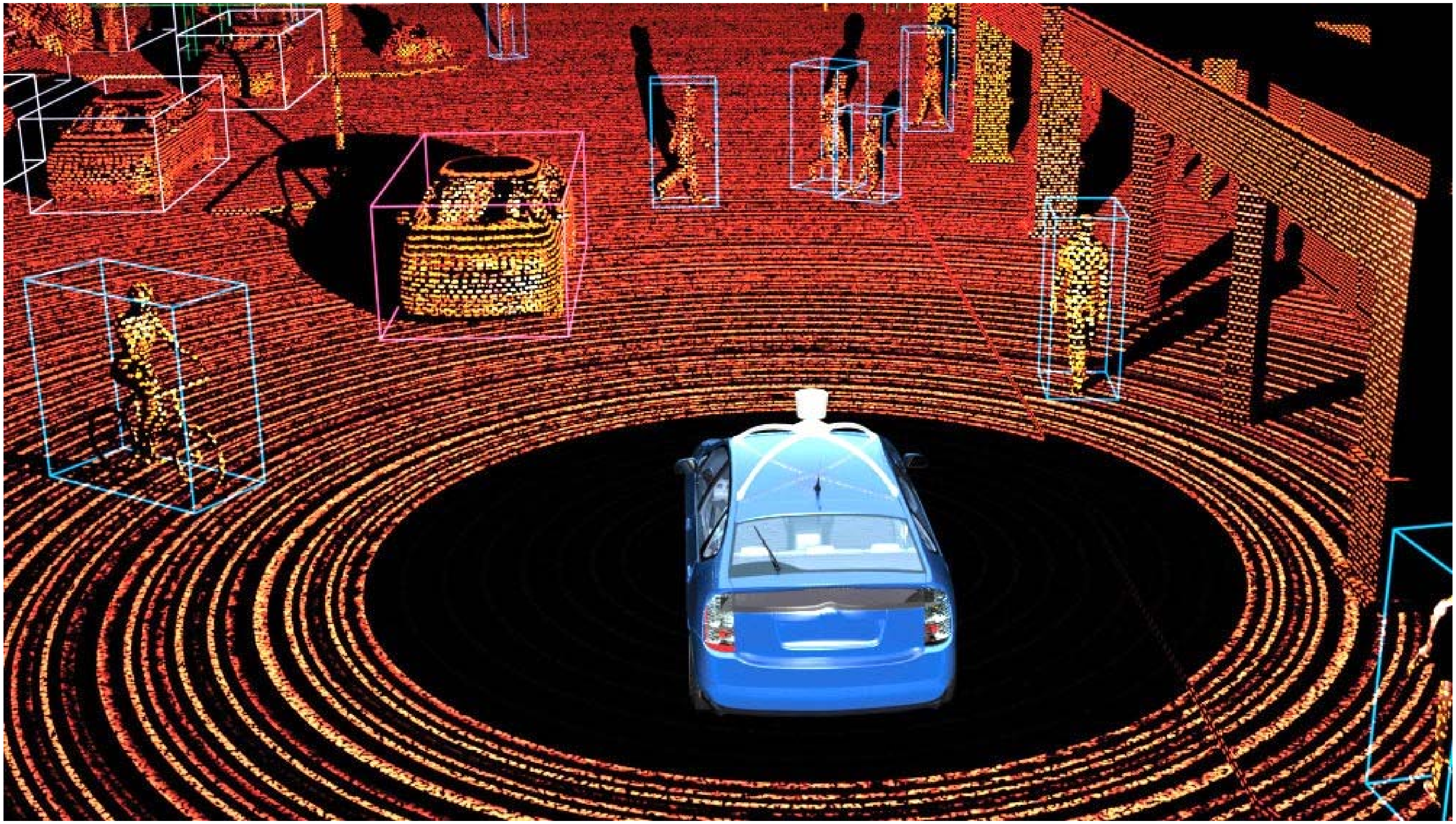
3D Deep Learning approaches
Point-based Networks + Registration

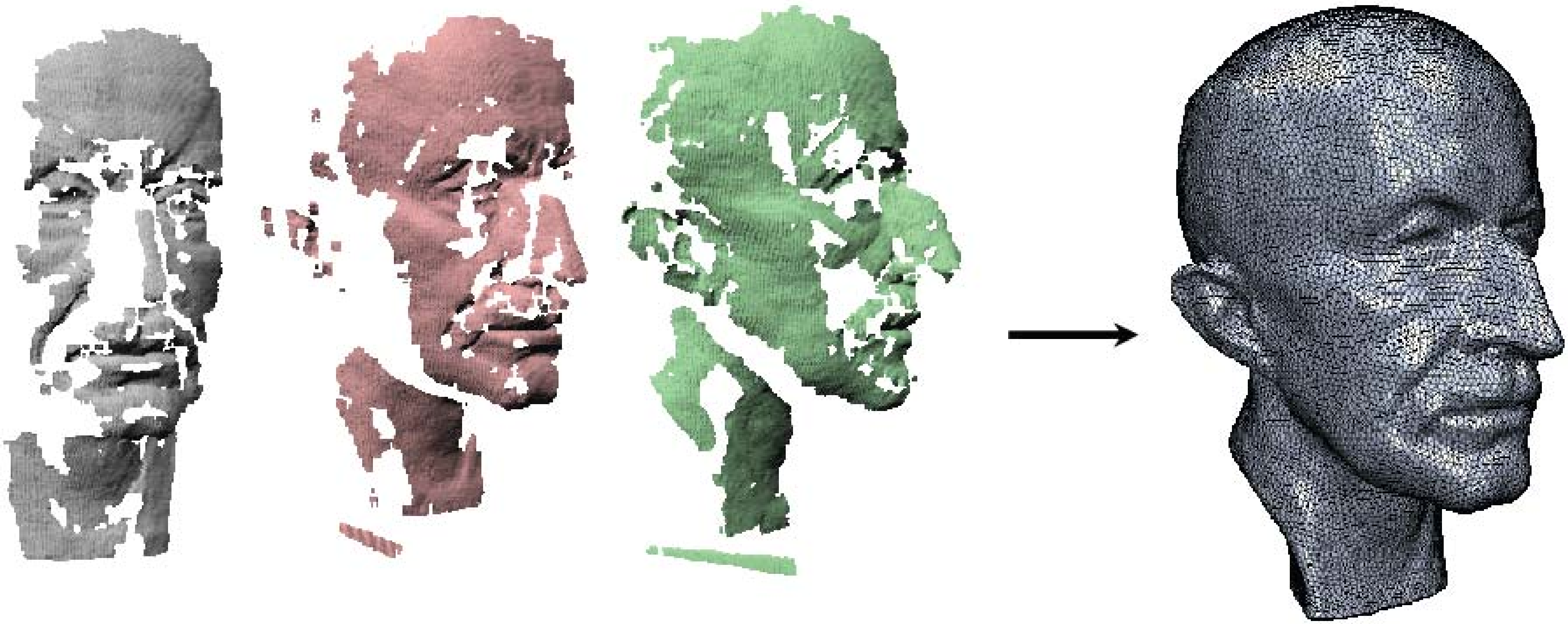


Evangelos Kalogerakis
574/674

3D Deep Learning approaches

- The Multi-View approach
- The Voxel approach
- The Point approach
  - PointNet
  - PointNet++
  - KPConv
  - ***Application: Point Cloud Registration***
  - *Point Transformers*
- The Graph approach

