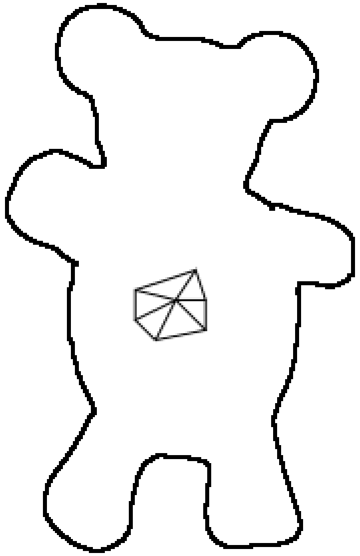
CNN Applications - 3D Mesh Labeling via CNNs

Guo et al.

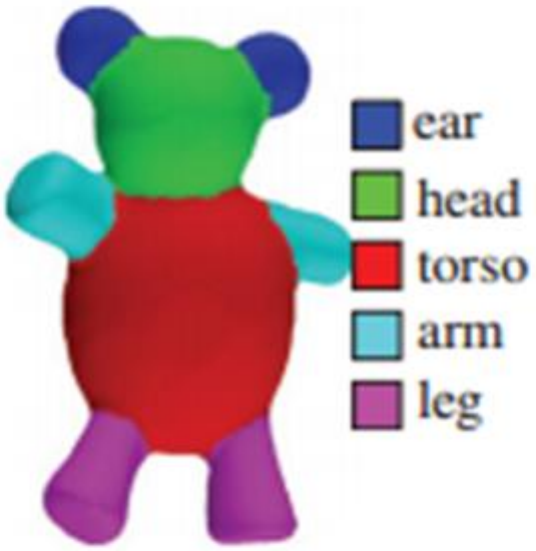
Images and materials sourced off:
3D Mesh Labeling via Deep CNNs – Guo et al.
The Princeton Shape Benchmark – Shilane et al.
Learning 3D Mesh Segmentation and Labeling – Kalogerakis et al.
Shape matching and recognition using shape contexts – Belongie et al.
Consistent Mesh Partitioning and Skeletonisation using SDFs – Shapira et al.

