Estimating 3D Structure From Images

Advanced Computer Vision

Niels Landwehr

Overview

- Introduction: Computer Vision
- Data, Models, Optimization
- Neural Networks and Automatic Differentiation
- Convolutional Architectures For Image Classification
- Metric Learning for Computer Vision
- Image Segmentation
- Object Detection
- Regularization and Robustness
- Estimating 3D Structure From Images

Problem Settings Discussed So Far

- Problem settings discussed so far interpret a 2D image and return a semantic classification or at most 2D-information (object location, segmentation map)

- **Classification**: cat versus dog

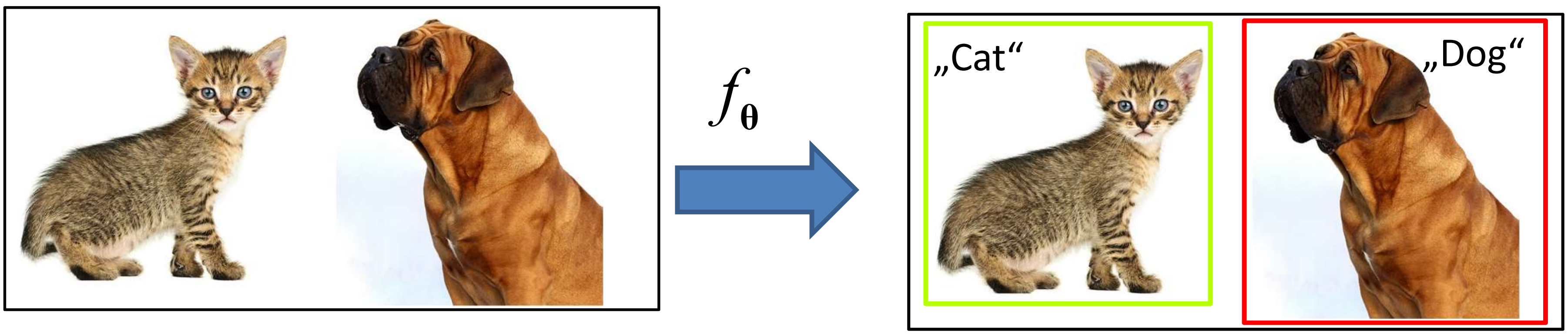
$\xrightarrow{f_{\boldsymbol{\theta}}}$ „Cat“

$\xrightarrow{f_{\boldsymbol{\theta}}}$ „Dog“

Problem Settings Discussed So Far

- Problem settings discussed so far interpret a 2D image and return a semantic classification or at most 2D-information (object location, segmentation map)

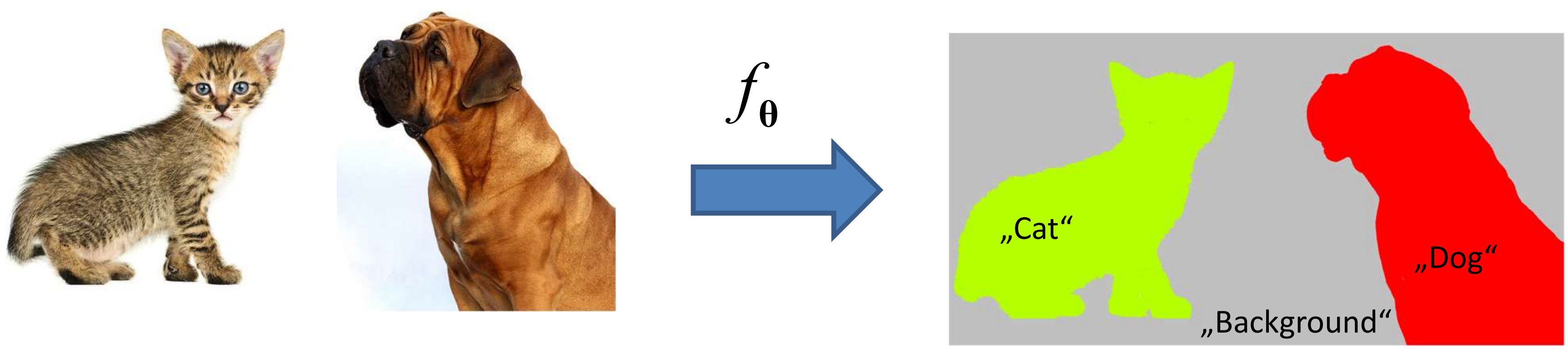
- **Object detection**: cat here, dog there



Problem Settings Discussed So Far

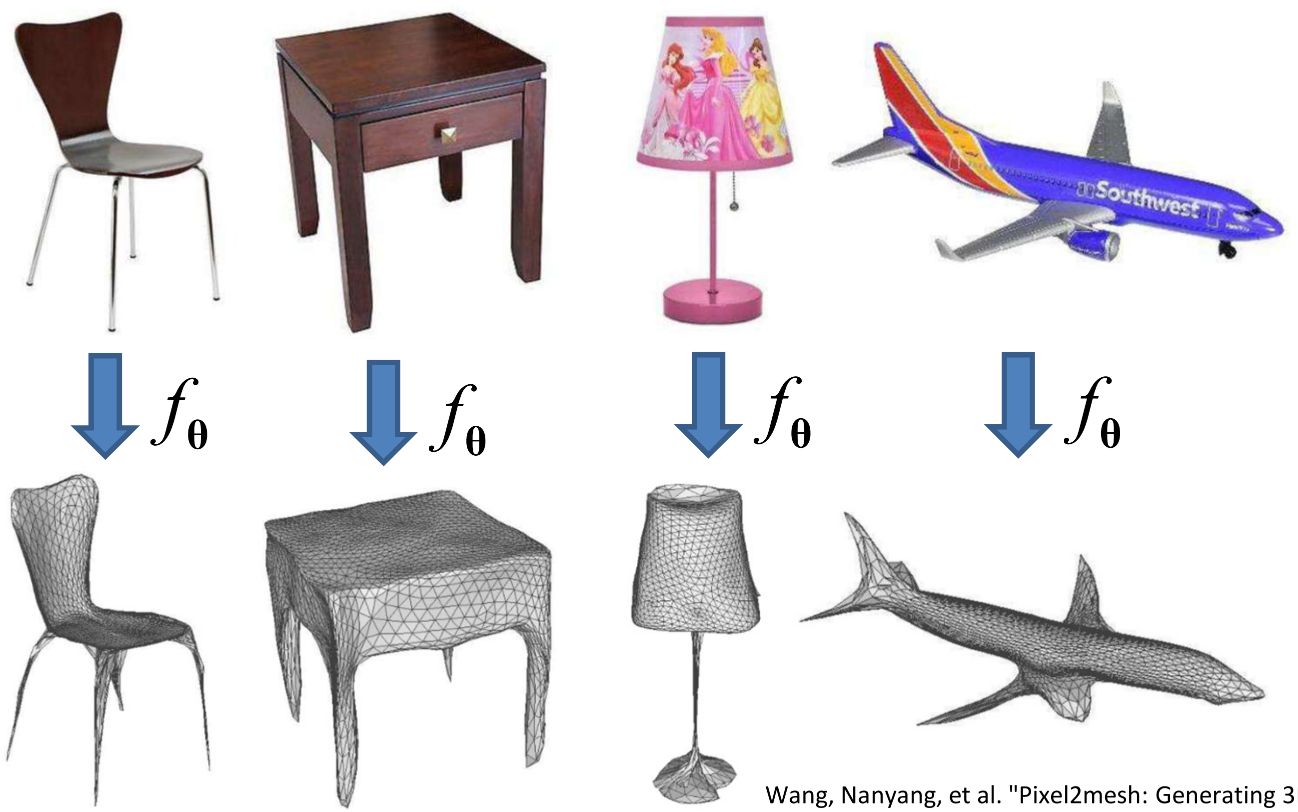
- Problem settings discussed so far interpret a 2D image and return a semantic classification or at most 2D-information (object location, segmentation map)

- **Segmentation**: assign semantic classes to all locations (pixels) in image



Inferring 3D Information From 2D Images

- Our world is 3D: Humans regularly infer 3D-structure from 2D (stereo) vision
- Can we build computer vision systems that solve the same problem?

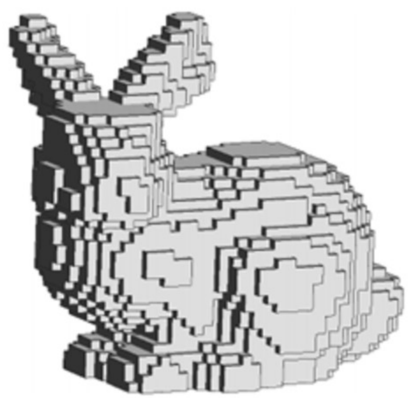


Not an easy problem:
2D => 3D mapping
often ambiguous,
complex outputs, ...

Wang, Nanyang, et al. "Pixel2mesh: Generating 3D mesh models from single RGB images." *Proceedings of the European Conference on Computer Vision,* 2018.

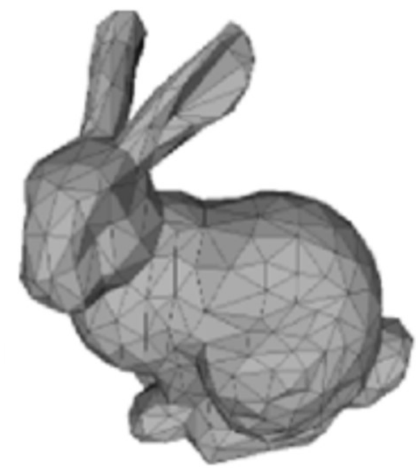
How to Represent 3D Output?

- Unlike for 2D images, there is no fully canonical way of representing 3D objects
  - Voxel grids: partition space into 3D cells, mark those occupied by object
  - Point cloud: represent surface of object by a set of points in 3D
  - Mesh-based: represent surface of object by triangulated mesh



voxels

point cloud



mesh

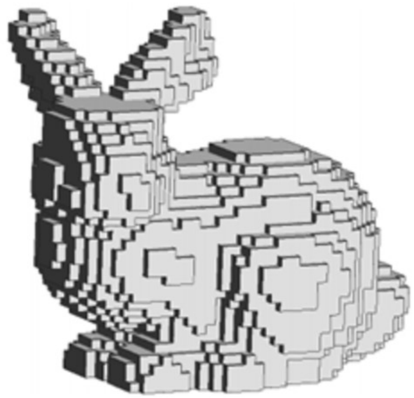
Hoang, Long, et al. "A Deep Learning Method for 3D Object Classification Using the Wave Kernel Signature and A Center Point of the 3D-Triangle Mesh." *Electronics* 8.10 (2019): 1196.

