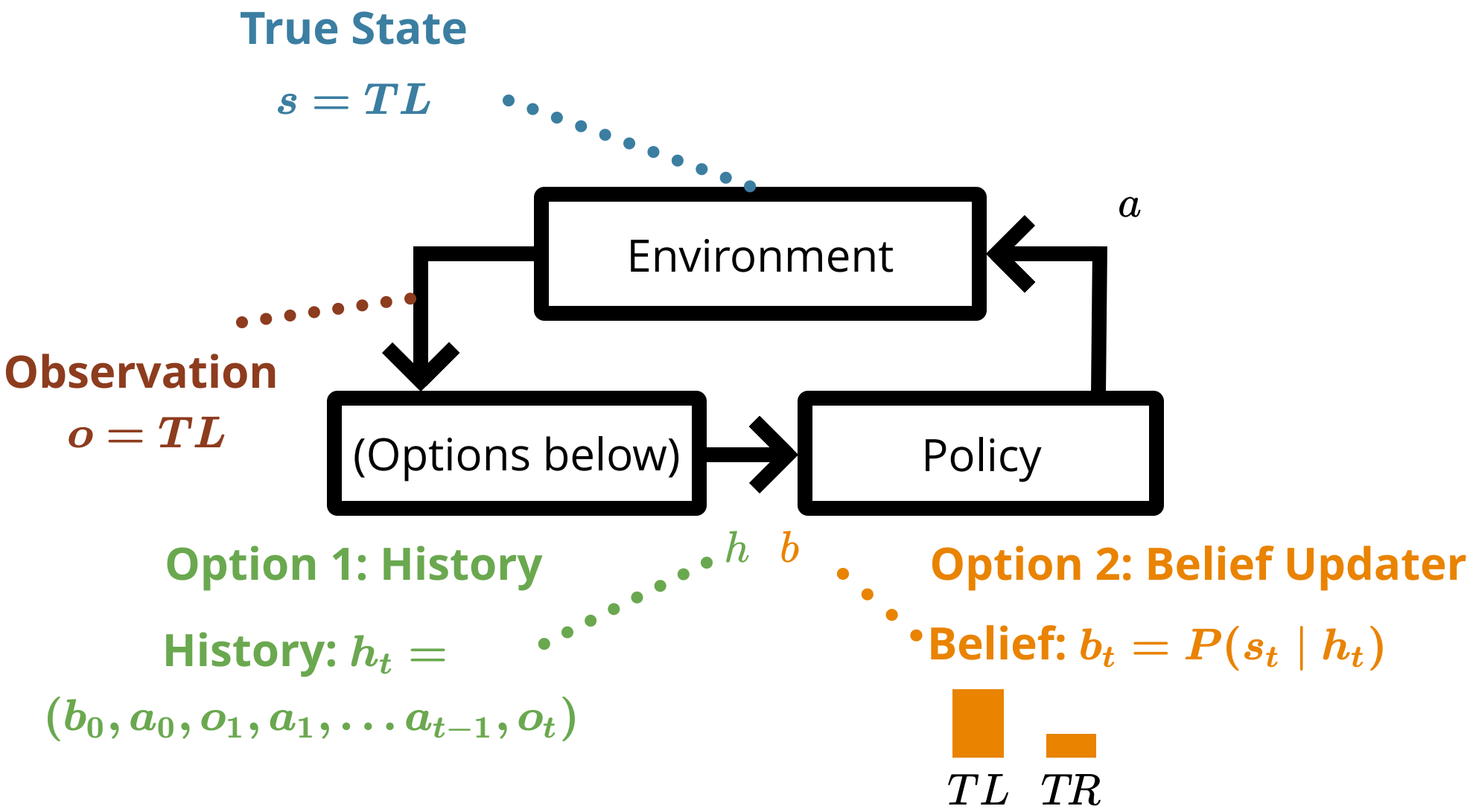
Particle Filters

POMDP Sense-Plan-Act Loop



Review: Bayesian Filter

```
function update(b::Vector{Float64}, 𝒫, a, o)
    𝒮, T, O = 𝒫.𝒮, 𝒫.T, 𝒫.O
    b′ = similar(b)
    for (i′, s′) in enumerate(𝒮)
        po = O(a, s′, o)
        b′[i′] = po * sum(T(s, a, s′) * b[i] for (i, s) in enumerate(𝒮))
    end
    if sum(b′) ≈ 0.0
        fill!(b′, 1)
    end
    return normalize!(b′, 1)
end
```

