Robotics Notes

Huu Duc Nguyen M.Sc.

07 May 2022

Contents

1 Introduction **1**

Abbreviations **2**

2 Probabilistic Robotics **3**

2.1 State Estimation 3

2.1.1 Bayes Filters 3

2.1.2 The Kalman Filter (KF) 3

2.1.3 Extended Kalman Filter (EKF) 4

2.1.4 Information Filter (IF) 5

2.1.5 Extended Information Filter (IF) 6

2.2 Measurements 6

2.2.1 Map Representation 6

2.2.2 Measurement Noise 7

2.3 Robot Motion 7

2.3.1 Motion Model 8

2.3.2 Velocity Motion Model 8

2.3.3 Odometry Motion Model 9

2.3.4 Map-based Motion Model 10

3 Markov Decision Process **12**

3.1 Definitions 12

3.1.1 Markov Chain 12

3.1.2 Markov Decision Process (MDP) 12

3.1.3 Partially Observable Markov Decision Process (POMDP) 13

3.2 Bellman equations 14

3.3 Partially Observable Markov Decision Process 14

4 Research Proposal **15**

4.1 Introduction and Background of Interest 15

4.2 Literature Review 15

4.2.1 Robot Grasping 15

4.3 Approaches and Choice of Methods 17

4.3.1 Key Points for Improvement 17

4.3.2 Approaches and Research Contributions 17

4.4 Proposed Research Plan 17

Bibliography **I**

1 Introduction

This is my personal learning notes for robotics.

[**TODO:**]

Abbreviations

prob.	probability
a.k.a.	also known as
func.	function
vs.	versus
KF	Kalman Filter
EKF	Extended Kalman Filter
IF	Information Filter
EIF	Extended Information Filter
MHEKF	Multi-Hypothesis Extended Kalman Filter
MDP	Markov Decision Process
POMDP	Partially Observable Markov Decision Process

2 Probabilistic Robotics

Reference from the great book: [TBF06].

2.1 State Estimation

2.1.1 Bayes Filters

$$
\begin{aligned}
bel(x_t) &= p(x_t|z_{1:t}, u_{1:t}) && \text{belief over a state} \\
\overline{bel}(x_t) &= p(x_t|z_{1:t-1}, u_{1:t}) && \text{a posterior (before adapt to } z_t) \\
\Rightarrow \text{Calculating } bel(x_t) &\text{ from } \overline{bel}(x_t) && \text{correction/measurement update}
\end{aligned}
$$

Bayes Filter Algorithm:

2 for all x_t do:

3 $\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx$ prediction step

4 $bel(x_t) = \eta\, p(z_t|x_t)\, \overline{bel}(x_t)$ update step

$\Rightarrow$ Can only be implemented for very simple estimation problems, finite state space

Important assumption: Markov property (each state s a complete summary of the past)

Problem: can not be implemented on digital computers

The next subsections describe two Gaussian filters (Kalman Filter (KF) and Information Filter (IF)) and two extensions of them (Extended Kalman Filter (EKF) and Extended Information Filter (EIF)). The Gaussian filters have major advantage in computational cost, with the disadvantage of having assumption on uni-model distribution.

2.1.2 The Kalman Filter (KF)

- ***Learning Resources:*** www.bzarg.com
- For continuous space, not discrete or hybrid

- **Assumption:** posterior are Gaussians and Markov property

$p(x_t \vert u_t, x_{t-1})$	must be linear func. (linear system dynamics)
$p(z_t, x_t)$	also linear
$bel(x_0)$	initial belief must be Gaussian

$$
\begin{aligned}
x_t &= A_t x_{t-1} + B_t u_t + \varepsilon_t \\
p(x_t|u_t, x_{t-1}) &= \det\left(2\pi\Sigma_t\right)^{-\frac{1}{2}} \exp\left[-\frac{1}{2}\left(x_t - A_t x_{t-1} - B_t u_t\right)^T R_t^{-1}\left(x_t - A_t x_{t-1} - B_t u_t\right)\right] \\
z_t &= C_t x_t + \delta_t \quad (= y) \\
p(z_t|x_t) &= \det\left(2\pi Q_t\right)^{-\frac{1}{2}} \exp\left[-\frac{1}{2}\left(z_t - C_t x_t\right)^T Q_t^{-1}\left(z_t - C_t x_t\right)\right] \\
bel(x_0) = p(x_0) &= \det\left(2\pi\Sigma_0\right)^{-\frac{1}{2}} \exp\left[-\frac{1}{2}\left(x_t - \mu_0\right)^T \Sigma_0^{-1}\left(x_t - \mu_0\right)\right]
\end{aligned}
$$

