Visual AI

CPSC 533R

Lecture 7. Representing and learning shapes

Helge Rhodin

Point clouds

Representation: A collection of 3D points

- Size: N x D (Number of points, space dimension)
- Sparse 3 D locations (usually, can be in a higher-dimensional)
  - Continuous and adaptive detail

Benefits

- Well suited for structure from motion form keypoints
- Compact representation of sparse keypoint locations
  - human joints, object edges, …
- Ordered point clouds carry semantics (e.g., first point is the head, the second the neck position)

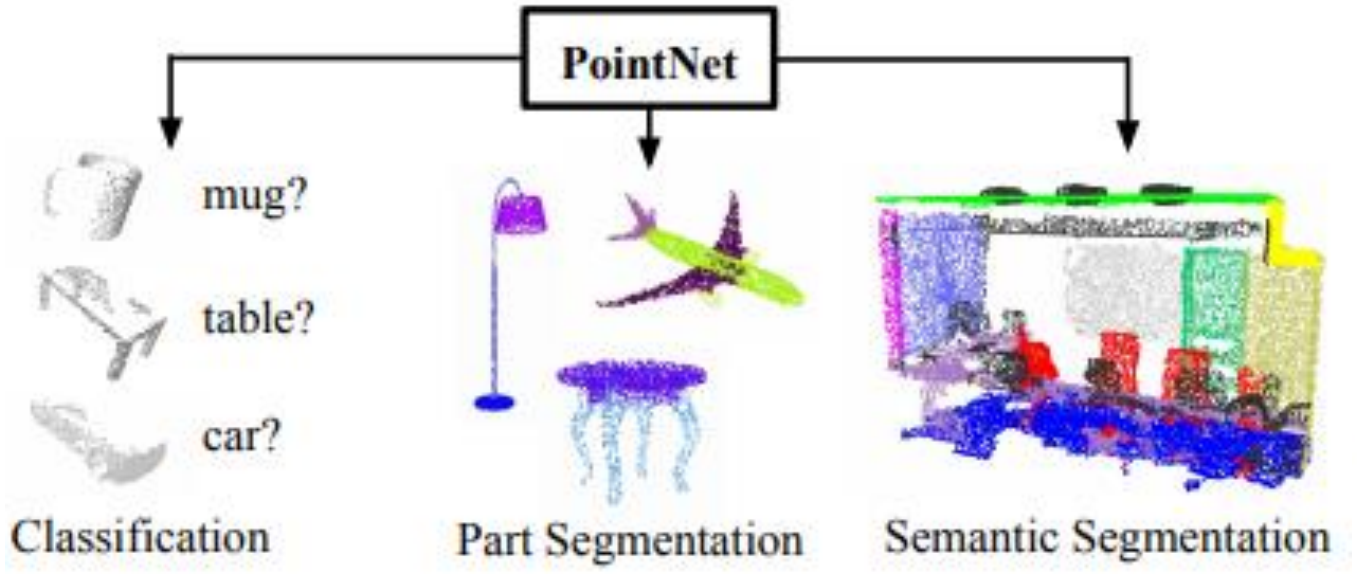
Drawbacks

- Unstructured, not well suited for convolutions etc.
- No orientation information

[Snavely et al., Photo Tourism: Exploring Photo Collections in 3D]

PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

Applications

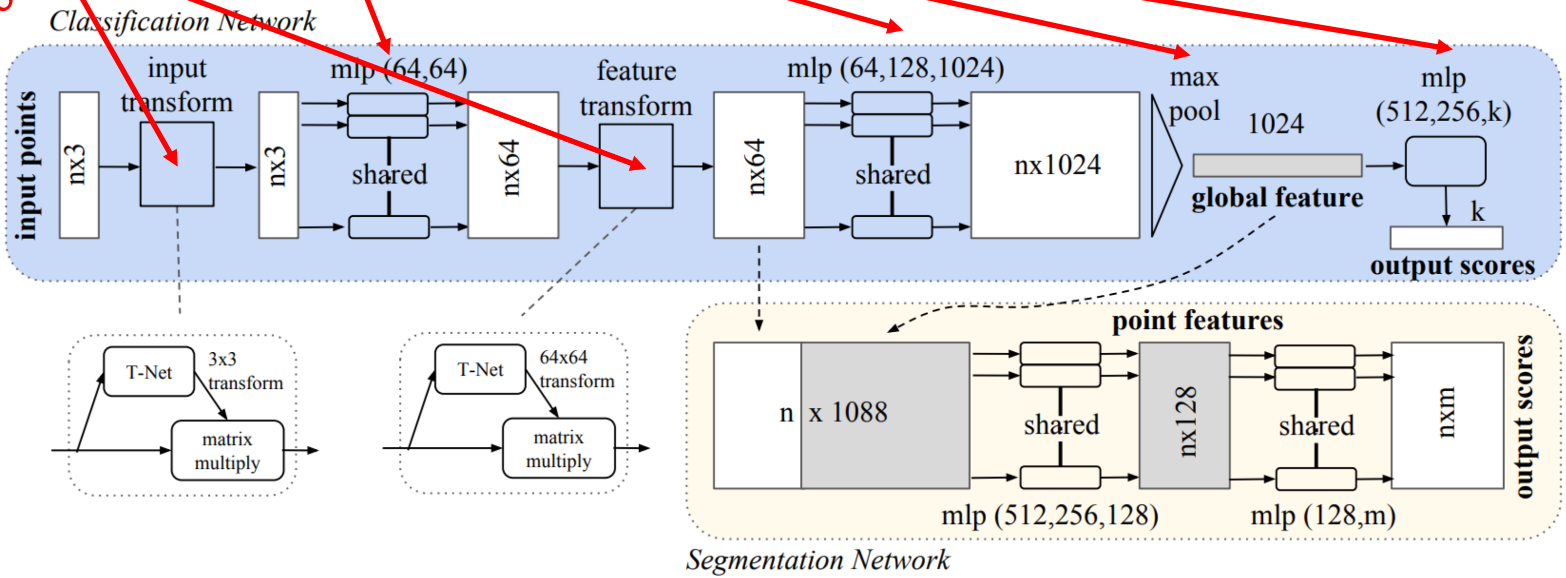


PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

A network architecture to make point cloud processing invariant to

- the point cloud order
- global rigid transform.

