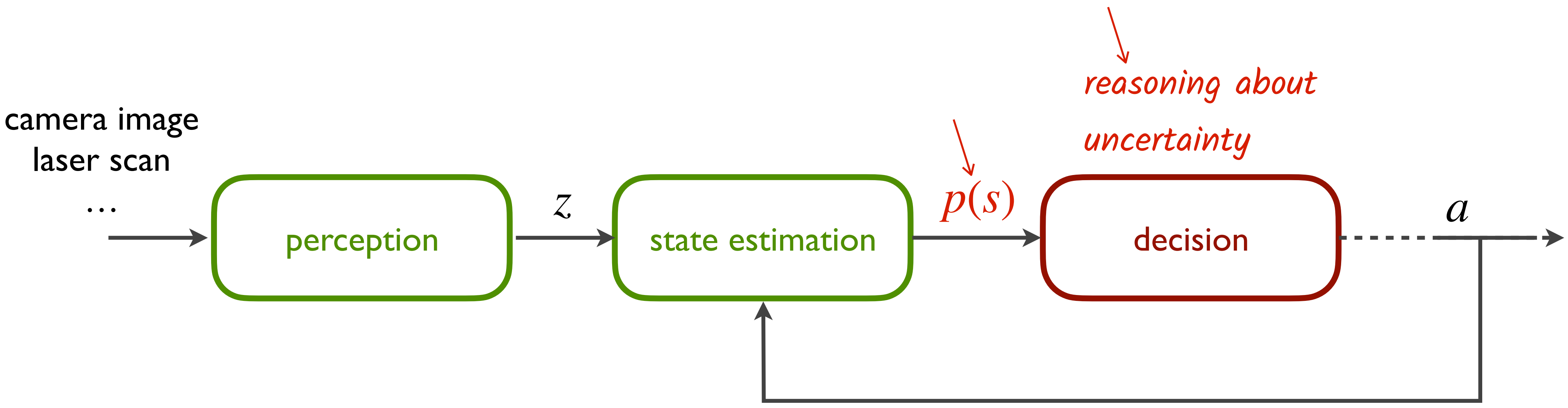
Last lectures ...



Partially observable Markov decision process. A (discrete) partially observable Markov decision process (POMDP) consists of the following basic elements:

- S is a set of states.
- A is a set of actions.
- Z is a set of observations.
- $T(s, a, s') = p(s' \mid s, a)$ is a probabilistic state-transition function.
- $M(a, s', z) = p(z \mid a, s')$ is a probabilistic observation function.
- $R(s,a)$ is a reward function.
- b_0 is the probability distribution for the initial state.

MDP

HMM

Our objective is to find a sequence of actions $(a_0, a_1, a_2, \ldots)$ that maximize the value (expected total reward):

$$V = \mathrm{E}\left[\sum_{t=0}^{N-1} R(s_t, a_t) \,\middle|\, b_0\right] \quad \text{for finite-horizon tasks,}$$

$$V = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, b_0\right] \quad \text{for discounted infinite-horizon tasks.}$$

We solve for

- a open-loop policy $a_t = \pi(t)$ at each time step t or
- a closed-loop policy $a_t = \pi(b_t)$, where b_t is the belief, i.e., the probability distribution over the robot state at time t.

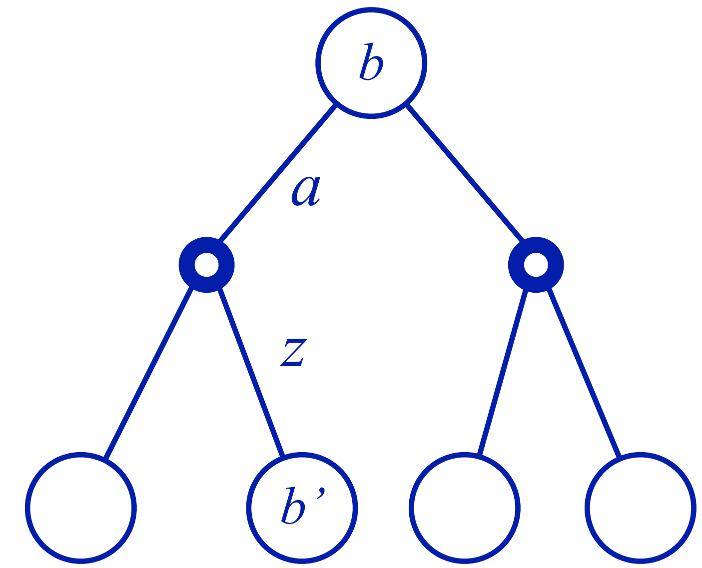
Question. An MDP closed-loop policy has the form $a_t = \pi(s_t)$. How does the POMDP closed-loop policy differ and why?

Belief-space MDP. To solve for a closed-loop policy, one idea is to convert a POMDP into a belief MDP. Given a POMDP, we construct an equivalent MDP with

- Each belief state $b \in \mathscr{B}$, where $\mathscr{B}$ is the set of all beliefs
- Each action $a \in A$, where A is the set of POMDP actions
- State-transition function $T(b, a, b')$
- Reward function $R(b, a)$

By conditioning on z, we have

$$T(b, a, b') = p(b'|b, a) = \sum_{z \in Z} p(b'|b, a, z) p(z|b, a)$$



What is $p(b' \mid b, a, z)$? Suppose that a robot, with initial belief b, takes action a and receives observation z. Let $\tau(b, a, z)$ denote the new belief as a result of Bayesian filtering, i.e., prediction update with action a and observation update with observation z. If $b' = \tau(b, a, z)$, then $p(b' \mid b, a, z) = 1$ and $T(b, a, b') = p(z \mid b, a)$. Otherwise, $p(b' \mid b, a, z) = 0$, and $T(b, a, b') = 0$