Presented by:
Mukul Sati
2/22/2016

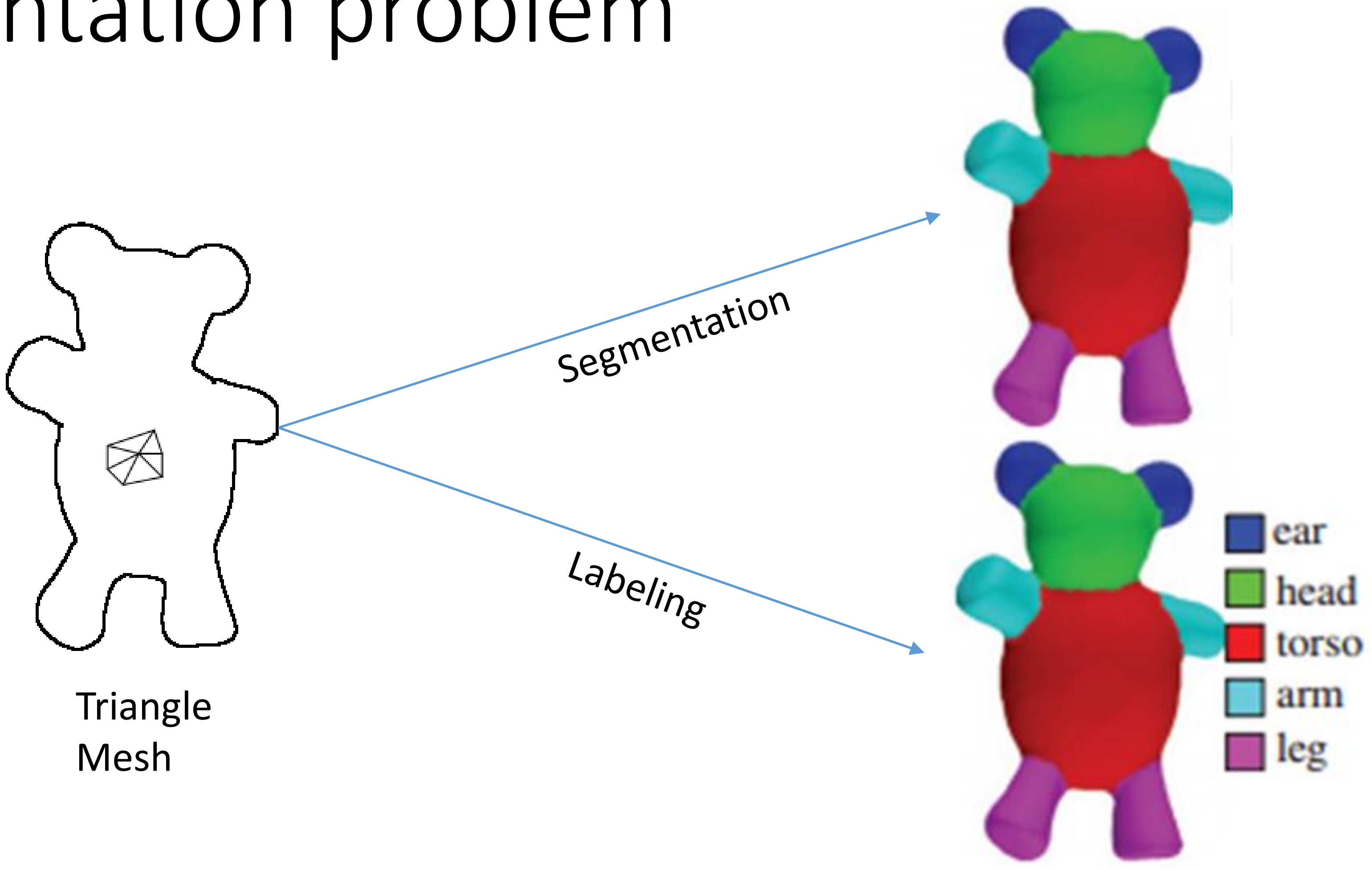
The 3D Mesh Labeling Problem



Triangle Mesh

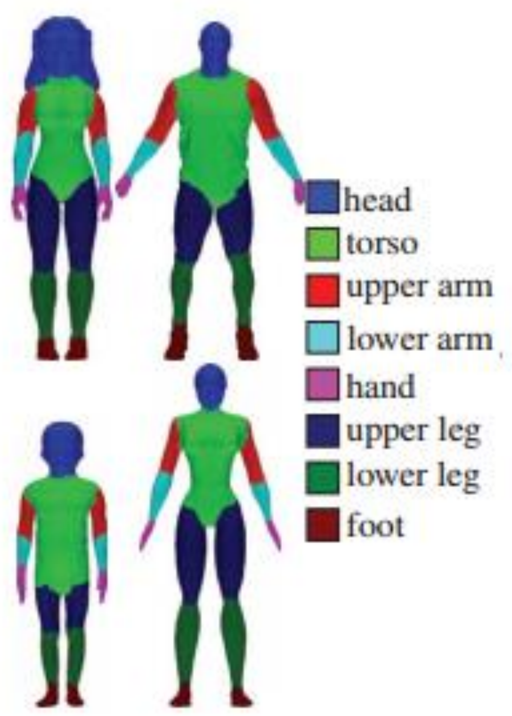


Slightly different from the 3D Mesh segmentation problem

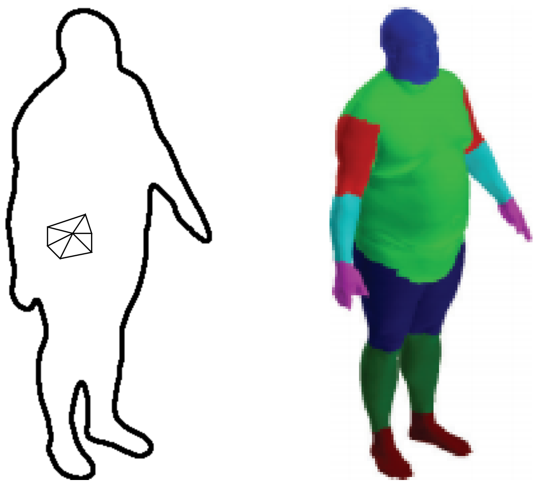


Labeling – Segmentation + Recognize as instances of **known** part types

Requires set of labeled “learning” examples

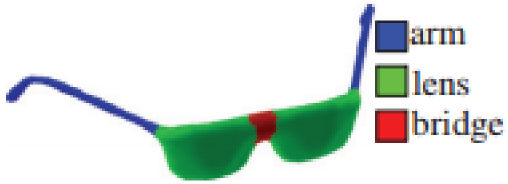


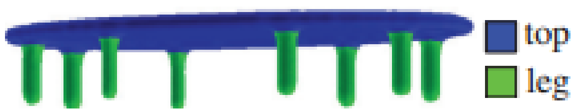
Training meshes

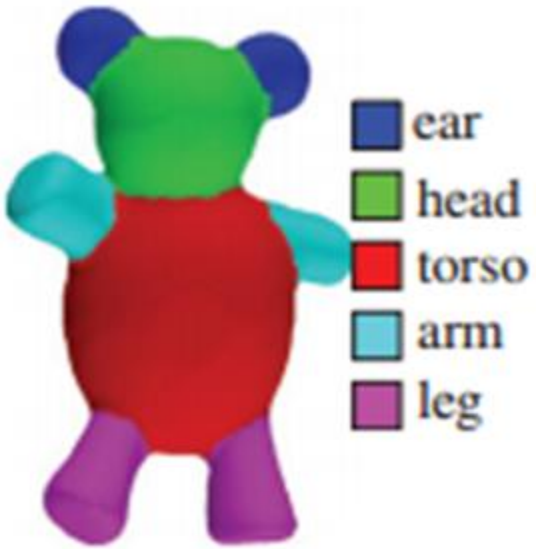


Novel Input Mesh

Labels are object category specific

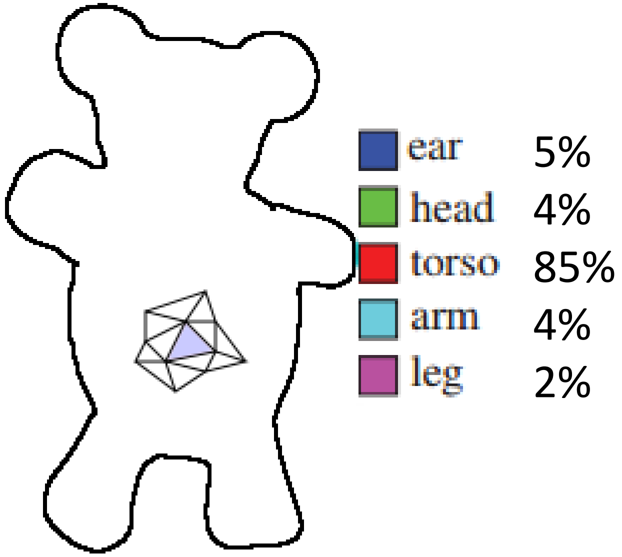






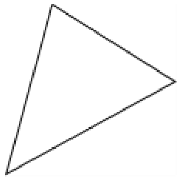
Possible approach - Learn a way of labeling each triangle

- From the training data, find a way to estimate p(label | triangle):



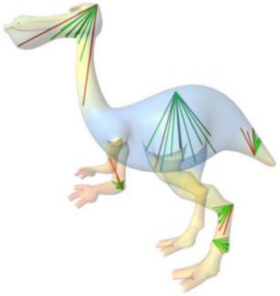
Learning triangle labeling

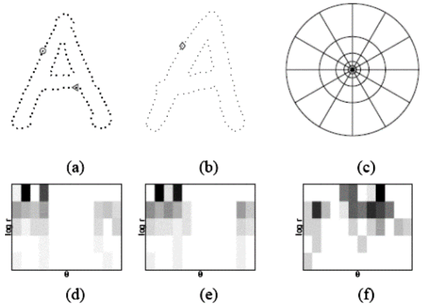
- Describe triangle by extracting relevant features.



$\rightarrow \quad R^n \{x_1, x_2, \ldots x_n\}$

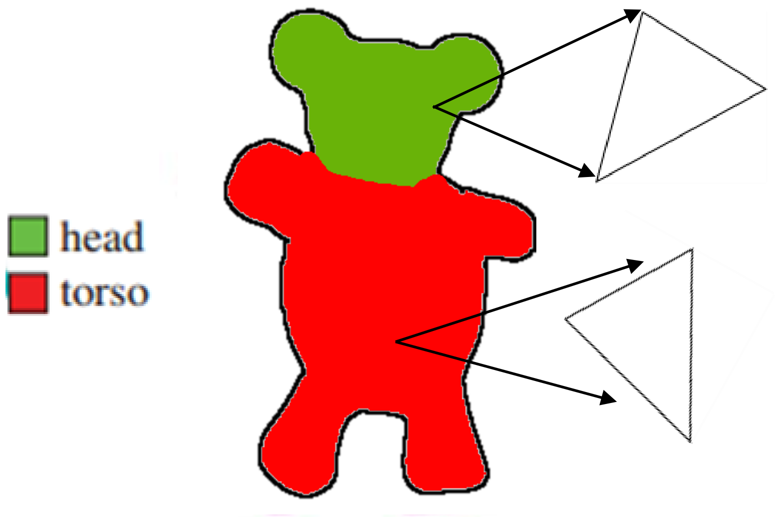
- Extract some number(s) representative of the geometric configuration for each triangle – extract some geometric features.
  - Curvature
  - Shape Diameter
  - Shape Context



