Comparing Kinematics-Based and Learning-Based Approaches to Robotic Arm Tasking – Using Pouring as an Example

Tzu-Chieh Chen
jessica990805@gmail.com

Chung-Ta King
king@cs.nthu.edu.tw
Presenter

Department of Computer Science
National Tsing Hua University, Taiwan

Prof. Chung-Ta King

- 1988: Ph.D., Michigan State University, USA
- 1988-1990: Assistant Professor, NJIT, USA
- 1990-1997: Associate Professor, NTHU, Taiwan
- 1997-: Professor, NTHU
- 2009-2012: Chair, Dept of CS, NTHU
- 2014-2015: Associate Dean, College of EECS, NTHU
- 2019-: Vice President and Chief of Staff, NTHU
- Research interests: embedded and mobile computing, computer architecture, parallel and distributed systems
- Conference organizer: Cluster 2016, ICPADS 2014, CloudCom 2012, ESWeek 2011, …

Acknowledgements

- The work reported in this paper is part of a 4-year project (2019-2022) funded by the Minister of Science and Technology, Taiwan, under the Moon-Shoot Project

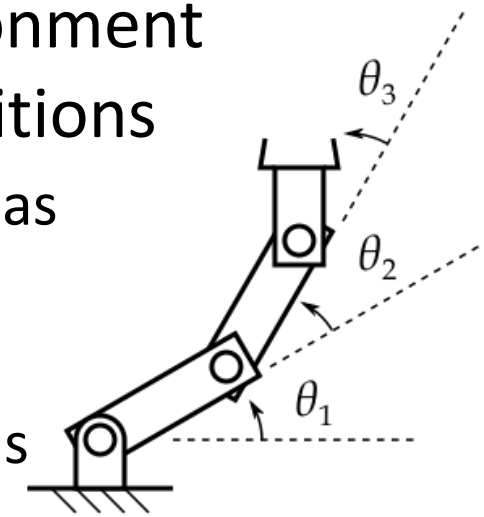
- Team members:
  - Professors Cheng-Wen Wu, Jen-Yuan Chang, Jing-Jia Liou, Chih-Tsun Huang, and Hung-Kuo Chu of National Tsing Hua University, Taiwan
  - Over 20 research staffs and graduate students

Outline

- Introduction
- Problem Statement
- Approaches to Be Compared
  - Kinematics-Based Approach
  - Learning-Based End-to-End Approach
- Experiments for Comparison
  - Environment
  - Data Collection
  - Evaluation
- Conclusions and Future Works

Introduction

- Traditional industrial robots normally work in a fixed environment performing repetitive tasks on fixed objects with know positions
  - Their operations can be pre-programmed, using techniques such as *inverse kinematics* (IK) for trajectory calculation
  - IK determines the joint parameters to move the end effectors of a robot to desired positions by calculating kinematics equations



- As robots move into the service domain, they face dynamic environments, requiring extra perception and planning capabilities
  - May involve scene recognition + arm trajectory planning + gripper operations

(https://en.wikibooks.org/wiki/File:Planar-three-link-manipulator.svg)

[1] "Inverse kinematics - Wikipedia" [Online]

Introduction

- Take pouring task by a robotic arm as an example
  - Identify objects and determine their coordinates
  - Plan arm trajectories and move arm to the designated position
  - Perform gripper pouring operations

- Straightforward extension of industrial robotic arms
  - Object recognition and coordinate determination:
    - Image-based: RGB-D camera, image processing and pattern recognition (perhaps by deep neural networks)
  - Trajectory planning
    - Inverse kinematics

Kinematic-based approach

Introduction

- Alternative: end-to-end learning-based
  - Take raw sensory inputs and generate the arm control outputs directly
  - Learn to map sensory inputs to control outputs

Learning-based approach

- Why end-to-end?
  - Can use a single neural network to perform the mapping
  - Possible to design a well performed model without deep knowledge of the problem, particularly suitable for complex problems too difficult to develop rules to solve

Introduction

- End-to-end is not without problems
  - May be difficult to improve or modify, e.g., applying structural changes → need to re-train all over again
  - May not know why the model does not work well
  - May be limited by the training data collected
  - May require more computations and memory space

- Which approach is better?
  - Kinematic-based versus learning-based end-to-end

Outline

- Introduction
- Problem Statement
- Approaches to Be Compared
  - Kinematics-Based Approach
  - Learning-Based End-to-End Approach
- Experiments for Comparison
  - Environment
  - Data Collection
  - Evaluation
- Conclusions and Future Works

Problem Statement

- Compare learning-based end-to-end and kinematics-based approaches to robot tasking, using pouring task as an example
  - *Kinematics-based approach*: object detection with deep neural network (Yolo) + depth from depth camera + trajectory planning with inverse kinematics (IK)
  - *Learning-based end-to-end approach*: deep neural network with RGBD images as input and joint parameters as output
    - Training data are collected using kinematics-based approach to ensure data consistency
  - Assume simple scenes that have no obstacles

Problem Statement

- Comparisons based on two different scenarios:
  - *Static*: the cups are at the same locations throughout the task
  - *Dynamic*: the cups will be moved during the task
- Metrics to compare:
  - Time usage, memory usage, actions under different scenarios

- Highlights of comparisons:
  - In static scenario, both approaches finished all the tasks
  - In dynamic scenario, kinematics-based approach could not finish some tasks while end-to-end learning-based approach completed all

Contributions

- We developed kinematics-based and learning-based end-to-end approaches to pouring task by a robotic arm
- To train the end-to-end model, we developed a data collection system based on the kinematics-based method
- We evaluate and compare these two approaches, analyzing their performance under static and dynamic scenarios