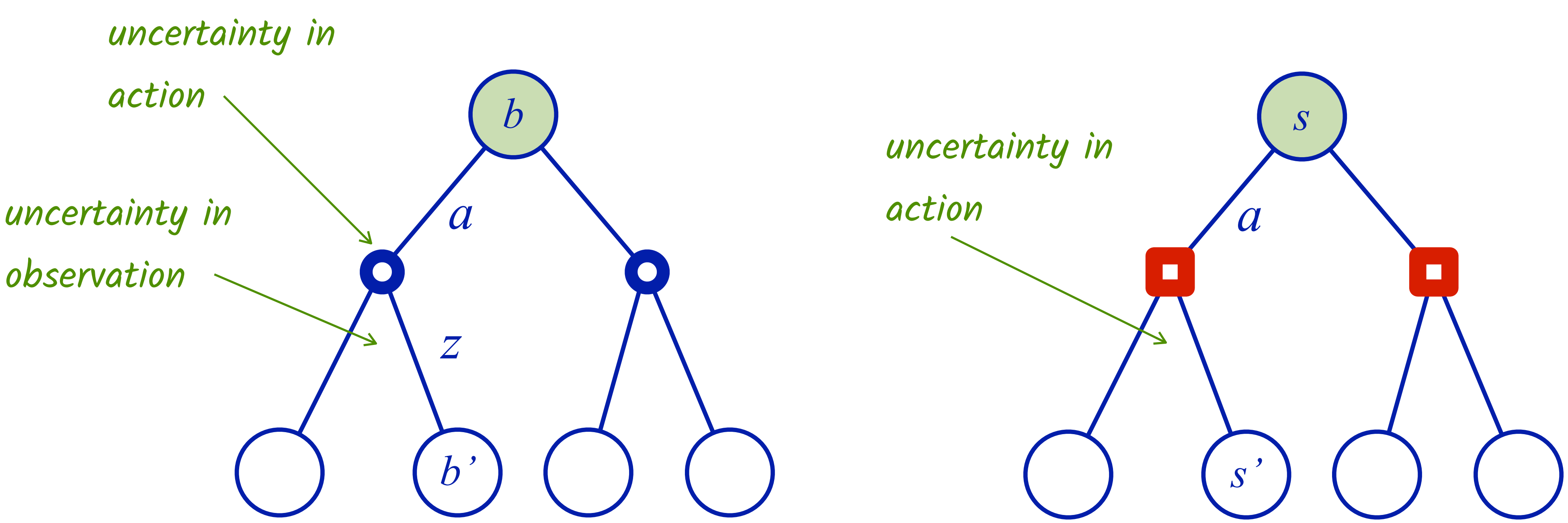
Similarly, we calculate $p(z \mid b, a)$ by conditioning and by applying the definitions:

$$\begin{aligned} p(z|b, a) &= \sum_{s' \in S} p(z|b, a, s') p(s'|b, a) && \textit{condition on the resulting state } s' \textit{ after action } a \\ &= \sum_{s' \in S} p(z|a, s') \sum_{s \in S} p(s'|s, a) p(s|b) && \textit{condition on the start state } s \\ & && z \textit{ is independent of } b, \textit{ given } s' \\ &= \sum_{s' \in S} M(a, s', z) \sum_{s \in S} T(s, a, s') b(s) && \textit{definition} \end{aligned}$$

For $R(b, a)$, we simply average the reward $R(s, a)$ weighted by the belief b:

$$R(b, a) = \sum_{s \in S} R(s, a) b(s)$$

Compare this belief MDP and the standard MDP:



Belief-space MDP

State-space MDP

Now it seems straightforward to apply value iteration:

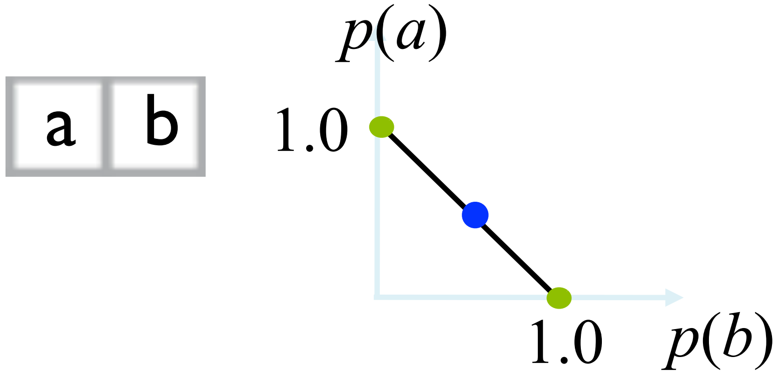
$$V_t(b) = \max_{a \in A} \Big\{ R(b, a) + \gamma \sum_{b' \in \mathcal{B}} T(b, a, b') V_{t-1}(b') \Big\}$$

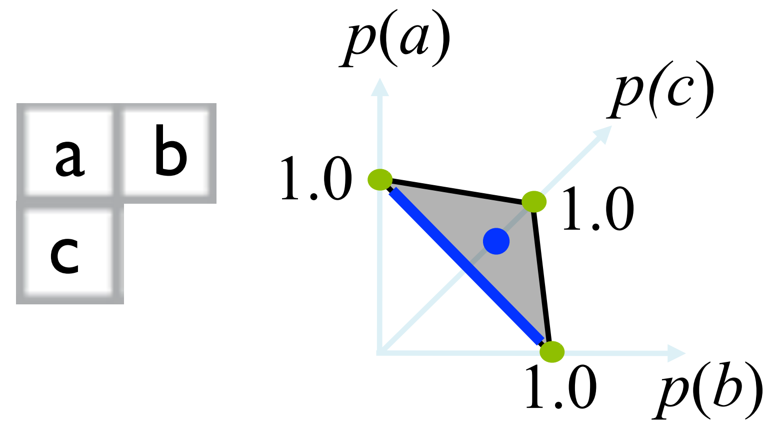
This is conceptually correct, but is it practical algorithmically?