Output of a scanner

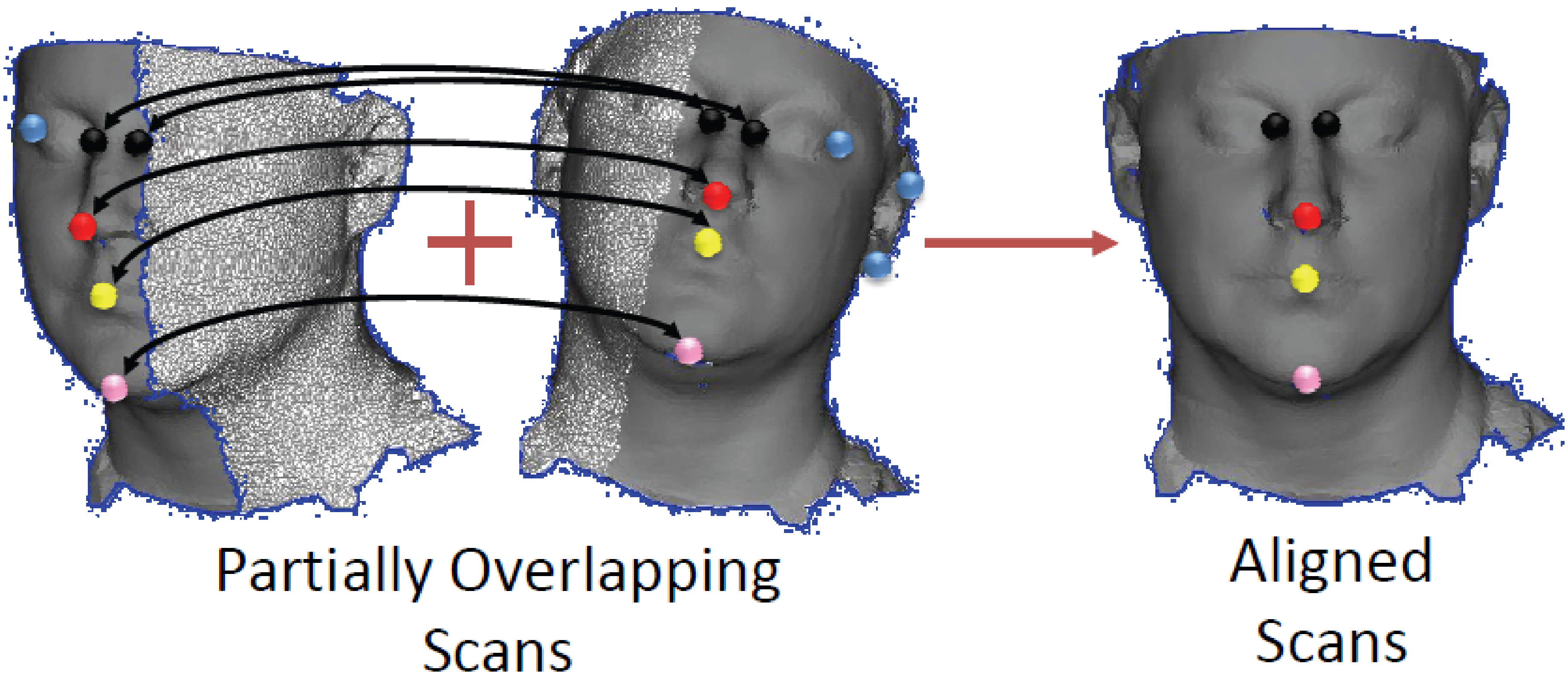


set of raw scans

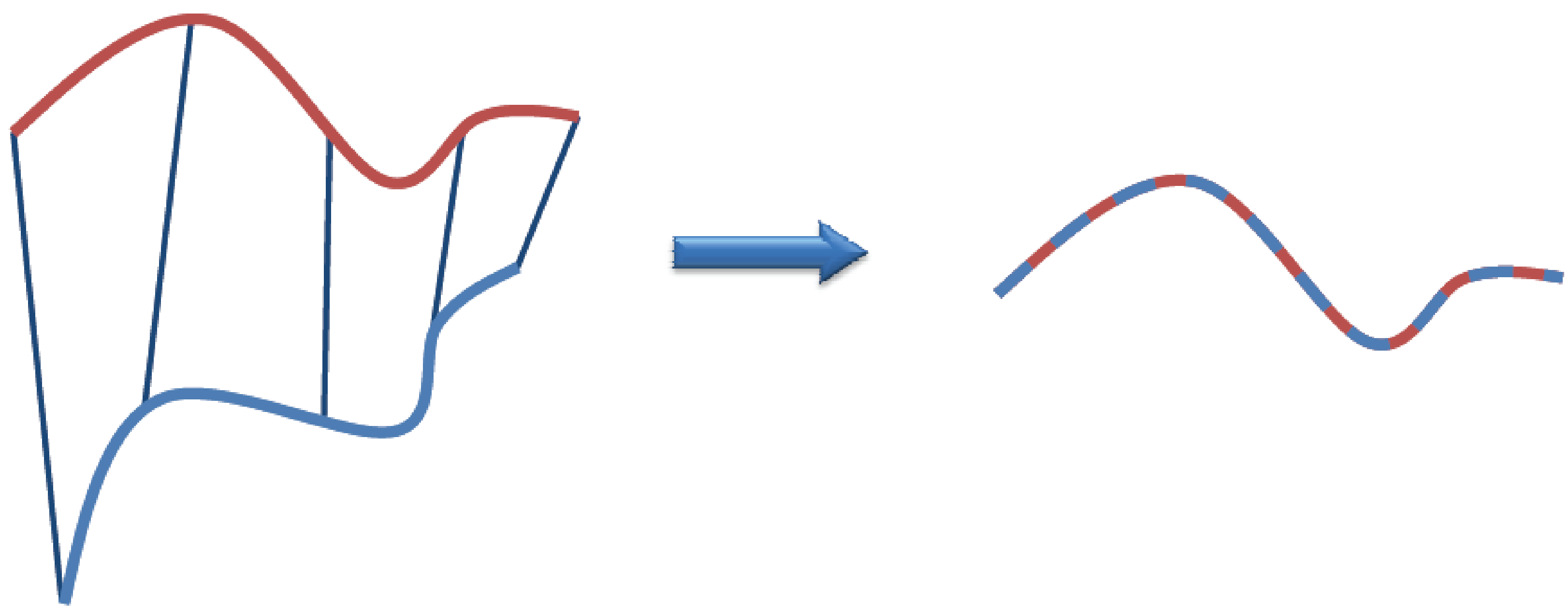
reconstructed model

Registration

Compute point-wise correspondences by comparing their point descriptors
(e.g., using PointNet++, KPConv, MinkowskiNet etc)



Registration



Problem Statement

Given *m* points from first scan: $\mathbf{P} = \{\boldsymbol{p_i}\}$

and *m* points from second scan: $\mathbf{Q} = \{\boldsymbol{q_i}\}$

Estimate ***R, t***: $$\min_{\mathbf{R,t}} \sum_{\text{points } i} \| \mathbf{p_i} - \mathbf{R}\mathbf{q_i} - \mathbf{t} \|^2$$

Solve for **t**

Error function: $$f_{q=>p}(\mathbf{R},\mathbf{t}) = \sum_i \| \mathbf{p_i} - \mathbf{R}\mathbf{q_i} - \mathbf{t} \|^2$$

Estimate centroids: $$\bar{\mathbf{p}} = \frac{\sum_i \mathbf{p_i}}{m}, \bar{\mathbf{q}} = \frac{\sum_i \mathbf{q_i}}{m}$$

Optimal translation: $$\mathbf{t} = \bar{\mathbf{p}} - \mathbf{R}\bar{\mathbf{q}}$$

Set: $$\mathbf{p_i}' = \mathbf{p_i} - \bar{\mathbf{p}}$$ $$\mathbf{q_i}' = \mathbf{q_i} - \bar{\mathbf{q}}$$

proof in [Horn 88]

Solve for **R**

The error function now becomes:

$$f_{q=>p}(\mathbf{R}) = \sum_i \| \mathbf{p_i}' - \mathbf{R}\mathbf{q_i}' \|^2$$

or more compactly:

$$f_{q=>p}(\mathbf{R}) = \| \hat{\mathbf{P}} - \hat{\mathbf{Q}}\mathbf{R}^T \|_F^2$$

***F:* Frobenius Norm**

$$\hat{\mathbf{P}} = \begin{pmatrix} \mathbf{p}_1^{T\prime} \\ \dots \\ \mathbf{p}_m^{T\prime} \end{pmatrix}_{m\times 3}, \hat{\mathbf{Q}} = \begin{pmatrix} \mathbf{q}_1^{T\prime} \\ \dots \\ \mathbf{q}_m^{T\prime} \end{pmatrix}_{m\times 3}$$

$$\|A_{n\times m}\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}^2$$

Solve for **R**

Minimize $\| \hat{\mathbf{P}} - \hat{\mathbf{Q}}\mathbf{R}^T \|_F^2$ subject to: $\mathbf{R}^\mathbf{T}\mathbf{R}=\mathbf{I}$
(R should be rotation i.e. orthogonal matrix!)

Known as orthogonal **Procrustes** problem. Solution:

$$\mathbf{S} = \hat{\mathbf{P}}^\mathbf{T}\hat{\mathbf{Q}}$$

then Singular Value Decomposition on S:

$$\mathbf{S} = \mathbf{U}\mathbf{D}\mathbf{V}^\mathbf{T}$$

Optimal Rotation (*could be reflection*):

$$\mathbf{R} = \mathbf{U}\mathbf{V}^\mathbf{T}$$

https://en.wikipedia.org/wiki/Orthogonal_Procrustes_problem

Solve for **R**

Minimize $\| \hat{\mathbf{P}} - \hat{\mathbf{Q}}\mathbf{R}^T \|_F^2$ subject to: $\mathbf{R}^{\mathbf{T}}\mathbf{R} = \mathbf{I}$
(R should be rotation i.e. orthogonal matrix!)