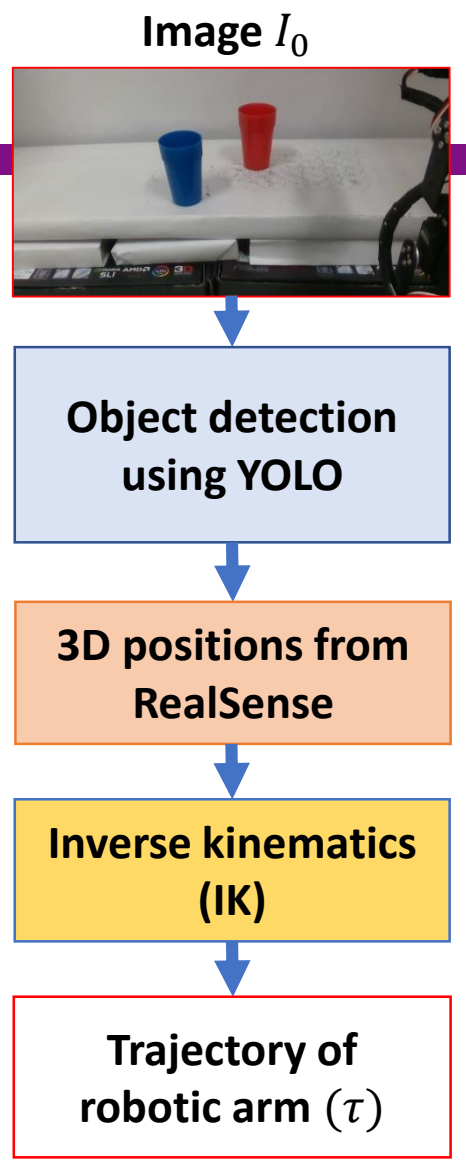
As far as we know, no previous works on systematic comparisons

Outline

- Introduction
- Problem Statement
- Approaches to Be Compared
  - Kinematics-Based Approach
  - Learning-Based End-to-End Approach
- Experiments for Comparison
  - Environment
  - Data Collection
  - Evaluation
- Conclusions and Future Works

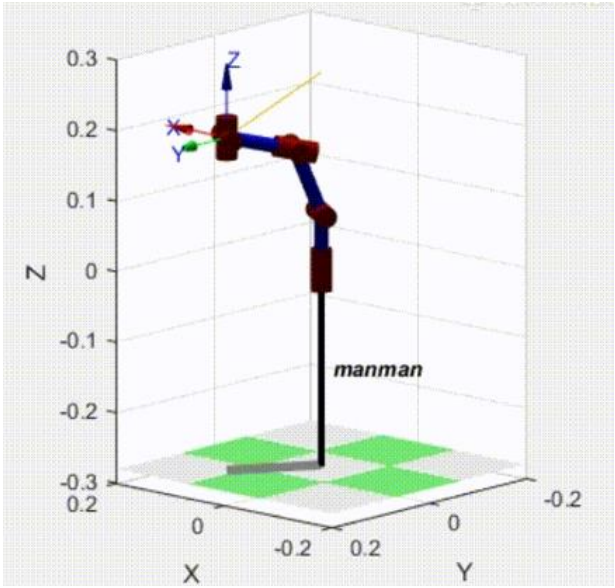
Kinematics-based Approach

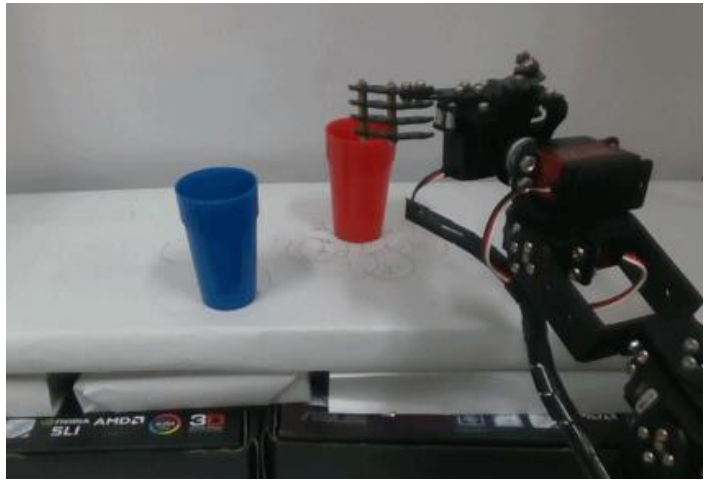
- Object detection + trajectory planning
  - Input image I_0: the first image seen by the robot camera
  - Output τ= $(A_0, A_1, .. A_t)$: trajectory of robot arm, where $A_0, A_1, .. A_t$ are joint angles of robotic arm at each time step
- Object detection module:
  - Input: image I_0
  - Output $c = (c_x, c_y)$: 2D coordinate of center of cups in I_0
  - Use YOLO with rules to obtain c
  - From c, get 3D coordinates from a RGBD camera with origin at camera's lens and then obtain the center coordinate $T = (T_x, T_y, T_z)$ with origin at robotic arm



Kinematics-based Approach

- Trajectory planning module:
  - Input: $T = (T_x, T_y, T_z)$
  - Output: $\tau = (\tau_0, \tau_1, .. \tau_t)$
  - Use IK to calculate joint parameters at each time step t to construct a path to reach T
  - τ is then used to control the robot to complete the task

