOMP}
Signature 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 $\theta_i$ 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`

---

**Shape Step** (S\(_S\))

**Color Step** (S\(_C\))

**Shape description**

**Texture description**
Code Example: descriptors

```cpp
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..

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

- Taxonomy for global descriptors [Akgul09]

- **Histogram-based**: accumulators of local or global features
  - Robustness, paid off with less descriptiveness
  - 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]
VFH

- Viewpoint Feature Histogram [Rusu 10]
  - Each 3D model is rendered into different views
  - Each view provides one descriptor
    - Explicitly encodes the viewpoint from where the surface was captured/sensed
  - Based on Point Feature Histogram (PFH)
  - For each point pair \((p_i, p_c)\):
    - Compute a LRF for the centroid
      \[
      \begin{align*}
      u &= n_c \\
      v &= \frac{p_i - p_c}{\|p_i - p_c\|} \times u \\
      w &= u \times v
      \end{align*}
      \]
    - \(\alpha = \arccos(v \cdot n)\)
    - \(\phi = \arccos\left(u \cdot \frac{p_i - p_c}{\|p_i - p_c\|}\right)\)
    - \(\theta = \text{atan2}(w \cdot n, u \cdot n)\)
VFH (2)

- Descriptor is built with:
  - 3 “PFH” angular values \((\alpha, \theta, \Phi)\) wrt. centroid (45 bins each)
  - 1 shape distribution-like component wrt. centroid (45 bins):
    \[ SDC = \frac{(p_c - p_i)^2}{\max((p_c - p_i)^2)} \]
  - 1 angular value (angle between normal and central view direction – \(\alpha\)) (128 bins)

- \texttt{pcl::VFHEstimation}
The reference frame from VFH is sensitive to missing parts in the surface.

Clustered VFH (CVFH) [Aldoma 11]
- Perform a further smooth region segmentation on each view
- Apply a VFH descriptor on each connected component (cluster) – no normalization to encode the real size of the object

VFH, CVFH et al. still present invariance (ambiguity) on the roll angle.

Camera Roll Histogram to determine a full 6DOF pose
- distribution of normal angles of all points projected on the camera plane
- «shift» along roll angle computed by matching CRHs

`pcl::CVFHEstimation`
Improvement [Aldoma DAGM13] to CVFH with
- More robust to missing data
- A descriptor for each smooth region composing the descriptor

One Local-Global Reference Frame for each cluster
- Locally: compute principal directions
- Globally: sign disambiguation

RF splits space in octants:
- For each octant, Shape distribution (D1) (13 bins)

Final descriptor:
- Octant-based shape dist. and color hist.
- “Global” normal distribution (CVFH) (45x3 els.)
- Viewpoint (CVFH) (64 els.) (half wrt. VFH/CVFH)
- Overall size: 13x8 + 199 = 303

\texttt{pcl::OURCVFHEstimation} (currently only trunk)
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`)
Thanks to: Samuele Salti, Aitor Aldoma