Lecture 37: Final Exam Review

Mark Hasegawa-Johnson

5/2023

Chinese imperial examination candidates gathering around the wall where the results are posted
By Qiu Ying (仇英) (attributed) - In the collection of the National Palace Museum, Public Domain, https://commons.wikimedia.org/w/index.php?curid=598340

Outline

- How to take the exam
- What is covered
- Sample problems

How to take the exam

- Be here in Lincoln Hall theater, 8am on Tuesday May 9
- Bring pencils, erasers, ID
- You can have up to three pages of notes, front & back

Outline

- How to take the exam
- What is covered
- Sample problems

What’s covered

- 2/3 of the exam (about 10 problems or problem parts): material since exam 2
- 1/6 of the exam (about 2.5 problems or problem parts): material from exam 1
- 1/6 of the exam (about 2.5 problems or problem parts): material from exam 2

Outline of the course

- Exam 1 material
  - Random variables & decision theory
  - Regression, classifiers, neural nets
- Exam 2 material
  - Search, Games, and Theorem Proving
  - Bayesian Networks & HMM
- Exam 3 new material
  - Computer vision, CNN, Kalman filter
  - MDP, reinforcement learning, and robotics

Outline of exam 3 new material

- Computer vision
  - Image formation
  - CNN
  - Kalman filter
- RL & Robotics
  - MDP, Bellman’s equation, Value iteration, Policy iteration
  - Model-based RL, exploration vs. exploitation
  - Model-free RL: Q-learning, deep Q-learning, Actor-critic learning
  - Robotics: Configuration space, inverse kinematics, PID controller

Vanishing point

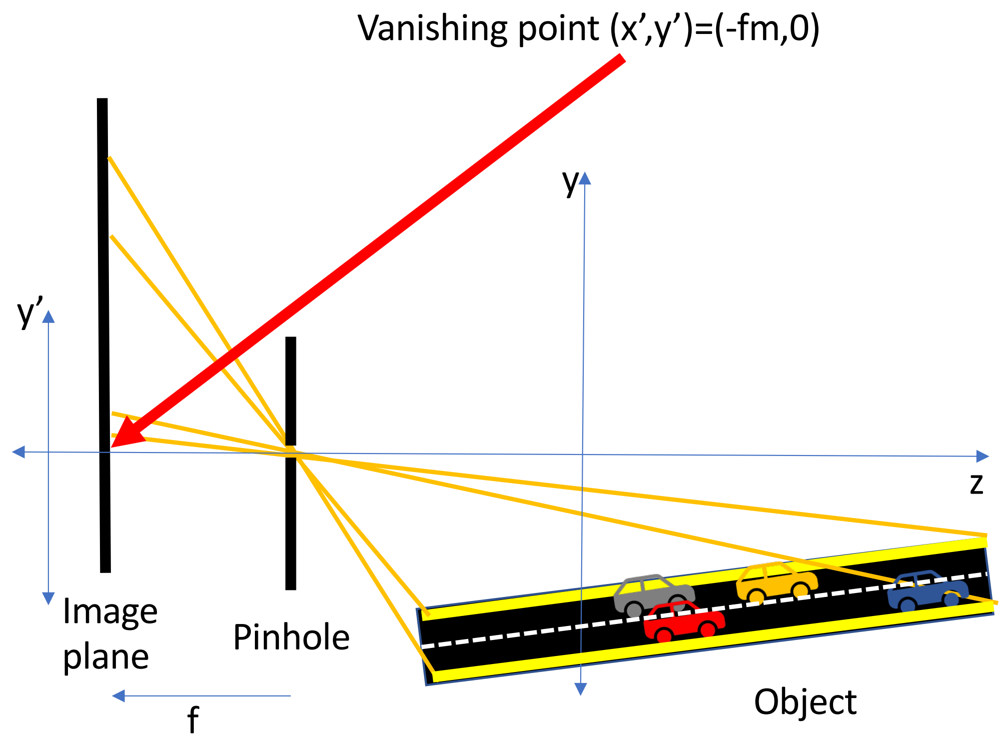
- Plug equations for the lines into the pinhole camera equations:

$$\frac{x_1'}{f} = -\frac{az + c_1}{z}, \qquad \frac{y_1'}{f} = -\frac{bz + d_1}{z}$$

$$\frac{x_2'}{f} = -\frac{az + c_2}{z}, \qquad \frac{y_2'}{f} = -\frac{bz + d_2}{z}$$