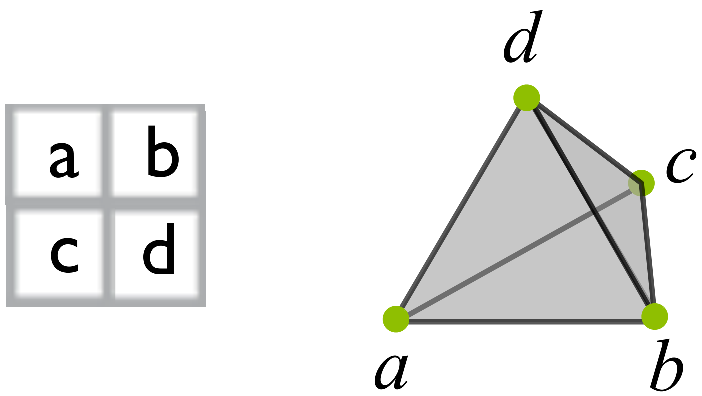
There are two main difficulties: policy representation and belief space size.

Policy representation. We can represent an MDP policy π: $S \rightarrow A$ as a table, at least, when S is small. How can we represent a POMDP policy $\pi: \mathscr{B} \rightarrow A$, as $\mathscr{B}$ is continuous? Maybe we should not attempt to represent π explicitly. We simply compute $\pi(b)$ on the fly, given b.

Belief space size. Consider the geometry of $\mathscr{B}$.







⋮ ⋮

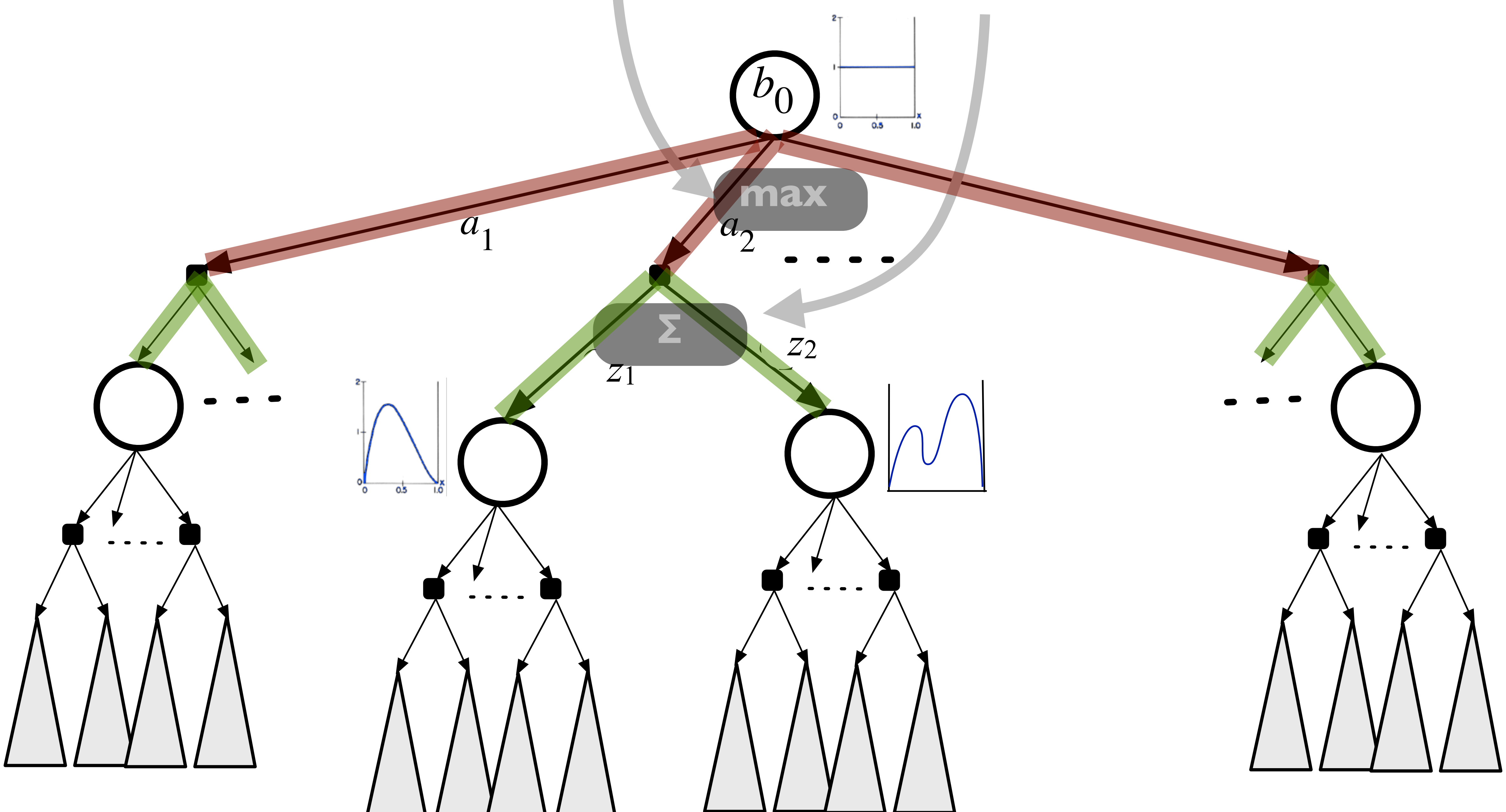
$|S|$ states $(|S|-1)$-dim simplex

$\mathscr{B}$ seems enormous. Value iteration, which is based on the idea of dynamic programming, would never work ,as it tries to process the entire space $\mathscr{B}$. Maybe we should consider forward search instead.

