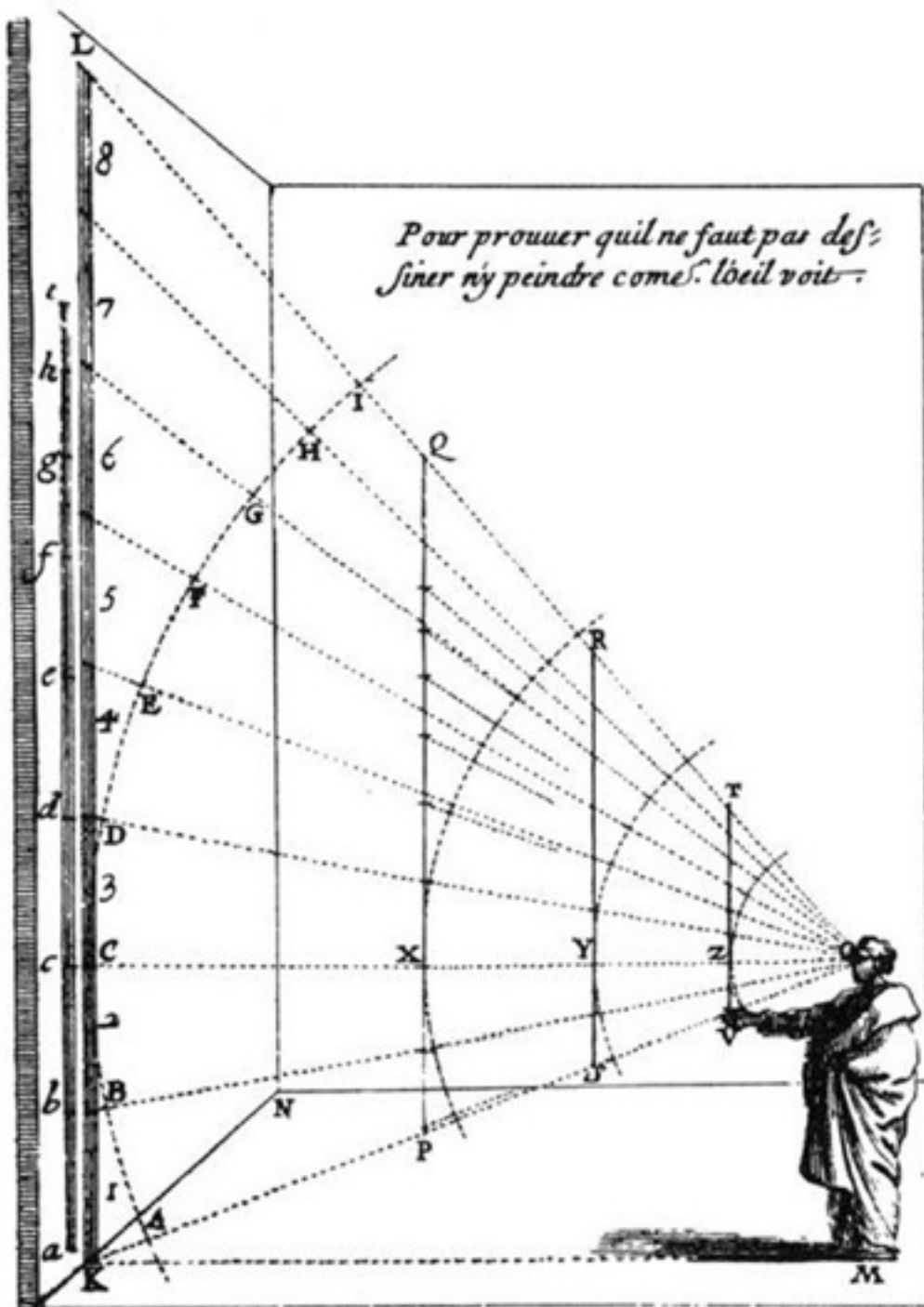


Sensorimotor Embedding

A developmental approach to learning geometry

Jeremy Stober

Final Defense
May 6th 2015



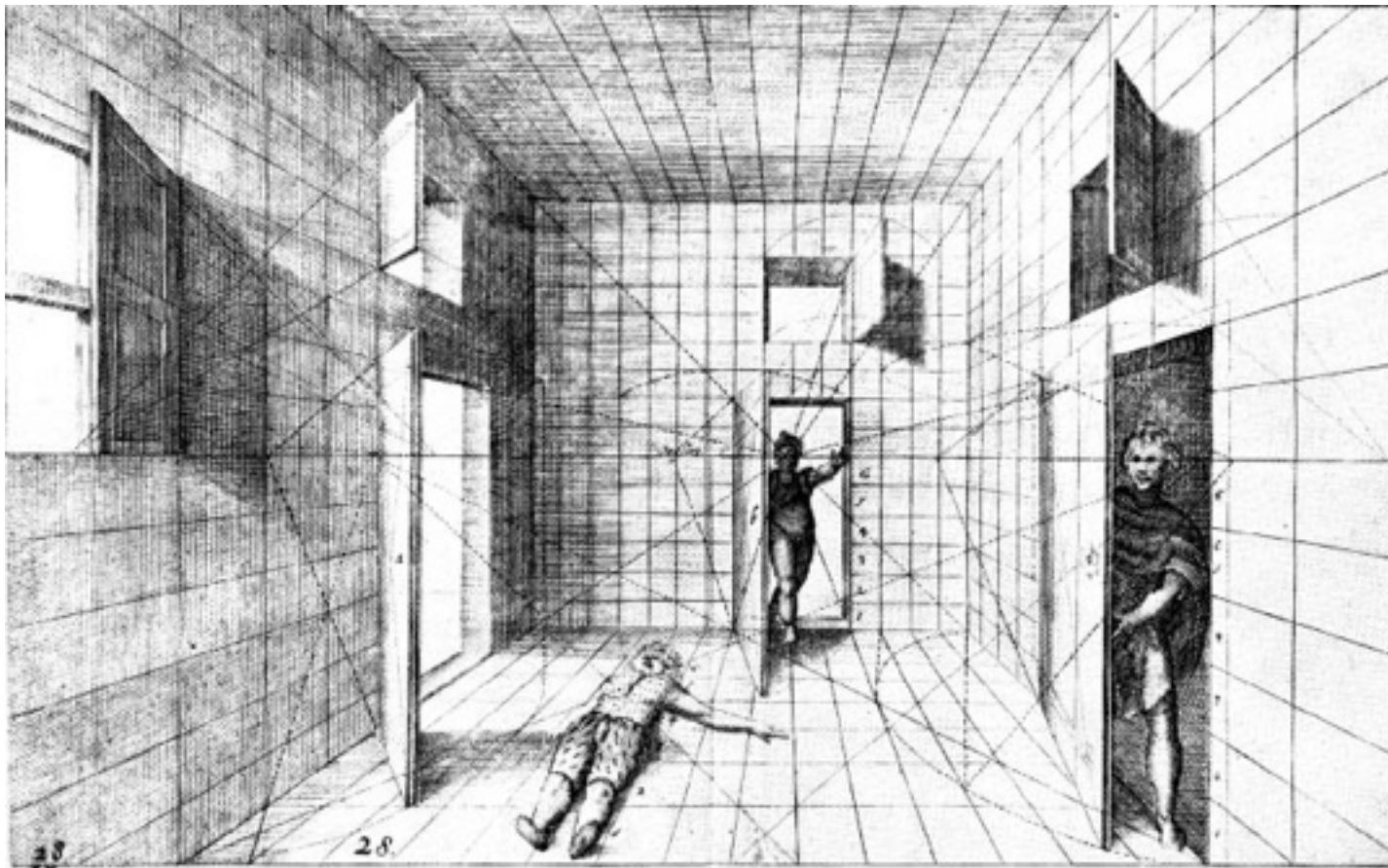
To prove that one can neither define nor paint as the eye sees
—Abraham Bosse, 1665

Outline

- Human Geometric Knowledge
- The Problem - Autonomously Learning Geometry
- A Review of Manifold Learning and Related Work
- A Description of Sensorimotor Embedding
- Applications of Sensorimotor Embedding
- Future Work and Conclusions



Human Geometric Knowledge

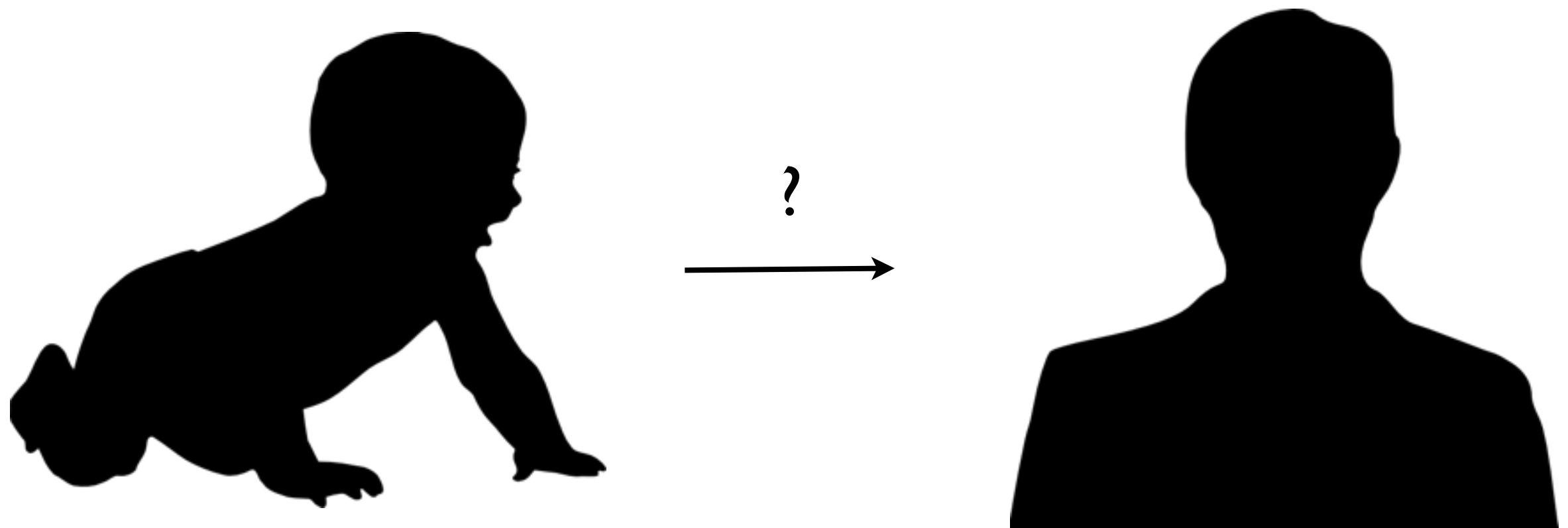


Jan Vredeman de Vries
Perspective, 1604

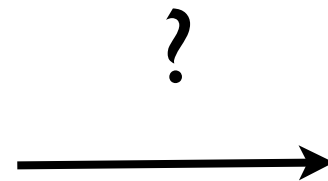


Eric Conklin
A Perspective Box with Views of a Vaulted Vestibule, 2003

Humans intuitively understand the dimension and geometry of space.

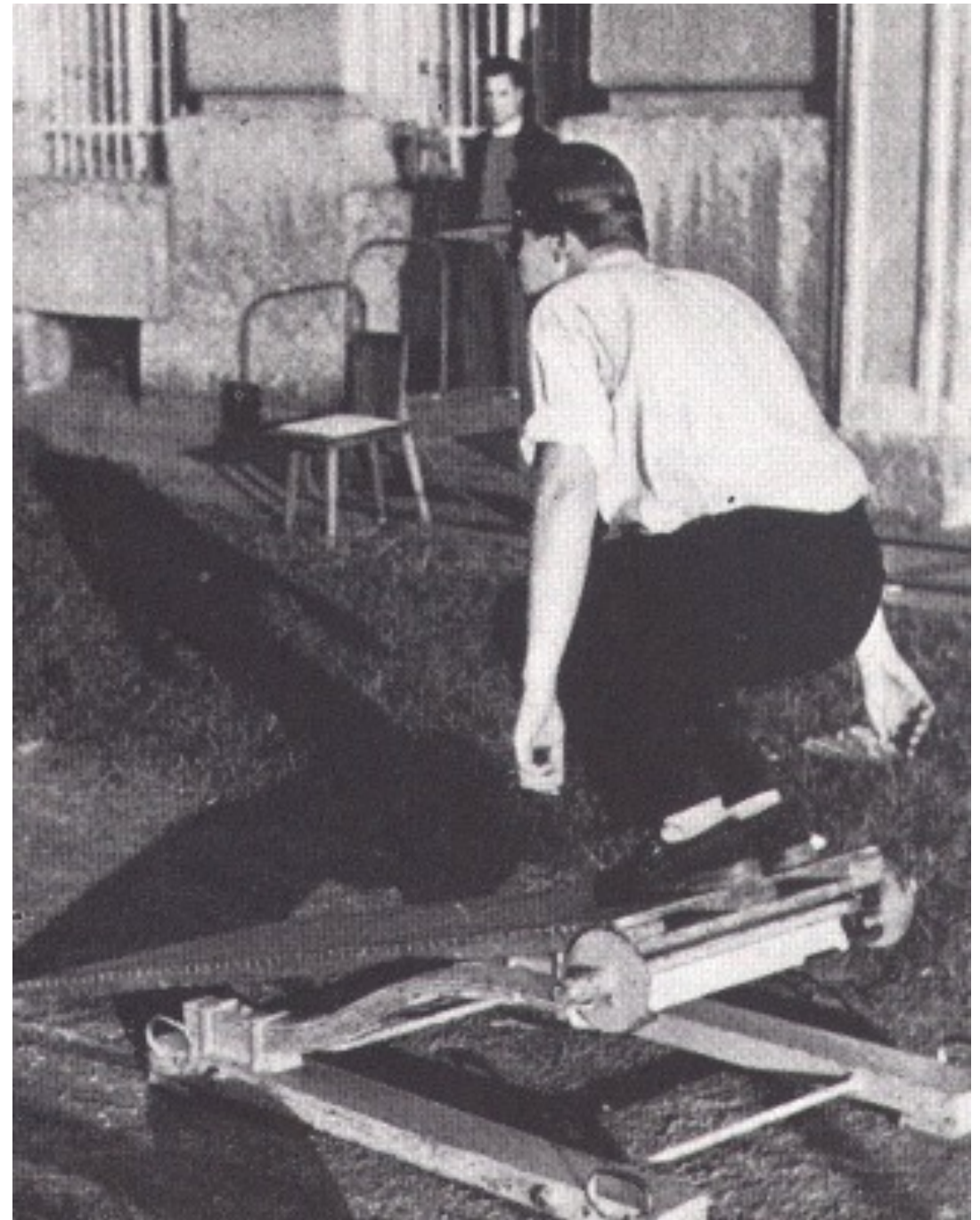


At some point during development, we acquire this knowledge of the dimension and geometry of space.

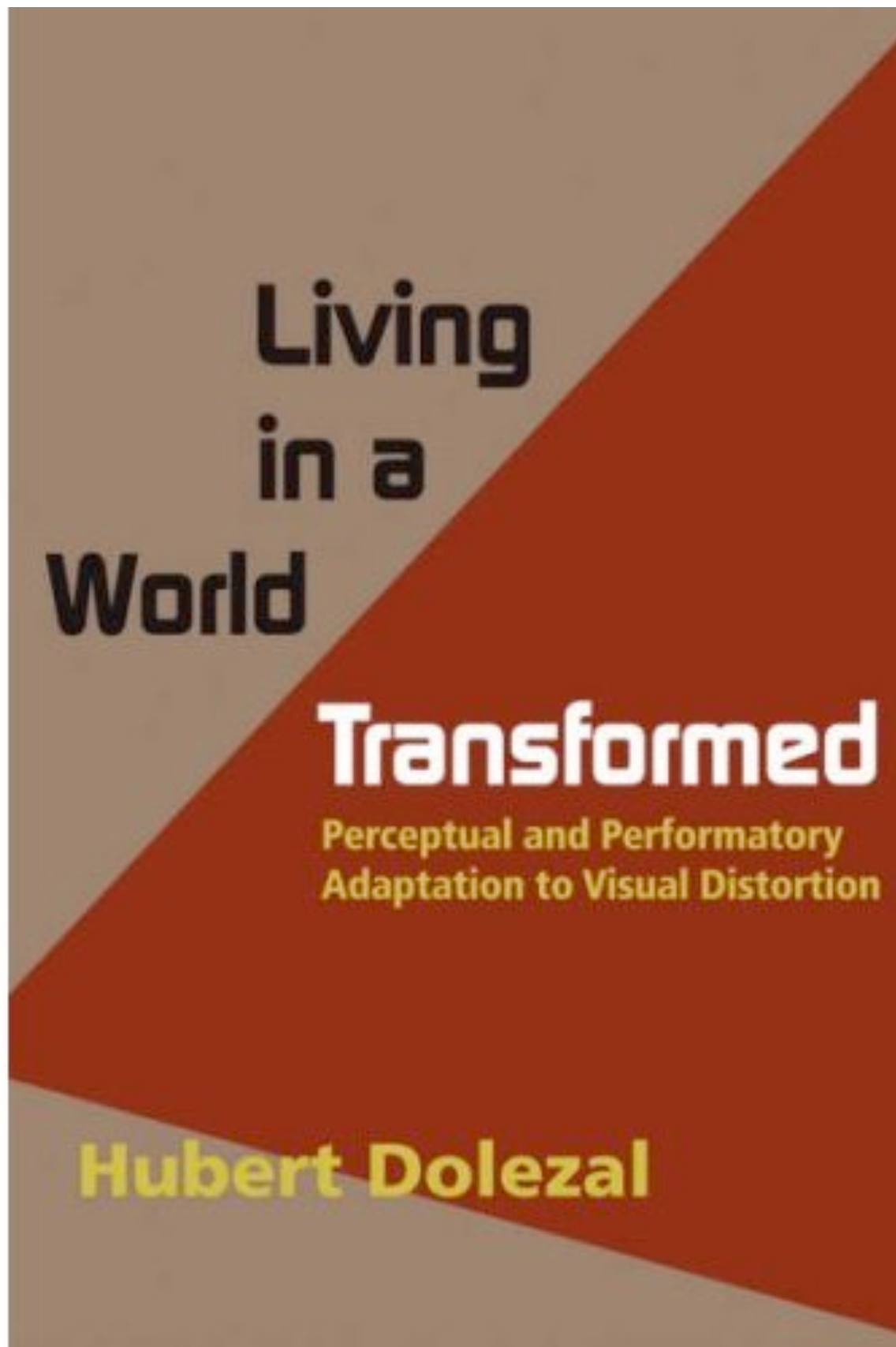


At some point during development, we acquire this knowledge of the dimension and geometry of space.

Humans are tremendously adaptable when confronted with sensory change even as adults.



Erismann (1930)



For example, humans can learn to ride scooters with vision inverting goggles.



Learning Geometry

- The agent is interested in external world geometric properties.
- Examples include robot coordinates; object pose; sensor location.
- **How can agents learn corresponding internal representations of these geometric properties?**

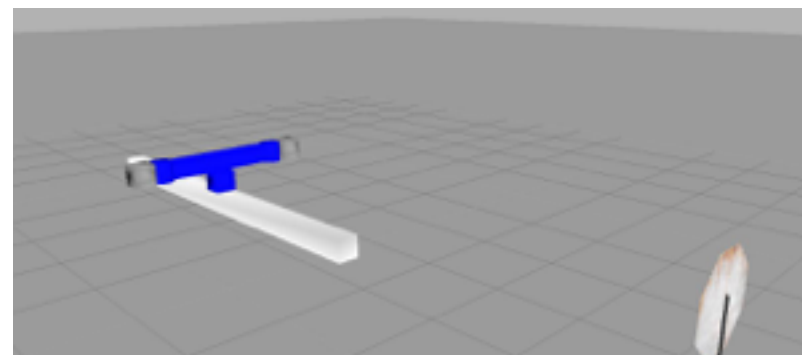
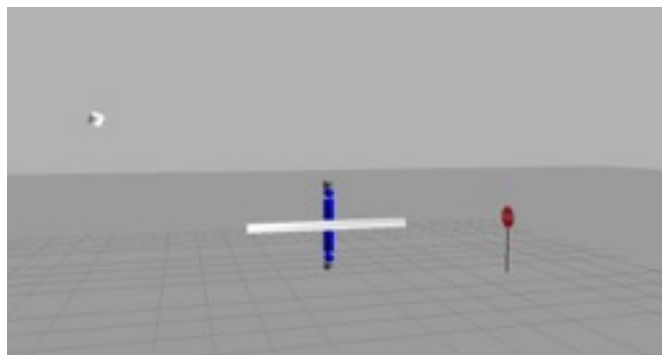
An agent that can autonomously learn geometry should be able to



adapt to
lesioning,



adapt to image
inversion,



run on different
robots.

