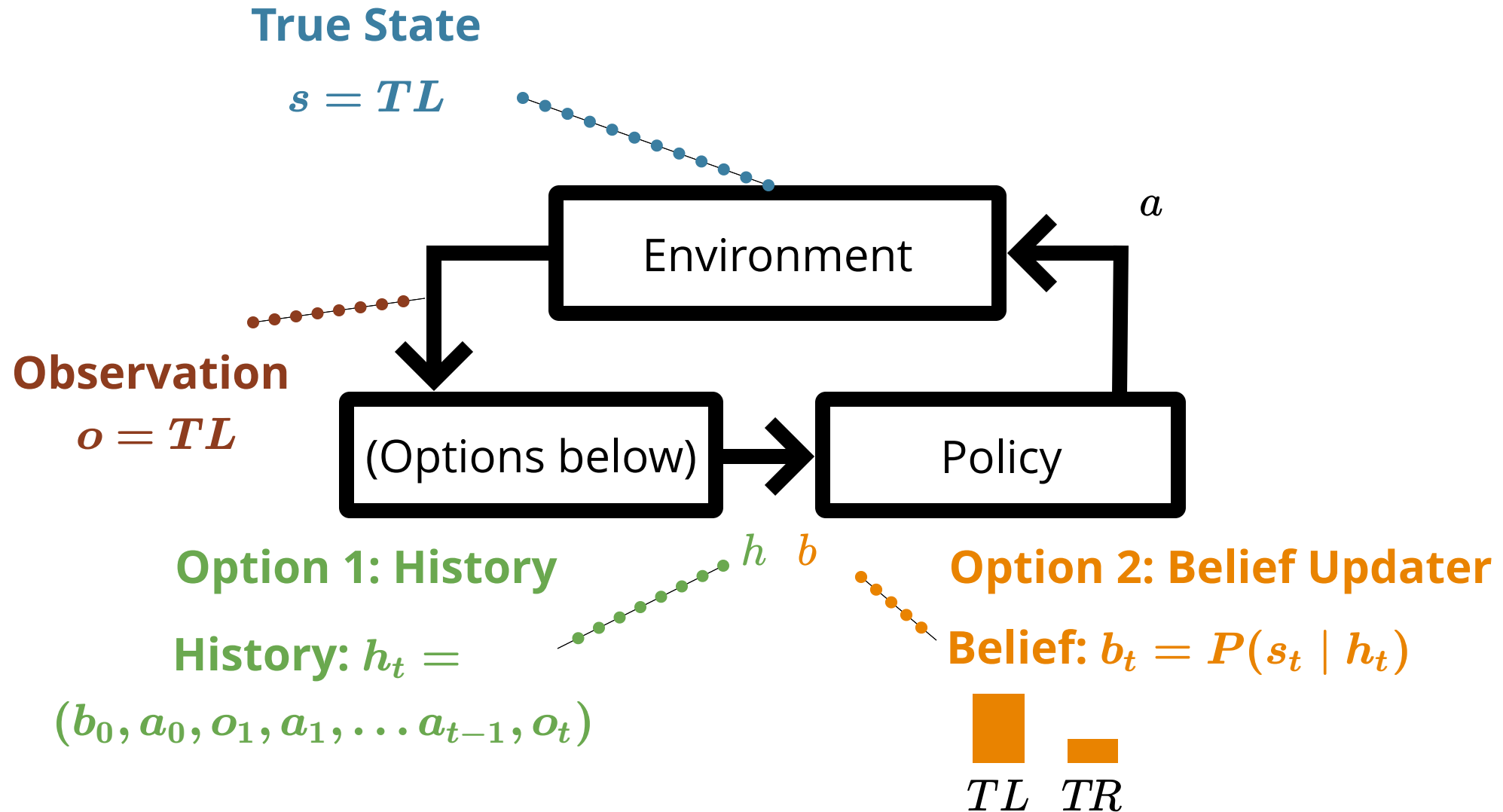


# Particle Filters

# POMDP Sense-Plan-Act Loop



# Review: Bayesian Filter

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

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

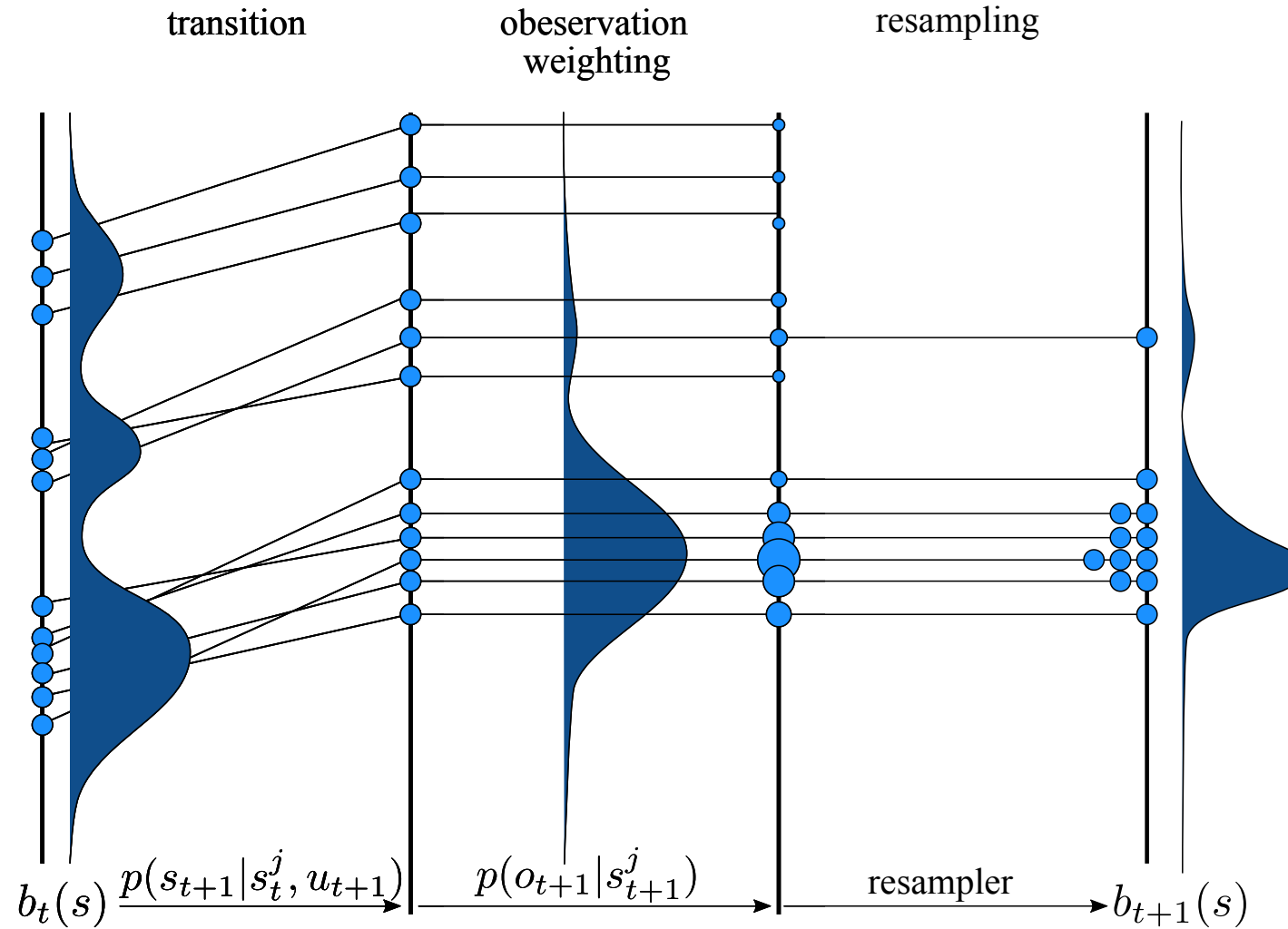
$$b'(s') \propto Z(o \mid a, s') \sum_s T(s' \mid s, a) b(s)$$

```
function update(b::Vector{Float64},  $\mathcal{P}$ , a, o)
     $\mathcal{S}$ , T, 0 =  $\mathcal{P}.\mathcal{S}$ ,  $\mathcal{P}.T$ ,  $\mathcal{P}.0$ 
    b' = similar(b)
    for (i', s') in enumerate( $\mathcal{S}$ )
        po = 0(a, s', o)
        b'[i'] = po * sum(T(s, a, s') * b[i] for (i, s) in enumerate( $\mathcal{S}$ ))
    end
    if sum(b')  $\approx$  0.0
        fill!(b', 1)
    end
    return normalize!(b', 1)
end
```

