# MeshRunner: Improving MeshWalker

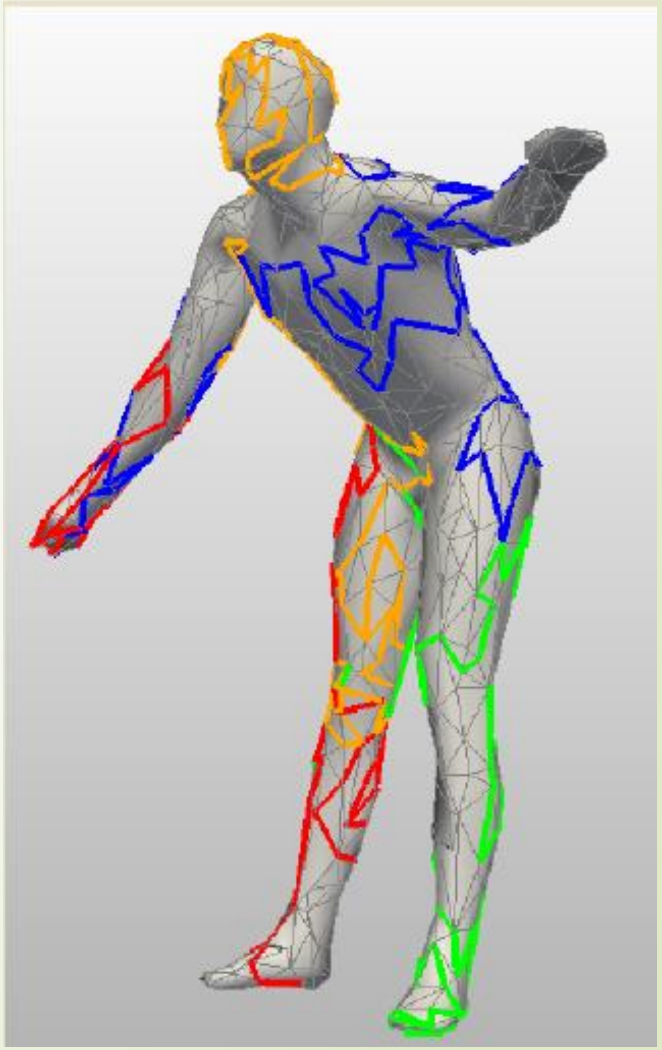
Presented by: Itay Levy, Itamar Zimerman

and Amit Cohen

# Background

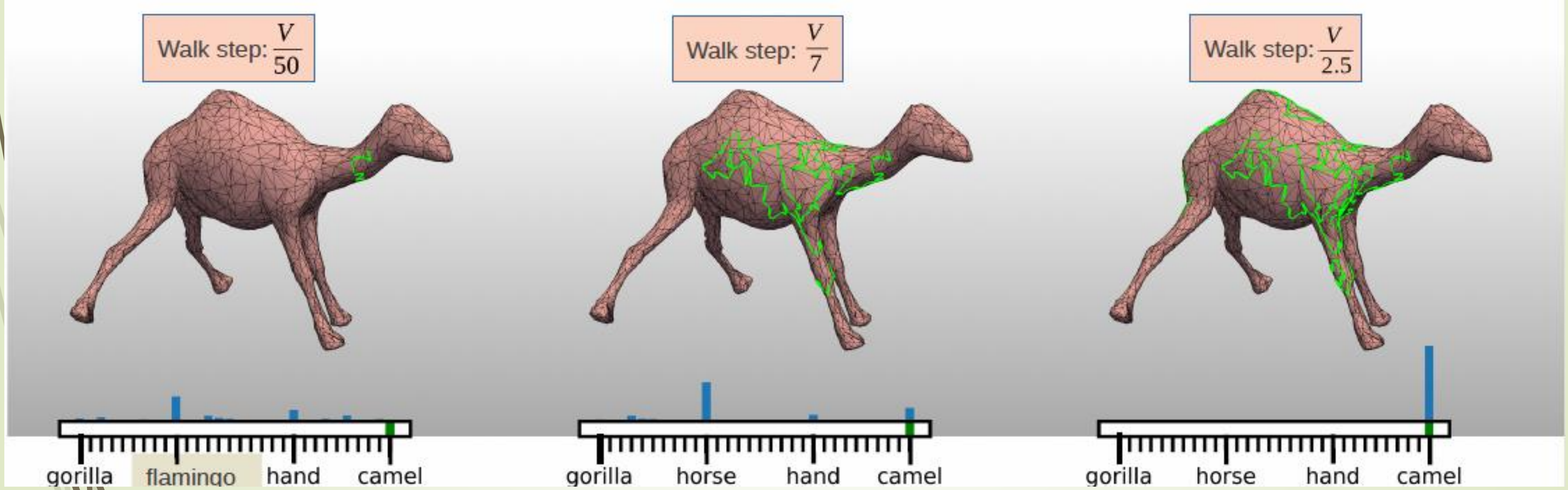
# Reminder – what is MeshWalker?