Each node of the search tree represents a belief b. At each node, the tree branches on the actions, and for each action, the tree further branches on the observations. A node b' is a child of node b, if and only if $b' = \tau(b, a, z)$.

same as $\Sigma_{b'} T(b, a, b')$ but more compact

$$V_t(b) = \max_{a \in A}\left\{R(b, a) + \gamma \sum_{z \in Z} p(z|b, a) V_{t-1}(b')\right\}$$



Example (tiger).

- A girl stands in front of two closed doors. She has 3 actions: open the left door (OL), open the right door (OR), or listen (LS).
- One door leads to a tiger, thus having a large penalty (-100). The other door leads to a reward (+10). The cost of LS is -1. The discount factor is 0.95.
- There are two possible observations as a result of listening: tiger on the left (TL) or tiger on the right (TR). The observation is correct with probability 0.85.
- After a door is opened, the game resets. The tiger goes behind one of the door with equal probabilities.

Suppose that the girl has initial belief b and can take only one action. What is her best action?

	$b = (0.5, 0.5)$	$b = (1, 0)$
LS	✓ $(-1) \times 0.5 + (-1) \times 0.5 = -1$	$(-1) \times 1 + (-1) \times 0 = -1$
OR	$10 \times 0.5 + (-100) \times 0.5 = -45$	✓ $10 \times 1 + (-100) \times 0 = 10$
OL	$-100 \times 0.5 + 10 \times 0.5 = -45$	$-100 \times 1 + 10 \times 0 = -100$

In general, we can calculate the value of a 1-step policy with initial belief $(b, 1 - b)$.

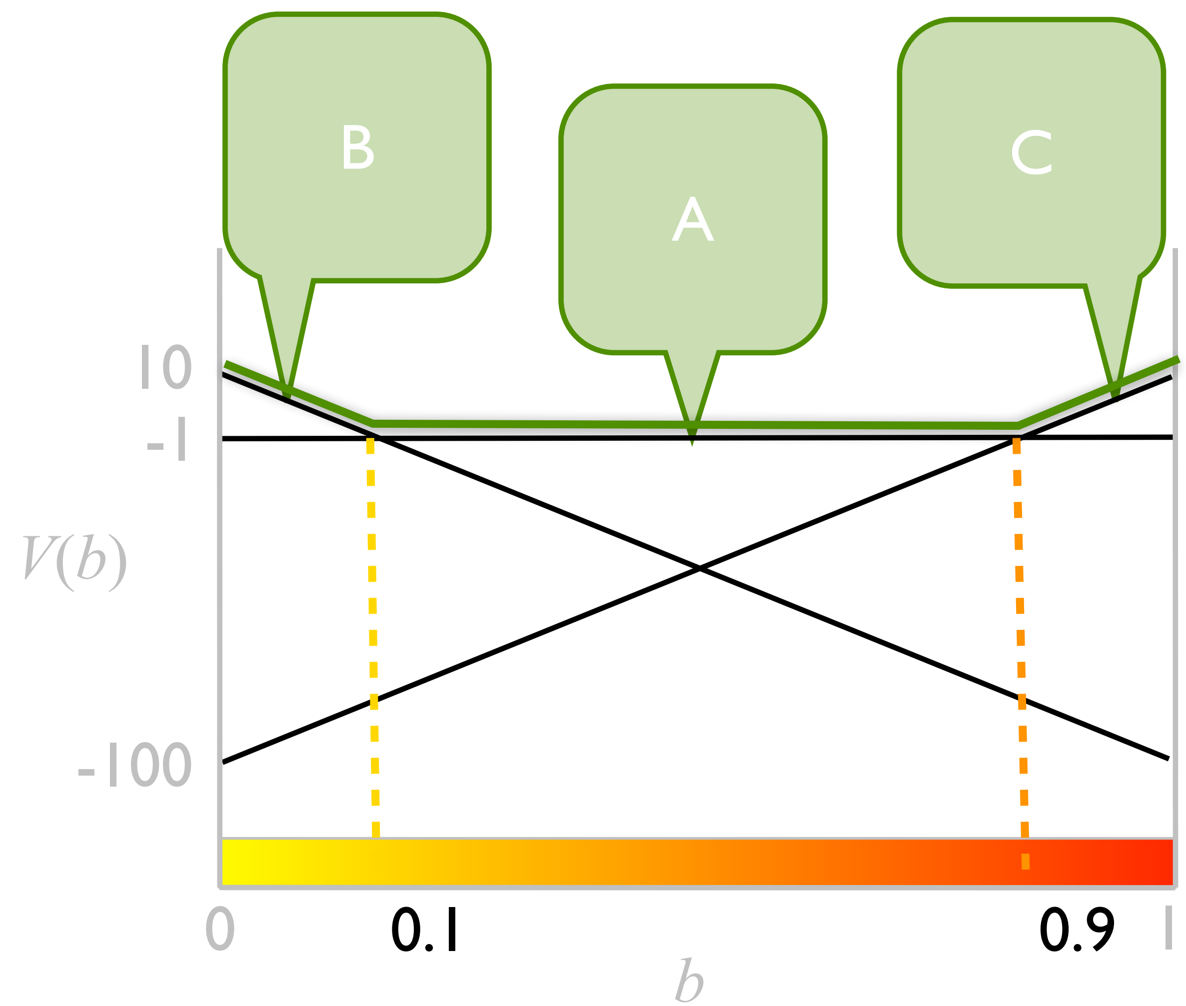
A: LS [0.1, 0.9] $V(b) = -1b - 1(1-b) = -1$

