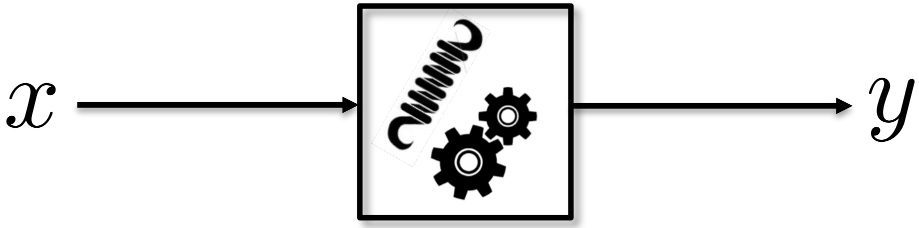
*Lecture: Deep Learning and Differential Programming*

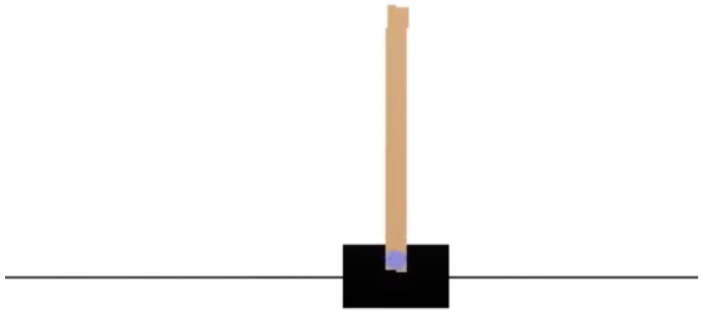
# 5.2 Should we model or learn?

https://liris.cnrs.fr/christian.wolf/teaching

INSA LYON Christian Wolf

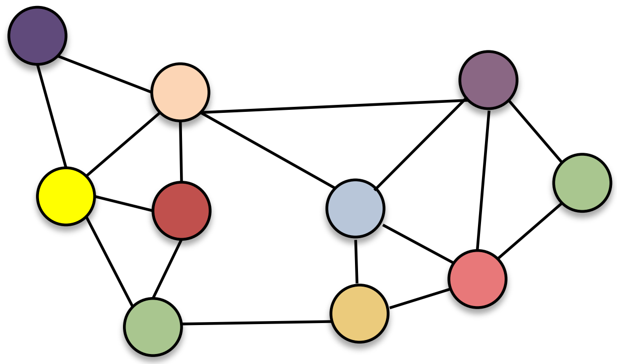



**(Physical) model**



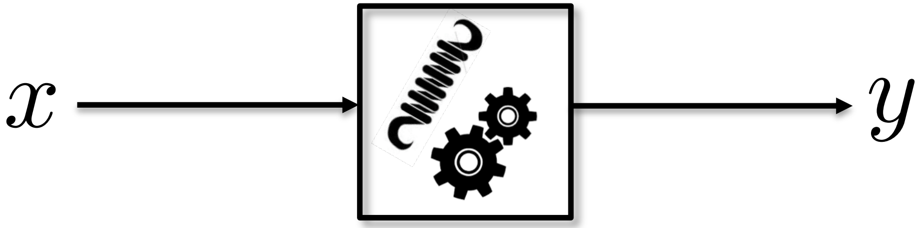
$$\ddot{\theta}_t = \frac{g\sin\theta_t + \cos\theta_t\left[\dfrac{-F_t - ml\dot{\theta}_t^2\sin\theta_t + \mu_c\,\mathrm{sgn}(\dot{x}_t)}{m_c + m}\right] - \dfrac{\mu_p\dot{\theta}_t}{ml}}{l\left[\dfrac{4}{3} - \dfrac{m\cos^2\theta_t}{m_c + m}\right]}$$

$$\ddot{x}_t = \frac{F_t + ml\left[\dot{\theta}_t^2\sin\theta_t - \ddot{\theta}_t\cos\theta_t\right] - \mu_c\,\mathrm{sgn}(\dot{x}_t)}{m_c + m}$$

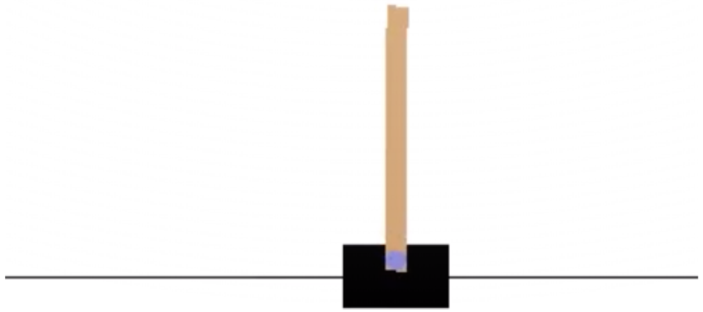


**Planing/shortest path**

- **Dijkstra**
- **A***
- **Front Propagation**

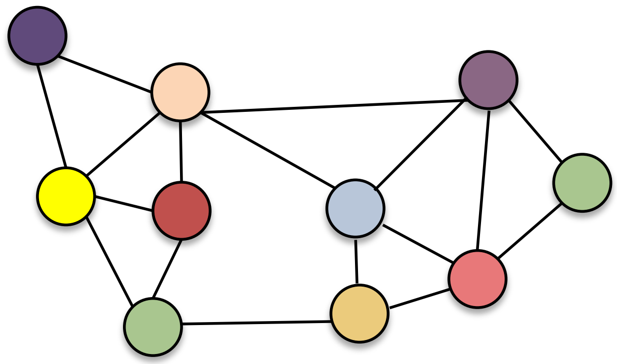



**(Physical) model**



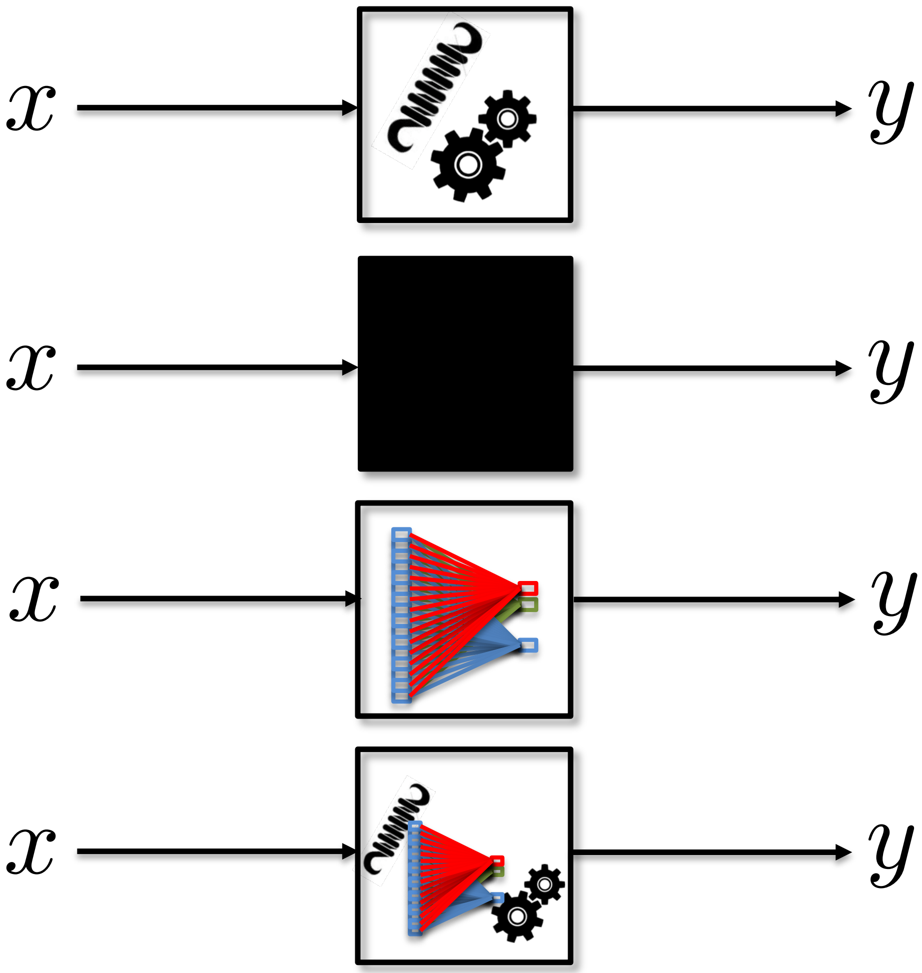
$$\ddot{\theta}_t = \frac{g\sin\theta_t + \cos\theta_t\left[\dfrac{-F_t - ml\dot{\theta}_t^2\sin\theta_t + \mu_c\,\mathrm{sgn}(\dot{x}_t)}{m_c + m}\right] - \dfrac{\mu_p\dot{\theta}_t}{ml}}{l\left[\dfrac{4}{3} - \dfrac{m\cos^2\theta_t}{m_c + m}\right]}$$

$$\ddot{x}_t = \frac{F_t + ml\left[\dot{\theta}_t^2\sin\theta_t - \ddot{\theta}_t\cos\theta_t\right] - \mu_c\,\mathrm{sgn}(\dot{x}_t)}{m_c + m}$$