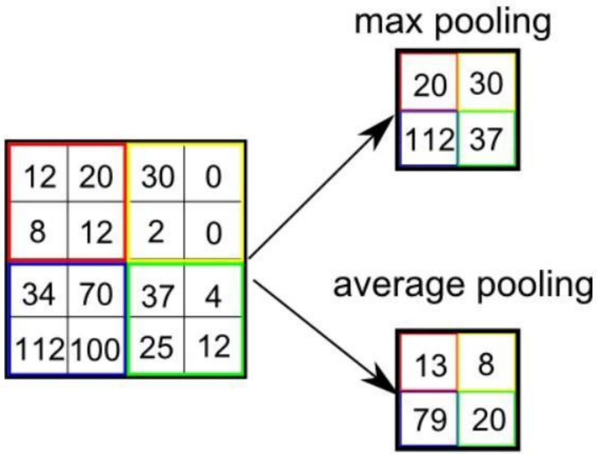
affine transformation

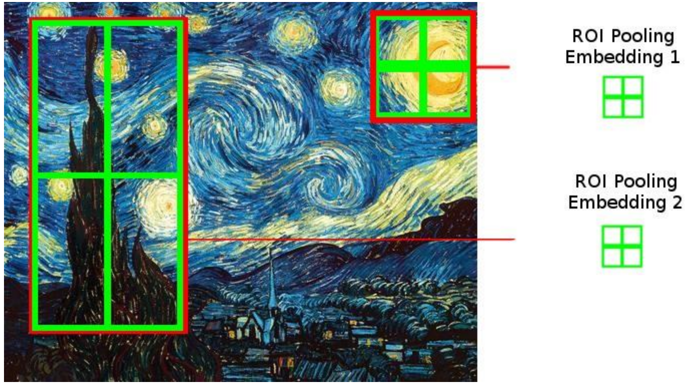


Pooling layers

- accumulate values from several cells
    - average
    - max
        - min?
        - median?
- usually over a fixed number of pixels
    - e.g. 2x2 reduces resolution by 2

- variants:
    - global pooling: accumulate over all pixels
    - Region of Interest (RoI) pooling
        - split region into regular number of cells
            - pool within each cell
            - dynamic!





https://medium.com/xplore-ai/implementing-attention-in-tensorflow-keras-using-roi-pooling-992508b6592b

Volumetric representations

Recap: Voxel representations

Idea: A 3D tensor that encodes occupancy

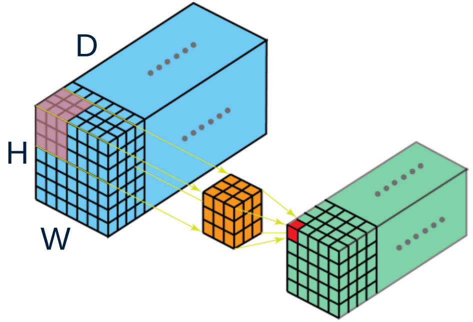
- stores binary values
  - occupied or empty cell

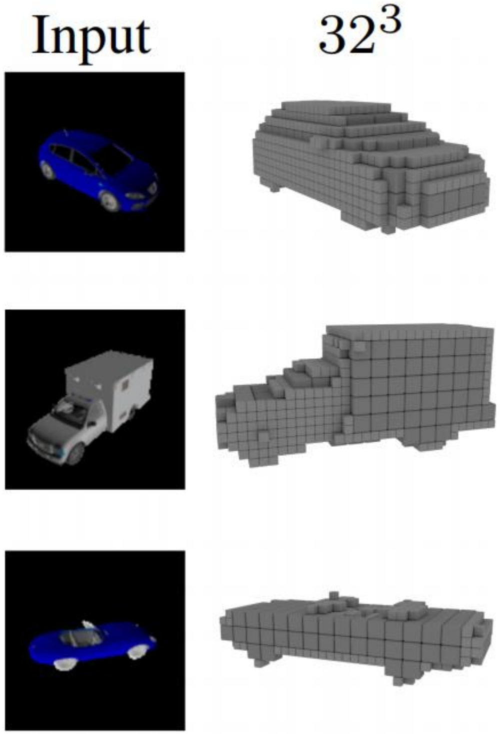
Size: C x D x H x W (C: channels, D: depth, H: height, W: width)

Batched size: N x D x H x W (N: number of elements in mini batch)

Benefits: We can apply 3D convolutions

- A generalization to 2D convolutions with a 3D kernel





[Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs]

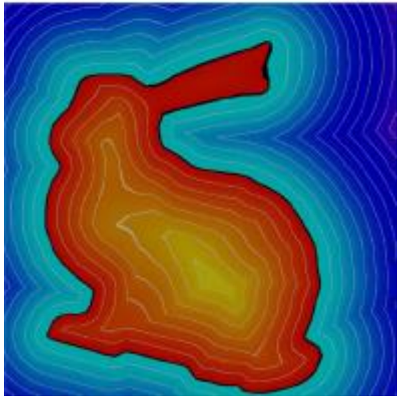
Drawback:

- cubic in memory footprint and computational complexity

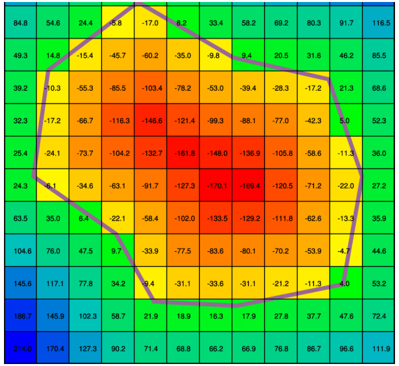
Signed Distance Field (SDF)

- input domain: dimension equal to the dimension of the space
  - usually two or three-dimensional
- output domain: a scalar
  - negative for inside of the object
  - positive outside

- continuous SDF: defined by a parametric function
  - e.g., sum of Gaussians, neural network
- discrete SDF: defined on a grid
  - e.g. 2D grid or 3D grid
- yields additional information on voxel grid: distance to surface
- easy to display SDF in color code
  (red to blue = negative to positive)
- non-trivial to reconstruct the exact shape boundary



Continuous SDF



Discrete SDF

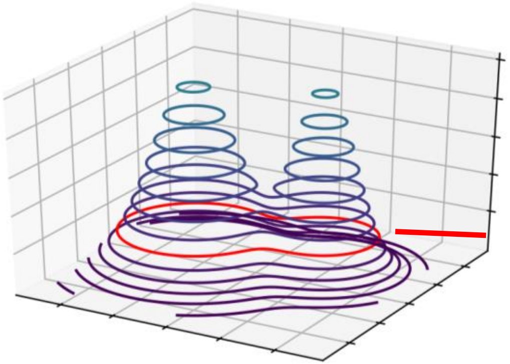
Implicit functions

Idea: define complex shapes as the zero-crossing of a function

Size: W (the number of parameters of the function)

- independent of output space dimension!

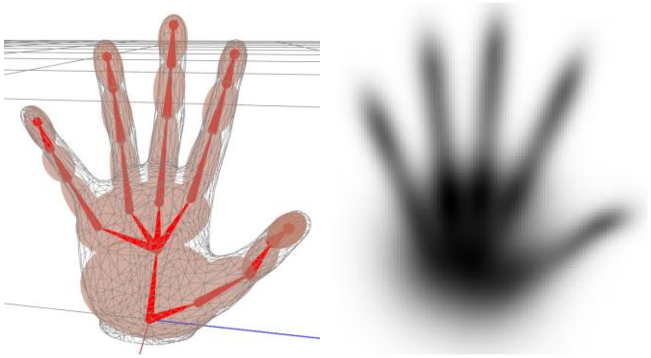
- Any parametric function works
  - e.g., mixtures of n Gaussian distributions
    with position mu and covariance Sigma



$$f(x) = \sum_{i=1}^{n} G(x, \mu_i, \sigma_i)$$

contour line / zero crossing

  - a neural network?!



