Topics in Reinforcement Learning: AlphaZero, ChatGPT, Neuro-Dynamic Programming, Model Predictive Control, Discrete Optimization, Applications

Arizona State University
Course CSE 691, Spring 2025

Links to Class Notes, Videolectures, and Slides at
http://web.mit.edu/dimitrib/www/RLbook.html

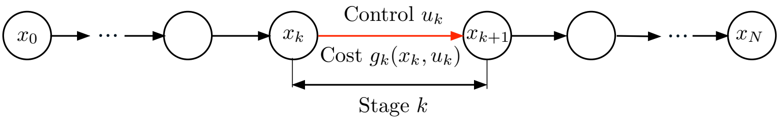
Prof. Dimitri P. Bertsekas (dimitrib@mit.edu)
and
Dr Yuchao Li (yuchaoli@asu.edu)
Video Credit: Alejandro P. Riveiros William Emanuelsson Pratyusha Musunuru

Lecture 7
Applications of Approximation in Value Space to Multiagent and Multiple Objects Tracking Problems

Outline

Outline

Review: DP Algorithm for Deterministic Problems



Go backward to compute the optimal costs $J_k^*(x_k)$ of the x_k-tail subproblems (off-line training - involves lots of computation)

Go forward to construct optimal control sequence $\{u_0^*, \ldots, u_{N-1}^*\}$ (on-line play)

- Start with

$$u_0^* \in \arg \min_{u_0 \in U_0(x_0)} \Big[g_0(x_0, u_0) + J_1^*\big(f_0(x_0, u_0)\big) \Big], \quad x_1^* = f_0(x_0, u_0^*).$$

- Sequentially, going forward, for $k = 1, 2, \ldots, N-1$, set

$$u_k^* \in \arg \min_{u_k \in U_k(x_k^*)} \Big[g_k(x_k^*, u_k) + J_{k+1}^*\big(f_k(x_k^*, u_k)\big) \Big], \quad x_{k+1}^* = f_k(x_k^*, u_k^*).$$

Review: Approximation in Value Space with One-Step Lookahead

$$\boxed{\text{At } x_k} \longrightarrow \min_{u_k} \Big\{ \overbrace{g_k(x_k, u_k)}^{\text{First Step}} + \overbrace{\tilde{J}_{k+1}\big(f_k(x_k, u_k)\big)}^{\text{“Future”}} \Big\} \quad \text{“On-Line Play”}$$

Simplified minimization **Cost-to-go approximation**

We replace $J_k^*(x_k)$ with an approximation $\tilde{J}_k$ during on-line play

- Start with

$$\tilde{u}_0 \in \arg \min_{u_0 \in U_0(x_0)} \Big[g_0(x_0, u_0) + \tilde{J}_1\big(f_0(x_0, u_0)\big) \Big].$$

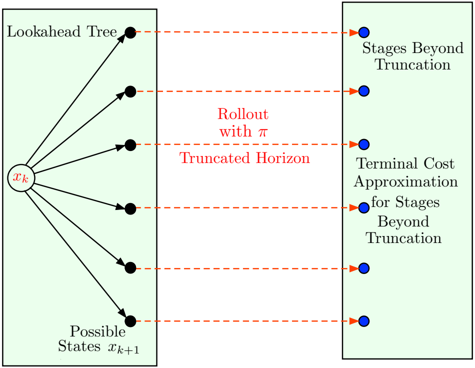
- We set $\tilde{x}_1 = f_0(x_0, \tilde{u}_0)$
- Sequentially, going forward, for $k = 1, 2, \ldots, N-1$, set

$$\tilde{u}_k \in \arg \min_{u_k \in U_k(\tilde{x}_k)} \Big[g_k(\tilde{x}_k, u_k) + \tilde{J}_{k+1}\big(f_k(\tilde{x}_k, u_k)\big) \Big], \quad \tilde{x}_{k+1} = f_k(\tilde{x}_k, \tilde{u}_k).$$

Two Challenges

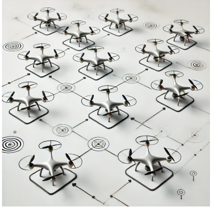
- How to construct the approximation $\tilde{J}_{k+1}$ offline?
- How to solve the minimization problem online?

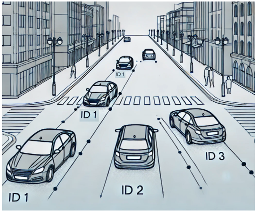
Review: Truncated Rollout with One-Step Lookahead



- Truncated rollout encompass all key ingredients: a base policy π and a terminal cost approximation $\hat{J}$: $\tilde{J}_{k+1}(x_{k+1}) = \hat{J}(x_{k+m+1}) + \sum_{\ell=k+1}^{k+m} g_\ell(x_\ell, \mu_\ell(x_\ell))$.
- When no rollout is involved, the terminal cost approximation $\hat{J}$ serves as $\tilde{J}$
- When there is no truncation, we use J_π as $\tilde{J}$
- However, the challenge of computing the minization remains!