Kalman Filter Algorithm: $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$

$$
\left.\begin{aligned}
&\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t \\
&\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t
\end{aligned}\right\} \text{incorporate } u_t \text{ - prediction step - } \mathcal{O}(n^2)
$$

$$
\left.\begin{aligned}
&K_t = \overline{\Sigma}_t C_t^T \left(C_t \overline{\Sigma}_t C_t^T + Q_t\right)^{-1} \qquad \text{(Kalman gain)} \\
&\mu_t = \overline{\mu}_t + K_t(z_t - C_t \overline{\mu}_t) \\
&\Sigma_t = (I - K_t C_t)\overline{\Sigma}_t
\end{aligned}\right\} \text{incorporate } z_t \text{ - correction step - } \mathcal{O}(n^{2,8})
$$

return μ_t, Σ_t $\Rightarrow$ belief at time t

$\Rightarrow$ quite computationally expensive, **everything are Gaussians**

2.1.3 Extended Kalman Filter (EKF)

Overcome the assumption on linearity by only approximate by Gaussians

$$
\begin{aligned}
x_t &= g(u_t, x_{t-1}) + \varepsilon_t \\
z_t &= h(x_t) + \delta_t
\end{aligned}
$$

$$\begin{aligned}
g(u_t, x_{t-1}) &\approx g(u_t, \mu_{t-1}) + g'(u_t, \mu_{t-1})(x_t - \mu_{t-1}) \\
&= g(u_t, \mu_{t-1}) + G_t(x_t - \mu_{t-1}) \\
&\quad g' \text{ is the Jacobian of state } (n \times n \text{ matrix}) \\
p(x_t|u_t, x_{t-1}) &\approx \det(2\pi R_t)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}[x_t - g(u_t, x_{t-1})]^T R_t^{-1}[x_t - g(u_t, x_{t-1})]\right\} \\
h(t) &\approx h(\overline{\mu}_t) + h'(\mu_t)(x_t - \mu_{t-1}) \\
&= h(\overline{\mu}_t) + H_t(x_t - \mu_{t-1}) \\
p(z_t|x_t) &= \det(2\pi Q_t)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}[z_t - h(x_t)]^T Q_t^{-1}[z_t - h(x_t)]\right\}
\end{aligned}$$

Extended Kalman Filter Algorithm: $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$

2 $\quad \overline{\mu}_t = g(u_t, \mu_{t-1})$

3 $\quad \overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$

4 $\quad K_t = \overline{\Sigma}_t H_t^T \left(H_t \overline{\Sigma}_t H_t^T + Q_t\right)^{-1}$

5 $\quad \mu_t = \overline{\mu}_t + K_t[z_t - h(\overline{\mu}_t)]$

6 $\quad \Sigma_t = (I - K_t H_t)\overline{\Sigma}_t$

7 $\quad$ return μ_t, Σ_t

- Can extend EKF $\Rightarrow$ Multi-Hypothesis Extended Kalman Filter (MHEKF)
- EKF's performance depends on degree of nonlinearities and uncertainty
- Unscented KF and moments matching KF are better

2.1.4 Information Filter (IF)

- Moment representation: $\mu \quad \& \quad \Sigma$
- Canonical representation: $\xi \quad \& \quad \Omega$

  Information precision matrix: $\Omega = \Sigma^{-1}; \qquad \Sigma = \Omega^{-1}$

  Information vector: $\xi = \Sigma^{-1}\mu; \qquad \mu = \Omega^{-1}\xi$

$$\begin{aligned}
p(x) &= \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\} \\
&= \eta \exp\left\{-\frac{1}{2}x^T \Omega x + x^T \xi\right\}
\end{aligned}$$