**Plannifcation/shortest path**

- **Dijkstra**
- **A***
- **Front Propagation**




**(Physical) model**

**Black box (unknown function)**

**White box (learned but known complex function)**

**Hybrid model**

# Example:
# Robot navigation

**Purely geometrical mapping + planning**

**Adding semantics**

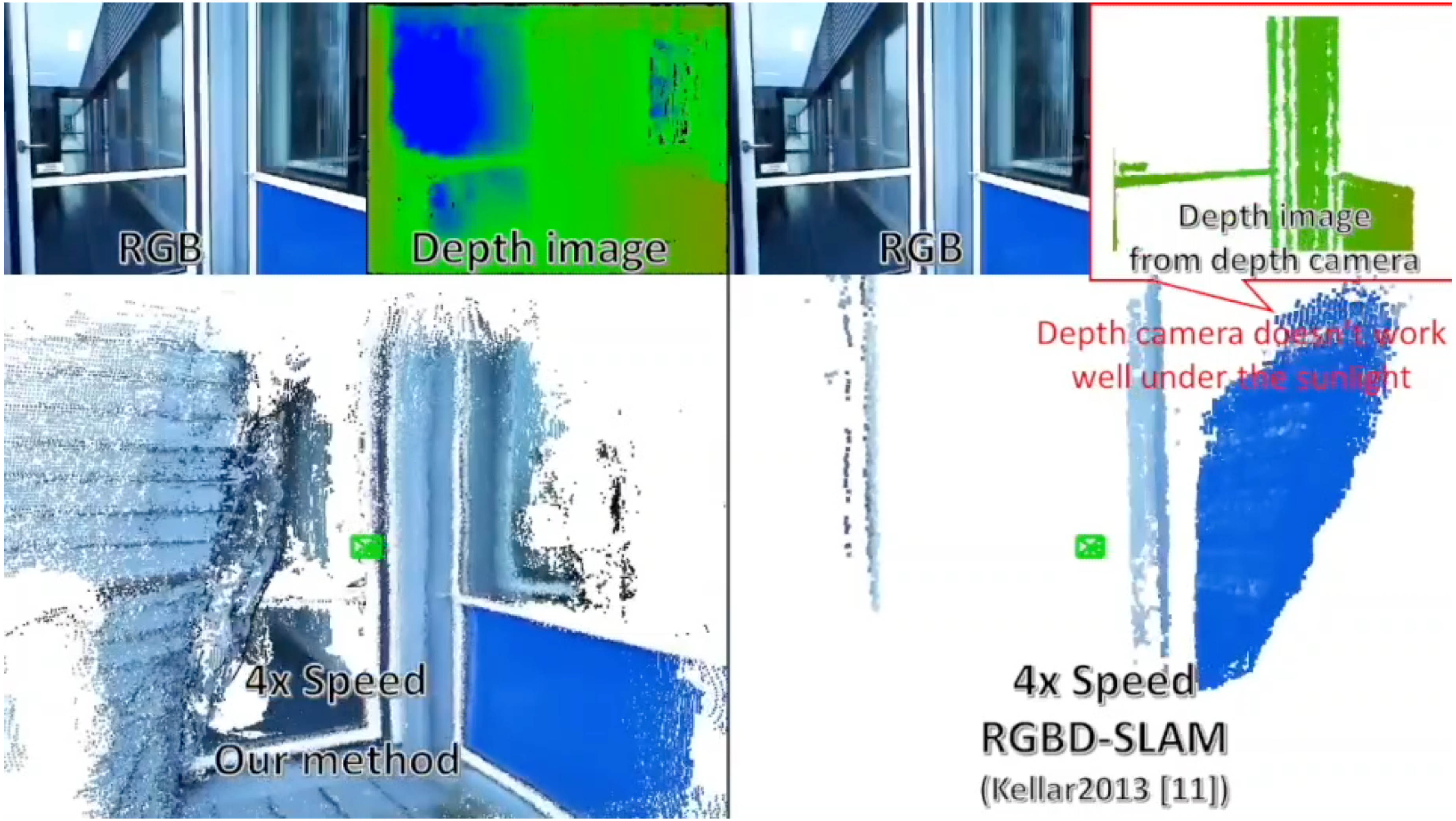
**Purely learned policies (Deep-RL)**

**Inductive biases: Geometry, topology, stability etc.**


*+ Real world solutions*
*- Simple tasks and reasoning (eg. waypoint navigation)*

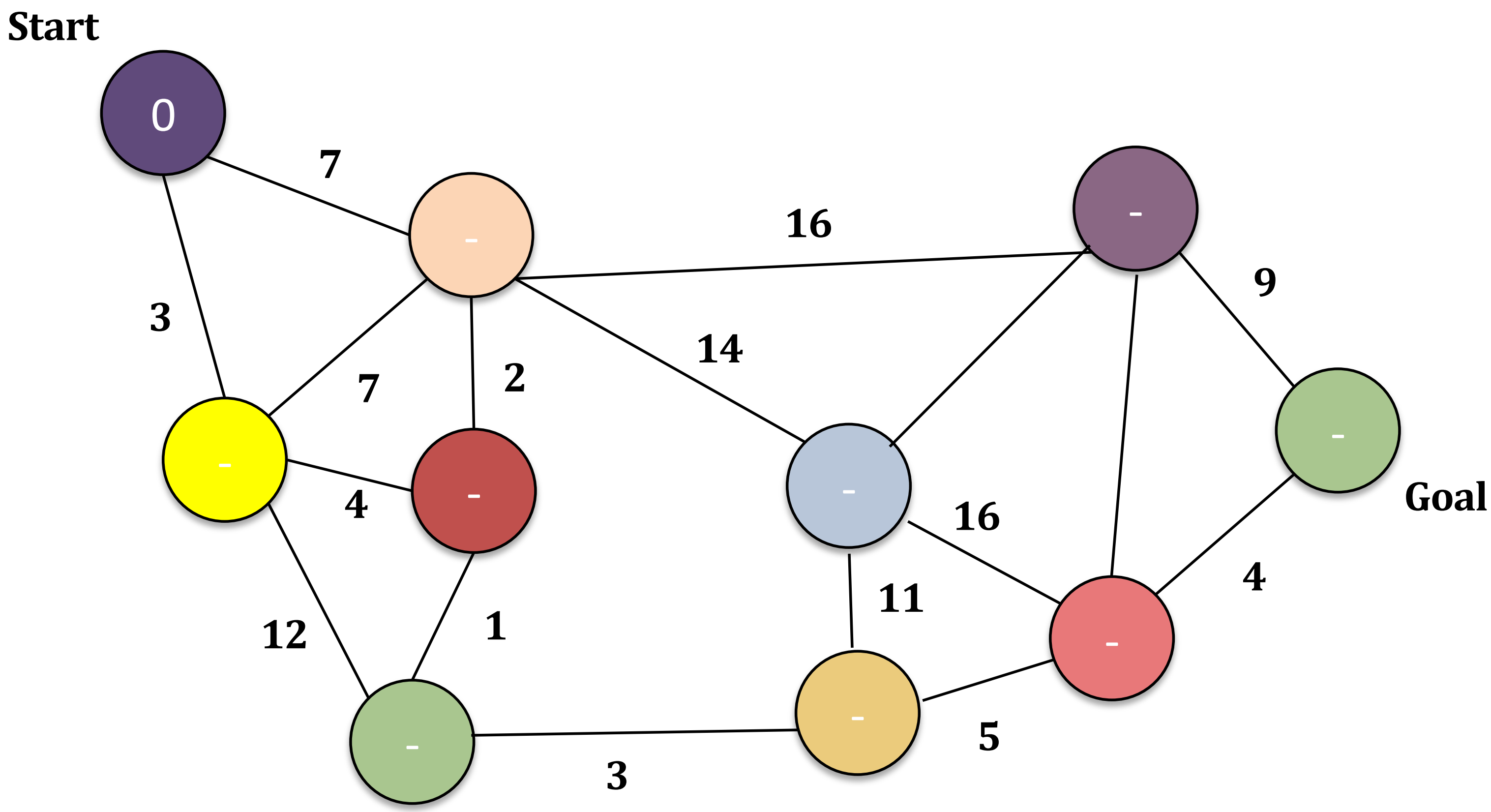
*+ High-level reasoning*
*+ Discover tasks from reward*
*- Does not transfer to the real world*

