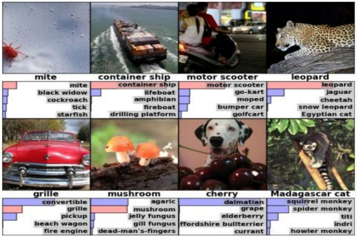
FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis

Nitika Verma, Edmond Boyer, Jakob Verbeek
INRIA, Grenoble, France

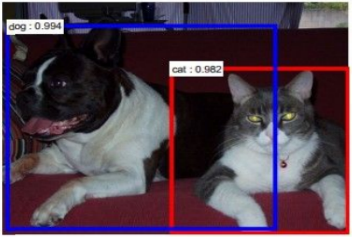
  

Computer Vision and Pattern Recognition, 2018

ConvNets are everywhere for 2D images



Classification



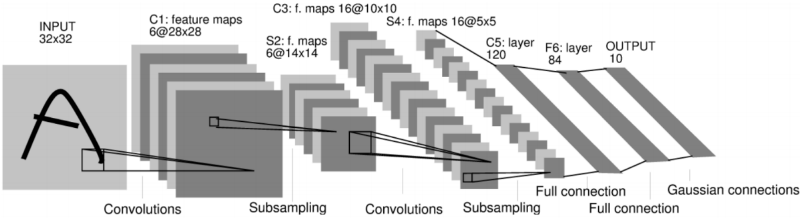
Detection

Retrieval

Segmentation

[Krizhevsky et al., 2012, Farabet et al., 2013, Ren et al., 2015, Gordo et al., 2016]

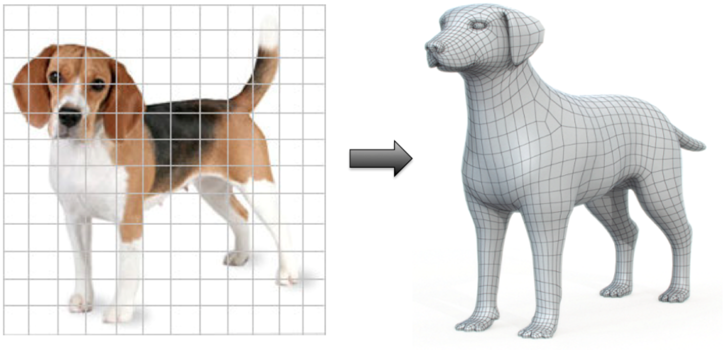
Convolutional Neural Networks (CNNs)



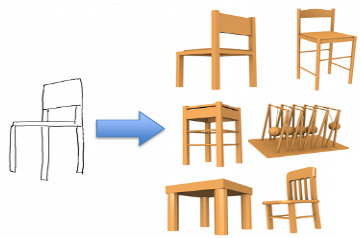
[LeCun et al., 1989]

How to generalize ConvNets to graph-structured data?

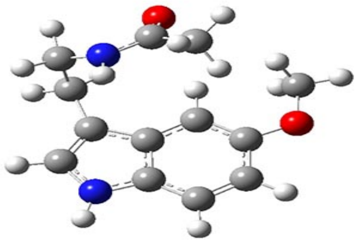
- RGB over regular grid of pixels
- XYZ(+RGB) over irregular graph of vertices



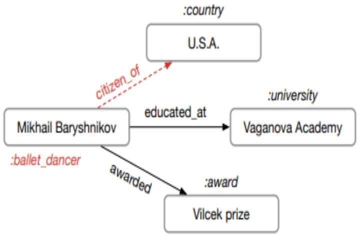
Applications



3D shape data



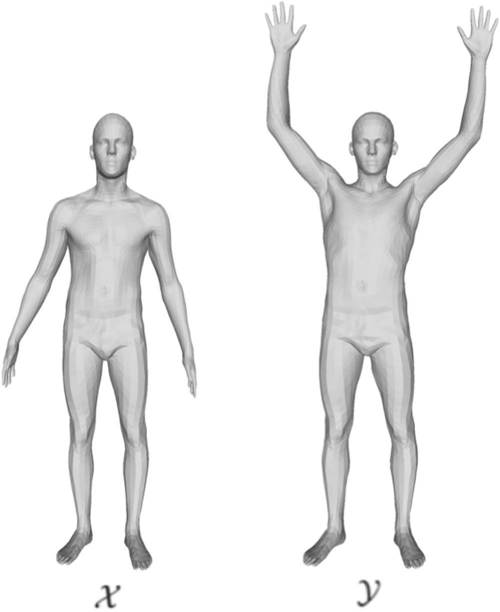
molecular graphs



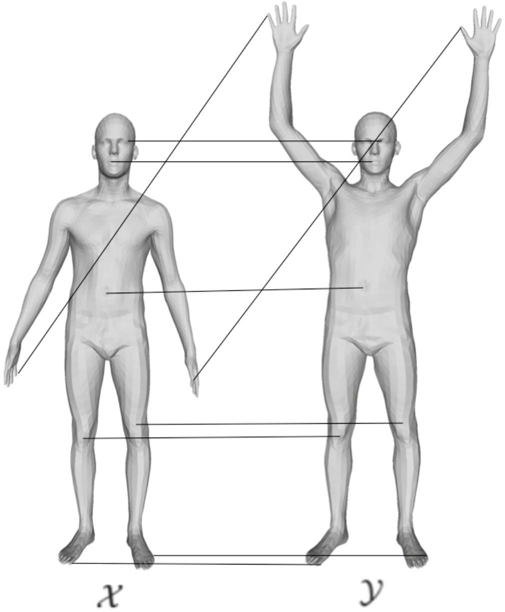
knowledge graphs

social network analysis

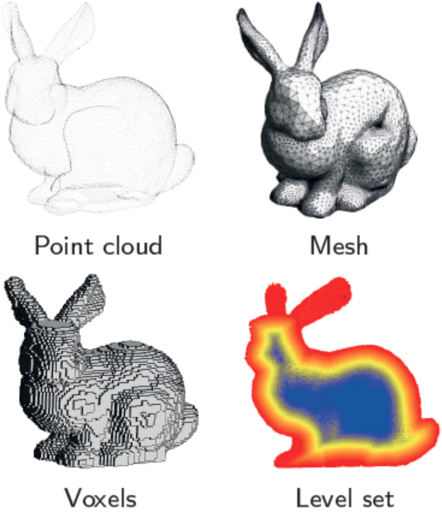
Problem: Shape correspondence



Problem: Shape correspondence



Representations for 3D shape data



Bronstein et al. 2016

Extrinsic *vs.* Intrinsic representations

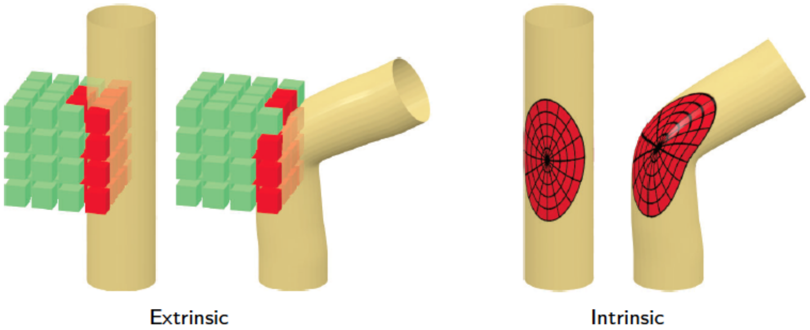
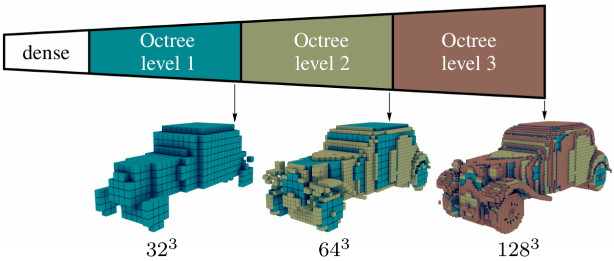


Figure from [Boscaini et al., 2016]

Extrinsic representation: Voxel grids

- Occupancy grids on input and/or output
- Quantize space rather than shape, lots of empty space
- 3D convolutions over grid, limited scalability
  - Sparse convolutions over input [Graham et al., 2018]
  - Octtrees on input and/or output [Tatarchenko et al., 2017]