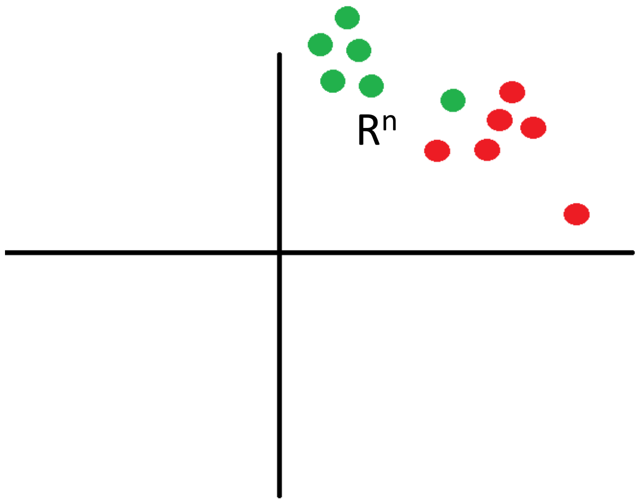


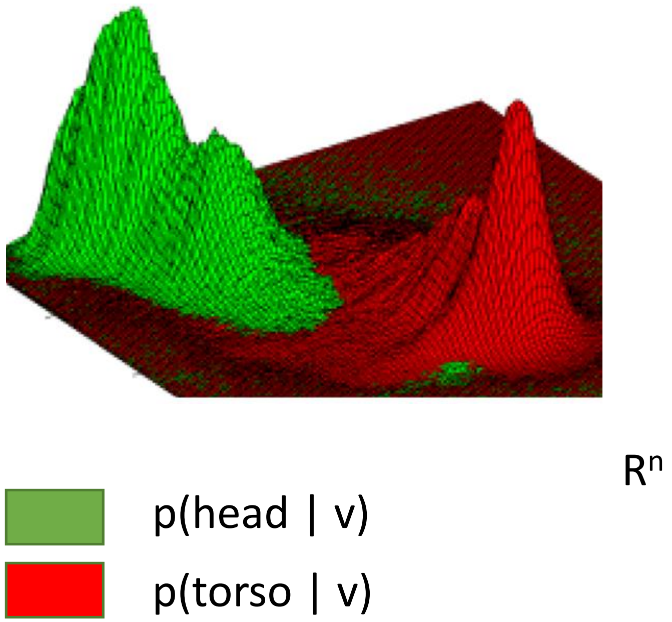
Learning triangle labeling – Training process – assuming one v/s all

- Triangles t is transformed to vector v in R^n.
- p(label | t) => p(label | v).
- Model the probability distribution of each label: p(label | v) = f(ϴ, v).
  (Or use a non-parameteric prob. density estimation technique).
- Learn parameters ϴ from input data for each label.



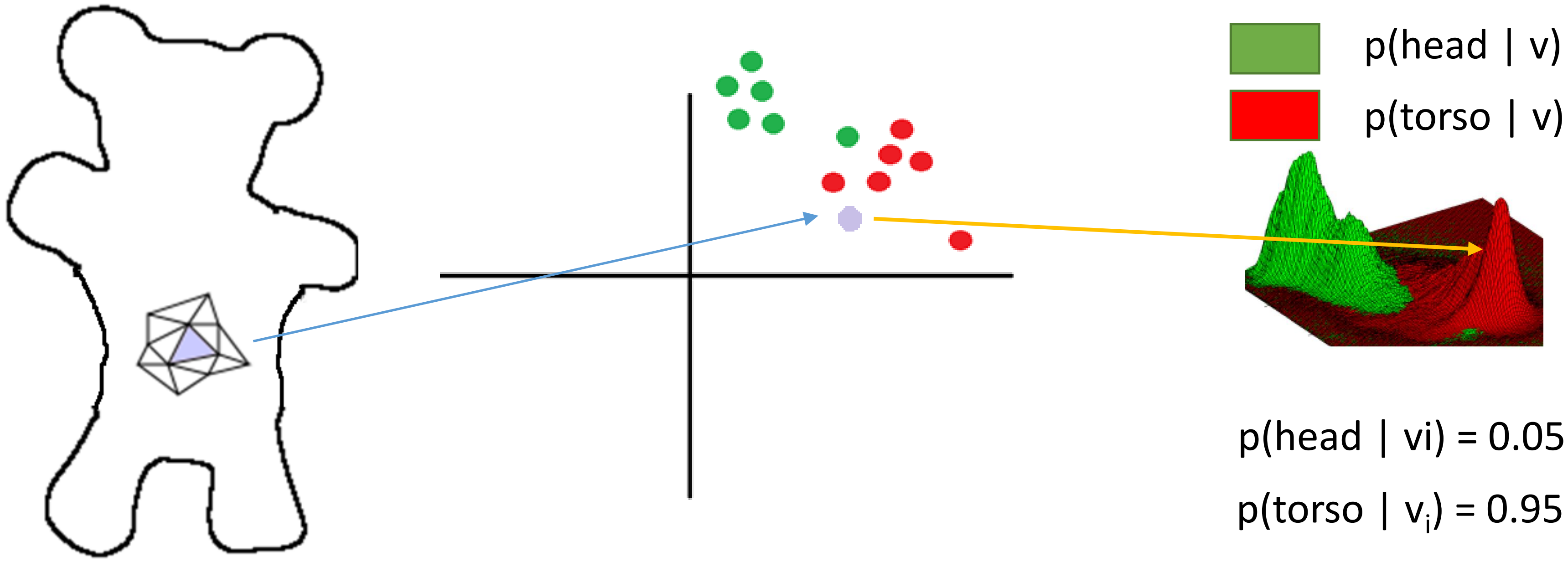
User provided training labeling





Learning triangle labeling – Handling novel data

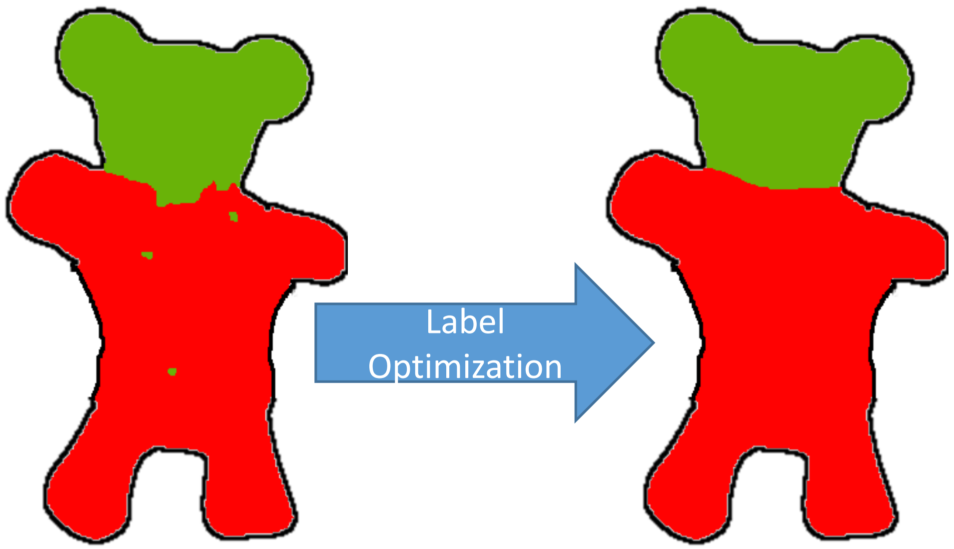
Transform triangle t_i to feature space v_i.
Compute the probability of each label l_k $p(l_k \mid v_i)$ from the learned probability distribution.