Information Filter Algorithm: $(\xi_{t-1}, \Omega_{t-1}, u_t, z_t)$

$$\left.\begin{aligned} \overline{\Omega}_t &= (A_t \Omega_{t-1}^{-1} A_t^T + R_t)^{-1} \\ \overline{\xi}_t &= \overline{\Omega}_t (A_t \Omega_{t-1}^{-1} \xi_{t-1} + B_t u_t) \end{aligned}\right\} \mathcal{O}(n^{2,8})$$

$$\left.\begin{aligned} \Omega_t &= C_t^T Q_t^{-1} C_t + \overline{\Omega}_t \\ \xi_t &= C_t^T Q_t^{-1} z_t + \overline{\xi}_t \end{aligned}\right\} \mathcal{O}(n^2)$$

return ξ_t, Ω_t

2.1.5 Extended Information Filter (IF)

$$\begin{aligned} x_t &= g(u_t, x_{t-1}) + \varepsilon_t \\ z_t &= h(x_t) + \delta_t \\ G_t &= g'(u_t, \mu_{t-1}) \\ H_t &= h'(\mu_t) \end{aligned}$$

Extended Information Filter Algorithm: $(\xi_{t-1}, \Omega_{t-1}, u_t, z_t)$

$$\begin{aligned} \mu_{t-1} &= \Omega_{t-1}^{-1} \xi_{t-1} \\ \overline{\Omega}_t &= (G_t \Omega_{t-1}^{-1} G_t^T + R_t)^{-1} \\ \overline{\xi}_t &= \overline{\Omega}_t \, g(u_t, \mu_{t-1}) \\ \overline{\mu}_t &= g(u_t, \mu_{t-1}) \\ \Omega_t &= \overline{\Omega}_t + H_t^T Q_t^{-1} H_t \\ \xi_t &= \overline{\xi}_t + H_t^T Q_t^{-1} \left[z_t - h(\overline{\mu}_t) - H_t \overline{\mu}_t\right] \end{aligned}$$

- ***Global uncertainty:*** set $\Omega = 0$ is better than set $|\Sigma| = \infty$
- IF tends to be numerically more stable than KF
- IF is better for multi-robot problems
- For high dimensional state, EKF is computational better than EIF

2.2 Measurements

2.2.1 Map Representation

Maps: $m = \{m_1, m_2, \ldots, m_N\}$
There are 2 ways to represent a map:

[TODO: Add images]

feature-based	location-based
m_n: properties of a feature and location of feature	a specific location
only the shape of the environment **at the specific locations**	volumetric: **label for any location** in the world
easy to adjust positions of objects $\Rightarrow$ popular in the robotic mapping field	occupancy grid map

2.2.2 Measurement Noise

The 4 types of measurement noise:

- Correct range with local measurement noise
  With z_t^{k*} as the correct distance

$$p_{hit}(z_t^k|x_t, m) = \begin{cases} \eta\, \mathcal{N}(z_t^k|z_t^{k*}, \sigma_{hit}^2) & \text{if } 0 \leq z_t^k \leq z_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

- Unexpected object

$$p_{short}(z_t^k|x_t, m) = \begin{cases} \eta \lambda_{short} e^{-\lambda_{short} z_t^k} & \text{if } 0 \leq z_t^k \leq z_t^{k*} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

- Failures

$$p_{\max}(z_t^k|x_t, m) = I(z = z_{\max}) \tag{2.3}$$

- Random measurements

$$p_{rand}(z_t^k|x_t, m) = \begin{cases} \frac{1}{z_{\max}} & \text{if } 0 \leq z_t^k < z_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

[**TODO: Add image, plot**]

$$p(z_t^k|x_t, m) = \begin{bmatrix} z_{hit} \\ z_{short} \\ z_{max} \\ z_{rand} \end{bmatrix}^T \cdot \begin{bmatrix} p_{hit}(z_t^k|x_t, m) \\ p_{short}(z_t^k|x_t, m) \\ p_{max}(z_t^k|x_t, m) \\ p_{rand}(z_t^k|x_t, m) \end{bmatrix} \tag{2.5}$$

2.3 Robot Motion

Pose: $[x, y, \theta]^T$ at location $[x, y]^T$ and orientation θ

2.3.1 Motion Model

Motion Model, also known as (a.k.a.) Probabilistic Kinematic Model: $p(x_t, u_t, x_{t-1})$

Velocity commands	Odometry (distance traveled, angle turned, etc.)
Use for Probabilistic motion planning	**more accurate** **but post-the-fact** **(not for motion planning)** Use for estimation

Each has closed form calculation and sampling algorithm.