Manifold Learning

Manifold Learning

- For a set of high-dimensional points, find a set of representative low-dimensional points.
- For example, the low-dimensional points should have the same inter-point distances.

Let $\{x_1, \dots, x_k\}$ where $x_i \in \mathbb{R}^n$

find $\{y_1, \dots, y_k\}$ where $y_i \in \mathbb{R}^m$ and $m \ll n$

such that $\|x_i - x_j\| \approx \|y_i - y_j\|$.

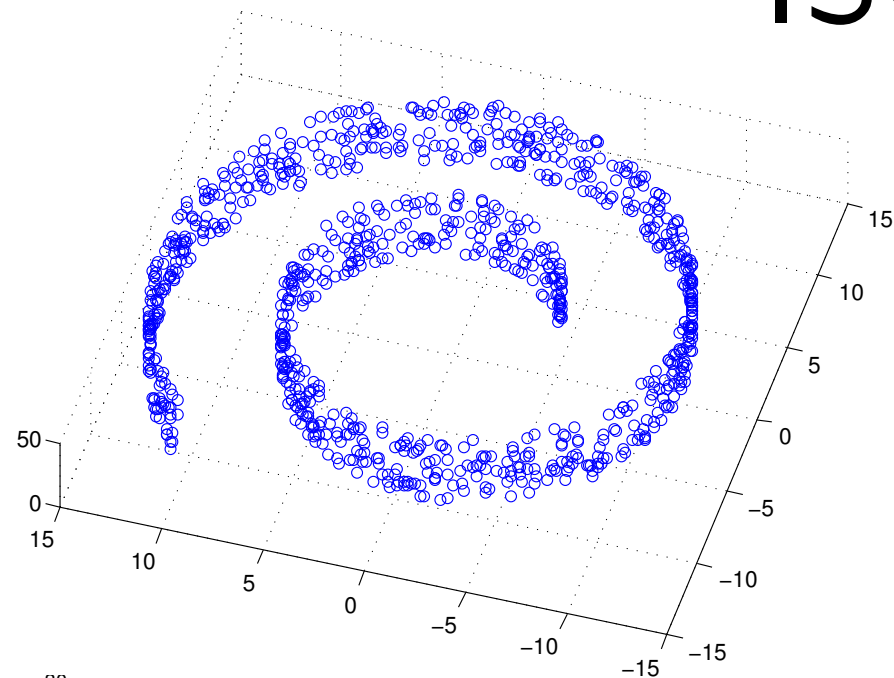
Multidimensional Scaling (MDS)

Generate a matrix of inter-point distances Δ
and transform $\Delta \rightarrow K$ where K is a Gram Matrix,
e.g. a matrix of dot products between all the original points.
Decompose K .

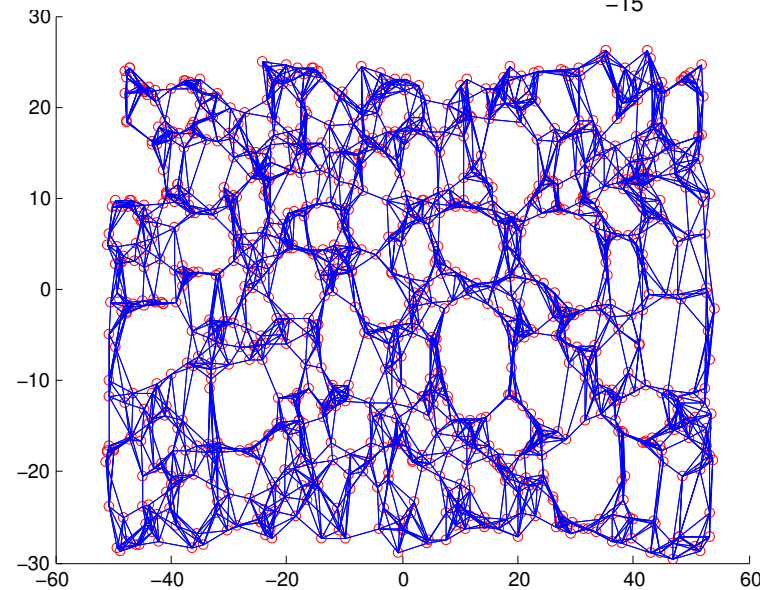
The highest weighted components represent the data.

Different methods of constructing Δ result in new algorithms for manifold learning.

Isomap



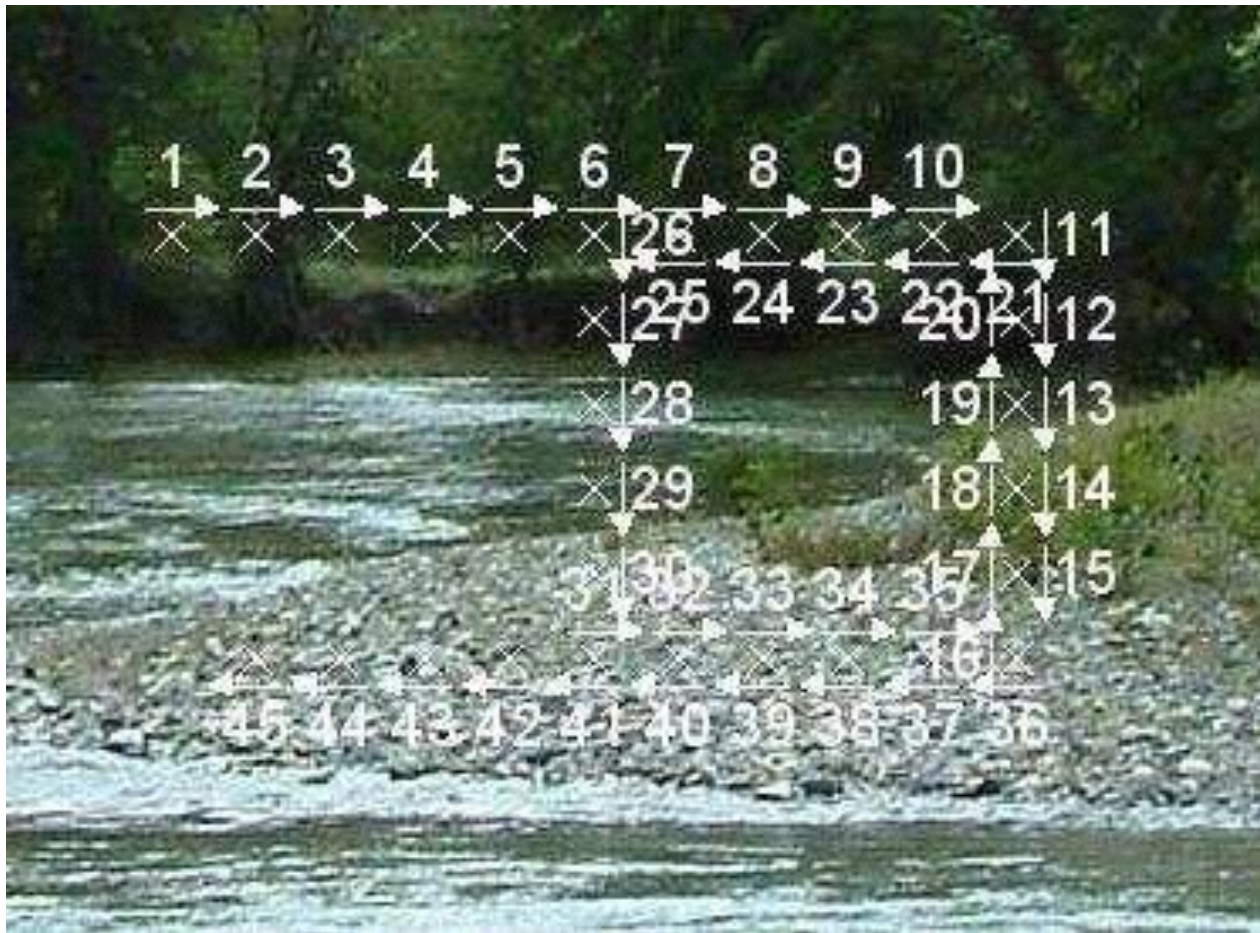
A local graph between point is constructed using nearest neighbors.



For distant points, the distance is computed as the shortest path along the graph.

Use distances across the manifolds surface to construct: Δ_{iso}

Action Respecting Embedding (ARE)



Roving Eye domain from
Bowling et al.

The robot infers its path around the image by applying action respecting embedding to the images it takes along the path.

Find Δ_{are} as the solution to a optimization problem that maximizes variance subject to action constraints.

Sensorimotor Embedding

Like Isomap and Action Respecting Embedding, Sensorimotor Embedding constructs a new kind of distance matrix based on action traces produced by policies:

$$\Delta_{\pi}$$

Sensorimotor Embedding

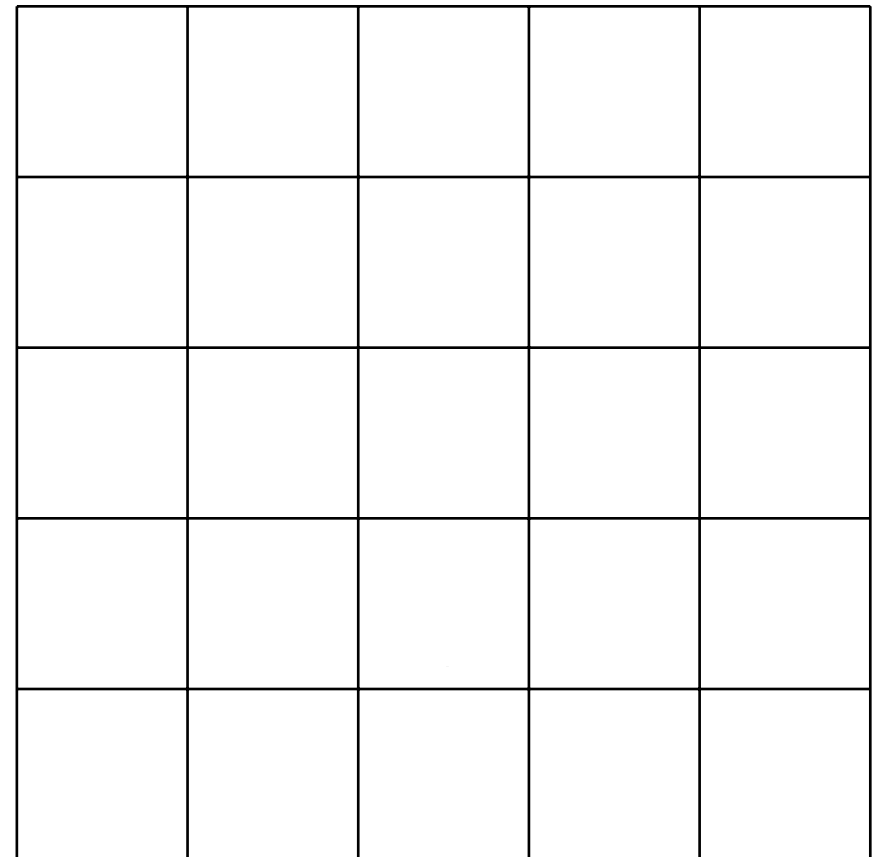
Sensorimotor embedding is a manifold learning algorithm that takes high dimensional sensorimotor experience as input and outputs geometric features.

Sensorimotor Embedding Algorithm

1. Learn a policy to acquire a perceptual goal.
2. Apply the policy from different start states and record the resulting action traces.
3. Generate a matrix of distances by comparing action traces.
4. Apply multidimensional scaling to the matrix of action trace distances to generate low dimensional points associated with start states.

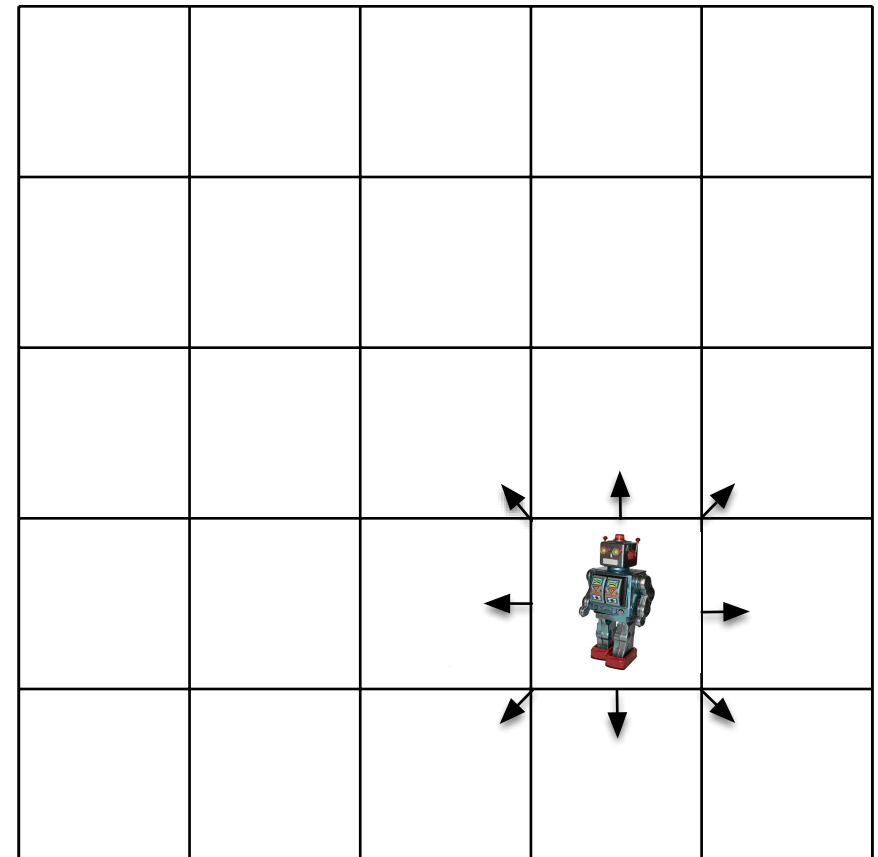
Example

- This example is based on a grid world.



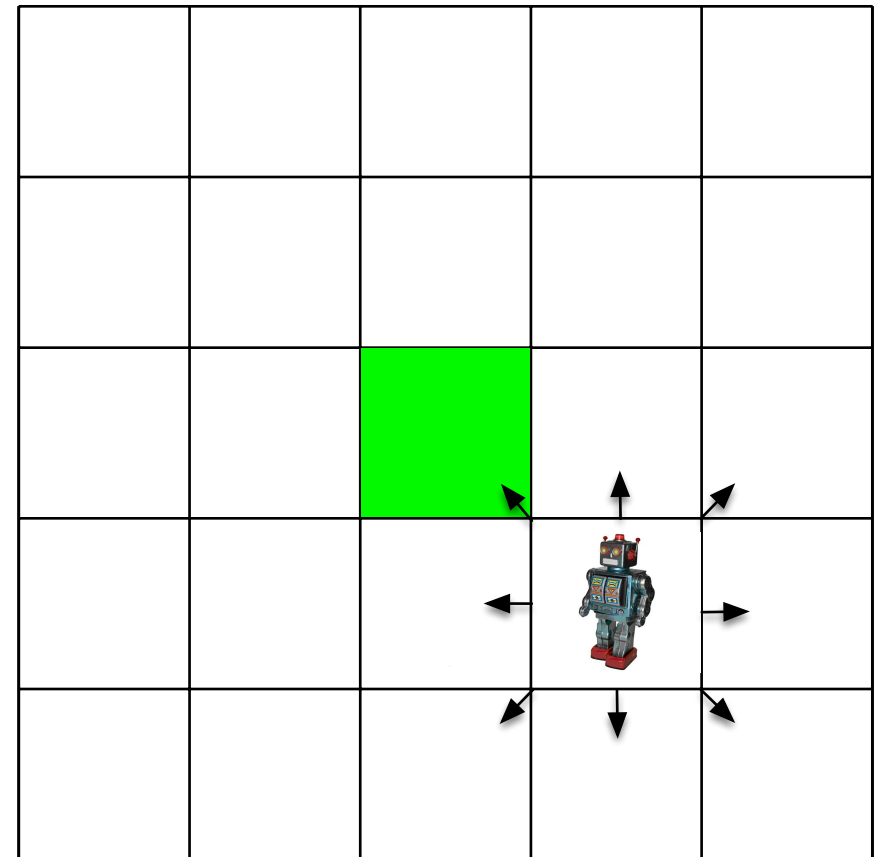
Example

- This example is based on a grid world.
- The agent can move to any of eight adjacent states.



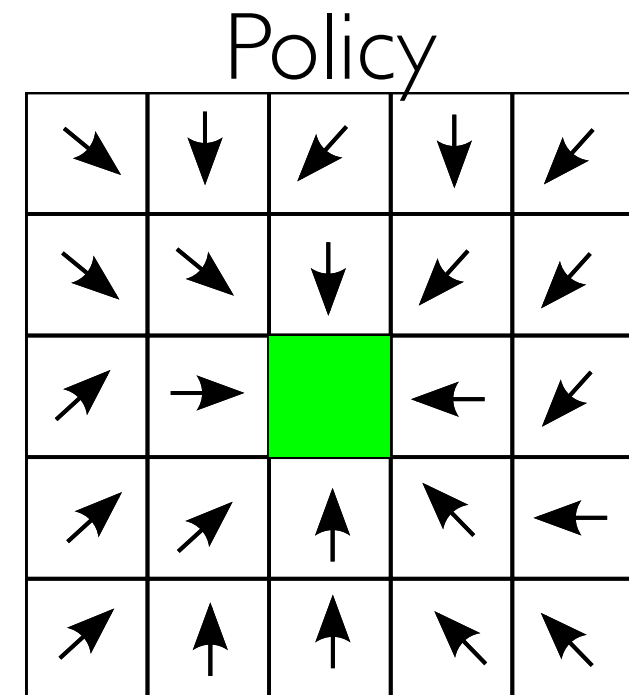
Example

- This example is based on a grid world.
- The agent can move to any of eight adjacent states.
- The agent's goal is to reach the center state.



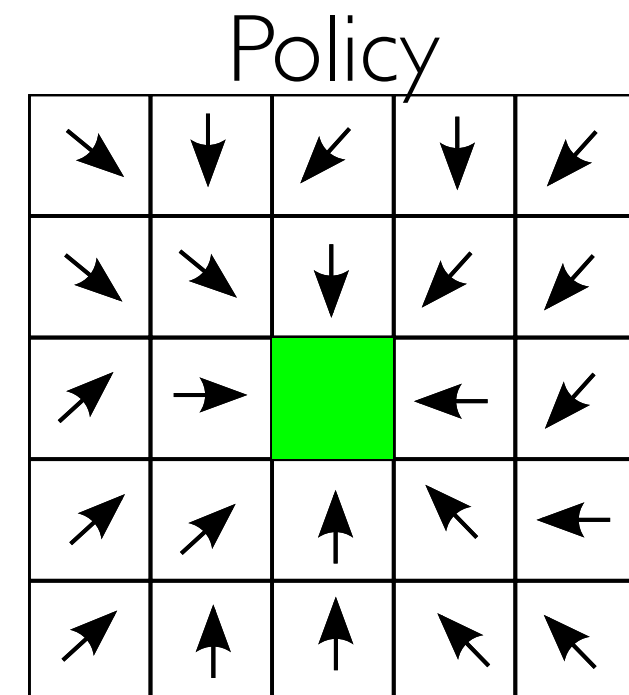
Step 1

- Learn a policy to achieve a perceptual goal.
- A policy is a function that chooses an action for each state.