Novel Mesh

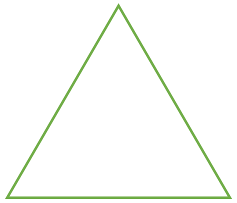
Additional step - Label optimization

- Obtained a labeled novel mesh. However, there may be slight inconsistencies, as decisions were taken for each triangle separately.
- “Smoothen” the labeling by optimizing an appropriate energy function.



Label Optimization

- For each triangle t (equivalently its feature vector v):
  - Term to minimize inconsistency between label k and p(t | k):

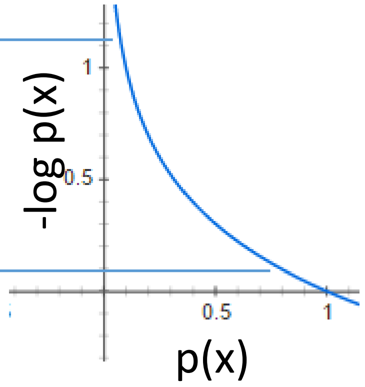


p(head | v) = 0.05

p(torso | v) = 0.95

Penalize if this t is being assigned a head label: –log(p(k)).

Penalty if t was labeled head

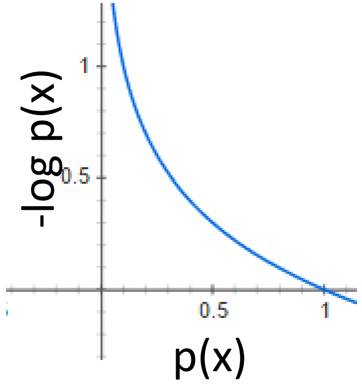


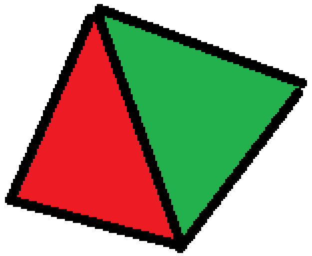
Penalty if t was labeled torso

Label Optimization

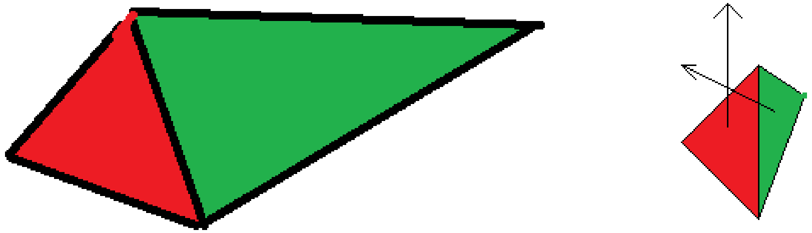
- For each triangle t (equivalently its feature vector v):
  - Term to minimize inconsistency between label k and p(t | k).
  - Term to ensure smoothness amongst neighboring triangles.
    - 0 penalty if the labels are the same.
    - Penalize otherwise based on distance between centroids and dihedral angles.

      -log(k * dist * dihedral angle) (k – tunable constant)





High penalty



Low penalties (High dist * dihedral)

Label Optimization

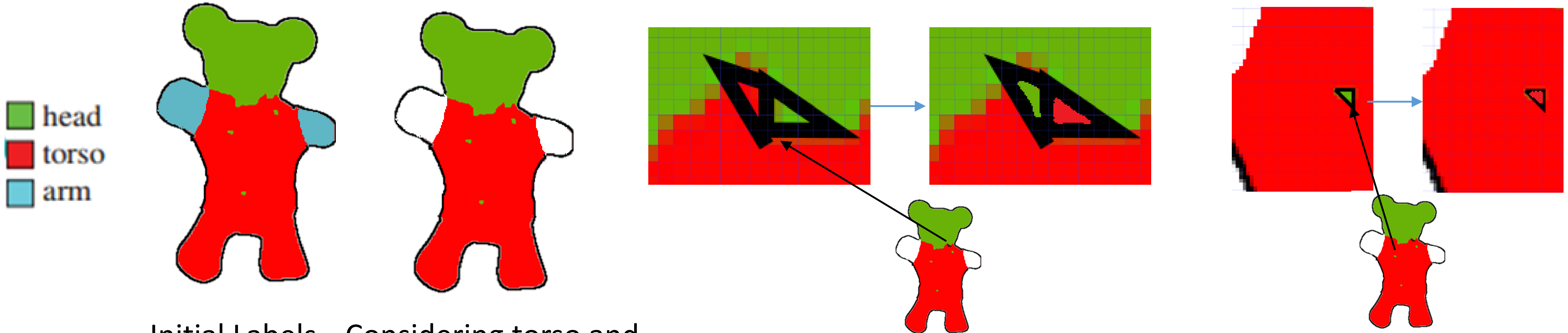
$$\min_{\{l_t, t\in\mathbb{T}\}} \sum_{t\in\mathbb{T}} \xi_U(\mathbf{p}_t, l_t) + \lambda \sum_{t\in\mathbb{T}, v\in\mathbb{N}_t} \xi_S(\mathbf{p}_t, \mathbf{p}_v, l_t, l_v),$$

$$\xi_U(\mathbf{p}_t, l_t) = -\log(\mathbf{p}_t(l_t)),$$

$$\xi_S(\mathbf{p}_t, \mathbf{p}_v, l_t, l_v) = \begin{cases} 0, & \text{if } \ l_t = l_v \\ -log(\theta_{tv}/\pi)\varphi_{tv}, & \text{otherwise} \end{cases}.$$

Label Optimization – (Local) Objective minimization using multi-label graph-cuts

- Fast Approximate Energy Minimization via Graph Cuts – Boykov et al.
- Start with the labels obtained using
- Consider each pair of labels and find the corresponding triangle sets.
- For each triangle set, allow two types of moves for optimization of objective:
    - Label-swap: The labels of two triangles are swapped.
    - Label-expansion: The label of a triangle is switched to the other label.
- The authors frame finding optimal swap and expansion moves as finding a minimium cut in an appropriate graph.



Initial Labels

Considering torso and head triangles

Find and execute optimal swap moves

Find and execute optimal expansion moves

One feature-set to rule them all

- Many different meshes.

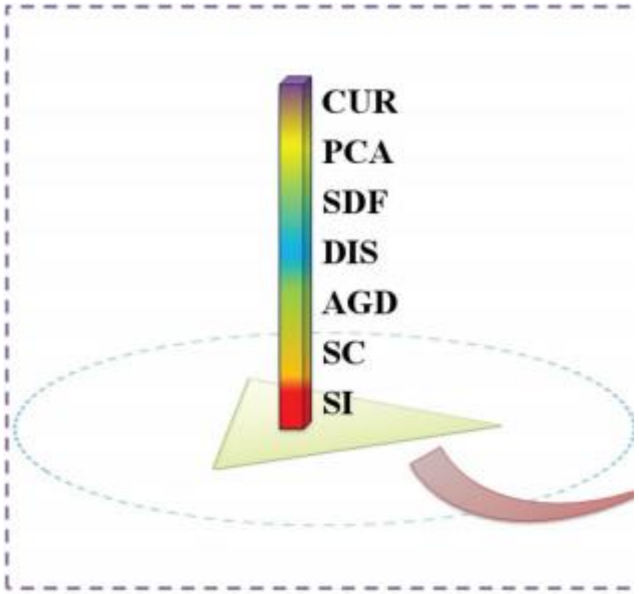
- A particular feature f_i that works for one class of meshes may not work for another.

