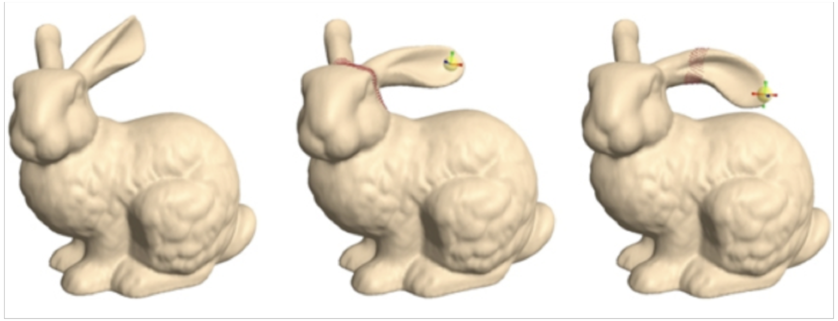
UC San Diego

Laplacian in Graph Embedding

Yikuan Xia, Yuzhe Qin, Chutong Yang

Feb 14, 2019

Surface Editing



Challenge:
How to preserve details of the surface as much as possible?

Laplacian Surface Editing

O. Sorkine[1], D. Cohen-Or[1], Y. Lipman[1], M. Alexa[2], C. Rössl[3] and H.-P. Seidel[3]

[1]School of Computer Science, Tel Aviv University
[2]Discrete Geometric Modeling Group, Darmstadt University of Technology
[3]Max-Planck Institut für Informatik, Saarbrücken

Why Laplacian?

$V = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$

Euclidean Coordinates $\longrightarrow$ Laplacian Coordinates

$\Delta = \{\delta_i\}$

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

$j \in \mathcal{N}_i$: indices of neighbors of $\mathbf{v}_i$

Encode Intrinsic Geometry (local shape details) of the Surface! (How?)

Whiteboard Time

Matrix Form: $L = I - D^{-1}A$ $\Delta = LV$

A is the mesh adjacency matrix

$D = \mathrm{diag}(d_1, \ldots, d_n)$ where d_i is the degree of v_i

***Theorem**: Let G be a graph. Then the dimension of the nullspace of L(G) is the number of connected components of G.*

In a connected mesh, L has rank n-1. Given Δ and L, V can be recovered by fixing one vertex. (Invariant to translation)

Objective

Given the original vertices $V = \{\mathbf{v_1}, \ldots, \mathbf{v_n}\}$,

Find a set of new vertices $V' = \{\mathbf{v'_1}, \ldots, \mathbf{v'_n}\}$ such that the desired constraints

$$\mathbf{v}'_i = \mathbf{u}_i, \quad i \in \{m, \ldots, n\},\ m < n$$

given by the user's operation is satisfied, and the detail of the mesh is preserved as much as possible.

Minimize the error function:

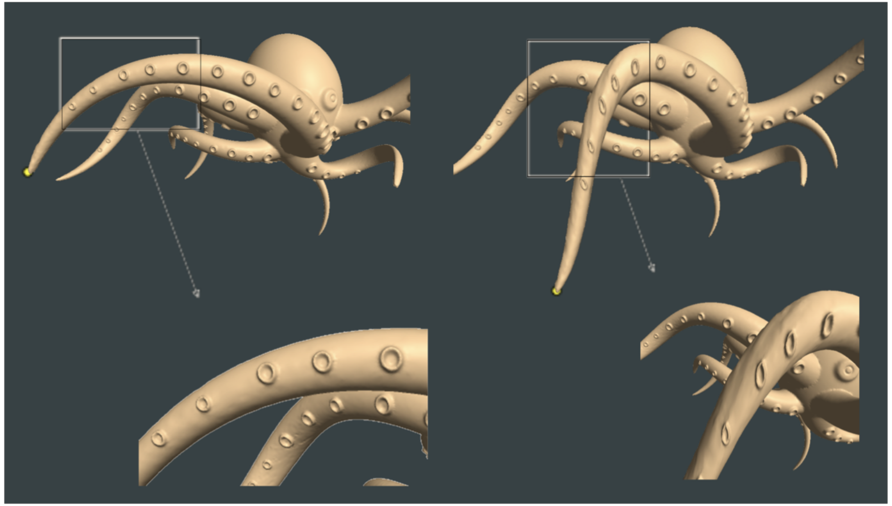
$$E(V') = \underbrace{\sum_{i=1}^{n} \left\| \delta_i - \mathscr{L}(\mathbf{v}'_i)) \right\|^2}_{\text{To preserve surface details}} + \underbrace{\sum_{i=m}^{n} \|\mathbf{v}'_i - \mathbf{u}_i\|^2}_{\text{To satisfy constraints specified by the user}},$$

Objective

$$E(V') = \sum_{i=1}^{n} \left\| \delta_i - \mathcal{L}(\mathbf{v}'_i) \right\|^2 + \sum_{i=m}^{n} \left\| \mathbf{v}'_i - \mathbf{u}_i \right\|^2,$$

Laplacian Coordinates are only invariant to translation, not scale or rotation.