Step 1

- Learn a policy to achieve a perceptual goal.
- A policy is a function that chooses an action for each state.



Step 2

- Apply the policy from different start states and record the resulting action traces.

State Labels

0	5	10	15	20
1	6	11	16	21
2	7	12	17	22
3	8	13	18	23
4	9	14	19	24






Policy

↘	↓	↙	↓	↘
↘	↘	↓	↙	↘
↗	→		←	↘
↗	↗	↑	↖	←
↗	↑	↑	↖	↖

Step 2

- Apply the policy from different start states and record the resulting action traces.

























Action Traces

0		
1		
...		
11		
...		

State Labels

0	5	10	15	20
1	6	11	16	21
2	7	12	17	22
3	8	13	18	23
4	9	14	19	24






Policy



Step 3

- Generate matrix of distances by comparing action traces.
- Parameterize actions using unit vectors in eight cardinal directions.


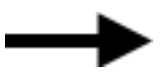

Action Traces

0		
1		
...		
11		
...		

Action Parameters

	(0.71,-0.71)
	(1,0)
...	...

Comparing Action Traces

1		
11		

- Compare the parameterized action traces using a sequence metric:

$$\delta_{\text{trace}}(\{a\}_1^n, \{b\}_1^m) = \sum_{t=1}^m d(a_t, b_t) + \sum_{t=m+1}^n d(a_t, 0).$$

- These distances are collected in a matrix of distances.

Comparing Action Traces

1	(0.71,-0.71)	(1,0)
11	(-1,0)	

- Compare the parameterized action traces using a sequence metric:

$$\delta_{\text{trace}}(\{a\}_1^n, \{b\}_1^m) = \sum_{t=1}^m d(a_t, b_t) + \sum_{t=m+1}^n d(a_t, 0).$$

- These distances are collected in a matrix of distances.

Comparing Action Traces

$$\delta_{\text{trace}}(\{\searrow, \rightarrow\}, \{\downarrow\})$$

- Compare the parameterized action traces using a sequence metric:

$$\delta_{\text{trace}}(\{a\}_1^n, \{b\}_1^m) = \sum_{t=1}^m d(a_t, b_t) + \sum_{t=m+1}^n d(a_t, 0).$$

- These distances are collected in a matrix of distances.

Comparing Action Traces

$$\sqrt{(0.71 - -1)^2 + (-0.71 - 0)^2} + \sqrt{(-1 - 0)^2 + (0 - 0)^2}$$

- Compare the parameterized action traces using a sequence metric:

$$\delta_{\text{trace}}(\{a\}_1^n, \{b\}_1^m) = \sum_{t=1}^m d(a_t, b_t) + \sum_{t=m+1}^n d(a_t, 0).$$

- These distances are collected in a matrix of distances.

Comparing Action Traces

$$\delta_{\text{trace}}(\{\searrow, \rightarrow\}, \{\downarrow\}) = 2.85$$

- Compare the parameterized action traces using a sequence metric:

$$\delta_{\text{trace}}(\{a\}_1^n, \{b\}_1^m) = \sum_{t=1}^m d(a_t, b_t) + \sum_{t=m+1}^n d(a_t, 0).$$

- These distances are collected in a matrix of distances.

Comparing Action Traces

$$\delta_{\pi}(1, 11) \equiv \delta_{\text{trace}}(\{\searrow, \rightarrow\}, \{\downarrow\}) = 2.85$$

- Compare the parameterized action traces using a sequence metric:

$$\delta_{\text{trace}}(\{a\}_1^n, \{b\}_1^m) = \sum_{t=1}^m d(a_t, b_t) + \sum_{t=m+1}^n d(a_t, 0).$$

- These distances are collected in a matrix of distances.

Comparing Action Traces

$$\Delta_{\pi} = \begin{bmatrix} 0 & \delta_{\pi}(0, 1) & \cdots & \delta_{\pi}(0, 24) \\ \delta_{\pi}(1, 0) & 0 & \cdots & \delta_{\pi}(1, 24) \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{\pi}(24, 0) & \delta_{\pi}(24, 1) & \cdots & 0 \end{bmatrix}$$

- Compare the parameterized action traces using a sequence metric:

$$\delta_{\text{trace}}(\{a\}_1^n, \{b\}_1^m) = \sum_{t=1}^m d(a_t, b_t) + \sum_{t=m+1}^n d(a_t, 0).$$

- These distances are collected in a matrix of distances.

Comparing Action Traces

$$\Delta_{\pi} = \begin{bmatrix} 0 & 0.76 & \dots & 3.96 \\ 0.76 & 0 & \dots & 2.74 \\ \vdots & \vdots & \ddots & \vdots \\ 3.96 & 2.74 & \dots & 0 \end{bmatrix}$$

- Compare the parameterized action traces using a sequence metric:

$$\delta_{\text{trace}}(\{a\}_1^n, \{b\}_1^m) = \sum_{t=1}^m d(a_t, b_t) + \sum_{t=m+1}^n d(a_t, 0).$$

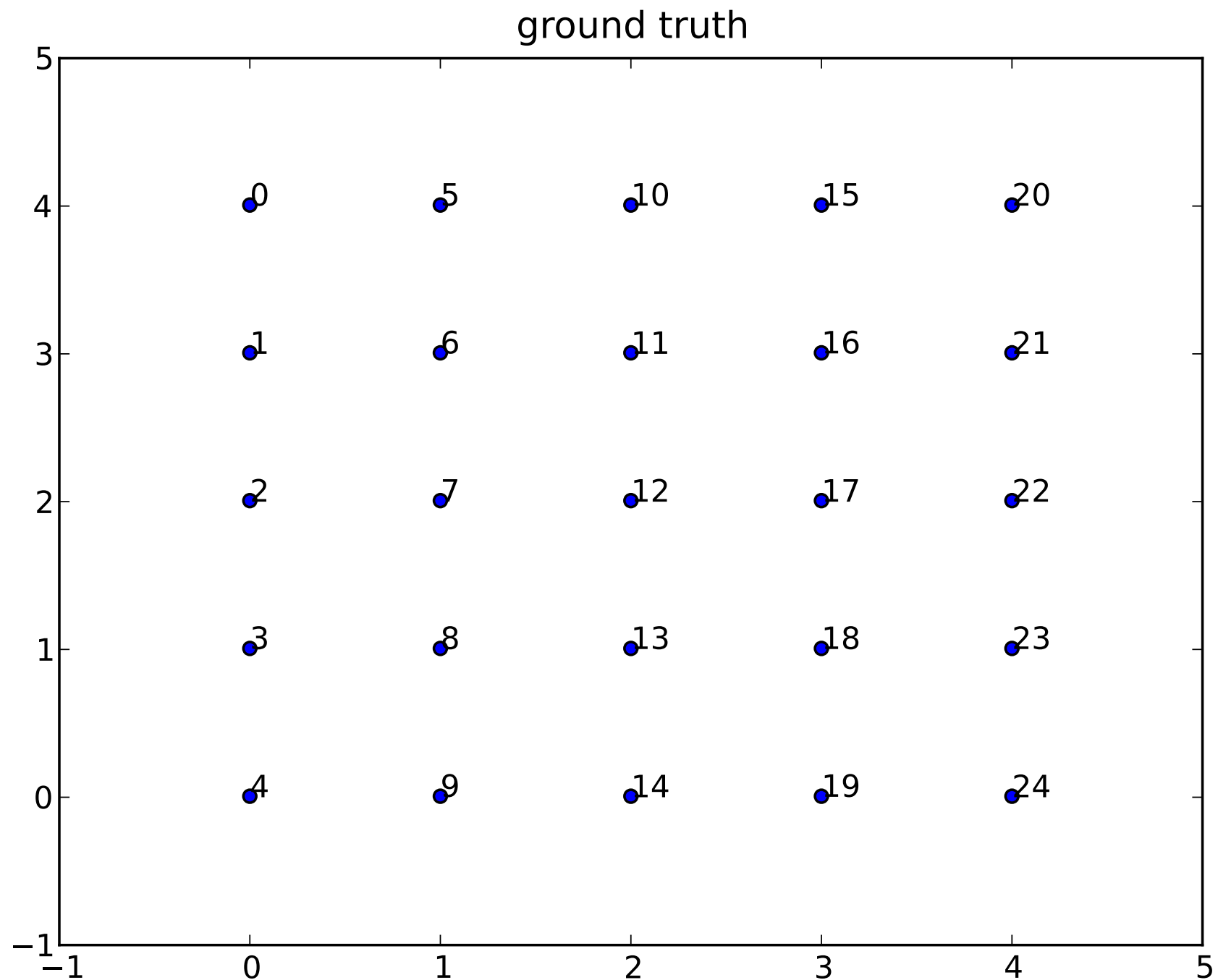
- These distances are collected in a matrix of distances.

Step 4

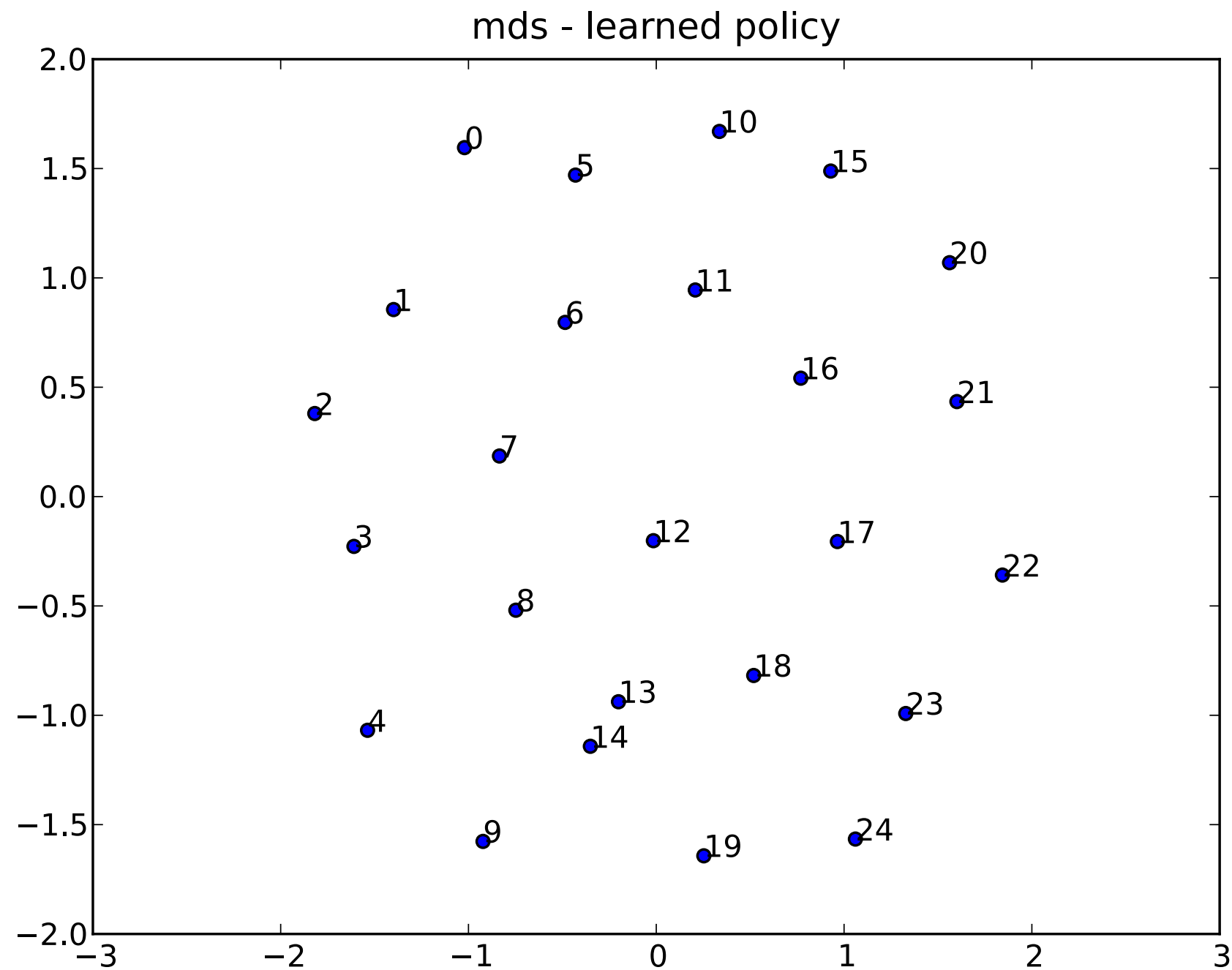
$$\begin{array}{c} \text{Sensorimotor Distances} \\ MDS \end{array} \begin{pmatrix} 0 & 0.76 & \dots & 3.96 \\ 0.76 & 0 & \dots & 2.74 \\ \vdots & \vdots & \ddots & \vdots \\ 3.96 & 2.74 & \dots & 0 \end{pmatrix} = \begin{array}{c} \text{Coordinates} \\ \begin{pmatrix} -1.1 & 1.6 \\ -1.5 & 0.9 \\ \vdots & \vdots \\ 1.1 & -1.5 \end{pmatrix} \end{array}$$

- Apply MDS to the matrix of distances to generate low-dimensional points associated with start states.

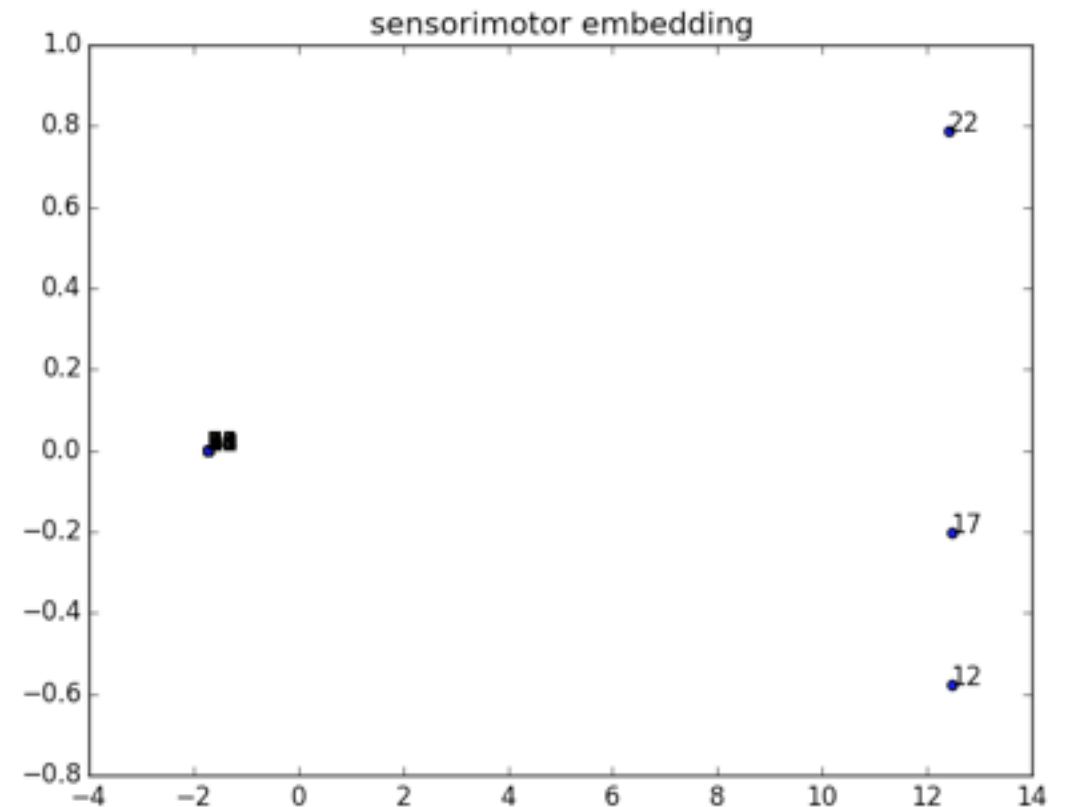
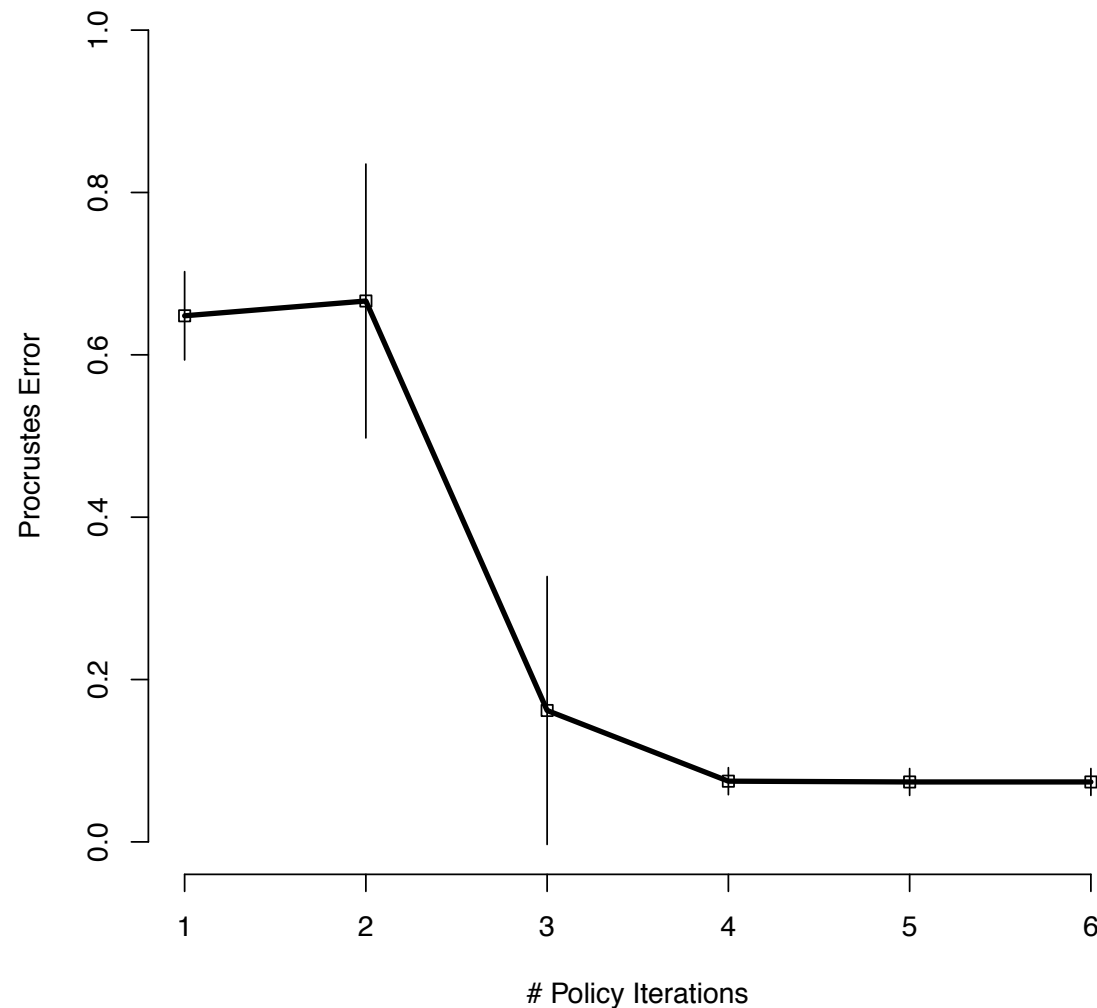
Comparison to Ground Truth