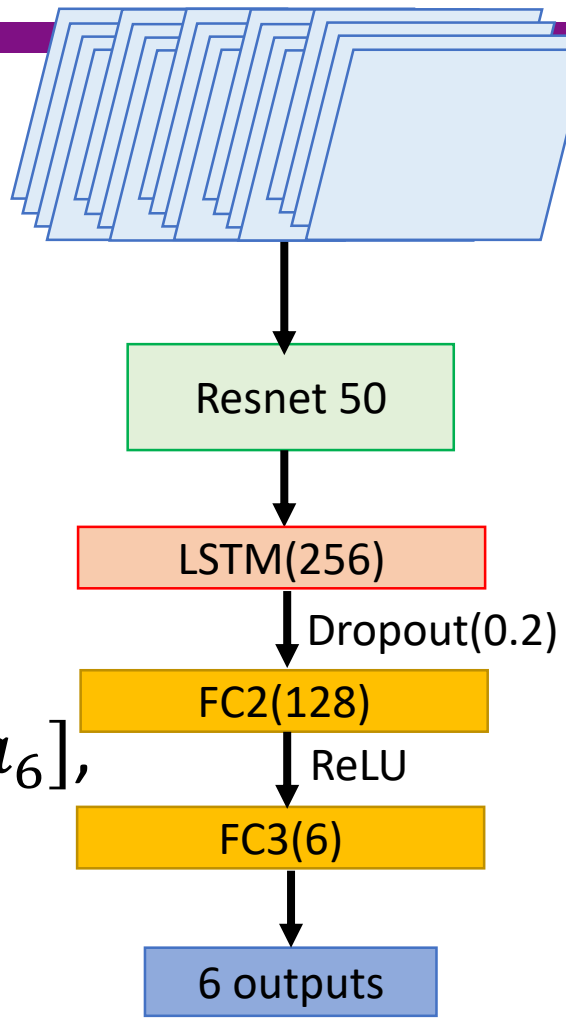




- Static scenario: both modules performed once
- Dynamic scenario: both modules performed once per unit time

Learning-based Approach

- Input at time t: $S_t = [\mathrm{I}_{t-4}, \mathrm{I}_{t-3}, \mathrm{I}_{t-2}, \mathrm{I}_{t-1}, \mathrm{I}_t]$
  - I_t: the RGBD image at time step t
- Output at time t: A_t
- ResNet-50 to extract features from S_t
  - Last fully connected layers replaced by LSTM with a dropout
    - LSTM to deal with time sequence, e.g., when to close the gripper
- One fully connected layer: ReLU as activation function
- Last fully connected layer: predicts action $A_t = [a_1, \ldots a_6]$, to set the six joint motors
- Optimizer (Adam), loss function (mean square error)



Outline

- Introduction
- Problem Statement
- Approaches to Be Compared
  - Kinematics-Based Approach
  - Learning-Based End-to-End Approach
- Experiments for Comparison
  - Environment
  - Data Collection
  - Evaluation
- Conclusions and Future Works

Experimental Environment

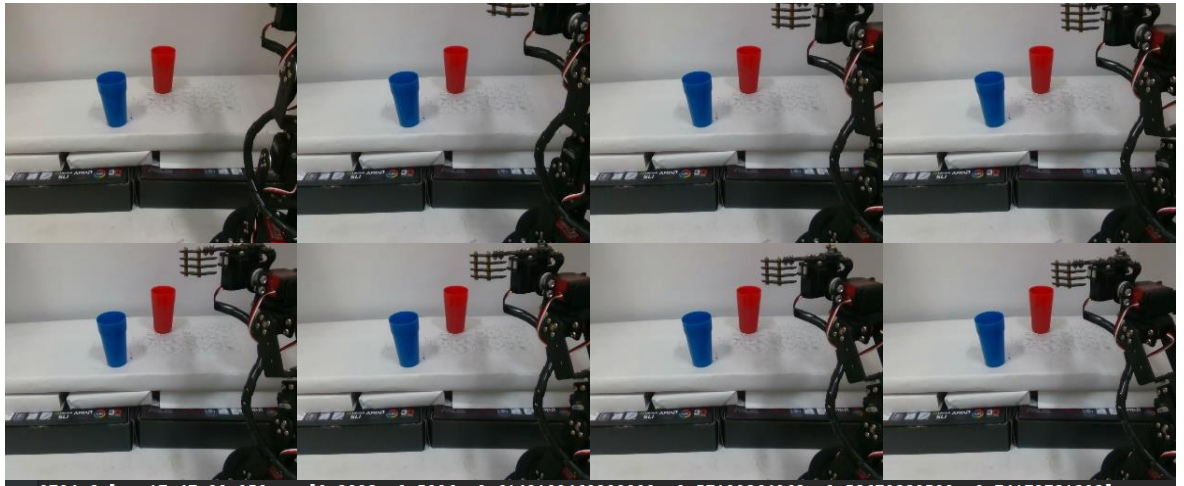
- 6DOF robot arm
- Intel RealSense Depth Camera D415
  - Behind at the robot arm's left side with a depression angle around 30 degrees
- Task: pour the contents of the red cup into the blue cup

The view from the camera

Training Data Collection

- Training data for learning-based approach
  - Collect states and actions during each demonstration by controlling the robotic arm using the kinematics-based approach for data consistency