[Real-time Hand Tracking Using a Sum of Anisotropic Gaussians Model]

Implicit functions through NNs

[Saito et al., PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization]

Idea: Train a neural network that takes an image as well as a 3D query point as input and outputs:

- negative for positions **inside** the object
- positive **outside** the object
- reconstruct by densely sampling / marching cubes algorithm

Advantage:

- No explicit limit on resolution (only limited by NN capacity)

Disadvantage:

- Reconstruction requires many network evaluations, its slow!

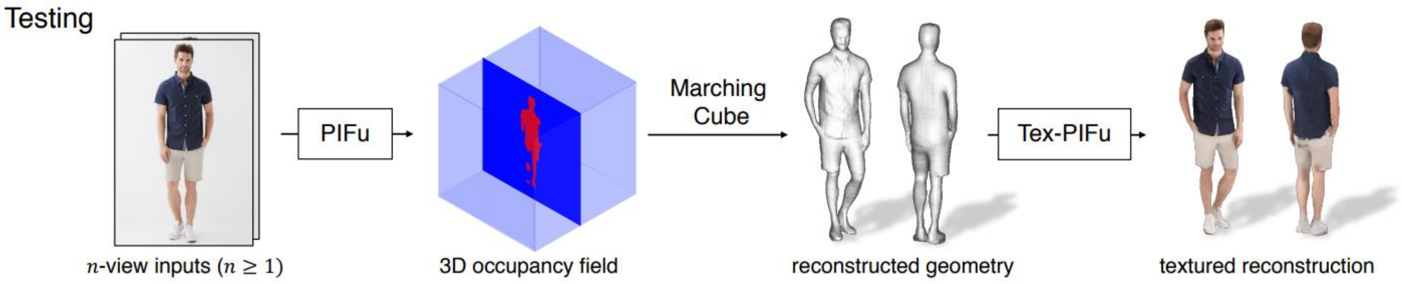
occupancy output
(positive within the object hull)

$$f(x) = \mathrm{CNN}_\theta(x, \mathbf{I})$$

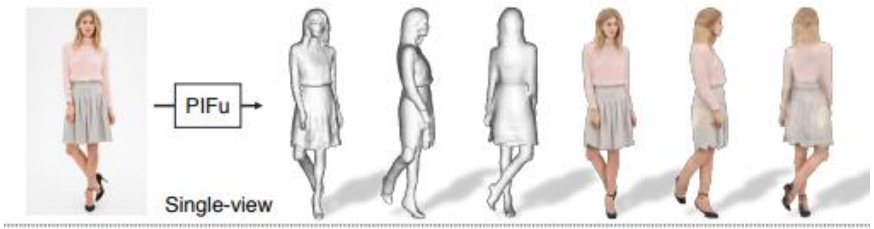
3D query location

Image to reconstruct

Not straightforward to train…
wait for the paper presentation



Additional examples



[Saito et al., PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization]

Not straightforward to train…
wait for the paper presentation

Ray-tracing and NERF

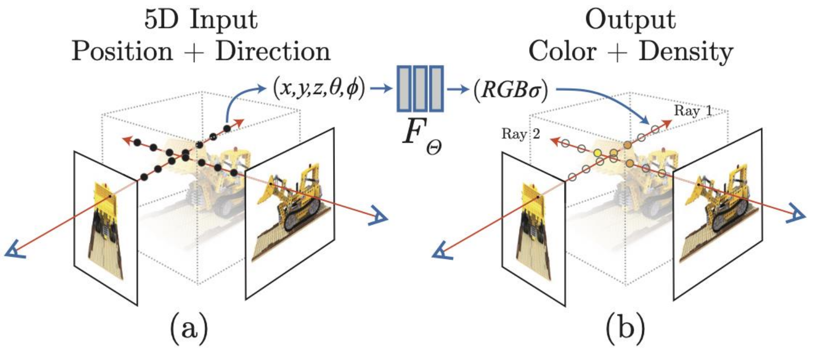
- Radiance Field
  - for every place & direction, output the radiance

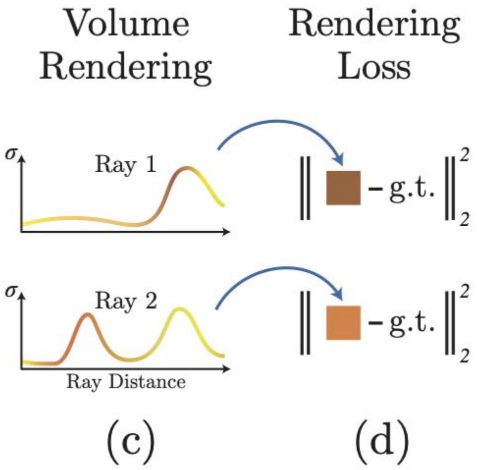
$$L : \mathbb{R}^3 \times S^2 \rightarrow \mathbb{R}^3$$

  - radiance: *'outgoing light', here in RGB space*

- Ray-trace the light reaching the camera
  - ray from camera to scene
  - accumulate visibility * radiance
    - visibility: a function of opacity/occupancy

- "Neural": learn the radiance & occupancy field with a NN
  - like the implicit function





More details in the paper presentation of:
[Mildenhall et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis]