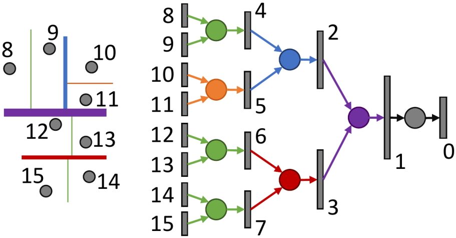


Extrinsic representation: Point clouds

- Avoid quantization, ignore (most) structure
- PointNet [Qi et al., 2017]
  - Local per-point processing (1×1 convolution)
  - Global max-pooling for global shape properties

Extrinsic representation: Point clouds

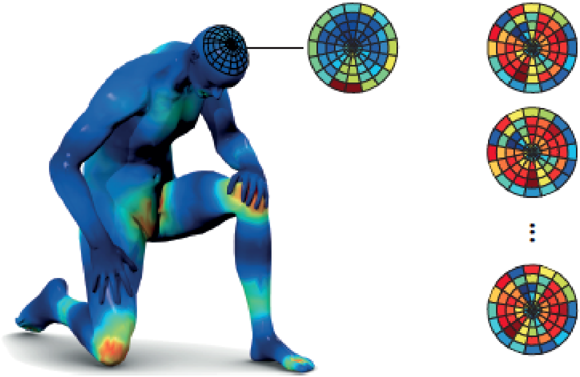
- Avoid quantization, ignore (most) structure
- PointNet [Qi et al., 2017]
  - Local per-point processing (1×1 convolution)
  - Global max-pooling for global shape properties
- Kd-Networks [Klokov and Lempitsky, 2017]
  - Propagate features across Kd-Tree over point cloud
  - Share parameters over branches with same split direction



Intrinsic representation: Geodesics over 3D mesh data

- Local geodesic polar coordinates
  [Masci et al., 2015, Boscaini et al., 2016]

$$\mathbf{u}(x, y) = (\rho(x, y), \theta(x, y))$$

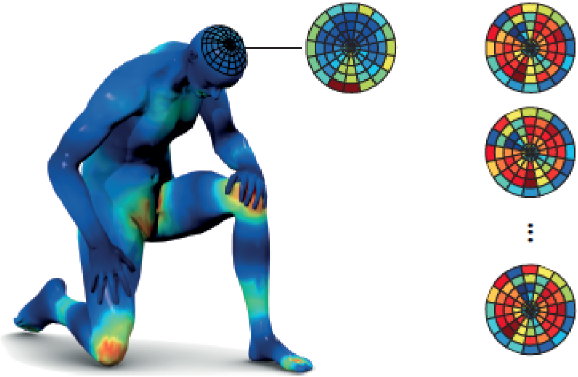


Intrinsic representation: Geodesics over 3D mesh data

- Local geodesic polar coordinates
  [Masci et al., 2015, Boscaini et al., 2016]

$$\mathbf{u}(x, y) = (\rho(x, y), \theta(x, y))$$

- Extract patch from mesh by "flattening" and interpolation

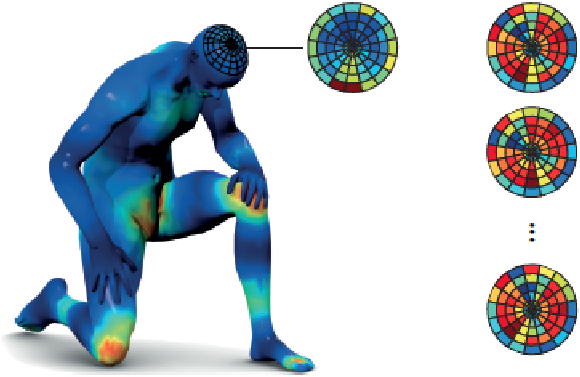


Intrinsic representation: Geodesics over 3D mesh data

- Local geodesic polar coordinates [Masci et al., 2015, Boscaini et al., 2016]

$$\mathbf{u}(x, y) = (\rho(x, y), \theta(x, y))$$

- Extract patch from mesh by "flattening" and interpolation
- Apply filter to local patch, max-pool over patch orientation

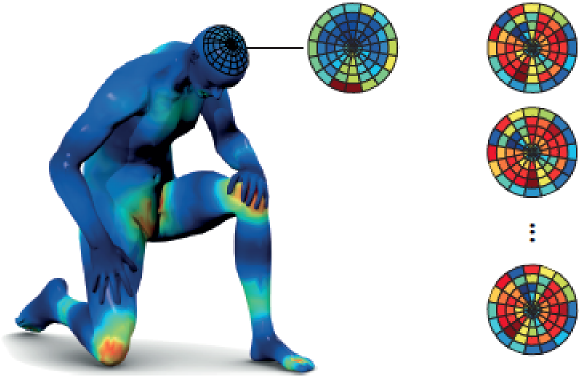


Intrinsic representation: Geodesics over 3D mesh data

- Local geodesic polar coordinates
  [Masci et al., 2015, Boscaini et al., 2016]

$$\mathbf{u}(x, y) = (\rho(x, y), \theta(x, y))$$

- Extract patch from mesh by "flattening" and interpolation
- Apply filter to local patch, max-pool over patch orientation
- Trained filters, hand-crafted patch function

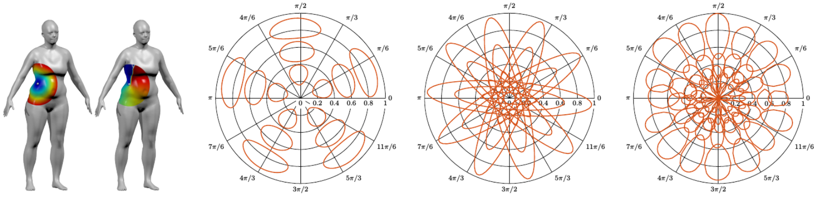


MoNet: Trainable patch function [Monti et al., 2017]

- Trainable Gaussian assignment to bin k of the patch

$$\mathbf{u}(x, y) = (\rho(x, y), \theta(x, y)) \tag{1}$$
$$w_k(\mathbf{u}) = exp((\mathbf{u} - \mu_k)^T \Sigma_k^{-1} (\mathbf{u} - \mu_k)) \tag{2}$$

- Trained filters, trained patch function, hand-crafted features for patch function



Polar coordinates ρ, θ GCNN ACNN MoNet

Polar coordinates ρ, θ

FeaStNet: Feature-Steered Graph Convolutions

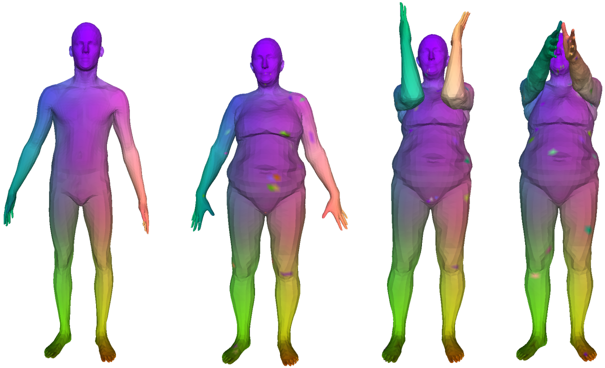
- Generic graph-convolutional network architecture

FeaStNet: Feature-Steered Graph Convolutions

- Generic graph-convolutional network architecture
- No hand-crafted features to design graph-convolution

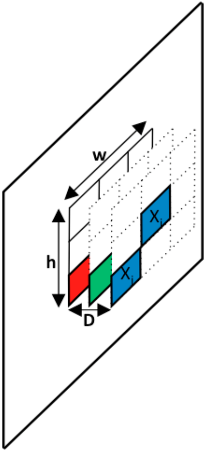
FeaStNet: Feature-Steered Graph Convolutions

- Generic graph-convolutional network architecture
- No hand-crafted features to design graph-convolution
- Validation: 3D shape correspondence and part labeling