- Some approaches learn a linear combination of features for each class of mesh: $f:t \rightarrow R^n = a_1 f_1 + a_2 f_2 + \ldots$

- The approach using CNNs learns a non-linear combination of features.

CNNs for 3D Mesh Labeling

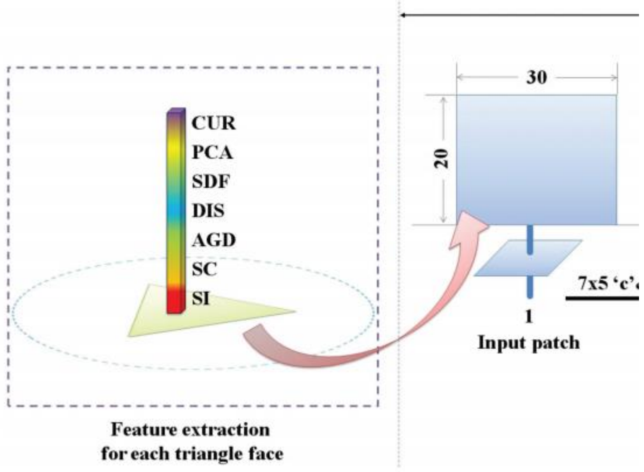
- Extract geometry features for each triangle face.



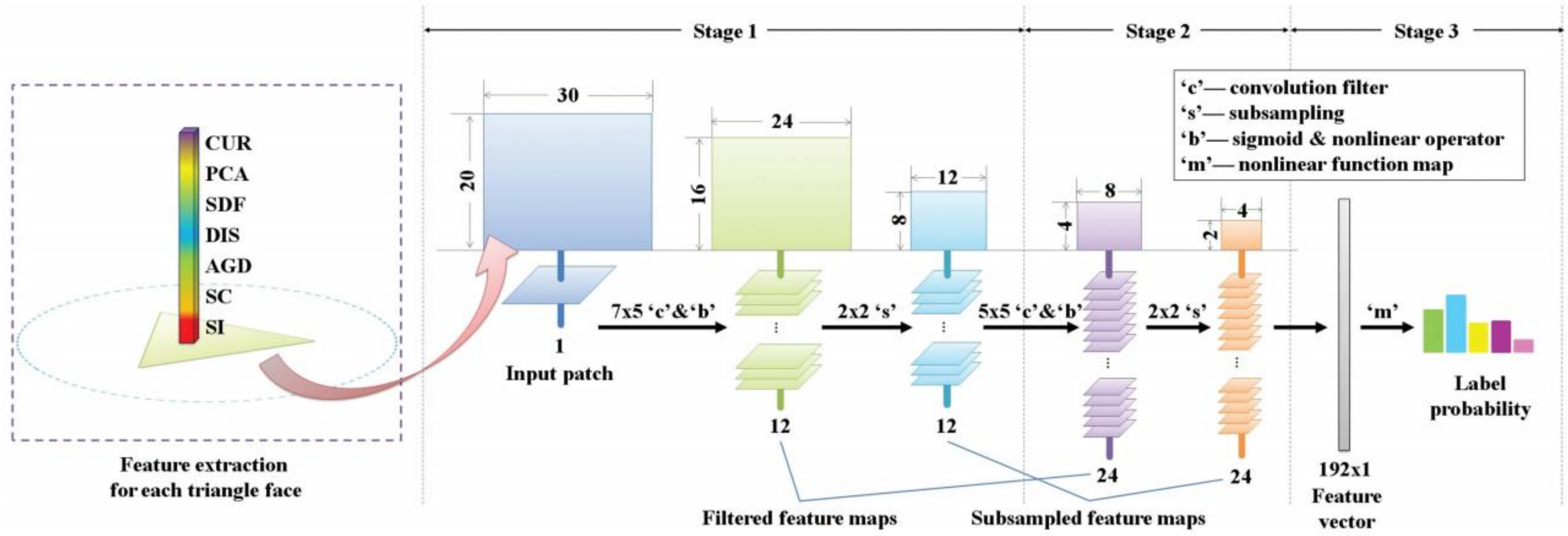
Feature extraction
for each triangle face

CNNs for 3D Mesh Labeling

- Place the features in a 2D grid that will serve as the input to the CNN.
- A key experiment in the work is showing that **any** mapping from the 1D set of features to the 2D grid leads to a performant labeler.

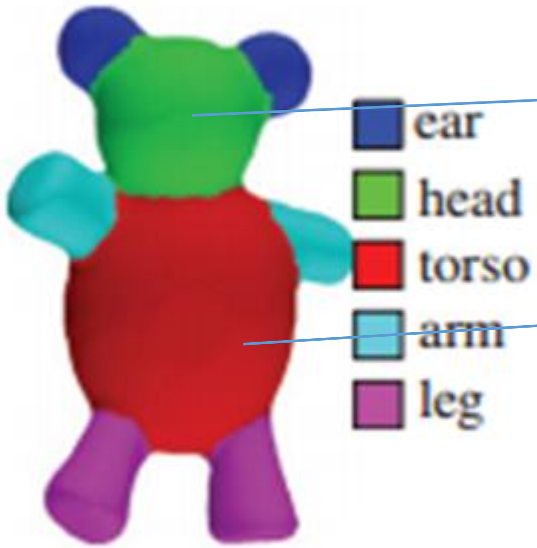


CNN structure



CNN training

- For each training triangle, for an indicator vector for its label.