If $\mathbf{u_i}$ implies a linear transformation consisting of rotation or scaling, then details of the surface cannot be transformed properly.



Objective

To make the laplacian coordinates robust to such linear transformations, compute an appropriate transformation T_i for each vertex and revise the error function:

$$E(V') = \sum_{i=1}^{n} \left\| T_i(V')\delta_i - \mathcal{L}(\mathbf{v}_i') \right\|^2 + \sum_{i=m}^{n} \|\mathbf{v}_i' - \mathbf{u}_i\|^2$$

T_i is unknown but can be expressed as a linear function of V':

T_i computed by: $$\min_{T_i} \left(\|T_i\mathbf{v}_i - \mathbf{v}_i'\|^2 + \sum_{j \in \mathcal{N}_i} \|T_i\mathbf{v}_j - \mathbf{v}_j'\|^2 \right)$$

(transformations should be similar for $\mathbf{v}_i$ and its neighbors)

Objective

Problem:

T_i is unconstrained and may lead to the irregular distortions which are not categorized as rigid body transformation or scaling.

Reasonable T_i should consist of **translation**, **isotropic scales** and **rotation**.

Objective

Linearize Rotation:

3D rotation determined by an axis $\mathbf{u}$ and a angle of rotation θ

Assume $\mathbf{u} = (u_1, u_2, u_3)^T$ and $\|\mathbf{u}\| = 1$

Associate a skew-symmetric matrix $S_{\mathbf{u}} = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}$

For all $\mathbf{v}$, $\mathbf{u} \times \mathbf{v} = S_{\mathbf{u}}\,\mathbf{v}$

Then the rotation matrix computed by:

$$R = e^{S_{\mathbf{u}}\theta} = I + \sin\theta\, S_{\mathbf{u}} + (1 - \cos\theta)\; S_{\mathbf{u}}^2$$

(Rodrigues formula)

A linear approximation is needed.

Omit $(1 - \cos\theta)\; S_{\mathbf{u}}^2$ when θ is small.

Objective:

Add in isotropic scaling s and translation $\mathbf{t}$, put in homogeneous system:

$$\begin{bmatrix} sR & \mathbf{t} \\ 0 & 1 \end{bmatrix}$$

Let $\mathbf{h} = (h_1, h_2, h_3)^T = s \sin\theta\ \mathbf{u}$

$$T_i = \begin{pmatrix} s & -h_3 & h_2 & t_x \\ h_3 & s & -h_1 & t_y \\ -h_2 & h_1 & s & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

T_i is characterized by s, $\mathbf{h}$ and $\mathbf{t}$

Objective:

Construct $$A_i = \begin{pmatrix} v_{k_x} & 0 & v_{k_z} & -v_{k_y} & 1 & 0 & 0 \\ v_{k_y} & -v_{k_z} & 0 & v_{k_x} & 0 & 1 & 0 \\ v_{k_z} & v_{k_y} & -v_{k_x} & 0 & 0 & 0 & 1 \\ \vdots & & & & & & \end{pmatrix}, k \in \{i\} \cup \mathcal{N}_i$$

and $$\mathbf{b_i} = \begin{pmatrix} v'_{k_x} \\ v'_{k_y} \\ v'_{k_z} \\ \vdots \end{pmatrix}, k \in \{i\} \cup \mathcal{N}_i$$