2.3.2 Velocity Motion Model

Assuming we can control a robot through velocities:

$$u_t = \begin{bmatrix} v_t \\ \omega_t \end{bmatrix}; \qquad x_{t-1} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}; \qquad x_t = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix}$$

Motion Model Velocity Algorithm: (x_t, u_t, x_{t-1})

$$\left.\begin{aligned}
&\mu = \frac{1}{2}\frac{(x-x')\cos\theta + (y-y')\sin\theta}{(y-y')\cos\theta - (x-x')\sin\theta} \\
&x^* = \tfrac{x+x'}{2} + \mu(y-y') \\
&y^* = \tfrac{y+y'}{2} + \mu(x'-x) \\
&r^* = \sqrt{(x-x^*)^2 + (y-y^*)^2}
\end{aligned}\right\}\text{Invert the motion model}$$

$$\Delta\theta = \operatorname{atan2}(y'-y^*, x'-x^*) - \operatorname{atan2}(y-y^*, x-x^*)$$

$$\left.\begin{aligned}
&\hat{v} = \tfrac{\Delta\theta}{\Delta t} r^* \\
&\hat{\omega} = \tfrac{\Delta\theta}{\Delta t} \\
&\hat{\gamma} = \tfrac{\theta'-\theta}{\Delta t} - \hat{\omega}
\end{aligned}\right\}\text{compared actual velocities with the desired}$$

return $p(v-\hat{v}, \alpha_1|v| + \alpha_2|\omega|) \cdot p(\omega-\hat{\omega}, \alpha_3|v| + \alpha_4|\omega|) \cdot p(\hat{\gamma}, \alpha_5|v| + \alpha_6|\omega|)$

Sample Motion Model Velocity Algorithm: (u_t, x_{t-1})

$$\hat{v} = v + sample(\alpha_1|v| + \alpha_2|\omega|)$$
$$\hat{\omega} = \omega + sample(\alpha_3|v| + \alpha_4|\omega|)$$
$$\hat{\gamma} = sample(\alpha_5|v| + \alpha_6|\omega|)$$
$$x' = x - \frac{\hat{v}}{\hat{\omega}}\sin\theta + \frac{\hat{v}}{\hat{\omega}}\sin(\theta + \hat{\omega}\Delta t)$$
$$y' = y + \frac{\hat{v}}{\hat{\omega}}\cos\theta - \frac{\hat{v}}{\hat{\omega}}\cos(\theta + \hat{\omega}\Delta t)$$
$$\theta' = \theta + \hat{\omega}\Delta t + \hat{\gamma}\Delta t$$
$$\text{return } x_t = [x', y', \theta']^T$$

- Probability normal distribution(a, b): return $\frac{1}{\sqrt{2\pi b}} e^{-\frac{a^2}{2b}}$
- Probability triangular distribution(a, b): return $\begin{cases} 0 & \text{if } |a| > \sqrt{6b} \\ \frac{\sqrt{6b} - |a|}{6b} & \text{else} \end{cases}$
- Sample normal distribution(b): return $\frac{b}{6} \sum_{i=1}^{12} rand(-1, 1)$
- Sample triangle distribution(b): return $b.rand(-1, 1).rand(-1, 1)$

2.3.3 Odometry Motion Model

Only available after the robot has moved
$\Rightarrow$ only use for filter algorithm
not for accurate motion planning and control