$g = [0, 1, 0, 0, 0]^T$

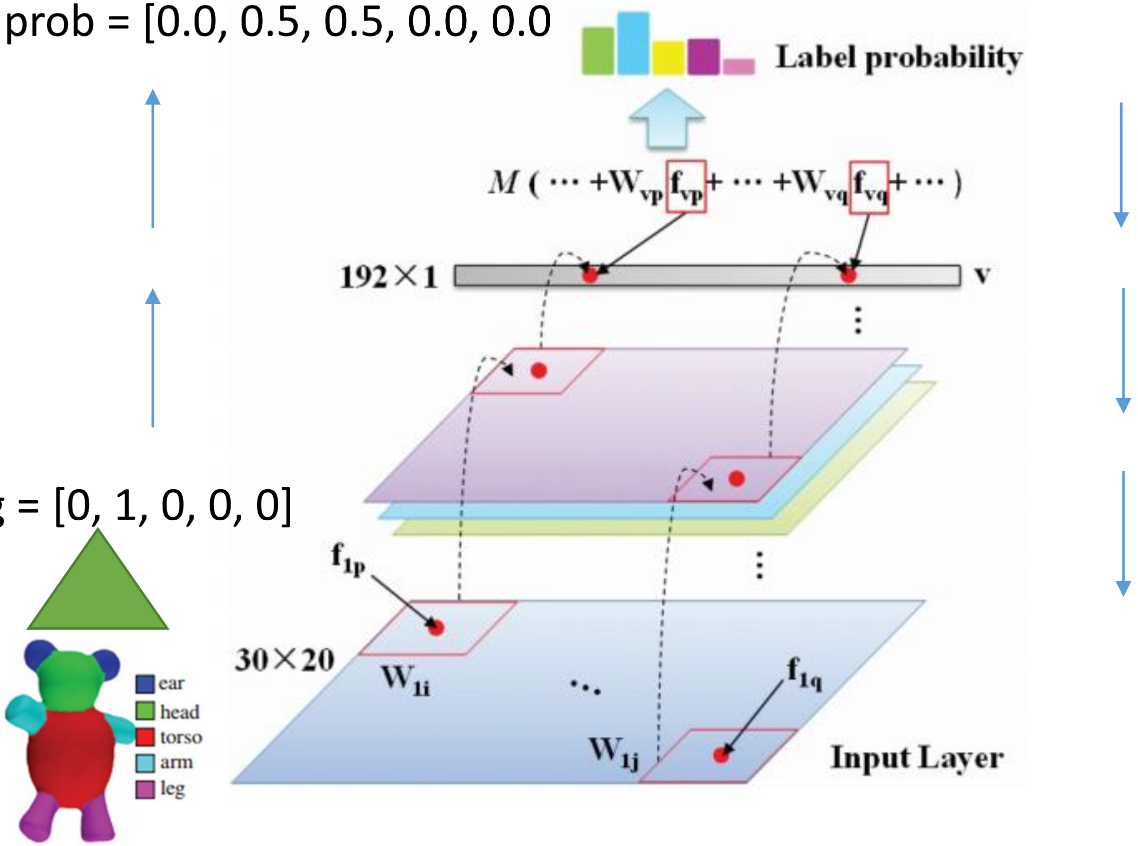
$g = [0, 0, 1, 0, 0]^T$

CNN training

- Drive the squared error between the label probabilities and the indicator matrix to 0

prob = [0.0, 0.5, 0.5, 0.0, 0.0

g = [0, 1, 0, 0, 0]



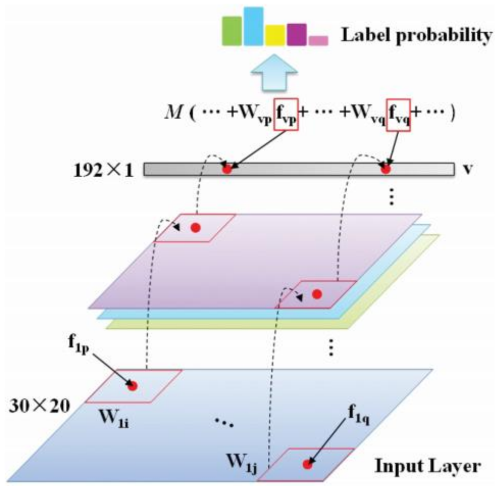
$$\Pi^* = \arg\min_{\Pi} \sum_{t \in T} \left| \mathbf{g}_t - \phi\left(\mathbf{M}\mathbf{v}_t + \mathbf{b}\right) \right|^2,$$

sigmoid

Indicator vector

Visualization of learned features

- After each stage of the CNN, the feature maps (learned weights $W_i = \{w_{i1}, w_{i2}, \ldots w_{in}\}$) can be used to transform each triangle's feature vector into some d-dimensional space.
  - Forward propagate feature vector v for triangle t upto a layer to get input P to the layer $P = \{p_1, p_2, \ldots . p_n\}$
  - For feature map W_i, compute $W_i^T p$.
  - Stack the above to get the transformed vector at each stage (input – X, final labels – p, intermediate – Y, v)



p

v

Y

X

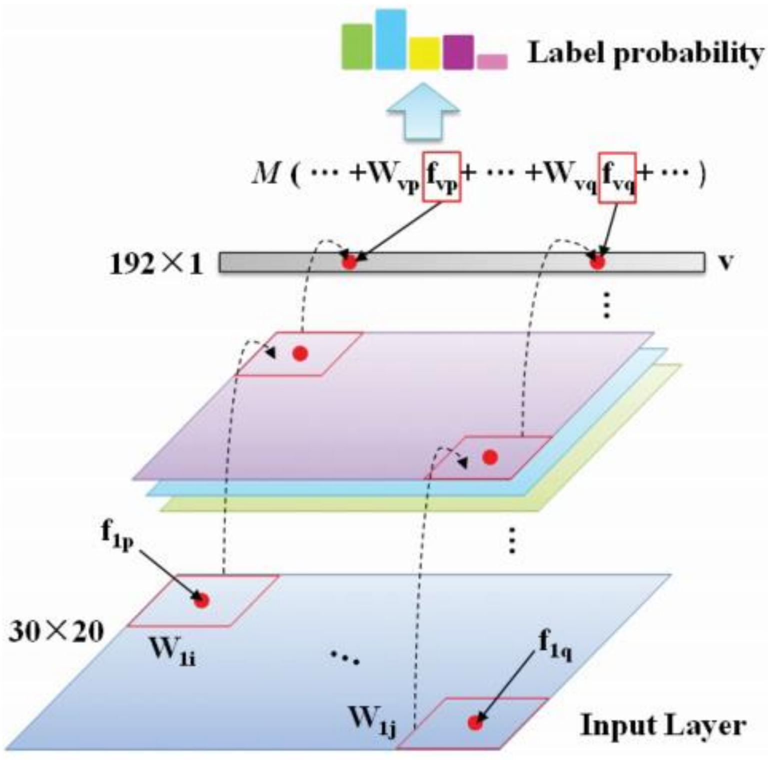
Visualization of learned features

- At each CNN layer, compute a representative feature vector for each label at by averaging all the feature vectors of triangles with the label.
- Compute and visualize the distance between the feature vector for a triangle and the average feature vector for its label.

X
(Geometry
features)