# Learn how to create a map (SLAM)

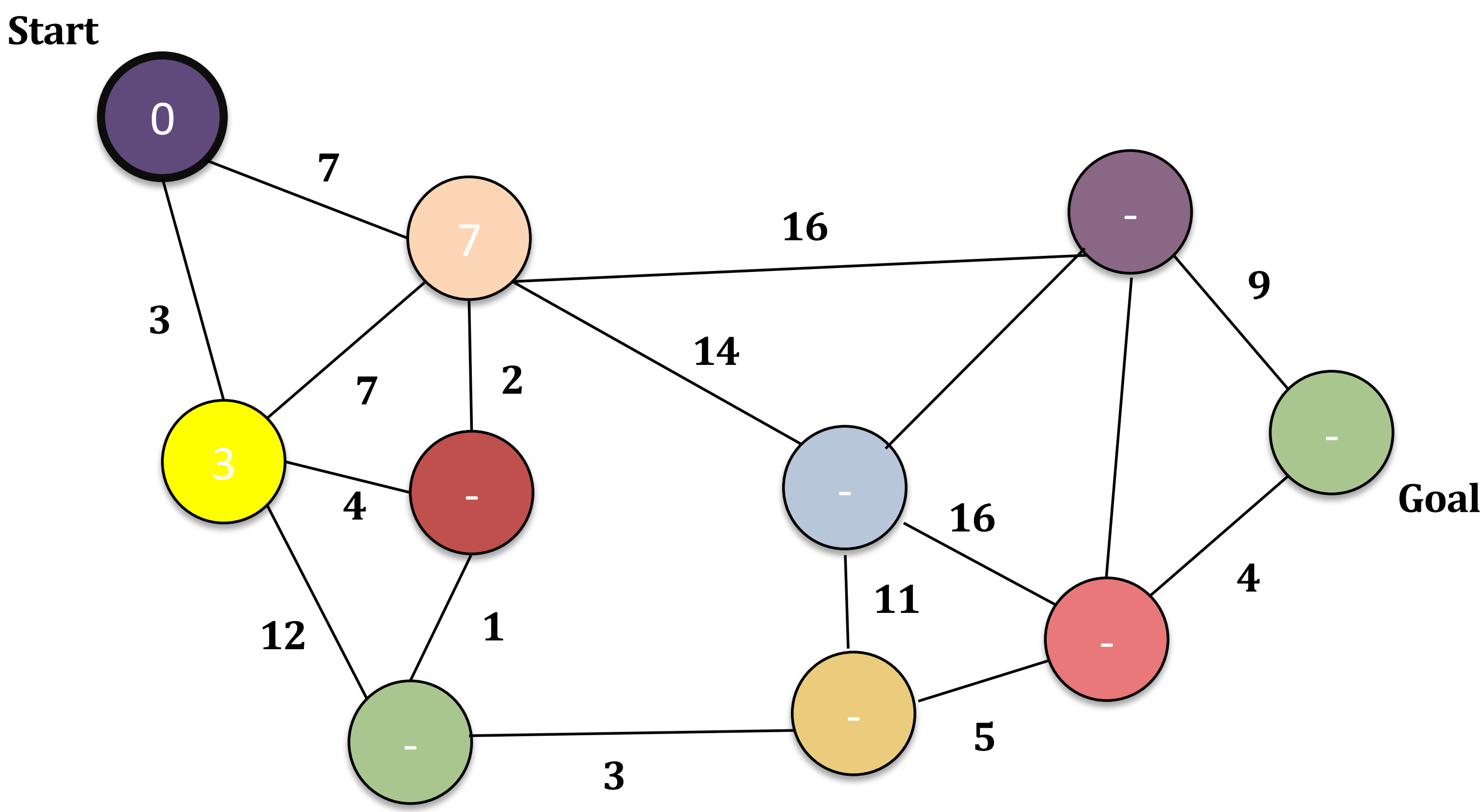



[Tateno, Tombari, Laina, Navab, CVPR 2017]

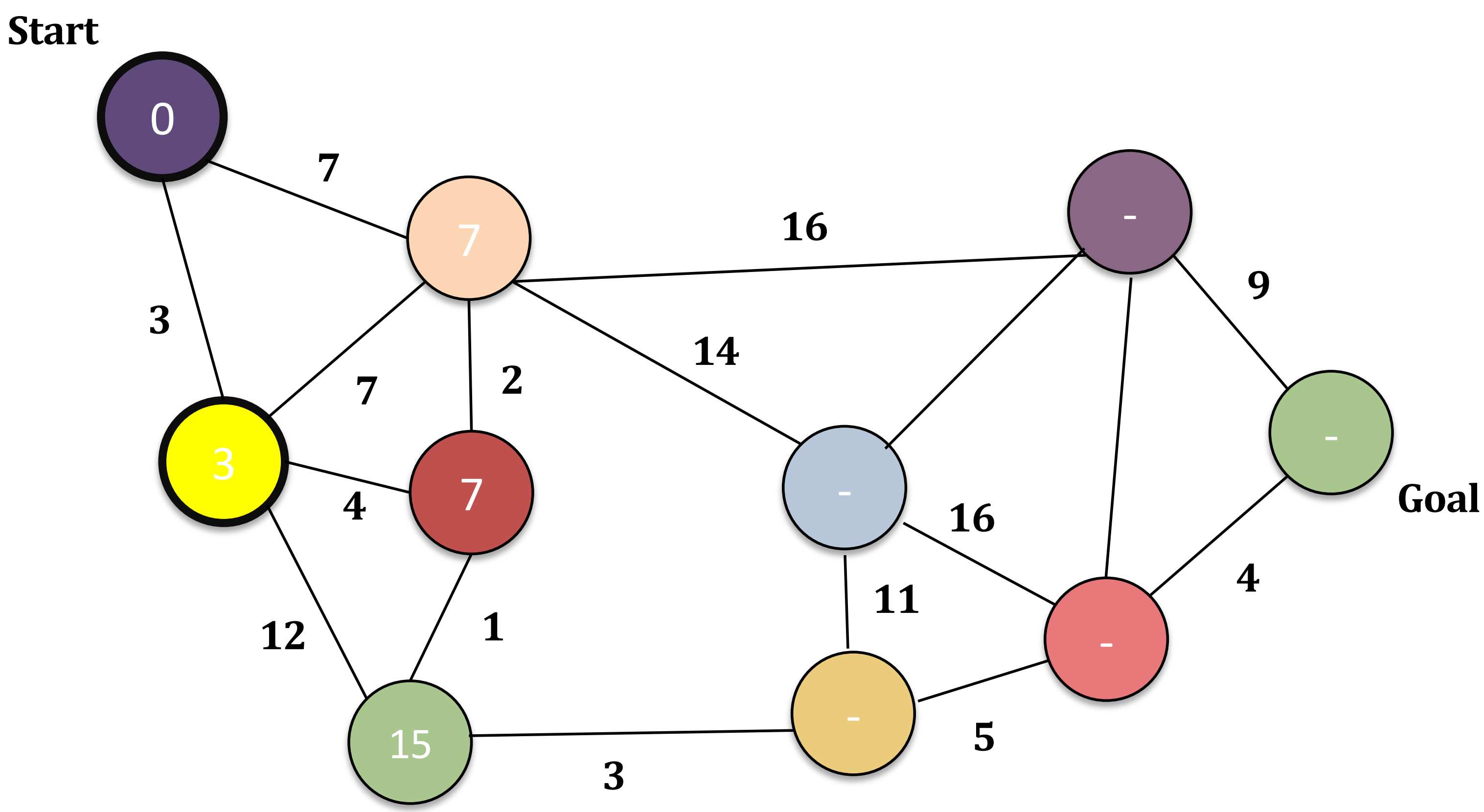
# Shortest path problems: Dijkstra's algorithm