- Apply Deep Learning directly on meshes
- Perform random walks on the mesh's surface and feed it to an RNN
- Use the RNN's hidden state to represent the mesh
- Perform classification (or segmentation) over the representation



# Visualization

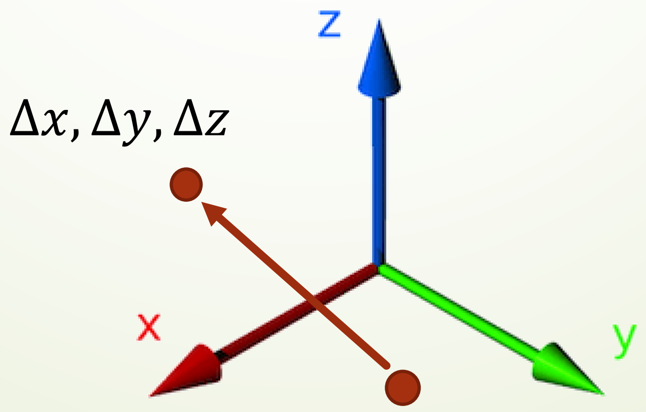
V – total number of vertices



# What is a walk?

## Walk Generation

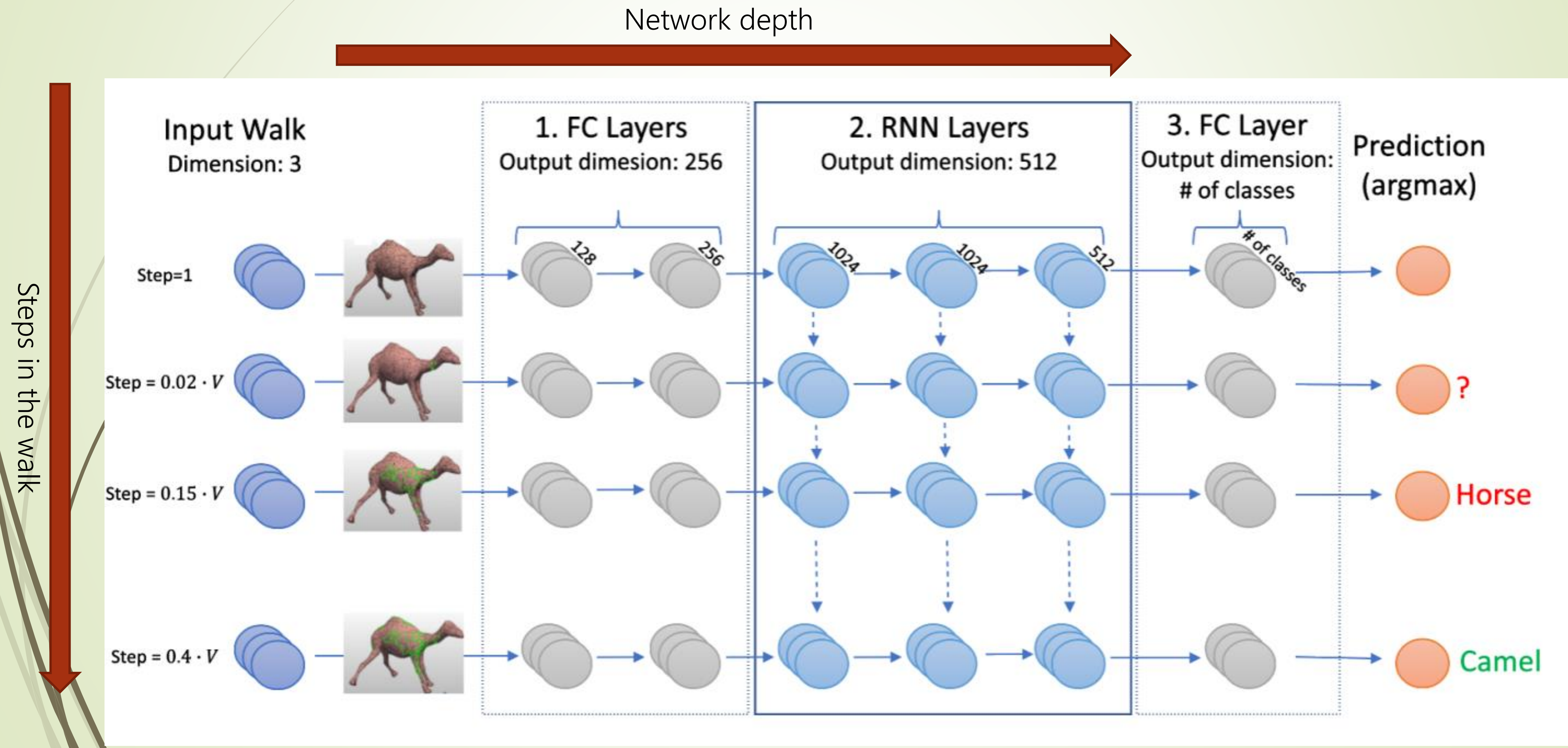
- Start with a random vertex in the mesh
- Iteratively adding vertices
  - next vertex ← randomly chosen from the **unvisited** adjacent vertices
  - If none exist, go back (Backtracking)
  - If stuck, jump to a new random vertex



# SHREC11 dataset

- Relatively simple mesh dataset
- Consists of 30 classes, 20 examples per class