- As $z \rightarrow \infty$, the two lines converge to the vanishing point, which depends only on the ***slope*** of the lines, not on their shift:

$$(x', y') = (-fa, -fb)$$



Convolution

$$h[k,l] * x[k,l] = \sum_{i} \sum_{j} h[k-i, l-j] x[i,j]$$

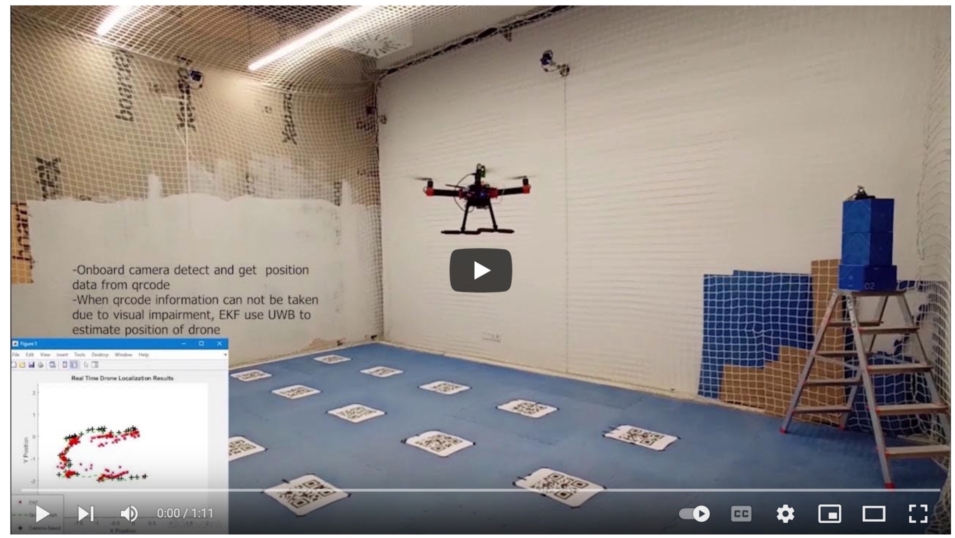
Kalman Filter

Prediction step: given $\mu_{t-1|t-1}$ and $\sigma^2_{t-1|t-1}$, we can predict where the fish might go at time t, but with increased uncertainty:

$$\mu_{t|t-1} = \mu_{t-1|t-1} + \mu_\Delta$$
$$\sigma^2_{t|t-1} = \sigma^2_{t-1|t-1} + \sigma^2_\Delta$$

Update step: given the observation x_t, we can refine our estimate, and reduce our uncertainty:

$$k_t = \frac{\sigma^2_{t|t-1}}{\sigma^2_{t|t-1} + \sigma^2_\epsilon}$$
$$\mu_{t|t} = \mu_{t|t-1} + k_t\left(x_t - \left(\mu_{t|t-1} + \mu_\epsilon\right)\right)$$
$$\sigma^2_{t|t} = \sigma^2_{t|t-1}(1 - k_t)$$



Drone Localization based on Extended Kalman Filter (EKF) with UWB sensors and camera, https://www.youtube.com/watch?v=kC8FgmhhSB8

Markov Decision Process

- Bellman Equation:

$$U(s) = R(s) + \gamma \max_{a} \sum_{s'} P(s'|s, a) U(s')$$

- Value Iteration:

$$U_i(s) = R(s) + \gamma \max_{a} \sum_{s'} P(s'|s, a) U_{i-1}(s')$$

- Policy Iteration:
  - Policy evaluation: $U_i(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s)) U_i(s')$
  - Policy improvement: $\pi_{i+1}(s) = \underset{a}{\operatorname{argmax}} R(s) + \gamma \sum_{s'} P(s'|s, a) U_i(s')$

Model-based RL

- Model = $P(s'|s,a)$ and $R(s)$
- The observation, model, policy loop
  - observe the results of your actions, re-estimate model, optimize policy
- Exploration versus Exploitation
  - Epsilon-first learning: try every action, in every state, at least $1/\epsilon$ times.
  - Epsilon-greedy learning: explore w/prob. ϵ, exploit w/prob $1-\epsilon$.