Known as orthogonal **Procrustes** problem. Solution:

$$\mathbf{S} = \hat{\mathbf{P}}^{\mathbf{T}}\hat{\mathbf{Q}}$$

then Singular Value Decomposition on S:

$$\mathbf{S} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathbf{T}}$$

$$\textit{Optimal Rotation (could be reflection)}:$$

$$\mathbf{R} = \mathbf{U}\mathbf{V}^{\mathbf{T}}$$

=> if determinant of **R** is negative (i.e., it’s reflection),
take the 3rd row of $\mathbf{V}^{\mathbf{T}}$ and multiply it by -1.

https://en.wikipedia.org/wiki/Orthogonal_Procrustes_problem

Aligning **Q** to match **P**

$$1.\ \textit{Estimate centroids}:\ \bar{\mathbf{p}} = \frac{\sum_i \mathbf{p}_\mathbf{i}}{m}, \bar{\mathbf{q}} = \frac{\sum_i \mathbf{q}_\mathbf{i}}{m}$$

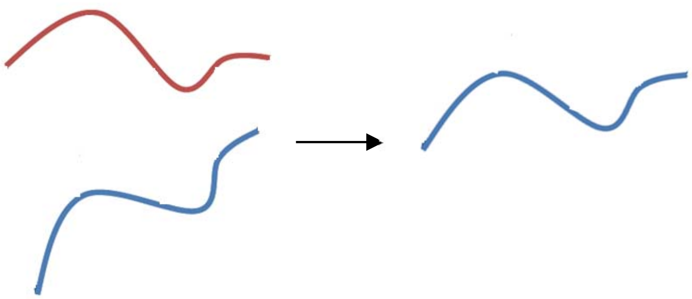
$$2.\ \textit{Set}\ \mathbf{p}_\mathbf{i}' = \mathbf{p}_\mathbf{i} - \bar{\mathbf{p}}, \mathbf{q}_\mathbf{i}' = \mathbf{q}_\mathbf{i} - \bar{\mathbf{q}}$$

$$4.\ \textit{Optimal rotation}: \mathbf{R} = \mathbf{U}\mathbf{V}^\mathbf{T}\ \ (\mathbf{U}, \mathbf{V}\ \textit{from SVD on}\ \hat{\mathbf{P}}^\mathbf{T}\hat{\mathbf{Q}})$$

$$5.\ \textit{Optimal translation applied to}\ \mathbf{Q}: \mathbf{t} = \bar{\mathbf{p}} - \mathbf{R}\bar{\mathbf{q}}$$

Initial Registration

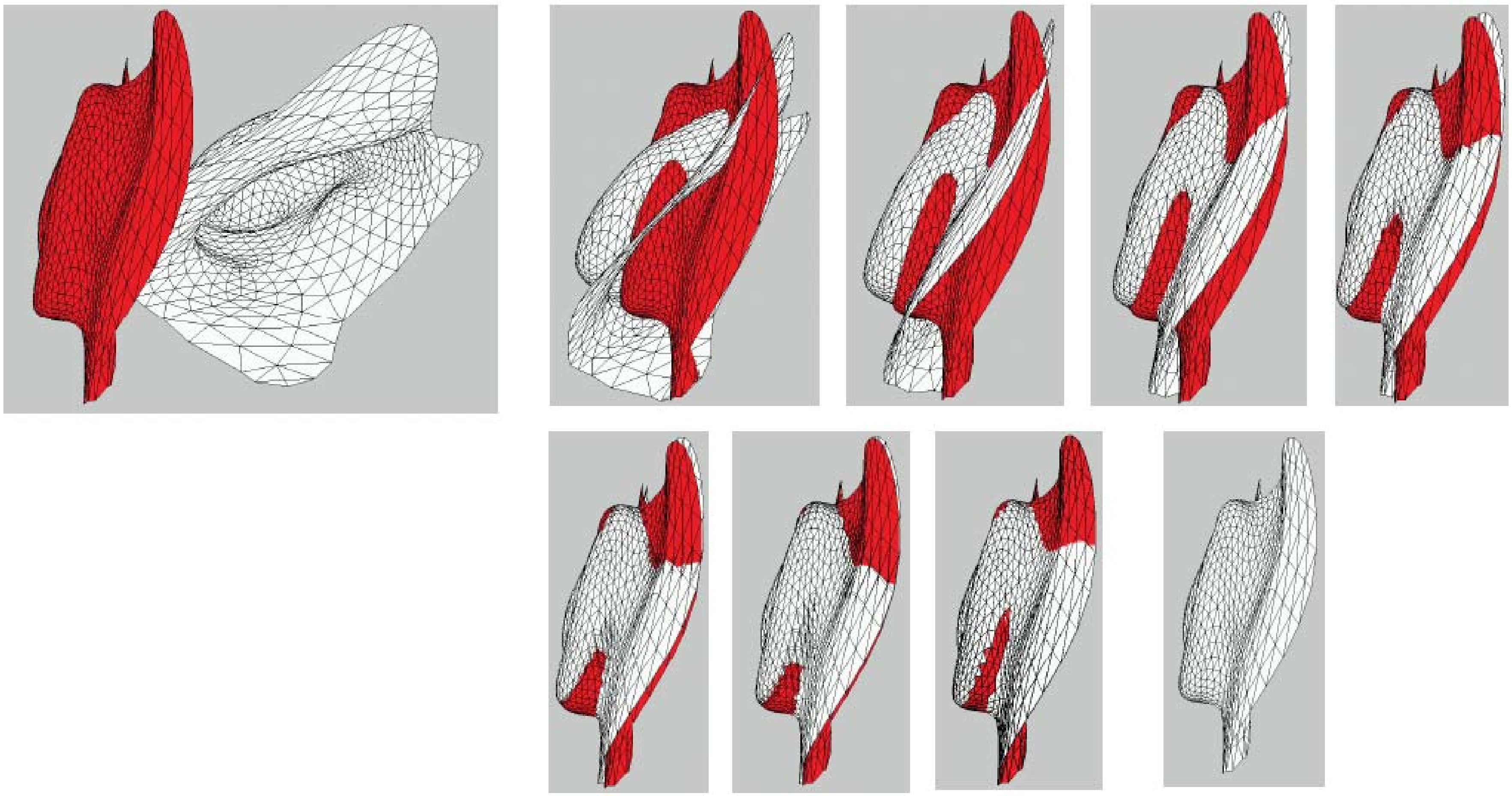
Initial registration based on a few correspondences might not be accurate



Use ICP: Iterative Closest Point

- Start with an initial estimate for {**R**,**t**} based on few point correspondences
- Update closest point matches (i.e. get new/more correspondences)
- Repeat (find new transformation, and so on)

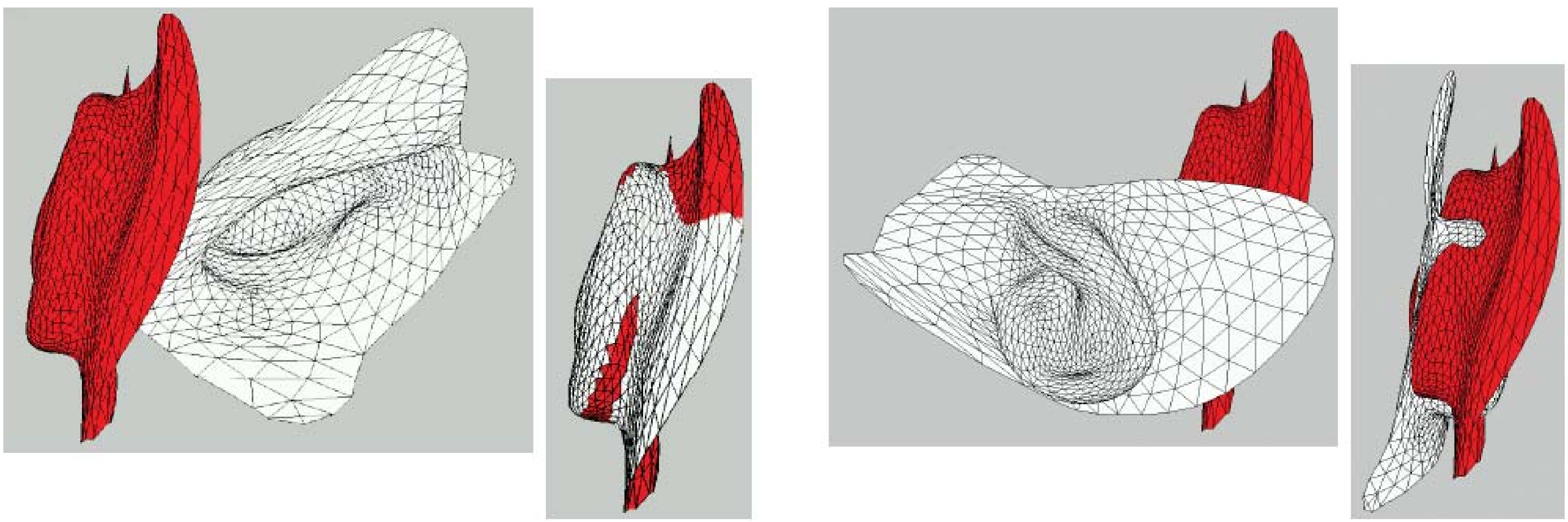
ICP



ICP

Converges to a local minimum

If initial estimate for transformation is good, then higher chances to reach to the global minimum (right solution)