- After simplification, the meshes contain 250 vertices

# Architecture



# Our Approach

# Our approach

- Improved walk generation
  - Choose starting point by analyzing saliency
  - Apply skips, jumps and ordering
- Improved walk representation
- Attention based information exchange between walks

# Exploring the Walk Generation

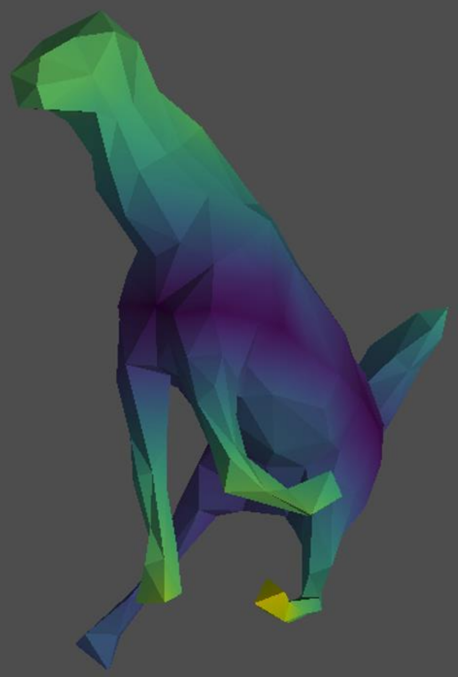
- What is a good walk?
  - Focus on important areas
  - High mesh coverage
- Limited Resources:
  - Running Time
  - Walk Length (RNN)

# Walk Generation - Saliency

- A heatmap of the mesh vertices
- Highlights the informative parts of the topology
- Used to improve the random vertex choice
- Our implementation is based on [the Mesh Saliency paper]