Review: Bayesian Filter

$$b_t(s) = P(s_t = s \mid h_t)$$

```
function update(b::Vector{Float64}, 𝒫, a, o)
    𝒮, T, O = 𝒫.𝒮, 𝒫.T, 𝒫.O
    b′ = similar(b)
    for (i′, s′) in enumerate(𝒮)
        po = O(a, s′, o)
        b′[i′] = po * sum(T(s, a, s′) * b[i] for (i, s) in enumerate(𝒮))
    end
    if sum(b′) ≈ 0.0
        fill!(b′, 1)
    end
    return normalize!(b′, 1)
end
```

Review: Bayesian Filter

$$b_t(s) = P(s_t = s \mid h_t)$$

$$b' = \tau(b, a, o)$$

```
function update(b::Vector{Float64}, 𝒫, a, o)
    𝒮, T, O = 𝒫.𝒮, 𝒫.T, 𝒫.O
    b′ = similar(b)
    for (i′, s′) in enumerate(𝒮)
        po = O(a, s′, o)
        b′[i′] = po * sum(T(s, a, s′) * b[i] for (i, s) in enumerate(𝒮))
    end
    if sum(b′) ≈ 0.0
        fill!(b′, 1)
    end
    return normalize!(b′, 1)
end
```

Review: Bayesian Filter

$$b_t(s) = P(s_t = s \mid h_t)$$

$$b' = \tau(b, a, o)$$

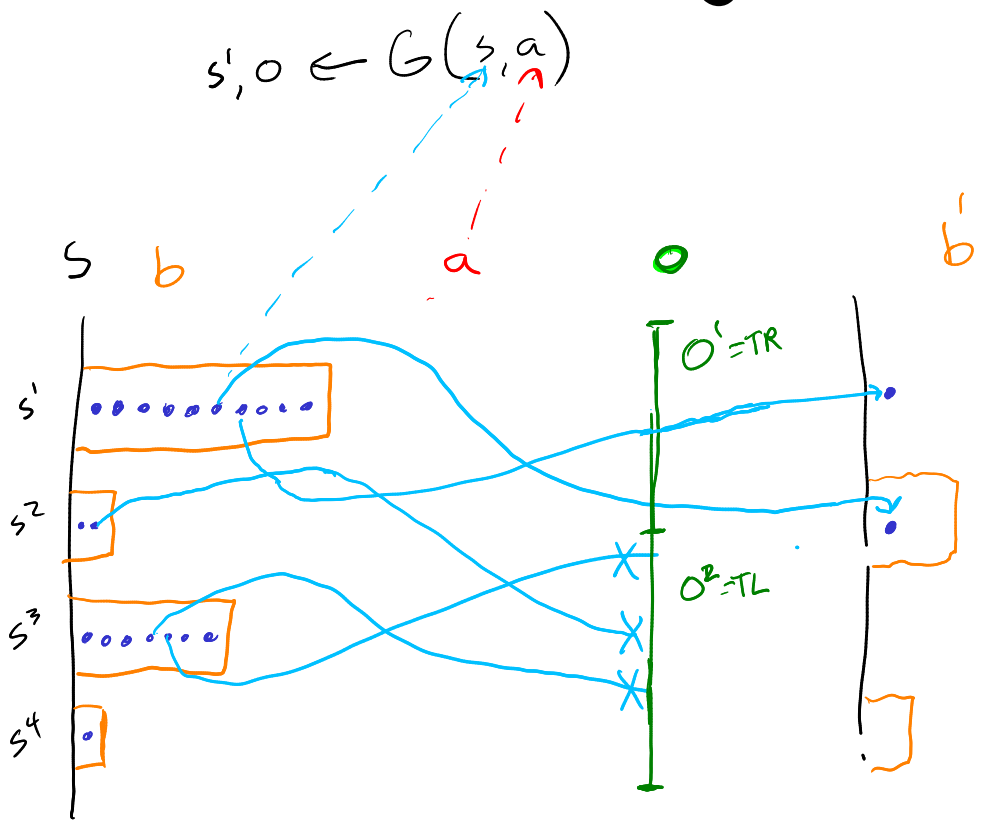
$$b'(s') \propto Z(o \mid a, s') \sum_s T(s' \mid s, a)\, b(s)$$

```
function update(b::Vector{Float64}, 𝒫, a, o)
    𝒮, T, O = 𝒫.𝒮, 𝒫.T, 𝒫.O
    b′ = similar(b)
    for (i′, s′) in enumerate(𝒮)
        po = O(a, s′, o)
        b′[i′] = po * sum(T(s, a, s′) * b[i] for (i, s) in enumerate(𝒮))
    end
    if sum(b′) ≈ 0.0
        fill!(b′, 1)
    end
    return normalize!(b′, 1)
end
```