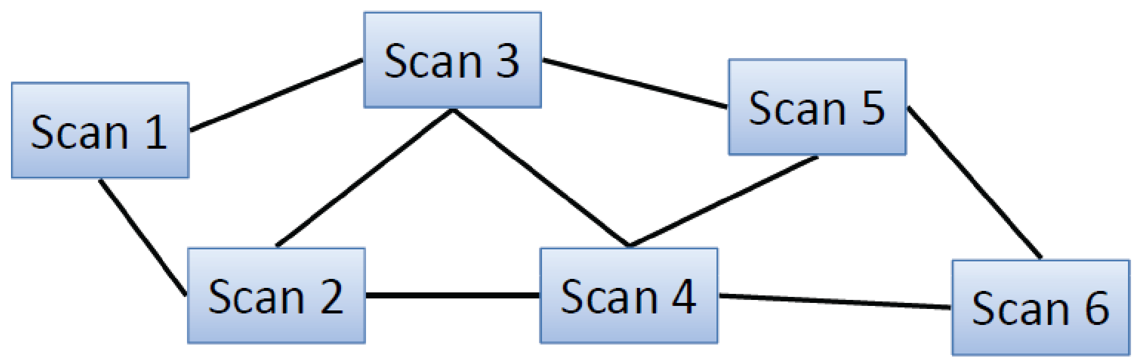
Multiple scans

For every pair of scans (s, s') that are overlapping

- minimize error wrt transform of scan s to match s'

Repeat until convergence

i.e. align pairs of scans
(edges mean overlapping scans)



3D Deep Learning approaches

- The Multi-View approach
- The Voxel approach
- The Point approach
  - PointNet
  - PointNet++
  - KPConv
  - *Application: Point Cloud Registration*
  - ***Point Transformers***
- The Graph approach

Attention vs Convolution

Convolution is extremely popular in 2D/3D vision:

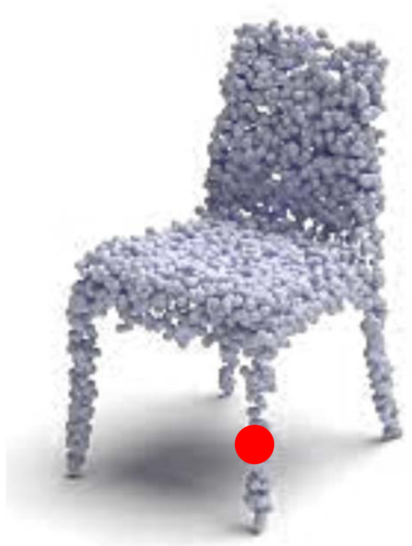
- ability to exploit local dependencies in the input data
- highly parallelizable / efficient to compute on GPUs

Capturing long-range interactions between pixels, points, voxels is not trivial with convolution

=> Let’s see how 3D attention works!

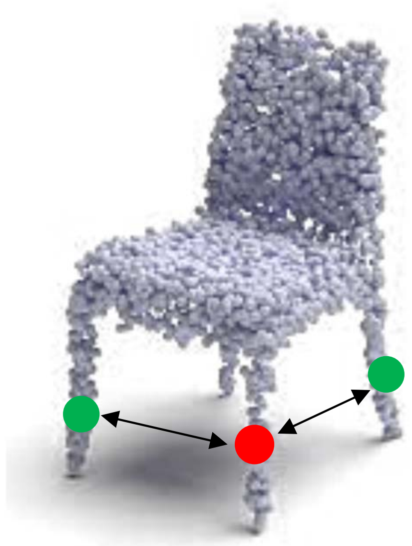
Attention for point clouds

Help encoder look at other points in the shape while encoding a point due to various relations existing in shapes



Attention for point clouds

Help encoder look at other points in the shape while encoding a point due to various relations existing in shapes



e.g., symmetry

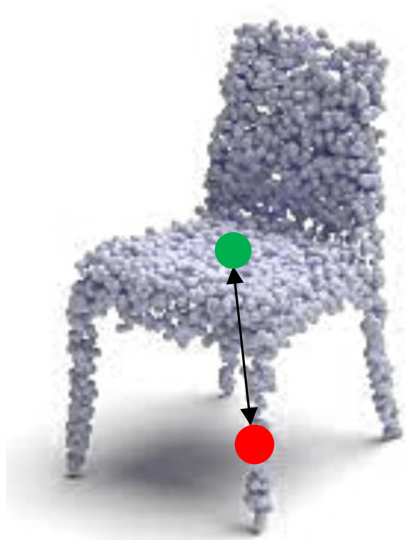
Attention for point clouds

Help encoder look at other points in the shape while encoding a point due to various relations existing in shapes

e.g., same part

Attention for point clouds

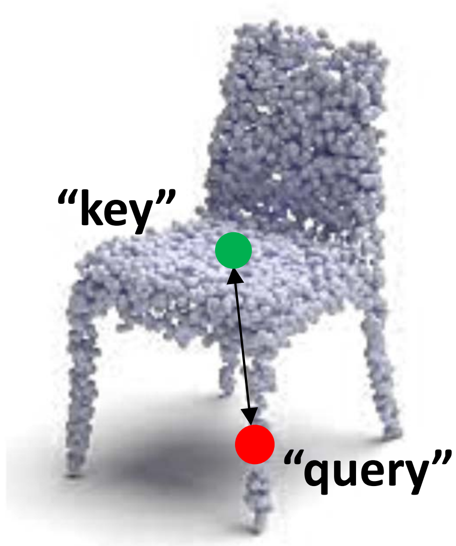
Help encoder look at other points in the shape while encoding a point due to various relations existing in shapes



e.g., parts perpendicular to each other ...

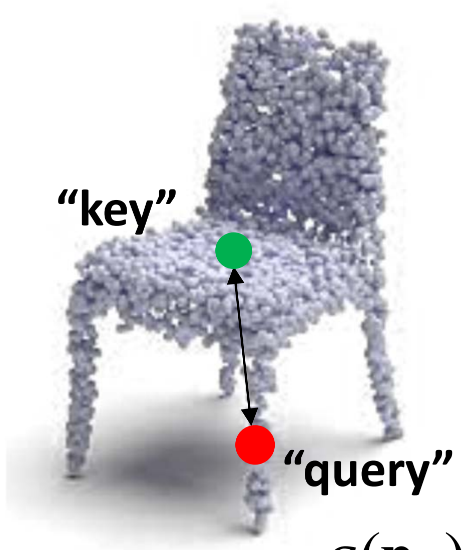
Attention for point clouds

How much a point (query) is related to others (keys)?



Attention for point clouds

How much a point (query) is related to others (keys)?



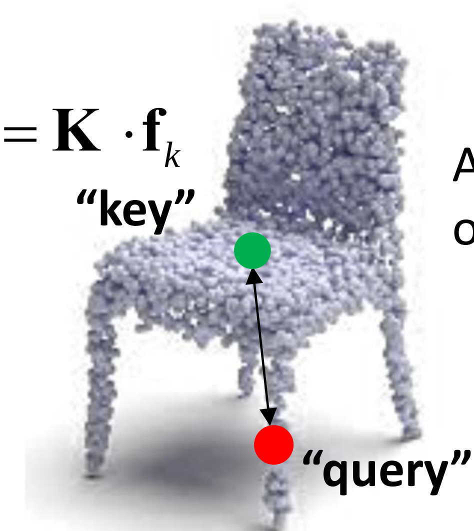
$$q(\mathbf{p}_q) = \mathbf{Q} \cdot \mathbf{f}_q$$

A linear transformation on the input feature vector of the query point
(e.g., MLP on input point positions to acquire the feature vector)

Attention for point clouds

How much a point (query) is related to others (keys)?

$$k(\mathbf{p}_k) = \mathbf{K} \cdot \mathbf{f}_k$$



Another transformation on the input feature vector of the key point

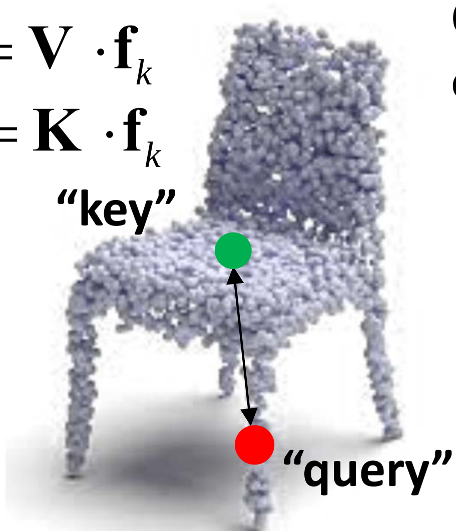
$$q(\mathbf{p}_q) = \mathbf{Q} \cdot \mathbf{f}_q$$

Attention for point clouds

How much a point (query) is related to others (keys)?