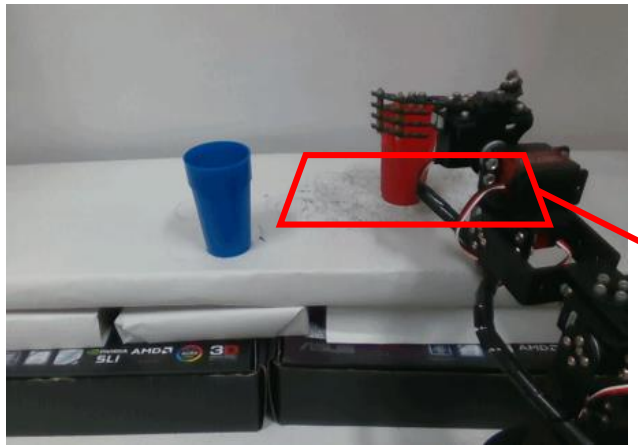
- Collected dataset
  - 40 pouring demonstrations, each containing 64 RGBD images and 64 corresponding actions of the six joint motors of the robotic arm
  - 2560 RGBD images and 2560 actions in total



```
0704_Color_17_47_29_250.png[0.3998, 0.5996, 0.6142102462399999, 0.57129864968, 0.58670889592, 0.74173781808]
0704_Color_17_47_31_158.png[0.3998, 0.5996, 0.70066348868, 0.43240609903999994, 0.53426958772, 0.74173781808]
0704_Color_17_47_40_105.png[0.3998, 0.5996, 0.7207414705999999, 0.40582940455999994, 0.52777087516, 0.74173781808]
0704_Color_17_47_42_416.png[0.3998, 0.5996, 0.7418309378, 0.37997899639999994, 0.5230099342000001, 0.74173781808]
0704_Color_17_47_44_104.png[0.3998, 0.5996, 0.76397615708, 0.35506467344, 0.52024083052, 0.74173781808]
0704_Color_17_47_46_188.png[0.3998, 0.5996, 0.7872143534, 0.33135041587999997, 0.51976476928, 0.74173781808]
0704_Color_17_47_48_101.png[0.3998, 0.5996, 0.81156248312, 0.30916849556, 0.52193097868, 0.74173781808]
0704_Color_17_47_49_210.png[0.3998, 0.5996, 0.83651594912, 0.28928974088, 0.5270056900000001, 0.74173781808]
```

Scenarios for Comparison

- Static: cups fixed at the same locations throughout the task
  - Placed the red cup in 30 different positions in the area that the robotic arm can reach, with the positions spread evenly across the workspace
- Dynamic: red cup will be moved while the task is in progress



26	21	16	11	6	1
27	22	17	12	7	2
28	23	18	13	8	3
29	24	19	14	9	4
30	25	20	15	10	5

Evaluation: Static Scenario

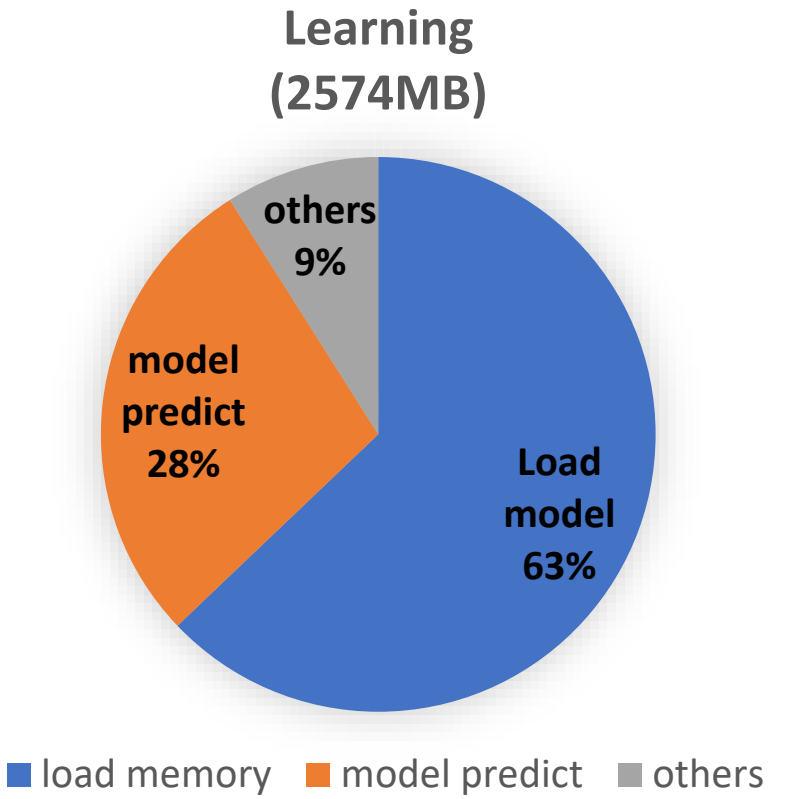
- Both approaches can complete all the tasks

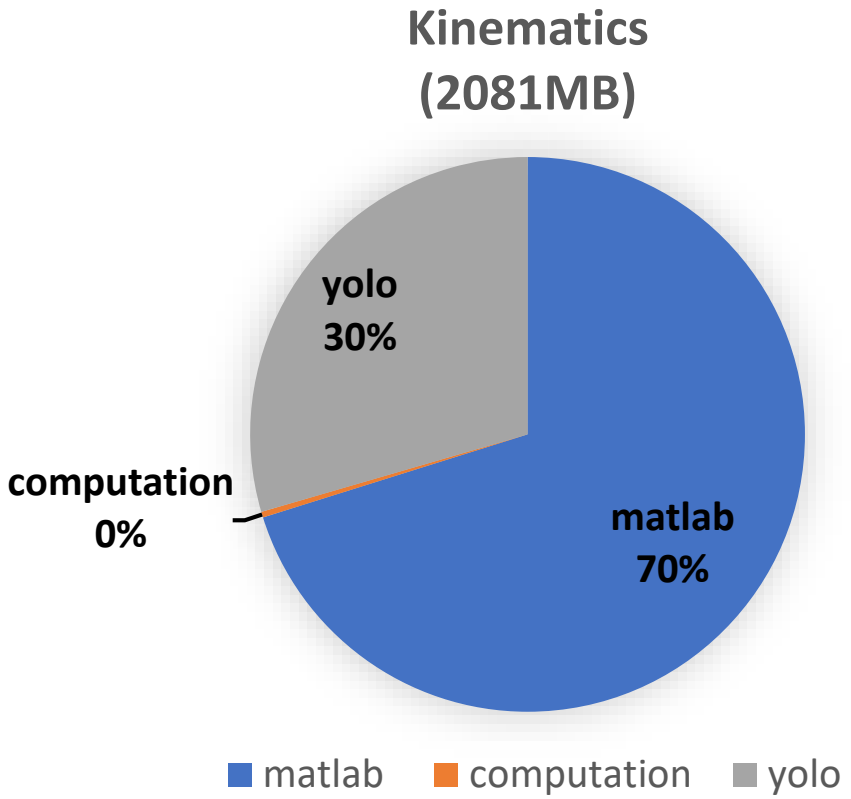
	Average computation time(CPU time)	Average number of moving steps	Average total time	Memory usage	Size
Kinematics	7.83 s	X	119.4 s	2081MB (614 loading model)	236MB
Learning	10.27 s	48	84.1 s	2574MB (1618 loading model)	278MB