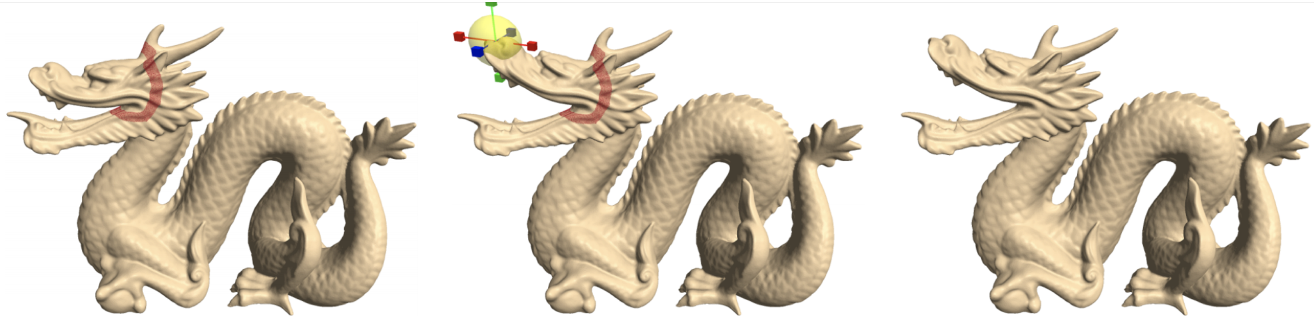
T_i determined by minimizing $\| A \begin{pmatrix} s_i \\ \mathbf{h}_i \\ \mathbf{t}_i \end{pmatrix} - \mathbf{b_i} \|$

Limitation

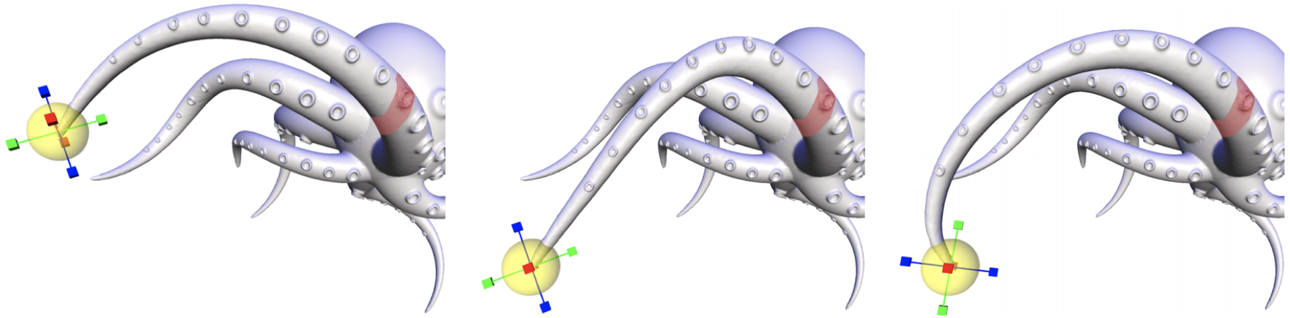
Can’t handle rotation with large angle.

Experiment

Basic Mesh Editing



(a) (b) (c)



(a) (b) (c)

Experiment

Coating transfer:

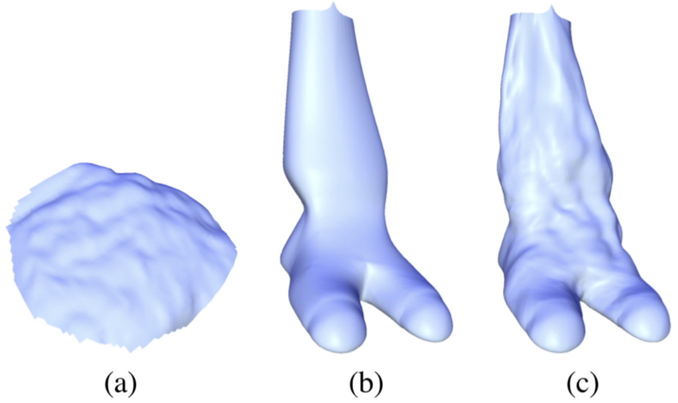


Figure 4: *Coating transfer; The coating of the Bunny (a) is transferred onto the mammal's leg (b) to yield (c).*