Lower Leg Torso Upper leg Foot Head Lower Arm Upper Arm Hand

Visualization of learned features



p

v

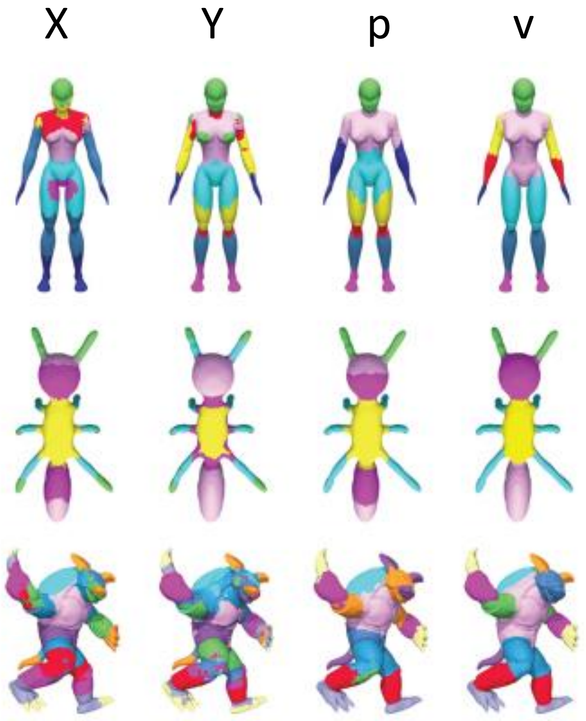
Y

X

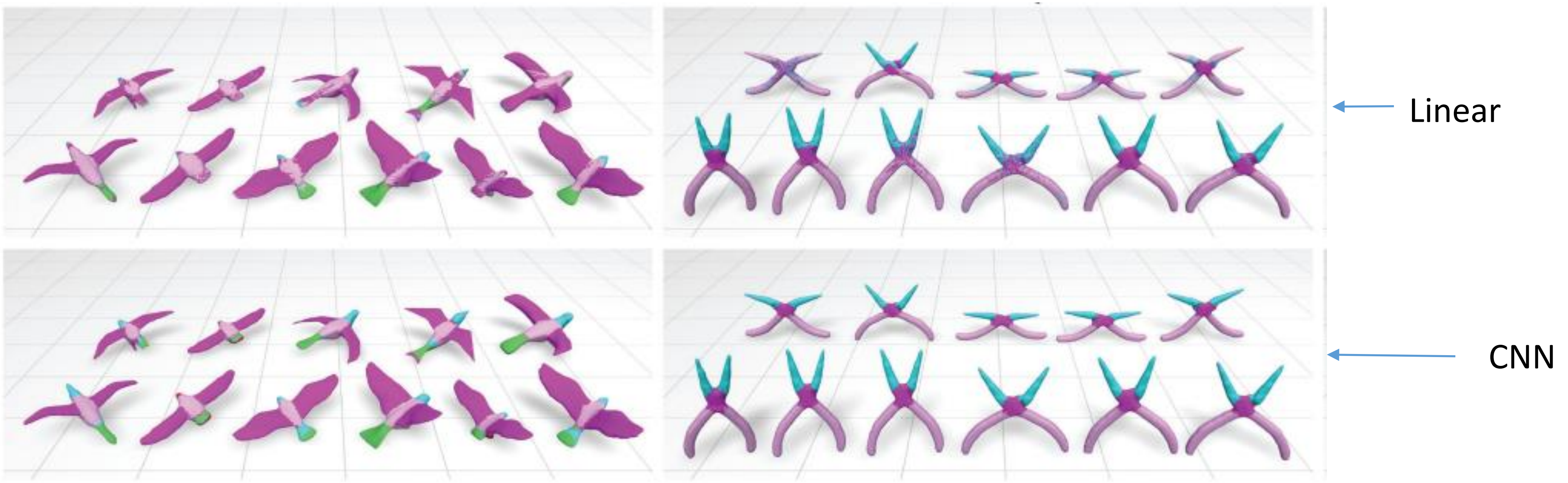
Lower Leg Torso Upper leg Foot Head Lower Arm Upper Arm Hand

Visualization of learned features – Approach 2

- K-Means cluster the vectors and color each triangle based on cluster it belongs to



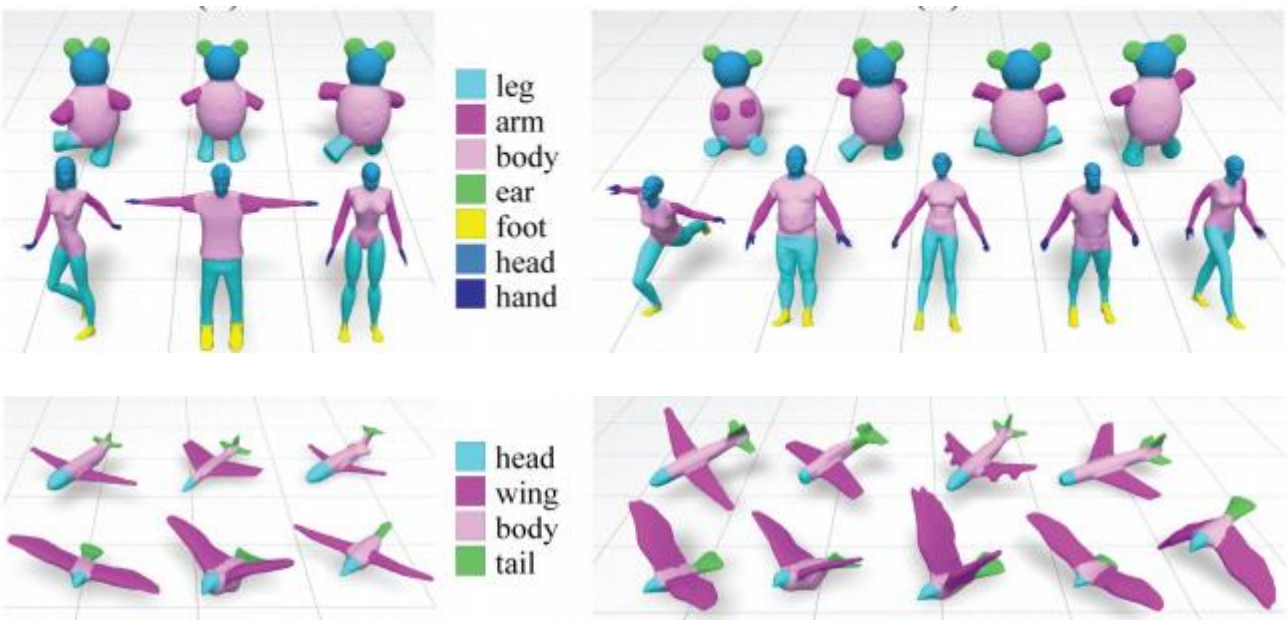
Comparison with learning a linear combination of features



(d) bird (e) plier

Blended categories and transfer of learning

Blended Categories



Training

Testing

Learning Transfers to similar, but novel object classes

Briefly - Learning to Generate Chairs, Tables and Cars with CNNs

Training – Take object O_i, a known camera viewpoint and transformation vector (rotation, translation, zoom, change hue, etc). Create image and segmentation mask for O_i

Runtime - Given object indicator vector ([0,…,1,…0]), novel viewpoint and transformation, generate an image of the object.