  - Computation time: CPU time running the model, not counting communication
    - Kinematics-based : object detection and trajectory planning
    - Learning-based: prediction time
  - Number of moving steps
    - Kinematics-based: fixed at 64
    - Learning-based: trained with 64 steps, but can complete in less than 64 steps

Evaluation: Static Scenario

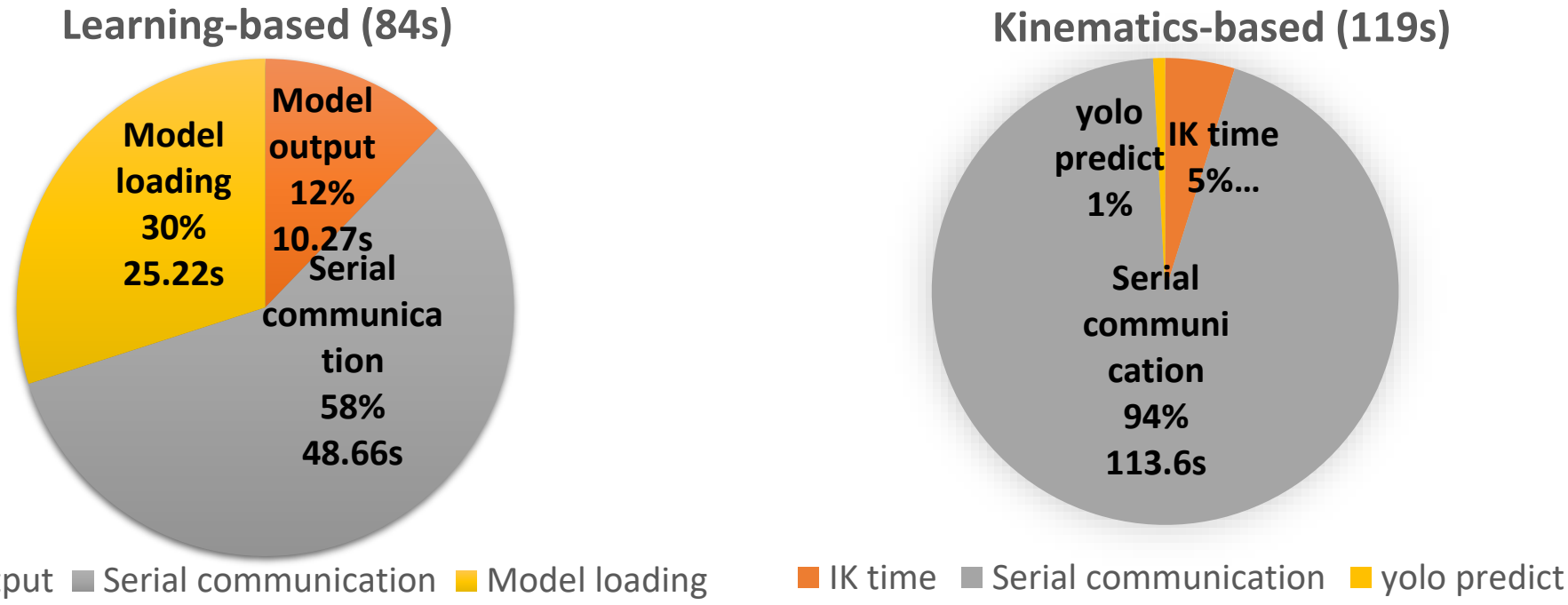
- Memory usage





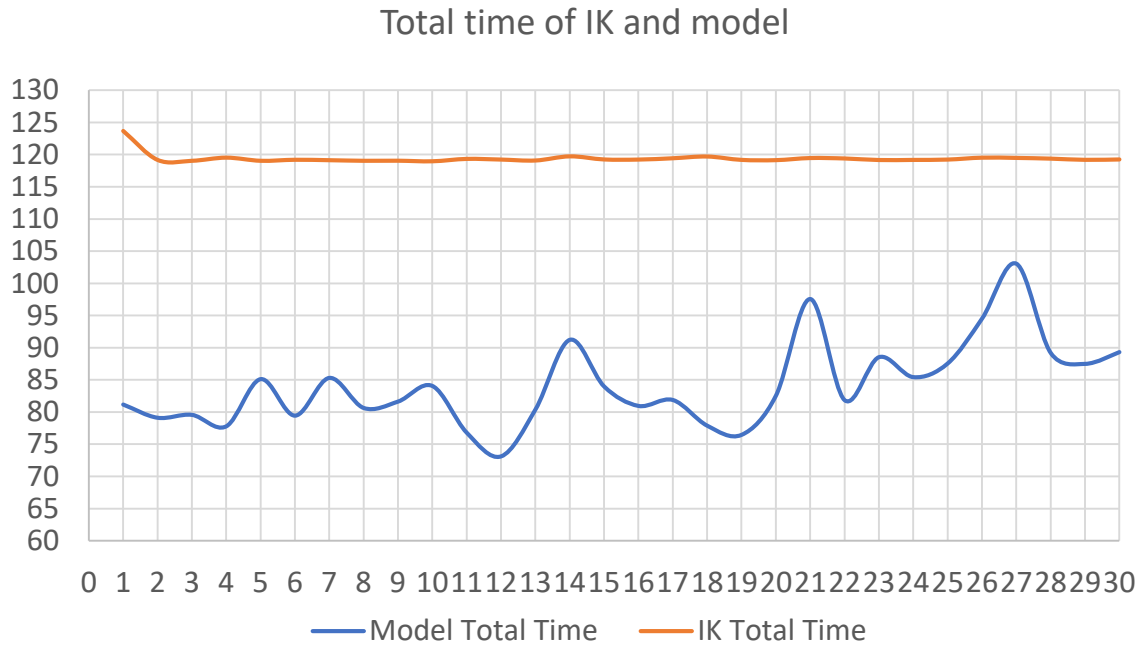
Evaluation: Static Scenario