B: OL [0,0.1] $V(b) = -100b + 10(1-b)$

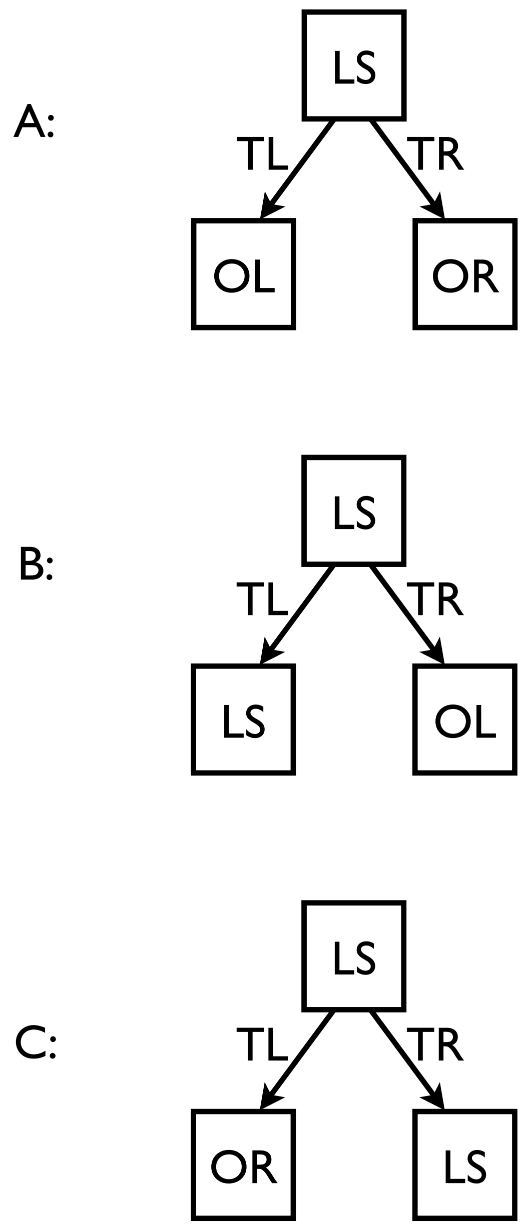
$= -110b + 10$

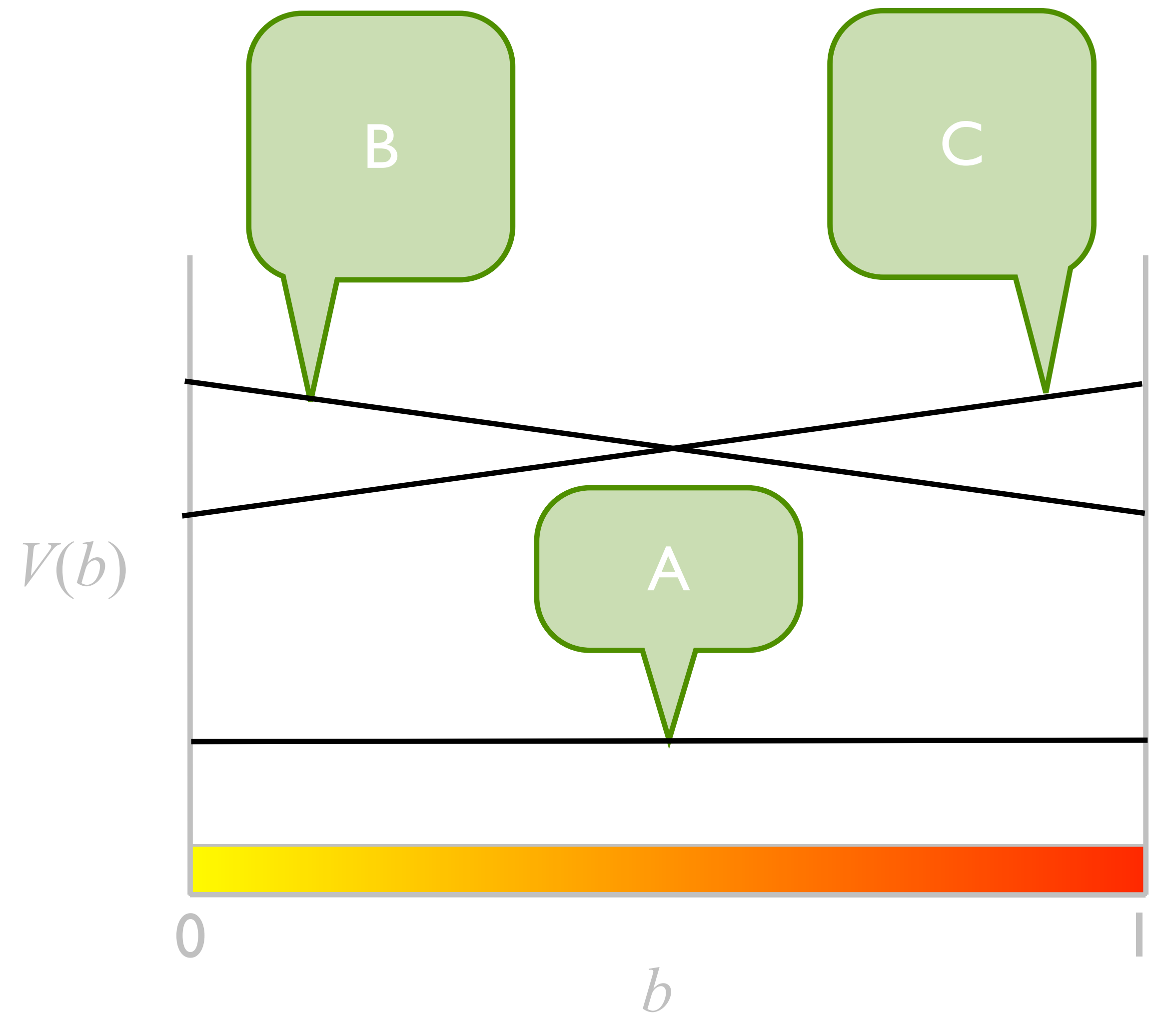
C: OR [0.9, 1] $V(b) = 10b - 100(1-b)$

$= 110b - 100$



Using the 1-step policies and their value functions, we can build the 2-step policies recursively.