Focus of This Lecture: Solving the Minimization in Approximation in Value Space Efficiently





First Step "Future"

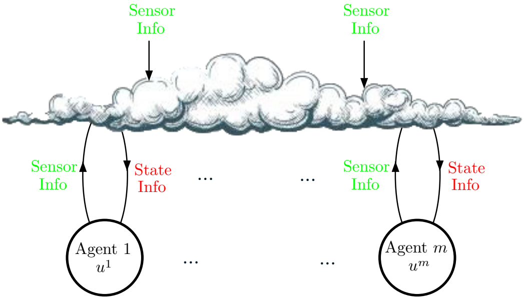
$$\boxed{\text{At } x_k} \longrightarrow \min_{u_k} \Big\{ g_k(x_k, u_k) + \tilde J_{k+1}\big(f_k(x_k, u_k)\big) \Big\} \quad \text{"On-Line Play"}$$

Simplified minimization **Cost-to-go approximation**

- We will consider multiagent coordination and multi-object tracking problems.
- The computational demand can be reduced by leveraging the problem structure, as in multiagent rollout, still yielding a legitimate Newton's step.
- The terminal cost approximation $\tilde J$ can be suitably designed to enhance the minimization calculation.

Outline

1. Review: Approximation in Value Space - Truncated Rollout
2. Multiagent Rollout for Warehouse Robots Path Planning
3. Approximation in Value Space for Multiple Object Tracking/Data Association Problem



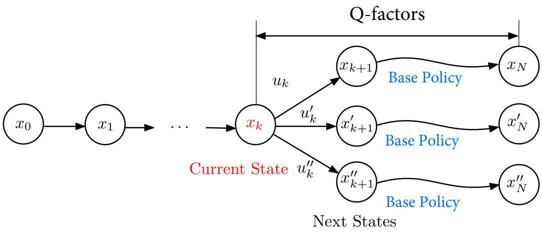
At each time: Agents have exact state info; choose their controls as a function of state

Model: A discrete-time (possibly stochastic) system with state x and control u

- Decision/control has m components $u = (u^1, \ldots, u^m)$ corresponding to m "agents"
- "Agents" is just a metaphor - the important math structure is $u = (u^1, \ldots, u^m)$
- For every policy $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$, the functions μ_k, $k = 0, \ldots, N-1$ take the form

$$\mu_k(x) = \big(\mu_k^1(x), \mu_k^2(x), \ldots, \mu_k^m(x)\big)$$

Standard (Truncated) Rollout for Multiagent Problem



Standard (truncated) rollout for multiagent problem

- Suppose for simplicity that control constraint set $U_k(x_k)$ has a Cartesian product structure:

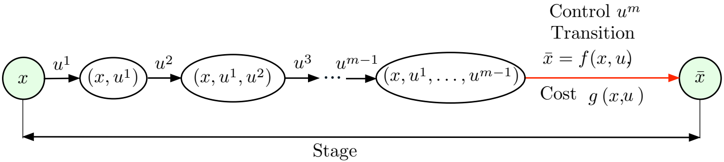
$$U_k(x_k) = U_k^1 \times U_k^2 \times \cdots \times U_k^m.$$

- Standard rollout, which considers all agents at once, involves the following minimization:

$$(\tilde{u}_k^1, \ldots, \tilde{u}_m^1) \in \arg \min_{(u_k^1, \ldots, u_k^m) \in U_k^1 \times U_k^2 \times \cdots \times U_k^m} Q_k(x_k, u_k^1, \ldots, u_k^m).$$

- Computing each Q-factor requires simulation: Infeasible even for modest m!

Multiagent Rollout - One Agent at a Time



Multiagent rollout requires much less computation

- At x_k, multiagent rollout solves sequentially m minimization problems:

$$\tilde{u}_k^1 \in \arg \min_{u_k^1 \in U_k^1} Q_k\big(x_k, u_k^1, \mu_k^2(x_k), \ldots, \mu_k^m(x_k)\big)$$

$$\tilde{u}_k^2 \in \arg \min_{u_k^2 \in U_k^2} Q_k\big(x_k, \tilde{u}_k^1, u_k^2, \mu_k^3(x_k), \ldots, \mu_k^m(x_k)\big)$$

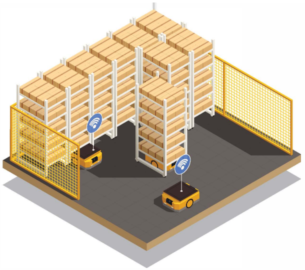
$$\vdots$$

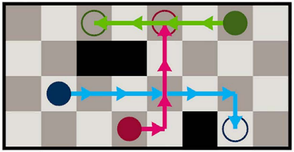
$$\tilde{u}_k^2 \in \arg \min_{u_k^m \in U_k^m} Q_k(\tilde{u}_k^1, \ldots, \tilde{u}_k^{m-1}, u_k^m)$$

- The computational demand is reduced from n^m to $n \cdot m$!
- Multiagent rollout is standard rollout applied to the reformulated problem - A legitimate Newton step!

A Fifteen-Minute Break

Catch our breath and think about issues relating to the first half of the lecture. Ask questions when you return.