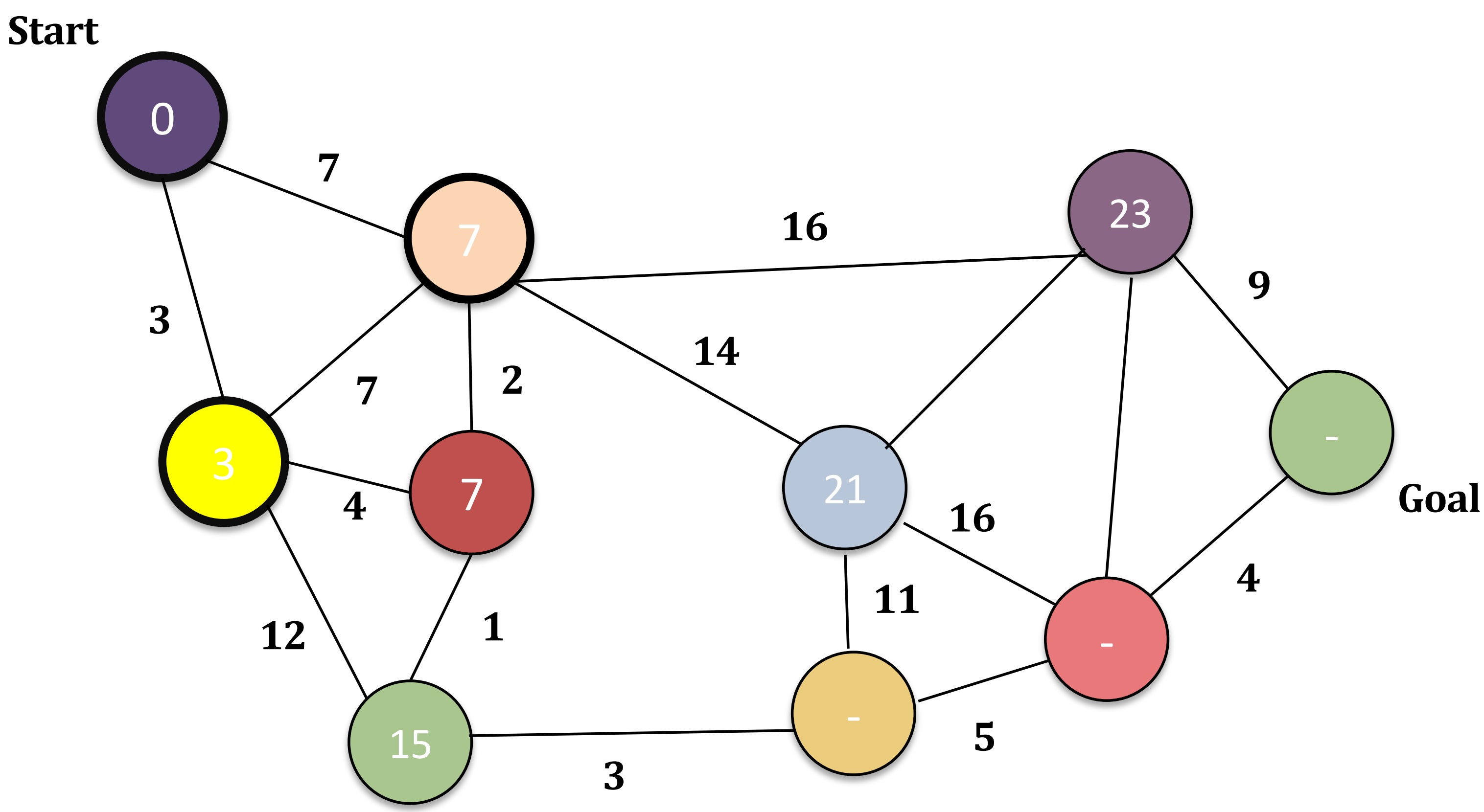
# Shortest path problems: Dijkstra's algorithm



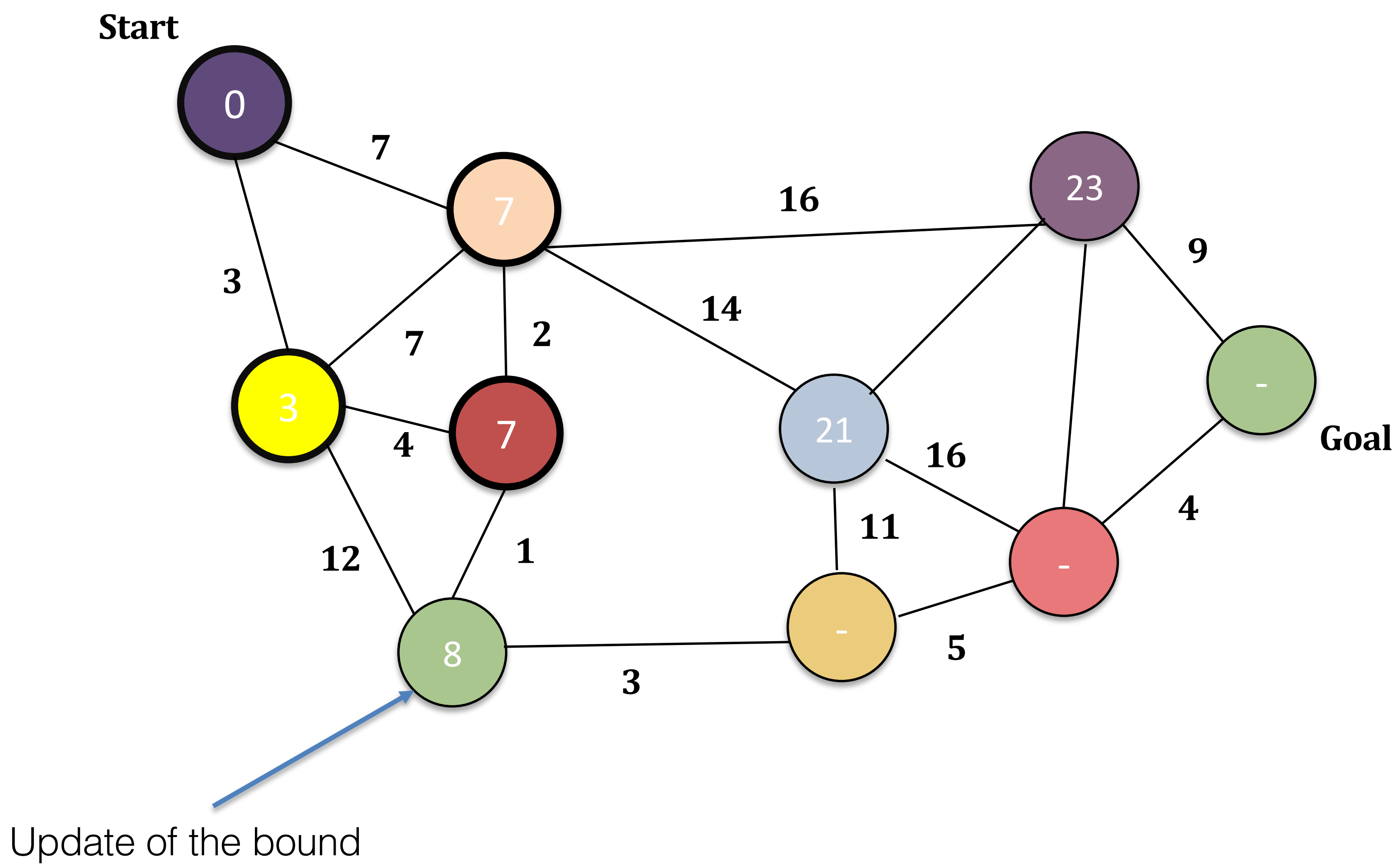
# Shortest path problems: Dijkstra's algorithm



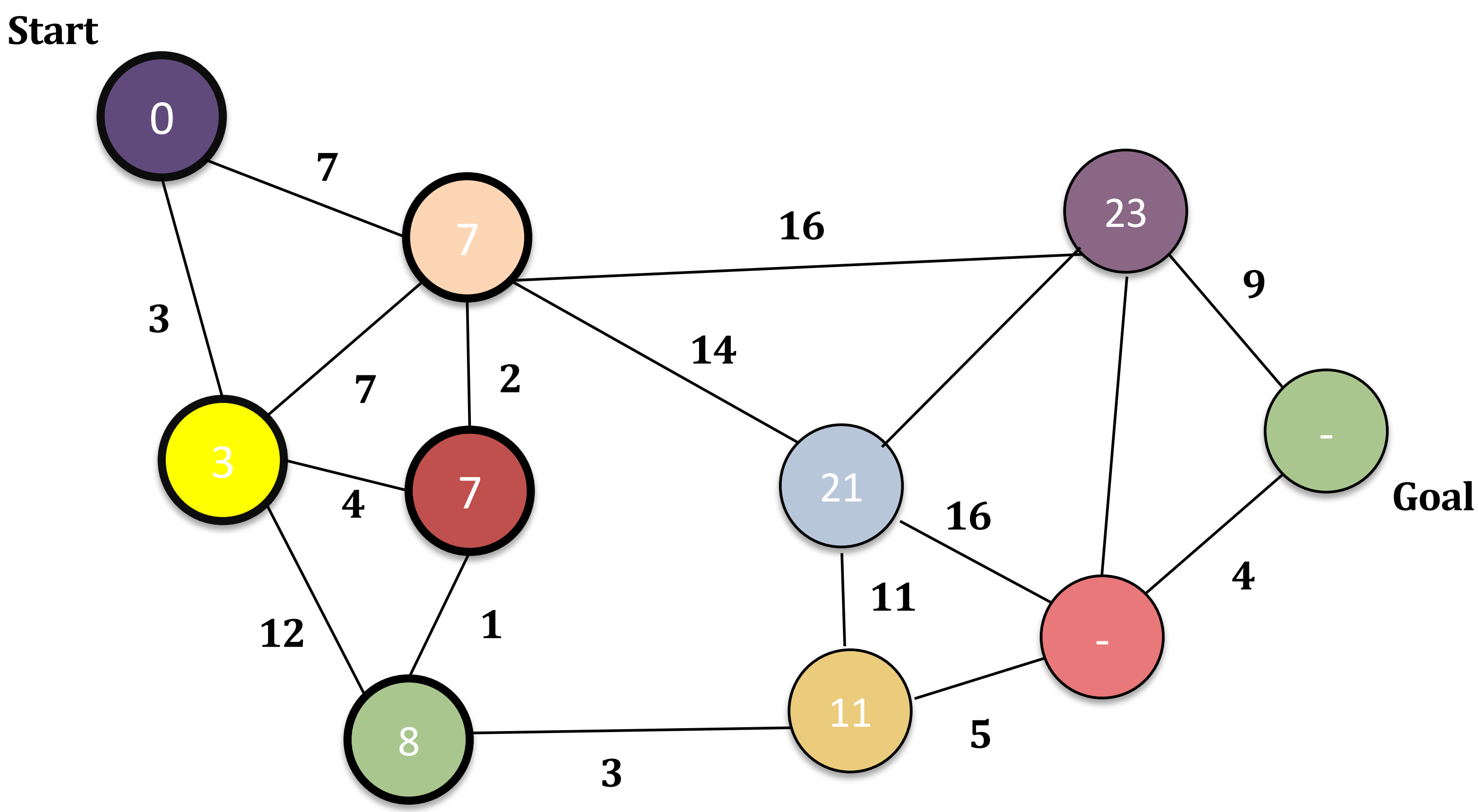
# Shortest path problems: Dijkstra's algorithm