Perspective spatial transformer

Goal: self-supervised training of reconstruction

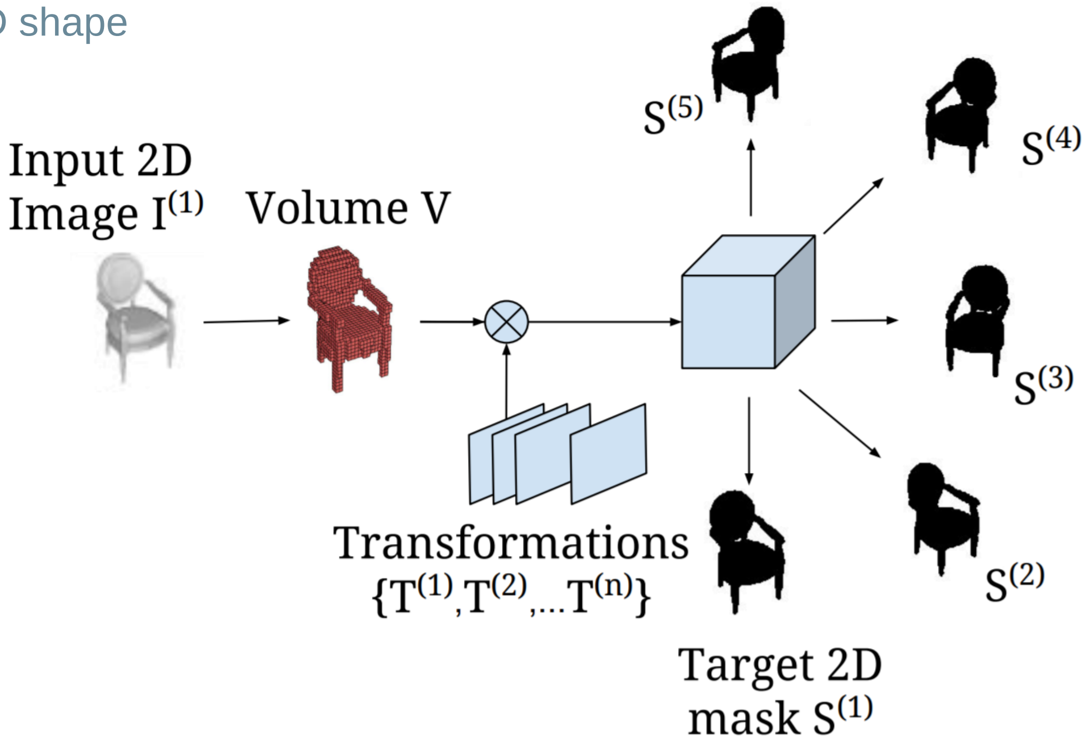
Given: set of multi-view images at training time

Training: a neural network that predicts a 3D shape

- consistent with all views
- using silhouette constraints

Requires:

- 2D to 3D correspondences
- a perspective 3D spatial transformer

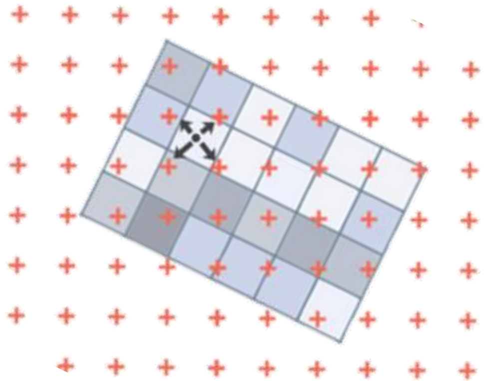


Sampling and interpolation

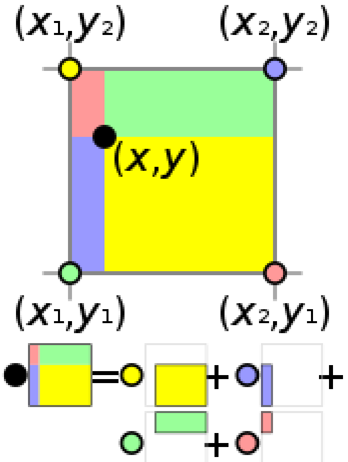
Re-sampling of images/features

1. grid generation
   - parametric
     - differentiable
2. grid sampling
   - bilinear interpolation
     - differentiable

grid (gray) on image grid (red)



- still efficient
  (compared to non-differentiable cropping and soft windows)
- moderate smoothness guarantees
  (piecewise linear)

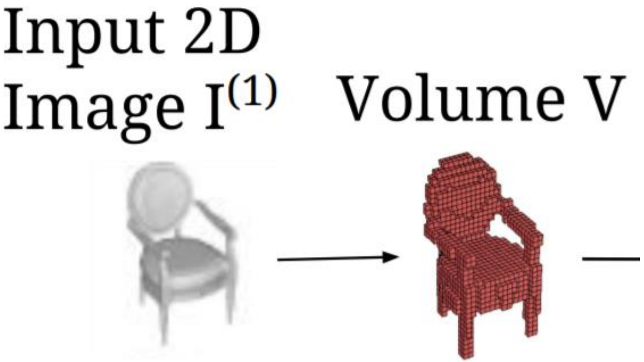


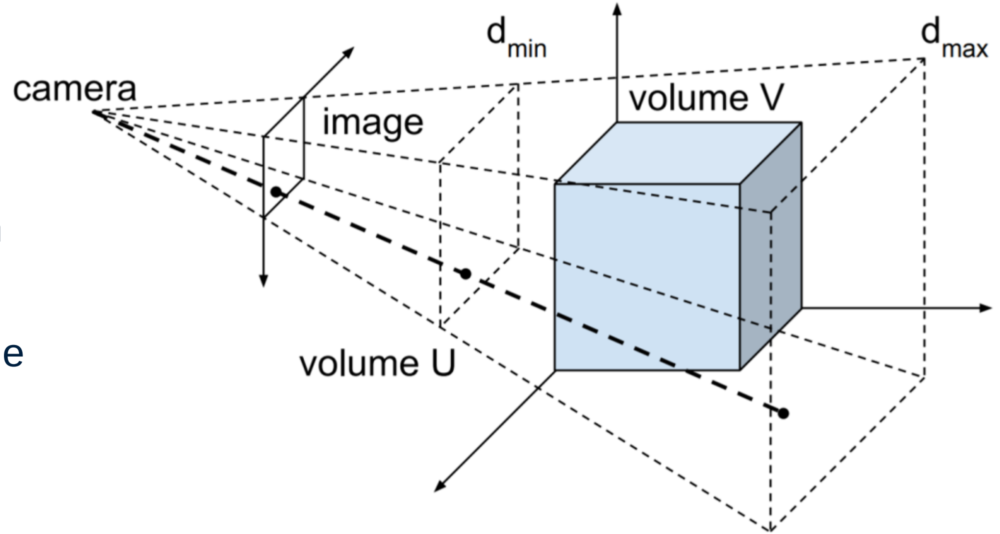
Bilinear interpolation

Perspective spatial transformer, details

Concept:

- predict a 3D occupancy grid given the input view
- construct N 3D grids (one for each reference view)
    - pyramidal form, with
        - position and orientation of reference cameras
    - models the perspective effect
- sample the 3D volume
    - as for 2D spatial transformers, but
      by trilinear interpolation
- take the maximum along the depth direction
    - models projection
- minimize the distance of this projection to the reference image silhouette (see prev. slide)





Surface representations

Surface mesh

Representation: Vertices connected by edges forming faces

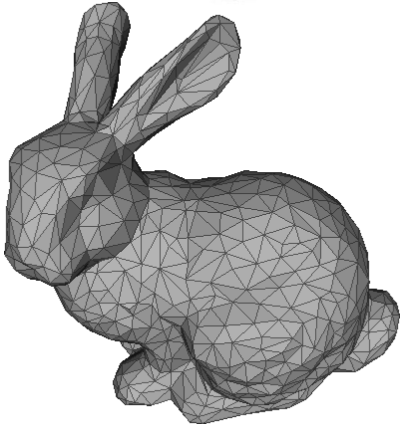
- Size: N x D + E x 2 (# points, space dimension, # edges)
- A 3D surface parametrization (can be higher-dimensional)
  - Piece-wise linear with adaptive detail; triangle faces are usual