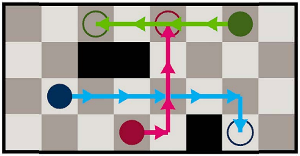




warehouse robots path planning $\Longrightarrow$ grid world representation

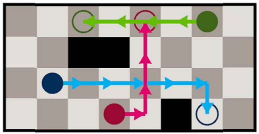
- There are $m = 3$ agents (solid circles) moving in 4 directions or standing still with perfect vision
- The agents have been assigned to some targets (open circles with the same color).
- The objective: reaching their respective targets in minimum time while avoiding collision with each other
- Simple heuristic: each agent follows the shortest path to the respective target, assuming other agents are not present (arrows in the figure)

Multiagent Path Finding Example: DP Formulation and Standard Rollout

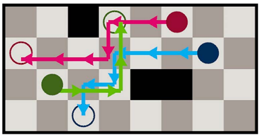


- States: current positions of all agents and their respective targets
- Control: each agent has at most 5 choices, their combination grows exponentially with m
- Stage cost: related to the number of collisions and the number of reached targets
- When applying standard rollout, we must evaluate $\approx 5^m$ approximate Q-factors

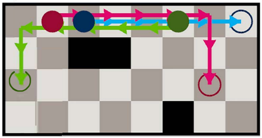
Multiagent Rollout for Multiagent Path Finding



case 1



case 2



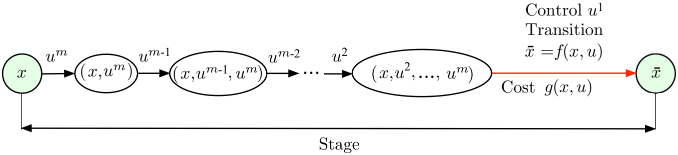
case 3

- Multiagent rollout reduces it to $5 \cdot m$ (while maintaining good properties)
- Key idea: Break down the control into a sequence of one-agent-at-a-time moves
- Each stage involves the following sequence of operations:
  - Minimizing Q-factors associated with the first agent, while the remaining two agents follow base heuristics
  - Minimizing Q-factors associated with the second agent, while the last agent follows base heuristics
  - Minimizing Q-factors associated with the last agent
- We allow a change of the order in which the agents are selecting their controls, at every stage

Multiagent Rollout for Multiagent Path Finding: Animations

Will be presented in class

Implementation Variants of Multiagent Rollout



- Reshuffling the order of agents results in a different, yet still equivalent problem
- Multiagent rollout allows parallel computation of Q-factors
- Multiple base heuristics can be applied to enhance the performance further
- All those ideas are independent of each other and can be combined
- See textbook for additional material for order optimization and other variants

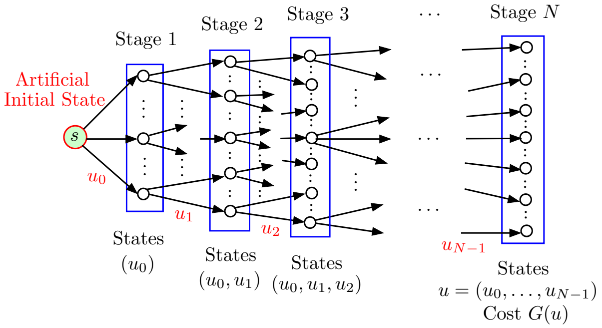
Multiagent Rollout for Multiagent Path Finding: Animation For A Large Scale Problem

Will be presented in class

- Base policy is computed offline and stored in a lookup table
- Can adapt to changing environment: some robots may breakdown halfway
- Tested up to 200 robots in simulation, required computational time less than 1 sec
- Code can be found at `https://github.com/will-em/multi-agent-rollout`
- Paper: "Multiagent Rollout with Reshuffling for Warehouse Robots Path Planning", by W. Emanuelsson et al., IFAC World Congress. Also see arXiv:2211.08201

Outline

General Discrete Optimization



Minimize $G(u)$ subject to $u \in U$

- Assume that each solution u has N components: $u_0, \dots, u_{N-1}$
- View the components as the controls of N stages
- Define $x_k = (u_0, \dots, u_{k-1})$, $k = 1, \dots, N$, and introduce artificial start state $x_0 = s$
- Define just terminal cost as $G(u)$; all other costs are 0

This formulation typically makes little sense for exact DP,
but often makes a lot of sense for approximate DP/approximation in value space

DP and Approximation in Value Space

DP solution to the discrete optimization problem

- Start with

$$J_N^*(x_N) = G(x_N) = G(u_0, \ldots, u_{N-1}) \quad \text{for all } x_N \in U$$

- For $k = 0, \ldots, N-1$, let

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} J_{k+1}^*(x_k, u_k) \quad \text{for all } x_k$$

  where $U_k(x_k)$ need to be suitably defined.