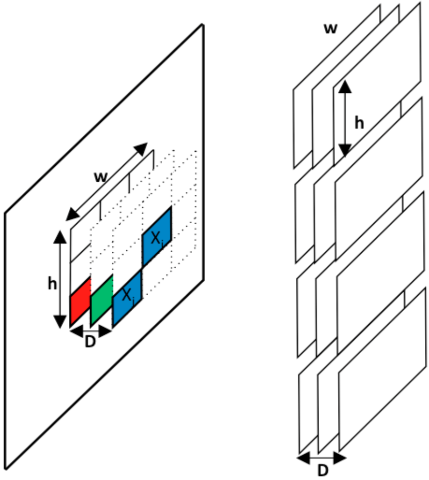


Template Texture transfer on test shapes

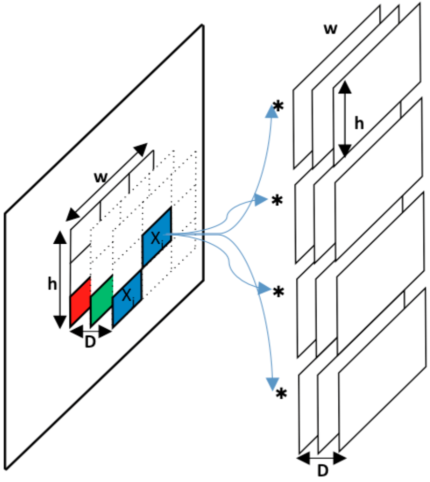
A brief recap of ConvNets



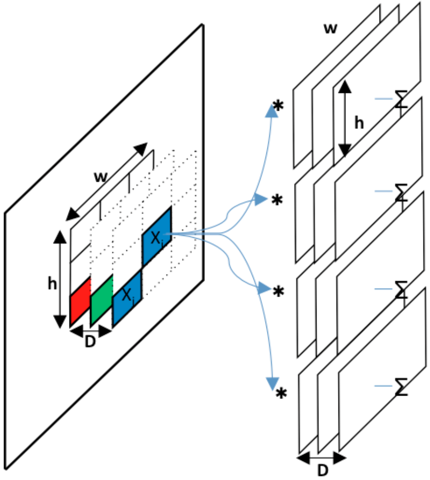
A brief recap of ConvNets



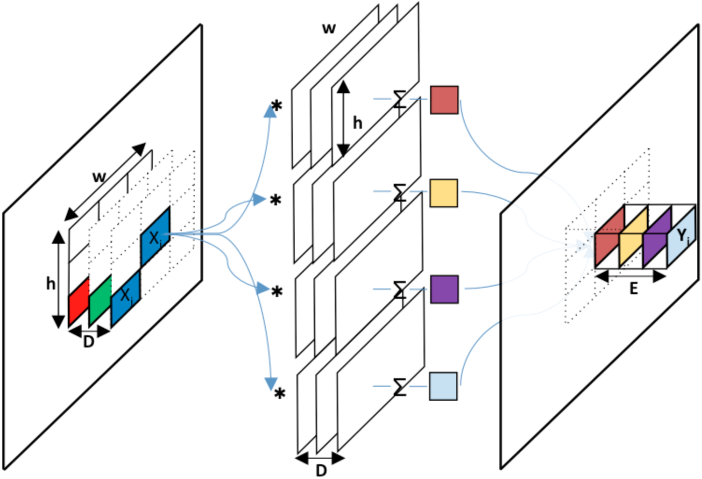
A brief recap of ConvNets



A brief recap of ConvNets

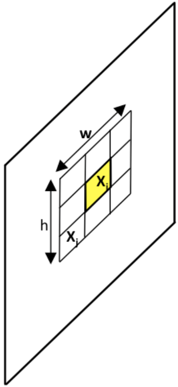


A brief recap of ConvNets



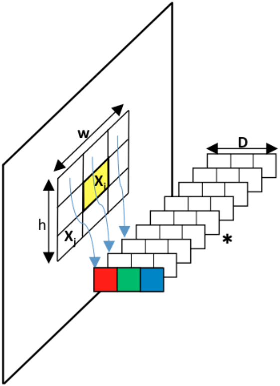
Reformulation of standard CNNs

$$y_i = b + \sum_{m=1}^{M} W_m x_{j(m,i)}$$



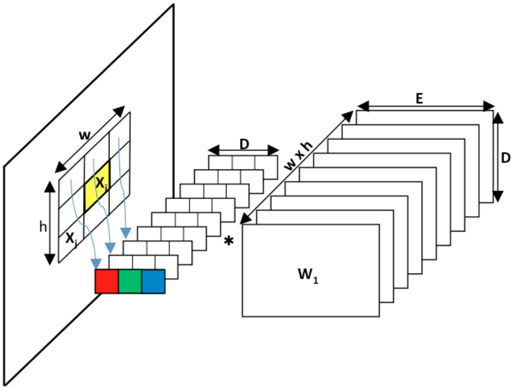
Reformulation of standard CNNs

$$y_i = b + \sum_{m=1}^{M} W_m x_{j(m,i)}$$



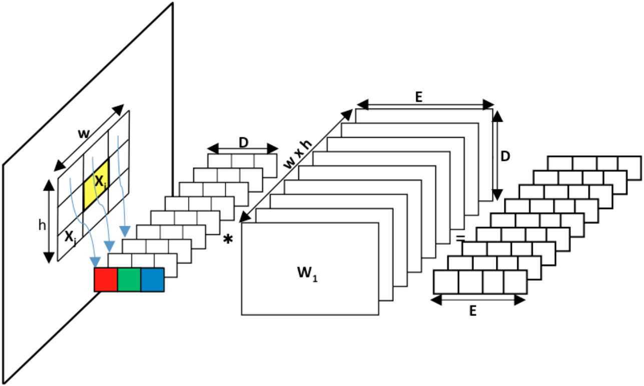
Reformulation of standard CNNs

$$y_i = b + \sum_{m=1}^{M} W_m x_{j(m,i)}$$



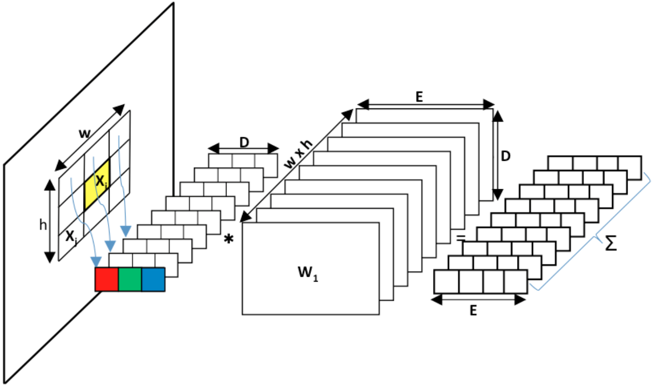
Reformulation of standard CNNs

$$y_i = b + \sum_{m=1}^{M} W_m x_{j(m,i)}$$



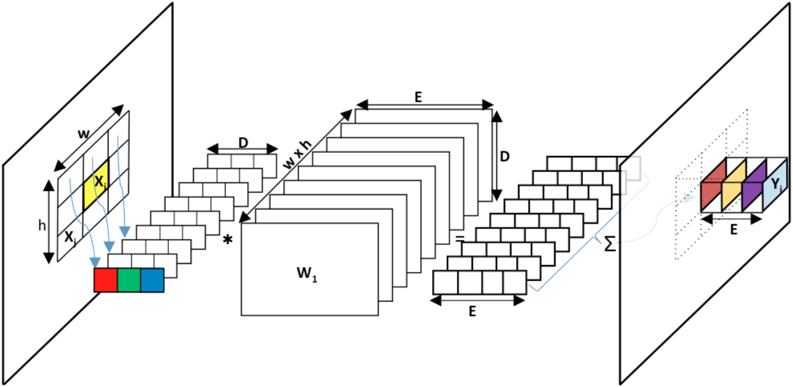
Reformulation of standard CNNs

$$y_i = b + \sum_{m=1}^{M} W_m x_{j(m,i)}$$



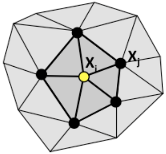
Reformulation of standard CNNs

$$y_i = b + \sum_{m=1}^{M} W_m x_{j(m,i)}$$



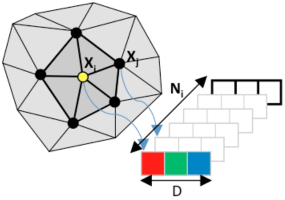
Graph convolutional approach in FeaStNet

- Varying number of neighbors, no intrinsic ordering
- No one-to-one mapping between neighbors and weights