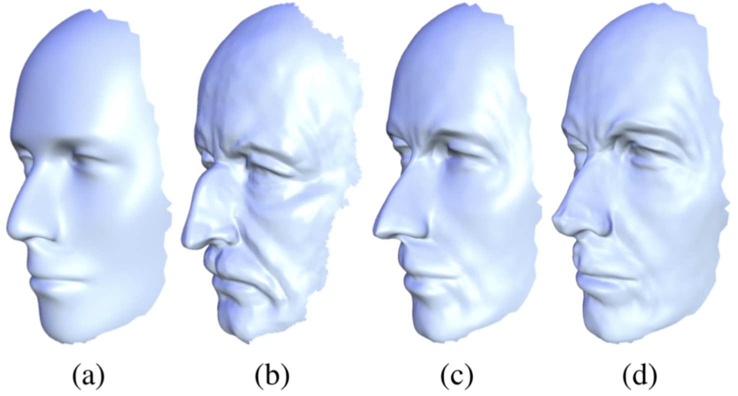


Figure 5: *The coating of the Max Planck is transferred onto the Mannequin. Different levels of smoothing were applied to the Max Planck model to peel the coating, yielding the results in (c) and (d).*

Coating transfer:

Let δ_i and $\tilde{\delta}_i$ be the Laplacian coordinates of the vertex i in the original surface and the same surface after smoothed (low-frequency surface). Then we can get the encoding of the coating at vertex i:

$$\xi_i = \delta_i - \tilde{\delta}_i$$

Coating transfer:

Assume that surface S and surface U share the same connectivity. Then, the coating transfer from surface S onto surface U is expressed as follows where Δ denotes the Laplacian coordinates of the vertices of U:

$$U' = L^{-1}(\Delta + \xi')$$

Mapping

To do the coating transfer between arbitrary surfaces with different connectivity, we need to define a mapping between the two surfaces. This mapping is established by parameterizing the meshes over a common domain.

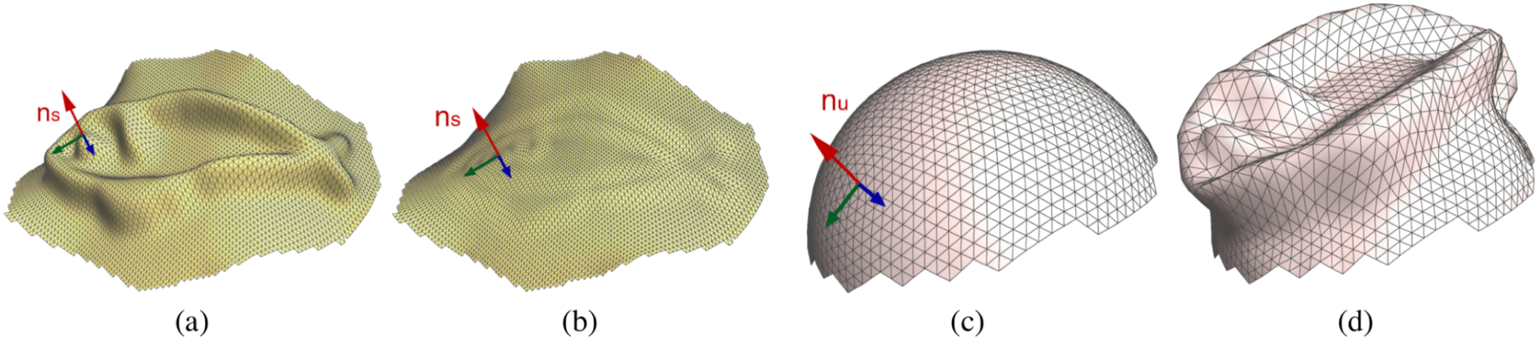
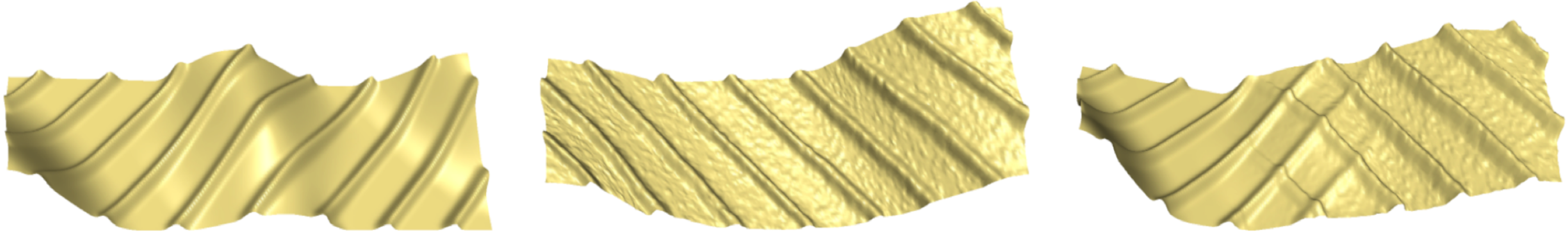