$v(\mathbf{p}_k) = \mathbf{V} \cdot \mathbf{f}_k$

$k(\mathbf{p}_k) = \mathbf{K} \cdot \mathbf{f}_k$



$q(\mathbf{p}_q) = \mathbf{Q} \cdot \mathbf{f}_q$

One more transformation on an input feature vector of the key point i.e., we have a "key-value" pair

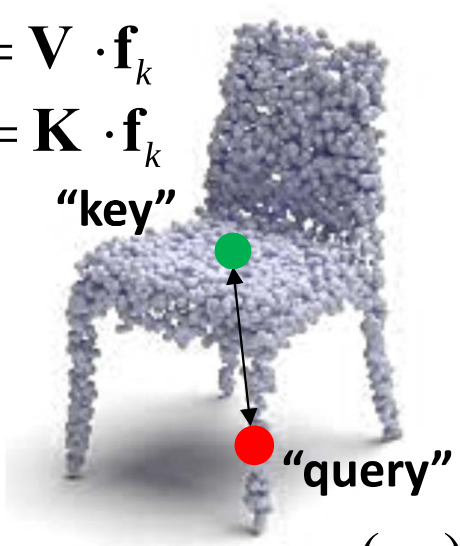
Attention for point clouds

How much a point (query) is related to others (keys)?

Attention "score": $a(\mathbf{p}_q, \mathbf{p}_k) = k(\mathbf{p}_k) \cdot q(\mathbf{p}_q)$

(dot product between key-query vectors => "scalar" attention)

$$v(\mathbf{p}_k) = \mathbf{V} \cdot \mathbf{f}_k$$
$$k(\mathbf{p}_k) = \mathbf{K} \cdot \mathbf{f}_k$$



$$q(\mathbf{p}_q) = \mathbf{Q} \cdot \mathbf{f}_q$$

Attention for point clouds

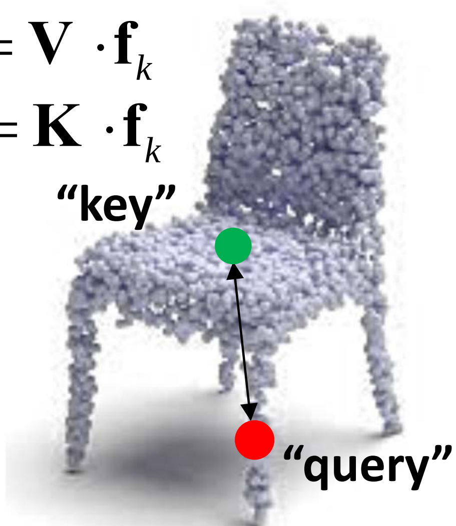
How much a point (query) is related to others (keys)?

Attention "score": $a(\mathbf{p}_q, \mathbf{p}_k) = k(\mathbf{p}_k) \cdot q(\mathbf{p}_q)$

(dot product between key-query vectors => "scalar" attention)

$v(\mathbf{p}_k) = \mathbf{V} \cdot \mathbf{f}_k$

$k(\mathbf{p}_k) = \mathbf{K} \cdot \mathbf{f}_k$



$q(\mathbf{p}_q) = \mathbf{Q} \cdot \mathbf{f}_q$

Note: there are other variants of attention e.g, instead of dot product, another way to compare keys and queries is through subtraction. This is called "vector" attention:

$$\mathbf{a}(\mathbf{p}_q, \mathbf{p}_k) = k(\mathbf{p}_k) - q(\mathbf{p}_q)$$

Point Transformers, ICCV 2021

Attention for point clouds

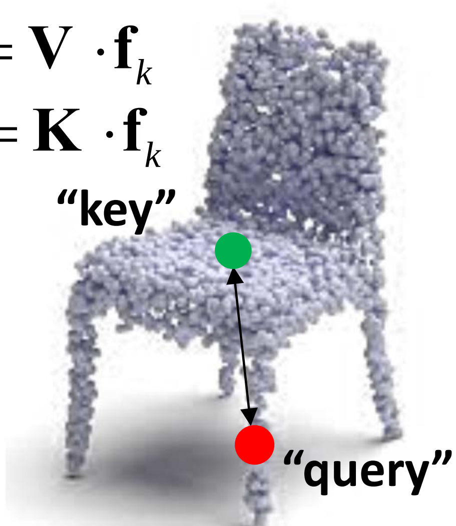
How much a point (query) is related to others (keys)?

Attention "score": $a(\mathbf{p}_q, \mathbf{p}_k) = k(\mathbf{p}_k) \cdot q(\mathbf{p}_q)$

(dot product between key-query vectors => "scalar" attention)

$v(\mathbf{p}_k) = \mathbf{V} \cdot \mathbf{f}_k$

$k(\mathbf{p}_k) = \mathbf{K} \cdot \mathbf{f}_k$



$q(\mathbf{p}_q) = \mathbf{Q} \cdot \mathbf{f}_q$

Note: there are other variants of attention e.g, instead of dot product, another way to compare keys and queries is through subtraction. This is called "vector" attention. Or use MLP on top of subtraction.

$$\mathbf{a}(\mathbf{p}_q, \mathbf{p}_k) = k(\mathbf{p}_k) - q(\mathbf{p}_q) \;\; or$$

$$\mathbf{a}(\mathbf{p}_q, \mathbf{p}_k) = MLP(\;\; k(\mathbf{p}_k) - q(\mathbf{p}_q) \;\;)$$

Point Transformers, ICCV 2021

Attention for point clouds

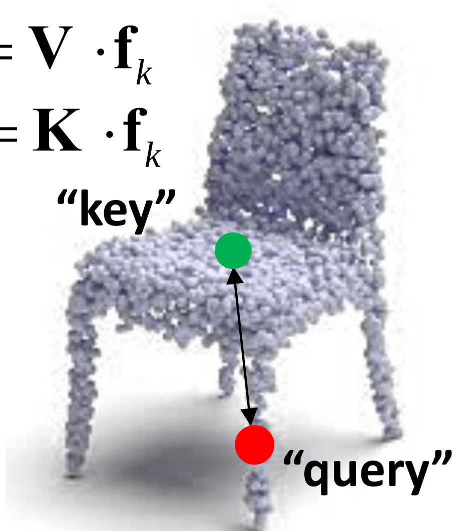
How much a point (query) is related to others (keys)?

Attention "score": $a(\mathbf{p}_q, \mathbf{p}_k) = k(\mathbf{p}_k) \cdot q(\mathbf{p}_q)$

(dot product between key-query vectors => "scalar" attention)

$v(\mathbf{p}_k) = \mathbf{V} \cdot \mathbf{f}_k$

$k(\mathbf{p}_k) = \mathbf{K} \cdot \mathbf{f}_k$



$q(\mathbf{p}_q) = \mathbf{Q} \cdot \mathbf{f}_q$

Note: there are other variants of attention e.g, instead of dot product, another way to compare keys and queries is through subtraction. This is called "vector" attention. Can also compare point coordinates as well:

$$\mathbf{a}(\mathbf{p}_q, \mathbf{p}_k) = k(\mathbf{p}_k) - q(\mathbf{p}_q) \quad or$$

$$\mathbf{a}(\mathbf{p}_q, \mathbf{p}_k) = MLP(\ \ k(\mathbf{p}_k) - q(\mathbf{p}_q)\ \ ) \quad or$$

$$\mathbf{a}(\mathbf{p}_q, \mathbf{p}_k) = MLP(\ \ k(\mathbf{p}_k) - q(\mathbf{p}_q) + MLP(\mathbf{p}_k - \mathbf{p}_q)\ \ )$$

Point Transformers, ICCV 2021

Attention for point clouds

How much a point (query) is related to others (keys)?