- Construct the optimal solution $(u_0^*, \ldots, u_{N-1}^*)$ by forward calculation

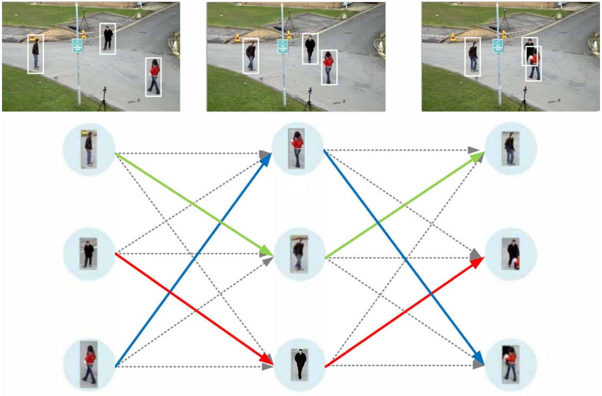
$$u_k^* \in \arg \min_{u_k \in U_k(x_k)} J_{k+1}^*(x_k, u_k) \quad \text{for all } x_k$$

Approximation in value space

- Use some $\tilde{J}_{k+1}$ in place of J_{k+1}^*
- Starting from the artificial initial state, for $k = 0, \ldots, N-1$, set

$$\tilde{u}_k \in \arg \min_{u_k \in U_k(x_k)} \tilde{J}_{k+1}(x_k, u_k) \quad \text{for all } x_k$$

Multiple Object Tracking

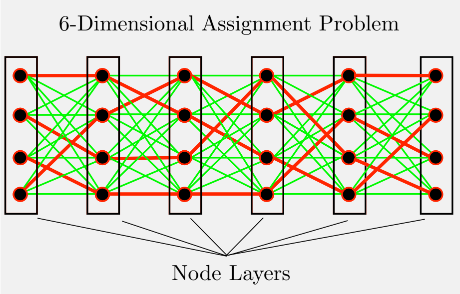


$k-1$ k $k+1$

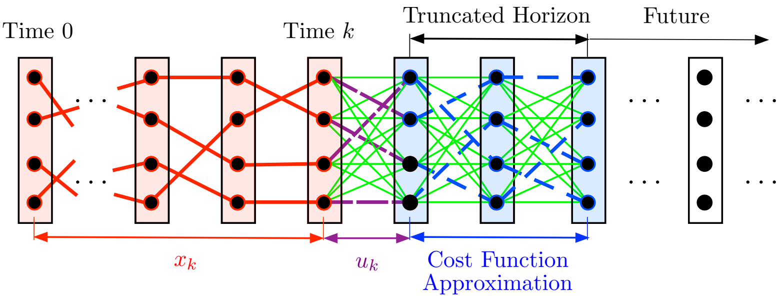
Figure source: [CGI22][1]

- Multiple object tracking (MOT) aims to match the same objects over various frames
- Nontrivial: occlusion, changes in object appearance, and real-time computation constraint
- Important problem with many applications: traffic monitoring, robotics, consumer analytics, augmented and virtual realities ...

Multidimensional Assignment Problem



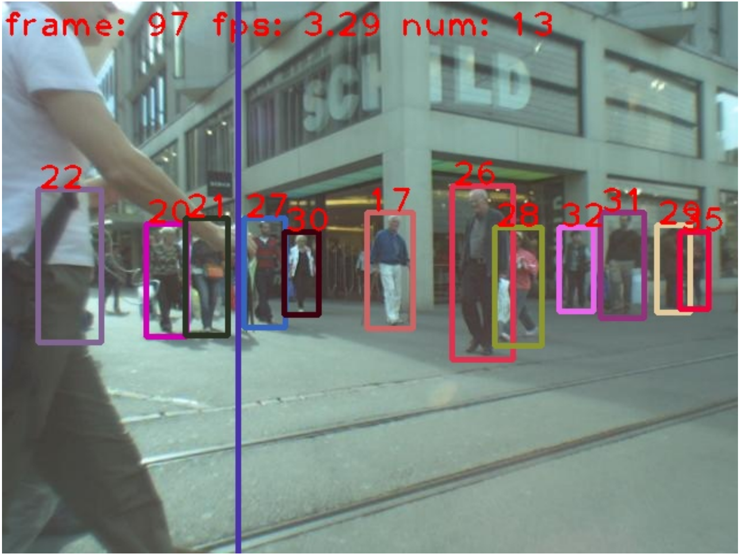
- MOT can be modeled as a multidimensional assignment problem
- There are $(N + 1)$ layers (frames) of nodes
- A grouping consists of $N + 1$ nodes $(i_0, \ldots, i_N)$ where i_k belongs to kth layer, and N corresponding arcs
- For each grouping, there is an associated cost depending on the entire grouping
- Our goal: find m groupings so that each node belongs to one and only one grouping and the sum of the costs of the groupings is minimized



- Approximation in value space involves the following key ideas:
  - One-step lookahead minimization
  - Truncated rollout
  - Cost approximation $\tilde{J}$ with structure that matches the assignment problem
- Q-factor minimization reduces to solving a 2-dimensional assignment problem
- Results: robust and consistent matching against occlusion
- Paper: "An Approximate Dynamic Programming Framework for Occlusion-Robust Multi-Object Tracking", by P. Musunuru et al., arXiv:2405.15137