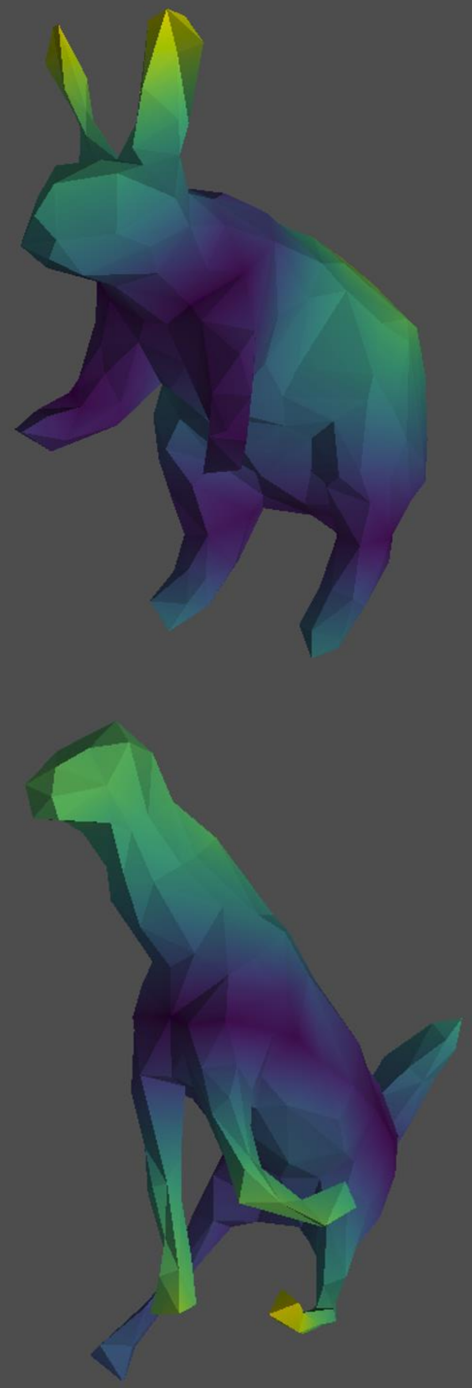
$$G(\mathscr{C}(v),\sigma) = \frac{\sum_{x \in N(v,2\sigma)} \mathscr{C}(x) exp[-\|x-v\|^2/(2\sigma^2)]}{\sum_{x \in N(v,2\sigma)} exp[-\|x-v\|^2/(2\sigma^2)]}$$