Comparison to Ground Truth

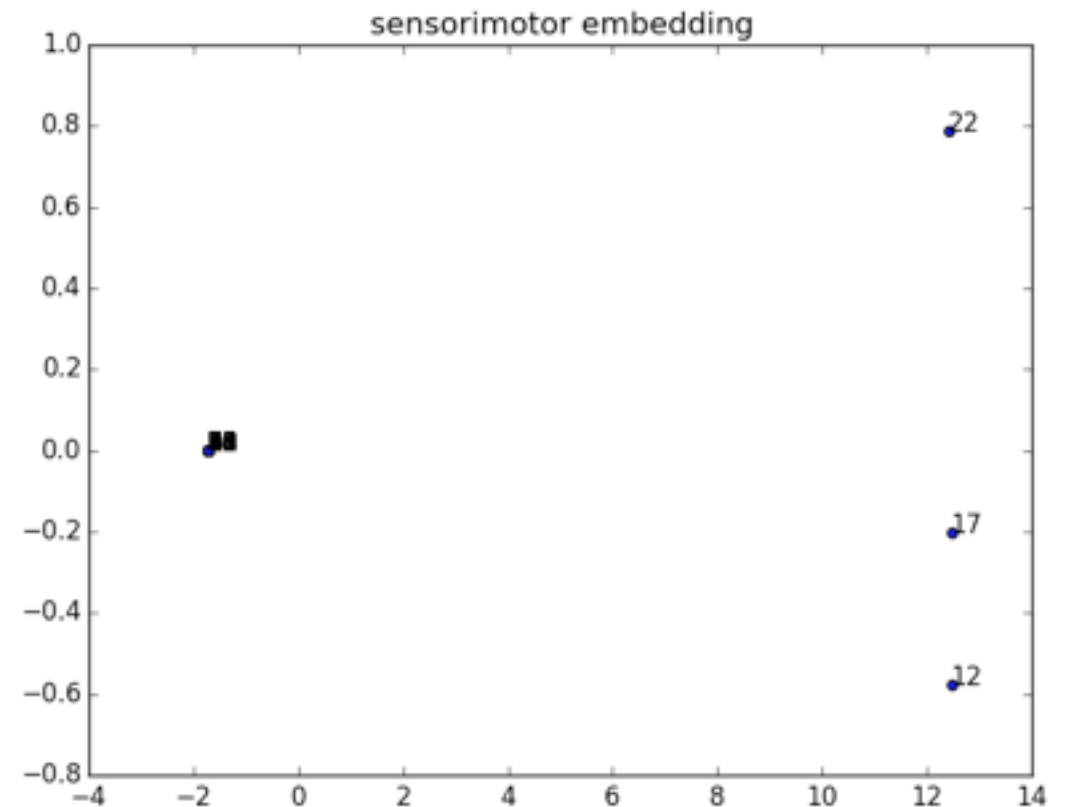
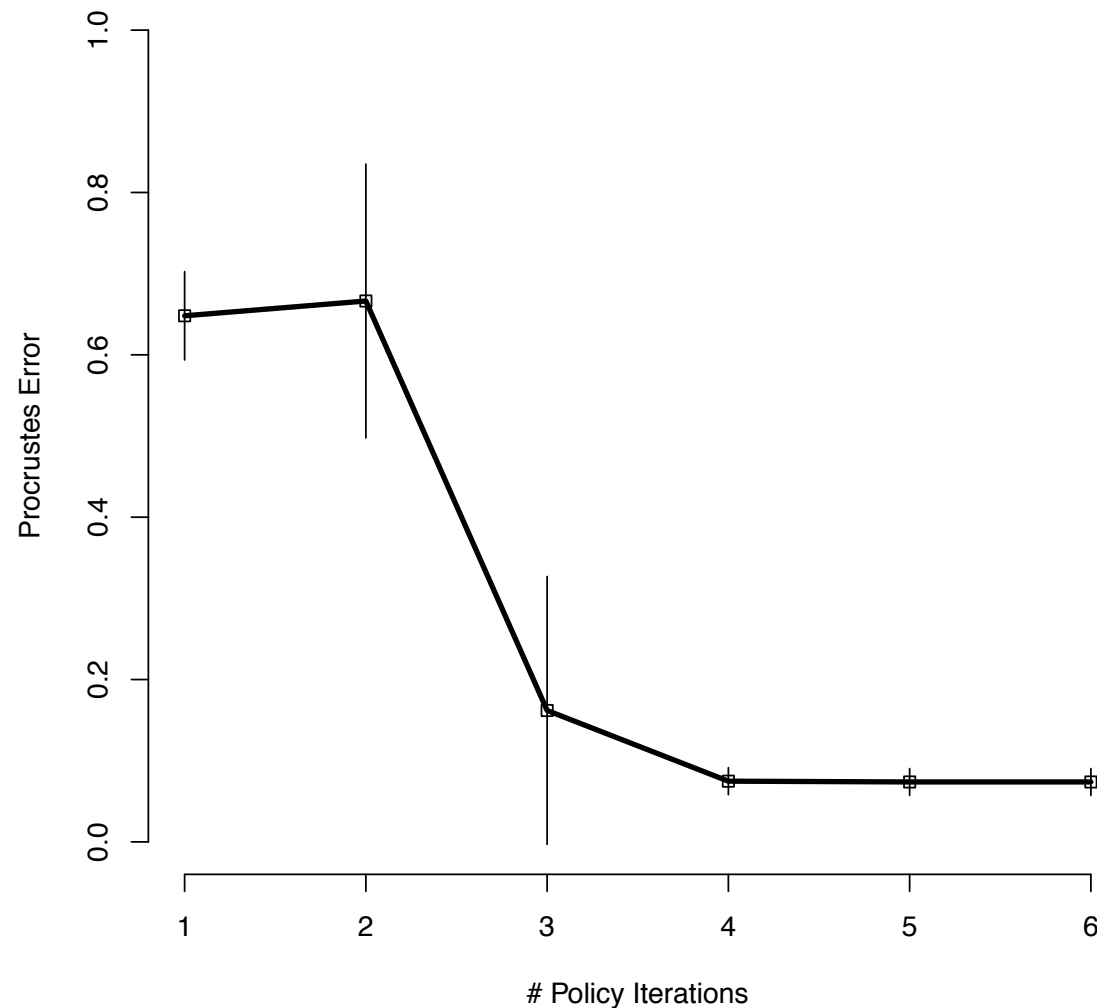


Policy and Geometry



As the policy improves, so does the learned geometry.

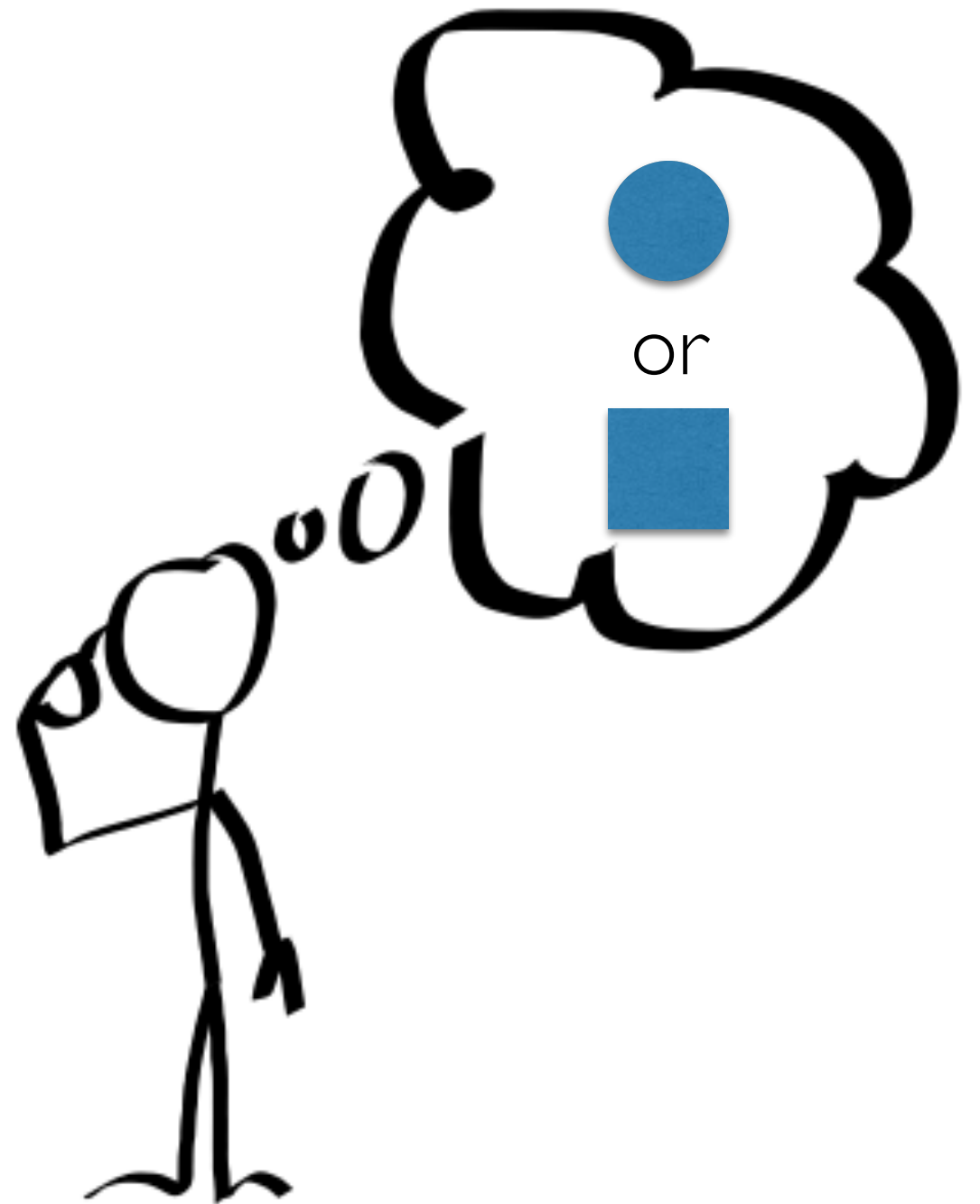
Policy and Geometry



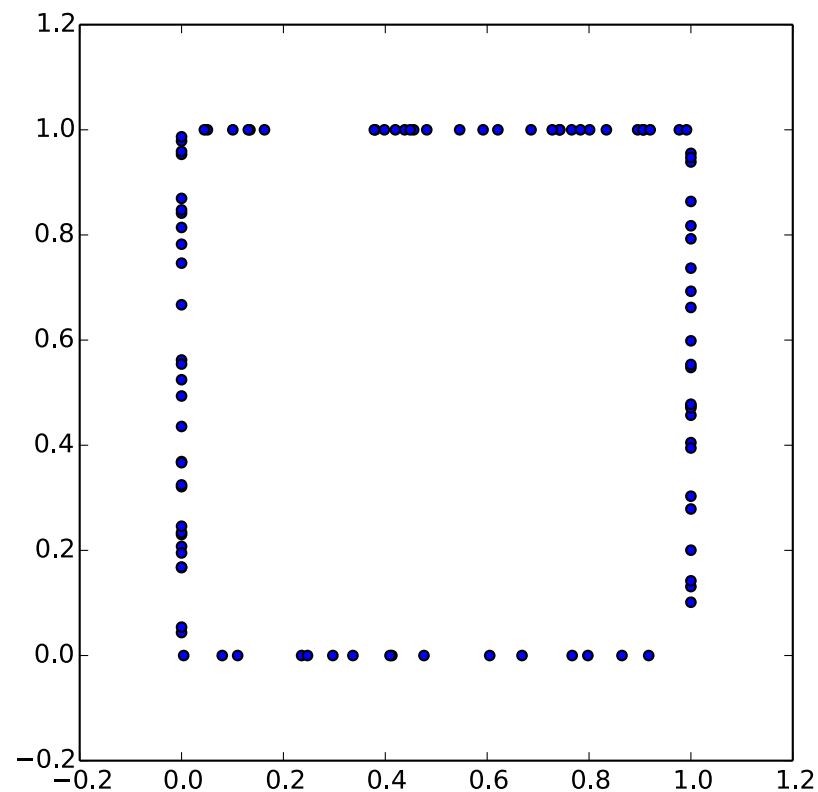
As the policy improves, so does the learned geometry.

Procrustes Error

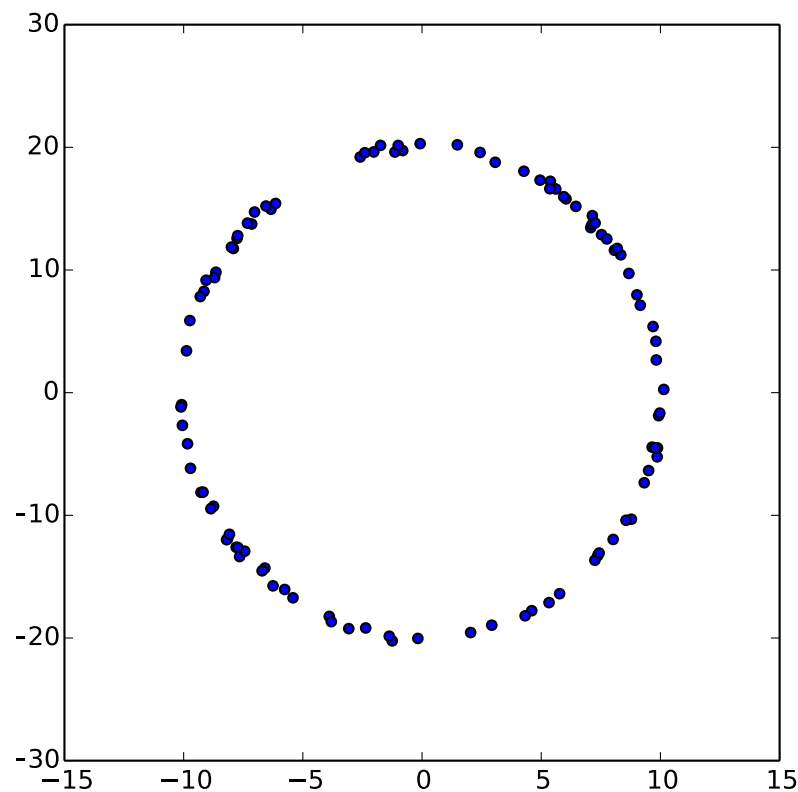
Shapes in the external world should equate to shapes in the agent's internal representation.



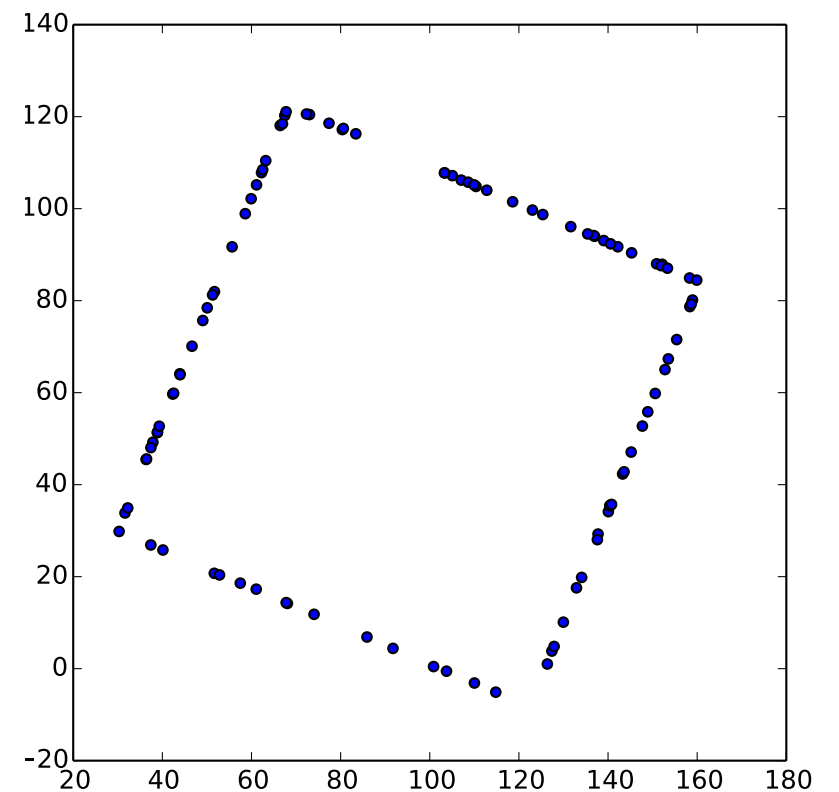
Procrustes Error



Original Points



Error 0.12



Error $6e-6$

This approach allows for changes in translation, scale, and rotation. If sampled points come from the same shape then the error will be low.

Key Observations

- Better policies result in more accurate geometry.
- After the policy is learned and traces are collected, Sensorimotor Embedding does not depend on the states.
- If the policy can adapt to change, so can the geometry.

Object Pose



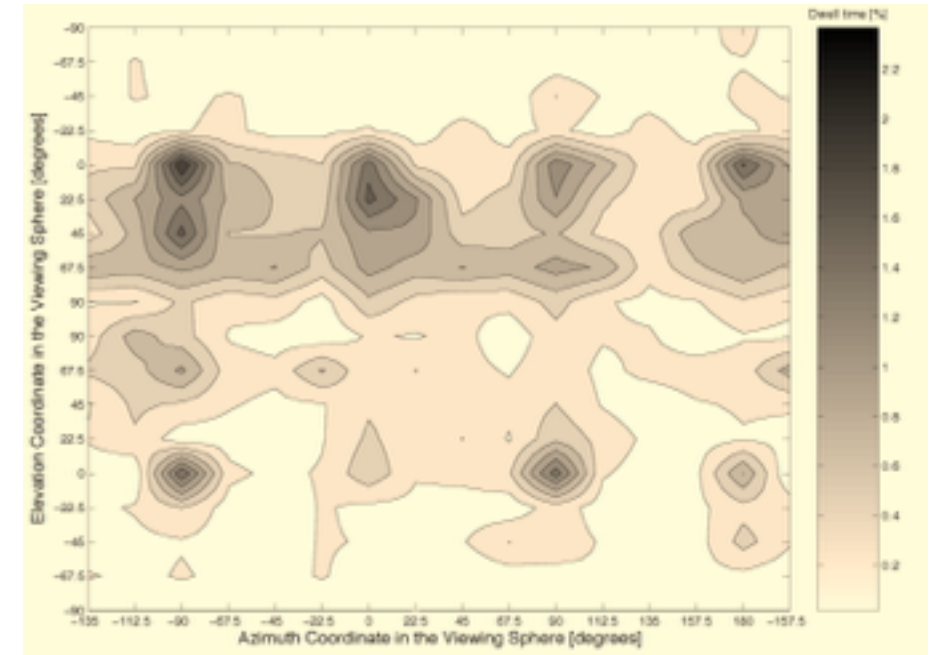
The agent observes an object while trying to achieve a perceptual goal state through manipulation.



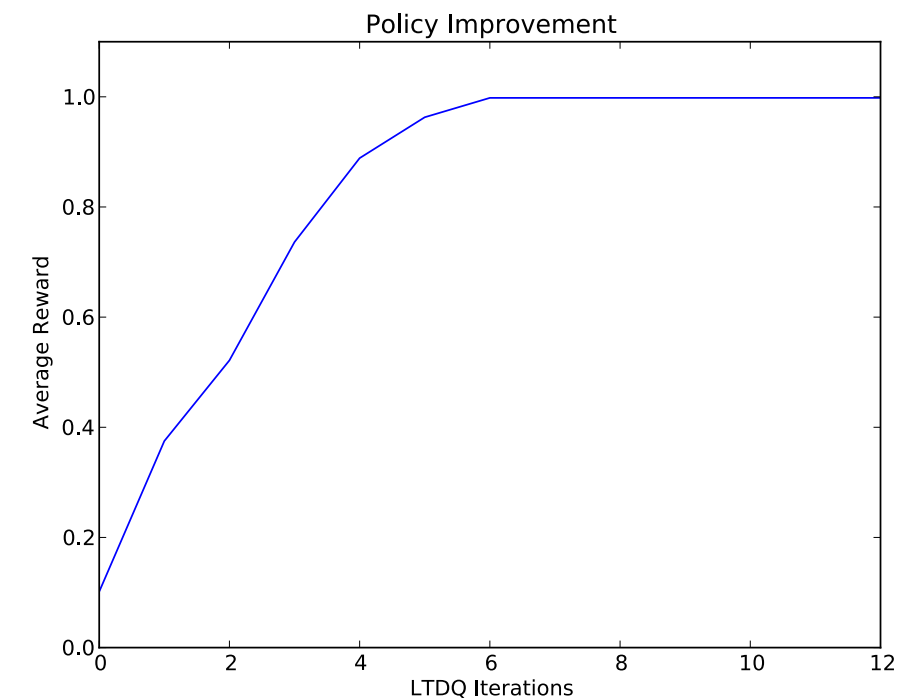
The agent observes an object while trying to achieve a perceptual goal state through manipulation.

