Robotics Notes

Huu Duc Nguyen M.Sc.

07 May 2022

Contents

1 Introduction

This is my personal learning notes for robotics.

[**TODO:**]

Abbreviations

ML	Machine Learning
DL	Deep Learning
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

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

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

$p(x_t \| u_t, x_{t-1})$	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: $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$

```
2   $\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t$                                  } incorporate $u_t$ - prediction step - $\mathcal{O}(n^2)$
3   $\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$                           }
4   $K_t = \overline{\Sigma}_t C_t^T \left(C_t \overline{\Sigma}_t C_t^T + Q_t\right)^{-1}$   (Kalman gain) }
5   $\mu_t = \overline{\mu}_t + K_t(z_t - C_t \overline{\mu}_t)$                    } incorporate $z_t$ - correction step - $\mathcal{O}(n^{2,8})$
6   $\Sigma_t = (I - K_t C_t)\overline{\Sigma}_t$                                  }
7   return $\mu_t, \Sigma_t$                                                       $\Rightarrow$ 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

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t$$

$$z_t = h(x_t) + \delta_t$$

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

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

```
2    $\overline{\mu}_t = g(u_t, \mu_{t-1})$
3    $\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
4    $K_t = \overline{\Sigma}_t H_t^T \left(H_t \overline{\Sigma}_t H_t^T + Q_t\right)^{-1}$
5    $\mu_t = \overline{\mu}_t + K_t[z_t - h(\overline{\mu}_t)]$
6    $\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t$
7    return $\mu_t, \Sigma_t$
```

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

```
2    $\overline{\Omega}_t = (A_t \Omega_{t-1}^{-1} A_t^T + R_t)^{-1}$
3    $\overline{\xi}_t = \overline{\Omega}_t (A_t \Omega_{t-1}^{-1} \xi_{t-1} + B_t u_t)$        } $\mathcal{O}(n^{2,8})$
4    $\Omega_t = C_t^T Q_t^{-1} C_t + \overline{\Omega}_t$
5    $\xi_t = C_t^T Q_t^{-1} z_t + \overline{\xi}_t$        } $\mathcal{O}(n^2)$
6    return $\xi_t, \Omega_t$
```

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

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

- ***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

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

- Unexpected object

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

- Failures

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

- Random measurements

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

[**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})

```
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    $r^* = \sqrt{(x-x^*)^2 + (y-y^*)^2}$
          } Invert the motion model
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    $\hat{\gamma} = \frac{\theta'-\theta}{\Delta t} - \hat{\omega}$
          } compared actual velocities with the desired
10   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})

```
2   $\hat{v} = v + sample(\alpha_1|v| + \alpha_2|\omega|)$
3   $\hat{\omega} = \omega + sample(\alpha_3|v| + \alpha_4|\omega|)$
4   $\hat{\gamma} = sample(\alpha_5|v| + \alpha_6|\omega|)$
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 $\frac{b}{6} \sum_{i=1}^{12} rand(-1, 1)$
- 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})