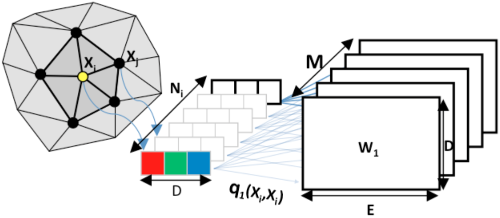
Graph convolutional approach in FeaStNet

- Varying number of neighbors, no intrinsic ordering
- No one-to-one mapping between neighbors and weights



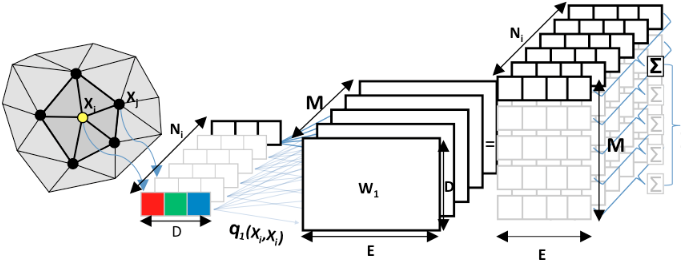
Graph convolutional approach in FeaStNet

- Varying number of neighbors, no intrinsic ordering
- No one-to-one mapping between neighbors and weights



Graph convolutional approach in FeaStNet

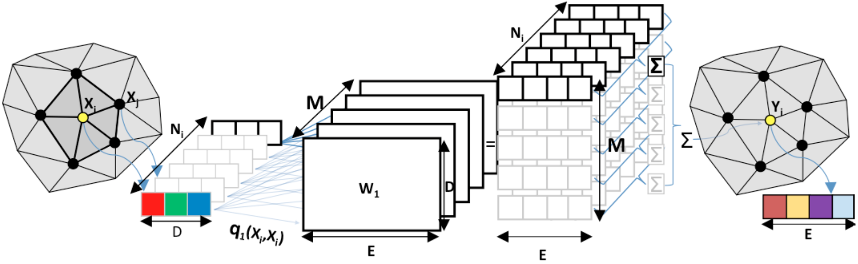
- Varying number of neighbors, no intrinsic ordering
- No one-to-one mapping between neighbors and weights



Graph convolutional approach in FeaStNet

- Varying number of neighbors, no intrinsic ordering
- No one-to-one mapping between neighbors and weights
- Weighted assignment of neighbors to weights

$$y_i = b + \frac{1}{|\mathcal{N}_i|} \sum_{m=1}^{M} W_m \sum_{j \in \mathcal{N}_i} q_m^{ij} x_j$$

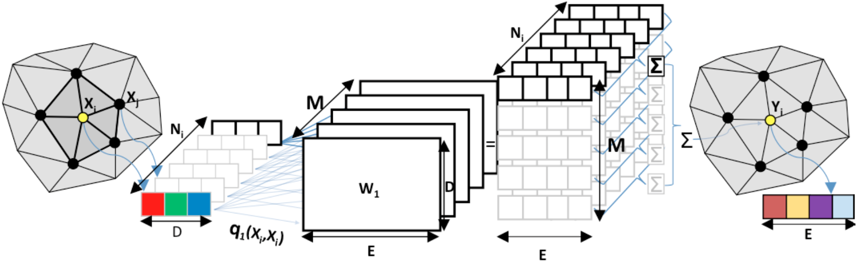


Graph convolutional approach in FeaStNet

- Varying number of neighbors, no intrinsic ordering
- No one-to-one mapping between neighbors and weights
- Weighted assignment of neighbors to weights

$$y_i = b + \frac{1}{|\mathcal{N}_i|} \sum_{m=1}^{M} W_m \sum_{j \in \mathcal{N}_i} q_m^{ij} x_j$$

- Using ring-1 neighbors in practice, but can be different



Feature-**St**eered assignment function

- Use features of previous layer to map neighbors to filters
- Can use arbitrary subnet, simplest case: 1-layer + softmax

Feature-Steered assignment function

- Use features of previous layer to map neighbors to filters
- Can use arbitrary subnet, simplest case: 1-layer + softmax

$$q_m^{ij} \propto \exp\left(u_m^\top x_i + v_m^\top x_j + c_m\right) \tag{3}$$

$$\sum_{m=1}^{M} q_m^{ij} = 1 \tag{4}$$

Feature-Steered assignment function

- Use features of previous layer to map neighbors to filters
- Can use arbitrary subnet, simplest case: 1-layer + softmax

$$q_m^{ij} \propto \exp\left(u_m^\top x_i + v_m^\top x_j + c_m\right) \tag{3}$$

$$\sum_{m=1}^{M} q_m^{ij} = 1 \tag{4}$$

- Total sum of weights independent of neighborhood size

$$\frac{1}{|\mathcal{N}_i|} \sum_{m=1}^{M} \sum_{j \in \mathcal{N}_i} q_m^{ij} = 1$$

Feature-Steered assignment function

- Use features of previous layer to map neighbors to filters
- Can use arbitrary subnet, simplest case: 1-layer + softmax

$$q_m^{ij} \propto \exp\left(u_m^\top x_i + v_m^\top x_j + c_m\right) \tag{3}$$

$$\sum_{m=1}^{M} q_m^{ij} = 1 \tag{4}$$

- Total sum of weights independent of neighborhood size

$$\frac{1}{|\mathcal{N}_i|} \sum_{m=1}^{M} \sum_{j \in \mathcal{N}_i} q_m^{ij} = 1$$

- Setting $u_m = -v_m$ in makes assignment translation invariant in feature space

$$q_m^{ij} \propto \exp\left(u_m^\top (x_j - x_i) + c_m\right)$$

Analysis: nr. parameters and computational cost

- Number of parameters:

Analysis: nr. parameters and computational cost

- Number of parameters:
  - For each weight matrix W_m, add vectors for assignment function u_m, v_m. Collect in new matrix $[W_m u_m v_m]$.

Analysis: nr. parameters and computational cost

- Number of parameters:
  - For each weight matrix W_m, add vectors for assignment function u_m, v_m. Collect in new matrix $[W_m u_m v_m]$.
  - Similar to two more output channels: from E to $E + 2$.

Analysis: nr. parameters and computational cost

- Number of parameters:
  - For each weight matrix W_m, add vectors for assignment function u_m, v_m. Collect in new matrix $[W_m u_m v_m]$.
  - Similar to two more output channels: from E to $E + 2$.
- Computational cost:

Analysis: nr. parameters and computational cost

- Number of parameters:
  - For each weight matrix W_m, add vectors for assignment function u_m, v_m. Collect in new matrix $[W_m u_m v_m]$.
  - Similar to two more output channels: from E to $E + 2$.
- Computational cost:
  - As in CNN: Compute one big matrix-matrix product between activations in previous layer (vertices $\times$ input dims) and weight matrices (input dims $\times$ output dims $\times$ nr. of filters). Cost: $O(NMED)$.

Analysis: nr. parameters and computational cost

- Number of parameters:
  - For each weight matrix W_m, add vectors for assignment function u_m, v_m. Collect in new matrix $[W_m u_m v_m]$.
  - Similar to two more output channels: from E to $E+2$.
- Computational cost:
  - As in CNN: Compute one big matrix-matrix product between activations in previous layer (vertices $\times$ input dims) and weight matrices (input dims $\times$ output dims $\times$ nr. of filters). Cost: $O(NMED)$.
  - Aggregate projections of each neighbor on each filter, instead of a single one: $O(NMEK)$

Analysis: nr. parameters and computational cost

- Number of parameters:
  - For each weight matrix W_m, add vectors for assignment function u_m, v_m. Collect in new matrix $[W_m u_m v_m]$.
  - Similar to two more output channels: from E to $E + 2$.
- Computational cost:
  - As in CNN: Compute one big matrix-matrix product between activations in previous layer (vertices $\times$ input dims) and weight matrices (input dims $\times$ output dims $\times$ nr. of filters). Cost: $O(NMED)$.
  - Aggregate projections of each neighbor on each filter, instead of a single one: $O(NMEK)$
  - Computational cost increased from $O(NMED)$ to $O(NME(D + K))$

Recovering standard CNNs

- Graph over the pixels in the image

Recovering standard CNNs

- Graph over the pixels in the image
- Edges: connect every pixel to ones needed by filter, *e.g.* eight-connected neighborhood for 3×3 filters

