3D Point Cloud Classification, Segmentation, and Normal estimation using Modified Fisher Vector and CNNs

Yizhak (Itzik) Ben-Shabat¹
Michael Lindenbaum²
Anath Fischer¹

Technion - ¹Mechanical Engineering and ²Computer Science
Outline

- Point clouds
- Point clouds and CNNs – why the connection is challenging?
- Fisher Vectors
- Representing Point clouds with Fisher vectors
- Deep learning with Fisher vectors input
- Three applications :
  - Classification
  - Semantic segmentation
  - Scale selection & Normal estimation
3D data acquisition

Direct 3D sensors are available:
- LiDAR
- RGBD Camera

and provide a set of 3D points = point cloud

Point clouds from KITTI dataset and NYU Depth V2 dataset
Task 1– Point Cloud Classification

<table>
<thead>
<tr>
<th>Input point cloud</th>
<th>Output Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mug</td>
<td>Black box</td>
</tr>
<tr>
<td>Table</td>
<td>Black box</td>
</tr>
<tr>
<td>Car</td>
<td>Black box</td>
</tr>
</tbody>
</table>
Task 2– Point Cloud Part Segmentation

- Point on plane tail
- Point on plane body
- Point on plane wing
- Point on ?
Task 3 – Point Cloud Normal Estimation
The preferred tool: Convolutional Neural Networks

- In 2D: Deep CNNs revolutionized image analysis
  - Convolutional neural nets learn shared weights filters
  - The input (Image) is specified on a grid structure
  - Number of pixels in the input image is fixed

How can we use them for analyzing 3D point cloud?

AlexNet Architecture
Challenges

- How can we use the power of CNNs with 3D point cloud data?

- Representing the input is not trivial:
  - A point cloud is not a natural input to a CNN
    - Number of input points is not constant
    - Data is unstructured (no a signal on a grid)
    - Linear ordering cannot reflect spatial proximity
    - No way for unique ordering (permutations)
  - Other challenges with point clouds
    - Missing data, noise, rotations

![Point Cloud Geometry](image)

![Point Cloud Representation](image)
Voxelization approach

The straightforward approach: transform the point cloud into a voxel grid by rasterizing and use 3D CNNs

A choice between
- Large memory cost and Slow processing time
- OR No Limited spatial resolution Quantization artifacts

A sparsely populated grid which seems un-natural

*Image source - AOI-Matlab Voxelizer*
Multi-View approach

- The multi-view approach: project multiple views to 2D and use

Direct point cloud approach (PointNet)

Direct approach:
- Process each point separately
- Pool using an order independent (symmetric) function

Previous work

Recent reported classification performance

*Accuracy is reported on the ModelNet40 Dataset
What are Fisher Vectors?

Context – Kernel based learning & classification

- \{ (X_i, S_i) \} - examples = \{ vector description, class label \}
- K(X_i, X_j) - an affinity function (kernel)
- The classifier uses learned weight \( \lambda_i \) and is
  \[ \hat{S} = \text{sign} \left( \sum_i S_i \lambda_i K(X_i, X) \right) \]

- Which kernel function is best?
- Every valid kernel function may be written as an inner product between feature vectors
  \[ K(X_i, X_j) = \phi_{X_i}^T \phi_{X_j} \] (Mercer theorem)

Which feature vectors are best?

Fisher Vectors

Deriving feature vectors using the class distributions

- Suppose you know the generative model (a distribution of the vector description)
- Then use the (simplified) Fisher score vector

\[ \phi_X = \nabla_\theta \log P(X|\theta) \]

Theoretical justification:

- A differential extension of a discrimination task: consider two similar classes
  \[ P(X|\theta_1), P(X|\theta_{-1}) \quad s.t. \quad \theta_1 \approx \theta_{-1} \approx \theta \]
- Then, use Taylor expansion
  \[ \log P(X|\theta_S) = \log P(X|\theta) + (\theta_S - \theta)^T \phi_X \]
- \( \rightarrow \) there is a linear classifier (in Fisher score space) which is consistent with maximum likelihood or MAP decision.
- Learning a linear, logistic regression, classifier gives a kernel classifier with
  \[ K(X_i, X_j) = \phi_{X_i}^T \phi_{X_j} \]
- \( \rightarrow \) using this kernel makes decisions that are as good as MAP, asymptotically