Model-free RL: Q-learning

- Q(s,a) – the "quality" of an action

$$Q(s,a) = R(s) + \gamma \sum_{s'} P(s'|s,a)U(s')$$
$$U(s) = \max_{a \in A(s)} Q(s,a)$$

- Q-learning
- Off-policy learning: TD

$$Q_{local}(s_t, a_t) = R_t(s_t) + \gamma \max_{a' \in A(s_{t+1})} Q_t(s_{t+1}, a')$$
$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha\big(Q_{local}(s_t, a_t) - Q_t(s_t, a_t)\big)$$

- On-policy learning: SARSA

$$a_{t+1} = \pi_t(s_{t+1})$$
$$Q_{local}(s_t, a_t) = R_t(s_t) + \gamma Q_t(s_{t+1}, a_{t+1})$$

Deep RL

- Imitation learning is not really RL. Assume that you have labeled data, labeled with a human's actions in the same state. Train the DNN with:

$$\mathcal{L} = -\log f_{a_t}(\vec{s}_t)$$

- Deep Q learning is model-free RL, like Q-learning, but compute Q(s,a) using a neural network

$$\mathcal{L} = \frac{1}{2} E[(f(\vec{s}_t, \vec{a}_t) - Q_{local}(\vec{s}_t, \vec{a}_t)\ )^2]$$

$$Q_{local}(\vec{s}_t, \vec{a}_t) = R_t(\vec{s}_t) + \gamma \max_{\vec{a}'} f(\vec{s}_{t+1}, \vec{a}')$$

The Actor-Critic Algorithm

$$\pi_a(s) = \text{Probability that } a \text{ is the best action in state } s$$

$$Q(s,a) = \text{Expected sum of future rewards if } (s,a)$$

- The critic is trained as a normal deep Q-learner:

$$\mathcal{L}_{critic} = \frac{1}{2} E[(f(\vec{s}_t, \vec{a}_t) - Q_{local}(\vec{s}_t, \vec{a}_t)\,)^2]$$

- The actor is trained as an imitation learner, trying to compute a policy that will maximize the expected value of future rewards:

$$\mathcal{L}_{actor} = -\sum_a \pi_a(s) Q(s,a)$$

Robotics: Inverse kinematics

- Obstacles are things in the workspace, $\mathcal{W}$, that we don't want to run into.
- We want to plan a path through configuration space, C, such that we don't run into any obstacle.
- In order to do that, we need **inverse kinematics**: a function that converts obstacles in the workspace, $\mathcal{W}_{\text{obs}}$, into equivalent obstacles in configuration space, C_{obs}.

$$C_{\text{obs}} = \{q: \exists b: \varphi_b(q) \in \mathcal{W}_{\text{obs}}\}$$

- For example: we usually do this by just exhaustively testing every point in configuration space, to see if it runs into an obstacle.

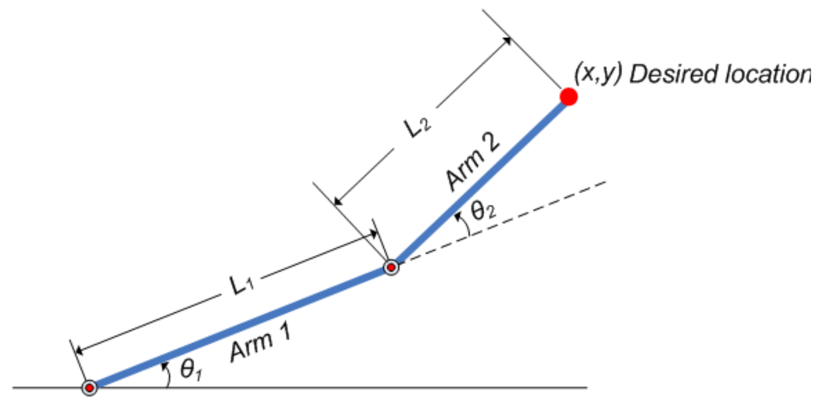


Image © https://www.mathworks.com/help/fuzzy/modeling-inverse-kinematics-in-a-robotic-arm.html

Outline

- How to take the exam
- What is covered
- Sample problems

Exam 3 new material: relevant problems

Topic	Sp22 Review Exam 1	Sp22 Exam 1	Sp22 Conflict Exam 1	Sp22 Review Exam 3	Sp22 Exam 3	Sp23 Review Exam 3
Image formation	17-18	7	7			
CNN	19-20					
Kalman filter						1,2
MDP				16-18, 20	10	
Model-based learning				19(bc), 23	11	
Model-free learning				19(ad), 21-2, 24	12, 13	
Robotics				25, 26	14	