

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
4.2.2	Robot Learning	16
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	18

Bibliography	I
---------------------	----------

1 Introduction

This is my personal learning notes for robotics.

[TODO:]

Abbreviations

RL	Reinforcement Learning
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

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

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 is 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

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

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

$$\begin{aligned}
2 \quad & \bar{\mu}_t = g(u_t, \mu_{t-1}) \\
3 \quad & \bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t \\
4 \quad & K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1} \\
5 \quad & \mu_t = \bar{\mu}_t + K_t [z_t - h(\bar{\mu}_t)] \\
6 \quad & \Sigma_t = (I - K_t H_t) \bar{\Sigma}_t \\
7 \quad & \text{return } \mu_t, \Sigma_t
\end{aligned}$$

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

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

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

$$\begin{array}{ll}
 2 & \bar{\Omega}_t = (A_t \Omega_{t-1}^{-1} A_t^T + R_t)^{-1} \\
 3 & \bar{\xi}_t = \bar{\Omega}_t (A_t \Omega_{t-1}^{-1} \xi_{t-1} + B_t u_t) \\
 4 & \Omega_t = C_t^T Q_t^{-1} C_t + \bar{\Omega}_t \\
 5 & \xi_t = C_t^T Q_t^{-1} z_t + \bar{\xi}_t \\
 6 & \text{return } \xi_t, \Omega_t
 \end{array} \left. \vphantom{\begin{array}{l} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}} \right\} \begin{array}{l} \mathcal{O}(n^{2,8}) \\ \mathcal{O}(n^2) \end{array}$$

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

$$\begin{array}{ll}
 2 & \mu_{t-1} = \Omega_{t-1}^{-1} \xi_{t-1} \\
 3 & \bar{\Omega}_t = (G_t \Omega_{t-1}^{-1} G_t^T + R_t)^{-1} \\
 4 & \bar{\xi}_t = \bar{\Omega}_t g(u_t, \mu_{t-1}) \\
 5 & \bar{\mu}_t = g(u_t, \mu_{t-1}) \\
 6 & \Omega_t = \bar{\Omega}_t + H_t^T Q_t^{-1} H_t \\
 7 & \xi_t = \bar{\xi}_t + H_t^T Q_t^{-1} [z_t - h(\bar{\mu}_t) - H_t \bar{\mu}_t]
 \end{array}$$

- **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: $m = \{m_1, m_2, \dots, m_N\}$

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

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

- Unexpected object

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

- Failures

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

- Random measurements

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

[TODO: Add image, plot]

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

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.)
	more accurate but post-the-fact (not for motion planning)
Use for Probabilistic 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:

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

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

$$\begin{array}{ll}
 2 & \mu = \frac{1}{2} \frac{(x - x') \cos \theta + (y - y') \sin \theta}{(y - y') \cos \theta - (x - x') \sin \theta} \\
 3 & x^* = \frac{x+x'}{2} + \mu(y - y') \\
 4 & y^* = \frac{y+y'}{2} + \mu(x' - x) \\
 5 & r^* = \sqrt{(x - x^*)^2 + (y - y^*)^2} \\
 6 & \Delta\theta = \text{atan2}(y' - y^*, x' - x^*) - \text{atan2}(y - y^*, x - x^*) \\
 7 & \hat{v} = \frac{\Delta\theta}{\Delta t} r^* \\
 8 & \hat{\omega} = \frac{\Delta\theta}{\Delta t} \\
 9 & \hat{\gamma} = \frac{\theta' - \theta}{\Delta t} - \hat{\omega} \\
 10 & \text{return } p(v - \hat{v}, \alpha_1|v| + \alpha_2|\omega|) \cdot p(\omega - \hat{\omega}, \alpha_3|v| + \alpha_4|\omega|) \cdot p(\hat{\gamma}, \alpha_5|v| + \alpha_6|\omega|)
 \end{array}
 \left. \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array} \right\} \begin{array}{l} \text{Invert the motion model} \\ \\ \\ \\ \text{compared actual velocities with the desired} \end{array}$$