```
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   $\hat{\delta}_{trans} = \sqrt{(x - x')^2 + (y - y')^2}$        } $\Rightarrow$ inverse motion model
7   $\hat{\delta}_{rot2} = \theta' - \theta - \hat{\delta}_{rot1}$
8   $p_1 = prob(\delta_{rot1} - \hat{\delta}_{rot1}, \alpha_1\hat{\delta}_{rot1} + \alpha_2\hat{\delta}_{trans})$
9   $p_2 = prob(\delta_{trans} - \hat{\delta}_{trans}, \alpha_3\hat{\delta}_{trans} + \alpha_4(\hat{\delta}_{rot1} + \hat{\delta}_{rot2}))$
10  $p_3 = prob(\delta_{rot2} - \hat{\delta}_{rot2}, \alpha_1\hat{\delta}_{rot2} + \alpha_2\hat{\delta}_{trans})$
11  return  $p_1.p_2.p_3$    $(= p(x_t|u_t, x_{t-1}))$
```

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    $\hat{\delta}_{rot1} = \delta_{rot1} - sample(\alpha_1 \delta_{rot1} + \alpha_2 \delta_{trans})$
6    $\hat{\delta}_{trans} = \delta_{trans} - sample(\alpha_3 \delta_{trans} + \alpha_4(\delta_{rot1} + \delta_{rot2}))$
7    $\hat{\delta}_{rot2} = \delta_{rot2} - sample(\alpha_1 \delta_{rot2} + \alpha_2 \delta_{trans})$
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)

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

3 Markov Decision Process

3.1 Definitions

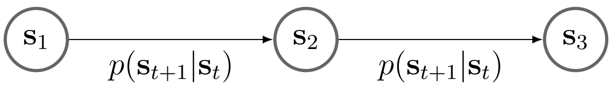
3.1.1 Markov Chain

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}.

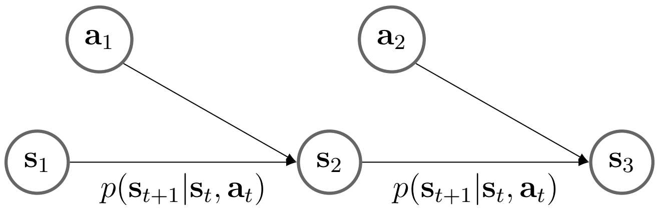

$\mathbf{s}_1$
$\mathbf{s}_2$
$\mathbf{s}_3$
$p(\mathbf{s}_{t+1}|\mathbf{s}_t)$
$p(\mathbf{s}_{t+1}|\mathbf{s}_t)$

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	$r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,	$r(s_t, a_t)$
γ – discount factor	$\gamma \in [0, 1]$	(optional)


$\mathbf{a}_1$
$\mathbf{a}_2$
$\mathbf{s}_1$
$\mathbf{s}_2$
$\mathbf{s}_3$
$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$

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:

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

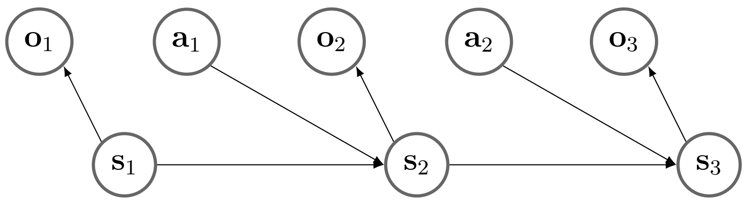
MDP state projects an expectimax-like search tree [**TODO: add image**]

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??**]

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

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)

$$Z^a_{s'o} = P\left[O_{t+1} = o|S_{t+1} = s', A_t = a\right] \tag{3.6}$$
$$\Rightarrow \begin{cases} V^*(b) \leftarrow \max_a \left[R^a_b + \gamma \sum_{s'} P^a_{bb'} V(b')\right] \\ \pi^*(b) = \arg\max_a \left[R^a_b + \gamma \sum_{s'} P^a_{bb'} V(b')\right] \end{cases} \tag{3.7}$$

4 Research Proposal

4.1 Introduction and Background of Interest

[**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. [**TODO: move to unstructured environments and challenges**]

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 dynamic and unstructured environments. State-of-the-art robot manipulators can deliver complex 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 the human hand's dexterity. In addition, instead of having the adaptability for a wide range of conditions, most models still operate in designed environments with known tools and settings.

4.2.1 Robotic Grasping

Grasping is one of the critical and unavoidable problems for robot arm manipulators, especially for logistic and service robots. Approaches, which take tactile input, use it as the sole input to improve grasp stability, adaptability to object's weight [BLJ+11; LBK+14], a few combine with other modality for improvement [COU+17]. Overall, most successful grasping approaches are empirical, using deep learning on a large dataset to find the best grasping candidates. Majority of models are supervised learning using single-modal input, i.e. RGB images. A few have considered multi-modal inputs: RGB with depth images or tactile input. [BMA+13; CRC18; LLC+19; KBK+20]

4.2.2 Reinforcement Learning

Reinforcement Learning (RL) is a branch of Machine Learning (ML) approaches for learning decision making from experiences. Given a sufficient amount of input data, it can potentially surpass human's ability for reasoning, e.g., playing games like Go, chess, Atari. Most works on robot manipulators are currently directed to object picking and hand-over tasks. To meet the need for data, sim-to-real transfer techniques [KBK+20] and collective learning are being studied [YLK+17]. Unlike in other domains of ML, it's difficult to implement transfer learning with RL, given different robot arms, gripper and sensor types. Various directions are being researched to tackle this issue, i.e. offline RL, novel exploration strategies, multi-task learning, meta-learning. [KBP13; Li17; ADB+17; SKS21]

4.3 Approaches and Choice of Methods

4.3.1 Key Points for Improvement

Prior works on robotic manipulators were limited on their utilization of tactile sensors. Complex hand and arm behaviors require some level of force and pressure feedback. Thus, these sensors can potentially leverage the manipulators to the dexterity level of human's arms. However, tactile sensors are currently been used mostly for the development of soft robotics [HJL18]. For grasping tasks, despite certain successes from using visual and tactile input separately, there hasn't being a multi-modal approach capable of combining the strengths of both sides, i.e., accurate localization, adaptability to object's weight and deformability. A model, which has knowledge from both visual and tactile sources, promises to deliver complex manipulation tasks with diverse objects.