Figure 6: *Coating transfer. The orientation of a coating detail (a) is defined by the local frame at the corresponding vertex in the low frequency surface in (b). The transferred coating vector needs to be rotated to match the orientation of the corresponding point in (c) to reconstruct (d).*

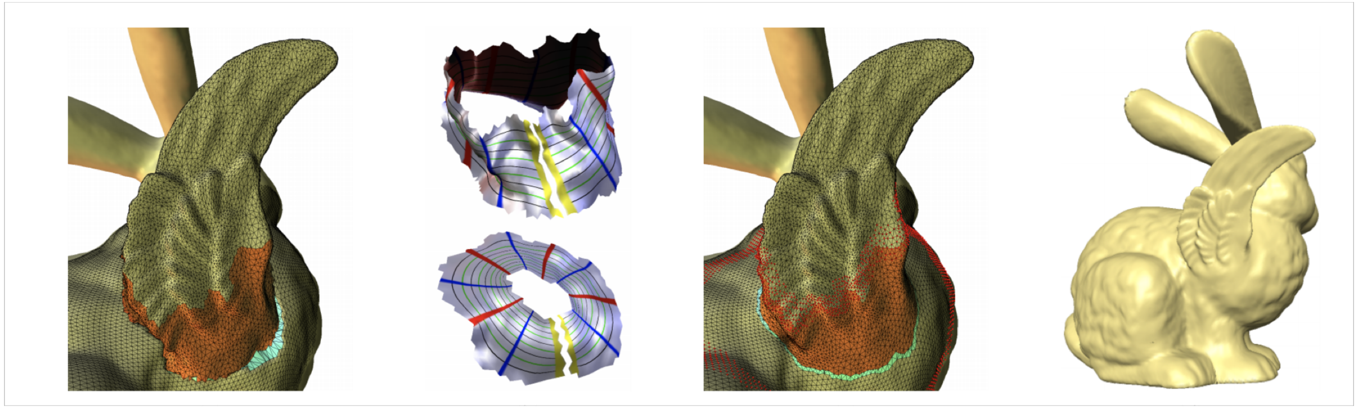
Mixed Details



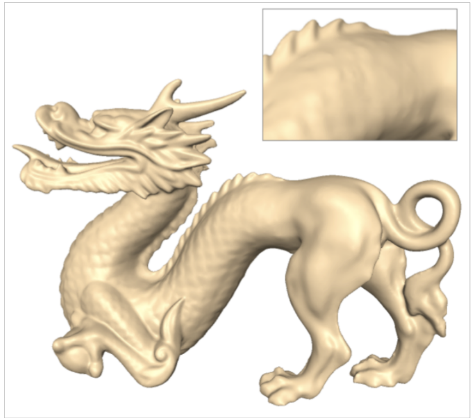
(a) (b) (c)

Figure 8: *Mixing details using Laplacian coordinates. The Laplacian coordinates of surfaces in (a) and (b) are linearly blended in the middle to yield the shape in (c).*

Transplanting Surface Patching



- Find the transitional regions
- Create the mapping
- Interpolate the Laplacian coordinates



The Laplacian in RL: Learning Representations with Efficient Approximations

Yifan Wu* Carnegie Mellon University yw4@cs.cmu.edu

George Tucker Google Brain gjt@google.com

Ofir Nachum Google Brain ofirnachum@google.com

Reinforcement Learning

- Discrete time frame t
- State of environment $s_t \in S$
- Agent take action $a_t \in A$ according to policy $P(a_t|s_t) := \pi(a_t|s_t)$
- Environment give agent reward $R_t(s_t, a_t)$
- Environment state change to $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$