Motion Model Odometry Algorithm: (x_t, u_t, x_{t-1})

$$\delta_{rot1} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$$
$$\delta_{trans} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$$
$$\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$$
$$\left.\begin{aligned} &\hat{\delta}_{rot1} = \text{atan2}(y' - y, x' - x) - \theta \\ &\hat{\delta}_{trans} = \sqrt{(x - x')^2 + (y - y')^2} \\ &\hat{\delta}_{rot2} = \theta' - \theta - \hat{\delta}_{rot1} \end{aligned}\right\} \Rightarrow \text{inverse motion model}$$
$$p_1 = prob(\delta_{rot1} - \hat{\delta}_{rot1}, \alpha_1\hat{\delta}_{rot1} + \alpha_2\hat{\delta}_{trans})$$
$$p_2 = prob(\delta_{trans} - \hat{\delta}_{trans}, \alpha_3\hat{\delta}_{trans} + \alpha_4(\hat{\delta}_{rot1} + \hat{\delta}_{rot2}))$$
$$p_3 = prob(\delta_{rot2} - \hat{\delta}_{rot2}, \alpha_1\hat{\delta}_{rot2} + \alpha_2\hat{\delta}_{trans})$$
$$\text{return} \quad p_1.p_2.p_3 \qquad (= p(x_t|u_t, x_{t-1}))$$

NOTE:

- Bar $\Leftrightarrow$ measurements

$$\bar{x}_{t-1} = [\bar{x} \quad \bar{y} \quad \bar{\theta}]^T$$
$$\bar{x}_t = [\bar{x}' \quad \bar{y}' \quad \bar{\theta}']^T$$

- Hat $\Leftrightarrow$ estimations
- No bar and hat $\Leftrightarrow$ hypothesized final pose x, y

Sample Motion Model Odometry Algorithm: (u_t, x_{t-1})

$$\delta_{rot1} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$$
$$\delta_{trans} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$$
$$\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$$
$$\hat{\delta}_{rot1} = \delta_{rot1} - sample(\alpha_1 \delta_{rot1} + \alpha_2 \delta_{trans})$$
$$\hat{\delta}_{trans} = \delta_{trans} - sample(\alpha_3 \delta_{trans} + \alpha_4 (\delta_{rot1} + \delta_{rot2}))$$
$$\hat{\delta}_{rot2} = \delta_{rot2} - sample(\alpha_1 \delta_{rot2} + \alpha_2 \delta_{trans})$$
$$x' = x + \hat{\delta}_{trans} \cos(\theta + \hat{\delta}_{rot1})$$
$$y' = y + \hat{\delta}_{trans} \sin(\theta + \hat{\delta}_{rot1})$$
$$\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$$
$$\text{return} \quad x_t = [x' \quad y' \quad \theta']^T$$

2.3.4 Map-based Motion Model

Map-based Motion Model: $p(x_t | u_t, x_{t-1}, m)$

- Occupancy maps: $p(x_t | m) = 0 \Leftrightarrow$ the robot collides
- If the distance from $x_{t-1} \rightarrow x_t$ is small enough ($<$ half the robot's diameter), we can estimate the probability (prob.) $p(x_t | u_t, x_{t-1}, m) \approx \eta p(x_t | u_t, x_{t-1}) p(x_t | m)$, which discards the info relating the robot's path to x_t

Motion Model with Map Algorithm: (x_t, u_t, x_{t-1}, m)
return $p(x_t | u_t, x_{t-1}).p(x_t | m)$

Sample Motion Model with Map Algorithm: (u_t, x_{t-1}, m)

$$
\begin{aligned}
&do: \\
&\qquad x_t = sample\_motion\_model(u_t, x_{t-1}) \\
&\qquad \pi = p(x_t|m) \\
&until \quad \pi > 0 \\
&return < x_t, \pi >
\end{aligned}
$$

3 Markov Decision Process

3.1 Definitions

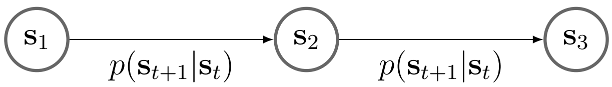
3.1.1 Markov Chain

A Markov Chain $\mathcal{M}$ is a tuple consisting of:

$\mathcal{M} = \{\mathcal{S}, \mathcal{T}\}$

$\mathcal{S}$ – state space	$s \in \mathcal{S}$ (discrete or continuous)
$\mathcal{T}$ – transition operator	$p(s_{t+1}\vert s_t)$ or $\vec{\mu}_{t+1} = \mathcal{T}\vec{\mu}_t$ ($\mathcal{T}$ is a matrix)

Markov chain is a process without memories. In other words, the next state s_{t+1} depends only on the current state s_t, not the previous state s_{t-1}.