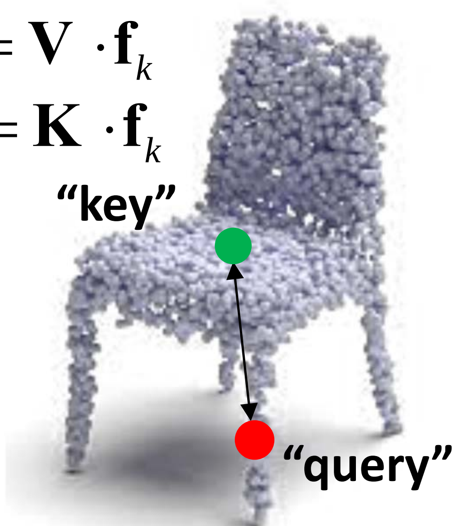
Attention "score":

$$\hat{a}(\mathbf{p}_q, \mathbf{p}_k) = \frac{\exp\{\ a(\mathbf{p}_q, \mathbf{p}_k)\}}{\sum_{k'} \exp\{\ a(\mathbf{p}_q, \mathbf{p}_{k'})\ \}}$$

(i.e., use softmax for scalar
or vector attention)

$$v(\mathbf{p}_k) = \mathbf{V}\ \cdot \mathbf{f}_k$$

$$k(\mathbf{p}_k) = \mathbf{K}\ \cdot \mathbf{f}_k$$



$$q(\mathbf{p}_q) = \mathbf{Q}\ \cdot \mathbf{f}_q$$

Attention for point clouds

How much a point (query) is related to others (keys)?

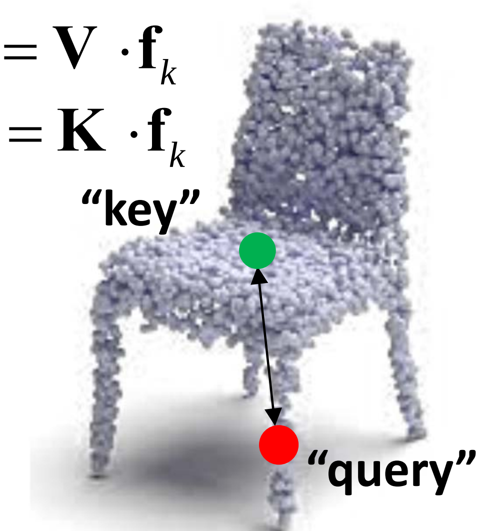
Attention "score":

(i.e., use softmax for scalar
or vector attention)

$$\hat{a}(\mathbf{p}_q, \mathbf{p}_k) = \frac{\exp\{\ a(\mathbf{p}_q, \mathbf{p}_k)\}}{\sum_{k'} \exp\{\ a(\mathbf{p}_q, \mathbf{p}_{k'})\ \}}$$

$$v(\mathbf{p}_k) = \mathbf{V} \cdot \mathbf{f}_k$$

$$k(\mathbf{p}_k) = \mathbf{K} \cdot \mathbf{f}_k$$



$$q(\mathbf{p}_q) = \mathbf{Q} \cdot \mathbf{f}_q$$

Problem with scalar attention: As the number of dimensions D of the input feature vector gets large, the variance of the dot product increased ...the input to softmax gets high values... the softmax is too peaked ...
=> tiny gradients

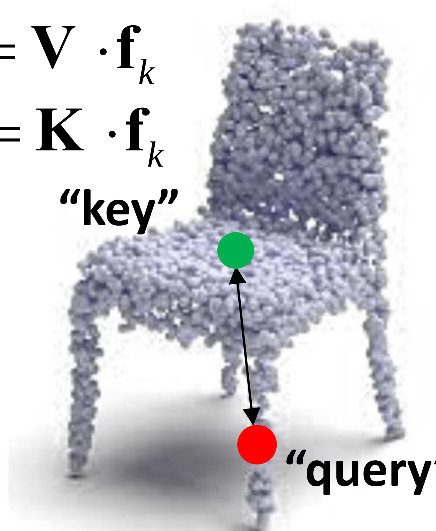
Attention for point clouds

How much a point (query) is related to others (keys)?

Attention "score":
(i.e., use softmax)

$$\hat{a}(\mathbf{p}_q, \mathbf{p}_k) = \frac{\exp\{\, a(\mathbf{p}_q, \mathbf{p}_k) / \sqrt{D} \}}{\sum_{k'} \exp\{\, a(\mathbf{p}_q, \mathbf{p}_{k'}) / \sqrt{D} \,\}}$$

$$v(\mathbf{p}_k) = \mathbf{V} \cdot \mathbf{f}_k$$
$$k(\mathbf{p}_k) = \mathbf{K} \cdot \mathbf{f}_k$$



$$q(\mathbf{p}_q) = \mathbf{Q} \cdot \mathbf{f}_q$$

"Scaled" dot product (scalar) attention

Vaswani, Attention Is All You Need, 2017

Attention for point clouds

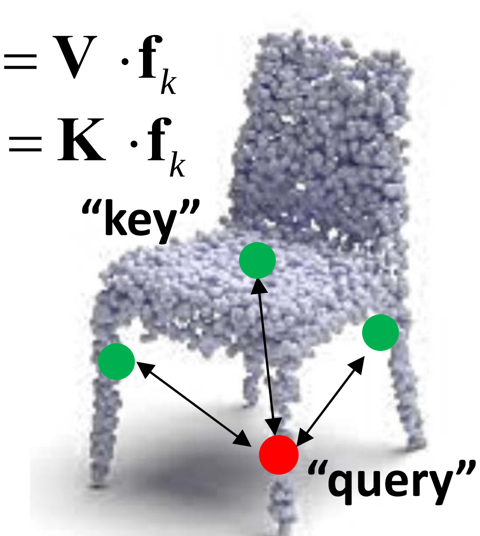
Compute a new encoding for query point as a
weighted sum of values of key points:

$$Scalar\ \ attention: \mathbf{f}_q\,' = \sum_k \hat{a}(\mathbf{p}_q, \mathbf{p}_k)\, v(\mathbf{p}_k)$$

$$Vector\ attention: \mathbf{f}_q\,' = \sum_k \hat{\mathbf{a}}(\mathbf{p}_q, \mathbf{p}_k) \odot v(\mathbf{p}_k)$$

$$v(\mathbf{p}_k) = \mathbf{V} \cdot \mathbf{f}_k$$

$$k(\mathbf{p}_k) = \mathbf{K} \cdot \mathbf{f}_k$$



$$q(\mathbf{p}_q) = \mathbf{Q} \cdot \mathbf{f}_q$$

Attention for point clouds

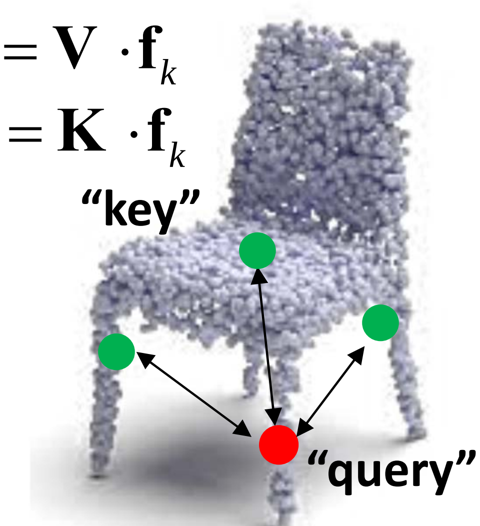
Compute a new encoding for query point as a
weighted sum of values of key points:

$$Scalar\ \ attention: \mathbf{f}_q\,' = \sum_k \hat{a}(\mathbf{p}_q, \mathbf{p}_k)\, v(\mathbf{p}_k)$$

$$Vector\ attention: \mathbf{f}_q\,' = \sum_k \hat{\mathbf{a}}(\mathbf{p}_q, \mathbf{p}_k) \odot v(\mathbf{p}_k)$$

$$v(\mathbf{p}_k) = \mathbf{V} \cdot \mathbf{f}_k$$

$$k(\mathbf{p}_k) = \mathbf{K} \cdot \mathbf{f}_k$$



$$q(\mathbf{p}_q) = \mathbf{Q} \cdot \mathbf{f}_q$$

... this can be further processed by an MLP and added back to the input feature vector as a residual:

... The result can be processed by more attention layers.