Goal: Learn policy maximize the accumulated reward

$$\sum_t \gamma^t R_t$$

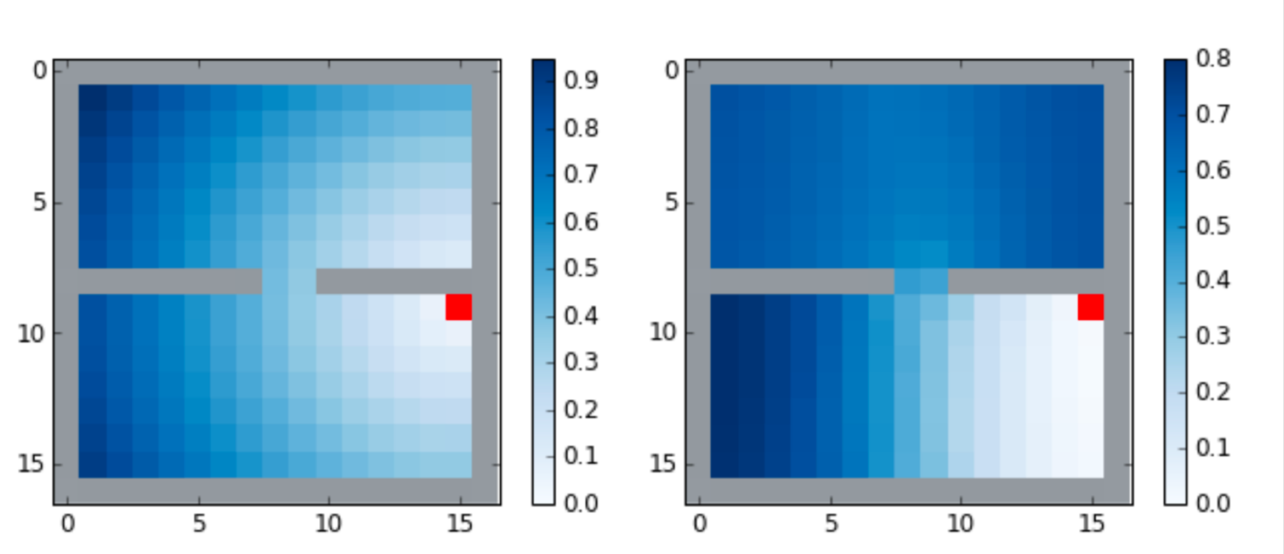
γ is discounted factor

- Often maintain replay buffer $B = [s_t, a_t, r_t, s_{t+1}]_{t=1,..,N}$

Representation in Reinforcement Learning

Performance of ML algorithm relies on the data representation

Natural question:

- How to get a better task-orientation representation in RL
- 

Reward map defined by two kind of distance to the red target

Theoretical framework

- A finite state space set S with $|S|$ elements
- Probability measure ρ distributed over S
- Hilbert space $\mathcal{H}$, for which element f are function $f: S \to \mathbb{R}$
- Linear operator $A: \mathcal{H} \to \mathcal{H}$
- Inner product: $\langle f, g \rangle_H := \int_S f(u) g(u) d\rho(u)$
- Then it define a complete Hilbert space

Definition of Laplacian

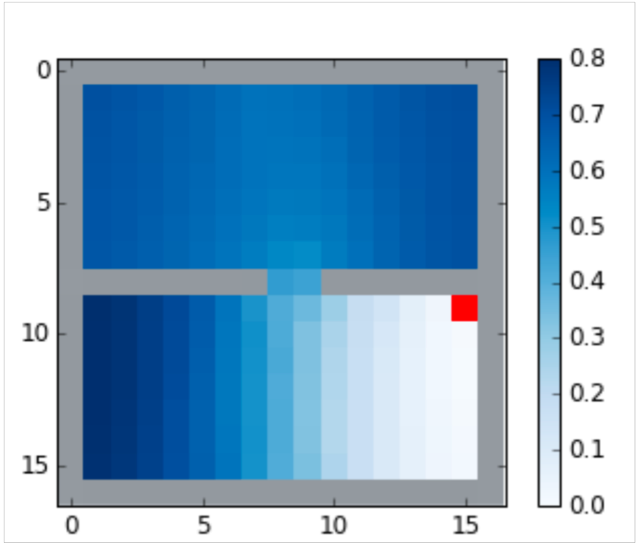
- Self-adjoint linear operator A: $\langle f, Ag\rangle_H = \langle Af, g\rangle_H$
- Self-adjoint affinity: $D : S \times S \to R^+$
- Linear operator A: $Af(u) := \int_S f(v)D(u,v)d\rho(v)$
- Graph Laplacian L: $Lf(u) = f(u) - Af(u) = (\mathrm{I} - \mathrm{A})f(\mathrm{u})$
- **Goal:** find d eigenfunction of the smallest d eigen value

Back to Maze Problem

- Reward: reach red target -> positive reward; otherwise 0
- If state representation encode the real distance

Policy will simply be:

  - Minimize the distance to target for each step



Intuition:

  - Laplacian keep info of real distance

Graph Formulation in RL

- Given policy $\pi, P(s_{t+1}|s_t, a_t) \rightarrow P^{\pi}(s_{t+1}|s_t)$
- Transitional distribution: $P^{\pi}(u|v)$