$$\mathscr{S}(v) = |G(\mathscr{C}(v),\sigma) - G(\mathscr{C}(v),2\sigma)|$$



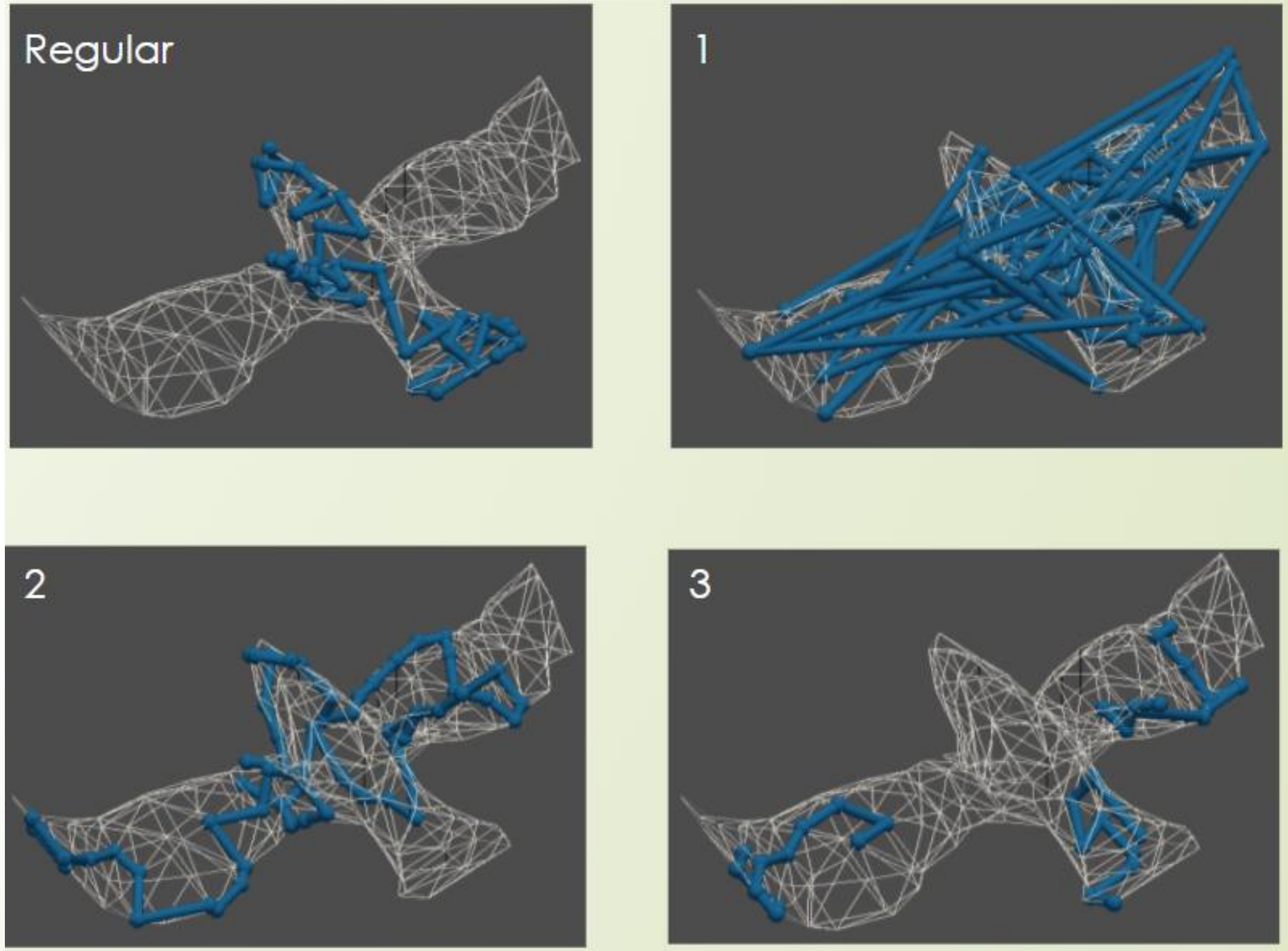
# Saliency - Implementation Issues

- Radius
- Temperature
- Performance



# Exploring the Walk Generation

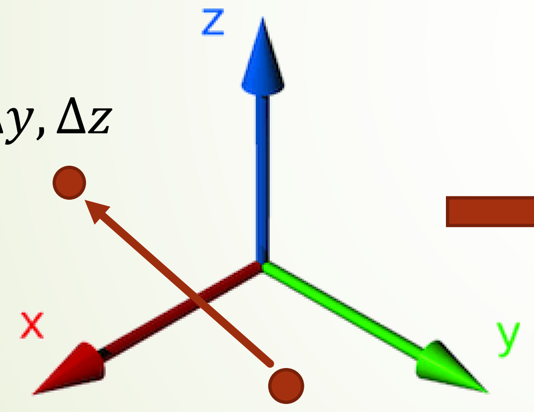
- Random Vertices(1)
- Skips(2)
- Jumps(3)
- Fixed sorting order
  - Inspired by [Polygen (DeepMind, 2021)]
  - Completely ignore edges