Voxel Grids

- Voxel grids: 3D-voxels as straightforward generalization of 2D-pixels
- So-called volumetric shape representation: voxels on a regular grid are marked as occupied by object or not

$$\mathbf{y} \in \{0,1\}^{D \times D \times D}$$

D is spatial resolution of output



- Main problem is that size of representation grows cubically with resolution: ok for low-resolution outputs, but does not scale well to higher resolutions
- Essentially, using 3D voxels to store a 2D object surface is somewhat wasteful
- Moreover, axis-aligned voxels are not ideal for representing general (e.g., diagonal) shape elements: smooth representation requires high resolution

Point Clouds

- Point cloud: represent a shape using a fixed number N of points in space, given by their 3D coordinate

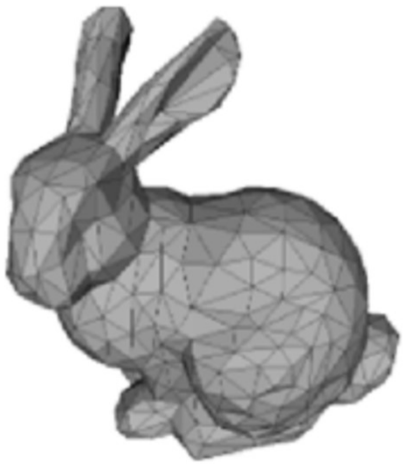
$$\mathbf{y} \in \mathbb{R}^{N \times 3}$$

N is the number of points

- This is only an implicit representation of the surface and volume of the shape
- Has to be transformed to other representation (voxels, meshes) for further tasks that require an explicit representation of the shape
- More efficient than voxel representation in terms of space requirements

Triangle Meshes

- **Triangle mesh: set of triangles in 3D that are connected by common edges and corners**



Can be represented as a graph structure, with 3D-coordinates attached to the vertices:

$$\mathbf{y} = (V, E, \mathbf{C})$$

$V = \{v_1, \ldots, v_N\}$ set of vertices,

$E \subseteq V \times V$ set of edges,

$\mathbf{C} = \{\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(N)}\}$ node coordinates with $\mathbf{c}^{(n)} \in \mathbb{R}^3$

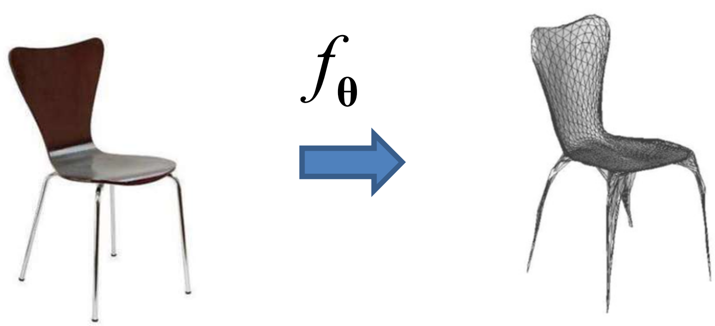
- Flexible, efficient representation for shapes through surface
- Explicit representation with clear mathematical semantics
- Widely used in e.g. computer graphics and simulation

Problem Setting: Estimate 3D-Mesh From Image

- **Problem setting: learning to estimate a 3D-mesh from a single image**

Given:
- Training inputs in the form of images $\{\mathbf{x}_1,\ldots,\mathbf{x}_n\},\ \mathbf{x}_i \in \mathbb{R}^{m\times l\times d}$, where on each image a single object is visible
- Training labels in the form of triangle meshes $\{\mathbf{y}_1,\ldots,\mathbf{y}_n\}$, where $\mathbf{y}_i = (V_i, E_i, \mathbf{C}_i) \in \mathcal{Y}$ with $\mathbf{C}_i = \{\mathbf{c}_i^{(1)},\ldots,\mathbf{c}_i^{(N_i)}\},\ \mathbf{c}_i^{(n)} \in \mathbb{R}^3$ is a 3D-mesh representation of the visible object

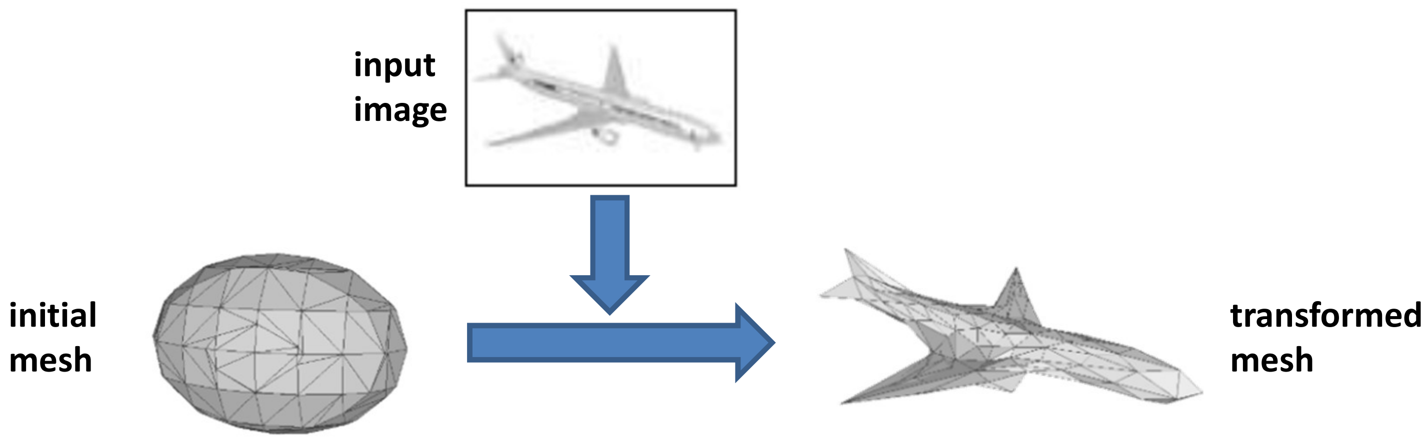
Find:
- A model $f_{\boldsymbol{\theta}} : \mathbb{R}^{m\times l\times d} \rightarrow \mathcal{Y}$ that maps the image of an object onto its 3D-mesh representation (where $\mathcal{Y}$ is the space of all 3D-triangle meshes)



Pixel2Mesh Model [Wang et al., 2018]

- We specifically discuss the **Pixel2Mesh** [Wang et al., 2018] model: direct estimation of 3D-meshes from single image using end-to-end trainable network
- This network needs to output a 3D-mesh, which is a graph structure together with 3D-coordinates as node labels: how do we encode this in the output?

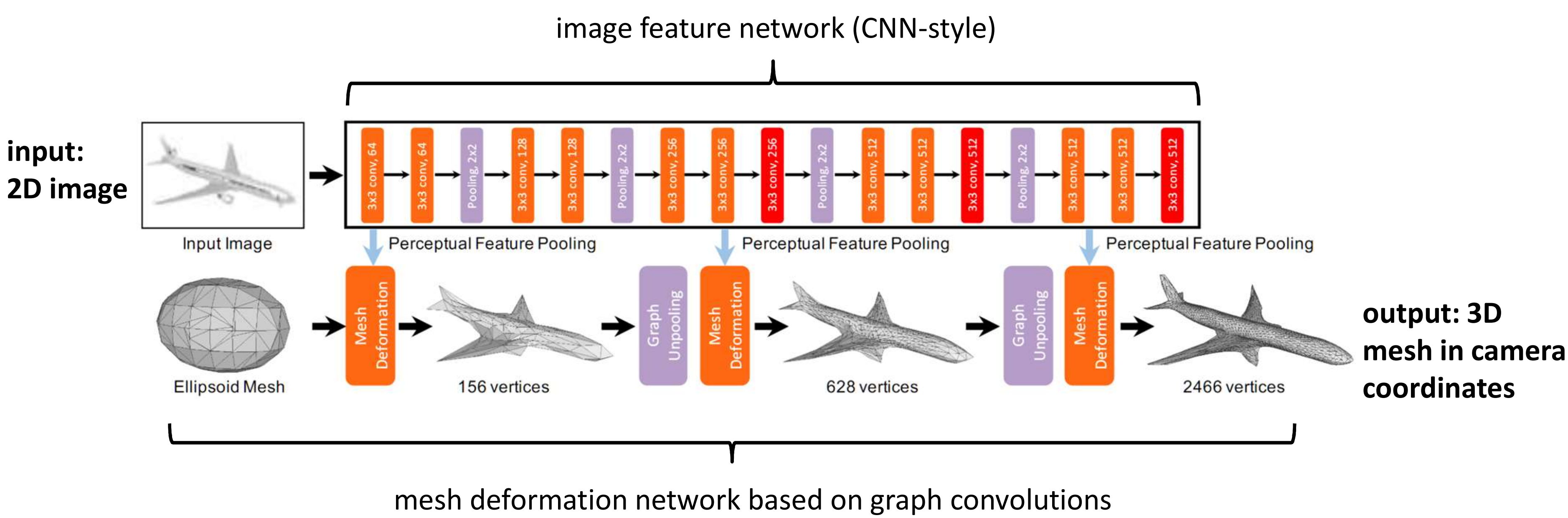
- Idea: network learns to deform an ellipsoidal standard mesh into the final mesh, based on the visual information in the input image



Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Pixel2Mesh Model: Overview

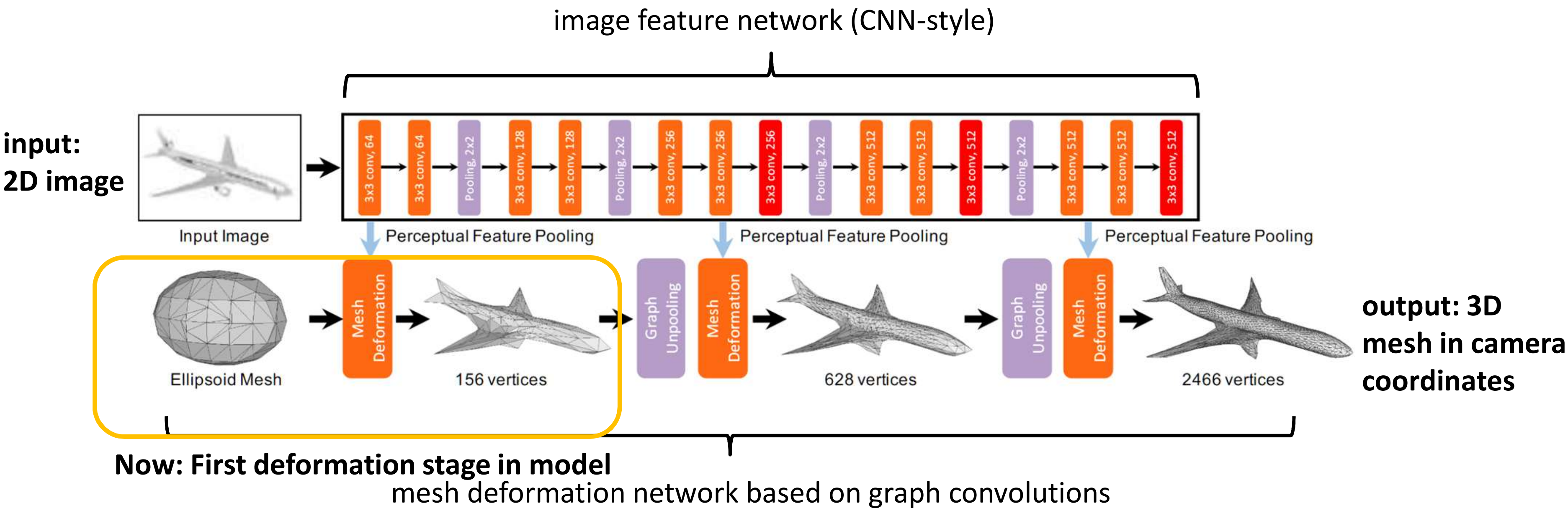
- Pixel2Mesh model architecture consists of two parts:
  - Image feature network: extract visual features from 2D-image
  - Mesh deformation network: deform an initial ellipsoidal mesh to target mesh, increase resolution of mesh during deformation



Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Pixel2Mesh Model: Overview

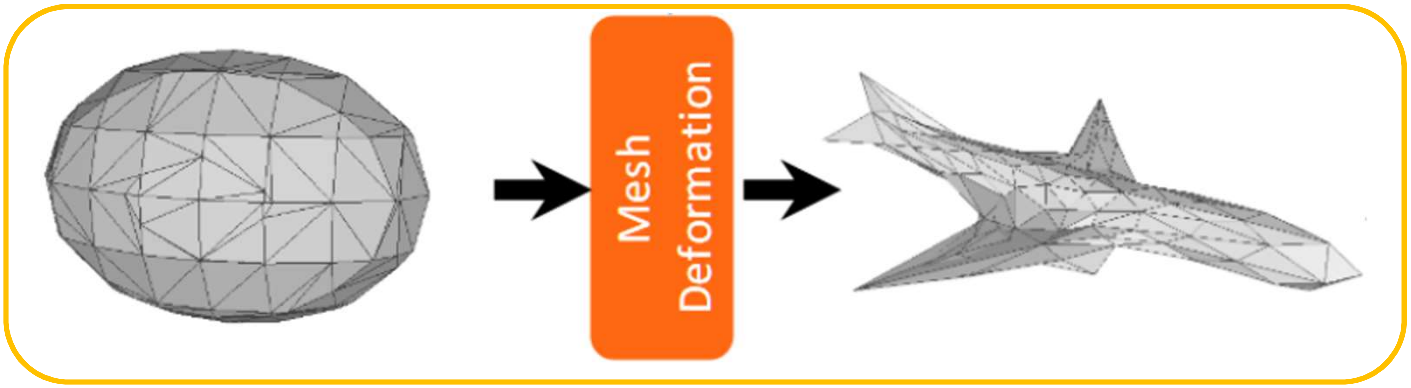
- Pixel2Mesh model architecture consists of two parts:
  - Image feature network: extract visual features from 2D-image
  - Mesh deformation network: deform an initial ellipsoidal mesh to target mesh, increase resolution of mesh during deformation



Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Mesh Deformation as Coordinate Reestimation

- Deformation means estimating new 3D-coordinates for all vertices in the graph representing the ellipsoidal initial mesh
- The graph structure of the mesh can stay the same
- Of course, deformation must be guided by 2D image content and jointly re-estimate coordinates of all vertices (vertices cannot be treated independently)



Ellipsoid: $\mathbf{y} = (V_1, E_1, \mathbf{C}_1)$

Set of vertices V_1

Set of edges $E_1 \subseteq V_1 \times V_1$

Initial vertex coordinates $\mathbf{C}_1$, with $\mathbf{c} \in \mathbb{R}^3$ for $\mathbf{c} \in \mathbf{C}_1$

Airplane: $\mathbf{y} = (V_1, E_1, \mathbf{C}_2)$

Set of vertices V_1

Set of edges $E_1 \subseteq V_1 \times V_1$

New vertex coordinates $\mathbf{C}_2$, with $\mathbf{c} \in \mathbb{R}^3$ for $\mathbf{c} \in \mathbf{C}_2$

Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Interlude: Graph Neural Networks

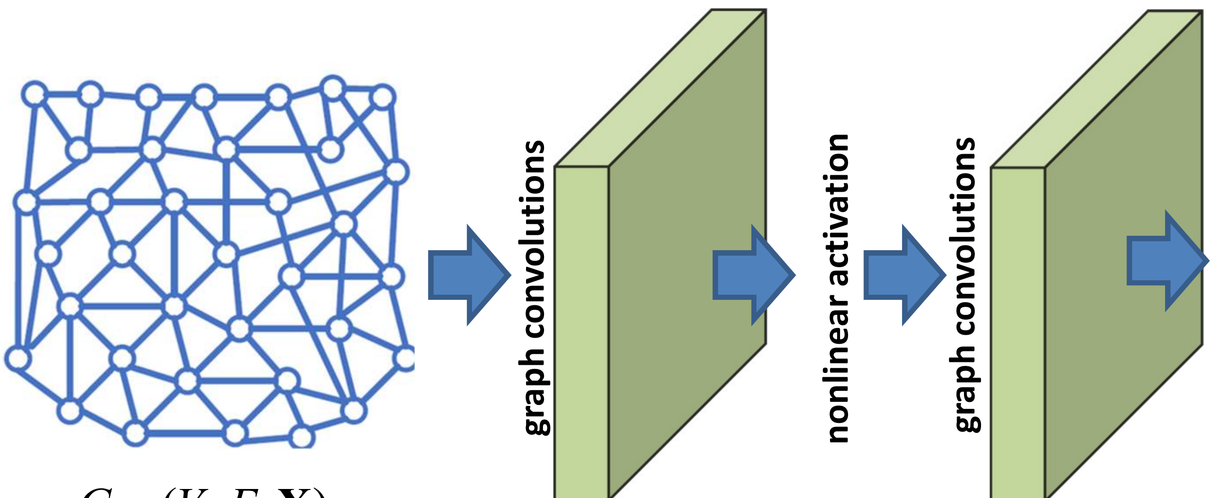
- **Graph neural networks** : extend ideas from convolutional neural networks to graph-format data

Input: graph $G = (V, E, \mathbf{X})$, consisting of nodes $V = \{v_1, \ldots, v_N\}$, edges $E \subseteq V \times V$, node features $\mathbf{X} = \{\mathbf{x}_{v_1}, \ldots, \mathbf{x}_{v_N}\}$, $\mathbf{x}_{v_n} \in \mathbb{R}^M$

Output: node embeddings $\mathbf{e}_{v_1}, \ldots, \mathbf{e}_{v_N}$, $\mathbf{e}_{v_n} \in \mathbb{R}^D$

Node embeddings should be computed jointly based on graph structure.

Embedding can be optimized for different criteria, e.g. to serve as features for node classification

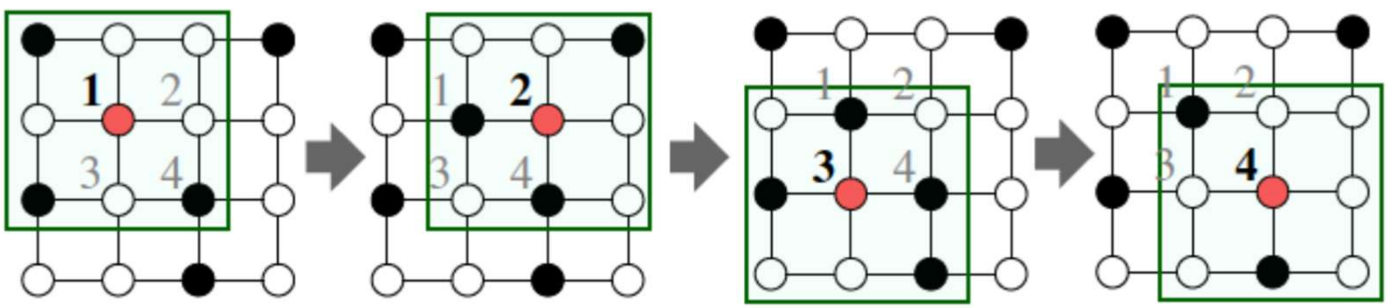


$G = (V, E, \mathbf{X})$

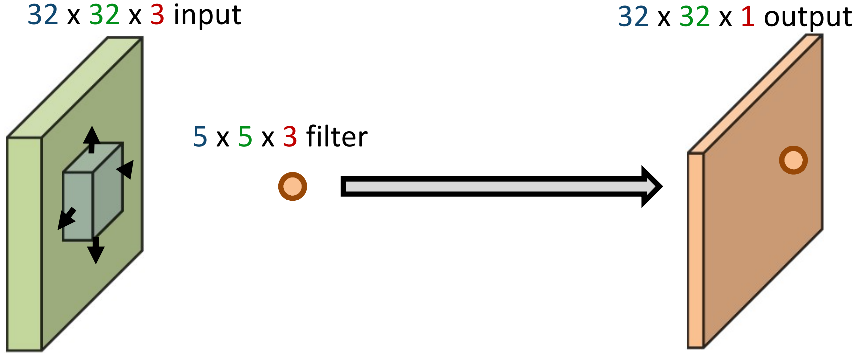
node embeddings $\mathbf{e}_{v_1}, \ldots, \mathbf{e}_{v_N}$

From Normal Convolution to Graph Convolution

- **Convolution in convolutional neural networks for image data**:
  - local operation in space on a regular grid of pixels or (in later layers) spatially ordered activations of learned features
  - combine information from neighboring spatial locations into activation for current spatial location



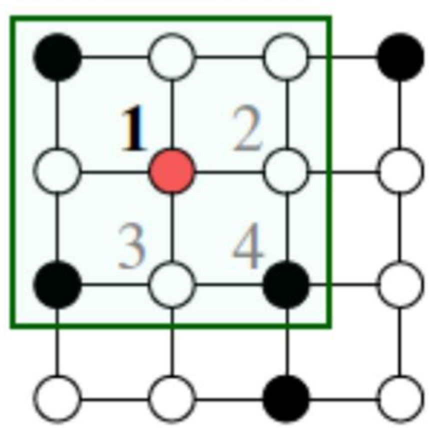
J. Leskovec, CS224W, Graph Neural Networks