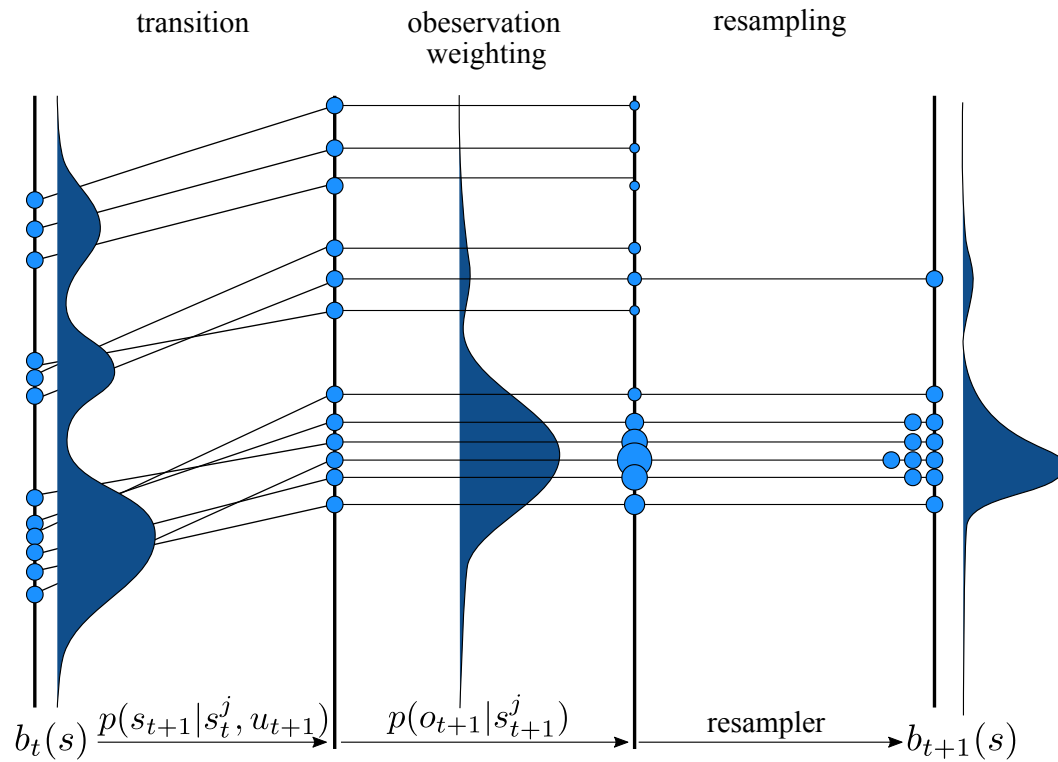
# Rejection Particle Filter

```
function update(b::RejectionParticleFilter,  $\mathcal{P}$ , a, o)
    T, 0 =  $\mathcal{P}$ .T,  $\mathcal{P}$ .0
    states = similar(b.states)
    i = 1
    while i ≤ length(states)
        s = rand(b.states)
        s' = rand(T(s,a))
        if rand(O(a,s')) == 0
            states[i] = s'
            i += 1
        end
    end
    return RejectionParticleFilter(states)
end
```