Fisher Vectors

For a set of independent observations \( \bar{X} = \{X_1, X_2, \ldots, X_n\} \)

\[
\phi_{\bar{X}} = \nabla_\theta \log P(\bar{X}|\theta) = \nabla_\theta \log \prod_i P(X_i|\theta) = \sum_i \phi_{X_i}
\]
Fisher Vectors – Application to 2D object recognition

1. Characterize the image:
   a. Describe the image by a set of dense SIFT descriptors
   b. Assume the SIFTs are generated by a Gaussians mixture mode,
   c. Learn the GMM using EM (from a large image set).
   d. Re-describe the image by a single Fisher vector

2. Use the feature vectors for learning and classification (using, say, SVM).

- The GMM

\[ u_{\lambda}(p) = \sum_{k=1}^{K} w_k u_k(p) \]

\[ u_k(p) = \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} \exp \left\{ -\frac{1}{2}(p - \mu_k)' \Sigma_k^{-1}(p - \mu_k) \right\} \]

- The model parameters \( w_k, \mu_k, \Sigma_k \)

Mixture weights

Centers

Covariance matrix

Perronnin et al. "Improving the fisher kernel for large-scale image classification." ECCV 2010
Here: A Gaussian Mixture Model (GMM) on a 3D grid

\[
u_p(p) = \sum_{k=1}^{K} w_k u_k(p)
\]

\[
u_k(p) = \frac{1}{(2\pi)^D |\Sigma_k|^{1/2}} \exp \left\{ -\frac{1}{2} (p - \mu_k)' \Sigma_k^{-1} (p - \mu_k) \right\}
\]

- The parameters
  \( w_k, \mu_k, \Sigma_k \)
  Mixture weights  Centers  Covariance matrix

- Here we use **spherical Gaussians** on a **coarse uniform grid**.
  (diagonal covariance matrix with equal values)

\[
m \times m \times m \quad \sigma_k = \frac{1}{m} \quad m \in [3, 16]
\]

- Uniformity is enforced to achieve space invariance as input to CNNs
Describing a Point Cloud with Fisher Vectors

- Characterizes data samples by their deviation from a GMM generative model.
- Computes the gradients (FVs) of the log likelihood at the cloud points w.r.t model parameters
- Aggregates the gradients by averaging (invariant to point ordering)
- Constant size output
- Theoretically justified

In general
\[ \mathcal{G}_\lambda^X = \sum_{t=1}^{T} L_\lambda \nabla_\lambda \log u_\lambda(p_t) \]
Vector of derivatives w.r.t model parameters

Here
\[ \mathcal{G}_{FV\lambda}^X = \left( \mathcal{G}_{\alpha_1}^X, \ldots, \mathcal{G}_{\alpha_k}^X, \mathcal{G}_{\mu_1}^X, \ldots, \mathcal{G}_{\mu_k}^X, \mathcal{G}_{\sigma_1}^X, \ldots, \mathcal{G}_{\sigma_k}^X \right) \]
\[ w_k = \frac{\exp(\alpha_k)}{\sum_{j=1}^{K} \exp(\alpha_j)} \]
- Normalize derivatives by sample size
Each Gaussian “generates” a vector which represents all the data w.r.t it.

\[ g_{\alpha_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} (\gamma_t(k) - w_k) \]

Derivative w.r.t Gaussian weights

\[ g_{\mu_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \left( \frac{p_t - \mu_k}{\sigma_k} \right) \]

Derivative w.r.t Gaussian expected value (centers)

\[ g_{\sigma_k} = \frac{1}{\sqrt{2w_k}} \sum_{t=1}^{T} \gamma_t(k) \left[ \left( \frac{p_t - \mu_k}{\sigma_k} \right)^2 - 1 \right] \]