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

```

2    $\hat{v} = v + \text{sample}(\alpha_1|v| + \alpha_2|\omega|)$ 
3    $\hat{\omega} = \omega + \text{sample}(\alpha_3|v| + \alpha_4|\omega|)$ 
4    $\hat{\gamma} = \text{sample}(\alpha_5|v| + \alpha_6|\omega|)$ 
5    $x' = x - \frac{\hat{v}}{\hat{\omega}} \sin \theta + \frac{\hat{v}}{\hat{\omega}} \sin(\theta + \hat{\omega}\Delta t)$ 
6    $y' = y + \frac{\hat{v}}{\hat{\omega}} \cos \theta - \frac{\hat{v}}{\hat{\omega}} \cos(\theta + \hat{\omega}\Delta t)$ 
7    $\theta' = \theta + \hat{\omega}\Delta t + \hat{\gamma}\Delta t$ 
8   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 $\frac{b}{6} \sum_{i=1}^{12} \text{rand}(-1, 1)$
- Sample triangle distribution(b): return $b.\text{rand}(-1, 1).\text{rand}(-1, 1)$

2.3.3 Odometry Motion Model

Only available after the robot has moved

⇒ only use for filter algorithm

not for accurate motion planning and control

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

```

2    $\delta_{rot1} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$ 
3    $\delta_{trans} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$ 
4    $\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$ 
5    $\hat{\delta}_{rot1} = \text{atan2}(y' - y, x' - x) - \theta$ 
6    $\hat{\delta}_{trans} = \sqrt{(x - x')^2 + (y - y')^2}$ 
7    $\hat{\delta}_{rot2} = \theta' - \theta - \hat{\delta}_{rot1}$ 
8    $p_1 = \text{prob}(\delta_{rot1} - \hat{\delta}_{rot1}, \alpha_1\hat{\delta}_{rot1} + \alpha_2\hat{\delta}_{trans})$ 
9    $p_2 = \text{prob}(\delta_{trans} - \hat{\delta}_{trans}, \alpha_3\hat{\delta}_{trans} + \alpha_4(\hat{\delta}_{rot1} + \hat{\delta}_{rot2}))$ 
10   $p_3 = \text{prob}(\delta_{rot2} - \hat{\delta}_{rot2}, \alpha_1\hat{\delta}_{rot2} + \alpha_2\hat{\delta}_{trans})$ 
11  return  $p_1.p_2.p_3$        $(= p(x_t|u_t, x_{t-1}))$ 

```

} ⇒ inverse motion model

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})