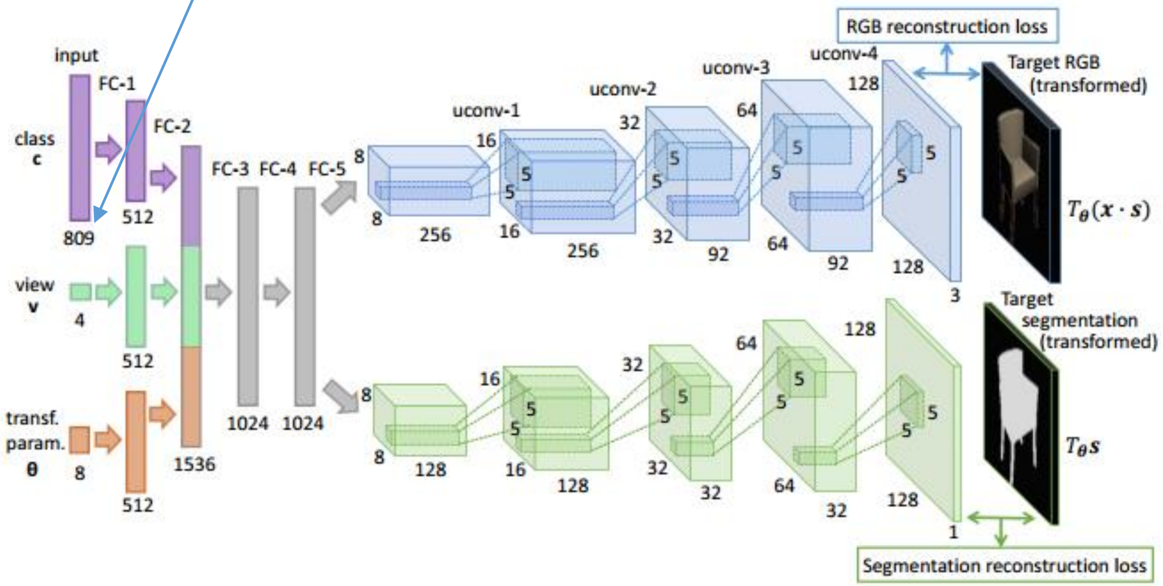
Learning to Generate Chairs, Tables and Cars with CNNs

- Reverse a CNN – start with input as the class, view and transformation.

- Uconv = Unpool + Convolve
    - Unpool – create an s X s block ‘B’ from a 1 X 1 block ‘A’ =>
        - B[0,0] = A[0,0].
        - B[i,j !(i=0&j=0)] = 0

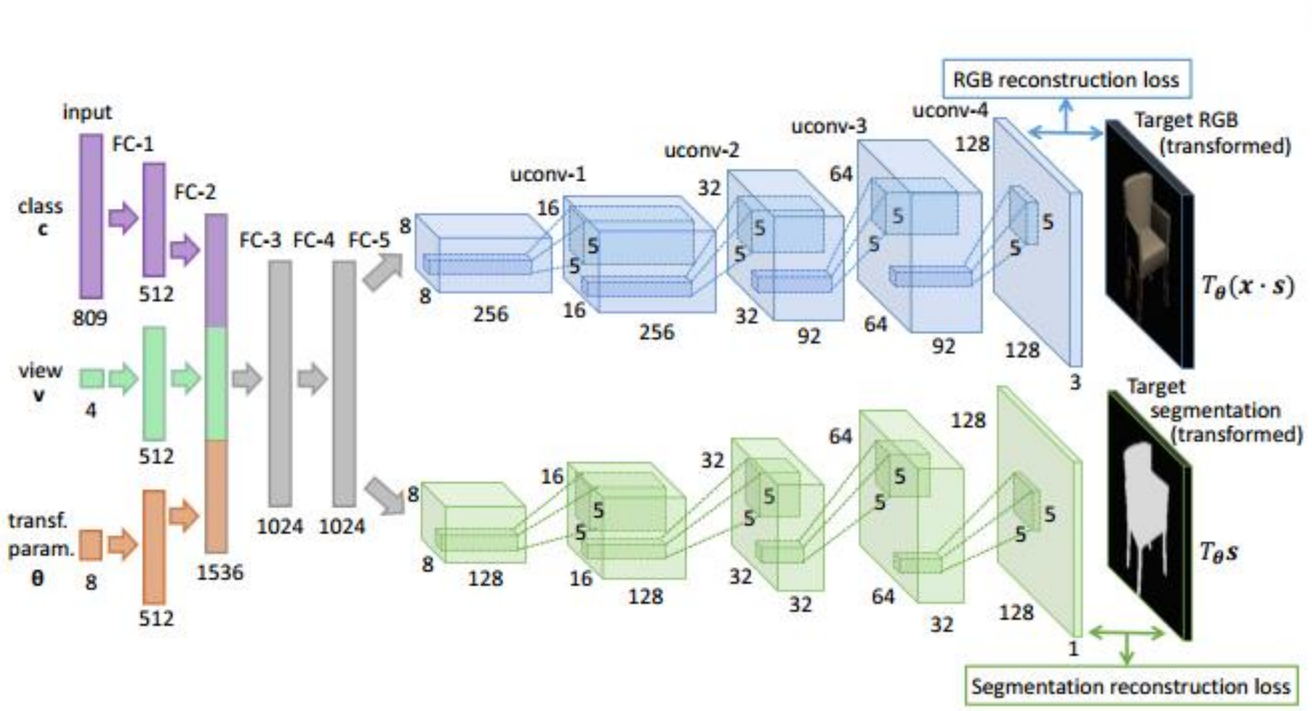
- The segmentation mask is used to render a white background around the generated image.

809 chairs in database



3

Training



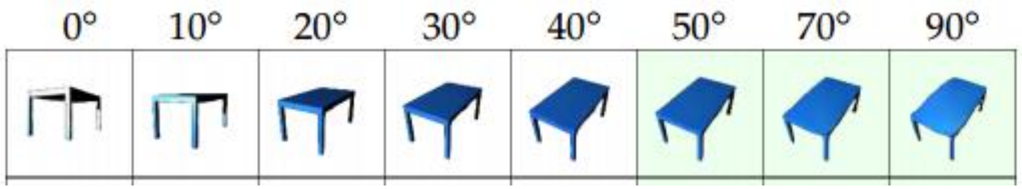
$$\min_{\mathbf{W}} \sum_{i=1}^{N} L_{RGB}\left(T_{\theta^i}(\mathbf{x}^i \cdot \mathbf{s}^i),\, u_{RGB}(h(\mathbf{c}^i, \mathbf{v}^i, \theta^i))\right)$$
$$+\lambda \cdot L_{segm}\left(T_{\theta^i}\mathbf{s}^i,\, u_{segm}(h(\mathbf{c}^i, \mathbf{v}^i, \theta^i))\right),$$

Minimize error between generated images by the network + masks and the images + masks generated using the same parameters by a rendering engine.

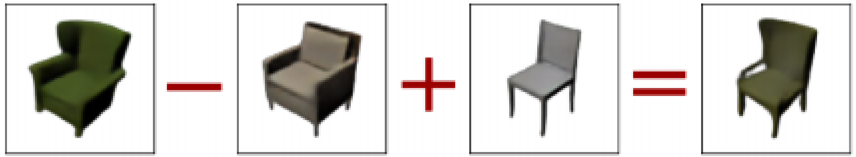
Training

- This learns a mapping f(object, view, transformation) = <image pixels, segmentation pixels>
- It essentially blends between objects in the training set (non-linearly). However, these may be “random” blends – without any information of the “shared” structure between different transforms and views of the same object.
- Hand-wave:
  - In graphics, for spatial transformations, this ties with the notion of aligning all the objects before blending.
  - The shared structure is captured and learned by assuming that there exists a probability distribution that corresponds to the manifold of chair images and by using some dense probability tricks ☺.

Runtime



Extrapolate novel views (green)



Feature Arithmetic (done in feature space and visualized in image space)

Thanks!