Derivative w.r.t Gaussian stds

\[ \gamma_t(k) = \frac{w_k u_k(p_t)}{\sum_{j=1}^{K} w_j u_j(p_t)} \]
Illustration: One Point, One Gaussian, FV

Each Gaussian “generates” a vector which represents all the data w.r.t it

\[ g_{\alpha_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} (\gamma_t(k) - w_k) \]

Derivative w.r.t Gaussian weights

\[ g_{\mu_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \left( \frac{p_t - \mu_k}{\sigma_k} \right) \]

Derivative w.r.t Gaussian expected value (centers)

\[ g_{\sigma_k} = \frac{1}{\sqrt{2w_k}} \sum_{t=1}^{T} \gamma_t(k) \left[ \frac{(p_t - \mu_k)^2}{\sigma_k^2} - 1 \right] \]

Derivative w.r.t Gaussian stds

\[ \gamma_t(k) = \frac{w_k u_k(p_t)}{\sum_{j=1}^{K} w_j u_j(p_t)} \]
Each Gaussian “generates” a vector which represents all the data w.r.t it

$$
\gamma_t(k) = \frac{w_k u_k(p_t)}{\sum_{j=1}^K w_j u_j(p_t)}
$$

Derivative w.r.t Gaussian weights

$$
\mathcal{G}_{\alpha_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^T (\gamma_t(k) - w_k)
$$

Derivative w.r.t Gaussian expected value (centers)

$$
\mathcal{G}_{\mu_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^T \gamma_t(k) \left( \frac{p_t - \mu_k}{\sigma_k} \right)
$$

Derivative w.r.t Gaussian stds

$$
\mathcal{G}_{\sigma_k} = \frac{1}{\sqrt{2w_k}} \sum_{t=1}^T \gamma_t(k) \left[ \frac{(p_t - \mu_k)^2}{\sigma_k^2} - 1 \right]
$$
3D \textbf{modified} Fisher vector (3DmFV) representation

- Uniform grid GMM
- Additional permutation invariant ("symmetric") function (min, max)

\[
\mathcal{g}_{3DmFV\lambda}^X = \begin{bmatrix}
\sum_{t=1}^{T} L_{\lambda} \nabla_{\lambda} \log u_{\lambda}(p_t) \bigg|_{\lambda=\alpha,\mu,\sigma} \\
\max_t (L_{\lambda} \nabla_{\lambda} \log u_{\lambda}(p_t)) \bigg|_{\lambda=\alpha,\mu,\sigma} \\
\min_t (L_{\lambda} \nabla_{\lambda} \log u_{\lambda}(p_t)) \bigg|_{\lambda=\mu,\sigma}
\end{bmatrix}
\]
3DmFV Representation

*No normalization for visualization purposes*
3DmFV Representation

*No normalization for visualization purposes*
3DmFV Representation

*No normalization for visualization purposes*
3DmFV Representation - Example
• Images are used for visualization purposes only.

• Every column corresponds to the gradient components associated with one Gaussian (and one 3D spatial position)

• The full descriptor is a 4D structure.

Point cloud reconstruction from FV

- FV is continuous on the point set (unlike voxels)
- Reconstructing from FV: simple cases
  - FV calculated relative to a single Gaussian representing a single point – analytic reconstruction of the point:
    \[ p_k = \sigma_k g_{\mu_k}^X + \mu_k \]
  - FV calculated relative to a single Gaussian representing multiple points on one plane – analytic reconstruction of the plane:
    \[
    \begin{align*}
    a &= \frac{g_{\mu x}}{\|g_{\mu}\|}, \\
    b &= \frac{g_{\mu y}}{\|g_{\mu}\|}, \\
    c &= \frac{g_{\mu z}}{\|g_{\mu}\|}, \\
    \rho &= \sigma \|g_{\mu}\| \sqrt{w}
    \end{align*}
    \]

Under the assumption of sharply peaked \( \gamma_t(k) \)
Point cloud reconstruction from FV

- Reconstructing points from FV:
  - FV consisting of multiple Gaussians and multiple points – reconstruction using a Deep decoder network
Benchmark Dataset