# Improved Walk Representation

- Inform the network about jumps or backtracking

$\Delta x, \Delta y, \Delta z$




$\Delta x, \Delta y, \Delta z, J, B$

$J, B \in \{0,1\}$

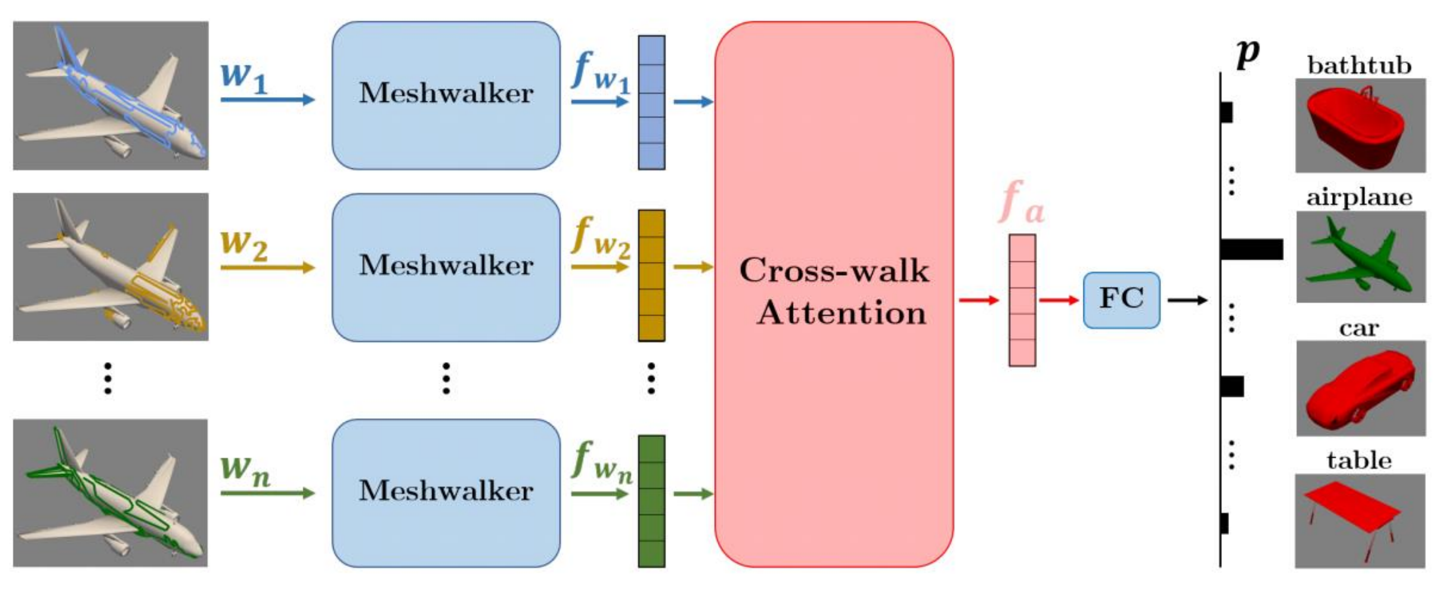
# Information exchange between walks

# Previous approaches - MeshWalker

- Training - No information exchange
- Inference
  - Each walk produces a vector of probabilities to belong to the different classes (dim=30)
  - These vectors are averaged to produce the final result