- $\rho(u) = \sum_S P^{\pi}(u|v)\rho(v)$ or $\rho(U) = \int_S P^{\pi}(u|v)d\rho(v)$
- Discrete: $D(u, v) = \frac{1}{2}\frac{P^{\pi}(v|u)}{\rho(v)} + \frac{1}{2}\frac{P^{\pi}(u|v)}{\rho(u)}$
- **Next:** Find smallest eigenfunction (eigen vector) by optimization

d-dim embedding: $\phi(u) = [f_1(u), \dots, f_d(u)]$

Spectral graph drawing

- Goal: Find embedding (eigen vector) preserve affinity
- Objective G:

$$G = \sum_k \langle f_k, Lf_k \rangle_H$$

$$= \frac{1}{2} \int_S \int_S \sum_k \left(f_k(u) - f_k(v)\right)^2 D(u,v) d\rho(u) d\rho(v)$$

- Intuition: Pushing the high affinity embedding closer
- Additional constrain:

$\langle f_k, f_j \rangle_H = \delta_{kj}$, impose orthonormal basis

Objective to learn

- Previous objective and constrain: Too hard in experiment!
- Solution: Using sampled expectation from buffer $[s_t, a_t, r_t, s_{t+1}]_{t=1,\dots,N}$

$$G = \frac{1}{2} E_{u \sim \rho, v \sim P^{\pi}(\cdot|u)}\left[\sum_k \left(f_k(u) - f_k(v)\right)^2\right]$$

Orthonormal constrain (loosed):

$$\mathcal{O} = \sum_{j,k} E_{u \sim \rho, v \sim \rho}\left[\left(f_j(u) f_k(u) - \delta_{jk}\right)\left(f_j(v) f_k(v) - \delta_{jk}\right)\right] < \epsilon$$

$$\phi(u) = [f_1(u), \dots, f_d(u)]$$

$$\left(\phi(u)^T \phi(v)\right)^2 - ||\phi(v)||_2^2 - ||\phi(u)||_2^2 + constant$$

Final loss = $G + \beta\, \mathcal{O}$

Small summary

- We want better state representation to preserve affinity
- It is known that Laplacian eigenfunction is such thing
- Direct computation of eigenfunction is intractable
- Using Neural Network to approximate
- Build a objective function
- Optimized NN with the derived loss