- **Modelnet40**
  - ~12.5K CAD models (triangle mesh)
  - 40 man-made object categories
  - ~10K for training
  - ~2.5K for testing

- **Modelnet10**
  - ~5K CAD models (triangle mesh)
  - 10 man-made object categories
  - ~4K for training
  - ~1K for testing

Training details

- **Number of points:** 2048 (for best performance)
- **Point cloud manipulation:** Centered around the origin and scaled to fit a cube of edge length 2.
- **Data augmentation:**
  - Random anisotropic scaling (range [0.66, 1.5])
  - Random translation (range: [-0.2, 0.2])
  - Gaussian noise (std of 0.01)
- Implemented in Tensorflow and trained on Nvidia Titan Xp GPU
- **Time:** ~7h
- **Optimizer:** Adam
- **Learning rate:** 0.001
- **Learning rate decay:** 0.7 every 20 epochs
- **Activation function:** ReLU
- **Dropout of 0.7 keep ratio between each FC layer**
- **Batch Size:** 64
# Classification Accuracy Results

## Voxel and Multi-view methods

<table>
<thead>
<tr>
<th>Method</th>
<th>ModelNet10</th>
<th>ModelNet40</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVCNN [28]</td>
<td>-</td>
<td>90.1</td>
</tr>
<tr>
<td>3DShapeNets [33]</td>
<td>83.5</td>
<td>77.32</td>
</tr>
<tr>
<td>VoxNet [18]</td>
<td>92.0</td>
<td>83.0</td>
</tr>
<tr>
<td>VRN [4]</td>
<td>93.6</td>
<td>91.33</td>
</tr>
<tr>
<td>FusionNet [11]</td>
<td>93.1</td>
<td>90.8</td>
</tr>
<tr>
<td>3DmFV+VoxNet</td>
<td>94.3</td>
<td>88.5</td>
</tr>
<tr>
<td><strong>Our 3DmFV-Net</strong></td>
<td><strong>95.2</strong></td>
<td><strong>91.4</strong></td>
</tr>
</tbody>
</table>

## Point methods

<table>
<thead>
<tr>
<th>Method</th>
<th>ModelNet10</th>
<th>ModelNet40</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointNet [22]</td>
<td>-</td>
<td>89.2</td>
</tr>
<tr>
<td>PointNet++ [24]</td>
<td>-</td>
<td>90.7</td>
</tr>
<tr>
<td>Kd-network [14]</td>
<td>93.3</td>
<td>90.6</td>
</tr>
<tr>
<td>3DmFV+VoxNet</td>
<td>94.3</td>
<td>88.5</td>
</tr>
<tr>
<td><strong>Our 3DmFV-Net</strong></td>
<td><strong>95.2</strong></td>
<td><strong>91.4</strong></td>
</tr>
</tbody>
</table>

*Note: Performance is measured in equivalent experimental conditions i.e. single architecture, 1024 points*
3DmFV parameter influence

- Grid or not?
- Grid size
- Standard deviation ($\sigma$)
- Symmetric function

<table>
<thead>
<tr>
<th>Rep.</th>
<th>ML + LinCls</th>
<th>ML + NonLinCls</th>
<th>Grid + NonLinCls</th>
</tr>
</thead>
<tbody>
<tr>
<td>FV</td>
<td>58.4</td>
<td>82.8</td>
<td>84.5</td>
</tr>
<tr>
<td>3DmFV-ss</td>
<td>58.8</td>
<td>85.0</td>
<td>84.4</td>
</tr>
<tr>
<td>3DmFV-min</td>
<td>67.7</td>
<td>87.7</td>
<td>86.1</td>
</tr>
<tr>
<td>3DmFV-max</td>
<td>68.6</td>
<td>87.4</td>
<td>85.3</td>
</tr>
<tr>
<td>3DmFV</td>
<td>76.8</td>
<td>88.0</td>
<td>87.7</td>
</tr>
</tbody>
</table>

$m = 8$
Run-time

Real-time performance
Theoretical time complexity of $O(TK)$ is validated empirically