Benefits

- Good for single and multi-view reconstruction
- Provides orientation information (surface normal)
- Graph convolutions possible

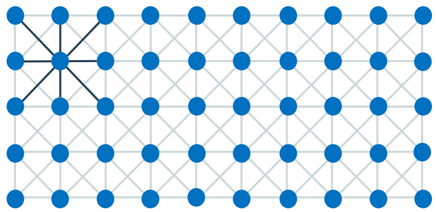
Drawbacks

- Irregular structure (number of neighbors, edge length, face area)
- Difficult to change topology
  (shape changes require to create new vertices and edges)



General graph convolution

- traditional 2D convolutions is convolution on a regular grid

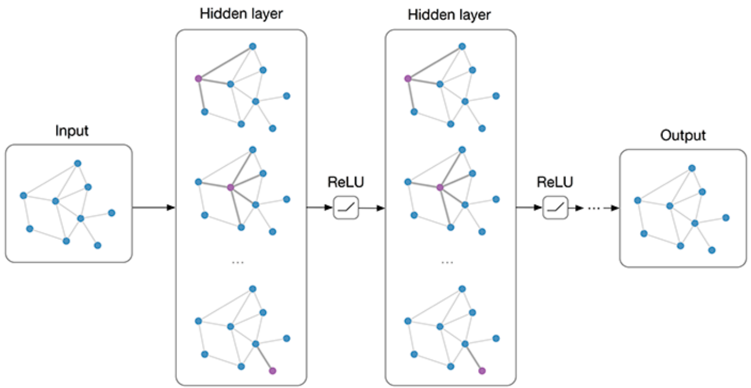


Convolution on a regular grid

Difficulties for general graph convolution
- no notion of left/right and up/down
- different number of neighbors
- distances between nodes

Solution
- per-node weight matrix for all nodes (like 1x1 conv.)
- weighted average over all neighbors (like average pooling)

$$h_i^{(l+1)} = \sigma\left(\sum_j \frac{1}{c_{ij}} h_j^{(l)} W^{(l)}\right)$$



Graph convolution network

https://tkipf.github.io/graph-convolutional-networks/

Details: Mesh Laplacian

Goal: A form of 2nd order derivative on the mesh

Laplacian for a function in 3D space:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$$

Difficulty:

- irregularity, where is left / right / up / down?

Solution:

- (weighted) average over all neighboring nodes Ni

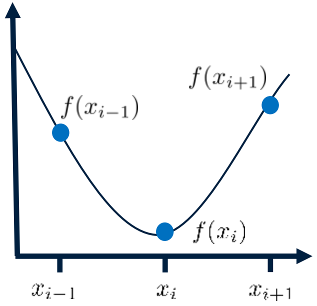
$$\mathscr{L}(\mathbf{v}_i) = \mathbf{v}_i - \frac{1}{d_i} \sum_{j \in \mathcal{N}_i} \mathbf{v}_j.$$

- Widely used to encode surface detail and to compare meshes
  - as a loss to compare surfaces

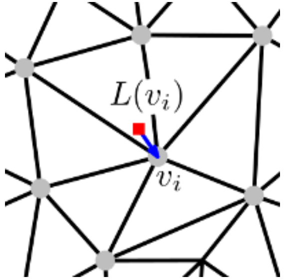
Finite differences approximation in 1D

$$f'(x_i, x_{i+1}) \approx \frac{f(x_{i+1}) - f(x_i)}{h}$$

$$f''(x_{i-1}, x_i, x_{i+1}) \approx \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2}$$

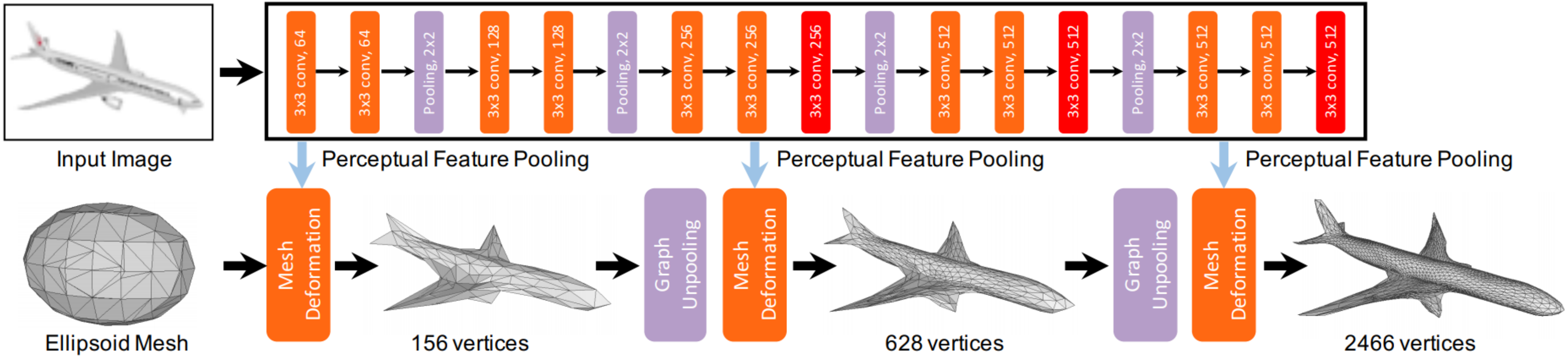


1D Laplacian



Graph Laplacian

Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images

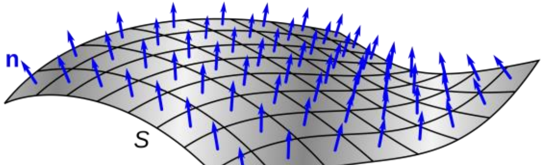


Desired:

- an output mesh that matches in position
  - Chamfer distance
- and has the same surface orientation
  - surface normal

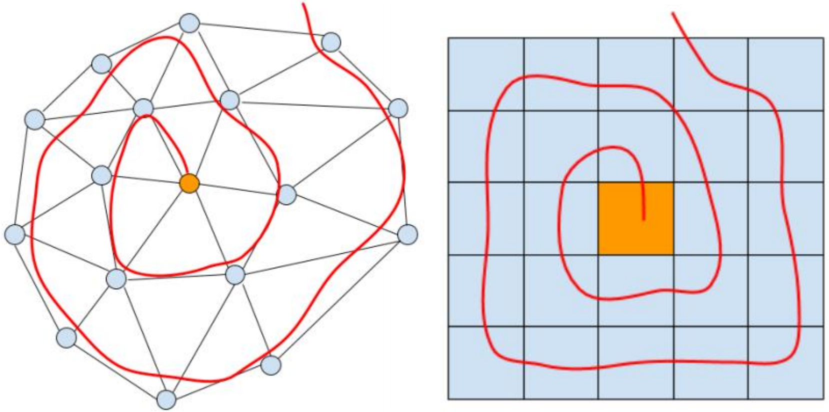
$$l_n = \sum_p \sum_{q=\arg\min_q(\|p-q\|_2^2)} \|\langle p-k, \mathbf{n}_q\rangle\|_2^2$$