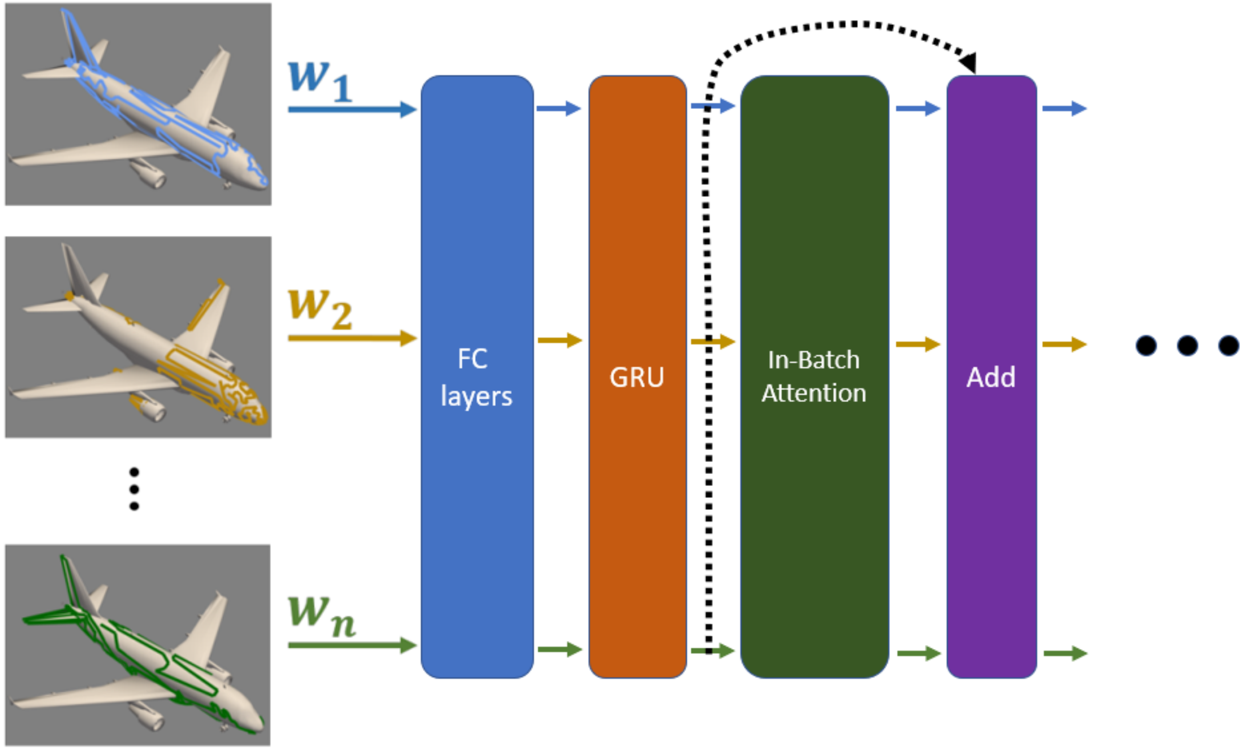
# Previous approaches - AttWalk

- Each walk produces a feature vector (dim=512)
- Uses Many-to-One attention to generate a single feature vector (no code yet)



# Our Approach – In-Batch Attention

- Integrating information exchange into the model
- Different walks are aware of each other
  - As they progress, not just at the end
  - Both in training and inference



# Challenges

- Dataset (SHREC11):
  - Mesh size
  - 100% classification acc
- Tailored design choices
- Randomness
- Hyper-Parameter strategy