Recovering standard CNNs

- Graph over the pixels in the image
- Edges: connect every pixel to ones needed by filter, *e.g.* eight-connected neighborhood for 3×3 filters
- Number of weight matrices M given by filter size, *e.g.* $M = 9$ for 3×3 filters

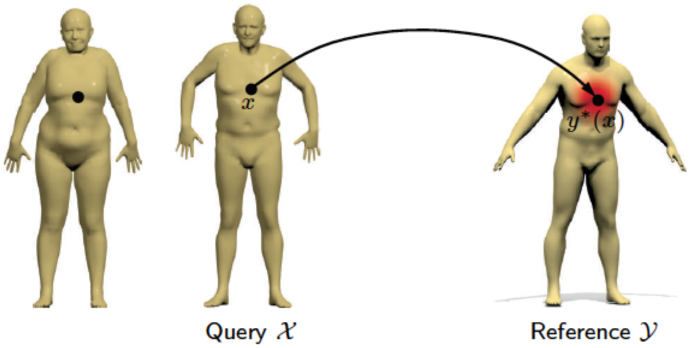
Recovering standard CNNs

- Graph over the pixels in the image
- Edges: connect every pixel to ones needed by filter, *e.g.* eight-connected neighborhood for 3×3 filters
- Number of weight matrices M given by filter size, *e.g.* $M = 9$ for 3×3 filters
- Binary assignments of neighbors to weight matrices, *i.e.* $q_m^{ij} \in \{0, 1\}$, based on position of i w.r.t. j

Recovering standard CNNs

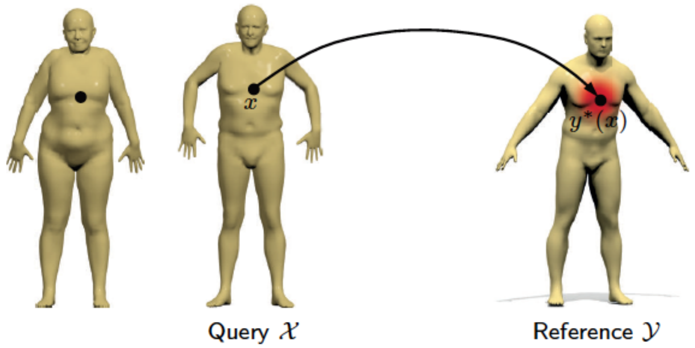
- Graph over the pixels in the image
- Edges: connect every pixel to ones needed by filter, *e.g.* eight-connected neighborhood for 3×3 filters
- Number of weight matrices M given by filter size, *e.g.* $M = 9$ for 3×3 filters
- Binary assignments of neighbors to weight matrices, *i.e.* $q_m^{ij} \in \{0, 1\}$, based on position of i w.r.t. j
- Can be implemented by translation invariant linear-softmax assignment function

Experimental evaluation — Shape correspondence



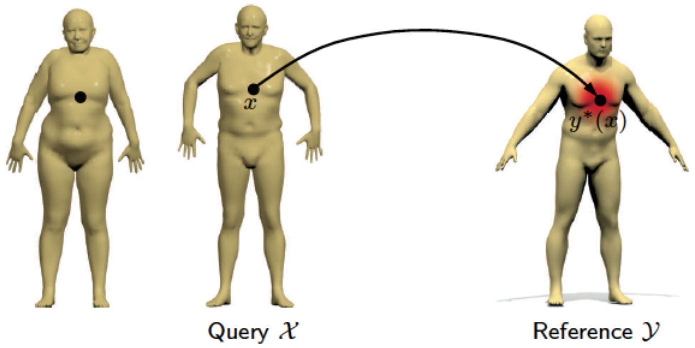
Experimental evaluation — Shape correspondence

- FAUST human shape dataset
  - 100 meshes with 6,890 vertices each
  - 10 shapes in 10 different poses: 80 train, 20 test



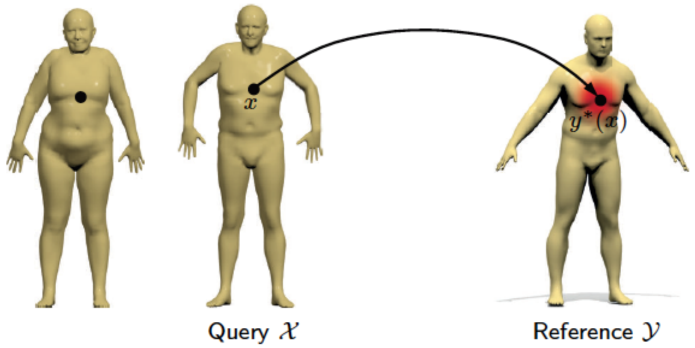
Experimental evaluation — Shape correspondence

- FAUST human shape dataset
  - 100 meshes with 6,890 vertices each
  - 10 shapes in 10 different poses: 80 train, 20 test
- Vertex descriptors
  - SHOT: Signature of Histograms of Orientations [Tombari et al., 2010]
  - XYZ: raw vertex coordinates



Experimental evaluation — Shape correspondence

- FAUST human shape dataset
  - 100 meshes with 6,890 vertices each
  - 10 shapes in 10 different poses: 80 train, 20 test
- Vertex descriptors
  - SHOT: Signature of Histograms of Orientations [Tombari et al., 2010]
  - XYZ: raw vertex coordinates
- Correspondence as dense labeling problem
  - Like semantic segmentation, but with 6,890 classes

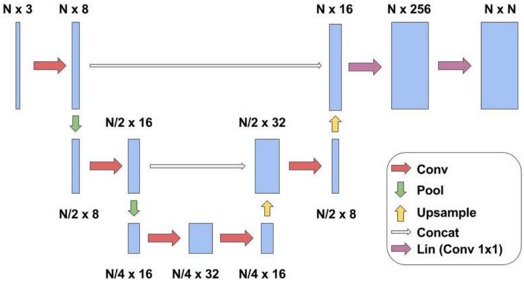


Shape correspondence: Architectures

- Single-scale architecture
  - Lin16 + Conv32 + Conv64 + Conv128 + Lin256 + Lin6890

Shape correspondence: Architectures

- Single-scale architecture
  - Lin16 + Conv32 + Conv64 + Conv128 + Lin256 + Lin6890
- Multi-scale architecture
  - Graph sub-sampling [Dhillon et al., 2007]
  - Max pooling, zero-pad up-sampling



Results single-scale architecture

- Metric: Percentage of correct (exact) correspondences

Trans.-inv.	yes	no
XYZ	86%	28%
SHOT	63%	58%

Results single-scale architecture

- Metric: Percentage of correct (exact) correspondences

Trans.-inv.	yes	no
XYZ	86%	28%
SHOT	63%	58%

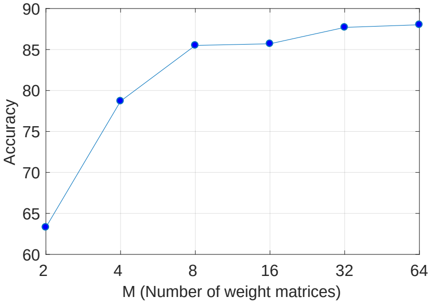
- Translation invariance helps, in particular for XYZ coordinates
- Learning from XYZ better than hand-crafted SHOT

Results single-scale architecture

- Metric: Percentage of correct (exact) correspondences

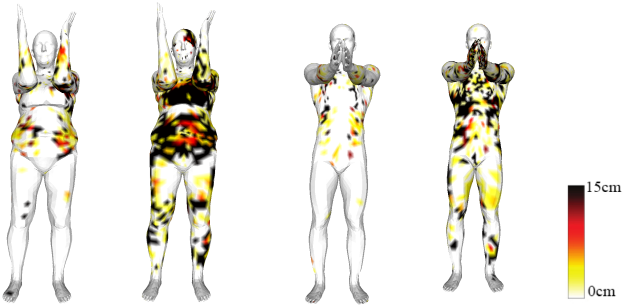
Trans.-inv.	yes	no
XYZ	86%	28%
SHOT	63%	58%

- Translation invariance helps, in particular for XYZ coordinates
- Learning from XYZ better than hand-crafted SHOT
- Impact nr. of weight matrices, using XYZ



Geodesic errors: SHOT *vs.* XYZ

- Geodesic distance between predicted and true correspondence
- Single-scale architecture in both cases