# Weighted Particle Filtering

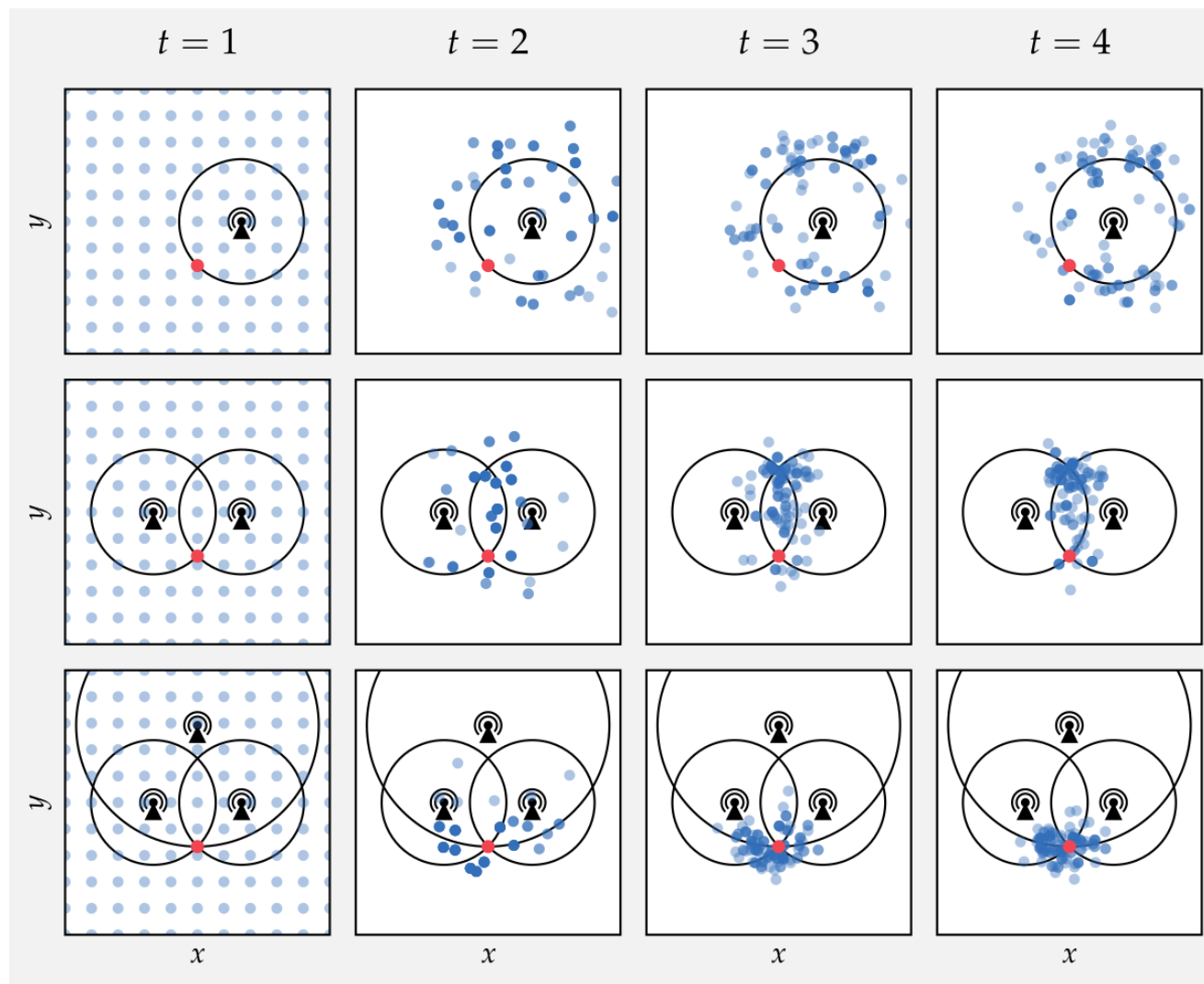


# Weighted Particle Filtering

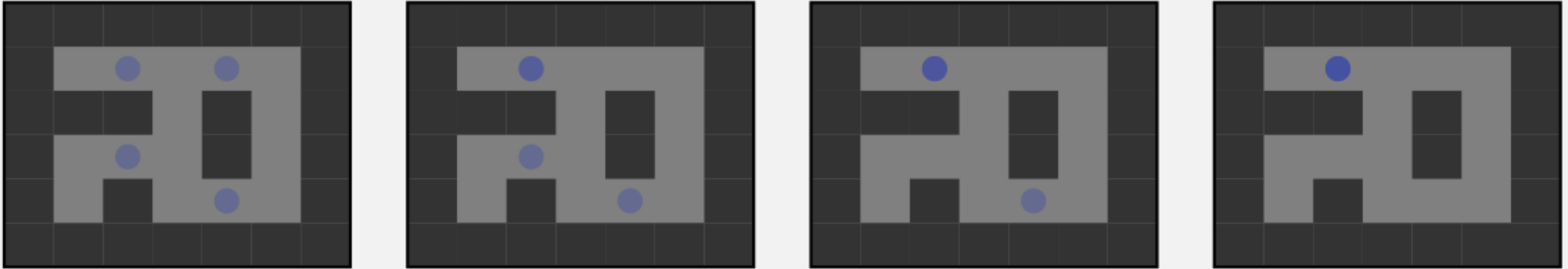


```
function update(b::ParticleFilter,  $\mathcal{P}$ , a, o)
    T, O =  $\mathcal{P}.T$ ,  $\mathcal{P}.O$ 
    states = [rand(T(s, a)) for s in b.states]
    weights = [O(a, s', o) for s' in states]
    D = SetCategorical(states, weights)
    return ParticleFilter(rand(D, length(states)))
end
```

# Weighted Particle Filtering



# Particle Depletion



**Solution:** Domain specific particle injection based on:

- Weights
- Particle Diversity



# Important Particle Filter Properties

- Often 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.