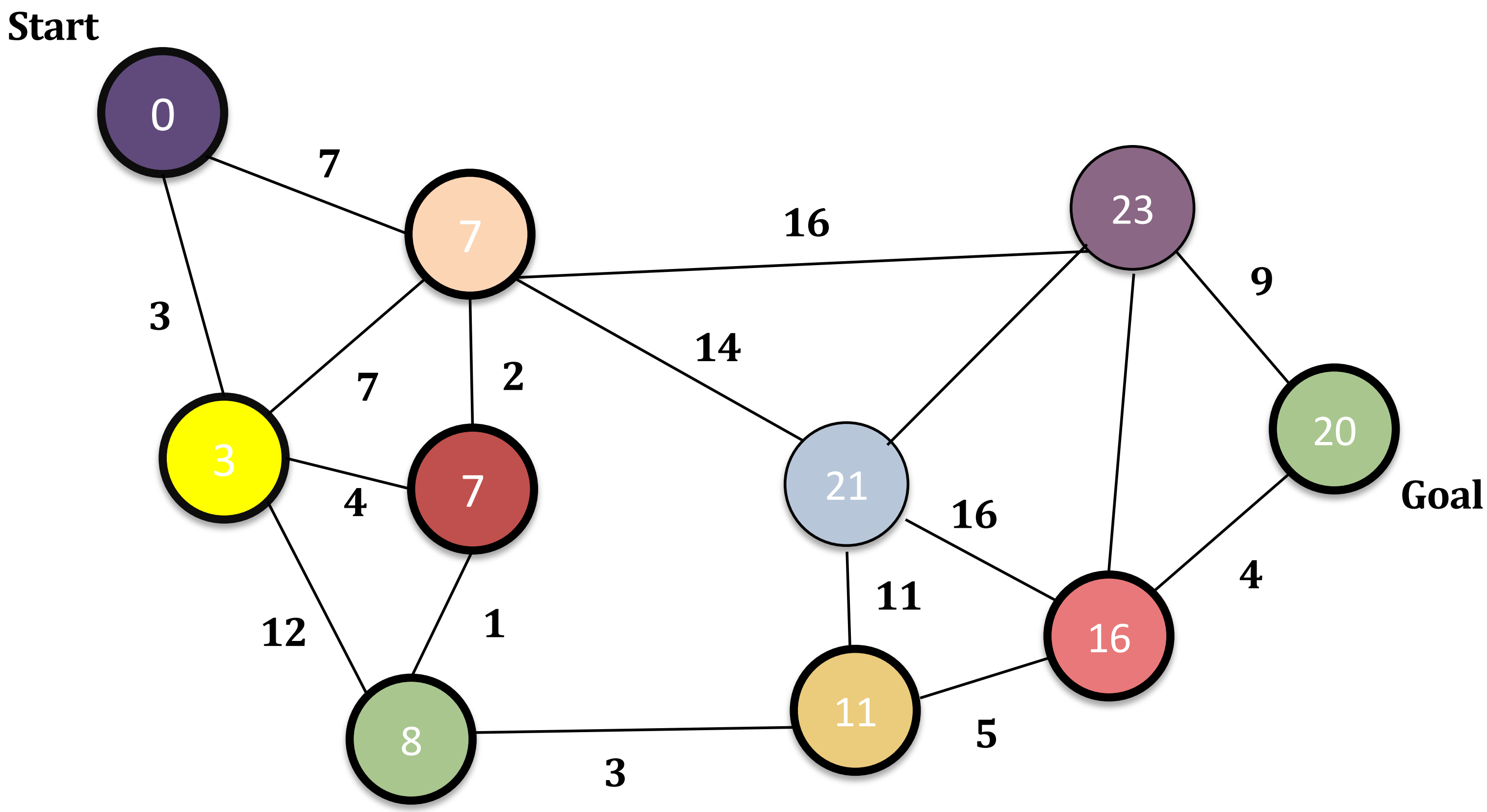
# Shortest path problems: Dijkstra's algorithm



# Shortest path problems: Dijkstra's algorithm



# Shortest path problems: Dijkstra's algorithm



# Active Neural Mapping

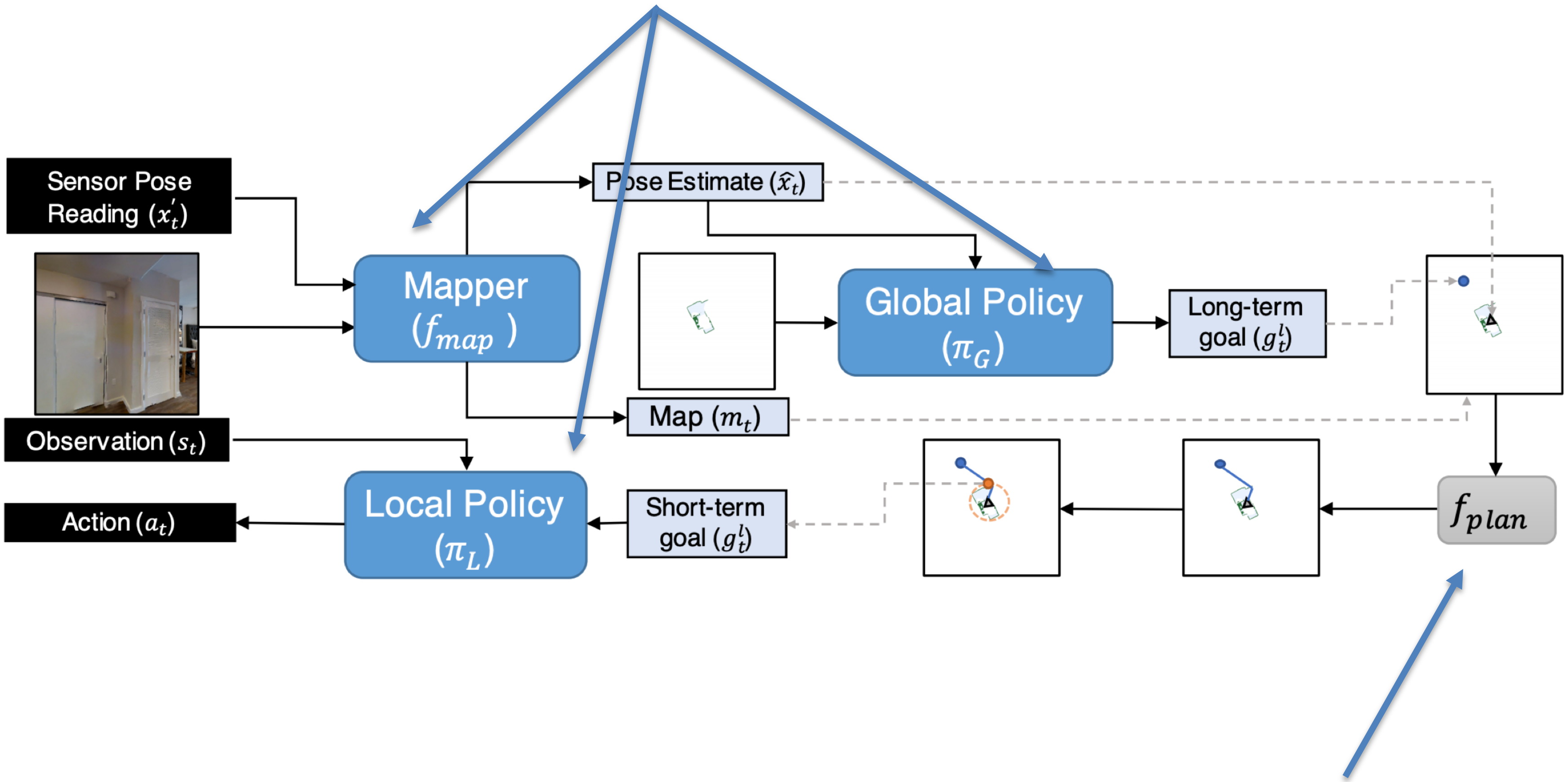
- Can we combine traditional path finding (Dijkstra, A*, front propagation etc.) with learning?
- Habitat AI Challenge (@CVPR 2019): Point Goal Task:
- Winner (Chapelot et al.):
  - A learned mapper predicts free space
  - A global policy is learned with RL (reward = coverage)
  - Planning with front-propagation
  - A local policy predicts navigation actions, learned with RL (reward=$L_2$ distance to global goal).