It's still challenging to generalize and apply RL to different robotic manipulator's tasks. In general, RL did make a convincing case about its potential to surpass human's ability in some specific applications, e.g., playing Go, chess. However, in other fields, practical applications are yet scattered and limited. For robotic manipulators, current focuses are still around object picking and handling tasks [SKS21]. One of the major causes for this situation is the current gap in generalization and transfer learning. Directions for improvements are meta-learning, multi-task learning, etc. [KBP13].

4.3.2 Approaches and Research Contributions

The progress I wish to work towards concerns robotic collective learning for collaborative tasks. For example, a system of multiple robot arms learning to play 2v2 table tennis, pack an item or cook. Building up towards this progress, I intend to work and gain more experiences on the following problems:

1. A multi-modal approach for robotic picking task: combines visual and tactile input into a Deep Learning (DL) model. I'm curious about what kinds of underlying information (weight, material, deformability) can we extract from (different) tactile sensors.
2. Visual imitation learning: copies behaviors from visual demonstrations (videos). Based on previous progress on teleoperation [HVWY+20], visual imitation learning [FYZ+17; SPG19], I want to extend further to learn in different problem settings, with generalized tasks, and also apply offline RL.
3. A data collection pipeline for robot manipulators: preferably from using the above visual imitation learning, to map human's arm movements to manipulator's. The data would contain both visual and tactile information. It would then be used for the learning of other robotic tasks or in other settings (different types of arm, gripper, etc.).
4. Meta-learning with different manipulation tasks.

Moreover, I'm also interested in exploring the usage of certain DL models, techniques:

- Generative model: Generating expert-like samples.
- Score matching
- Exploration for imitation learning

4.4 Proposed Research Plan

[**TODO:**]

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

- There is alignment in our directions
- Fit the position

[**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

[ADB+17] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. “Deep reinforcement learning: A brief survey”. In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38.

[BLJ+11] Y. Bekiroglu, J. Laaksonen, J. A. Jorgensen, V. Kyrki, and D. Kragic. “Assessing grasp stability based on learning and haptic data”. In: *IEEE Transactions on Robotics* 27.3 (2011), pp. 616–629.

[BMA+13] J. Bohg, A. Morales, T. Asfour, and D. Kragic. “Data-driven grasp synthesis—a survey”. In: *IEEE Transactions on robotics* 30.2 (2013), pp. 289–309.

[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.

[FYZ+17] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. “One-shot visual imitation learning via meta-learning”. In: *Conference on Robot Learning*. PMLR. 2017, pp. 357–368.

[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.

[HVWY+20] A. Handa, K. Van Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox. “Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system”. In: *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*. IEEE. 2020, pp. 9164–9170.

[KBK+20] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber. “A survey on learning-based robotic grasping”. In: *Current Robotics Reports* 1.4 (2020), pp. 239–249.

[KBP13] J. Kober, J. A. Bagnell, and J. Peters. “Reinforcement learning in robotics: A survey”. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274.

[LBK+14] M. Li, Y. Bekiroglu, D. Kragic, and A. Billard. “Learning of grasp adaptation through experience and tactile sensing”. In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE. 2014, pp. 3339–3346.

[Li17] Y. Li. “Deep reinforcement learning: An overview”. In: *arXiv preprint arXiv:1701.07274* (2017).

[LLC+19] Y. Li, Q. Lei, C. Cheng, G. Zhang, W. Wang, and Z. Xu. “A review: Machine learning on robotic grasping”. In: *Eleventh International Conference on Machine Vision (ICMV 2018)*. Vol. 11041. SPIE. 2019, pp. 775–783.

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

[SKS21] B. Singh, R. Kumar, and V. P. Singh. “Reinforcement learning in robotic applications: a comprehensive survey”. In: *Artificial Intelligence Review* (2021), pp. 1–46.

[SPG19] P. Sharma, D. Pathak, and A. Gupta. “Third-person visual imitation learning via decoupled hierarchical controller”. In: *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)* 32 (2019).

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

[YLK+17] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine. “Collective robot reinforcement learning with distributed asynchronous guided policy search”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 79–86.