Rejection Particle Filter

Rejection Particle Filter

state




```
function update(b::RejectionParticleFilter, 𝒫, a, o)
    T, O = 𝒫.T, 𝒫.O
    states = similar(b.states)
    i = 1
    while i ≤ length(states)
        s = rand(b.states)
        s′ = rand(T(s,a))
        if rand(O(a,s′)) == o
            states[i] = s′
            i += 1
        end
    end
    return RejectionParticleFilter(states)
end
```

o'

G(s, a)

update(filter, pomdp, a, o)

choose a=L

receive o=TR

b

50 50

L R

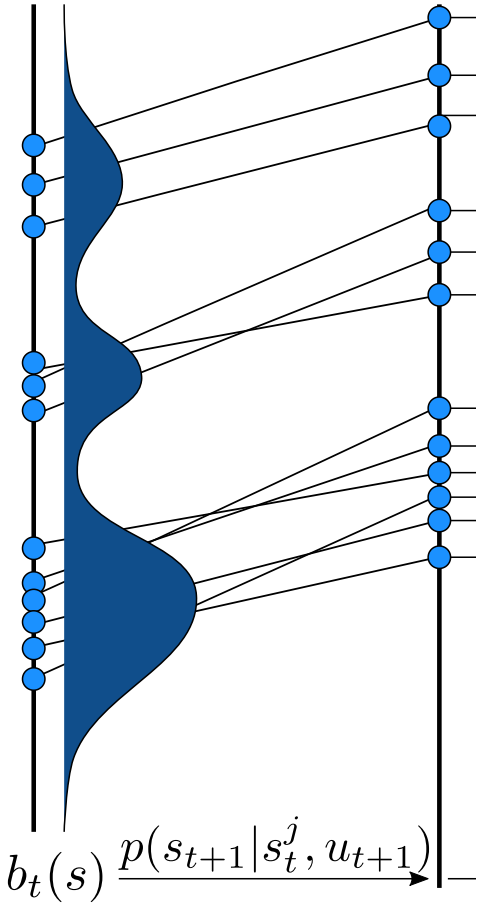
Weighted Particle Filtering

$Z(o|a,s')$

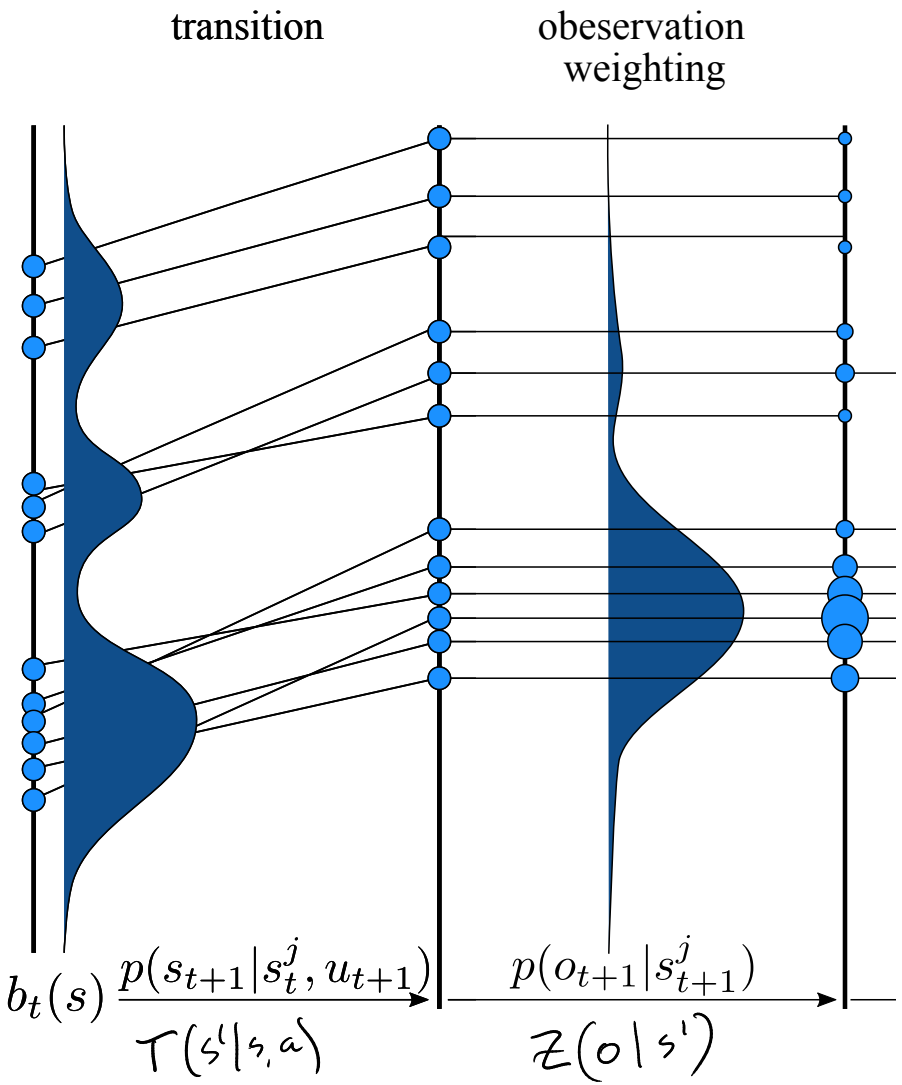
Weighted Particle Filtering

$s' \leftarrow G(s,a)$

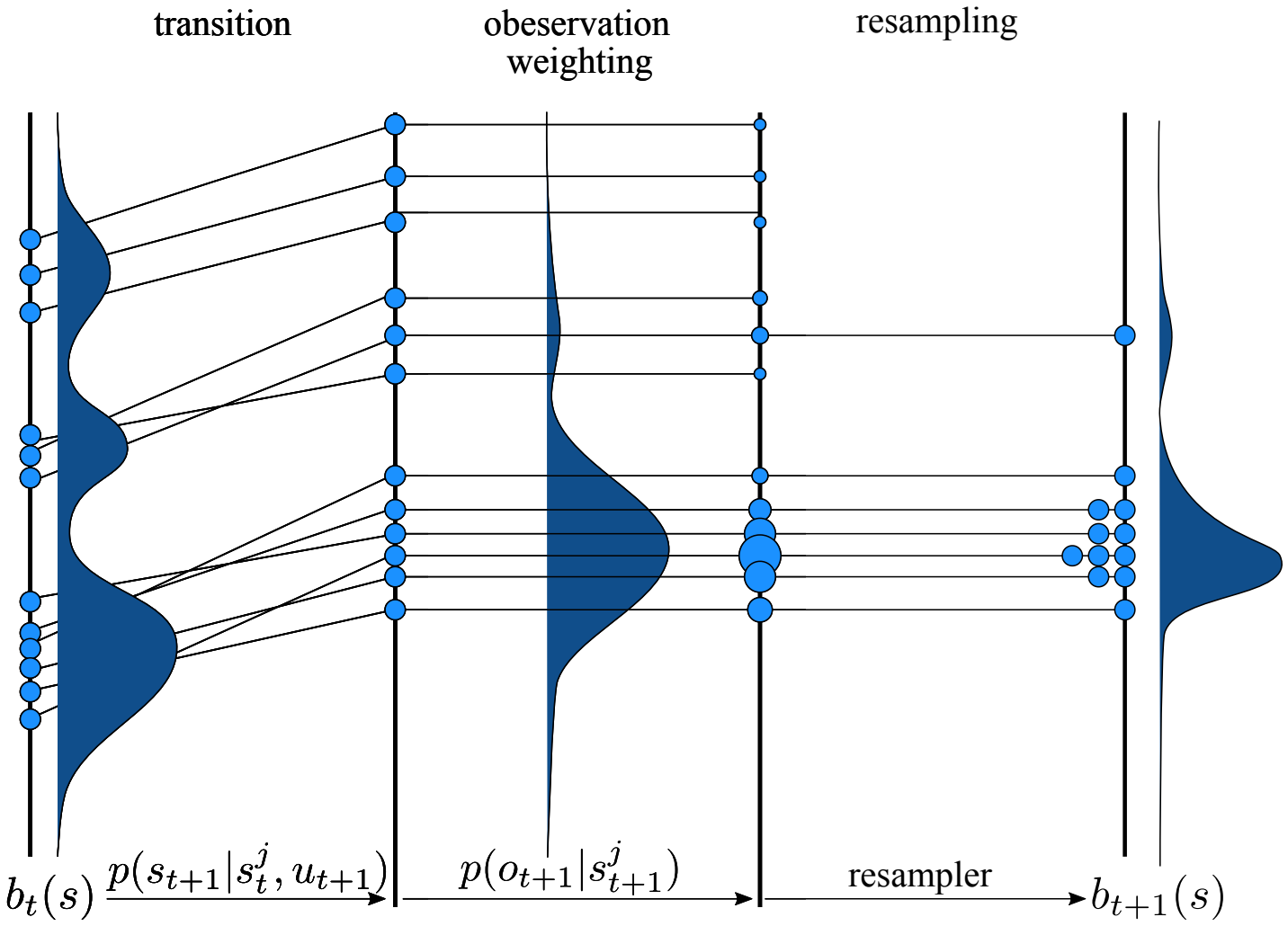
transition



Weighted Particle Filtering