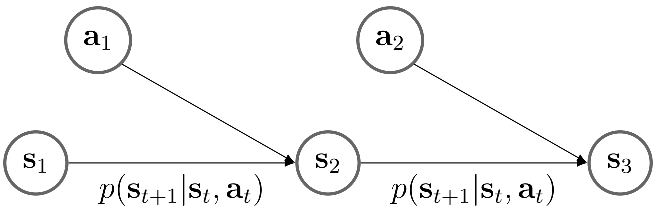


3.1.2 Markov Decision Process (MDP)

A Markov Decision Process (MDP) $\mathcal{M}$ is a tuple consisting of:

$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma\}$

$\mathcal{S}$ – state space	$s \in \mathcal{S}$ (discrete or continuous)
$\mathcal{A}$ – action space	$a \in \mathcal{A}$ (discrete or continuous)
$\mathcal{T}$ – transition operator	$p(s_{t+1}\vert s_t)$ (now a tensor)
r – reward function	$r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, $r(s_t, a_t)$
γ – discount factor	$\gamma \in [0, 1]$ (optional)



There are definitions relating to MDP:

- Policy: choice of action (at each state): $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$
- Utility: sum of (discounted) rewards

A MDP can also be considered as a Markov chain on the augmented state $(\mathbf{s}, \mathbf{a})$, a.k.a. Q-state. Knowing the state transition $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ and policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$, the transition of these Q-states can be derived as follows:

$$p((\mathbf{s}_{t+1}, \mathbf{a}_{t+1})|(\mathbf{s}_t, \mathbf{a}_t)) = p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\pi_\theta(\mathbf{a}_{t+1}|\mathbf{s}_{t+1}) \tag{3.1}$$

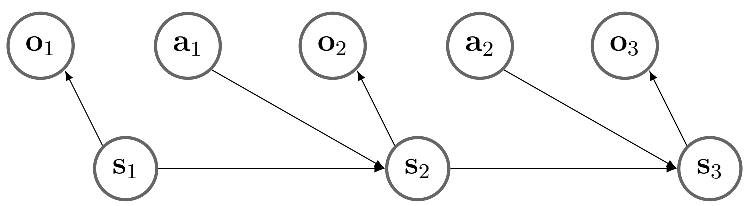
MDP state projects an expectimax-like search tree [**TODO: add image**]

3.1.3 Partially Observable Markov Decision Process (POMDP)

A Partially Observable Markov Decision Process (POMDP) $\mathcal{M}$ is a tuple consisting of:

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{E}, r, \gamma\}$$

$\mathcal{S}$ – state space	$s \in \mathcal{S}$ (discrete or continuous)
$\mathcal{A}$ – action space	$a \in \mathcal{A}$ (discrete or continuous)
$\mathcal{O}$ – observation space	$o \in \mathcal{O}$ (discrete or continuous)
$\mathcal{T}$ – transition operator	$p(s_{t+1}|s_t)$
$\mathcal{E}$ – emission prob.	$p(o_t|s_t)$
r – reward function	$r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
γ – discount factor	$\gamma \in [0, 1]$ (optional)



Finite versus (vs.) infinite horizon: [**TODO:**]

To solve infinite utilities:

- Finite horizon
- Discounting
- Absorbing state (like the fire hole, overheating)

3.2 Bellman equations

- $T(s, a, s') = P(s'|s, a)$ the prob. of reaching state s' from taking action a at state s
- $R(s, a, s')$ the reward of making the transition
- $Q^*(s, a)$: expected utility starting in state s and having taken action a, then act optimally

$$Q^*(s, a) = \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^*(s') \right] \tag{3.2}$$