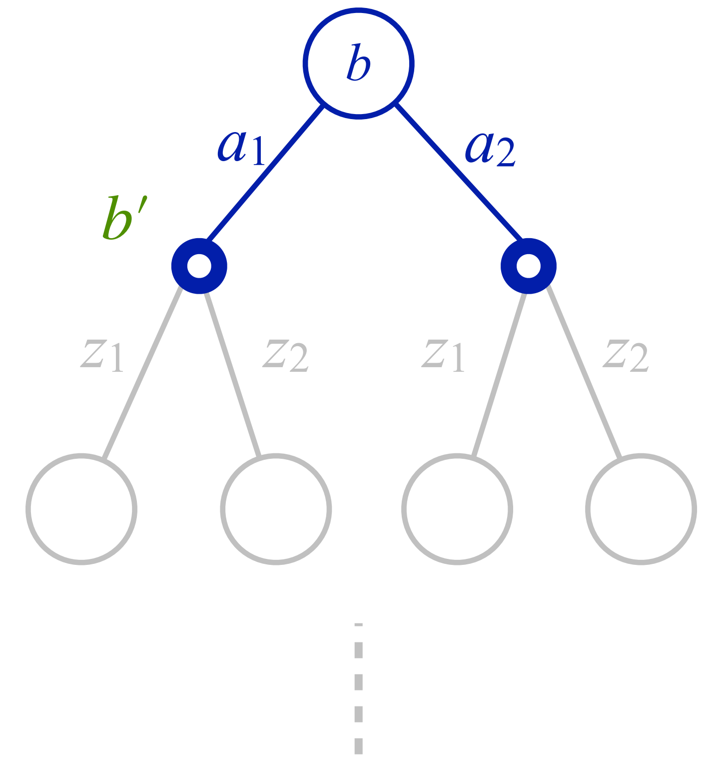




Question. Will we ever act according to policy A?

QMDP. The QMDP algorithm drastically cuts the search depth, using a heuristic approximation to the value function $V(b)$. Recall

$$V(b) = \max_{a \in A} \left\{ R(b,a) + \gamma \sum_{z \in Z} p(z \mid b,a) V(b') \right\}$$



QMDP truncates the tree after the very first action and ignores the rest, including the first observation. We then have

$$V(b) = \max_{a \in A} \left\{ R(b,a) + h(b') \right\}$$

What is a suitable heuristic $h(b)$? While it may be difficult to solve the POMDP, we can solve the corresponding MDP and obtain $V_{\text{MDP}}(s)$. Set $h(b) = \sum_{s \in S} b(s) V_{\text{MDP}}(s)$.

Question. What approximation does $h(b)$ make? Can you say anything about the relationship between $h(b)$ and $V(b)$?

To summarize the QMDP algorithm,

- Solve the MDP for $V_{\mathrm{MDP}}(s)$.
- For each action $a \in A$, calculate

uncertainty of s　　　uncertainty of actions

$$
\begin{aligned}
Q(b,a) &= R(b,a) + h(b') \\
&= \sum_{s \in S} \left\{ R(s,a)b(s) + \sum_{s' \in S} V_{\mathrm{MDP}}(s') \underbrace{p(s' \mid s,a)b(s)}_{\text{calculate } b'} \right\}
\end{aligned}
$$

- Finally,

$$
\begin{aligned}
\pi_{\mathrm{QMDP}}(b) &= \arg\max_{a \in A} Q(b,a) \\
V(b) &= \max_{a \in A} Q(b,a)
\end{aligned}
$$

QMDP reasons about the uncertainty of the current state as well as the uncertainties of actions. Effectively, it assumes that the state is fully observable after the first action and does not reason about the benefit of acquiring useful observations in the future. In this sense, it loses one main benefit of POMDP reasoning.

DESPOT. Intuitively, we can think of uncertainty as unknown future scenarios, e.g., unknown future action outcomes or unknown future observations. In a deterministic setting, we know or we can predict future action outcomes, as there is only one. In a probabilistic setting, we have many possible action outcomes weighted by probabilities and do not know *a priori* which one actually will occur. The challenge of decision making under uncertainty is the direct consequence of myriad future scenarios, resulting in exponential growth of the search tree.

Among the myriad scenarios, we may choose a subset of **representative** ones as the basis for decision making.

