Vessel-to-Vessel Motion Compensation with Reinforcement Learning

Sverre Herland, Kerstin Bach

Norwegian University of Science and Technology
sverre.herland@ntnu.no, kerstin.bach@ntnu.no

Abstract

Actuation delay poses a challenge for robotic arms and cranes. This is especially the case in dynamic environments where the robot arm or the objects it is trying to manipulate are moved by exogenous forces. In this paper, we consider the task of using a robotic arm to compensate for relative motion between two vessels at sea. We construct a hybrid controller that combines an Inverse Kinematic (IK) solver with a Reinforcement Learning (RL) agent that issues small corrections to the IK input. The solution is empirically evaluated in a simulated environment under several sea states and actuation delays. We observe that more intense waves and larger actuation delays have an adverse effect on the IK controller's ability to compensate for vessel motion. The RL agent is shown to be effective at mitigating large parts of these errors, both in the average case and in the worst case. Its modest requirement for sensory information, combined with the inherent safety in only making small adjustments, also makes it a promising approach for real-world deployment.

Introduction

Automation is needed in the offshore industry to keep up with demand. Shipboard robotic arms and cranes are likely to be an integral part of autonomous marine operations in the future. Potential applications include vessel-to-vessel load handling (Tørdal and Hovland 2017), routine service on remote aquaculture locations (Bjelland et al. 2015), and automooring for autonomous ships (Jha et al. 2020).

A challenge related to offshore robots is the highly dynamic environment they operate in. In contrast to their land-fixed counterparts, shipboard manipulators are mounted to the deck of a floating vessel, meaning that they act in non-inertial coordinate frames (Cao and Li 2020). In settings where the manipulation target is on another vessel or structure, the arm also has to compensate for the relative motion between the two, leading to a complex control problem.

The difficulties of dynamic environments are further exacerbated by actuation delays. Real-world robots have a latency between the time an action is sent to the controller and the time it is reflected in the robot's configuration (Andersen et al. 2015). This poses a problem for arms and cranes that compensate for motion since a configuration that is correct at one point in time might be incorrect a moment later. In the in-lab experiments presented by (Brandt et al. 2022), actuation delays are identified as the dominating source of error when compensating for vessel motion.

One way to deal with actuation delays is to plan ahead and schedule commands so that they will manifest at the right time in the future. This, of course, requires knowledge of the future and a model that can be used for planning. Assuming perfect knowledge of future vessel motion, (From et al. 2009) solves the problem of optimal motion planning manipulator arms mounted to the deck of a ship. A later work relaxes this assumption by additionally constructing a predictive model of the vessel motion (From et al. 2011).

An alternative to classical planning techniques is Reinforcement Learning, which has seen many applications in robotics and manipulation over the last decades (Kaelbling, Littman, and Moore 1996; Kober, Bagnell, and Peters 2013; Nguyen and La 2019). An advantage of RL, in particular model-free RL which treats the control problem as a black box, is that it can be applied end-to-end. Additionally, RL-based controllers typically benefit from fast response times, as the optimization cost of planning is amortized during the training phase. However, the field still has open problems that need to be resolved if it is to see large-scale application for real-world robotics, such as sample efficiency and safe exploration while training.

A promising sub-field of RL that addresses some of these issues is residual policy learning (RPL) (Silver et al. 2018; Johannink et al. 2019; Zhang et al. 2022). The key idea of RPL is to train an RL agent to make small adjustments to an otherwise conventional controller. RPL is well-suited for complex control problems where good but sub-optimal controllers are readily available. Since most of the work is handled by the conventional controller, these methods are often less data hungry. Additionally, it typically restricts the agency of the RL agent, which in turn can limit its hazard potential.

In this work, we use model-free RL to solve the problem of vessel-to-vessel motion compensation while subject to actuation delays. Inspired by residual policy learning, we propose a hybrid solution where an RL agent learns to produce small corrections to the input of an Inverse Kinematics (IK) solver. The approach is evaluated in simulations of various sea states and actuation delays. We show that the corrections

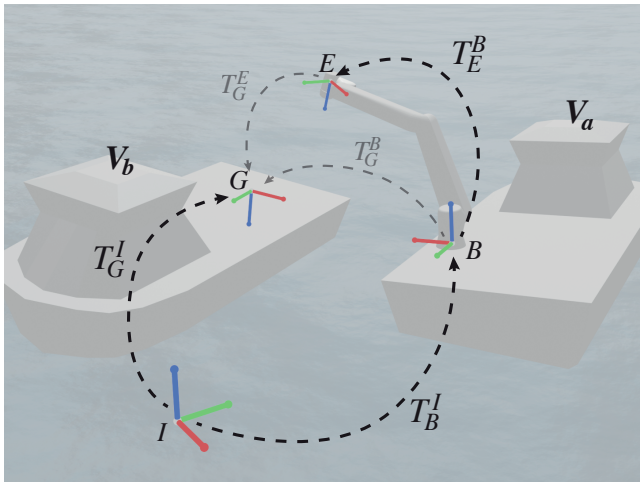

Figure 1: Overview of the most relevant coordinate frames and the transforms between them. I is the inertial frame, B is the base of the robot (on V_a), E is the TCP of the robotic arm, and G is a reference pose on the deck of V_b . A perfectly controlled arm would ensure that frames E and G coincide at all times.

predicted by the RL agent reliably reduces the tracking errors incurred by a pure IK controller, and discuss the path to real-world application.

The remainder of the paper is structured as follows. Section 2 formally defines the considered vessel-to-vessel motion compensation problem. Section 3 frames the problem as a Markov Decision Process (MDP) and describes the proposed RL solution. Section 4 documents the numerical simulation and presents the results. Section 5 discusses the findings and Section 6 concludes the paper.