From Normal Convolution to Graph Convolution

- **Generalizing convolutions to graphs**
  - spatial arrangement of pixels or features can be seen as graph grid
  - in general graphs, should combine information from local graph neighborhood rather than neighboring spatial locations

standard convolution



graph convolution

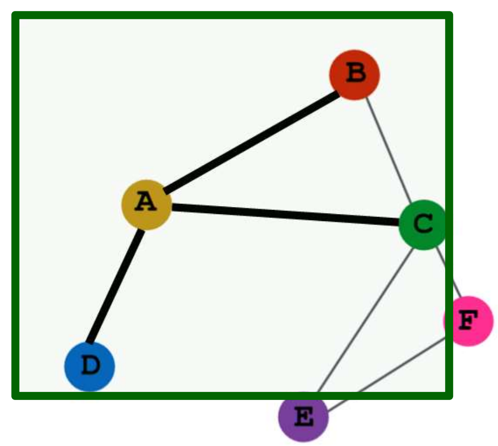


Figure: J. Leskovec, CS224W, Graph Neural Networks

- Question is how to combine and aggregate information from neighborhood

A Neural Network Model for Node Embeddings

- **Graph neural network: compute layer-wise node embedding**
  - At each layer $l \in \{0, \ldots, L\}$, we compute a node embedding $\mathbf{h}_v^{(l)}$ (real vector) for each node $v \in V$
  - In the first layer ($l = 0$), the node embedding is simply the node feature vector $\mathbf{x}_v$
  - The final layer outputs the final node embeddings $\mathbf{e}_v = \mathbf{x}_v^L$

$$\mathbf{h}_v^{(0)} = \mathbf{x}_v$$

Lowest layer (input): node features

Trainable weight matrix

Trainable weight matrix

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W}_{l+1} \underbrace{\sum_{u \in N(v)} \frac{\mathbf{h}_u^{(l)}}{|N(v)|}}_{\text{Average of neighbor embeddings in previous layer}} + \mathbf{B}_{l+1} \mathbf{h}_v^{(l)} \right)$$

Embedding of node itself in previous layer

Nonlinear activation like ReLU

A Neural Network Model for Node Embeddings

- **Graph neural network: compute layer-wise node embedding**
  - At each layer $l \in \{0, \ldots, L\}$, we compute a node embedding $\mathbf{h}_v^{(l)}$ (real vector) for each node $v \in V$
  - In the first layer ($l = 0$), the node embedding is simply the node feature vector $\mathbf{x}_v$
  - The final layer outputs the final node embeddings $\mathbf{e}_v = \mathbf{x}_v^L$

$$\mathbf{h}_v^{(0)} = \mathbf{x}_v$$

Lowest layer (input): node features

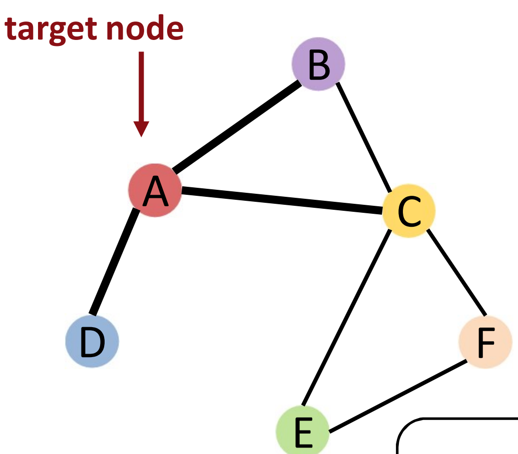
$$\mathbf{h}_v^{(l+1)} = \sigma\left(\underbrace{\mathbf{W}_{l+1} \sum_{u \in N(v)} \frac{\mathbf{h}_u^{(l)}}{|N(v)|}}_{\text{contribution of embeddings of nodes in neighborhood of node } v} + \underbrace{\mathbf{B}_{l+1} \mathbf{h}_v^{(l)}}_{\text{contribution of embedding of node } v \text{ in previous layer}} \right)$$

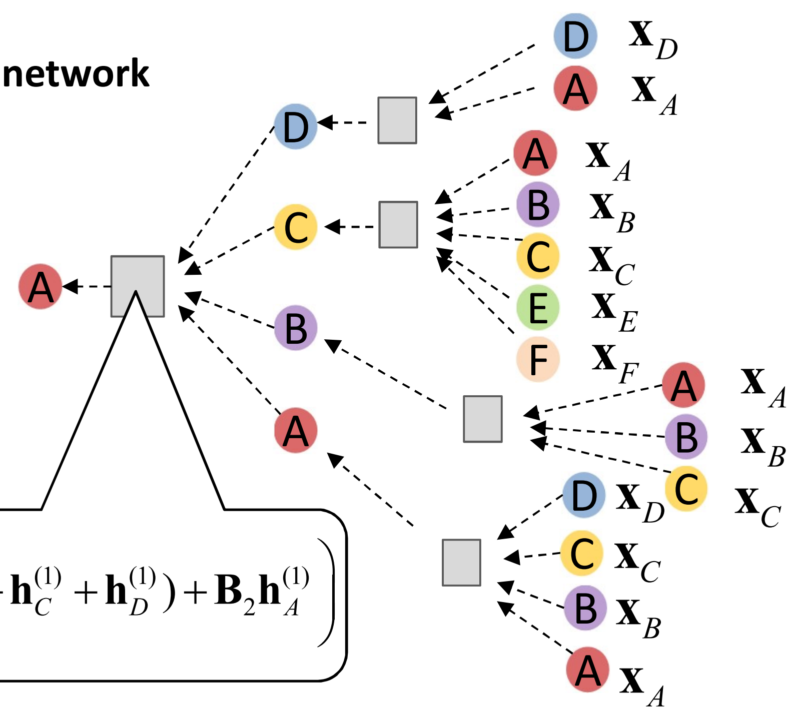
If we only had this part, would be a normal (fully connected) neural network that operates independently on each node

Compute Graph For Target Node

- Graph neural network, from perspective of a single node:
  - At Layer 0, embedding is identical to the node features
  - At Layer 1, embedding contains information from node features and features of direct neighbors
  - At Layer 2, embedding contains information from nodes with distance ≤2

Example: 2-layer network





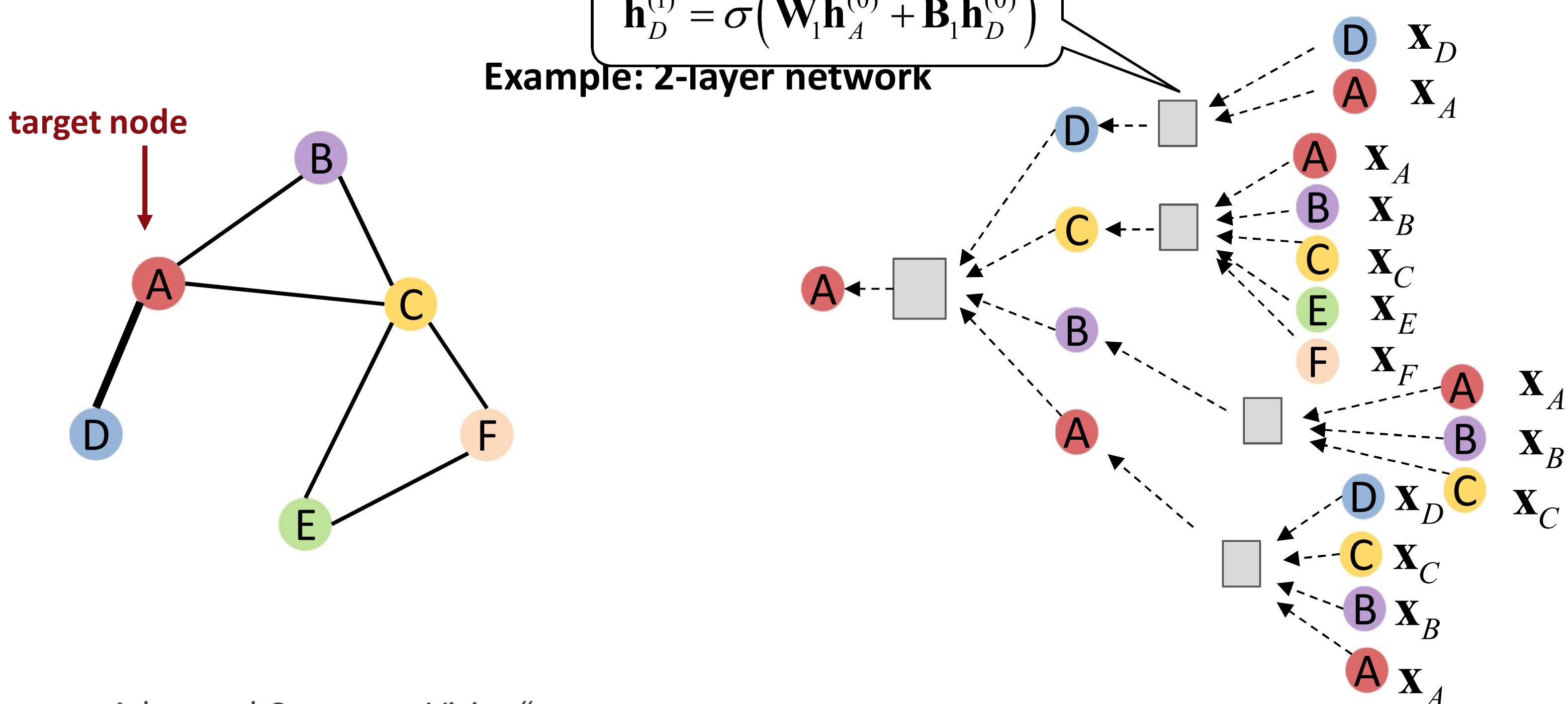
$$\mathbf{h}_A^{(2)} = \sigma\left(\mathbf{W}_2 \frac{1}{3}(\mathbf{h}_B^{(1)} + \mathbf{h}_C^{(1)} + \mathbf{h}_D^{(1)}) + \mathbf{B}_2 \mathbf{h}_A^{(1)}\right)$$

Compute Graph For Target Node

- Graph neural network, from perspective of a single node:
  - At Layer 0, embedding is identical to the node features
  - At Layer 1, embedding contains information from node features and features of direct neighbors
  - At Layer 2, embedding contains information from nodes with distance $\leq$2

$$\mathbf{h}_D^{(1)} = \sigma\left(\mathbf{W}_1\mathbf{h}_A^{(0)} + \mathbf{B}_1\mathbf{h}_D^{(0)}\right)$$