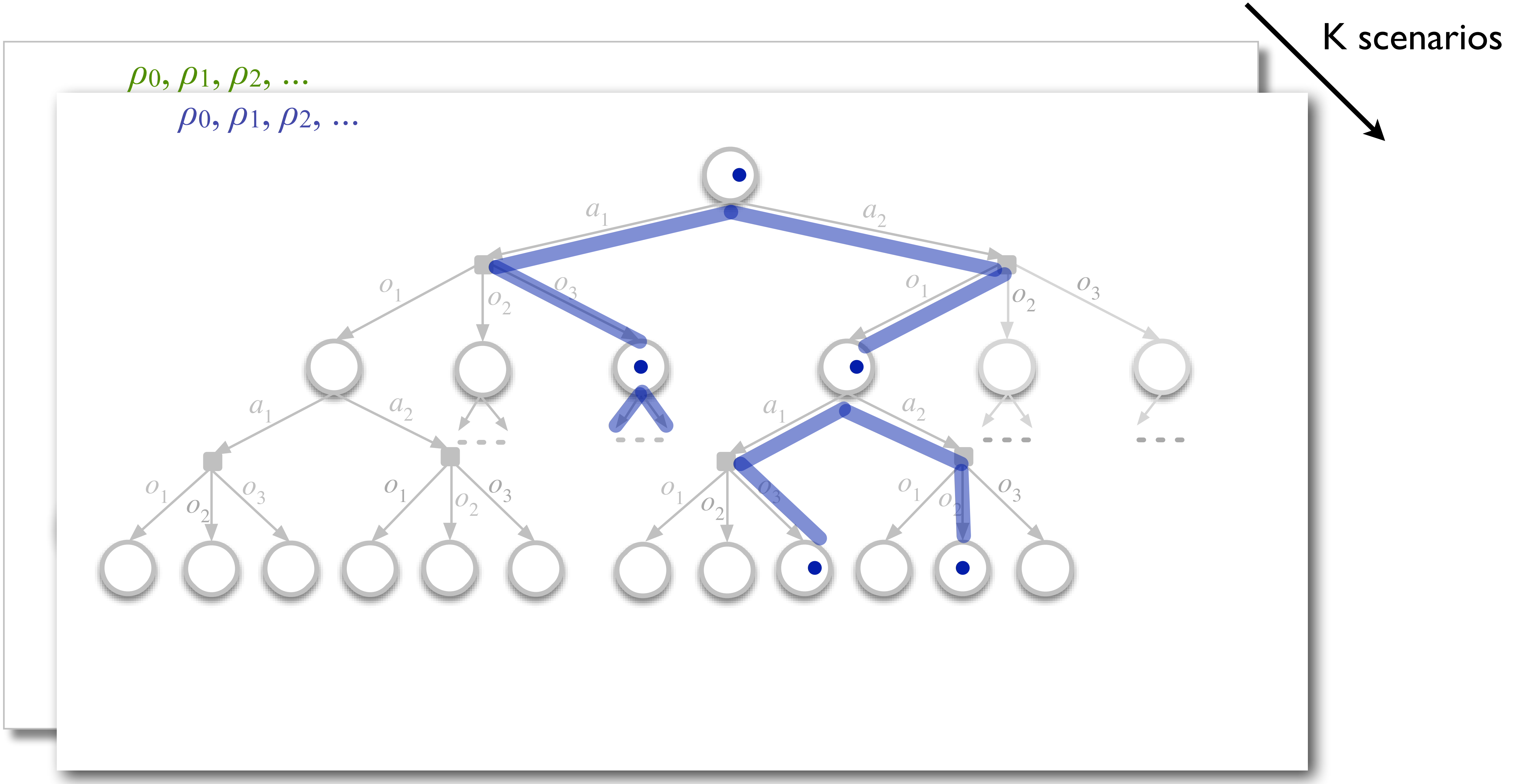
To generate a scenario, we perform "simulation". Given a sequence of actions $a_1, a_2, \ldots$:

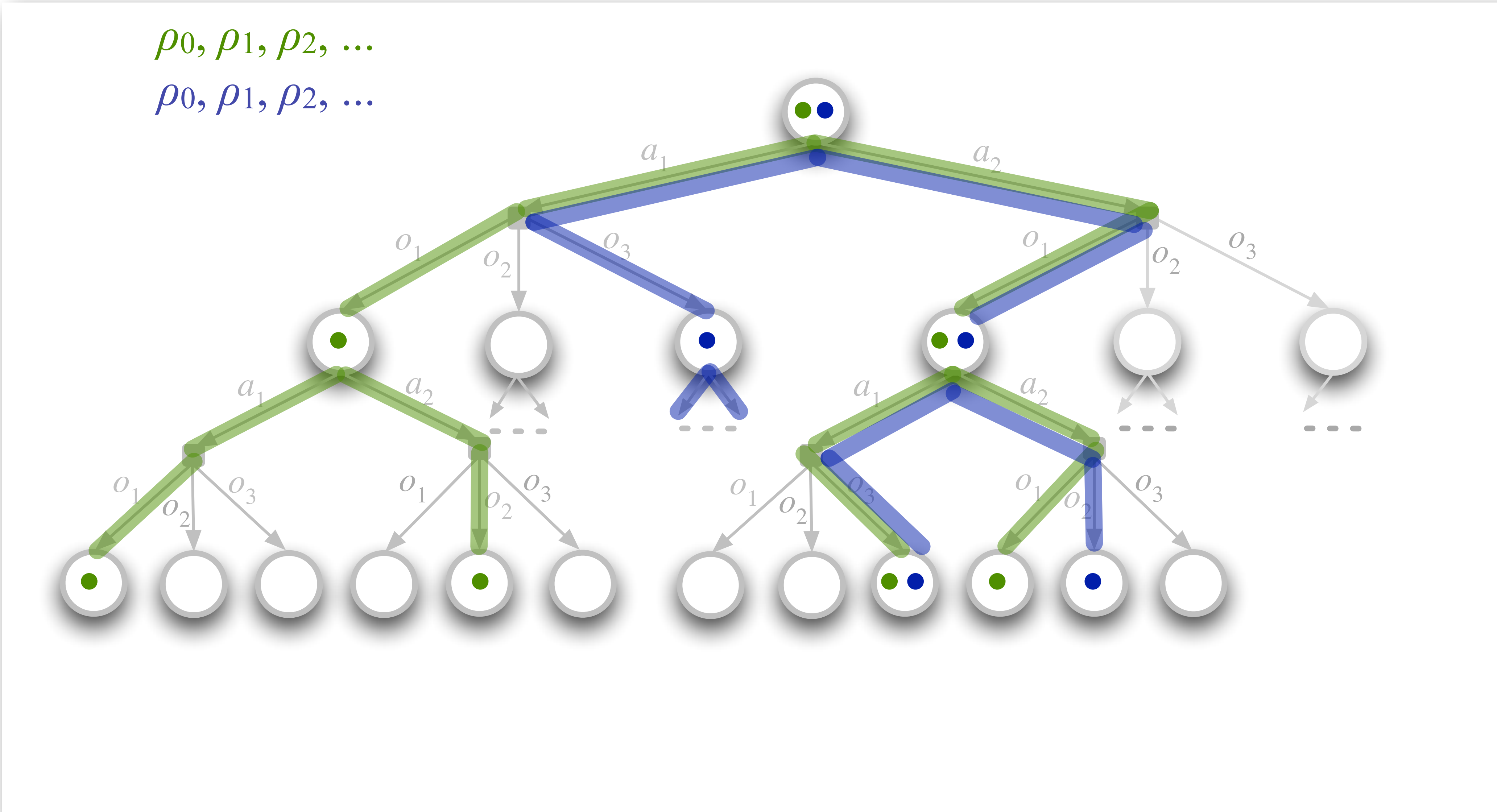
random sequence: $\rho_0, \rho_1, \rho_2, \rho_3, \rho_4, \ldots$

simulation trajectory: $s_0, a_1, s_1, o_1, a_2, s_2, o_2, \ldots$
"scenario"

belief tree path: $a_1, o_1, a_2, o_2, \ldots$

Repeat the above for all possible action sequences of length h.