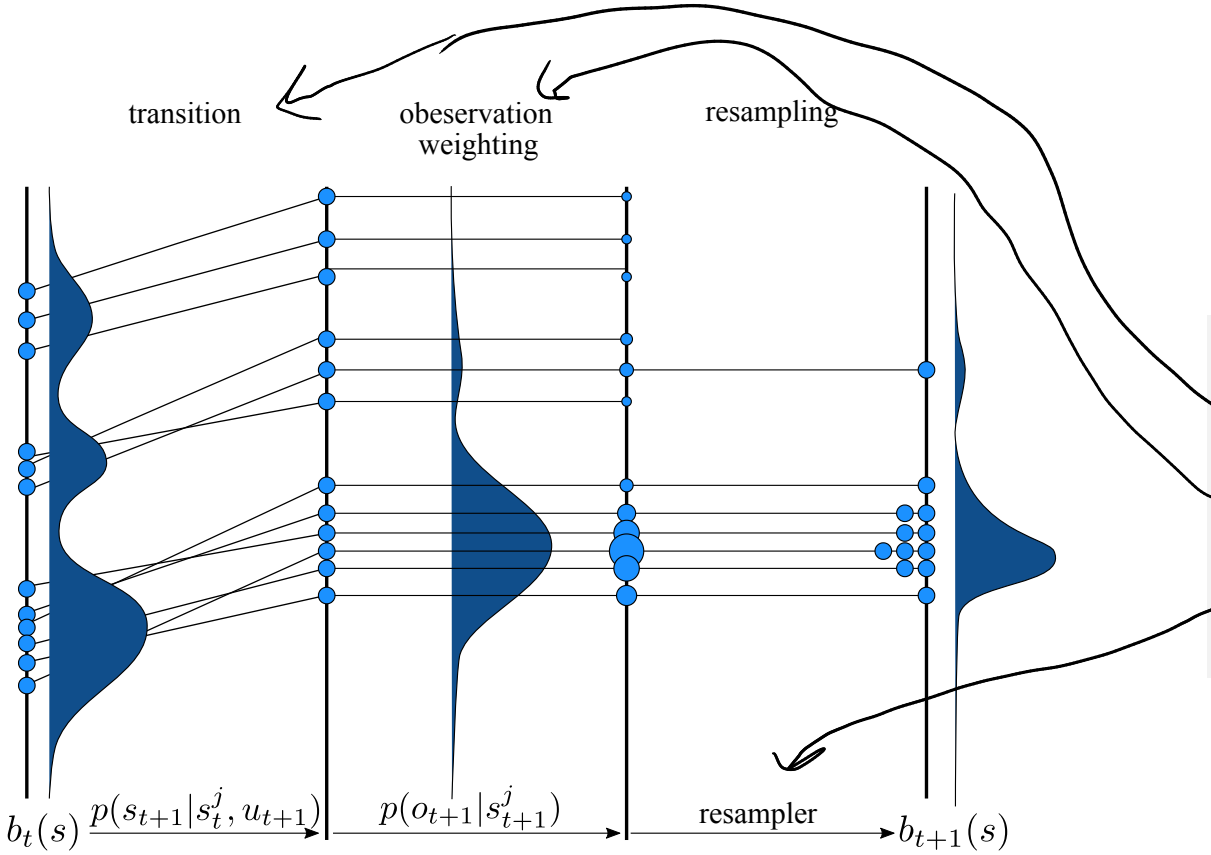
Weighted Particle Filtering



Weighted Particle Filtering

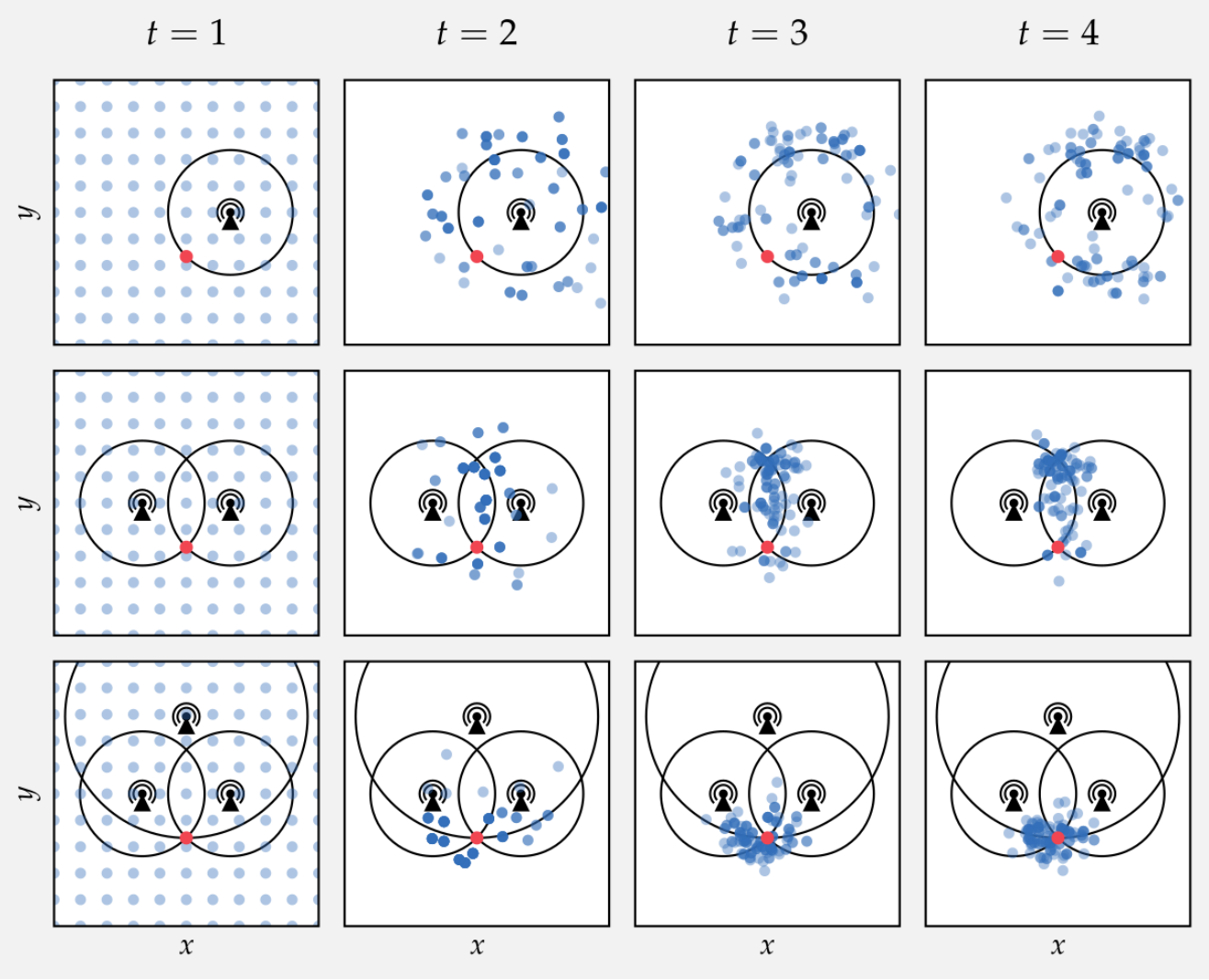
```
function update(b::ParticleFilter, 𝒫, a, o)
    T, O = 𝒫.T, 𝒫.O
    states = [rand(T(s, a)) for s in b.states]
    weights = [O(a, s′, o) for s′ in states]
    D = SetCategorical(states, weights)
    return ParticleFilter(rand(D, length(states)))
end
```

Weighted Particle Filtering



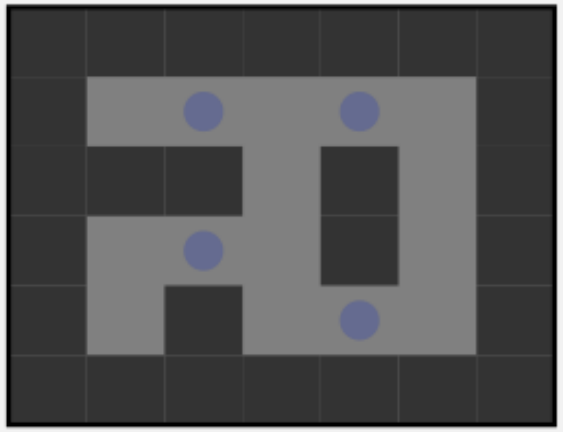

```
function update(b::ParticleFilter, 𝒫, a, o)
    T, O = 𝒫.T, 𝒫.O
    states = [rand(T(s, a)) for s in b.states]
    weights = [O(a, s′, o) for s′ in states]
    D = SetCategorical(states, weights)
    return ParticleFilter(rand(D, length(states)))
end
```