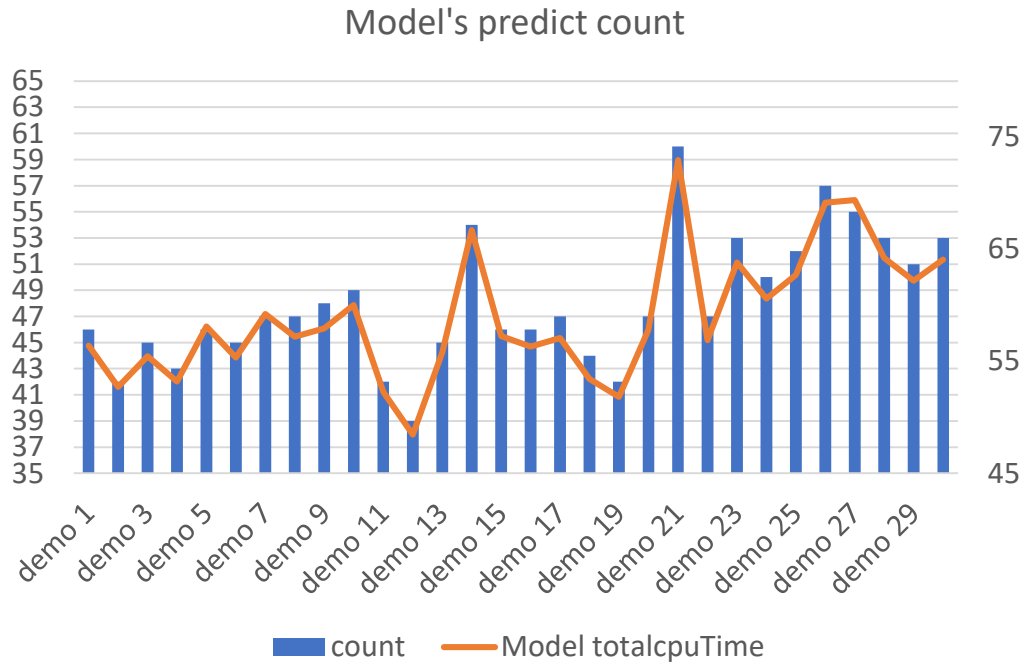
- Total computation time
  - Learning-based: 10.27 sec → output an action in 0.21 sec
  - Both approaches spent a lot of time on serial communication



Evaluation: Static Scenario

- Total time and # of predictions vs total time

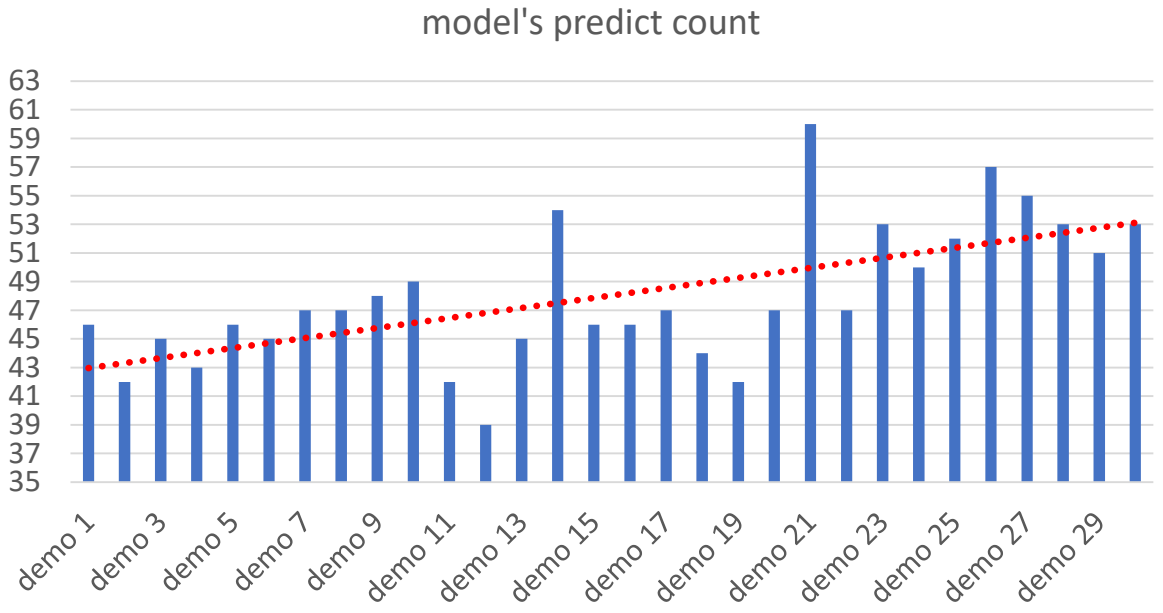




  - Kinematics-based: total time remains the same because # of steps is fixed
  - Learning-based: total time depends on number of predictions during the task