XYZ SHOT XYZ SHOT

Comparison to state of the art

- Metric: Percentage of correct (exact) correspondences

Method	Input	Accuracy
Logistic Regr., w/o refinement	SHOT	39.9%
GCNN [Masci et al., 2015], w/o refinement	SHOT	42.3%
PointNet [Qi et al., 2017], w/o refinement	SHOT	49.7%
ACNN [Boscaini et al., 2016], w/ refinement	SHOT	62.4%
GCNN [Masci et al., 2015], w/ refinement	SHOT	65.4%
MoNet [Monti et al., 2017], w/o refinement	SHOT	73.8%
MoNet [Monti et al., 2017], w/ refinement	SHOT	88.2%
FeaStNet, w/o refinement	XYZ	88.1%

Comparison to state of the art

- Metric: Percentage of correct (exact) correspondences

Method	Input	Accuracy
Logistic Regr., w/o refinement	SHOT	39.9%
GCNN [Masci et al., 2015], w/o refinement	SHOT	42.3%
PointNet [Qi et al., 2017], w/o refinement	SHOT	49.7%
ACNN [Boscaini et al., 2016], w/ refinement	SHOT	62.4%
GCNN [Masci et al., 2015], w/ refinement	SHOT	65.4%
MoNet [Monti et al., 2017], w/o refinement	SHOT	73.8%
MoNet [Monti et al., 2017], w/ refinement	SHOT	88.2%
FeaStNet, w/o refinement	XYZ	88.1%
FeaStNet, w/ refinement	XYZ	92.2%

Comparison to state of the art

- Metric: Percentage of correct (exact) correspondences

Method	Input	Accuracy
Logistic Regr., w/o refinement	SHOT	39.9%
GCNN [Masci et al., 2015], w/o refinement	SHOT	42.3%
PointNet [Qi et al., 2017], w/o refinement	SHOT	49.7%
ACNN [Boscaini et al., 2016], w/ refinement	SHOT	62.4%
GCNN [Masci et al., 2015], w/ refinement	SHOT	65.4%
MoNet [Monti et al., 2017], w/o refinement	SHOT	73.8%
MoNet [Monti et al., 2017], w/ refinement	SHOT	88.2%
FeaStNet, w/o refinement	XYZ	88.1%
FeaStNet, w/ refinement	XYZ	92.2%
FeaStNet, multi scale, w/o refinement	**XYZ**	**98.6%**
FeaStNet, multi scale, w/ refinement	**XYZ**	**98.7%**

Comparison to state of the art

- Metric: Percentage of correct (exact) correspondences

Method	Input	Accuracy
Logistic Regr., w/o refinement	SHOT	39.9%
GCNN [Masci et al., 2015], w/o refinement	SHOT	42.3%
PointNet [Qi et al., 2017], w/o refinement	SHOT	49.7%
ACNN [Boscaini et al., 2016], w/ refinement	SHOT	62.4%
GCNN [Masci et al., 2015], w/ refinement	SHOT	65.4%
MoNet [Monti et al., 2017], w/o refinement	SHOT	73.8%
MoNet [Monti et al., 2017], w/ refinement	SHOT	88.2%
FeaStNet, w/o refinement	XYZ	88.1%
FeaStNet, w/ refinement	XYZ	92.2%
FeaStNet, multi scale, w/o refinement	**XYZ**	**98.6%**
FeaStNet, multi scale, w/ refinement	**XYZ**	**98.7%**
FeaStNet, multi scale, w/o refinement	SHOT	90.9%

Comparison to state of the art

- Metric: Percentage of correct (exact) correspondences

Method	Input	Accuracy
Logistic Regr., w/o refinement	SHOT	39.9%
GCNN [Masci et al., 2015], w/o refinement	SHOT	42.3%
PointNet [Qi et al., 2017], w/o refinement	SHOT	49.7%
ACNN [Boscaini et al., 2016], w/ refinement	SHOT	62.4%
GCNN [Masci et al., 2015], w/ refinement	SHOT	65.4%
MoNet [Monti et al., 2017], w/o refinement	SHOT	73.8%
MoNet [Monti et al., 2017], w/ refinement	SHOT	88.2%
FeaStNet, w/o refinement	XYZ	88.1%
FeaStNet, w/ refinement	XYZ	92.2%
FeaStNet, multi scale, w/o refinement	**XYZ**	**98.6%**
FeaStNet, multi scale, w/ refinement	**XYZ**	**98.7%**
FeaStNet, multi scale, w/o refinement	SHOT	90.9%

- New state of the art result, with both XYZ and SHOT

Comparison to state of the art

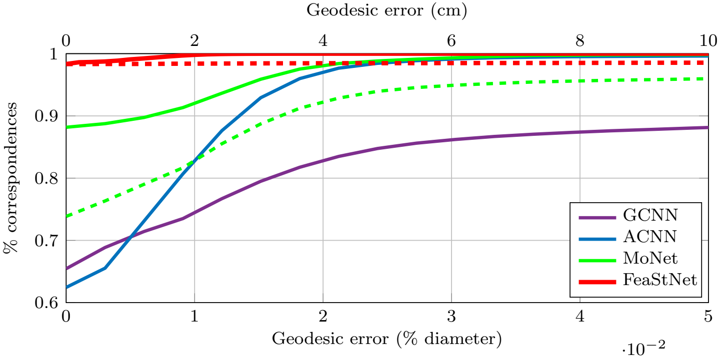
- Metric: Percentage of correct (exact) correspondences

Method	Input	Accuracy
Logistic Regr., w/o refinement	SHOT	39.9%
GCNN [Masci et al., 2015], w/o refinement	SHOT	42.3%
PointNet [Qi et al., 2017], w/o refinement	SHOT	49.7%
ACNN [Boscaini et al., 2016], w/ refinement	SHOT	62.4%
GCNN [Masci et al., 2015], w/ refinement	SHOT	65.4%
MoNet [Monti et al., 2017], w/o refinement	SHOT	73.8%
MoNet [Monti et al., 2017], w/ refinement	SHOT	88.2%
FeaStNet, w/o refinement	XYZ	88.1%
FeaStNet, w/ refinement	XYZ	92.2%
FeaStNet, multi scale, w/o refinement	**XYZ**	**98.6%**
FeaStNet, multi scale, w/ refinement	**XYZ**	**98.7%**
FeaStNet, multi scale, w/o refinement	SHOT	90.9%

- New state of the art result, with both XYZ and SHOT
- Relative reduction of 89% in error rate w.r.t. Monti *et al.*

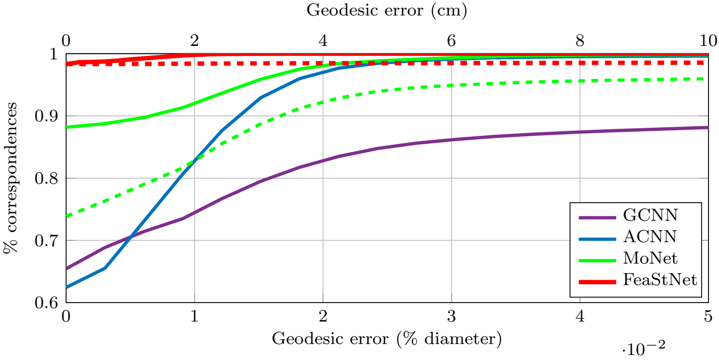
Geodesic errors

- Metric: Percentage of correspondences within tolerance
- Dashed curves: without refinement



Geodesic errors

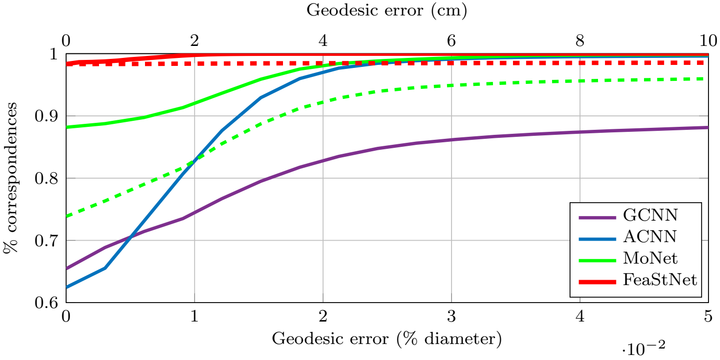
- Metric: Percentage of correspondences within tolerance
- Dashed curves: without refinement