K scenarios
$\rho_0, \rho_1, \rho_2, \ldots$
$\rho_0, \rho_1, \rho_2, \ldots$
a_1
a_2
o_1
o_2
o_3



N. Ye, A. Somani, D. Hsu, and W.S. Lee. **DESPOT: Online POMDP planning with regularization**. JAIR, 2017.

A full belief search tree of height h has size $O(|A|^h|Z|^h)$. The DESPOT tree has size $O(|A|^h K)$ for K scenarios. Further we can prove that a moderate K value is sufficient for near-optimal decision making. So the DESPOT tree size is much smaller, leading to much faster tree search.

Using a DESPOT tree, we can derive the best action or the value at the root node b using either bottom-up dynamic programming or any forward search algorithm.

Example. POMDP planning for autonomous driving, using DESPOT.

Summary.

- The POMDP is a powerful tool. It reasons about uncertainty in both actions and observations. It connects robot perception and action.
- A POMDP can be converted into an equivalent belief MDP. However, the belief MDP cannot be solved using the standard MDP algorithms, such as value iteration. The belief space is enormous.
- POMDP planning is challenging, but recent algorithmic advances, e.g., DESPOT, enables us to use it for real-time decision making in dynamic environments.

Required readings.

- [Siegwart, Nourbakhsh & Scaramuzza] Sect 6.5

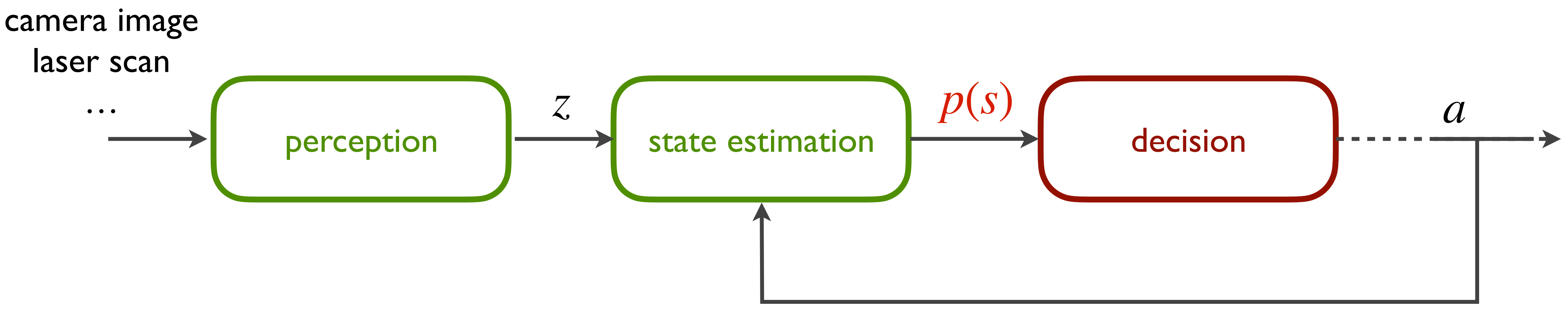
Supplementary readings.

Key concepts.

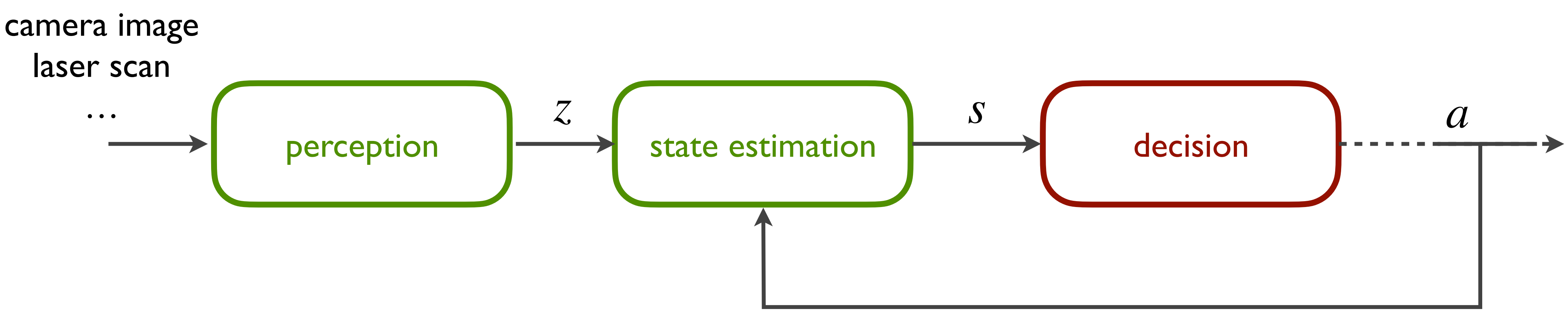
- POMDP
- Model-based modular system
- End-to-end learning system

Tools.

- SARSOP
- DESPOT


camera image
laser scan
...
perception
z
state estimation
$p(s)$
decision
a

Discussion



camera image
laser scan
...
perception
z
state estimation
s
decision
a