  It is the sum over possible next state s', because there is uncertainty of which state s' will be reached, even with the same starting state s and action a.
- $V^*(s)$: value of a state - expected utility starting in state s and acting optimally

$$V^*(s) = \max_a Q^*(s, a) \tag{3.3}$$

$$\Rightarrow \quad V^*(s) = \max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^*(s') \right] \tag{3.4}$$

- $\pi^*(s)$: optimal action / policy from state s

[**TODO: Explain this??**]

$$\mathbf{V(s)} = \mathbb{E}[\mathbf{G_t}|\mathbf{S_t} = \mathbf{s}] = \mathbb{E}\left[\mathbf{R_{t+1}} + \gamma \mathbf{V(S_{t+1})}|\mathbf{S_t} = \mathbf{s}\right] = \mathbf{R_s} + \gamma \sum_{\mathbf{s'}} \mathbf{P_{ss'}} \mathbf{V(s')} \tag{3.5}$$

3.3 Partially Observable Markov Decision Process

POMDP is defined with a tuple: $< S, A, O, P, R, Z, \gamma >$:

- S: state
- A: action
- O: **observation**
- P: transition matrix
- R: reward
- Z: **observation func.**
- γ: discount factor (optional)

$$Z^a_{s'o} = P\left[O_{t+1} = o|S_{t+1} = s', A_t = a\right] \tag{3.6}$$

$$\Rightarrow \quad \begin{cases} V^*(b) \leftarrow \max_a \left[R^a_b + \gamma \sum_{s'} P^a_{bb'} V(b') \right] \\ \pi^*(b) = \arg\max_a \left[R^a_b + \gamma \sum_{s'} P^a_{bb'} V(b') \right] \end{cases} \tag{3.7}$$

4 Research Proposal

4.1 Introduction and Background of Interest

Robots are complex machines designed to support our lives.

Since mid 19th century, its emergence has received research attention in both the academic and industry. The first major application was in the automotive industry. Its presence is entering our lives in different fields is spreading even more and more.

Its appearance has improved

4.2 Literature Review

To my best current knowledge, robot arms have made significant improvement, but are still far from delivering general-purpose tasks. Initially, robot arms are programmed explicitly and only capable of working in the known and constrained environment of factories. Progress in different fields of technology has provided the robots more inputs (e.g., vision, audio, haptic) to operate in more dynamics and challenging environment. State-of-the-art robot arm models can deliver complex human tasks, e.g., cook [Mol], make coffee [Cof], and clean around the house [Bot]. However, the behaviors of these models are still awkwardly disruptive and far from the mastery level of human hand's dexterity. In addition, instead of having the adaptability for a wide range of working conditions, most models still operate in designed environments with known tools and configurations.

4.2.1 Robot Grasping

Grasping object has always been a fundamental problem for robot arms, whether for industrial or service robots. It is a unavoidable task for pick-and-place task and object manipulation. Prior works approach the grasping problem as finding a grasping representation or grasping position. The most successful approaches are empirical, which use deep learning on large dataset to find the best grasping positions or grasping representations. [CRC18]

and use RGB images. Recent works start to dive into multi-modal model, but mostly still for RGB-D images. Only a few consider tactile input and it's still unclear if it bring significant benefit for grasping. [COU+17]

Current state-of-the-art robots has slow and disruptive object grasping and manipulating behavior.

cite Model T (japan) is slow, disruptive, bot handy, bot chef, moley, orient star

the data gap between different approaches: some with million of samples (Sanctuary AI) in simulation, some with collective robots learning (google) → a data-efficient approach for grasping

efficient and smoothing object grasping and manipulation would concerns with both both visual and force sensor. Thus input from computer vision and tactile sensor have to be integrated [HJL18]

The gap between what robots can utilize and what human can utilize

faster learning from demonstration

different type of demonstration (visual, instead of guiding physically)

A meta learning approach for grasping by human demonstration. Prior works approach: Given 3D models, using advanced physical simulation, ... sensing for grasping examining object shape, geometries, center of gravity, etc.

a meta learning approach may also be applicable to non-rigid, deformable object (e.g., clothes, towels)

I argue that it's irrelevant to learn and optimize where to grasp an object or the object characteristics. The optimal grasp depends on the usage of the tools, objects. Someday the robot will have more context about what all tools/objects are for, but, a few demonstration would be straight forward. Thus, I want to develop a model that able to input only a few of human demonstration and be able to learn how to grasp the object appropriately. plan to approach this problem with meta learning. Meta learning is also helpful for transfer learning experience to other task, deal with the current lack of large available dataset for different arms and different tasks.