```

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

```

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)

do :

$x_t = \text{sample_motion_model}(u_t, x_{t-1})$

$\pi = p(x_t|m)$

until $\pi > 0$

return $\langle x_t, \pi \rangle$

3 Markov Decision Process

3.1 Definitions

3.1.1 Markov Chain

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

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

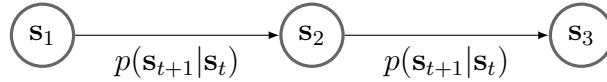
\mathcal{S} – state space

$s \in \mathcal{S}$ (discrete or continuous)

\mathcal{T} – transition operator

$p(s_{t+1}|s_t)$ 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

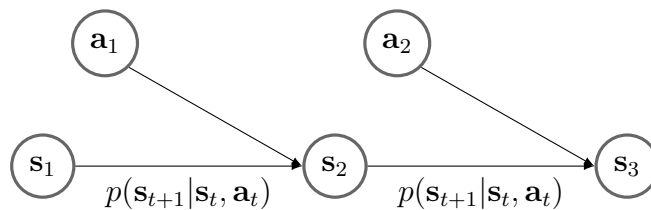
$p(s_{t+1}|s_t)$ (now a tensor)

r – reward function

$r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}, \quad r(s_t, a_t)$

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:

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

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.**

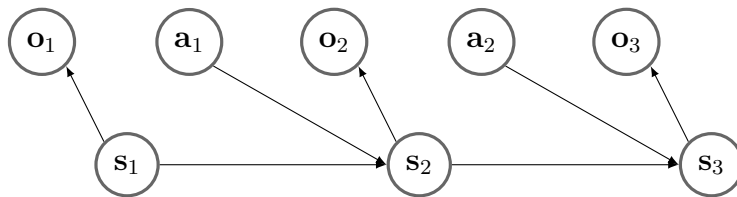
$p(o_t|s_t)$

r – reward function

$r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

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

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

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

$$V^*(s) = \max_a Q^*(s, a) \quad (3.3)$$

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

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

[TODO: Explain this??]

$$\mathbf{V}(s) = \mathbb{E}[\mathbf{G}_t | \mathbf{S}_t = s] = \mathbb{E}[\mathbf{R}_{t+1} + \gamma \mathbf{V}(\mathbf{S}_{t+1}) | \mathbf{S}_t = s] = \mathbf{R}_s + \gamma \sum_{s'} \mathbf{P}_{ss'} \mathbf{V}(s') \quad (3.5)$$

3.3 Partially Observable Markov Decision Process

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

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

$$Z_{s'o}^a = P[O_{t+1} = o | S_{t+1} = s', A_t = a] \quad (3.6)$$

$$\Rightarrow \begin{cases} V^*(b) \leftarrow \max_a \left[R_b^a + \gamma \sum_{s'} P_{bb'}^a V(b') \right] \\ \pi^*(b) = \arg \max_a \left[R_b^a + \gamma \sum_{s'} P_{bb'}^a V(b') \right] \end{cases} \quad (3.7)$$

4 Research Proposal

4.1 Introduction and Background of Interest

[TODO: ignore for now]

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 dynamics and unstructured environment. State-of-the-art robot arm models can deliver complex human tasks, e.g., cooking [Mol], making coffee [Cof], and cleaning 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 is one of the fundamental and unavoidable problems for robot arms, whether for the logistics or service robots. Most successful approaches are empirical, using deep learning on large dataset to find the best grasping representations. Majority of models are supervised training using single-modal input, i.e. RGB images. A few has considered multi-modal input: RGB with depth images, tactile input. [CRC18]

[TODO: previous grasping approach that use tactile input?]

[[COU+17](#)]

4.2.2 Robot Learning

Works on reaching target, handling objects

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

The progress I wish to work towards concerns robotic collective learning for collaborative tasks. For example, a system of multiple robot arms playing 2v2 table tennis, packaging an item or cooking. Building up towards the mentioned progress, the improvement points I want to make are as follows:

- Extend the usage of tactile sensors to learning object grasping and manipulation. Tactile sensor has mostly been used for the development of safety robot / soft robotics [HJL18]. Previous works using tactile sensor on robotic grasping focus on [TODO: sth, read UC Bekerly's paper]
- Robotic grasping mostly with rigid object.
- Meta Reinforcement Learning (RL) / generalization problem of RL
- Data efficiency problem

[FORMAT: Prior works suffer from, limited, make assumptions on, I want to extend ...]

4.3.2 Approaches and Research Contributions

Directions / smaller problems I want to pursue:

- A data collection pipeline for robot arms. Preferably a mapping to imitate human's movements, which, at the same time, collecting visual and haptic information.
- Integrate tactile information to robotic object grasping. Combining with visual data, process the haptic data to extract other possible object's characteristics, e.g., weight, material, deformability. This will then possibly extend to some learning on object manipulation.
- An approach for faster, purposeful exploration and generalization: not necessarily exploration from scratch, but based on some prior expert inputs.
-

Integration with generative models. 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?

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

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

simplified models for rigid, liquid, deformable objects

Prior work mostly concern with rigid objects. tactile inputs would bring adaptability, optimal grip force, and perhaps more underlying information regarding material, etc. for non-rigid, soft, deformable object. dynamics load

in addition, it would enable more complex object manipulation tasks that computer vision alone would be insufficient

- A approach to map human behaviors to single-arm manipulator. Use for visual and tactile data collection. Possibly generalized for different arm manipulator, gripper, sensor input, etc. As demonstration for other approach, imitation learning, object grasping, reaching, object manipulation, etc.
- Robotic grasping as few-shot meta-[RL](#)

I also curious with the usage of some deep learning techniques: generative model, score matching, etc.

4.4 Proposed Research Plan

[TODO:]

[TODO: How the IAS lab is related to this research plan?]

- There is alignment in our directions
- t thay vi tri m tuyen phu hop vs t => nen t muon join
- 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.

[TODO: Give compliments to their works!]

- mention more explicitly regarding the topics, what research they have done
- I read your papers about ... let alone the works on ...

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. Johannessmeier, 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.