Evaluation: Static Scenario

- Up trend of moving steps in learning-based approach
  - Since all the tasks are trained with 64 steps, if gripper is very close to the target position and we still move the arm with 64 steps, the gap between every states and actions will be too close → waster time in completing task



26	21	16	11	6	1
27	22	17	12	7	2
28	23	18	13	8	3
29	24	19	14	9	4
30	25	20	15	10	5

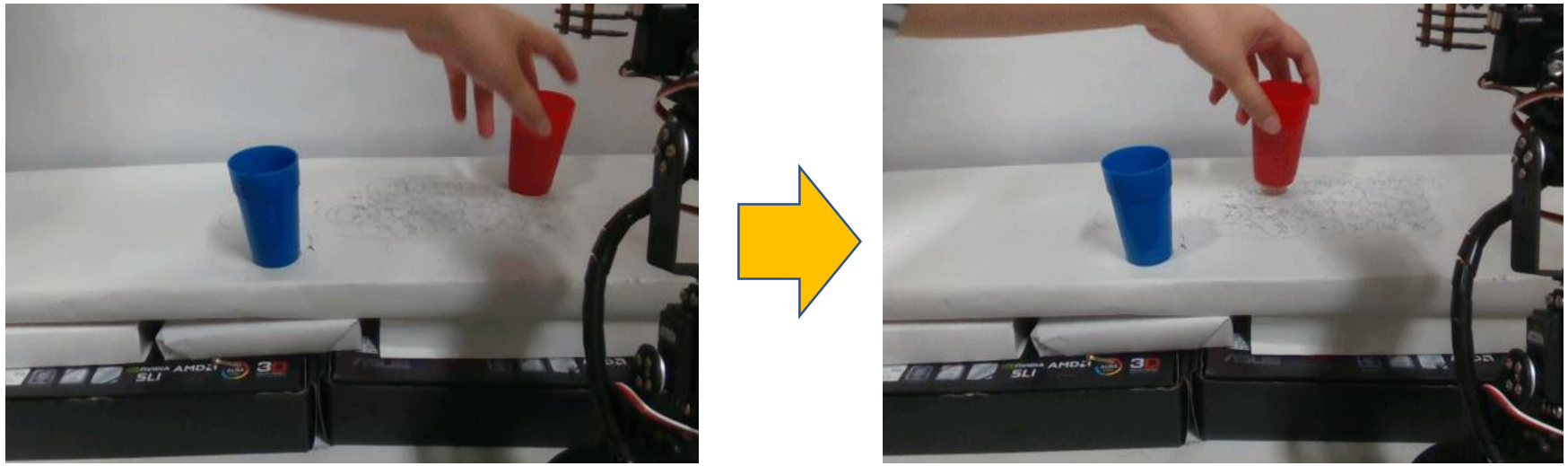
Evaluation: Dynamic Scenario

- To allow kinematics-based approach to handle dynamic scenario, extra rules are added:

1: **While** the distance between the robotic arm and the target d > threshold h **do**
2: Object detection
3: **If** the target's migration distance m > 1.5mm **then**
4: Trajectory replanning
5: Divide the trajectory into n points (n is based on d)
6: Move the robotic arm
7: Move the robotic arm and grab the target

Evaluation: Dynamic Scenario

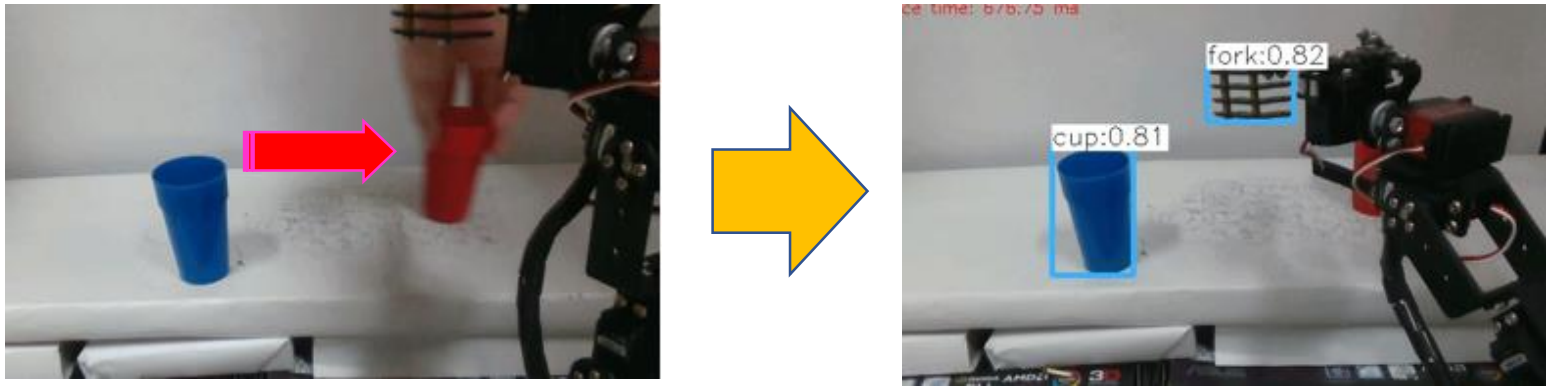
- The test can be split into 4 parts, moving the target cup by 4 different ways in the workspace of the robotic arm:
  - Right-to-left