With the current attention on 3D CV, I believe that soon there will be a approach to extract more information regarding object geometries, materials and usage context. Thus, a multi-modal approach would be nice to pave the way later.

a mapping from human arm behavior to robot arm behavior. helpful for different tasks, either grasping, object manipulation, pick and place

4.3 Approaches and Choice of Methods

4.3.1 Key Points for Improvement

Prior works suffer from, is limited on, make assumptions on ...

4.3.2 Approaches and Research Contributions

sth can be broken down into different approaches, sub-problem, aspects, ...

I intend to clarify these questions, extend these directions, improve ...

pipeline to mapping human arm behavior to robot arm behavior, movement, grasping object

simplified models for rigid, liquid, deformable objects

4.4 Proposed Research Plan

The progress I want to make in robotics, which innately goes back to my bachelor's thesis, concerns the learning of multiple robots for collaborative tasks. For example, a system of two (mobile) robot arms playing 2v2 table tennis, or multiple robot arms carrying on packaging tasks. I wonder, whether there should be a single unified controller or multiple separated ones, how to teach them to delegate different subtasks, will the model optimize each arm with one single subtask or switch between different ones.

The above is a complex long-term challenge. To gradually formalize and break down the problem, there are still knowledge gaps for me to fill. Some specific problems I want to research are: A pipeline for robot arms to imitate human's movements: concerns with pose estimation, controller designs. An approach for faster, purposeful exploration and generalization: not necessarily exploration from scratch, but based on some prior expert inputs.

Integration with generative models: In CV, these models can already output quite realistic samples. Could generative models be incorporated effectively into RL algorithms? They can perhaps generate expert-like samples for imitation learning. Should and how can we evaluate the importance of different samples? Increase the efficiency of the robot arm's movement, in terms of speed and accuracy: integration of time metric into the reward function, data from tactile sensors.

I read your papers about ... let alone the works on ...

There is alignment in our directions

There is an African quote: “If you want to go fast, go alone. If you want to go far, go together”. Robotics and ML are interdisciplinary. This makes the study of RL even more challenging, yet immensely appealing. Thus, it’s amazing that there is a great diversity at IAS, not just in terms of backgrounds, but also academic focuses. Each person works on different topics, MCTS, robot dynamics, imitation learning, etc. Regardless, different works support others and the whole team is pushing the frontier of robotics. I believe that my direction aligns with the team’s to great extent. It would mean a lot for me, to be a part of a team, to learn and cooperate with others.

give compliment to their works t thay vi tri m tuyen phu hop vs t => nen t muon join mention more explicitly regarding the topics, what research they have done

Bibliography

[Bot] *Bot Handy Samsung*. https://research.samsung.com/robot. Accessed: 2022/06/16.

[Cof] *Coffee Master OrionStar*. https://en.orionstar.com/coffeemaster.html. Accessed: 2022/06/16.

[COU+17] R. Calandra, A. Owens, M. Upadhyaya, W. Yuan, J. Lin, E. H. Adelson, and S. Levine. "The feeling of success: Does touch sensing help predict grasp outcomes?" In: *arXiv preprint arXiv:1710.05512* (2017).

[CRC18] S. Caldera, A. Rassau, and D. Chai. "Review of deep learning methods in robotic grasp detection". In: *Multimodal Technologies and Interaction* 2.3 (2018), p. 57.

[HJL18] S. Haddadin, L. Johannsmeier, and F. D. Ledezma. "Tactile robots as a central embodiment of the tactile Internet". In: *Proceedings of the IEEE* 107.2 (2018), pp. 471–487.

[Mol] *Moley Robotics*. https://moley.com/. Accessed: 2022/06/16.

[TBF06] S. Thrun, W. Burgard, and D. Fox. *Probalistic robotics*. Emerald Group Publishing Limited, 2006.