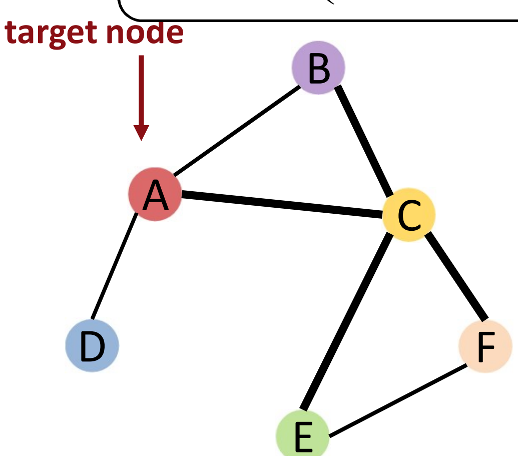
Example: 2-layer network

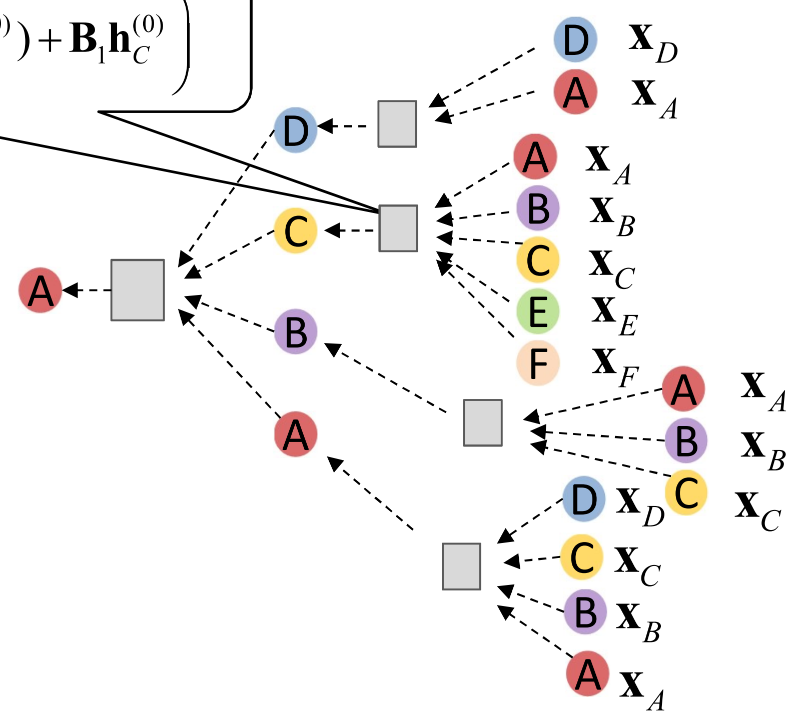


Compute Graph For Target Node

- Graph neural network, from perspective of a single node:
  - At Layer 0, embedding is identical to the node features
  - At Layer 1, embedding contains information from node features and features of direct neighbors
  - At Layer 2, embedding contains information from nodes with distance ≤ 2

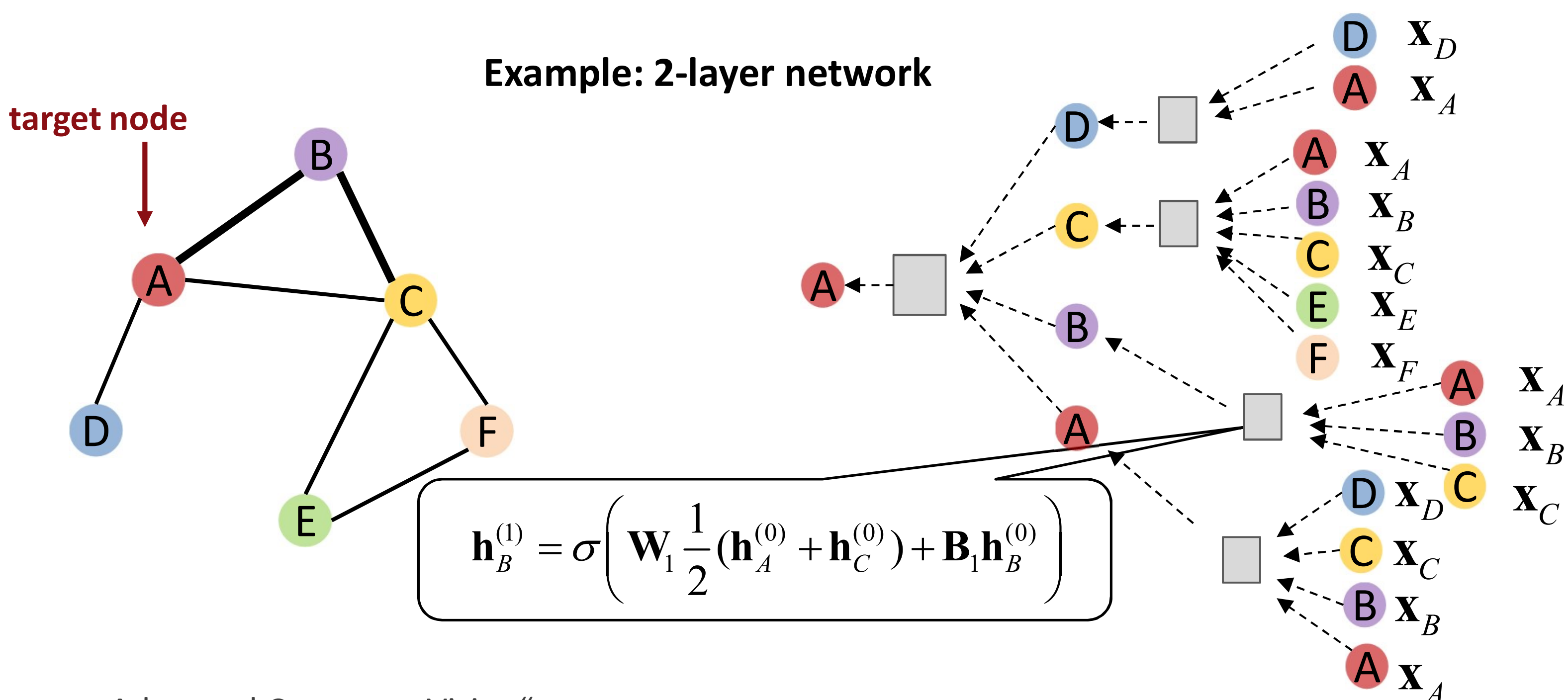
$$\mathbf{h}_C^{(1)} = \sigma\left(\mathbf{W}_1 \frac{1}{4}(\mathbf{h}_A^{(0)} + \mathbf{h}_B^{(0)} + \mathbf{h}_E^{(0)} + \mathbf{h}_F^{(0)}) + \mathbf{B}_1 \mathbf{h}_C^{(0)}\right)$$





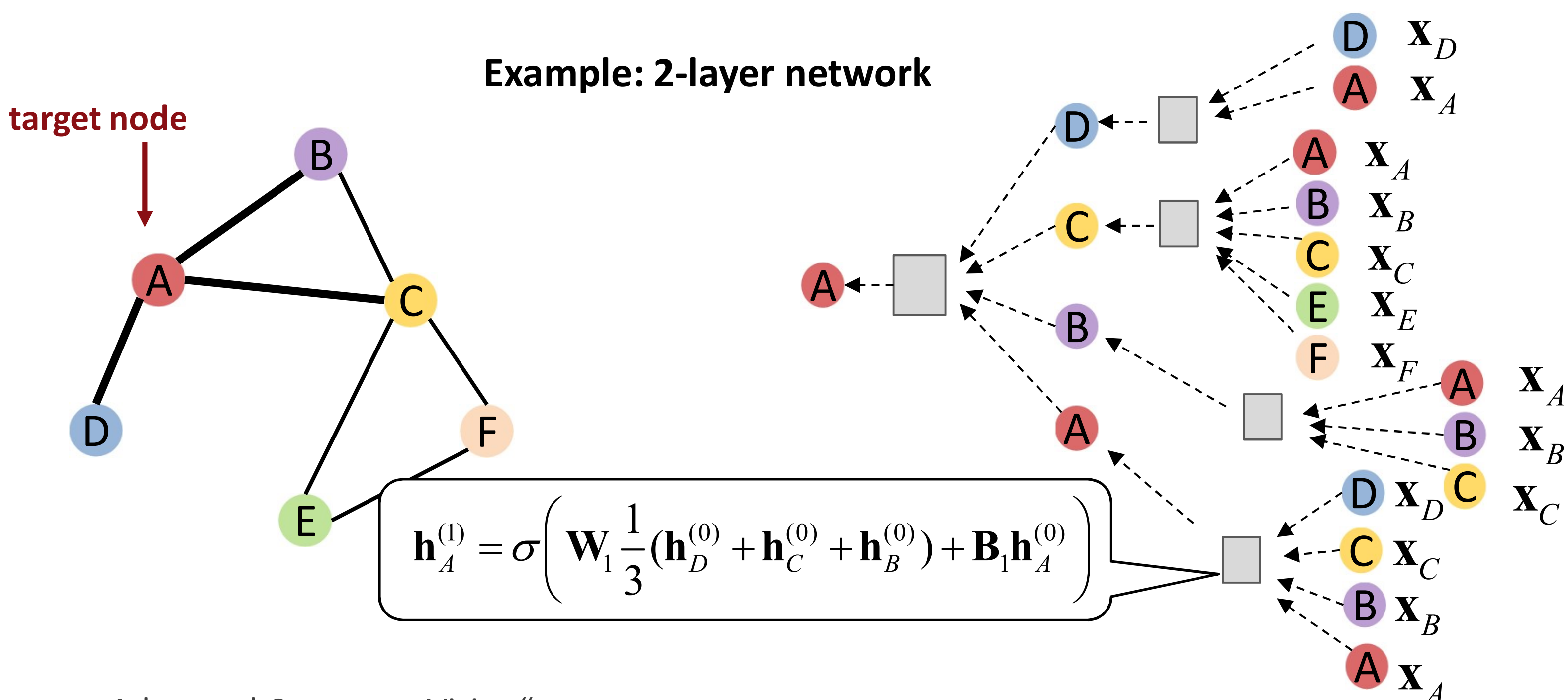
Compute Graph For Target Node

- Graph neural network, from perspective of a single node:
  - At Layer 0, embedding is identical to the node features
  - At Layer 1, embedding contains information from node features and features of direct neighbors
  - At Layer 2, embedding contains information from nodes with distance ≤ 2



Compute Graph For Target Node

- Graph neural network, from perspective of a single node:
  - At Layer 0, embedding is identical to the node features
  - At Layer 1, embedding contains information from node features and features of direct neighbors
  - At Layer 2, embedding contains information from nodes with distance ≤ 2



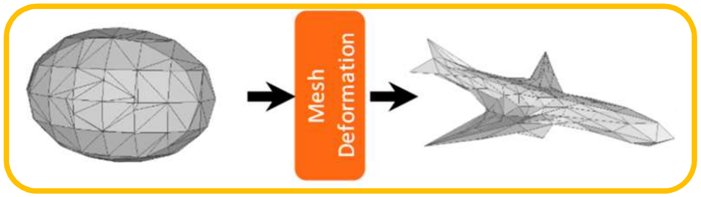
Notes on Graph Neural Network So Far

- Trainable parameters $\boldsymbol{\theta}$ in the model: matrices $\mathbf{W}_l, \mathbf{B}_l$ for $l \in \{0, ..., L\}$
- The weights are different for each layer, but shared across all aggregation operations within one layer (the square nodes in the figure above)
- Dimensionality of $\mathbf{h}_v^{(l)}$ can change between different layers (as number of channels in CNN)
- Can build networks of any depth: the deeper, the more long-range dependencies in the graph are modeled