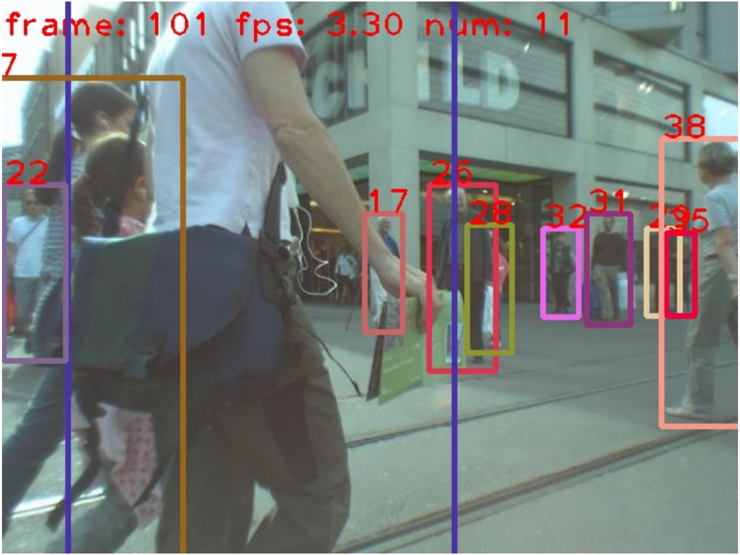
Exact Rollout for MOT: Animation

Will be presented in class


frame: 97 fps: 3.29 num: 13
22
20
21
27
30
17
26
28
32
31
29
35

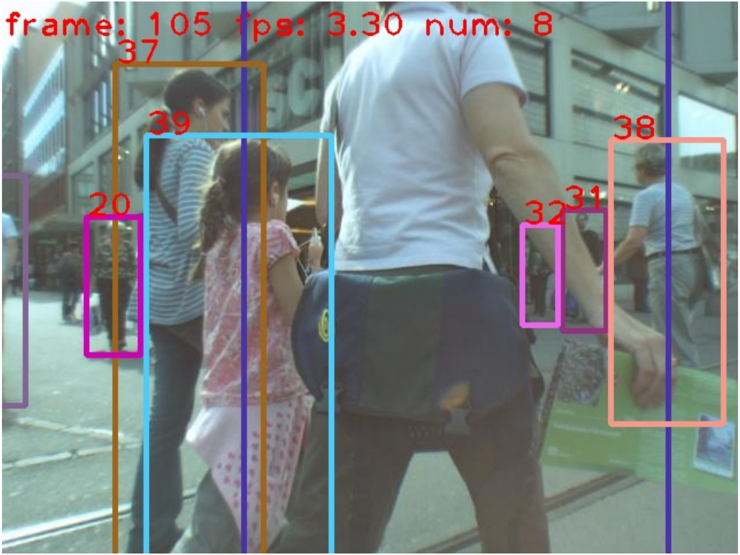
MOT Example: Base Heuristic

MOT Example: Base Heuristic




frame: 103 fps: 3.30 num: 9
37
2
17
26
32
31
29
38

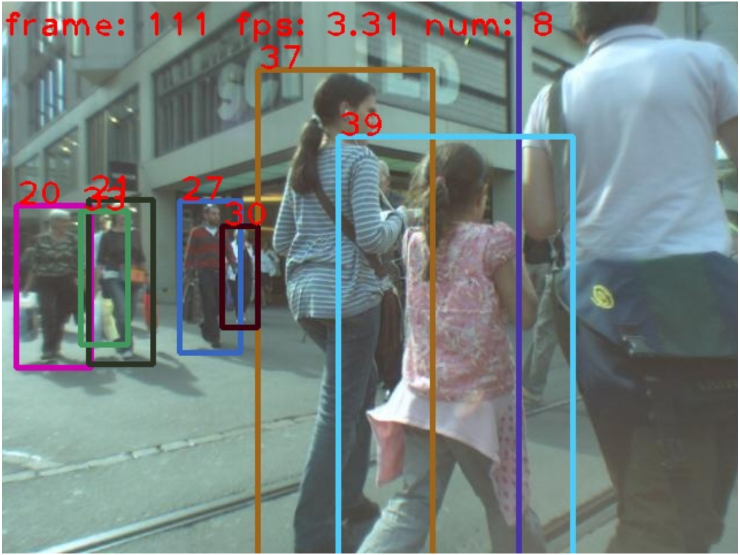
MOT Example: Base Heuristic