*Results are averaged over 2448 point clouds subdivided into batches of 16 on a Titan Xp GPU.*
Robustness

- Point deletion: Uniform deletion, focused region deletion
- Outlier points: Random in space
Robustness

Perturbation noise

Random rotation
Classification Failure Cases

Many failures occur in specific pairs:
- Table – desk
- Dresser – night stand
- Flower pot – plant
Classification Failure Cases

Plant  Flower pot  Failure cases (plant classified as flower pot)

Table  Desk  Failure cases (table classified as desk)
Results on Sydney Dataset - Outdoor

- LiDAR scans
- 14 object classes
- 588 total objects (subdivided into 4 folds)
- Imbalanced classes

3DmFV-Net – Part segmentation

Part Segmentation Qualitative Results

- ShapeNet part dataset
- Contains ~17K point clouds with 50 annotated parts from 16 categories.
- Imbalanced dataset
Part Segmentation Quantitative Results

Evaluation metric (mean IoU)

\[ \text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \]

<table>
<thead>
<tr>
<th>method</th>
<th>mean</th>
<th>aero</th>
<th>bag</th>
<th>cap</th>
<th>car</th>
<th>chair</th>
<th>ear</th>
<th>guitar</th>
<th>knife</th>
<th>lamp</th>
<th>laptop</th>
<th>motor</th>
<th>bike</th>
<th>mug</th>
<th>pistol</th>
<th>rocket</th>
<th>skate</th>
<th>board</th>
<th>table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yi [34]</td>
<td>81.4</td>
<td>81.0</td>
<td>78.4</td>
<td>77.7</td>
<td>75.7</td>
<td>87.6</td>
<td>61.9</td>
<td>92.0</td>
<td>85.4</td>
<td>82.5</td>
<td>95.7</td>
<td>70.6</td>
<td>91.9</td>
<td>85.9</td>
<td>53.1</td>
<td>69.8</td>
<td>75.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3DCNN [22]</td>
<td>79.4</td>
<td>75.1</td>
<td>72.8</td>
<td>73.3</td>
<td>70.0</td>
<td>87.2</td>
<td>63.5</td>
<td>88.4</td>
<td>79.6</td>
<td>74.4</td>
<td>93.9</td>
<td>58.7</td>
<td>91.8</td>
<td>76.4</td>
<td>51.2</td>
<td>65.3</td>
<td>77.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PointNet [22]</td>
<td>83.7</td>
<td>78.7</td>
<td>82.5</td>
<td>74.9</td>
<td>89.6</td>
<td>68.0</td>
<td>72.4</td>
<td>86.4</td>
<td>94.9</td>
<td>55.8</td>
<td>65.2</td>
<td>93.</td>
<td>91.2</td>
<td>57.9</td>
<td>72.8</td>
<td>80.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kd-Net [14]</td>
<td>77.2</td>
<td>79.9</td>
<td>71.2</td>
<td>80.9</td>
<td>68.8</td>
<td>88.0</td>
<td>72.4</td>
<td>88.9</td>
<td>86.4</td>
<td>79.8</td>
<td>94.9</td>
<td>55.8</td>
<td>86.5</td>
<td>79.3</td>
<td>50.4</td>
<td>71.1</td>
<td>80.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PointNet++ [24]</td>
<td><strong>85.1</strong></td>
<td>82.4</td>
<td>79.0</td>
<td><strong>87.7</strong></td>
<td><strong>77.3</strong></td>
<td>90.8</td>
<td>71.8</td>
<td>91.0</td>
<td>85.9</td>
<td><strong>83.7</strong></td>
<td>95.3</td>
<td><strong>71.6</strong></td>
<td><strong>94.1</strong></td>
<td>81.3</td>
<td><strong>58.7</strong></td>
<td><strong>76.4</strong></td>
<td><strong>82.6</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>84.3</td>
<td>82.0</td>
<td><strong>84.3</strong></td>
<td>86.0</td>
<td>79.9</td>
<td>73.9</td>
<td>90.8</td>
<td>85.7</td>
<td>92.6</td>
<td>95.2</td>
<td>66.0</td>
<td>94.0</td>
<td>82.6</td>
<td>51.5</td>
<td>73.5</td>
<td>81.8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Part Segmentation Results