- … and follows a coarse-to-fine manner
  - minimize change of Laplacian between layers



Spiral convolution

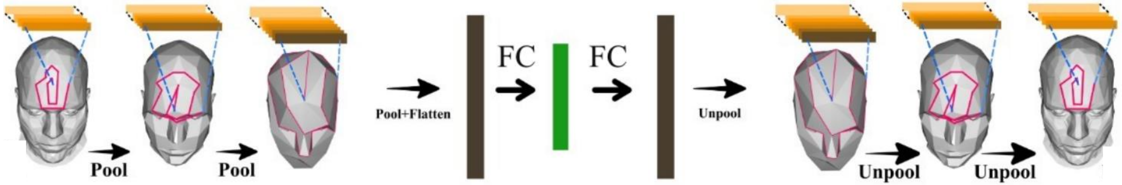
Goal: break the permutation invariance of neighbors



Idea: Order neighbors by simple rules
1. collect all neighbors (d hops in the graph)
2. pick the closest one (geodesic distance)
3. continue counterclockwise until spiral is of length k
4. multiply features h along spiral with weight matrix

$$\mathbf{h}_i^{(l+1)} = \sigma\left(h_{\text{spiral(neighbors}(i))} W^{(l)}\right)$$

Advantages:
- fixed number of points in each spiral
- efficient to compute
- anisotropic and topology-aware
- easy to optimize



Surface texture

Representation: A map that assigns a color to every point of a surface

- Size: W x H + N x 2 (W: width, H: height, N: #points for uv-coordinates)
- Dimensions: 2 D (embedded in 3D space via a mesh)
  - Discrete in space, continuous in color
- UV-coordinates attached to each mesh vertex define the spatial association

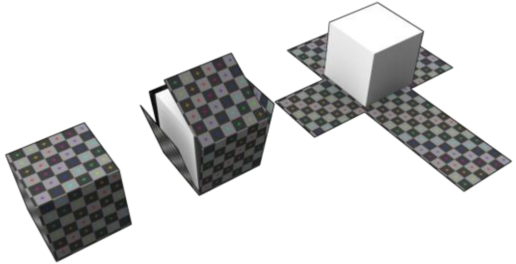
Benefits

- Appearance modelling for graphics and vision
  (e.g., rendering and reconstruction)
- Can carry more than color
  (shadowmaps, normal maps, **feature maps)**

Drawbacks

- Texture mapping (assigning vertices to texture map location) is hard
- Only a surface, not volumetric

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



UV mapping

- describe points on the texture with u,v coordinates
  - the horizontal and vertical position
- equip each vertex with the u,v coordinate
  - a 2D point