Step 1

- Learn a policy to achieve a perceptual goal.
- The perceptual goal state is the normal view on a flat surface.



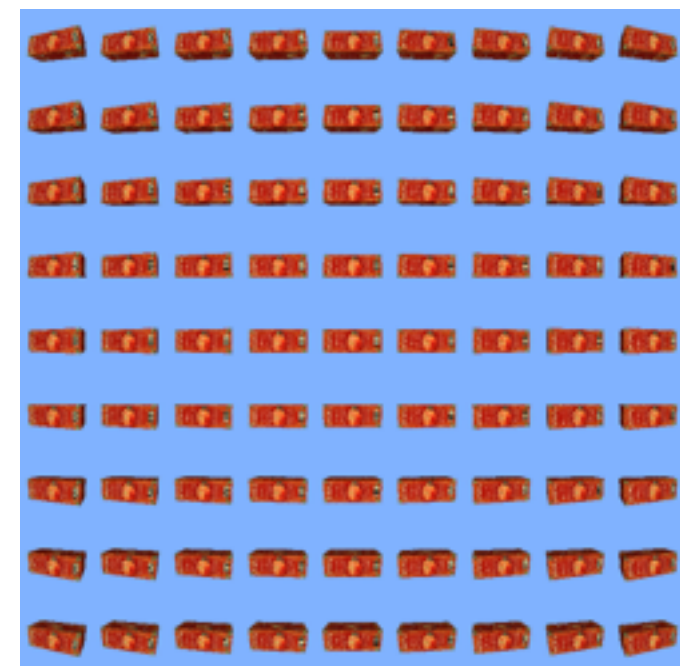
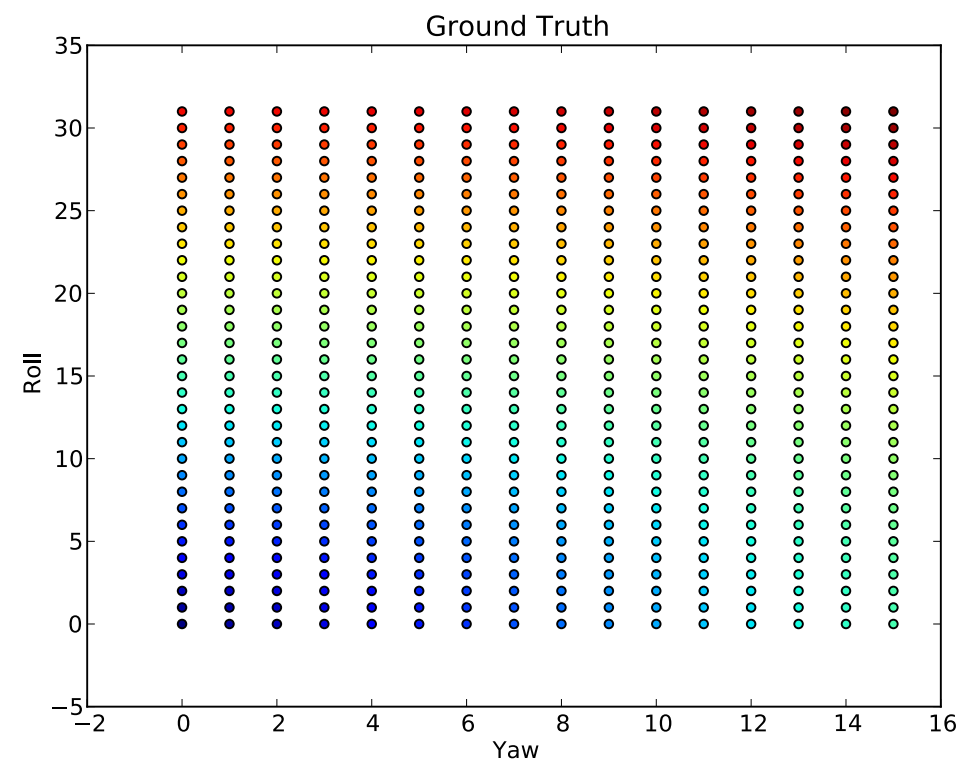
View biases in 30-36 month olds [Pereira et al, 2010].



Policy improvement in the object pose domain.

Step 2

- Apply the policy from different start states and record the resulting action traces.
- Sample action traces from different initial (roll, yaw) configurations.



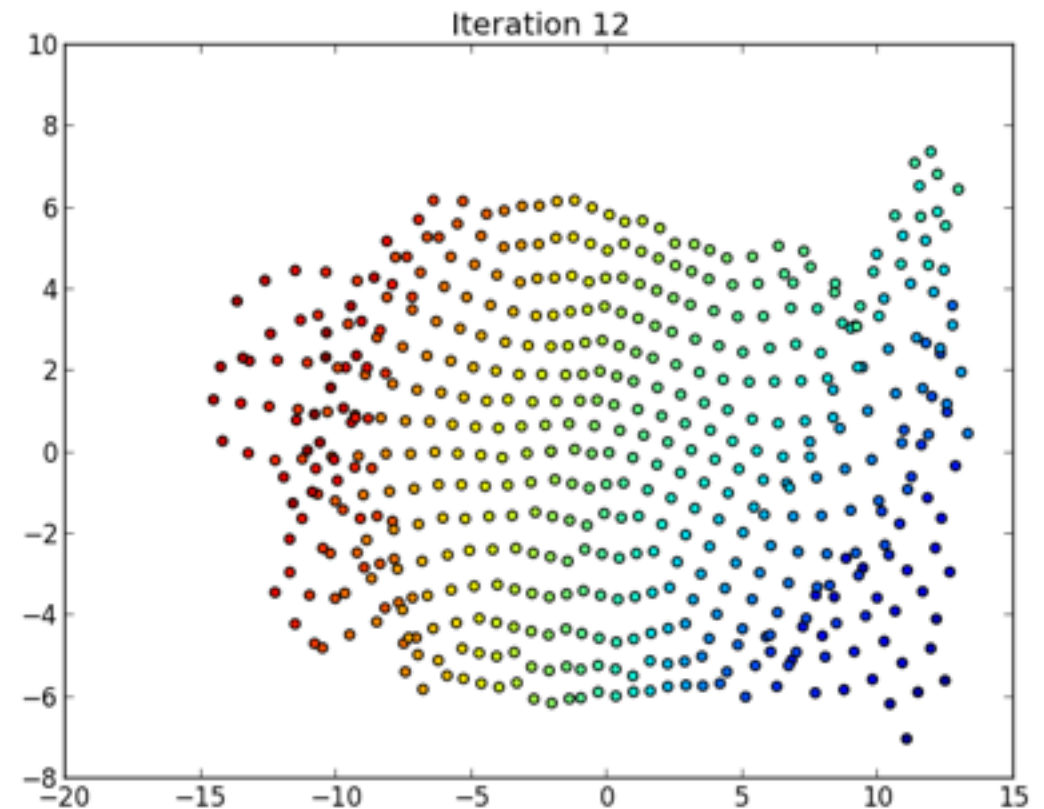
Step 3

- Generate matrix of distances by comparing action traces.

$$\Delta_{\pi} = \begin{bmatrix} 0. & 15.5 & 15. & \dots & 12. & 12.5 & 14. \\ 15.5 & 0. & 5. & \dots & 24.5 & 23.5 & 25.5 \\ 15. & 5. & 0. & \dots & 26.5 & 23.5 & 26.5 \\ & & & \vdots & & & \\ 12. & 24.5 & 26.5 & \dots & 0. & 6. & 4.5 \\ 12.5 & 23.5 & 23.5 & \dots & 6. & 0. & 7. \\ 14. & 25.5 & 26.5 & \dots & 4.5 & 7. & 0. \end{bmatrix}$$

Step 4

$$\text{MDS}(\Delta_{\pi}) =$$

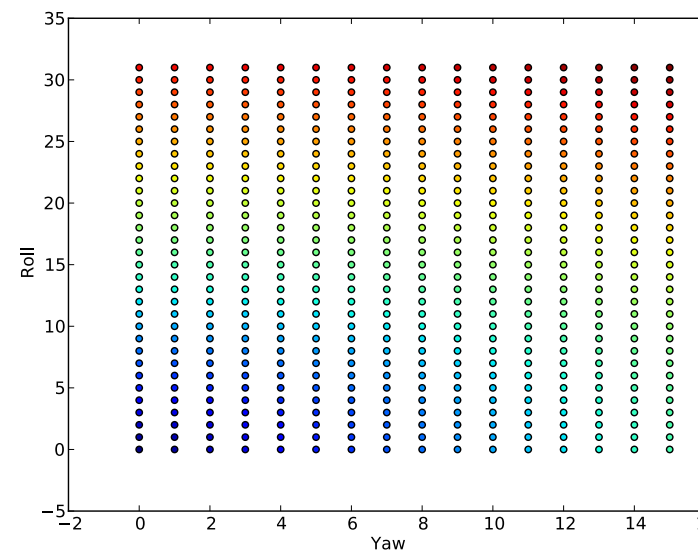


- Apply MDS to the matrix of distances to generate low-dimensional points associated with start states.

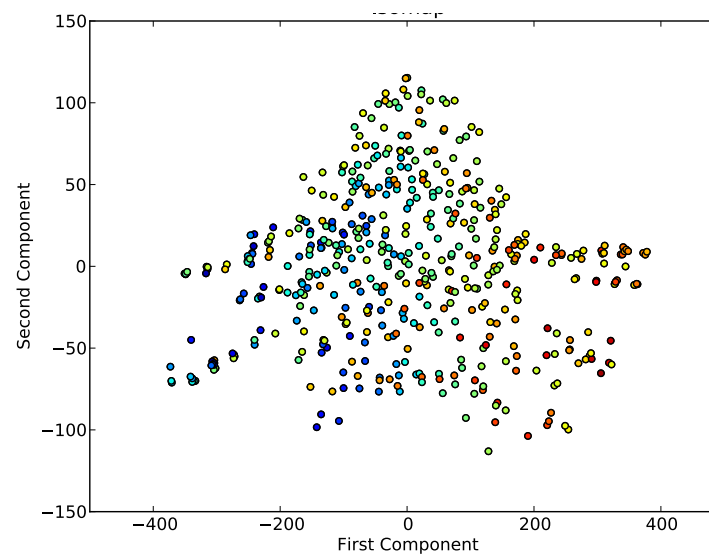
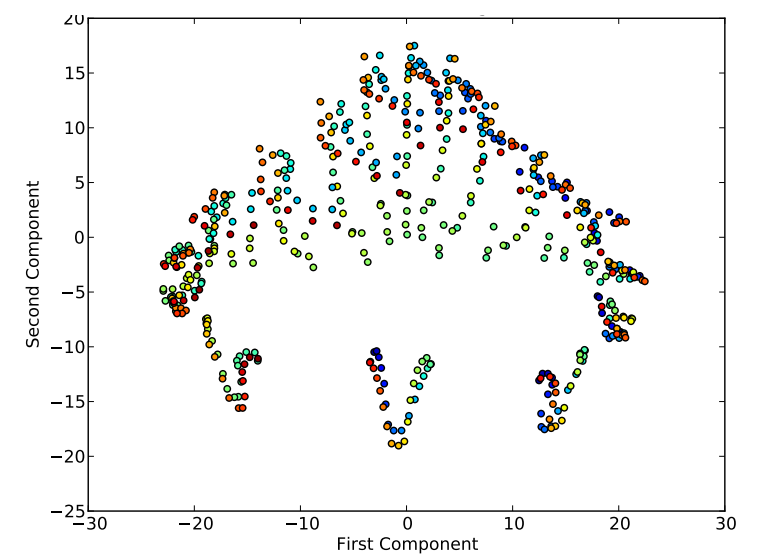
Comparison

- Sensorimotor Embedding outperforms both PCA and Isomap.

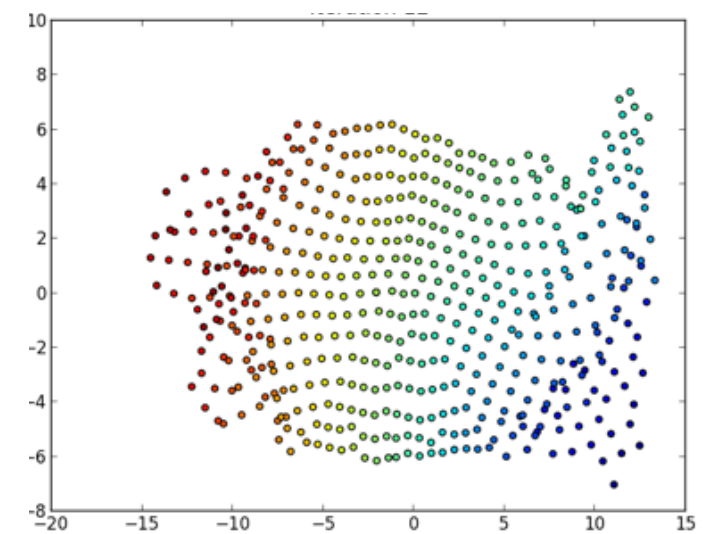
Ground Truth



PCA

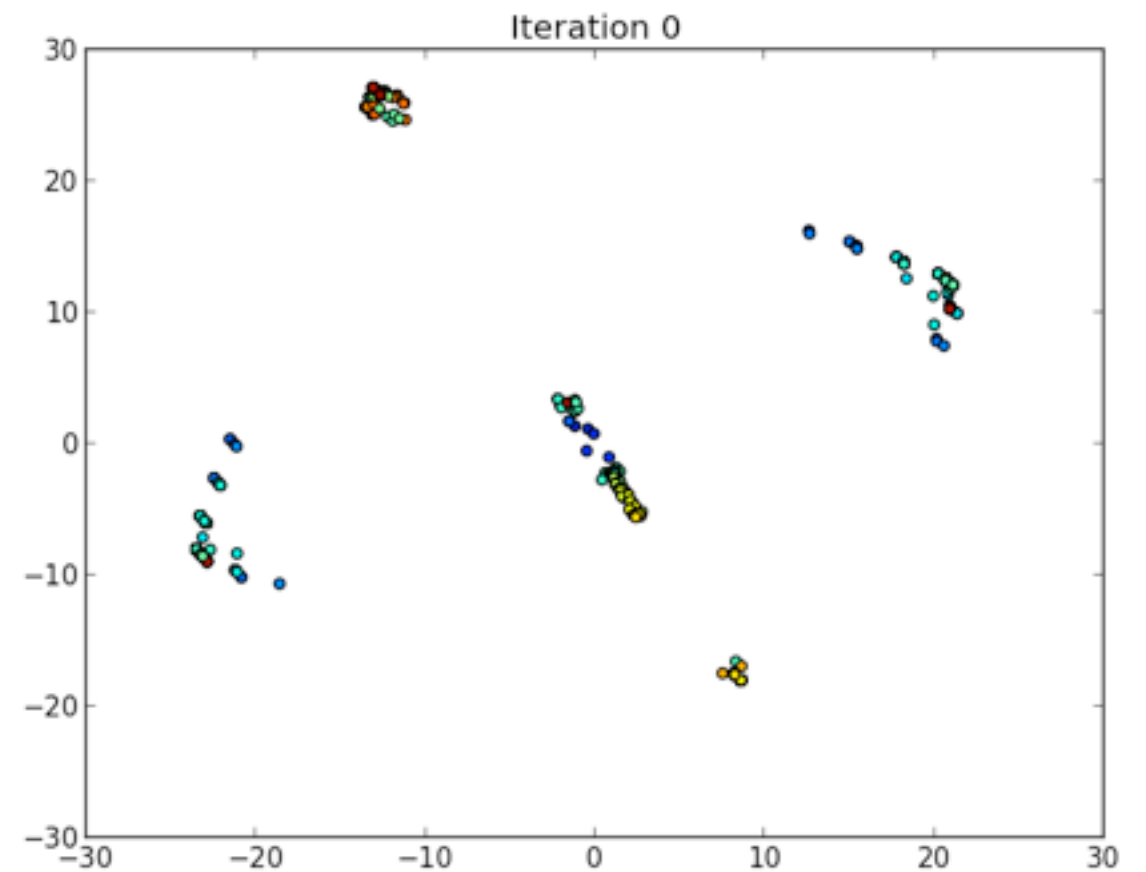
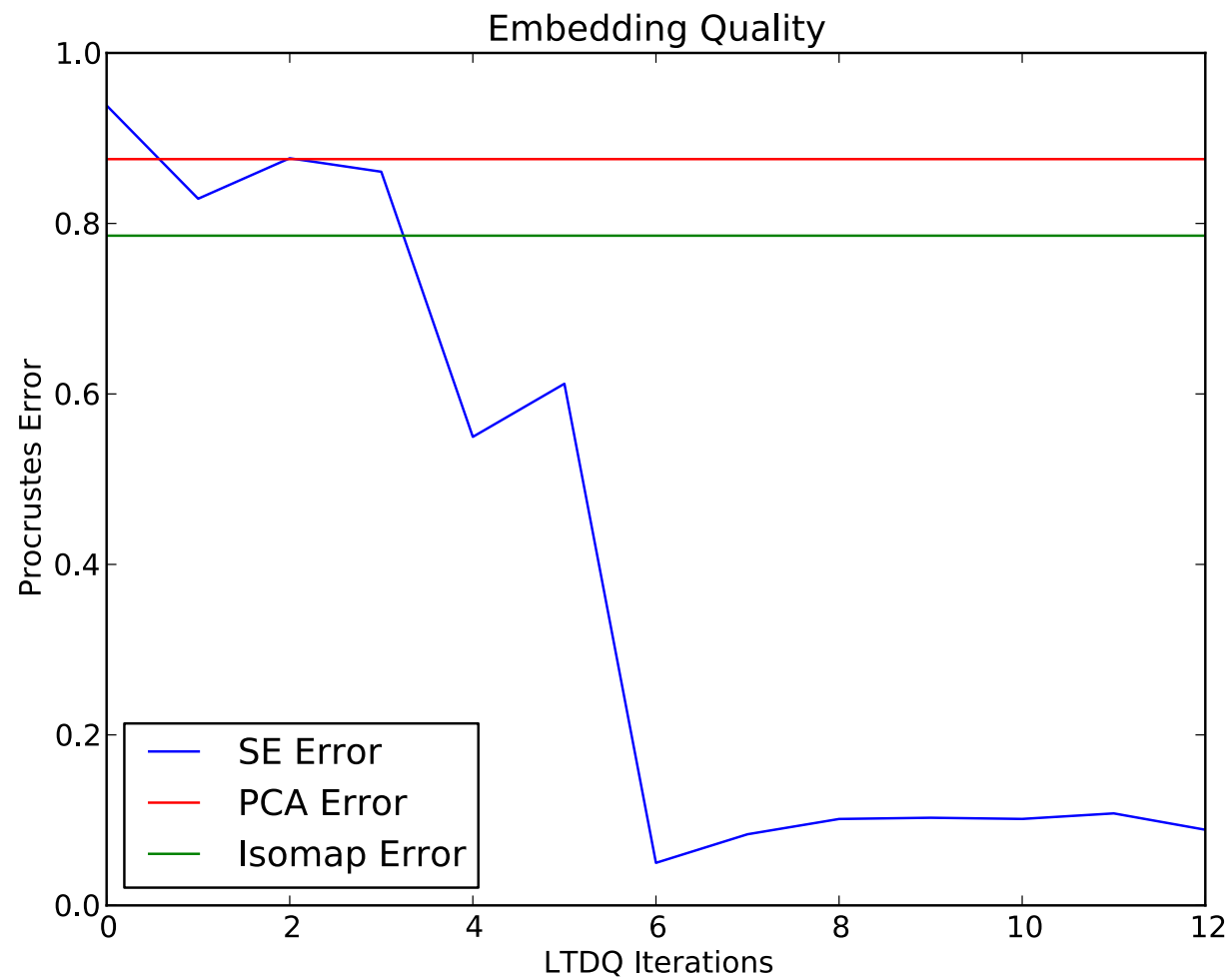