<table>
<thead>
<tr>
<th>GT</th>
<th>Prediction</th>
<th>Difference</th>
<th>GT</th>
<th>Prediction</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Normal Estimation

Normal estimation algorithm
Previous work

Input point cloud

Extract subset
Previous work

Input point cloud
Extract subset
Fit a surface
Previous work

Input point cloud

Extract subset

Fit a surface
Previous work

Can we learn to select the best radius?

Input point cloud

Extract subset

Fit a surface
Nesti-Net pipeline

Scale

Points

Multi scale point statistics (MuPS)

\[ r_1 \]

\[ r_2 \]

\[ \cdots \]

\[ r_n \]
Nesti-Net pipeline

Multi scale point statistics (MuPS)

\[ \tilde{X}_1(r_1) \]

\[ \tilde{X}_2(r_2) \]

\[ \tilde{X}_n(r_n) \]
Nesti-Net pipeline

Multi scale point statistics (MuPS)

Mixture of Experts (MoE)

3D CNN Expert 1

3D CNN Expert 2

3D CNN Expert n

Scale Manager Network

$\hat{x}_{3DmFV}^{1}$

$\hat{x}_{3DmFV}^{2}$

$\hat{x}_{3DmFV}^{n}$

$\mathcal{M}_i$ for $i \in \{1, \ldots, n\}$

$N_{\text{argmax}}(q_i)$

Nesti-Net pipeline

Loss: \[ L_{MoE} = \sum_{i=1}^{n} q_i D_N = \sum_{i=1}^{n} q_i \frac{\|N_i \times N_{GT}\|}{\|N_i\| \cdot \|N_{GT}\|} \]
## Quantitative results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>scale</td>
<td>small</td>
<td>med</td>
<td>large</td>
<td>small</td>
</tr>
<tr>
<td>None</td>
<td>6.99</td>
<td>8.31</td>
<td>12.29</td>
<td>16.77</td>
<td>7.60</td>
</tr>
<tr>
<td>Noise</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>σ = 0.00125</td>
<td>10.11</td>
<td>12.00</td>
<td>12.87</td>
<td>16.87</td>
<td>12.36</td>
</tr>
<tr>
<td>σ = 0.006</td>
<td>17.63</td>
<td>40.36</td>
<td>18.38</td>
<td>18.94</td>
<td>41.39</td>
</tr>
<tr>
<td>σ = 0.012</td>
<td>22.28</td>
<td>52.63</td>
<td>27.5</td>
<td>23.5</td>
<td>53.21</td>
</tr>
<tr>
<td>Density</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gradient</td>
<td>9.00</td>
<td>9.14</td>
<td>12.81</td>
<td>17.26</td>
<td>8.49</td>
</tr>
</tbody>
</table>

Table 1. Comparison of the RMS angle error for unoriented normal vector estimation of our Nesti-Net method to classic geometric methods (PCA [12], Jet [7]) with three scales, and deep learning methods (PCPNet [11], HoughCNN [6]).
Qualitative results
Error visualization

18.25

Nesti-Net  
16.93

PCA (best)  
21.12

Jet (best)  
20.08

PCPNet  
23.9

0  
18.25

19.57

20.03

23.19

60
Scale prediction results
Normal estimation results on scanned data

We introduce a new hybrid representation for 3D point clouds (3DmFV) which is structured, order and sample size independent. It enables the use of CNNs with point cloud data.

3DmFV offers an efficient way for encoding global and local spatial distributions.

We design a new deep CNN architecture (3DmFVNet) based on this representation and use it for point cloud classification, obtaining state of the art results in real-time.

We extend the 3DmFV-Net to part segmentation of point clouds and to Normal Estimation.

Note: These best results are obtained without “end to end” training.
Questions?

For code and tutorials visit www.itzikbs.com