Graph Neural Network For Mesh Deformation

- Back to Pixel2Mesh: Carry out mesh deformation with a graph neural network

Ellipsoid: $\mathbf{y} = (V_1, E_1, \mathbf{C}_1)$
Set of vertices V_1
Set of edges $E_1 \subseteq V_1 \times V_1$
Initial vertex coordinates $\mathbf{C}_1$,
with $\mathbf{c} \in \mathbb{R}^3$ for $\mathbf{c} \in \mathbf{C}_1$



Airplane: $\mathbf{y} = (V_1, E_1, \mathbf{C}_2)$
Set of vertices V_1
Set of edges $E_1 \subseteq V_1 \times V_1$
New vertex coordinates $\mathbf{C}_2$,
with $\mathbf{c} \in \mathbb{R}^3$ for $\mathbf{c} \in \mathbf{C}_2$

Input to graph neural network:

Graph $(V_1, E_1, \mathbf{X}_0)$

V_1, E_1 describe graph structure in ellipsoid mesh

Node features are of the form $\mathbf{X}_0 = \mathbf{C}_0 \oplus \mathbf{P}_0$, where $\oplus$ denotes concatenation:

$\mathbf{x} = \begin{pmatrix} \mathbf{c} \\ \mathbf{p} \end{pmatrix}$ with $\mathbf{c} \in \mathbf{C}_0$ and $\mathbf{p} \in \mathbf{P}_0$

$\mathbf{c} \in \mathbb{R}^3$ initial coordinate for vertex

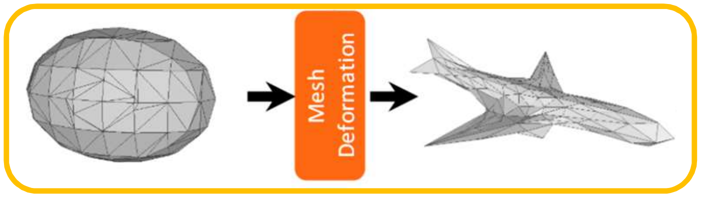
$\mathbf{p} \in \mathbb{R}^{1280}$ "perceptual" features extracted from 2D-image (details later)

Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Graph Neural Network For Mesh Deformation

- Back to Pixel2Mesh: Carry out mesh deformation with a graph neural network

Ellipsoid: $\mathbf{y} = (V_1, E_1, \mathbf{C}_1)$
Set of vertices V_1
Set of edges $E_1 \subseteq V_1 \times V_1$
Initial vertex coordinates $\mathbf{C}_1$,
with $\mathbf{c} \in \mathbb{R}^3$ for $\mathbf{c} \in \mathbf{C}_1$



Airplane: $\mathbf{y} = (V_1, E_1, \mathbf{C}_2)$
Set of vertices V_1
Set of edges $E_1 \subseteq V_1 \times V_1$
New vertex coordinates $\mathbf{C}_2$,
with $\mathbf{c} \in \mathbb{R}^3$ for $\mathbf{c} \in \mathbf{C}_2$

Output of graph neural network:

New node features (embeddings) $\mathbf{E}_1 = \mathbf{C}_1 \oplus \mathbf{F}_1$

$$\mathbf{e} = \begin{pmatrix} \mathbf{c} \\ \mathbf{f} \end{pmatrix} \text{ with } \mathbf{c} \in \mathbf{C}_1 \text{ and } \mathbf{f} \in \mathbf{F}_1$$

$\mathbf{c} \in \mathbf{C}_1$ are the new coordinates of a node, with $\mathbf{c} \in \mathbb{R}^3$

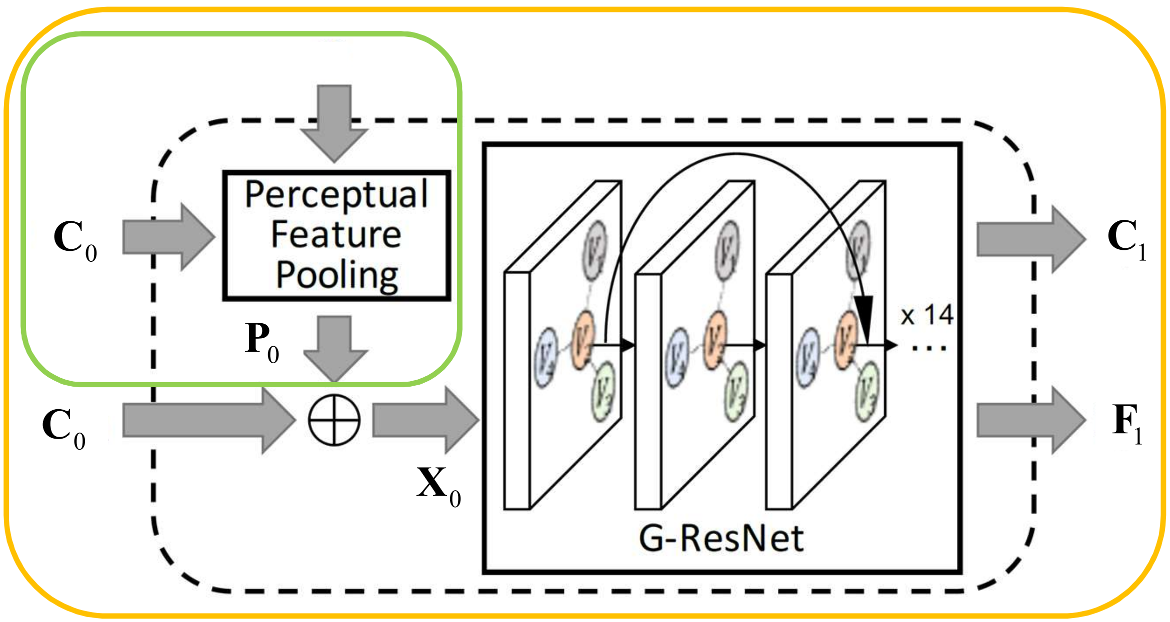
$\mathbf{f} \in \mathbf{F}_1$ is additional learned 3D-shape feature information for a node, with $\mathbf{f} \in \mathbb{R}^{128}$

Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Residual Graph Neural Network

- Mesh deformation: graph neural network using residual architecture layers
  - 14 graph convolution layers, with shortcut connections
  - Number of channels (dimensionality of $\mathbf{h}_v^{(l+1)}$) is 128 in intermediate layers, in final layer it is 3+128=131 to produce $\mathbf{C}_1 \oplus \mathbf{F}_1$

Features coming from image (details see below)



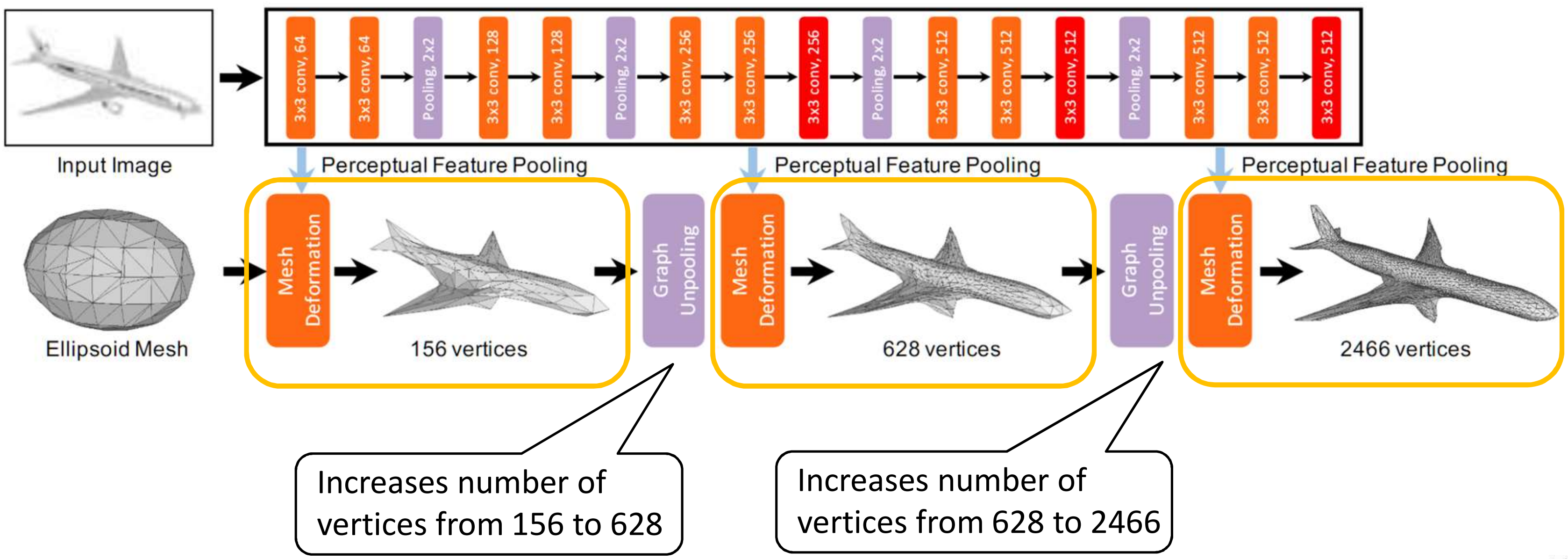
First (low-resolution) 3D-mesh for object in image

Learned 3D-shape features that can represent context within the 3D-shape and are used in later blocks of the model

Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Succesively Increasing Mesh Resolution

- **Graph unpooling layers: increase resolution of mesh**
  - It is easier to first infer a relatively low-resolution 3D-shape and refine it later than to directly infer a high-resolution mesh
  - Network contains three mesh deformation blocks with intermediate "graph unpooling" layers that increase the resolution of the mesh



Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Graph Unpooling Layer

- Graph unpooling layer: transform graph structure by increasing the resolution of the mesh

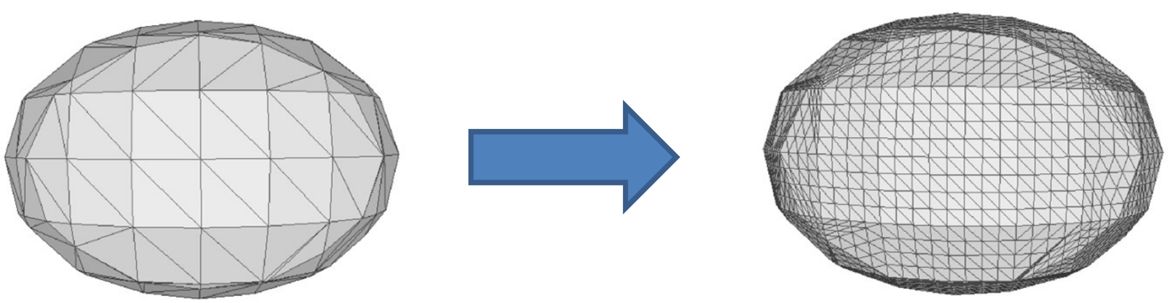
Input:

Graph (V_i, E_i), coordinates $\mathbf{C}_i$ and 3D-shape features $\mathbf{F}_i$

Output:

Graph $(\mathrm{V}_{i+1}, E_{i+1})$, with coordinates $\overline{\mathbf{C}}_i$ and 3D-shape features $\overline{\mathbf{F}}_i$

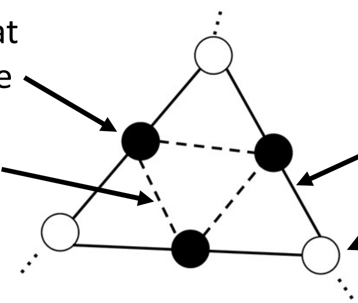
$|V_{i+1}| > |V_i|, \ |E_{i+1}| > |E_i|$

(V_i, E_i)  $(\mathrm{V}_{i+1}, E_{i+1})$

Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Graph Unpooling Layer

- Graph unpooling by splitting edges:
  - For each edge in E_i , introduce a novel vertex at the center of the edge
  - For any three novel vertices that are adjacent to the same triangle on the surface, connect them by novel edges

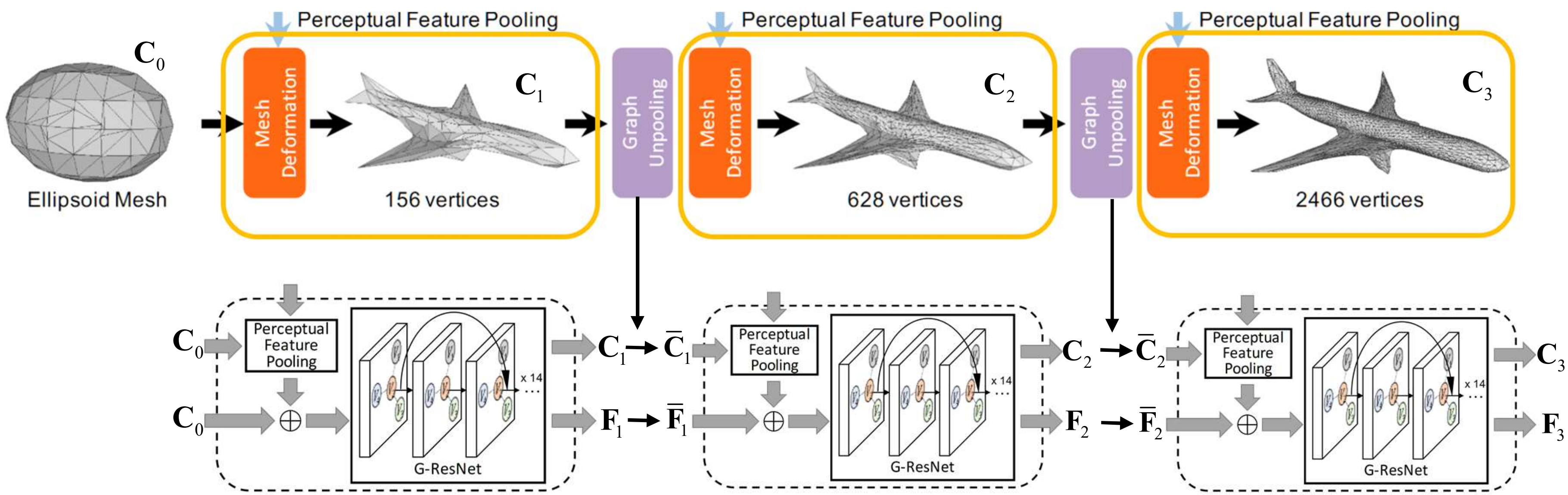


- 3D-shape features $\mathbf{F}_i$ and coordinates $\mathbf{C}_i$ must be adapted to the new vertex set (then called $\bar{\mathbf{F}}_i$ and $\bar{\mathbf{C}}_i$)
  - For new vertices, set features to the average of the two vertices that were the endpoints of the edge that was split
  - For old vertices, keep old feature vectors

Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Full Mesh Deformation Pipeline

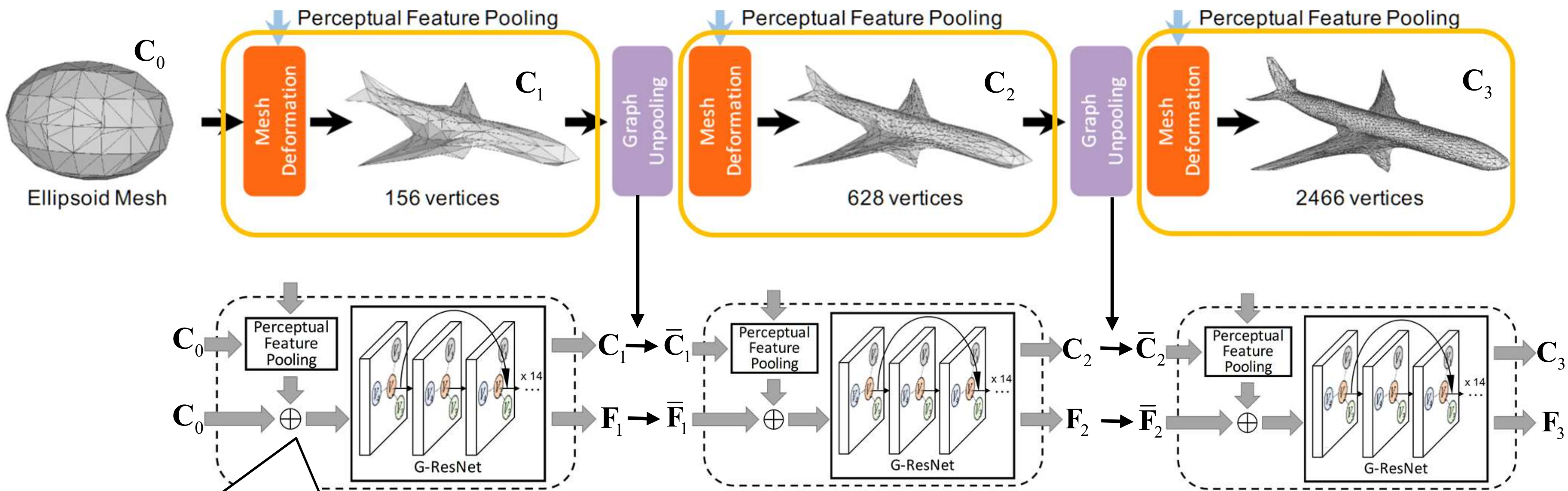
- Full mesh deformation pipeline, including graph unpooling layers and additional residual graph neural network blocks:



Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Full Mesh Deformation Pipeline

- Full mesh deformation pipeline, including graph unpooling layers and additional residual graph neural network blocks:

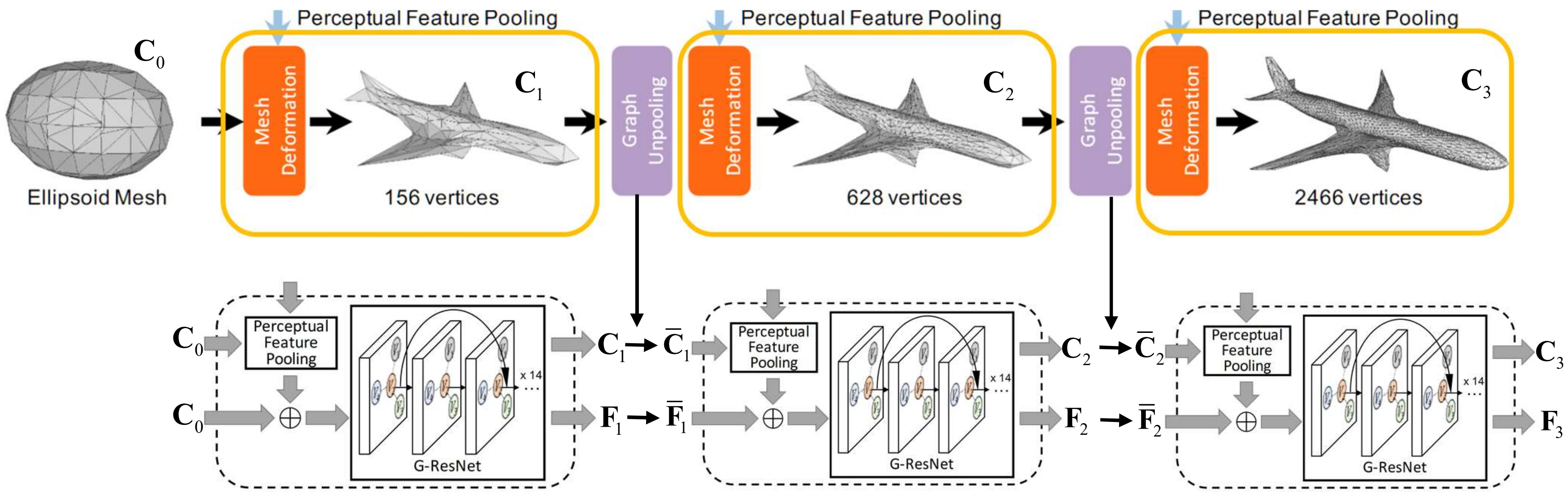


Initial mesh size is 156 vertices.
Initial coordinates $\mathbf{C}_0$ define ellipsoid.
Node feature inputs in first deformation block are coordinates $\mathbf{C}_0$ and perceptual features from 2D-image.

Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Full Mesh Deformation Pipeline

- Full mesh deformation pipeline, including graph unpooling layers and additional residual graph neural network blocks:

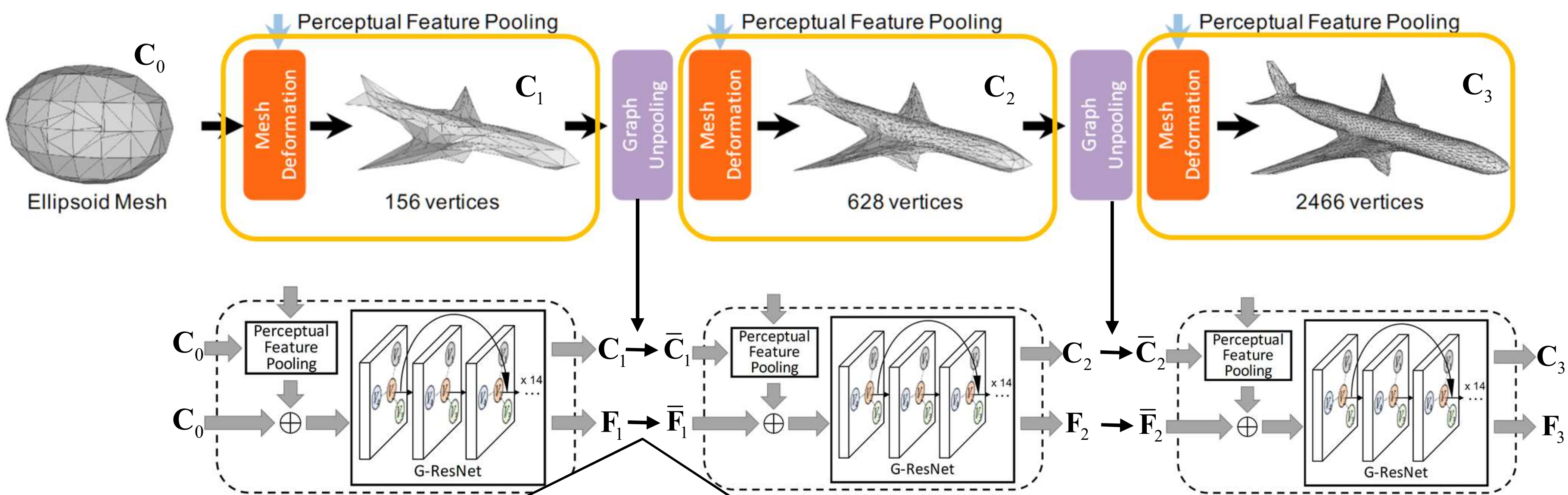


Residual graph neural network transforms initial coordinates $\mathbf{C}_0$ into new coordinates $\mathbf{C}_1$.
Also outputs 128-dim 3D-shape features $\mathbf{F}_1$ for use in later layers.

Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Full Mesh Deformation Pipeline

- Full mesh deformation pipeline, including graph unpooling layers and additional residual graph neural network blocks:

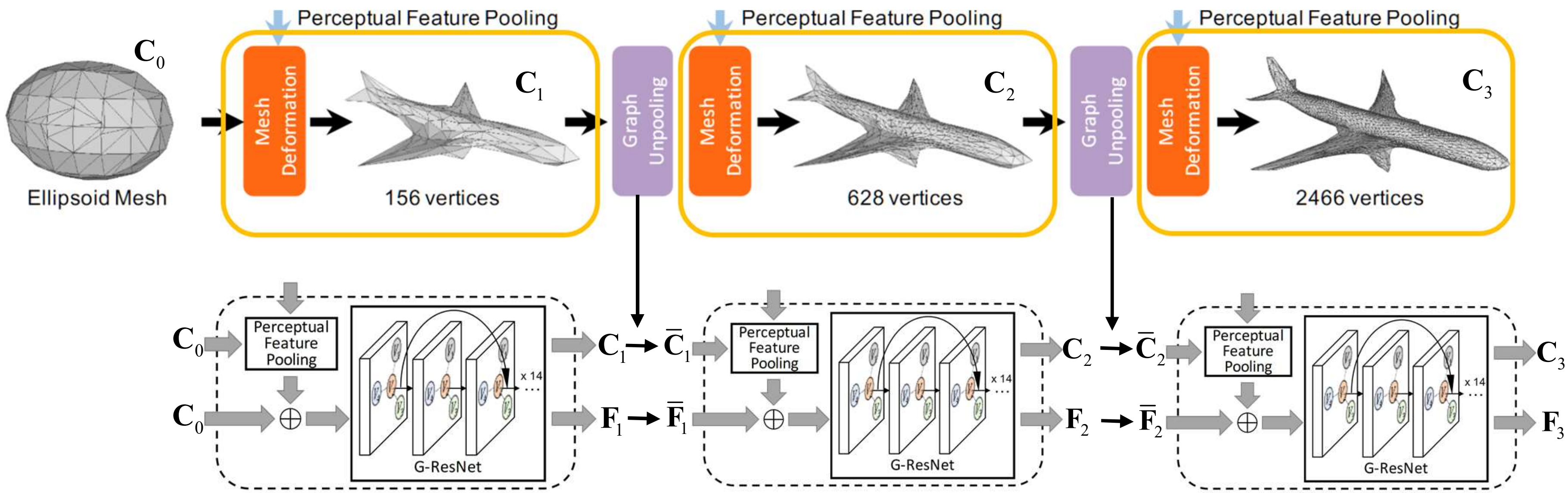


Graph unpooling layer increases the resolution of mesh, leading to more nodes and vertices.
Constructs new coordinate and 3D-shape feature representations $\bar{\mathbf{C}}_1$, $\bar{\mathbf{F}}_1$ for the new vertex set.
Note that the compute graph of the network only deals with node vectors $\mathbf{C}_i, \bar{\mathbf{C}}_i, \mathbf{F}_i, \bar{\mathbf{F}}_i$.
The graph structure determines connectivity patterns between the vectors, but is not itself part of the compute graph.

Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Full Mesh Deformation Pipeline

- Full mesh deformation pipeline, including graph unpooling layers and additional residual graph neural network blocks:

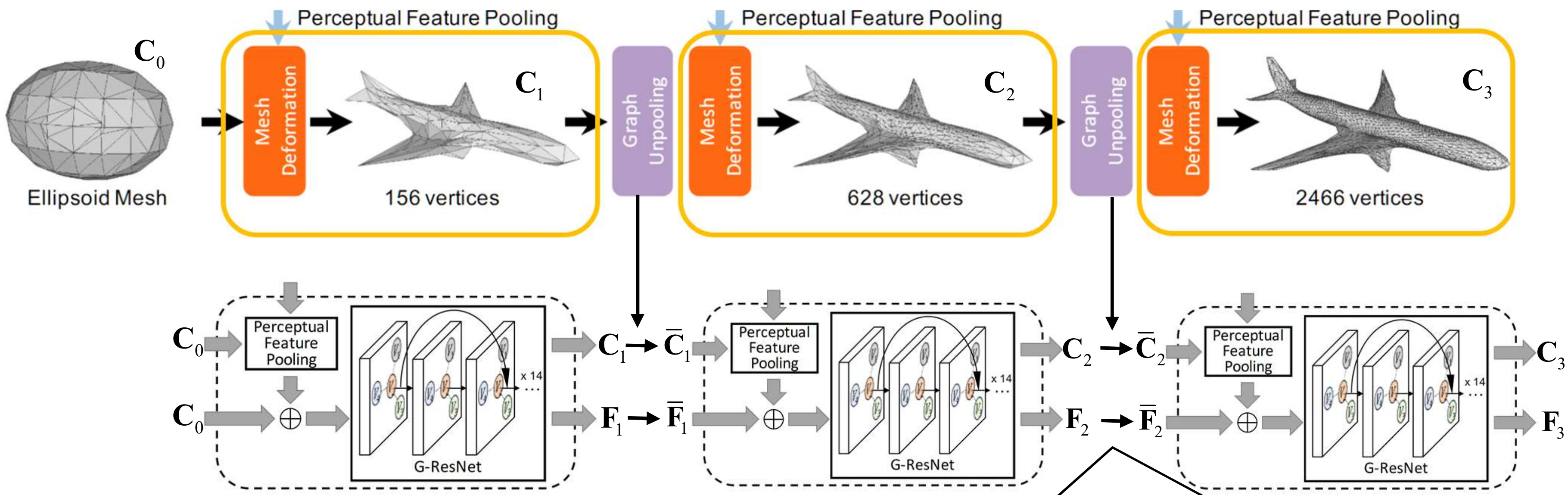


Second residual graph neural network block computes refined coordinates $\mathbf{C}_2$.
Input node features are the 3D-shape features $\bar{\mathbf{F}}_1$ learned in the first graph neural network, and the perceptual image features.
Outputs are again new coordinates and new 3D-shape features $\mathbf{F}_2$.

Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Full Mesh Deformation Pipeline

- Full mesh deformation pipeline, including graph unpooling layers and additional residual graph neural network blocks:

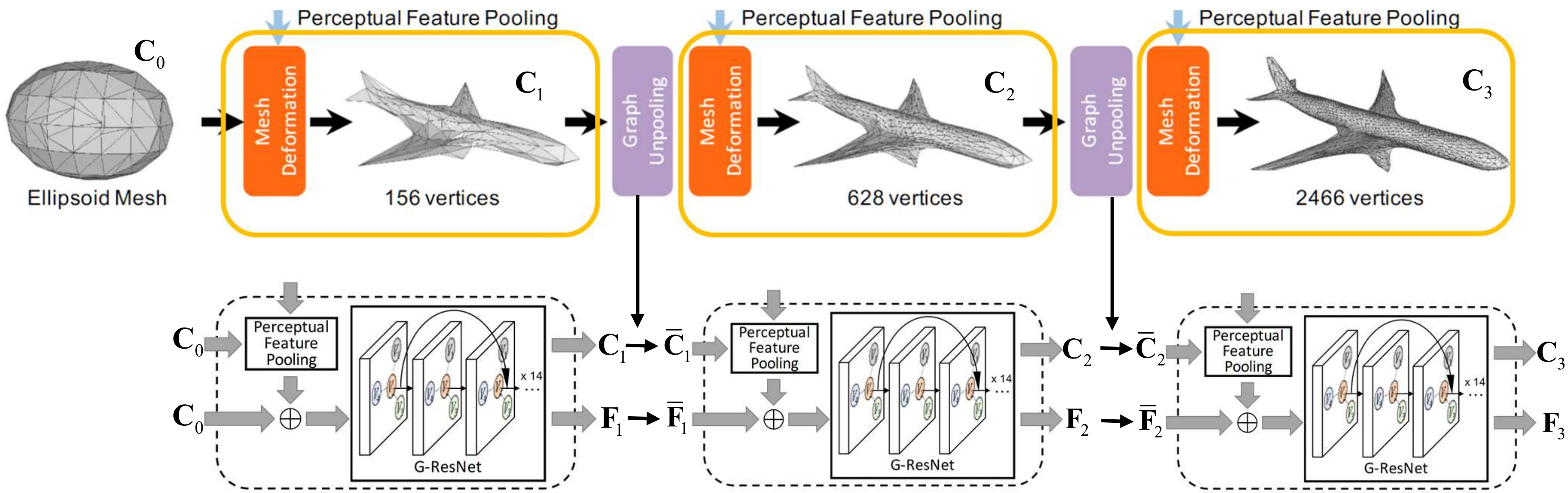


Graph unpooling layer again increases the resolution of mesh, leading to more nodes and vertices and correspoding new coordinate and 3D-shape features $\bar{C}_2$, $\bar{F}_2$

Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Full Mesh Deformation Pipeline

- Full mesh deformation pipeline, including graph unpooling layers and additional residual graph neural network blocks:



Third residual graph neural network block computes final coordinates C_3 .
End-to-end trainable structure: everything goes into one compute graph.

Wang, Nanyang, et al. "Pixel2mesh: Generating 3d mesh models from single rgb images." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

What is Still Missing

- **What we did not yet talk about**:
  - Perceptual image features: how is the 2D-image information utilized in the residual graph neural network blocks to infer the 3D-shape features and 3D-coordinates?
  - For training the model: what is the loss function? In particular, how do we measure the distance between two 3D-meshes?