Isomap



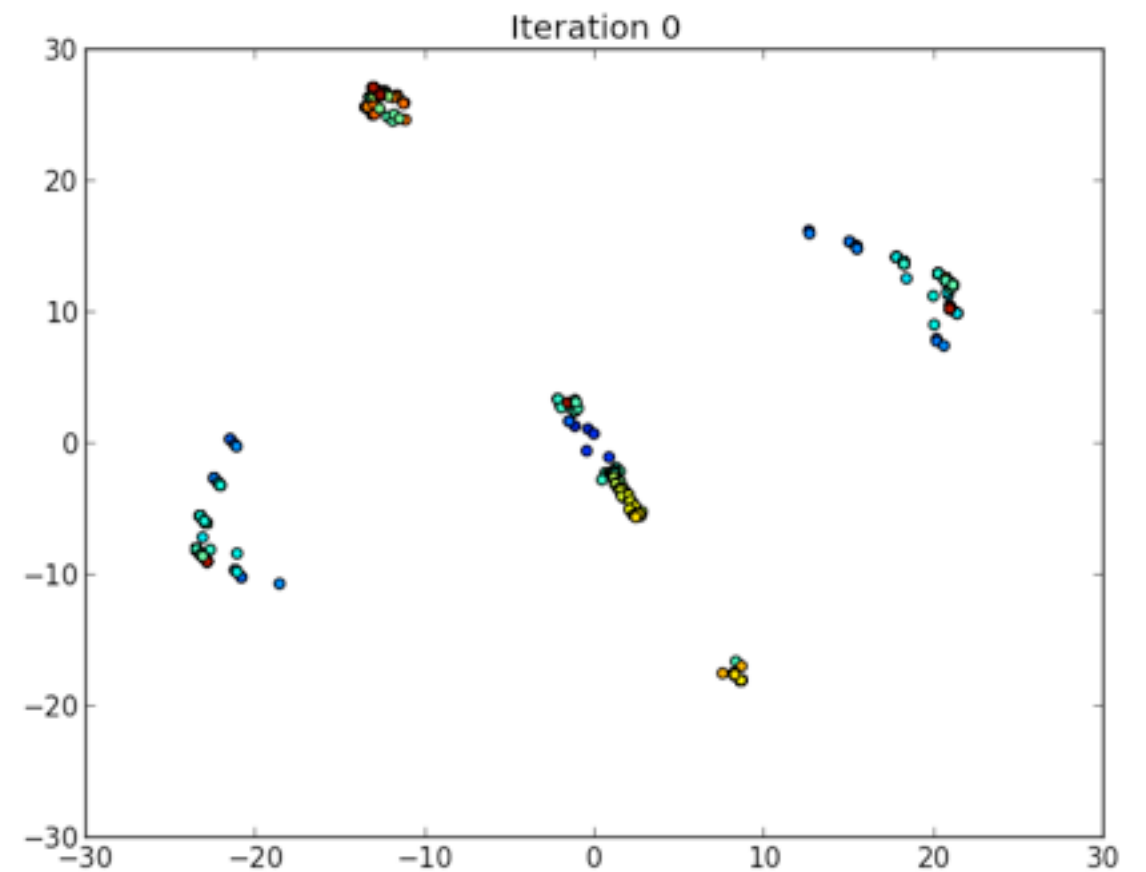
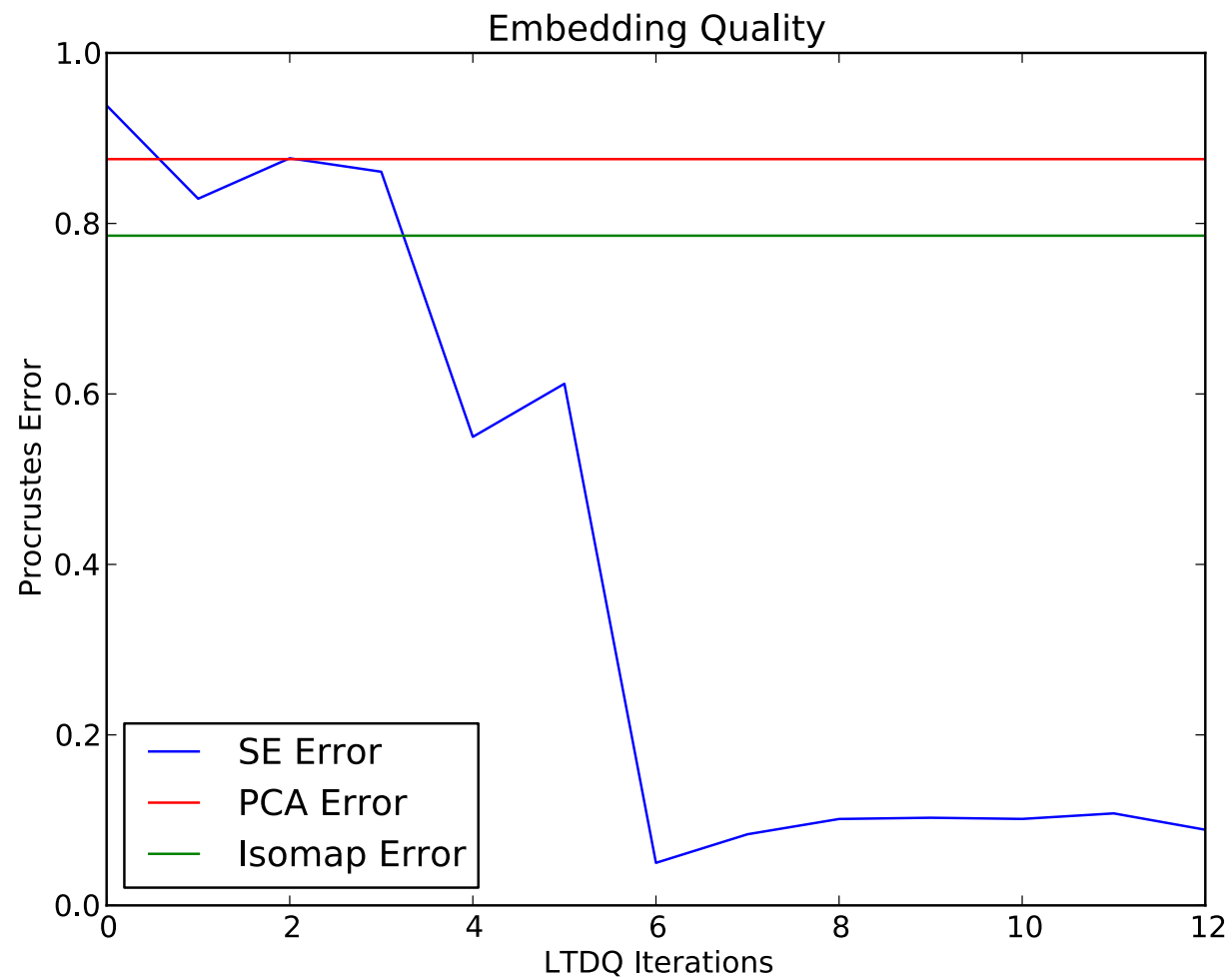
Sensorimotor Embedding

Policy and Geometry



As the policy improves, so does the learned geometry.

Policy and Geometry



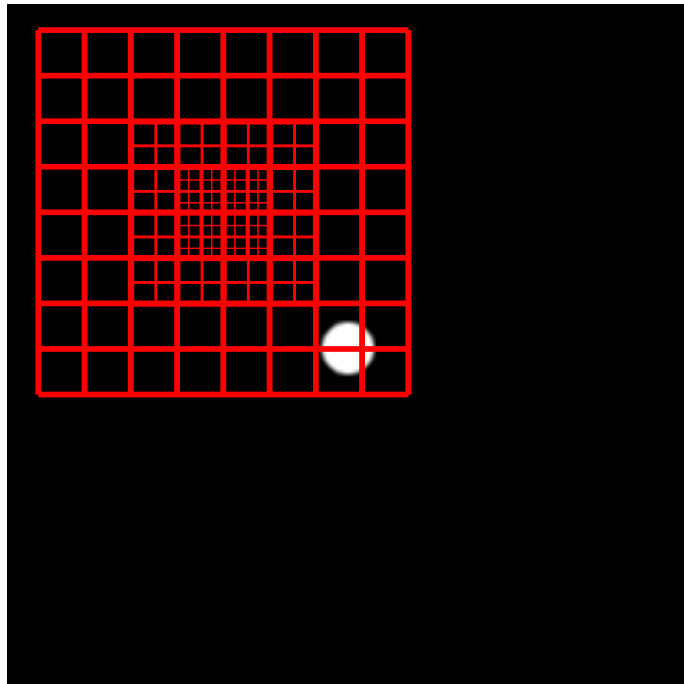
As the policy improves, so does the learned geometry.

Applications of Sensorimotor Embedding

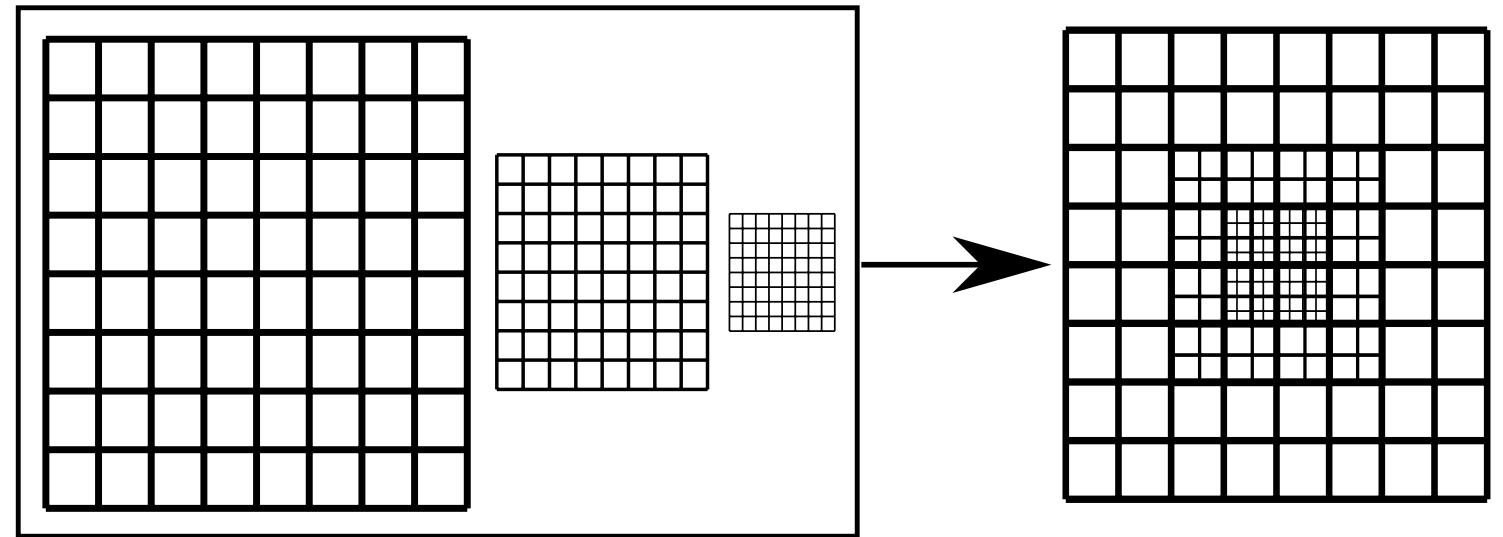
- Learning structure of the foveated retina
- Learning robot position
- Learning object pose
- Learning stereo image depth
- Visual mountain car task

Foveated Retina

Sensor Structure



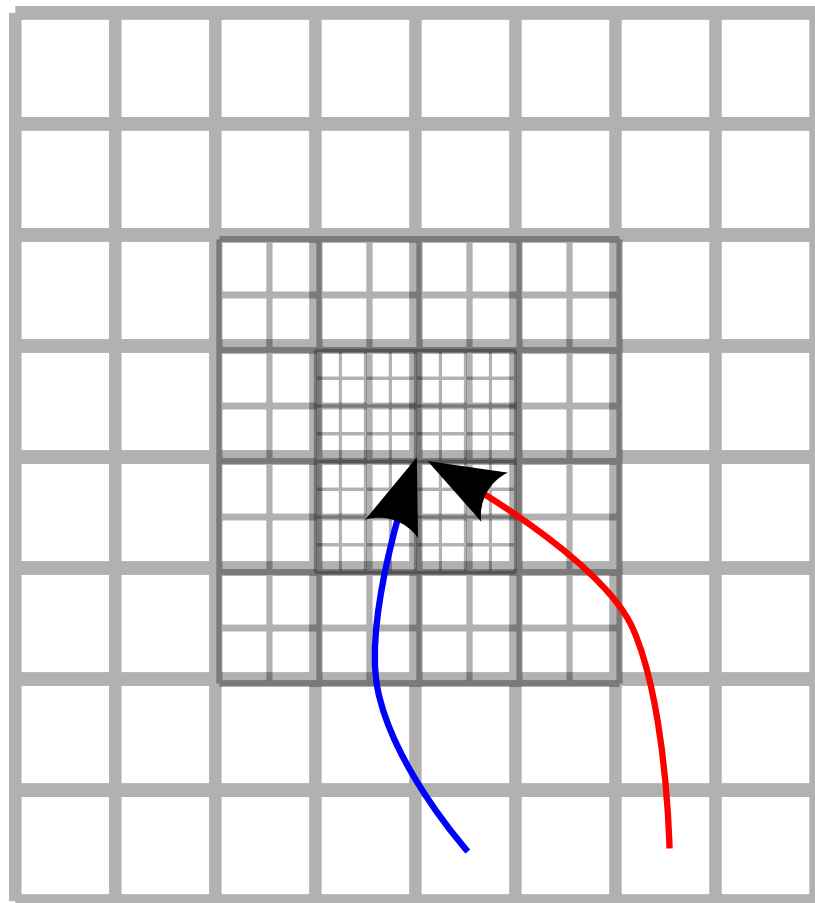
The goal is to maximize activation by moving the fovea to the brightest point.



A foveated sensor collects data at multiple overlapping resolutions.

Agent manipulates a foveated sensor. The task is to maximize activation via saccades, e.g. single ballistic actions.

Sensor Structure



Functional form of the policy:

$$\pi(s) = \frac{1}{\mathcal{R}_{\mathcal{I}}(s)} \sum_{I_k \in \mathcal{I}} \delta(I_k, s) \cdot \pi_k(s)$$

where

$\pi_k(s)$ is the policy for the k th receptive field

$\delta(I_k, s)$ is the activation of the k th receptive field

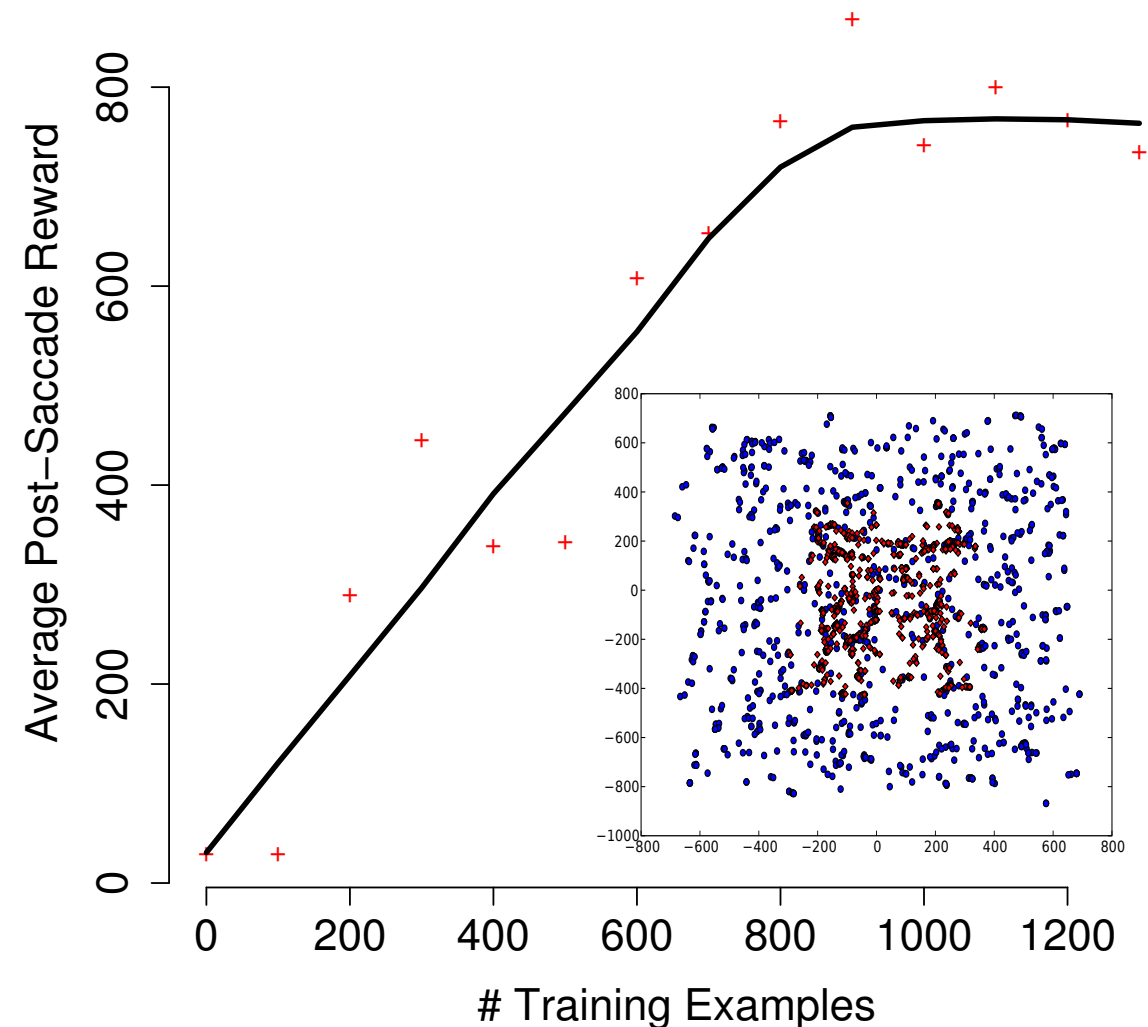
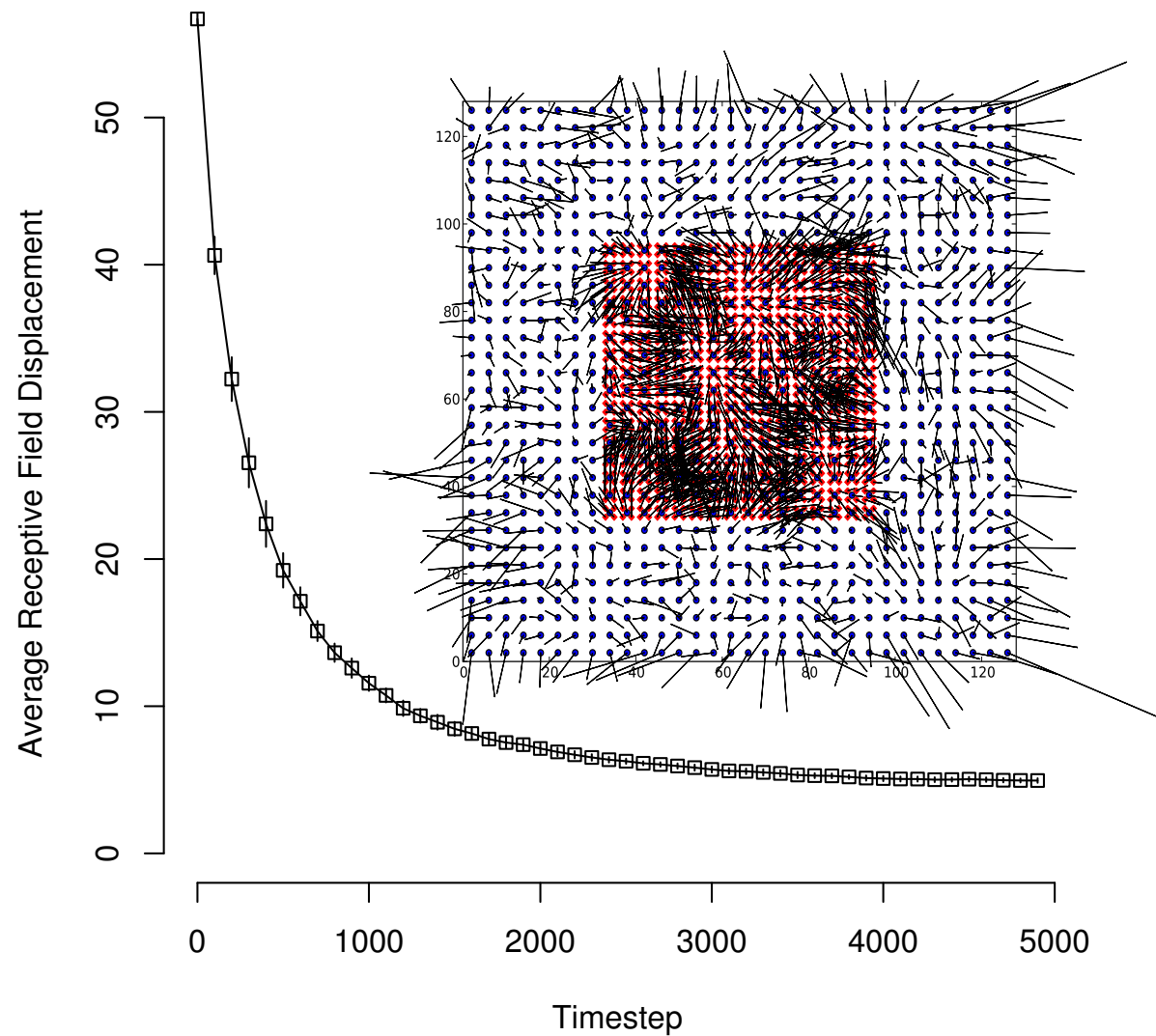
$\mathcal{R}_{\mathcal{I}}(s)$ is the total retina activation

Each sensor “votes” for a saccade.
Votes are weighted by activation.