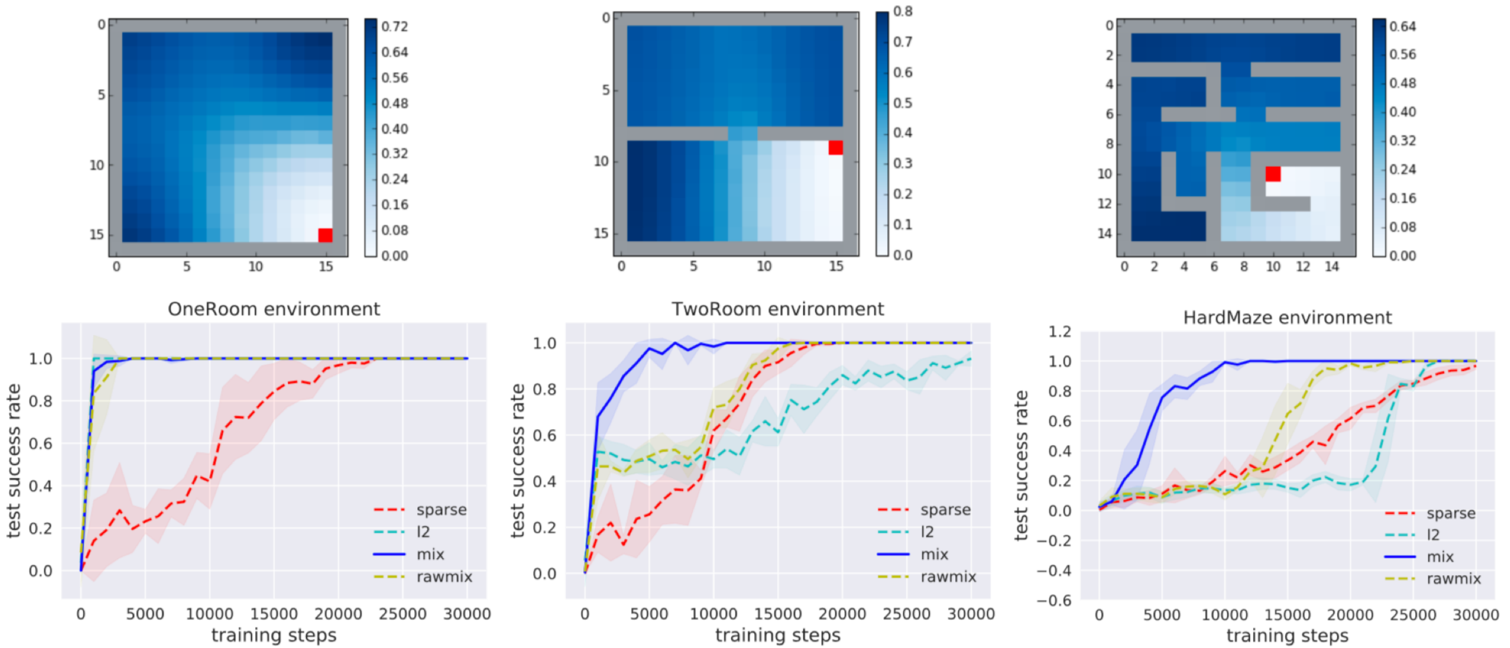
Experiment

1. Collect trajectory with model-free policy (random) for buffer
2. Learning state embedding based on objective
3. Evaluate the embeddings: (maze environment)

    - Goal achieving tasks: rewarded for reaching a given goal state z_g
    - Two type of reward: sparse reward and shaped reward
    - Shaped reward: $r_t = -||\phi(s_{t+1}) - \phi(z_g))||$

Experiment

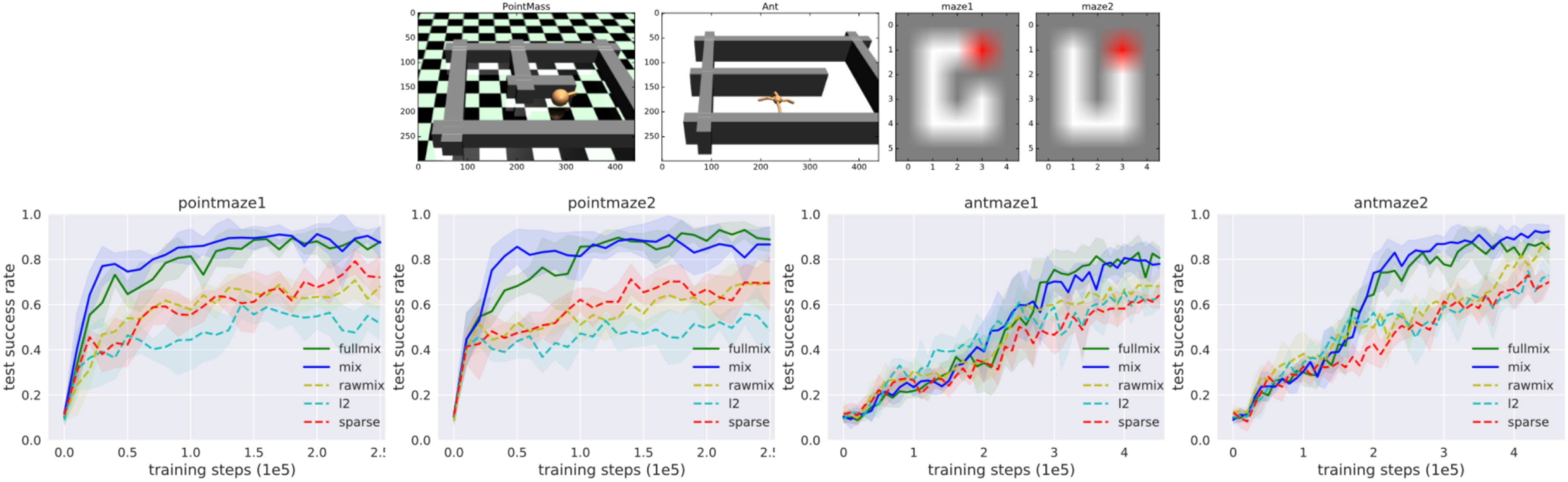
RL algorithm: DQN[1]



1. Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529.

Experiment on continues state

RL algorithm: DDPG[1]



1. Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015).

Summary

- Laplacian: both map and operator
- Laplacian induced representation can encode local structure
- Eigenfunction/eigenvalue of L can be good basis

Thank you and happy valentine day !