Multi-head attention

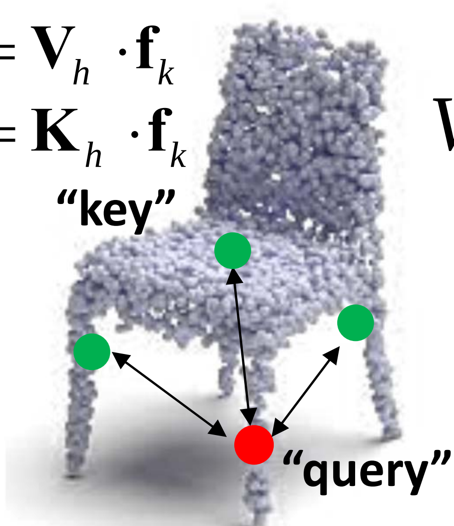
Learn different query, key, value transformations for each attention "head".

$$Scalar\ \ attention: \mathbf{f}_{q,h}\,' = \sum_k \hat{a}_h(\mathbf{p}_q, \mathbf{p}_k)\, v_h(\mathbf{p}_k)$$

$$Vector\ attention: \mathbf{f}_{q,h}\,' = \sum_k \hat{\mathbf{a}}_h(\mathbf{p}_q, \mathbf{p}_k) \odot v_h(\mathbf{p}_k)$$

$$v_h(\mathbf{p}_k) = \mathbf{V}_h \cdot \mathbf{f}_k$$

$$k_h(\mathbf{p}_k) = \mathbf{K}_h \cdot \mathbf{f}_k$$



$$q_h(\mathbf{p}_q) = \mathbf{Q}_h \cdot \mathbf{f}_q$$

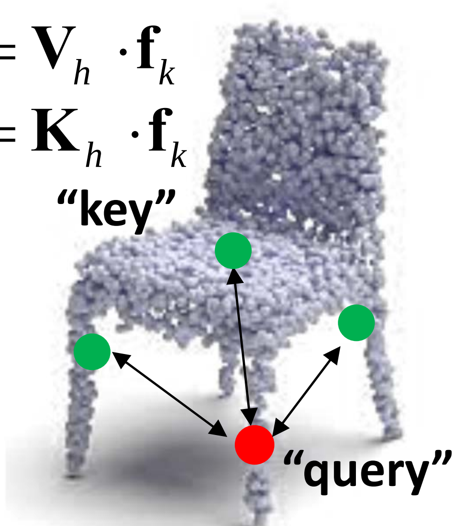
... then concatenate the resulting features from all heads, process them with a MLP

Quadratic complexity!

Comparing each query (N points) with every key (N points) yields **quadratic complexity**!

$$v_h(\mathbf{p}_k) = \mathbf{V}_h \cdot \mathbf{f}_k$$

$$k_h(\mathbf{p}_k) = \mathbf{K}_h \cdot \mathbf{f}_k$$



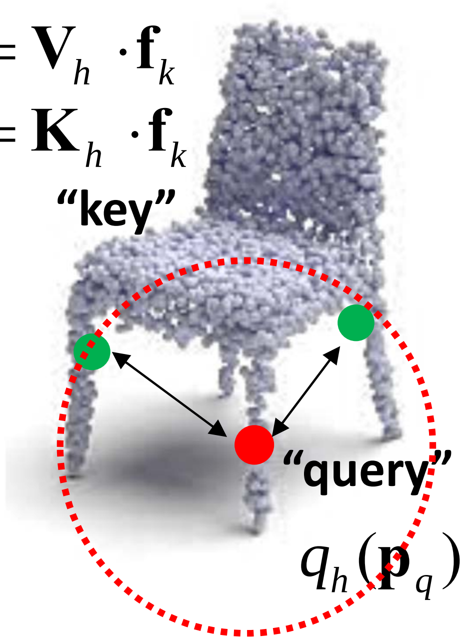
$$q_h(\mathbf{p}_q) = \mathbf{Q}_h \cdot \mathbf{f}_q$$

Quadratic complexity!

Common strategy is to restrict keys within a k-NN neighborhood (or ball) around the query point

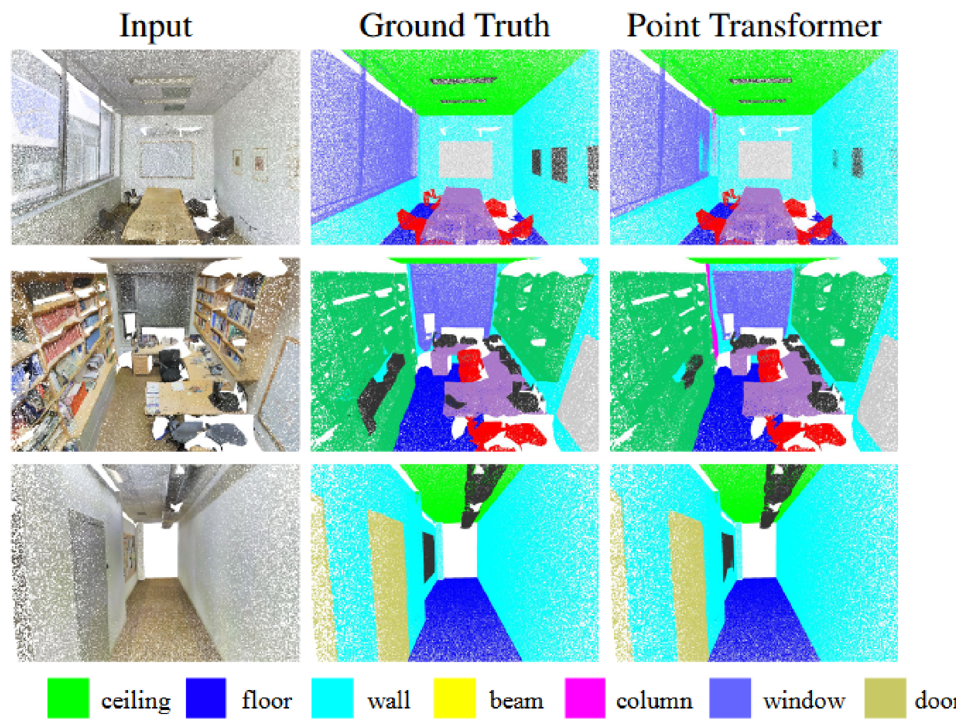
$$v_h(\mathbf{p}_k) = \mathbf{V}_h \cdot \mathbf{f}_k$$