  - No end to end training!
  - Easy transfer from coverage objective to point goal (replace global policy by fixed one).

[Chapelot, Gupta, Gandhi, Gupta, Salakhutdinov, 2019 (unpublished)]

# Active Neural Mapping

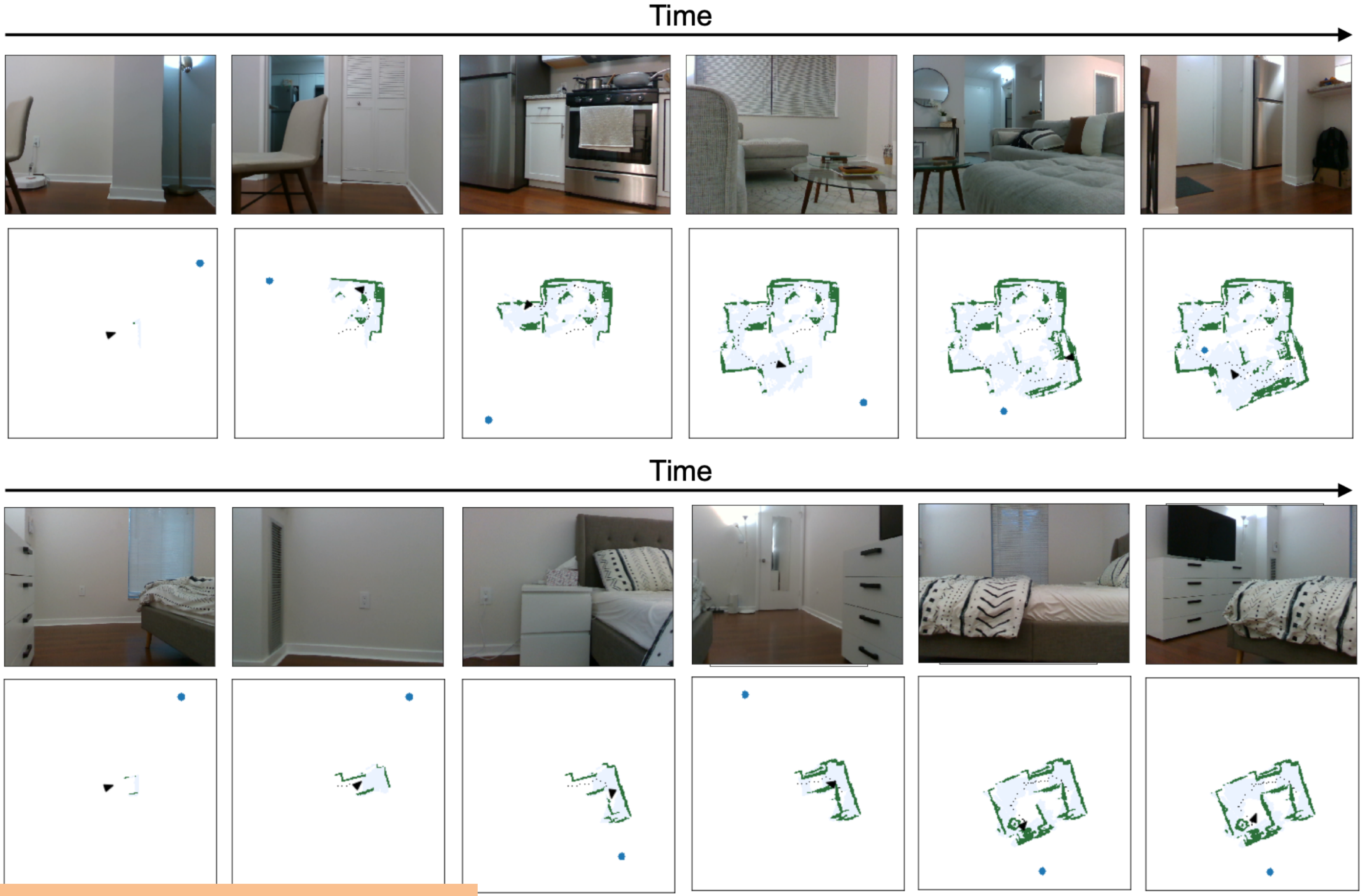
Deep networks trained with RL




Classical planning: shortest path, not traineable

[Chapelot, Gupta, Gandhi, Gupta, Salakhutdinov, 2019 (unpublished)]

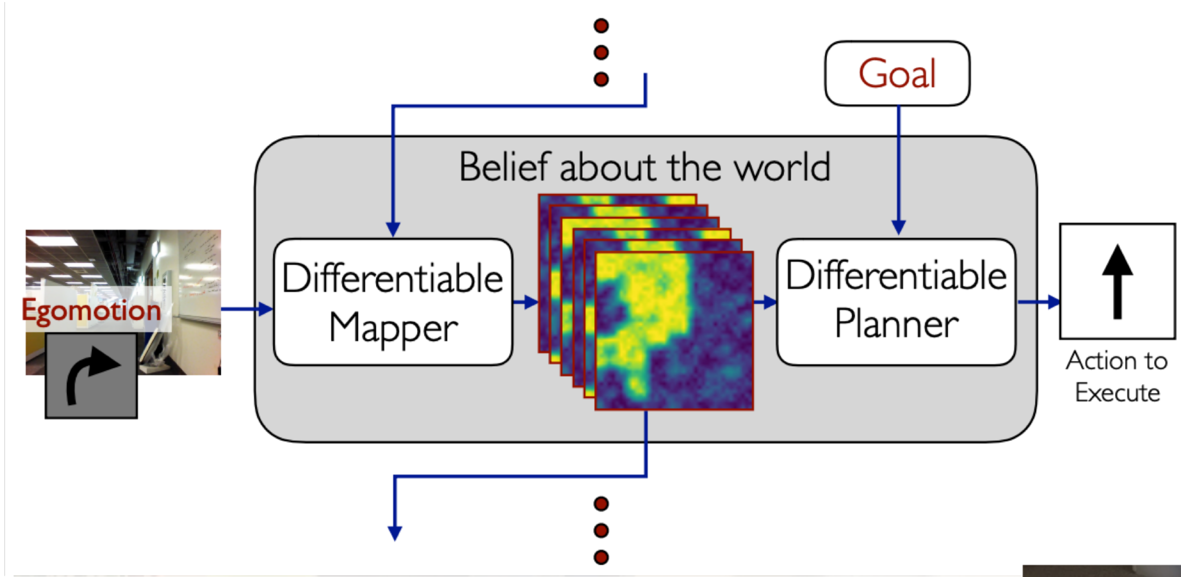
# Active Neural Mapping




[Chapelot, Gupta, Gandhi, Gupta, Salakhutdinov, 2019 (unpublished)]

# Cognitive Mapping and Planning

- Differentiable planner (value iteration networks)
- End to end training, but no RL (imitation learning)



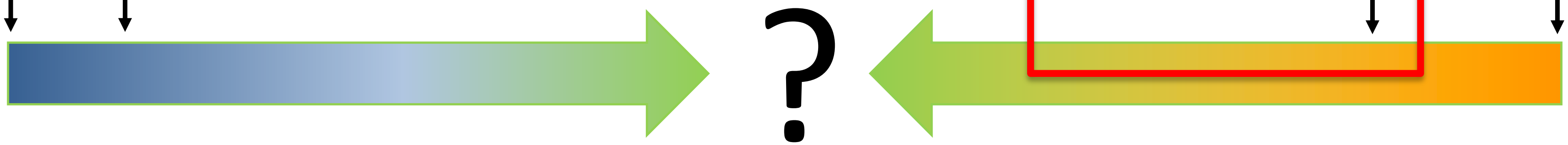

[Gupta, Tolani, Davidson, Levine, Sukthankar, Malik, CVPR 2017]