  - Left-to-right
  - Top-to-bottom
  - Bottom-to-top

Evaluation: Dynamic Scenario

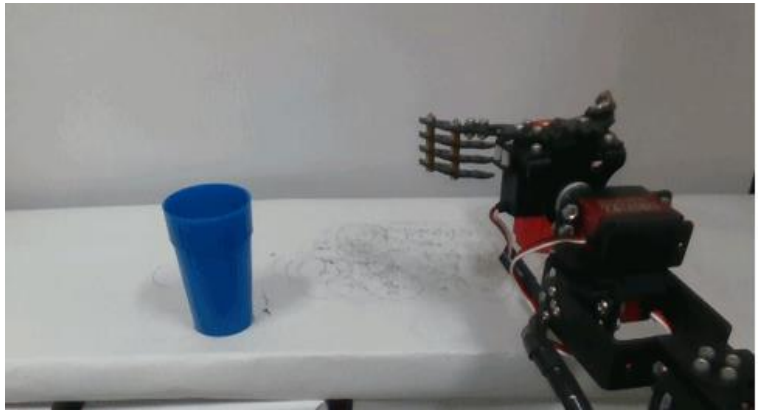
- **Right-to-left:** both approaches can complete right-to-left tasks
- **Left-to-right:** the kinematics-based approach failed in upper-left and middle-left task, because the arm cut off the camera's view of the cup



- **Top-to-bottom & Bottom-to-top :** Both approaches can complete all the tasks

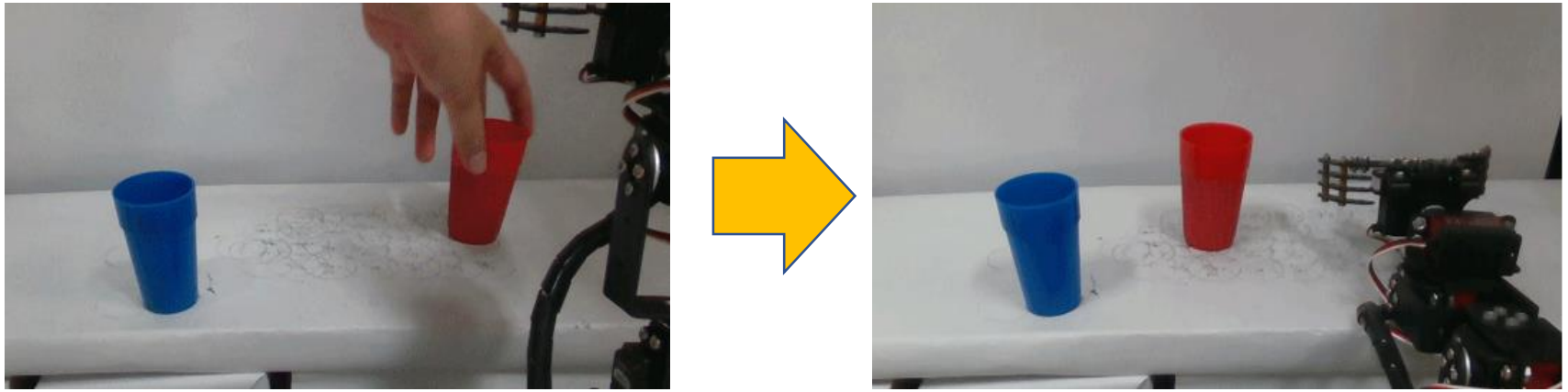
Evaluation: Dynamic Scenario

- Learning-based approach succeeded in all the tasks
  - Robust: when the robotic arm faces a state it has not seen before, it will do the action corresponding to similar states
  - In-time: it can still complete the task even when the target cup is moved just before the robotic arm grabs it



Evaluation: Dynamic Scenario

- Kinematics-based approach succeeds if object detection module can recognize the target object correctly
  - The timing that the object detection module perceives the current state
  - The target's final position
  - Even with the help of extra rules, there are always unexpected states
  - Can only handle the change of the position of the cup early in the process



Evaluation: Dynamic Scenario

	Average computation time(cpu time)	Average number of moving steps	Average total time	Memory usage	Size
Kinematics	21.53 s	66	137.21 s	2279MB (630 loading model)	236MB
Learning	9.33 s	47	81.67 s	2572MB (1615 loading model)	278MB

- Main difference between static and dynamic scenario is computation time:
  - Kinematics-based method spends additional time on object detection and trajectory replaning when the target object is moved
- By observing the number of moving steps, learning-based approach finishes the task efficiently in dynamic scenario too

Outline

- Introduction
- Problem Statement
- Approaches to Be Compared
  - Kinematics-Based Approach
  - Learning-Based End-to-End Approach
- Experiments for Comparison
  - Environment
  - Data Collection
  - Evaluation
- Conclusions and Future Works

Conclusions

- We proposed a kinematics-based approach and an learning-based end-to-end approach for robot pouring task
- The end-to-end learning network was trained by the dataset collected based on our kinematics-based approach
- We compared the two approached in static scenario and dynamic scenario by evaluating their time and memory usage
- Our experimental results show that kinematics-based approach is suitable for static scenario, because it has less time usage and memory usage, and learning-based approach is more suitable for complicated and dynamic scenarios

Future Works

- Use different object detection methods to improve the kinematics-based approach, e.g., grasp detection model
- Develop an automated data collection system for collecting data easier
- Perform comparisons using more complicated scenarios, e.g., complicated background, blue cup in different positions
- Extend to more complicated tasks, e.g., obstacle avoidance or semantic control