- Without refinement: very few, relatively big errors

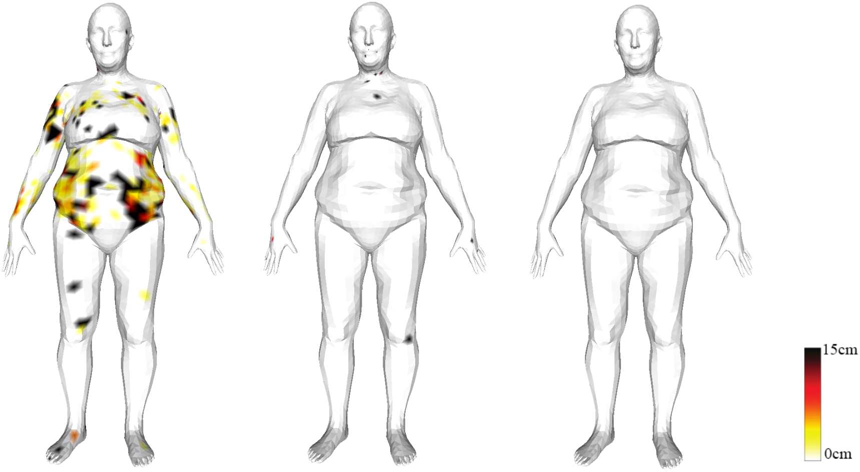
Geodesic errors

- Metric: Percentage of correspondences within tolerance
- Dashed curves: without refinement



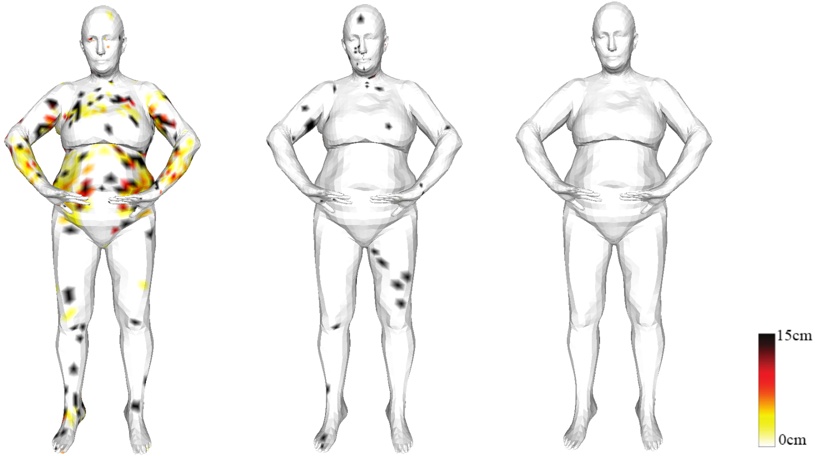
- Without refinement: very few, relatively big errors
- With refinement: very few, very small errors

Shape correspondence: Geodesic errors



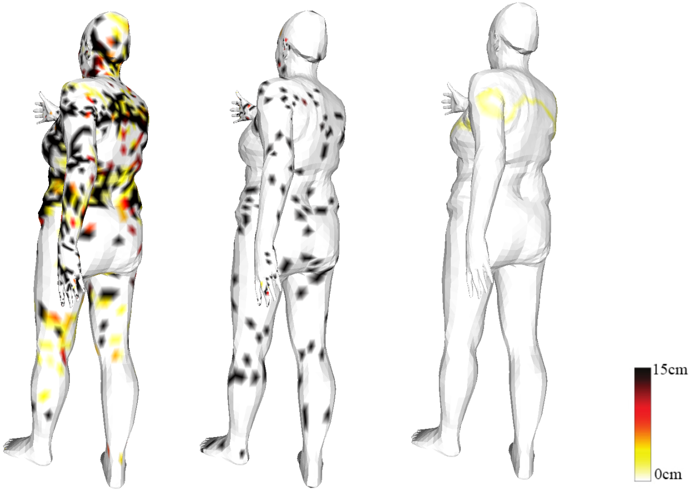
Single-scale　　Multi-scale　　+ refinement

Shape correspondence: Geodesic errors



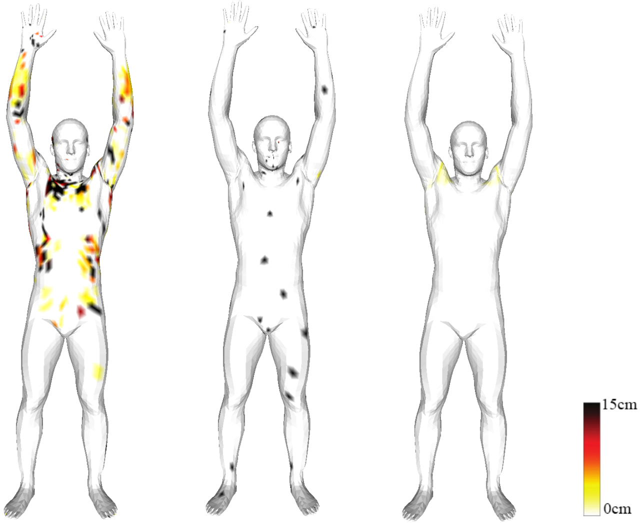
Single-scale Multi-scale + refinement

Shape correspondence: Geodesic errors



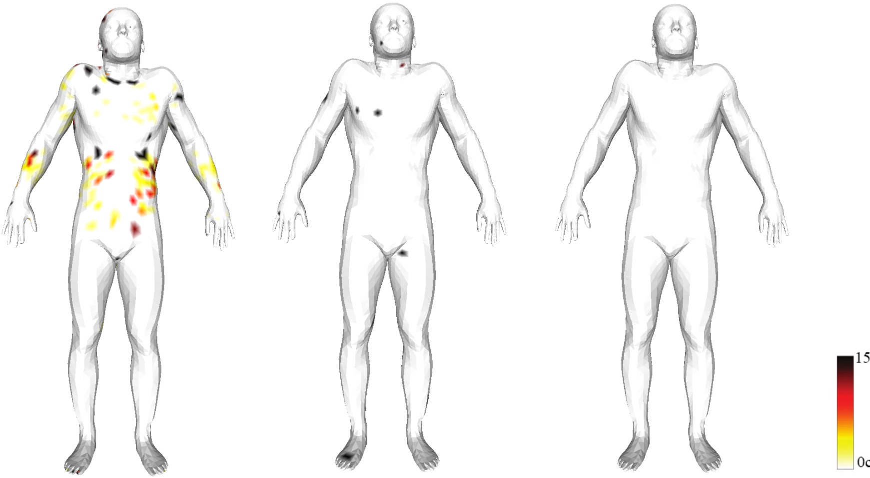
Single-scale Multi-scale + refinement

Shape correspondence: Geodesic errors



Single-scale Multi-scale + refinement

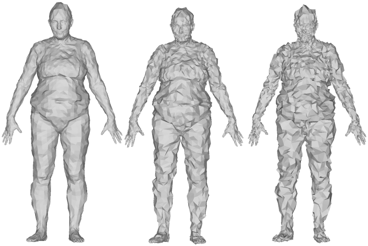
Shape correspondence: Geodesic errors



Single-scale   Multi-scale   + refinement

Shape correspondence: Noise robustness

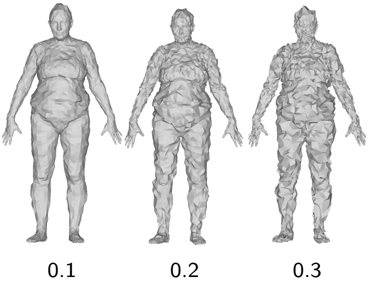
- Gaussian noise on vertex coordinates, proportional to average distance to neighbors

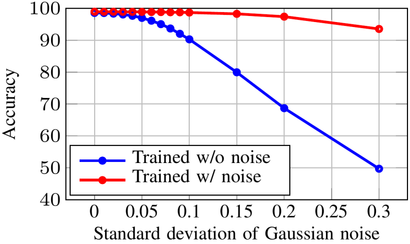


0.1 0.2 0.3

Shape correspondence: Noise robustness

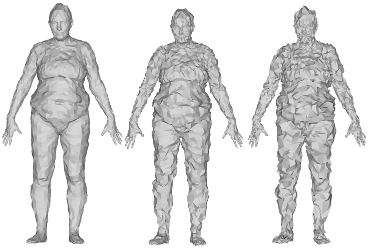
- Gaussian noise on vertex coordinates, proportional to average distance to neighbors