Sensor Structure

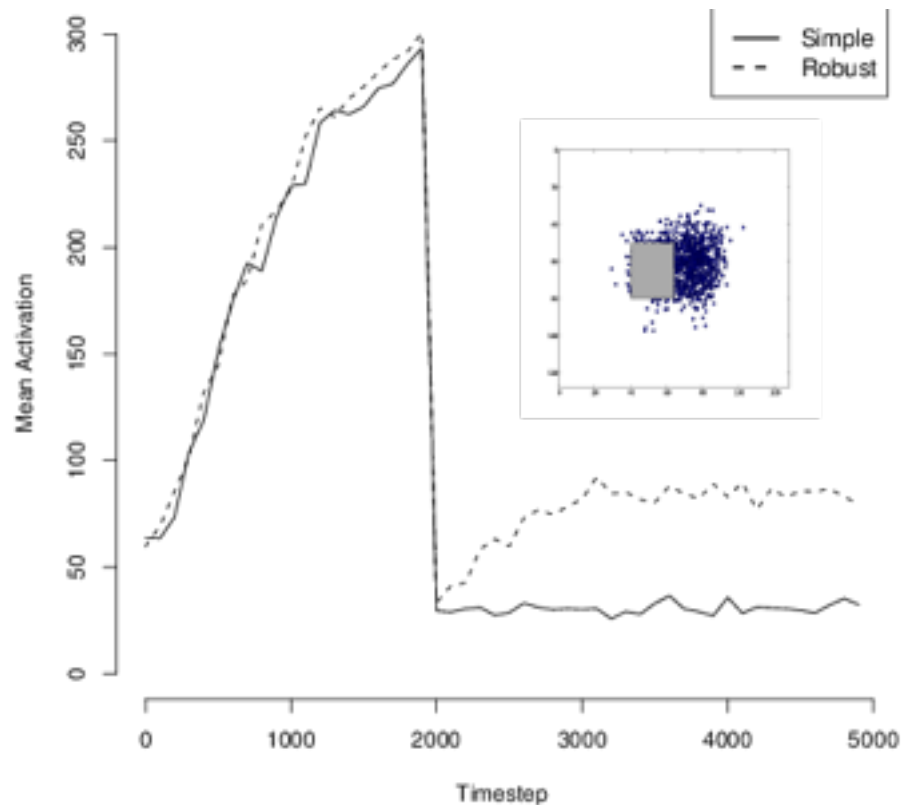
Since actions are ballistic, each fields' policy is a vector that can be interpreted as that fields' position in the sensor.

Sensor Structure



As the saccade policy improves (right), the learned structure of the the sensor improves (left).

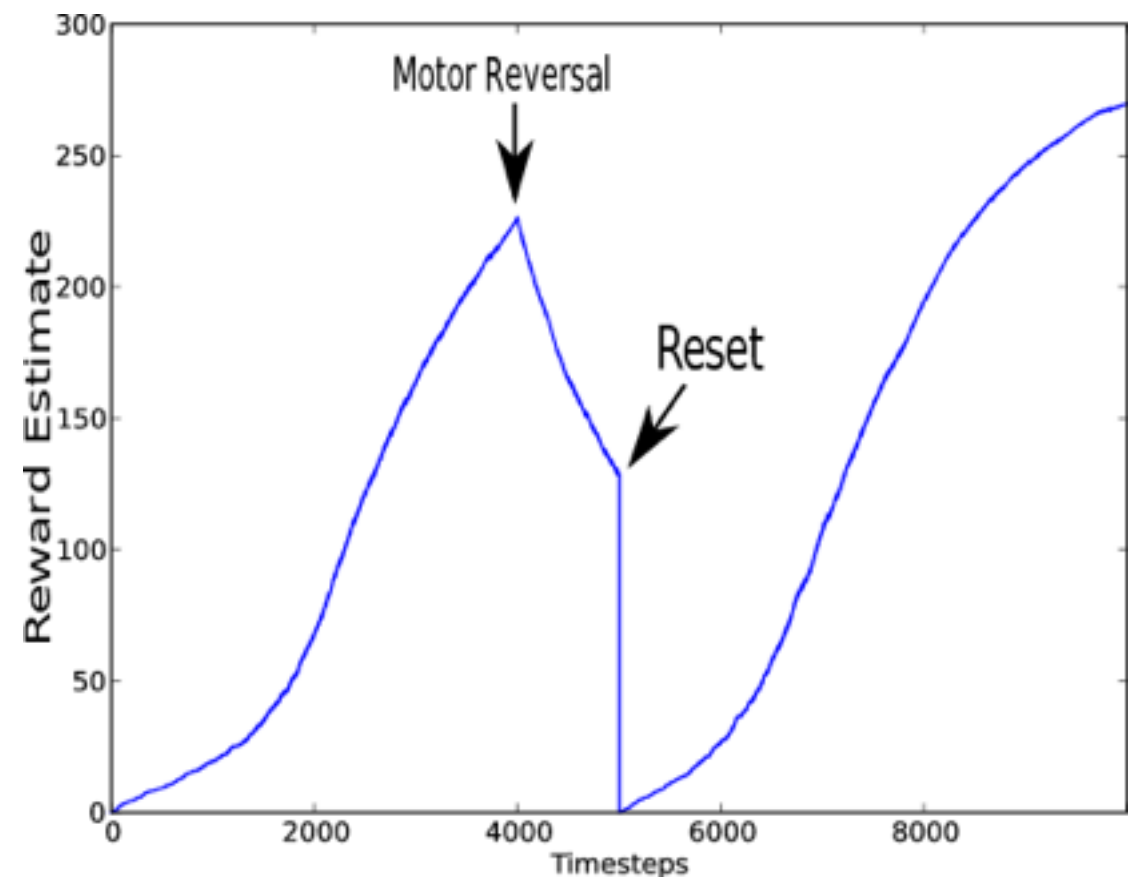
Lesions



Simulate macular degeneration by lesioning part of the fovea.

The underlying policy can adapt to lesioning. As the policy adapts, so to does the learned geometry.

Inversion

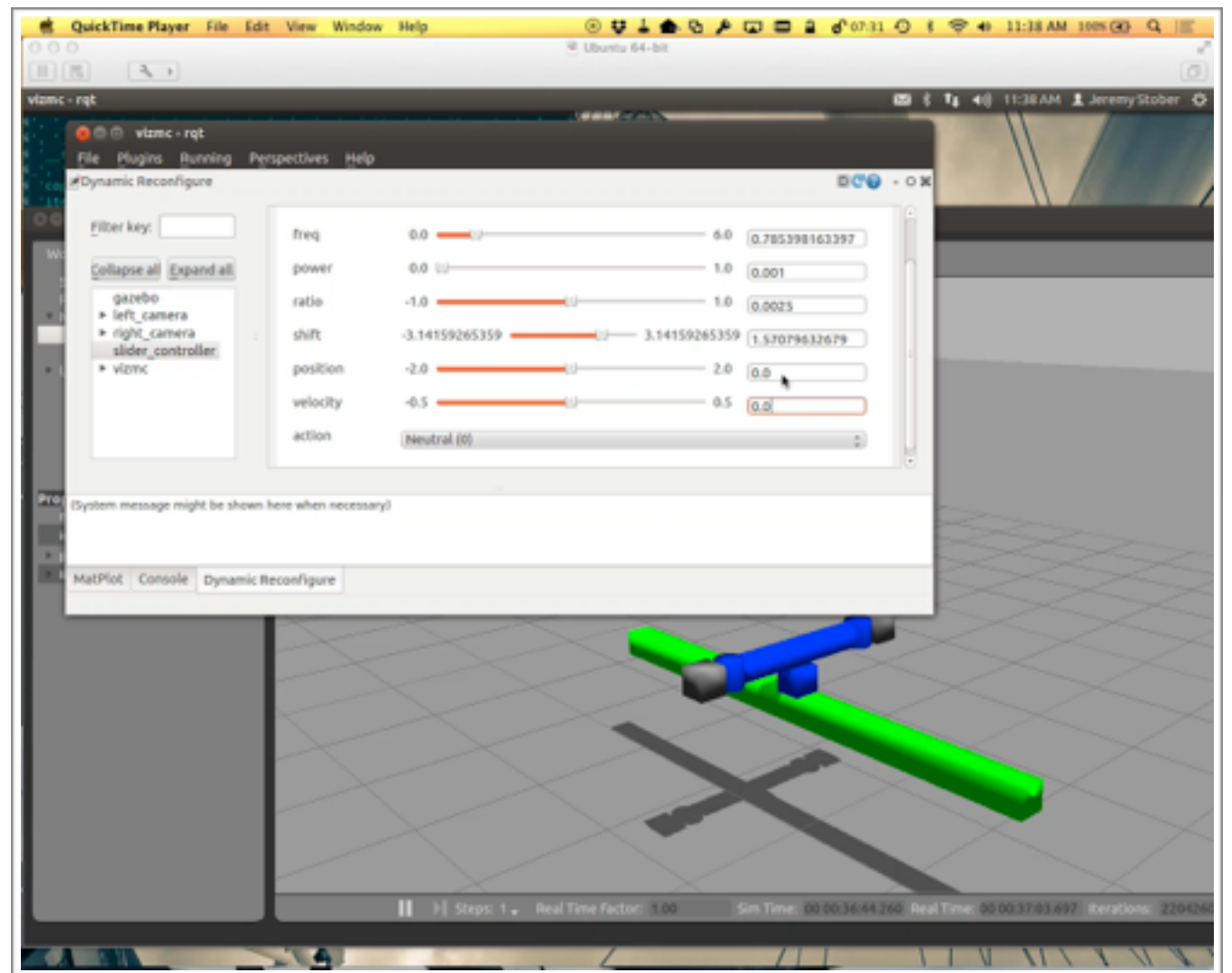
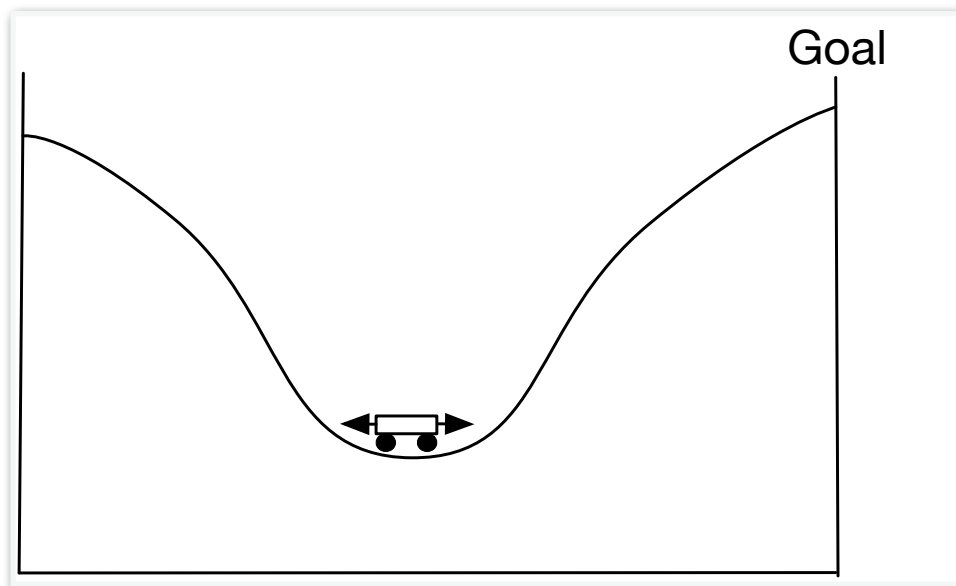


Simulate vision inversion by switching up and down motor commands.

The agent's policy can adapt to motor reversal, and as the policy adapts, so does the learned geometry.

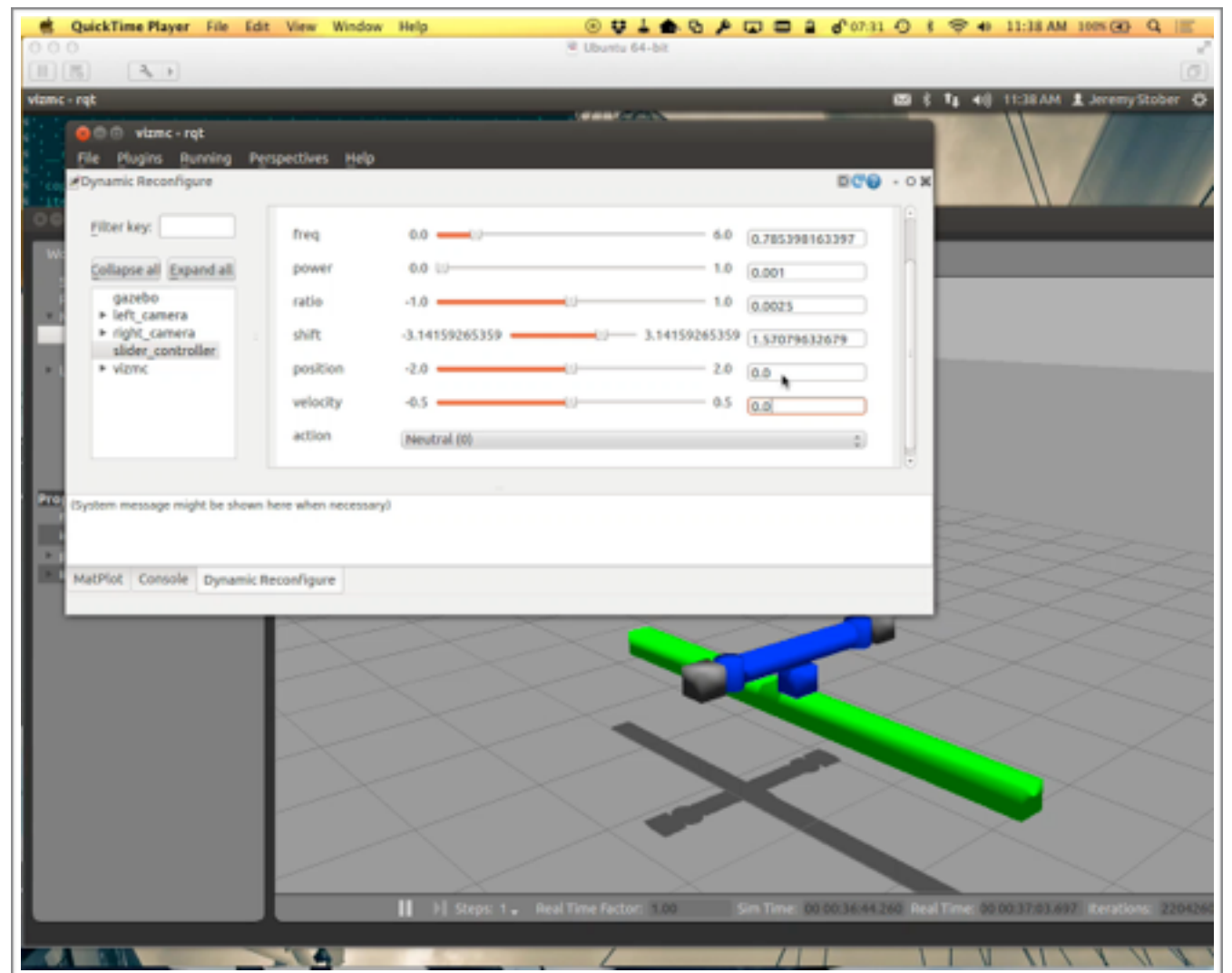
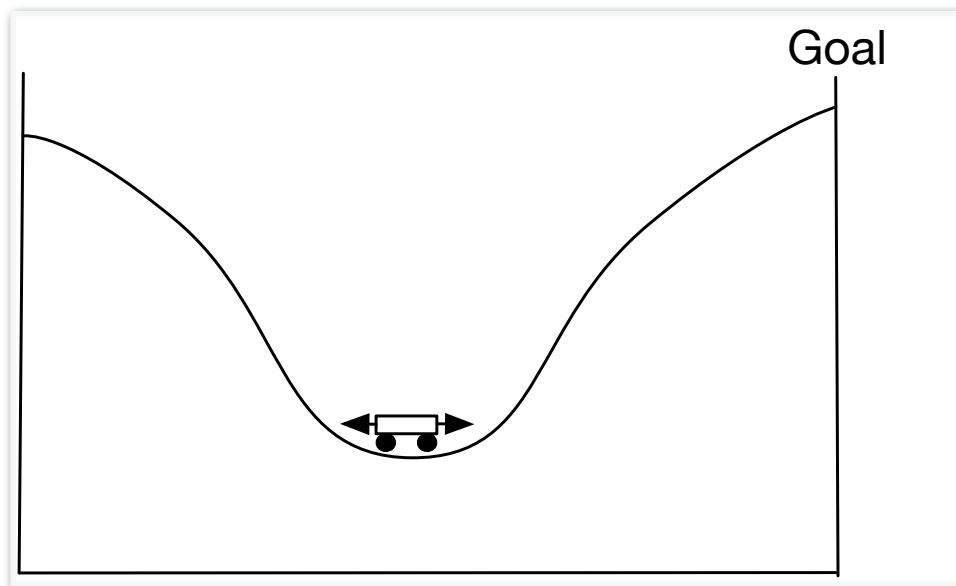
Visual Mountain Car

Task Description



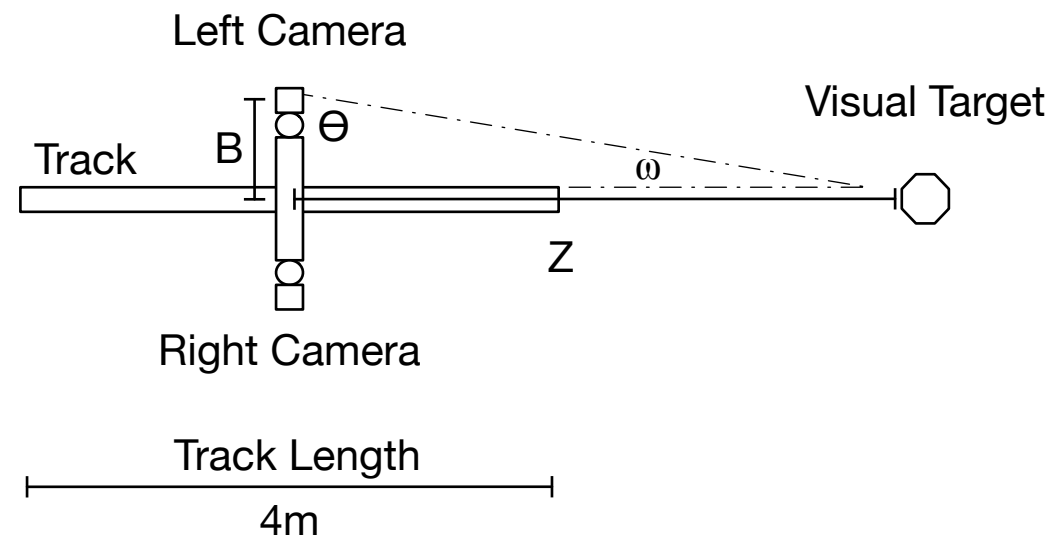
The agent has to power an under powered car out of a valley. Instead of having access to position and velocity, the agent has to infer position and velocity from sensorimotor experience.