Weighted Particle Filtering

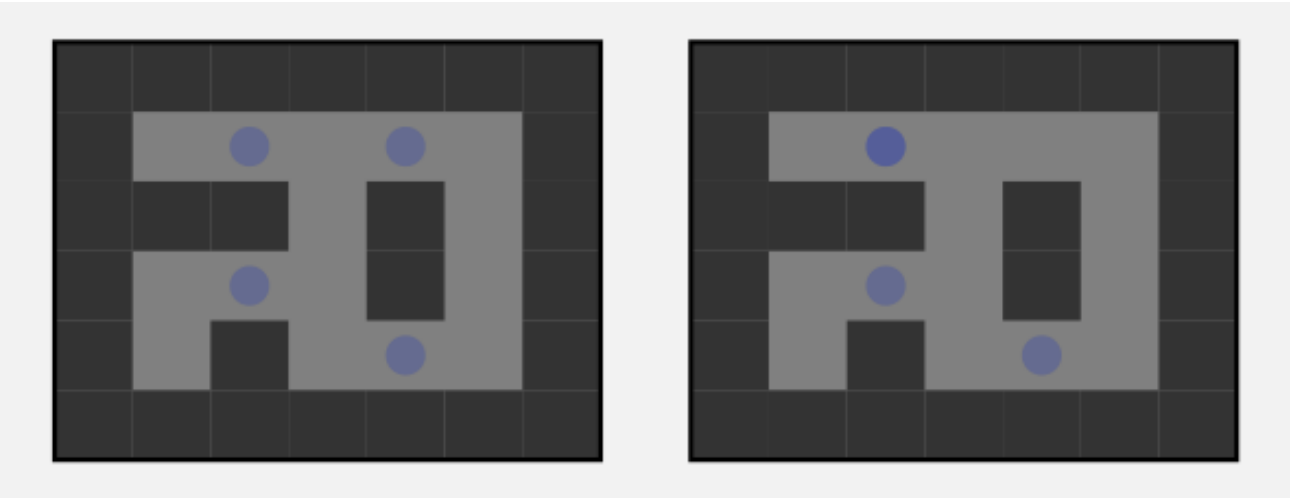


Particle Depletion

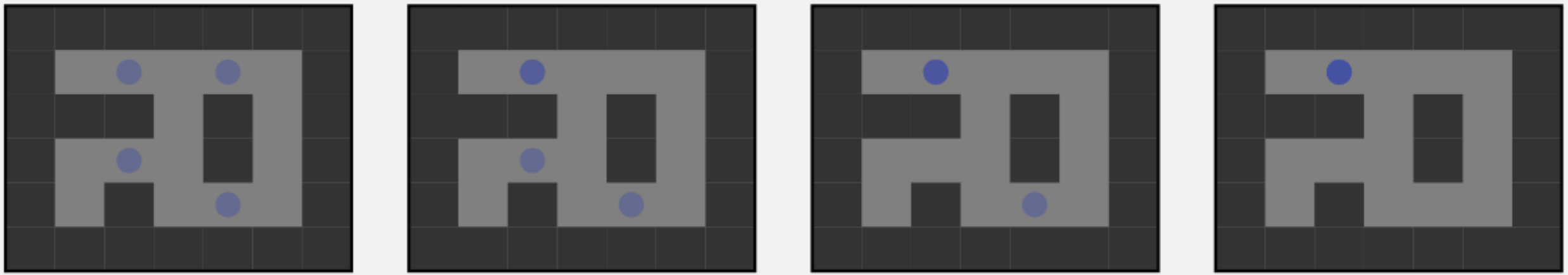
Particle Depletion



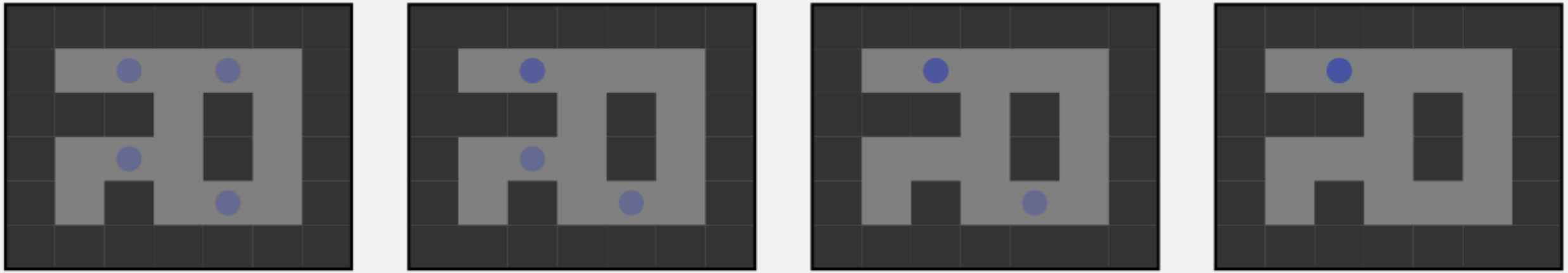
Particle Depletion



Particle Depletion

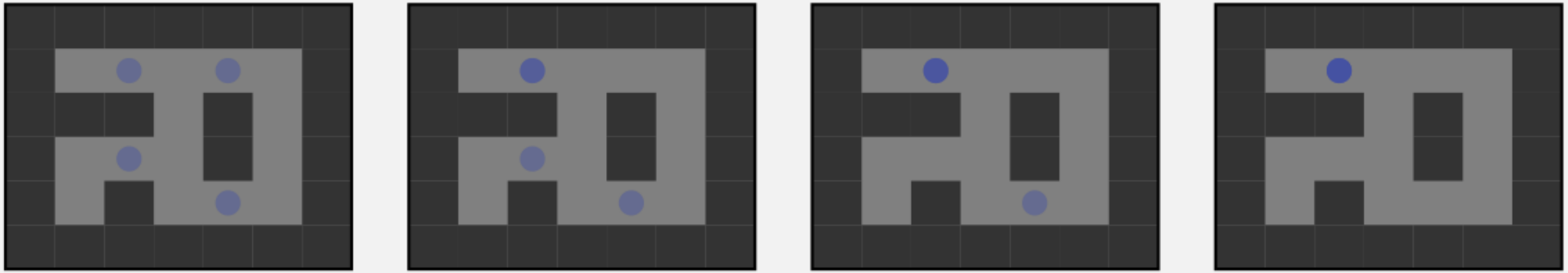


Particle Depletion



Solution: Domain specific particle injection based on:

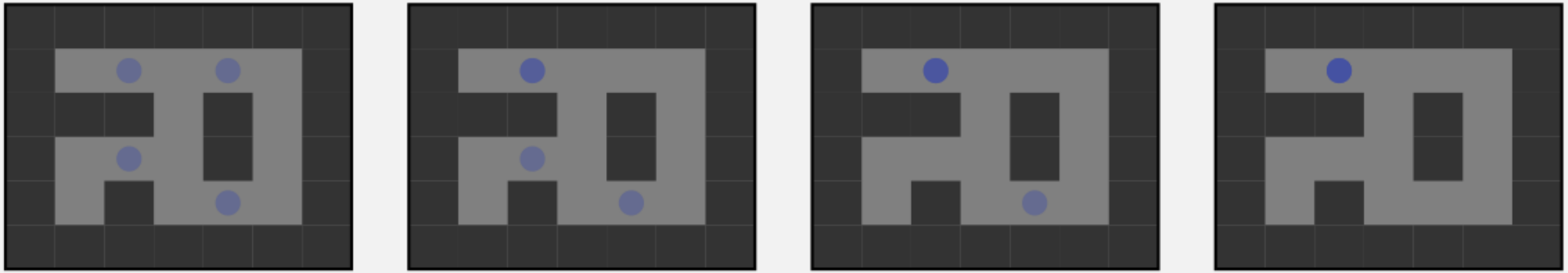
Particle Depletion



Solution: Domain specific particle injection based on:

- Weights

Particle Depletion



Solution: Domain specific particle injection based on:

- Weights
- Particle Diversity

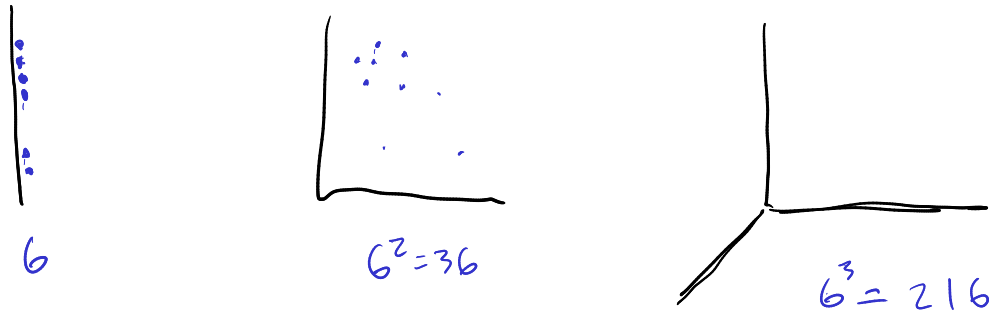
Important Particle Filter Properties

Important Particle Filter Properties

- When only estimating the reward, the number of particles does **NOT** need to scale exponentially with the dimension (i.e. $n \neq k^d$)

Important Particle Filter Properties

- When only estimating the reward, the number of particles does **NOT** need to scale exponentially with the dimension (i.e. $n \neq k^d$)
- Implementation should have $O(n)$ complexity. ← $O(n^2)$ $O(n^3)$



$$SEM = \frac{std}{\sqrt{N}}$$

```
D = Categorical(particles)
for i in 1:n
    push(bp, rand(D))
end
```

$O(n^2)$

Low-variance resample