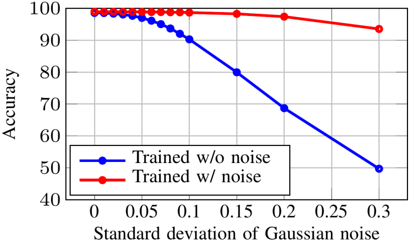


Shape correspondence: Noise robustness

- Gaussian noise on vertex coordinates, proportional to average distance to neighbors
- Robust model when training with noisy shapes

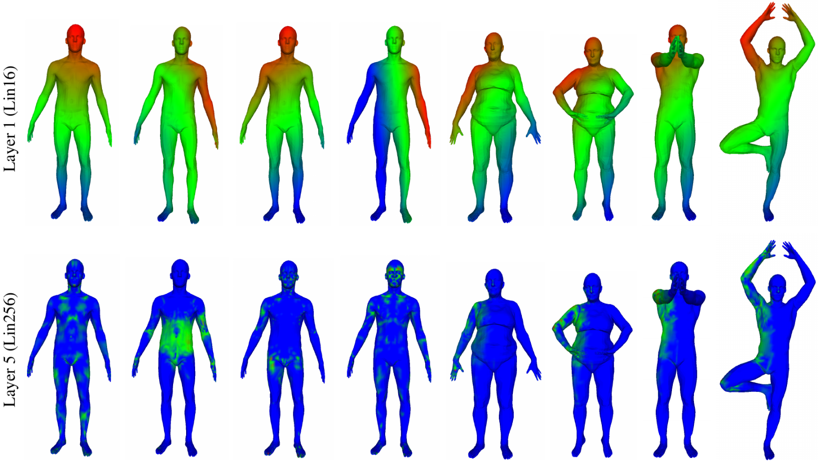


0.1 0.2 0.3



Feature activations

- Left: 4 features on same shape
- Right: same feature on 4 shapes



ShapeNet Part labeling benchmark

- 16,881 hand-designed shapes from 16 object categories
- Labeled with 50 parts across all categories
- Metric: Mean intersection-over-union (mIoU)

ShapeNet Part labeling benchmark

- 16,881 hand-designed shapes from 16 object categories
- Labeled with 50 parts across all categories
- Metric: Mean intersection-over-union (mIoU)
- Nearest neighbor graph on point cloud, using k=16

ShapeNet Part labeling benchmark

- 16,881 hand-designed shapes from 16 object categories
- Labeled with 50 parts across all categories
- Metric: Mean intersection-over-union (mIoU)
- Nearest neighbor graph on point cloud, using k=16
- Single-scale architecture, with global max-pooling

ShapeNet Part labeling benchmark

- 16,881 hand-designed shapes from 16 object categories
- Labeled with 50 parts across all categories
- Metric: Mean intersection-over-union (mIoU)
- Nearest neighbor graph on point cloud, using k=16
- Single-scale architecture, with global max-pooling
- Descriptors: XYZ coordinates

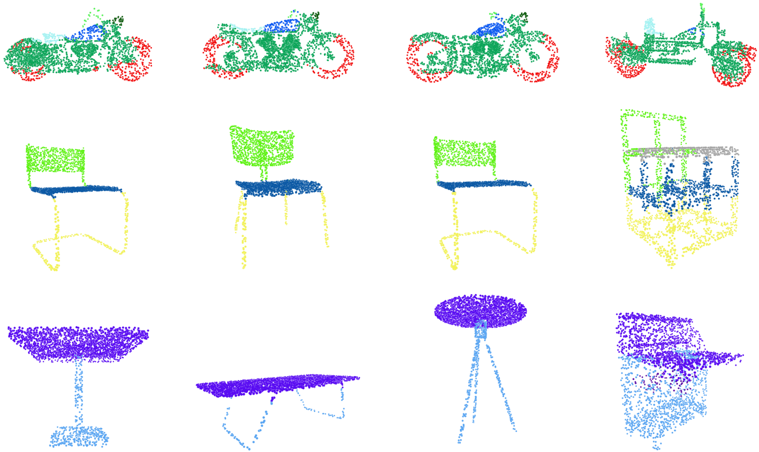
Part labeling: Quantitative results

- Performance similar to methods designed for point-cloud data

	overall	aero plane	car	chair	guitar	knife	lamp	laptop	motor bike	pistol	table
Number of shapes	16,881	2690	898	3758	787	392	1547	451	202	283	5271
PointNet [Qi et al., 2017]	83.7	83.4	74.9	89.6	91.5	85.9	80.8	95.3	65.2	81.2	80.6
KdNet [Klokov and Lempitsky, 2017]	82.3	80.1	70.3	88.6	90.2	87.2	81.0	94.9	57.4	78.1	80.3
FeaStNet	81.5	79.3	71.7	87.5	90.0	80.1	78.7	94.7	62.4	78.3	79.6

Part labeling examples

- Test shapes with accurate labeling,
  and one with worst labeling in category.



Conclusion

- Graph-convolutional architecture based on local filtering
- Learned features drive the graph convolutions

Conclusion

- Graph-convolutional architecture based on local filtering
- Learned features drive the graph convolutions
- State-of-the-art 3D shape correspondence from raw XYZ
- Comparable to previous work on point cloud labeling

Conclusion

- Graph-convolutional architecture based on local filtering
- Learned features drive the graph convolutions
- State-of-the-art 3D shape correspondence from raw XYZ
- Comparable to previous work on point cloud labeling

- Perspectives
  - Application to raw/real scanned 3D meshes
  - Integrate global correspondence refinement
  - Generalize across meshes/templates: local correspondences
  - Modeling meshes in motion: (shape + pose) x time

FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis

Nitika Verma, Edmond Boyer, Jakob Verbeek
INRIA, Grenoble, France

Computer Vision and Pattern Recognition, 2018

References I

[Boscaini et al., 2016] Boscaini, D., Masci, J., Rodolà, E., and Bronstein, M. (2016).
Learning shape correspondence with anisotropic convolutional neural networks.
In *NIPS*.

[Dhillon et al., 2007] Dhillon, I., Guan, Y., and Kulis, B. (2007).
Weighted graph cuts without eigenvectors: A multilevel approach.
PAMI, 29(11).

[Farabet et al., 2013] Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013).
Learning hierarchical features for scene labeling.
PAMI, 35(8):1915–1929.

[Gordo et al., 2016] Gordo, A., Almazan, J., Revaud, J., and Larlus, D. (2016).
Deep image retrieval: Learning global representations for image search.
In *ECCV*.

[Graham et al., 2018] Graham, B., Engelcke, M., and van der Maaten, L. (2018).
3d semantic segmentation with submanifold sparse convolutional networks.
In *CVPR*.

[Klokov and Lempitsky, 2017] Klokov, R. and Lempitsky, V. (2017).
Escape from cells: Deep kd-networks for the recognition of 3D point cloud models.
In *ICCV*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. (2012).
Imagenet classification with deep convolutional neural networks.
In *NIPS*.

[LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1989).
Handwritten digit recognition with a back-propagation network.
In *NIPS*.

References II

[Masci et al., 2015] Masci, J., Boscaini, D., Bronstein, M., and Vandergheynst, P. (2015).
Geodesic convolutional neural networks on Riemannian manifolds.
In *ICCV Workshops*.

[Monti et al., 2017] Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M. (2017).
Geometric deep learning on graphs and manifolds using mixture model CNNs.
In *CVPR*.

[Qi et al., 2017] Qi, C., Su, H., Mo, K., and Guibas, L. (2017).
Pointnet: Deep learning on point sets for 3D classification and segmentation.
In *CVPR*.

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015).
Faster R-CNN: towards real-time object detection with region proposal networks.
In *NIPS*.

[Tatarchenko et al., 2017] Tatarchenko, M., Dosovitskiy, A., and Brox, T. (2017).
Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs.
In *ICCV*.

[Tombari et al., 2010] Tombari, F., Salti, S., and Stefano, L. D. (2010).
Unique signatures of histograms for local surface description.
In *ECCV*.