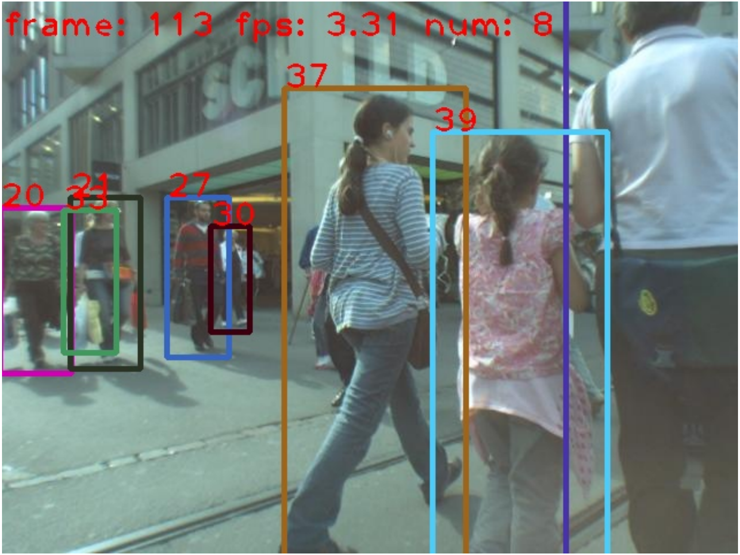
MOT Example: Base Heuristic

MOT Example: Base Heuristic

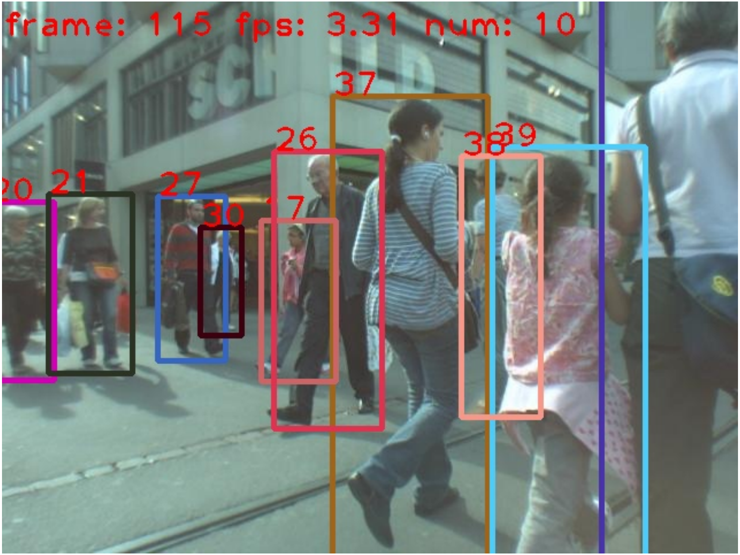
MOT Example: Base Heuristic



MOT Example: Base Heuristic

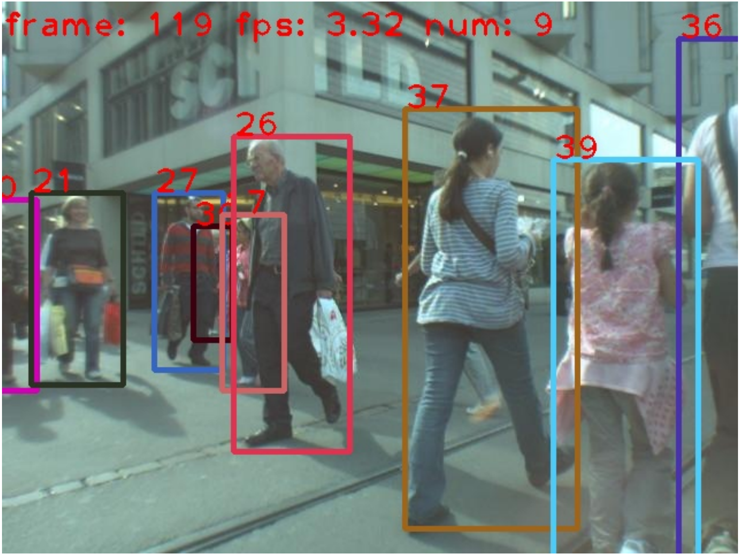


MOT Example: Base Heuristic

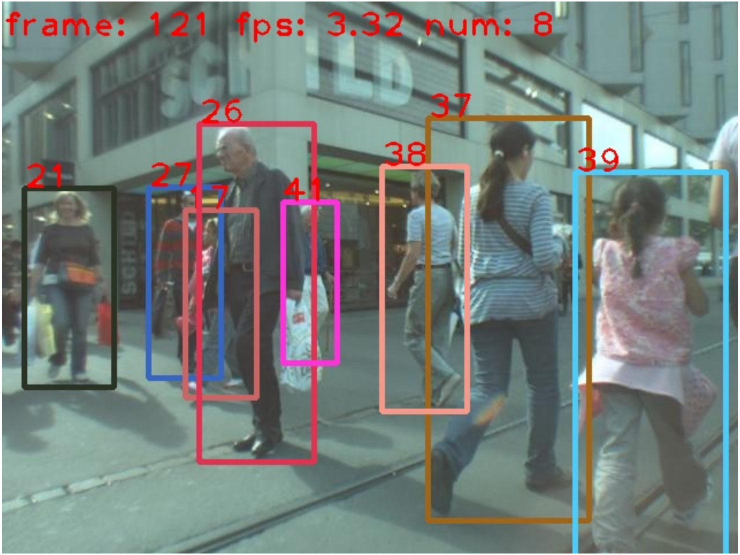


MOT Example: Base Heuristic

MOT Example: Base Heuristic



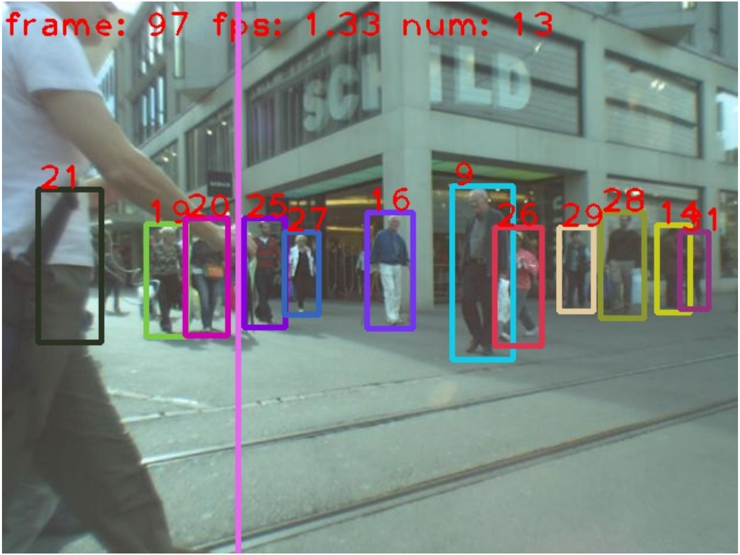
MOT Example: Base Heuristic