# Results

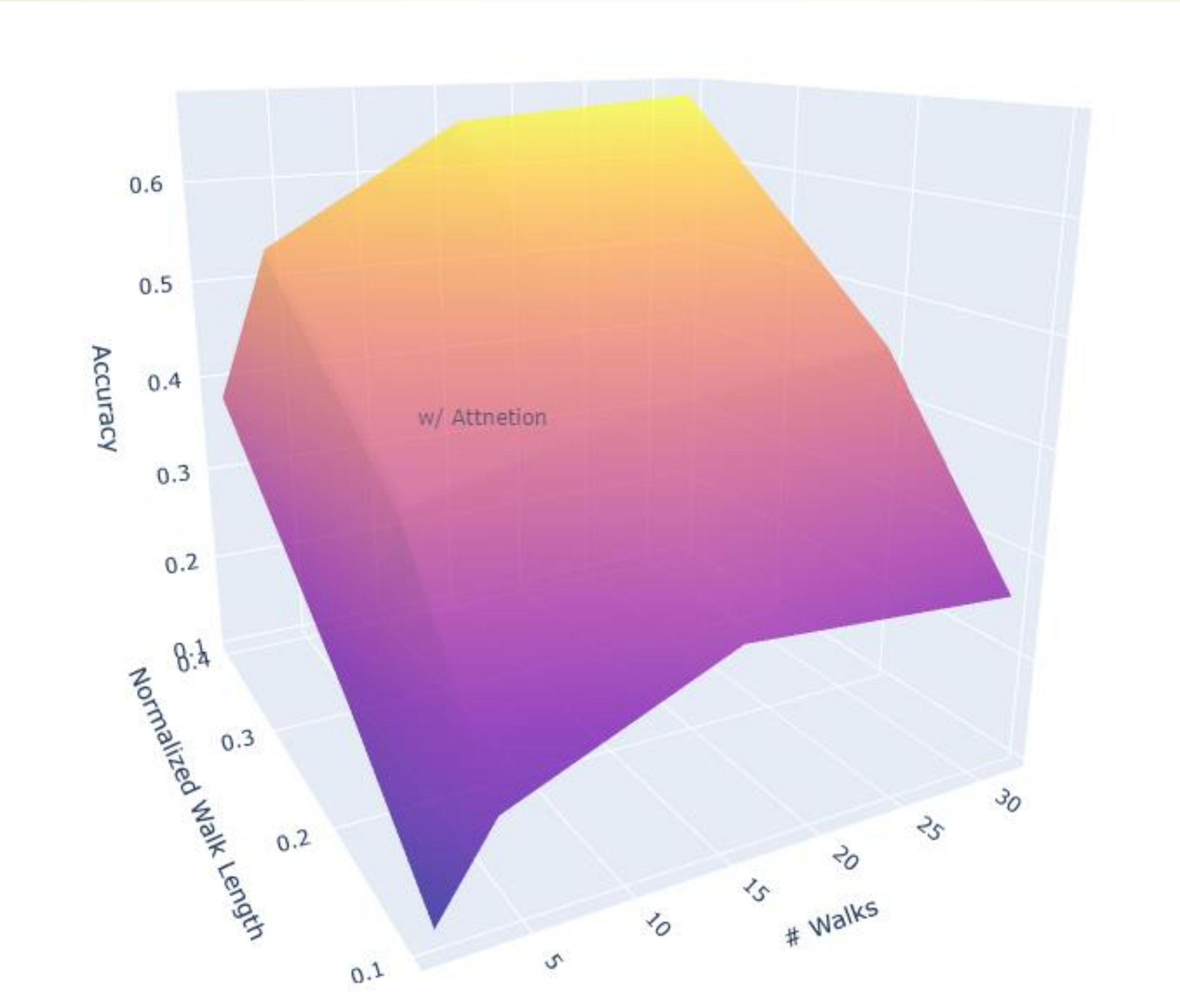
# Different Approaches

| | Number of walks | | | | | |
|---|---|---|---|---|---|---|
| | 1 | | 4 | | 16 | |
| | acc | rank | acc | rank | acc | rank |
| Baseline | 0.4 | 4 | 0.59 | 5 | 0.71 | 4 |
| + Attention | 0.37 | 5 | 0.67 | 2 | 0.79 | 1 |
| + Representation | 0.44 | 3 | 0.63 | 4 | 0.74 | 3 |
| + Saliency | 0.46 | 2 | 0.65 | 3 | 0.70 | 5 |
| All | 0.49 | 1 | 0.68 | 1 | 0.77 | 2 |

# Changing the walk generation method

| | Number of walks | | | | | |
|---|---|---|---|---|---|---|
| | 1 | | 4 | | 16 | |
| | acc | rank | acc | rank | acc | rank |
| Baseline | 0.42 | 4 | 0.56 | 3 | 0.73 | 2 |
| Representation | 0.49 | 1 | 0.67 | 1 | 0.8 | 1 |
| Skips(2) | 0.46 | 3 | 0.59 | 2 | 0.68 | 3 |
| Skips(4) | 0.42 | 5 | 0.53 | 5 | 0.64 | 4 |
| Jumps(0.1) w representation | 0.32 | 7 | 0.47 | 6 | 0.54 | 6 |
| Jumps(0.03) w representation | 0.47 | 2 | 0.58 | 4 | 0.63 | 5 |
| Random jumps (dx,dy,dz) | 0.09 | 9 | 0.11 | 9 | 0.13 | 9 |
| Random jumps (x,y,z) | 0.38 | 6 | 0.46 | 7 | 0.51 | 7 |
| Order | 0.12 | 8 | 0.19 | 8 | 0.25 | 8 |

# Effect of # walks & walk length



# Conclusion