Example: teapot.obj

v -3.000000 1.800000 0.000000 *(vertex definition)*

v -2.991600 1.800000 -0.081000

…

vt 0.000100 0.000100 *(uv texture coordinates)*

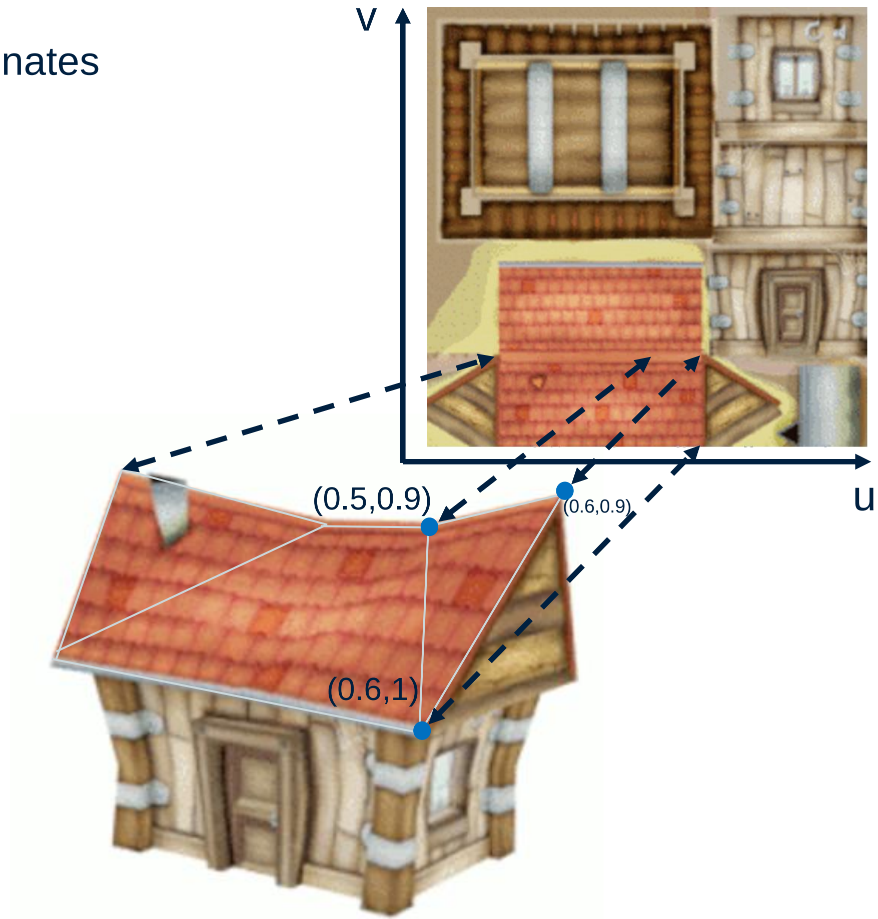
vt 0.999900 0.000100

…

f 1252 1248 1122 *(edges of a triangle/face)*

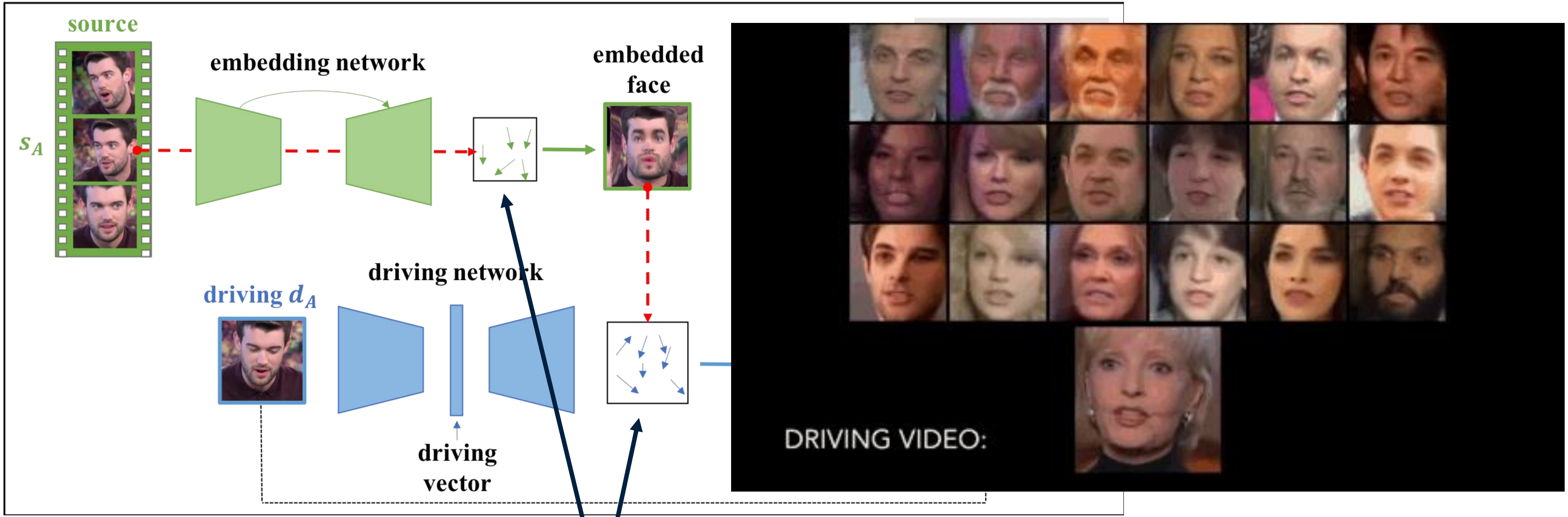
f 1027 1035 1133

…



Example: mapping a face to a texture

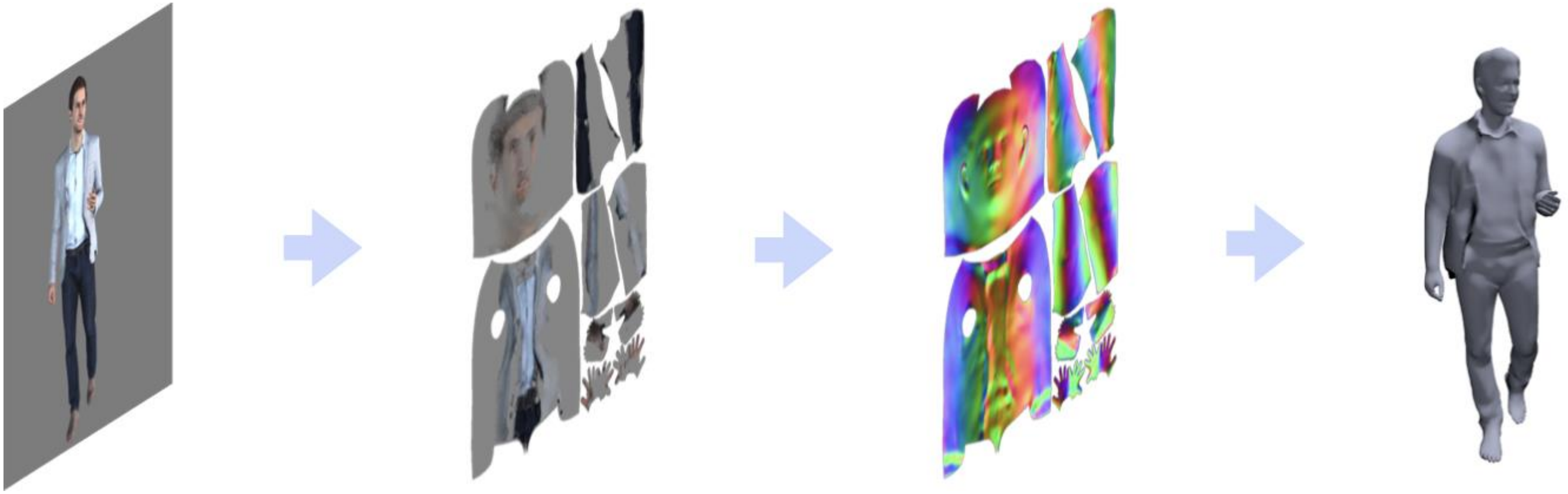
Wiles et al., X2Face: A network for controlling face generation by using images, audio, and pose codes



a form of uv mapping

Tex2Shape: Detailed Full Human Body Geometry From a Single Image

- Convolutional detail estimation via texture and normal maps



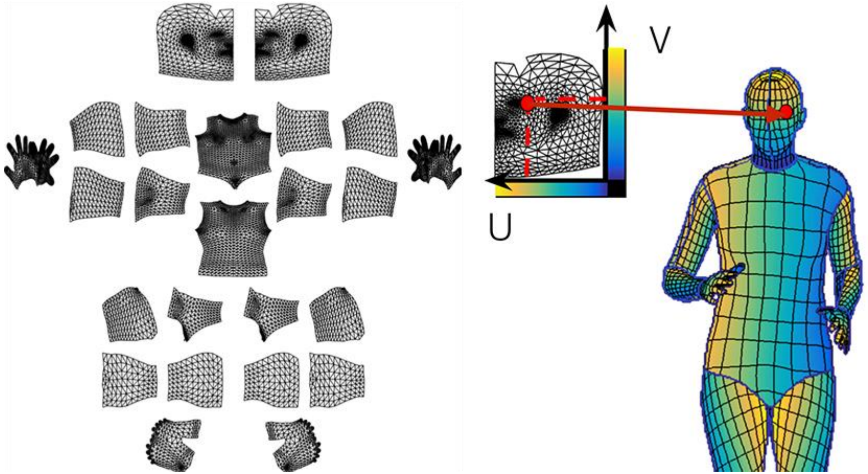
Dense Pose: Dense Human Pose Estimation In The Wild

Issue: Heatmap representations don’t generalize well to many points (one map per point)