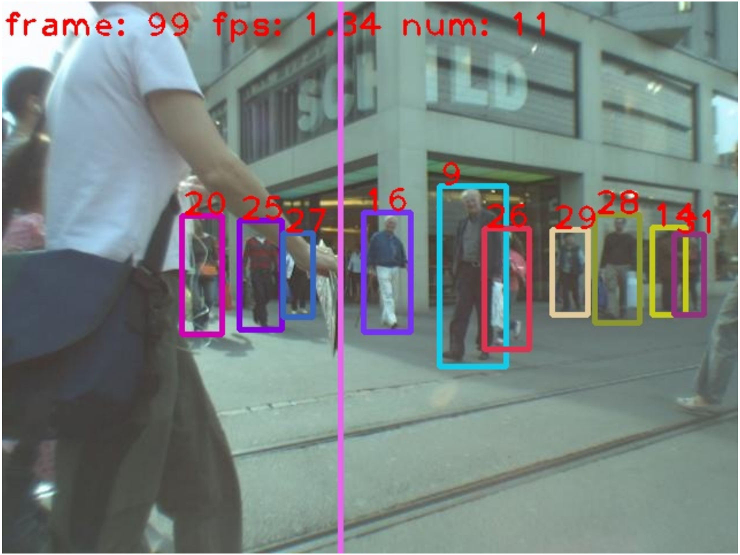
MOT Example: Base Heuristic

MOT Example: Base Heuristic

MOT Example: Approximation in Value Space



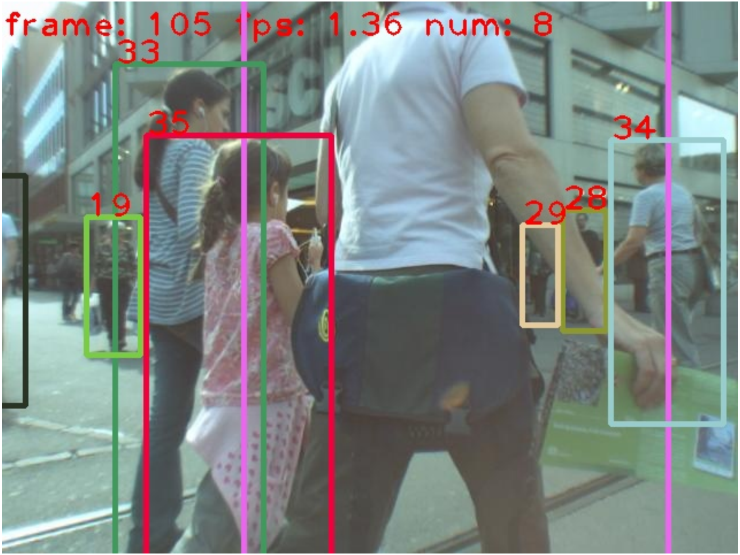
MOT Example: Approximation in Value Space



MOT Example: Approximation in Value Space

MOT Example: Approximation in Value Space

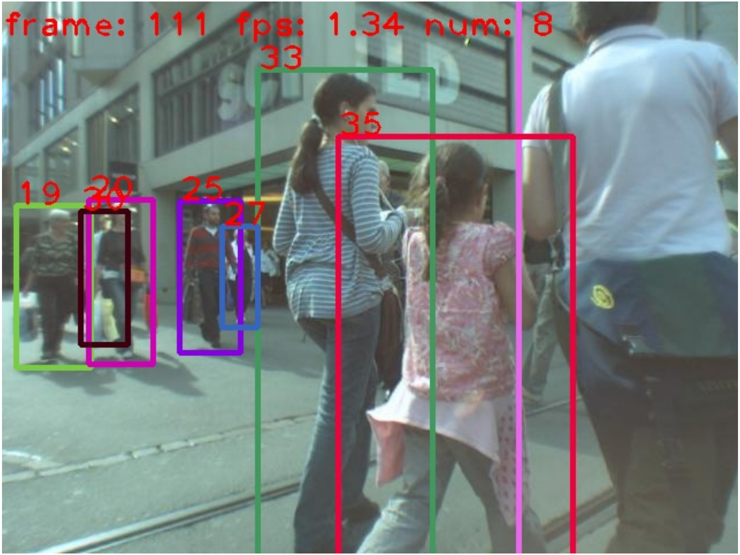
MOT Example: Approximation in Value Space