**Purely geometrical mapping + planning**

**Adding semantics**

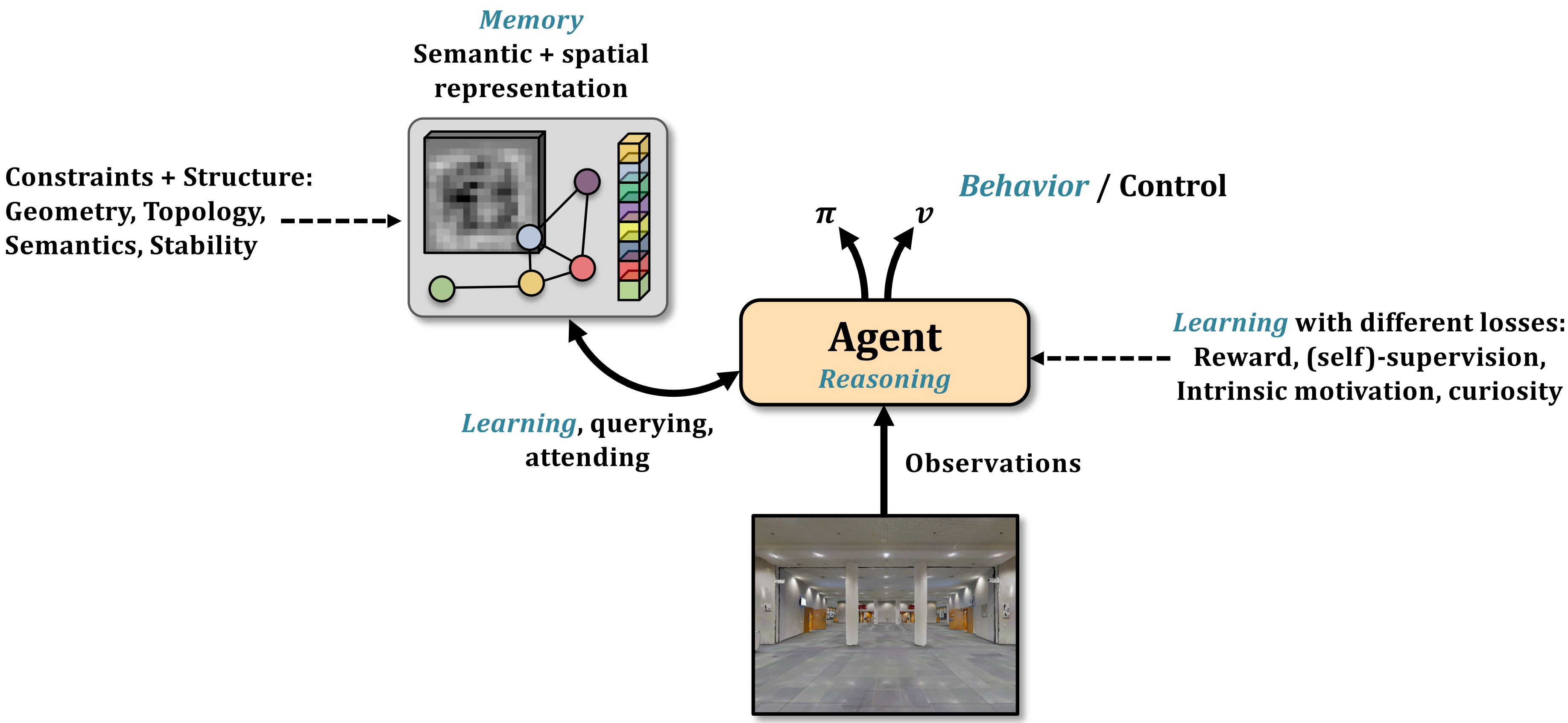
**Purely learned policies (Deep-RL)**

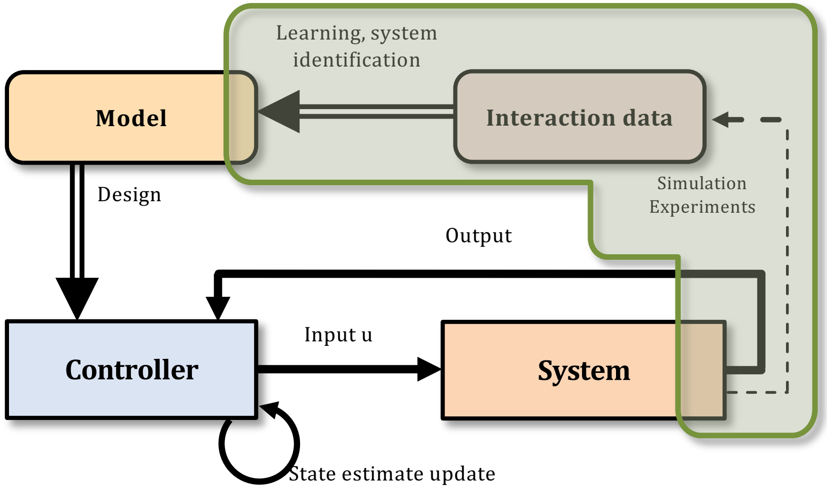
**Inductive biases: Geometry, topology, stability etc.**



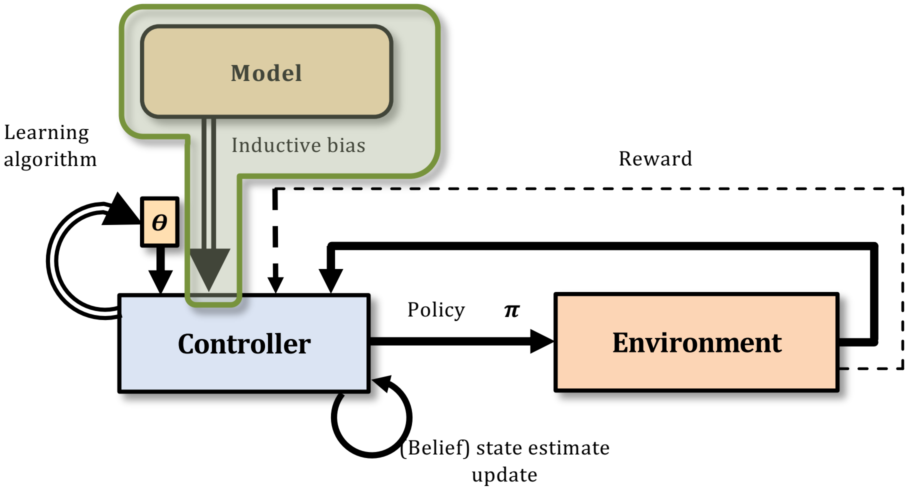

*+ Real world solutions*
*- Simple tasks and reasoning (eg. waypoint navigation)*

*+ High-level reasoning*
*+ Discover tasks from reward*
*- Does not transfer to the real world*


Memory
Semantic + spatial representation
Constraints + Structure:
Geometry, Topology,
Semantics, Stability
Behavior / Control
π
v
Agent
Reasoning
Learning with different losses:
Reward, (self)-supervision,
Intrinsic motivation, curiosity
Learning, querying,
attending
Observations




## Control Theory



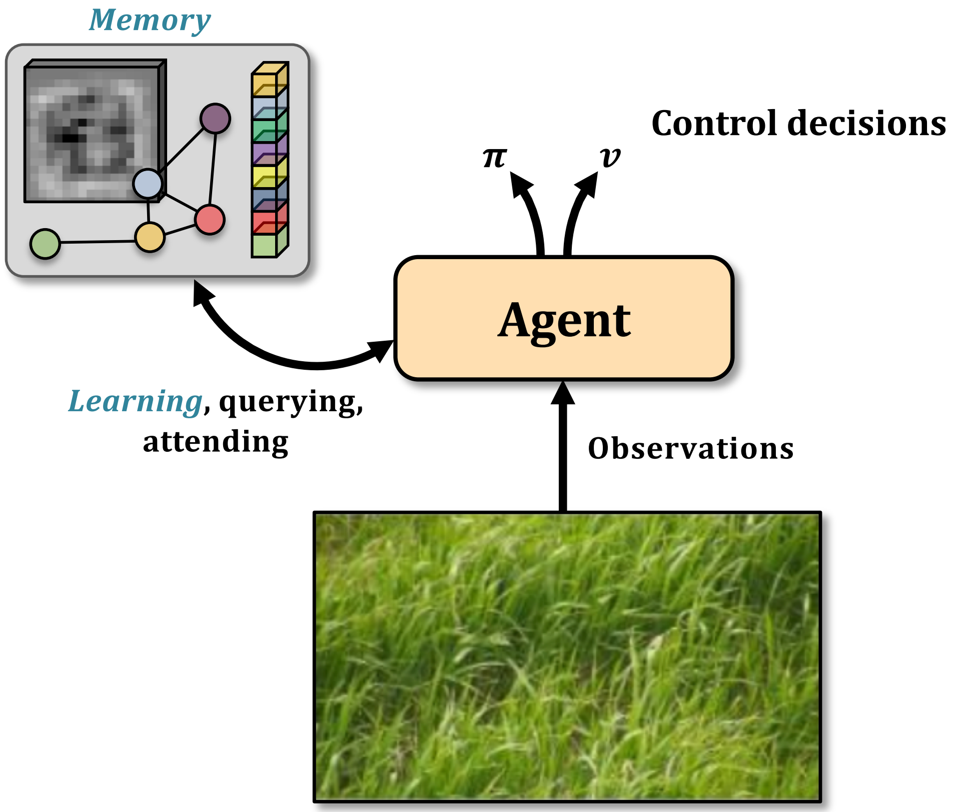

## Reinforcement Learning

# Computer vision from UAVs

Several problems:

- Computer vision for Control
- Vehicle Control
- Task related computer vision (eg. plant detetcion)

# End to end learning?

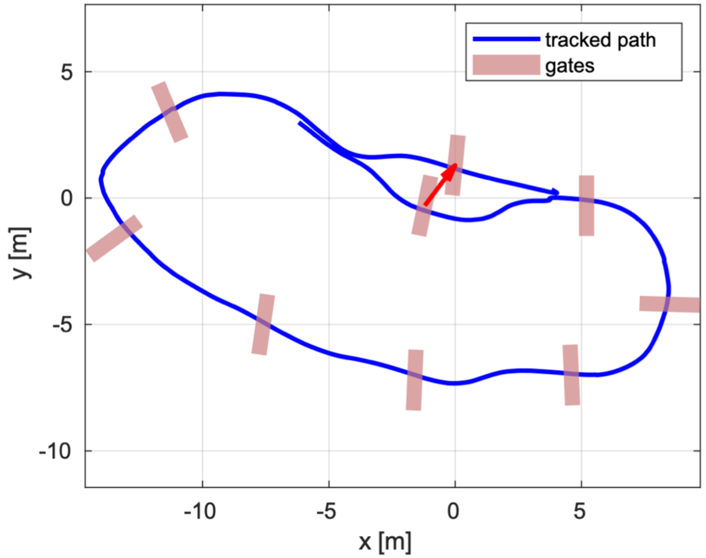



Requires a training strategy:

- imitation from human trajectories?
- Reinforcement Learning

Very low explainability

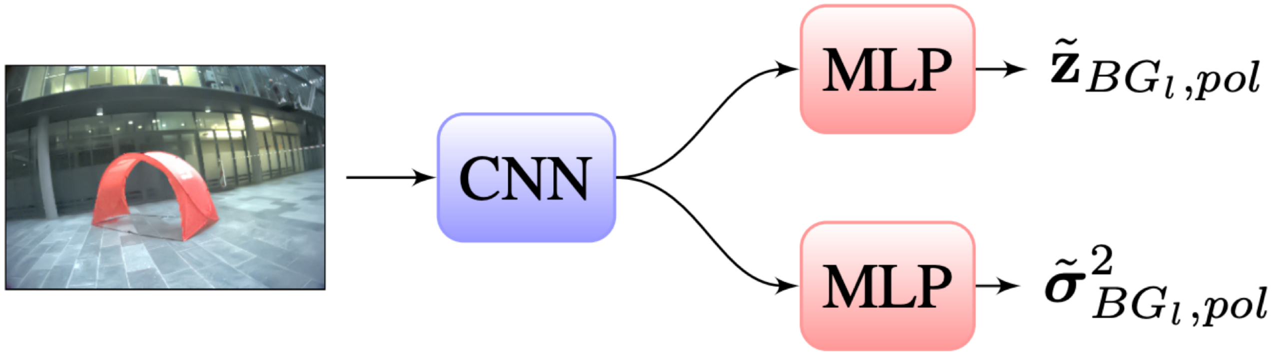
# The beauty and the beast


[Kaufmann, Gehrig, Foehn, Ranftl, Dosovitskiy2, Koltun, Scaramuzza1, 2018]

# The beauty and the beast

Vision / perception: DNNs for state estimation and obstacle / door detection.




[Kaufmann, Gehrig, Foehn, Ranftl, Dosovitskiy, Koltun, Scaramuzza, 2018]

# The beauty and the beast

Control: classical model predictive control

$$\min_{\mathbf{u}} \int_{t_0}^{t_f} \left(\bar{\mathbf{x}}_t^\top(t)\mathbf{Q}\bar{\mathbf{x}}_t(t) + \bar{\mathbf{u}}_t^\top(t)\mathbf{R}\bar{\mathbf{u}}_t(t)\right) dt$$

$$\bar{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}_r(t) \quad \bar{\mathbf{u}}(t) = \mathbf{u}(t) - \mathbf{u}_r(t)$$

$$\text{subject to} \quad \mathbf{r}(\mathbf{x},\mathbf{u}) = 0 \qquad \mathbf{h}(\mathbf{x},\mathbf{u}) \leq 0.$$

[Kaufmann, Gehrig, Foehn, Ranftl, Dosovitskiy, Koltun, Scaramuzza, 2018]

# Neural Lander

- A drone is controlled with a PID controller based on a model of the drone
- Unknown disturbances (unsteady airflow) is learned by a deep neural network and supervised learning

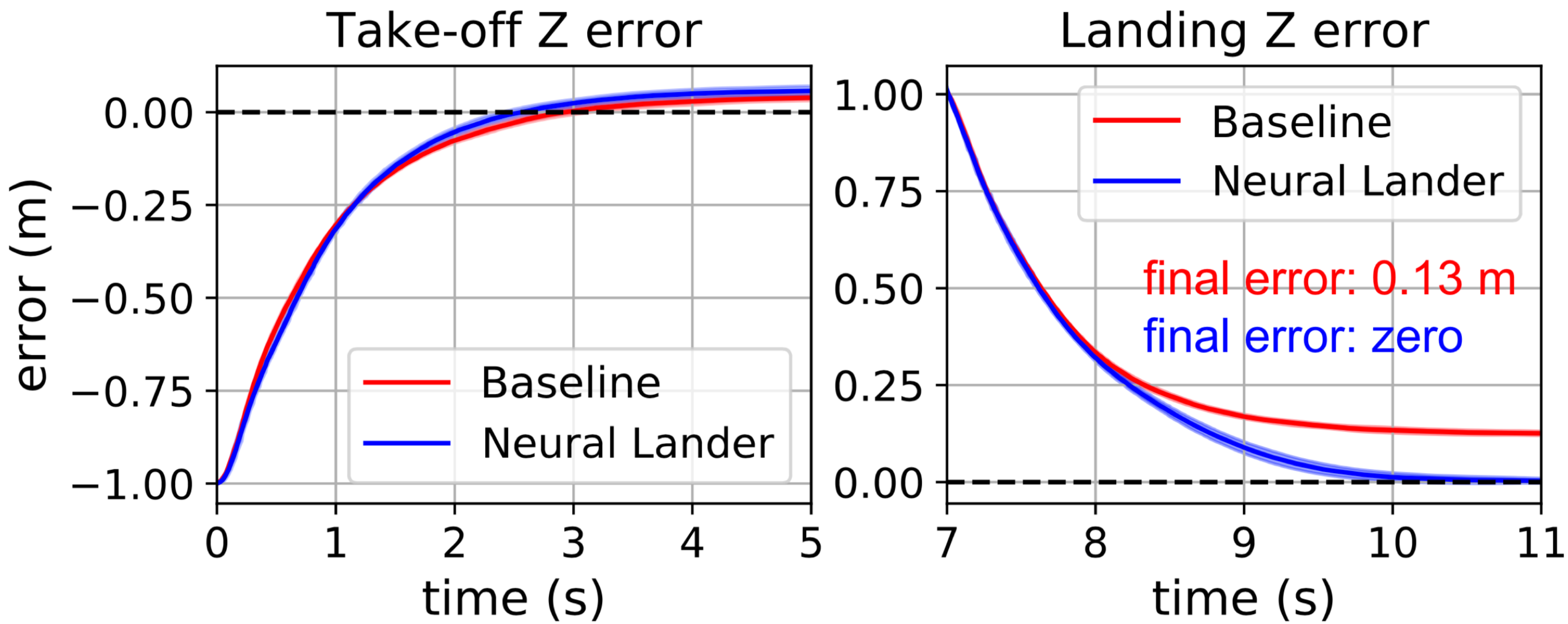
$$\dot{\mathbf{p}} = \mathbf{v}, \qquad m\dot{\mathbf{v}} = m\mathbf{g} + R\mathbf{f}_u + \boxed{\mathbf{f}_a,} \tag{1a}$$

$$\dot{R} = RS(\boldsymbol{\omega}), \qquad J\dot{\boldsymbol{\omega}} = J\boldsymbol{\omega} \times \boldsymbol{\omega} + \boldsymbol{\tau}_u + \boxed{\boldsymbol{\tau}_a,} \tag{1b}$$

Estimated from state and control inputs

[Shi et al., 2019]

# Neural Lander




[Shi et al., 2019]