Task Description



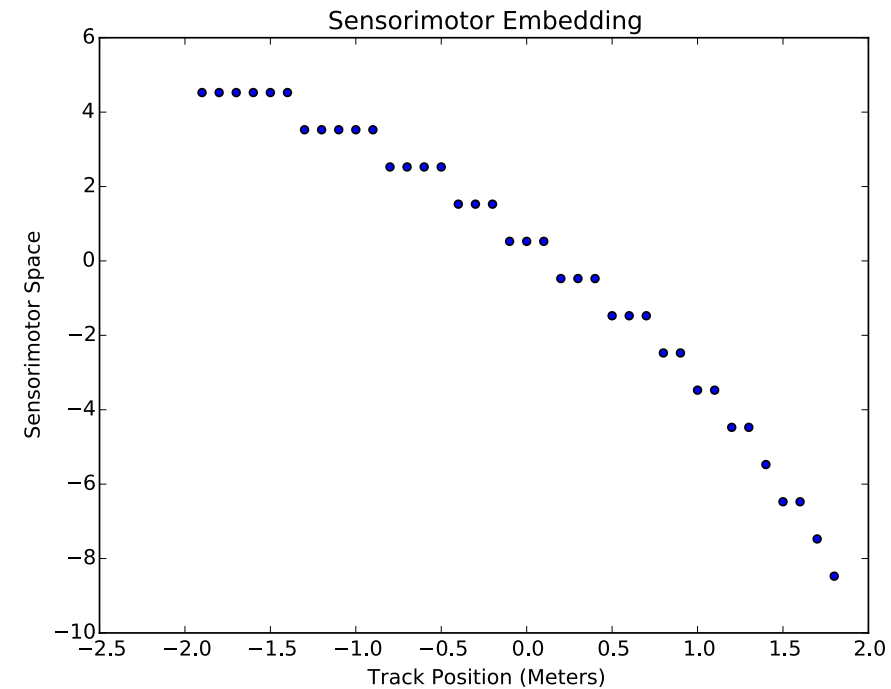
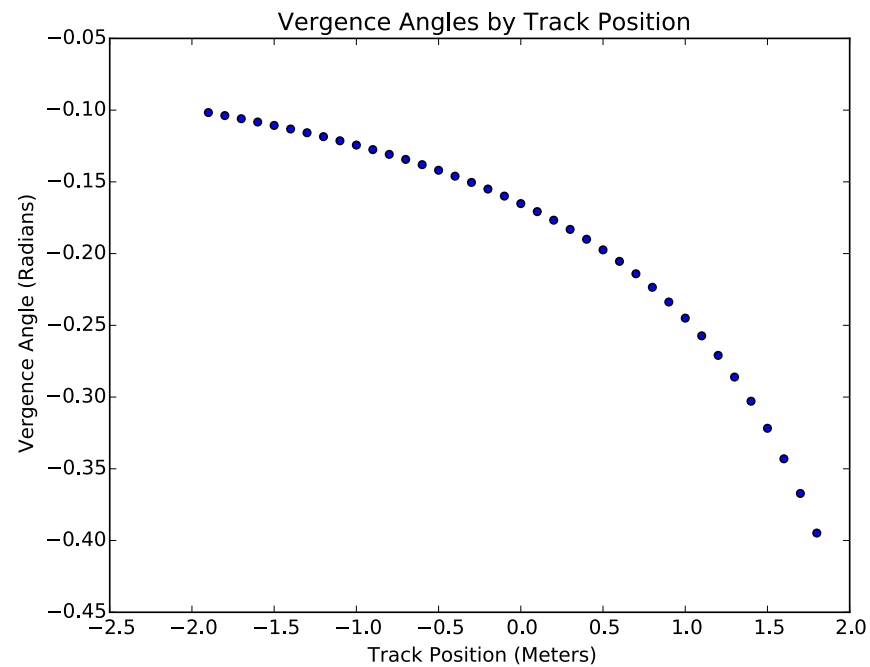
The agent has to power an under powered car out of a valley. Instead of having access to position and velocity, the agent has to infer position and velocity from sensorimotor experience.

Depth Features



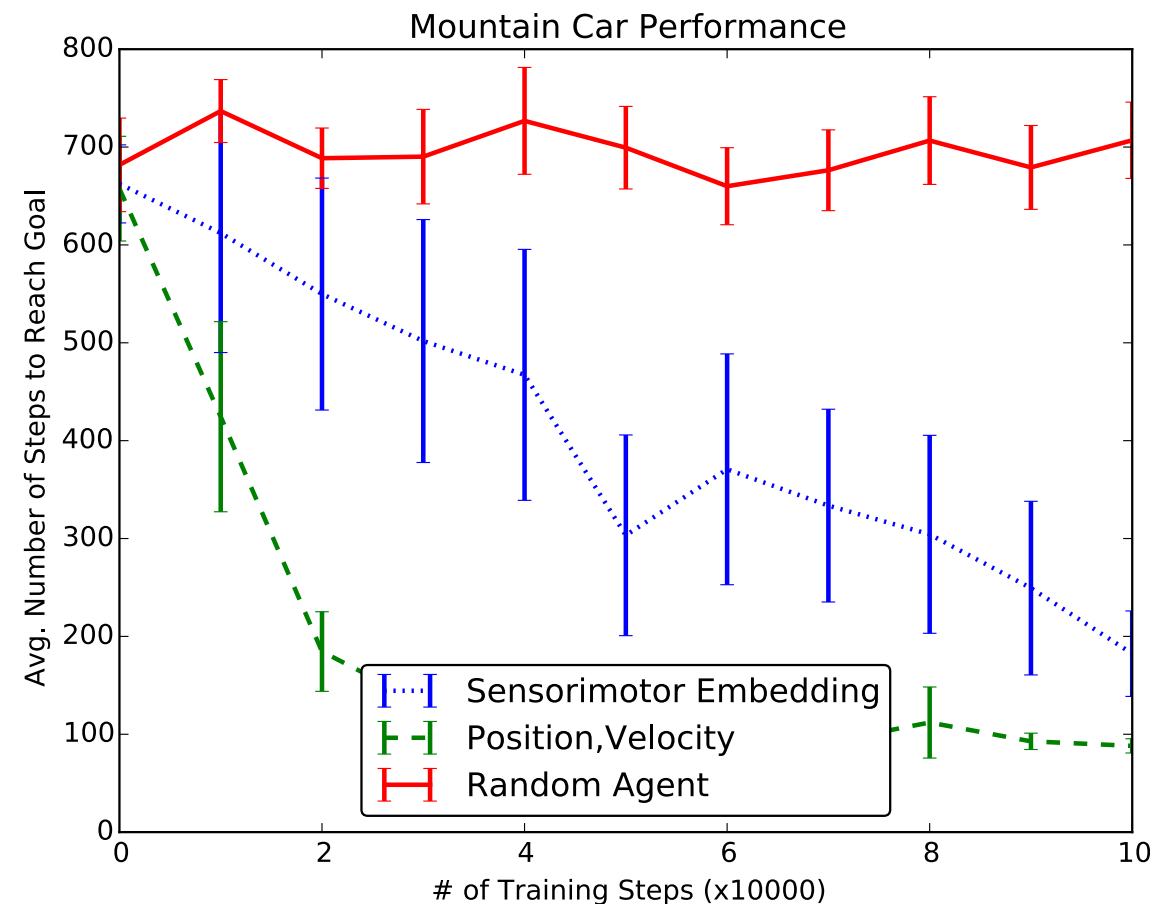
A robot moves along a track and at different points performs a sequence of vergence actions to align left and right cameras on a target object.

Depth Features



After sensorimotor embedding, the one dimensional depth feature resembles the true vergence angles. More distant positions result in aliasing.

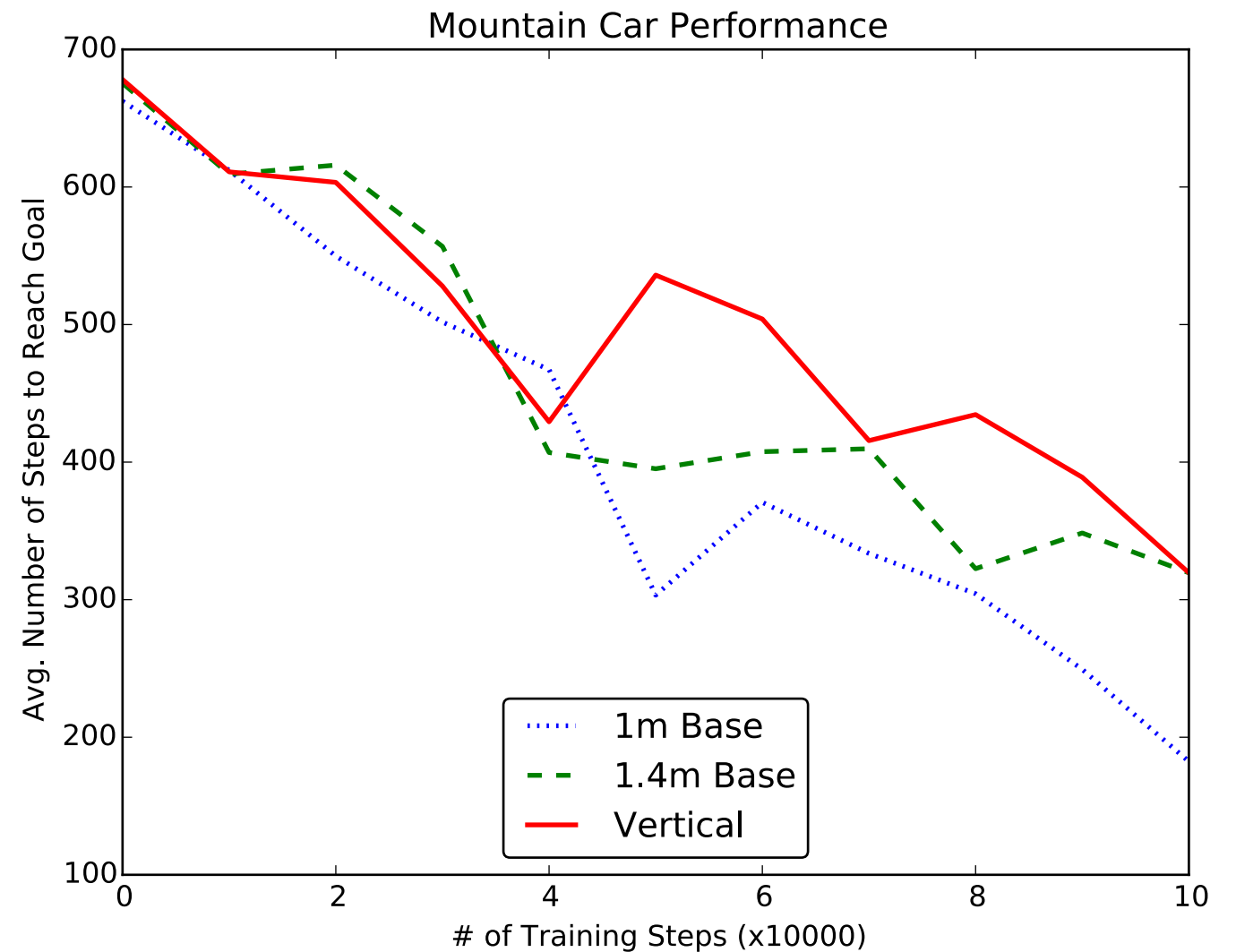
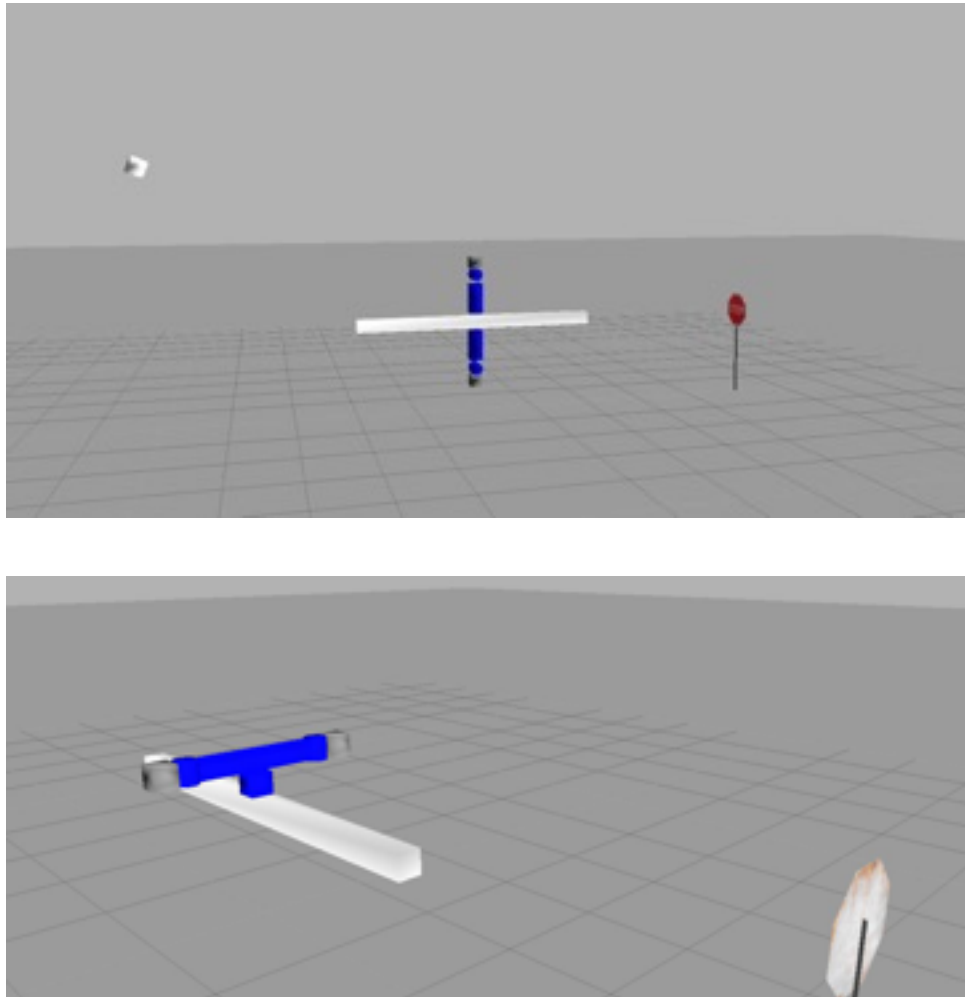
Results



An agent using sensorimotor embedding is compared to a random agent and an agent training from (position, velocity) directly.

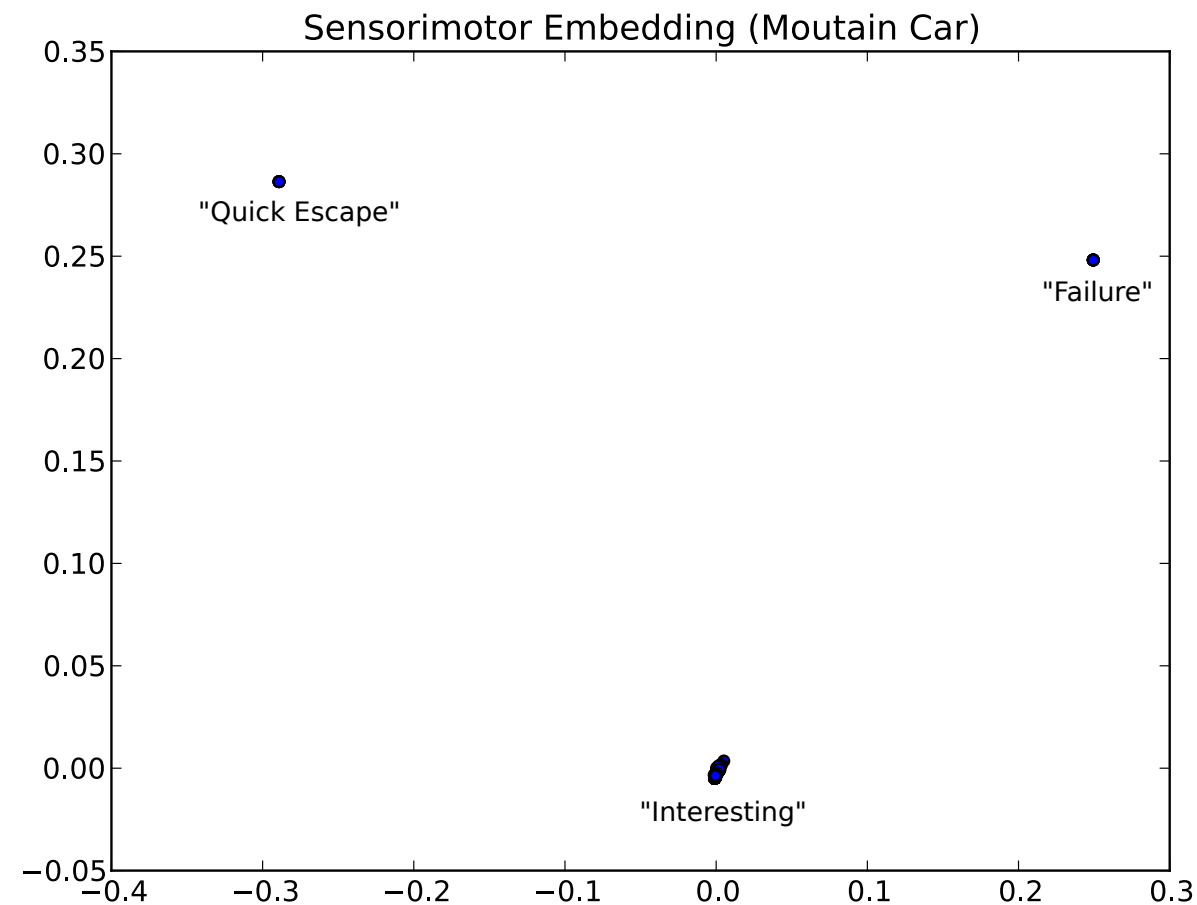
After sufficient training, the sensorimotor agent performs almost as well as a regular agent.

Different Morphologies



Sensorimotor embedding performs just as well on two alternate robot morphologies without any modifications.

Future Work



Policy visualization with sensorimotor embedding. This may allow for easier inspection and debugging of learned policies when developing RL agents.

Conclusions

- Sensorimotor embedding is a new manifold learning algorithm.
- Sensorimotor embedding is a developmental approach to learning geometry.
- Sensorimotor embedding is robust to lesion and inversion events, and can run on different robots.
- Sensorimotor embedding generates geometric features that help agents solve tasks.





Questions?