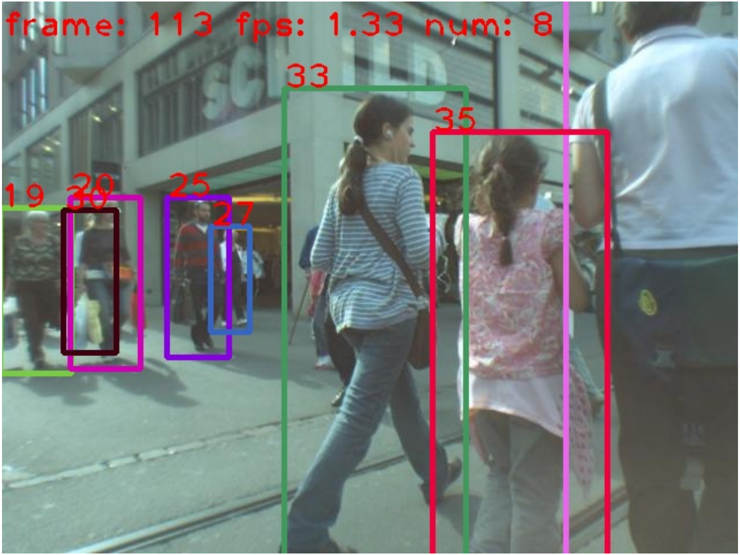

frame: 107 fps: 1.36 num: 7
33
35
34
19
20
36

MOT Example: Approximation in Value Space

MOT Example: Approximation in Value Space

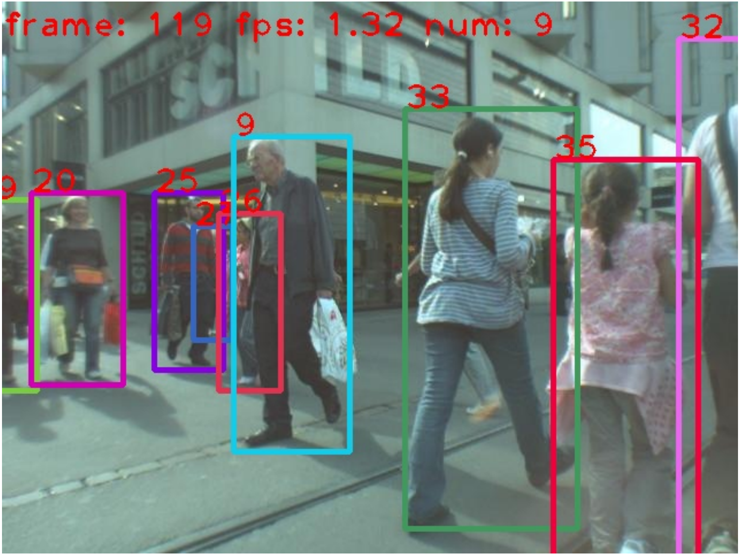


MOT Example: Approximation in Value Space

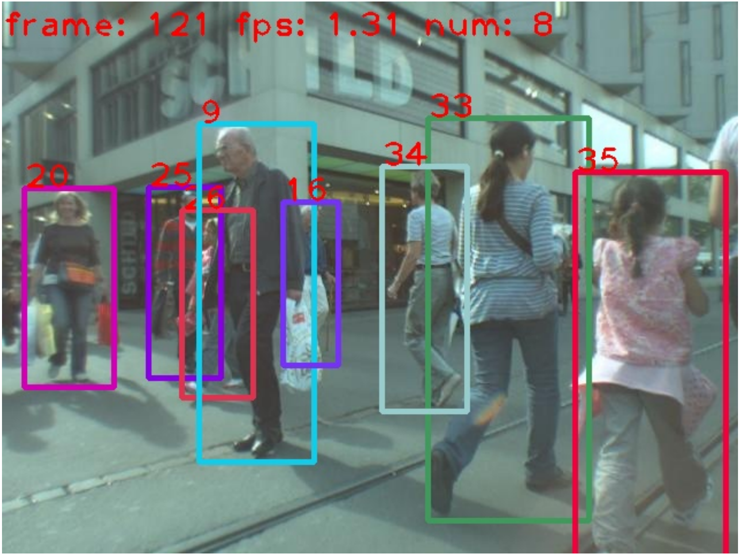


MOT Example: Approximation in Value Space

MOT Example: Approximation in Value Space


frame: 119 fps: 1.32 num: 9
32
33
9
35
20
25

MOT Example: Approximation in Value Space



MOT Example: Approximation in Value Space

MOT Example: Approximation in Value Space

About Next Lecture

In the next lecture we will cover:

- Most likely sequence generated by *n*-grams and HMM
- Rollout algorithm for approximate solution
- Application to large language models