$$k_h(\mathbf{p}_k) = \mathbf{K}_h \cdot \mathbf{f}_k$$



$$q_h(\mathbf{p}_q) = \mathbf{Q}_h \cdot \mathbf{f}_q$$

Scene labeling results

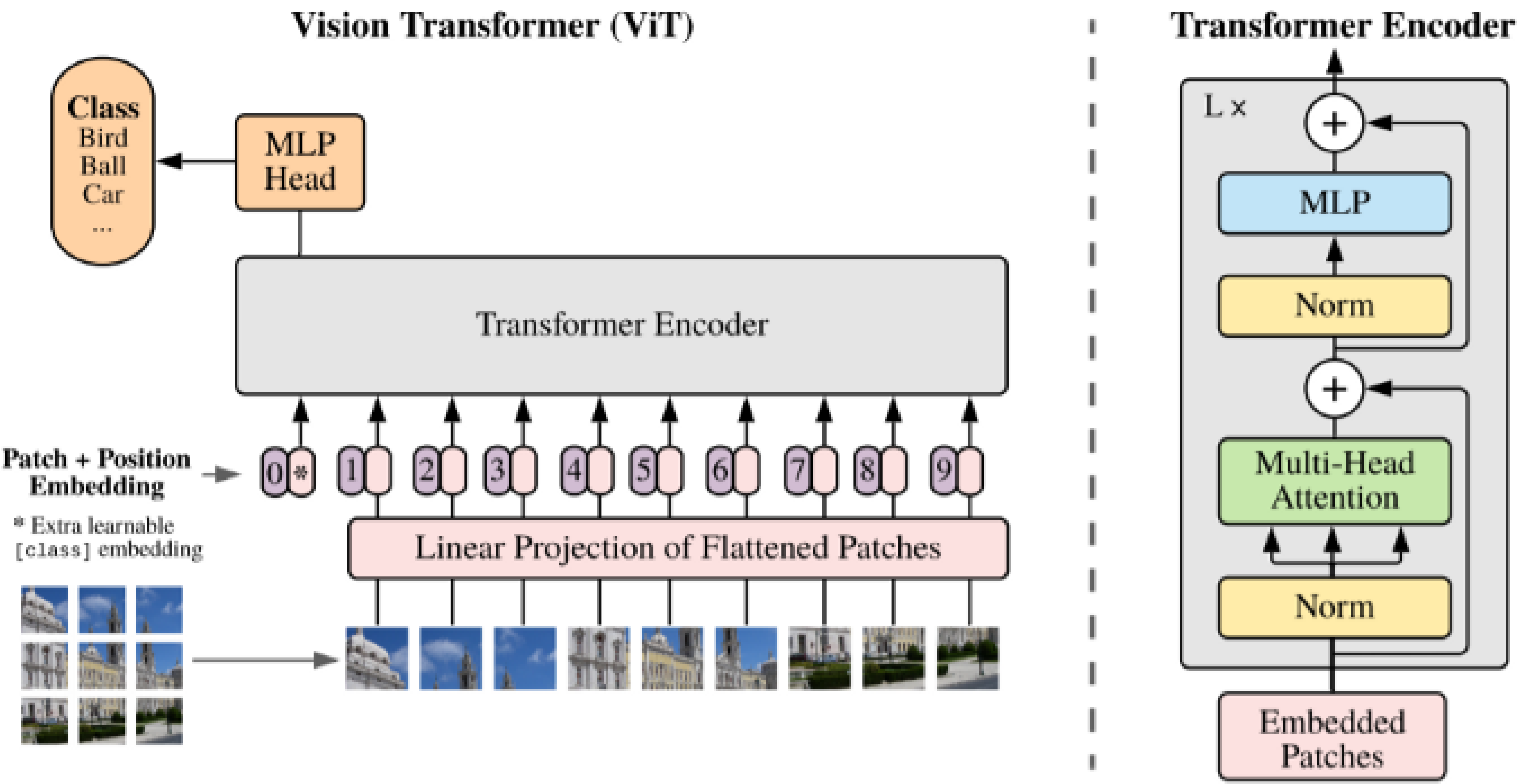


Method	OA	mAcc	mIoU	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PointNet [25]	–	49.0	41.1	88.8	97.3	69.8	0.1	3.9	46.3	10.8	59.0	52.6	5.9	40.3	26.4	33.2
SegCloud [36]	–	57.4	48.9	90.1	96.1	69.9	0.0	18.4	38.4	23.1	70.4	75.9	40.9	58.4	13.0	41.6
TangentConv [35]	–	62.2	52.6	90.5	97.7	74.0	0.0	20.7	39.0	31.3	77.5	69.4	57.3	38.5	48.8	39.8
PointCNN [20]	85.9	63.9	57.3	92.3	98.2	79.4	0.0	17.6	22.8	62.1	74.4	80.6	31.7	66.7	62.1	56.7
SPGraph [15]	86.4	66.5	58.0	89.4	96.9	78.1	0.0	42.8	48.9	61.6	84.7	75.4	69.8	52.6	2.1	52.2
PCCN [42]	–	67.0	58.3	92.3	96.2	75.9	0.3	6.0	69.5	63.5	66.9	65.6	47.3	68.9	59.1	46.2
PAT [50]	–	70.8	60.1	93.0	98.5	72.3	1.0	41.5	85.1	38.2	57.7	83.6	48.1	67.0	61.3	33.6
PointWeb [55]	87.0	66.6	60.3	92.0	98.5	79.4	0.0	21.1	59.7	34.8	76.3	88.3	46.9	69.3	64.9	52.5
HPEIN [13]	87.2	68.3	61.9	91.5	98.2	81.4	0.0	23.3	65.3	40.0	75.5	87.7	58.5	67.8	65.6	49.4
MinkowskiNet [37]	–	71.7	65.4	91.8	98.7	86.2	0.0	34.1	48.9	62.4	81.6	89.8	47.2	74.9	74.4	58.6
KPConv [37]	–	72.8	67.1	92.8	97.3	82.4	0.0	23.9	58.0	69.0	81.5	91.0	75.4	75.3	66.7	58.9
PointTransformer	**90.8**	**76.5**	**70.4**	94.0	98.5	86.3	0.0	38.0	63.4	74.3	89.1	82.4	74.3	80.2	76.0	59.3

Table 1. Semantic segmentation results on the S3DIS dataset, evaluated on Area 5.

Point Transformers, ICCV 2021

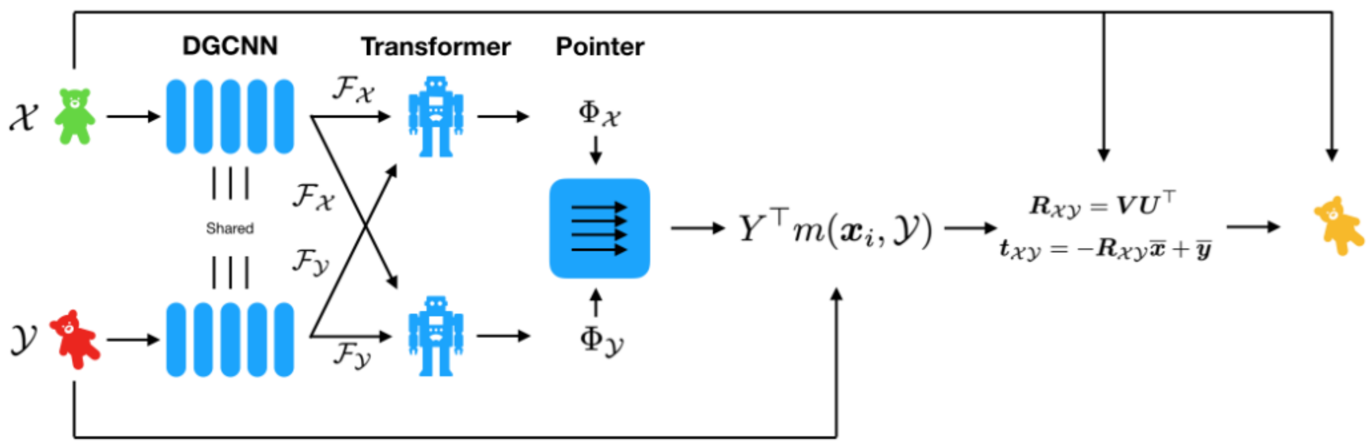
"Vision Transformers"

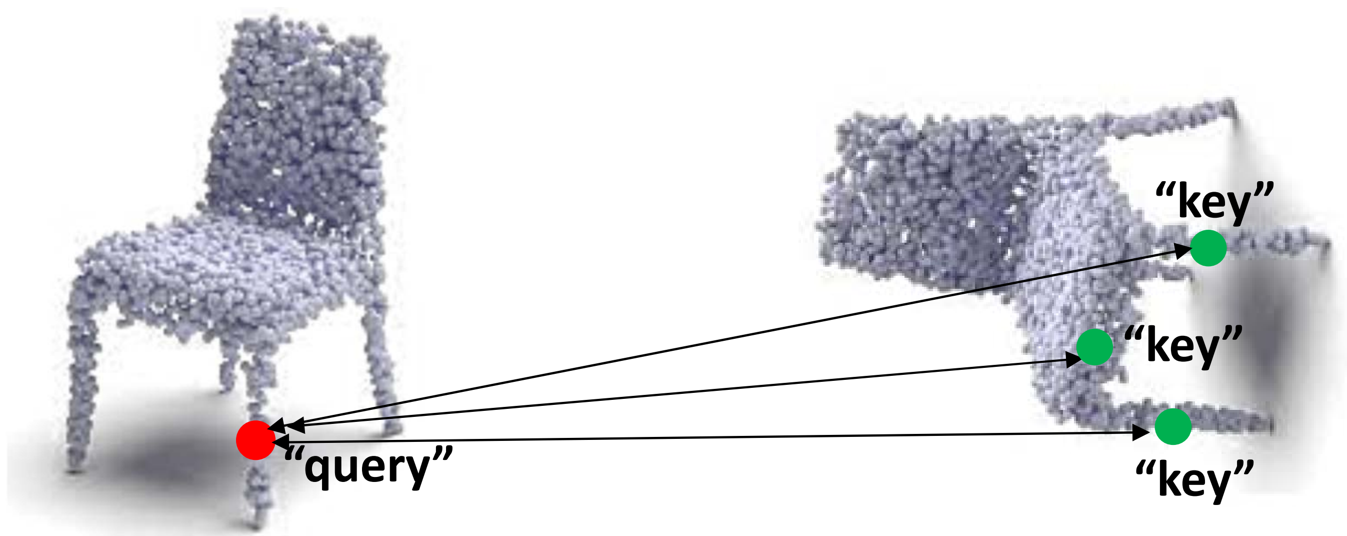


Dosovitskiy, ICLR 2021, https://github.com/google-research/vision_transformer https://www.youtube.com/watch?v=TrdevFK_am4

Cross-shape attention

Attention can be used also as point similarity between point clouds useful for registration.





Deep Closest Point: Learning Representations for Point Cloud Registration, ICCV 2019, Wang and Solomon

Point-based 3D Deep Learning Advantages

- **Well-suited to analyze point clouds**
  (no pre-processing conversion to views/voxels are needed that may create artifacts)

- **Low memory requirements / highly efficient computationally**

- **Robust to varying sampling density**

Point-based 3D Deep Learning Disadvantages

- **Mainly use 3D shape/scene training data**
  (not that abundant as 2D image data)

- **Harder to implement**
  (require generalizations of traditional image convolution)