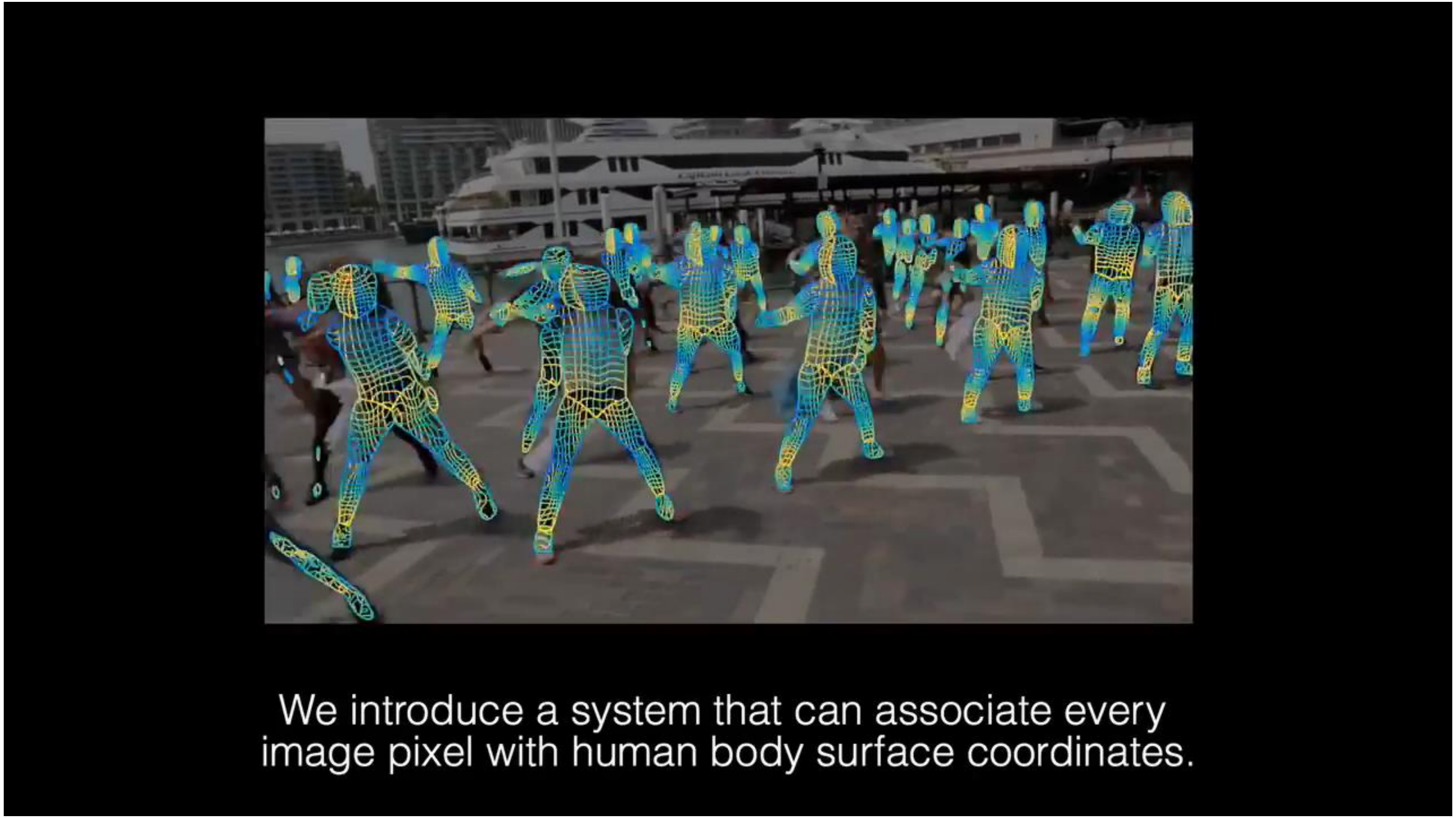
Idea: Encode locations as continuous value

- as u,v coordinates
- generalizes well to multiple people

[Dense Pose: Dense Human Pose Estimation In The Wild]



Dense Pose results



Hybrid representations

3D ‘uv coordinates’

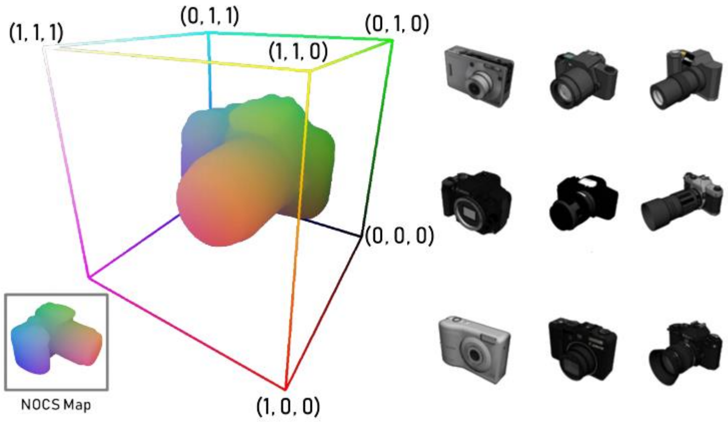
Idea: Learn to map to 3D coordinates

Solution:

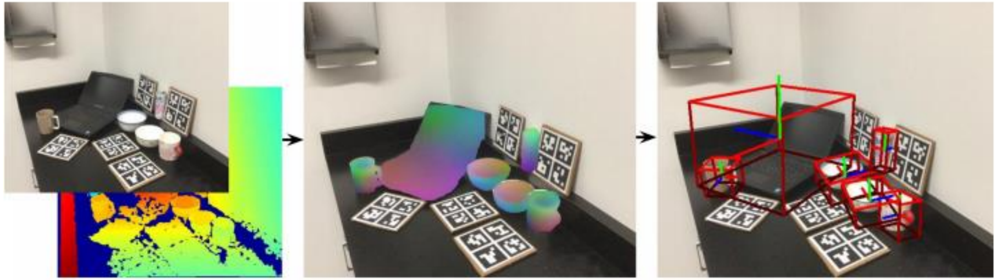
- a generalization of uv-coordinates in 3D

Benefits:

- compact, continuous, accurate



[Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation]



(a) Input: Single RGB-D Image

(b) Normalized Object Coordinate Space (NOCS) Map Estimation

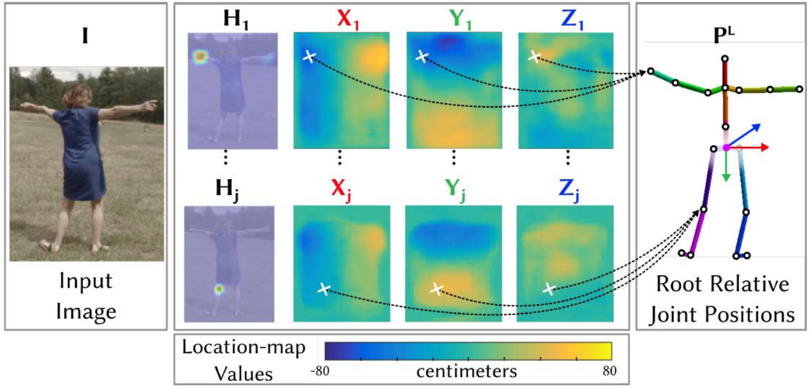
(c) Output: Category-Level 6D Pose and Size

Location maps

Idea: Predict 3D pose in a convolutional manner

Implementation:

1. predict three location maps alongside the heatmap H
   - respectively one for the x,y,z positon
2. retrieve the arg max of the heatmap (2D joint location)
3. Read out the x,y,z maps at the predicted 2D location



Admantages:

- fully convolutional networks, which apply to varying image resolution
- (convolutional) operations are centered around the area of interest (joints)
- generalized well to multiple persons

VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera

- Using location maps
- A combination of feed forward prediction with NNs and optimization of skeleton parameters