Problem Description

We consider a situation where a manipulator arm is mounted on top of a floating vessel (V_a). The objective of the arm is to keep the Tool Center Point (TCP) in a fixed pose relative to the local coordinate system of another floating vessel (V_b), thus compensating for the relative motion between the two.

Overview and Notation

An overview of the problem setup is visualized with relevant coordinate frames and transforms in Figure 1. We use the notation $T_Y^X = \left(\begin{smallmatrix} R_Y^X & \mathbf{p}_Y^X \\ 0 & 1 \end{smallmatrix}\right) \in \mathbb{R}^{4\times4}$, composed of a rotation matrix $R_Y^X \in \mathbb{R}^{3\times3}$ and a translation vector $\mathbf{p}_Y^X \in \mathbb{R}^3$, to denote a rigid transformation between coordinate frames X and Y. I.e., a (homogeneous) point $p_Y = [x, y, z, 1]$ in coordinate frame Y is related to a point $p_X = [\hat{x}, \hat{y}, \hat{z}, 1]$ in coordinate frame X through the equation $p_X = T_Y^X p_Y$. Moreover, transformations can be chained together so that $T_Y^X T_Z^Y = T_Z^X$ and are always invertible so that $(T_Y^X)^{-1} = T_X^Y$.

The transforms T_B^I , T_E^B , T_G^I are not static over time. T_B^I and T_G^I move with vessels V_a and V_b respectively. Meanwhile, T_E^B is given by the joint configuration q of the robotic arm through forward kinematics. Control of the vessels is considered beyond the scope of this paper. Instead, we focus on motion compensation through optimization of $T_E^B\ (q)$.

Objective

The objective of the arm is to maintain a static reference pose within the local coordinate frame of V_b . That is, the goal is to ensure that $T_E^I = T_G^I$, or equivalently $T_E^B = T_G^B$, implying that $T_G^E = T_B^E\, T_G^B$ should be the identity transform. Deviations from the reference pose is broken down into a positional (Δp , length of the shortest correcting translation) and orientational ($\Delta\alpha$, angle of the smallest correcting rotation) error, both calculated from T_G^E .

$$\text{Position Error} = \Delta p = \left\| \mathbf{p}_G^E \right\|_2 \tag{1a}$$

$$\text{Orientation Error} = \Delta\alpha = \arccos\left(\frac{\text{tr}\left(R_G^E\right) - 1}{2}\right) \tag{1b}$$

Base Controller

It is assumed that the robotic arm already has a controller that is capable of moving the end effector to any pose specified within the dexterous workspace of the robot. We use an Inverse Kinematic solver (IK) plus Proportional Derivative (PD) controller, hereby referred to as the *IK controller*.

The IK controller takes as input a desired transform T^*, relative to the base of the robot, and solves for a joint configuration q^* that causes the end effector to satisfy $T_E^B\ (q^*) = T^*$. Because the objective is to ensure that frames E and G coincide, we set $T^* = T_G^B$ as the baseline reference pose for the IK solver. The IK solver solves for suitable joint configuration $q_{t_i}^*$ at discrete time steps t_i. The result is forwarded to the PD controller which uses it to generate motor signals for the joint actuators.

Actuation delays

In this study, we investigate the adverse effects actuation delays have on robot arms in dynamic environments, and to what degree we can mitigate them with RL. As mentioned in the introduction, all robotic arms have a minimal actuation delay, caused by factors such as low-pass filtering of input and physical limitations. However, we also look at the impact of larger delays. Towards this end, we consider two filter functions that impose additional delays by transforming q_{t_i} into $\hat{q}_{t_i}$.

$$\mathcal{D}_{E(k)} : \hat{q}_{t_i} = q_{t_{i-k}} \tag{2a}$$

$$\mathcal{D}_{B(N,f_c)} : \hat{q}_{t_i} = \sum_{j=0}^{N} b_j^{(N,f_c)} q_{t_{i-j}} - \sum_{j=1}^{N} a_j^{(N,f_c)} \hat{q}_{t_{i-j}} \tag{2b}$$

The first function ($\mathcal{D}_{E(k)}$) creates an explicit delay by buffering the raw IK output for k timesteps, effectively imposing an artificial latency of length $t_i - t_{i-k}$. The second

function ($\mathcal{D}_{B(N,f_c)}$) models a more realistic delay by imposing a low-pass filter over the values of q. It implements an online Nth order Butterworth filter with cutoff frequency f_c, which produces coefficients a and b. Similar to other low-pass filters, $\mathcal{D}_{B(N,f_c)}$ imposes an implicit delay on the input signal in addition to the smoothing (Manal and Rose 2007).

Methods

The proposed solution is a hybrid system where an RL agent learns to predict adjustments for the IK controller's input. That is, instead of solving for robot configurations that satisfies $T_E^B\ (q) = T_G^B$, the IK controller solves for a modified pose $T_H^B\ = T_G^B\ T_H^G$, where T_H^G is a small correction predicted by the RL controller.

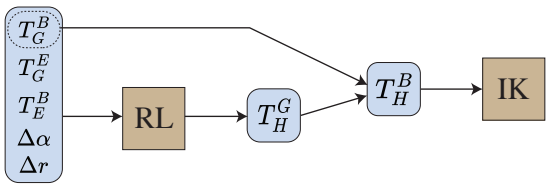

Figure 2: Proposed system. The RL policy predicts a small correction that is used to create an adjusted IK target.

MDP Formulation

The RL problem is modeled as a Markov Decision Process (MDP) (Sutton and Barto 2018). It consists of a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ is the reward function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function, and $\gamma \in (0, 1)$ is a discount factor. The transition function represents the environment, which maps a state and the action selected by the agent to a new state at discrete time steps. Note that from the RL agent's point of view, the IK controller is part of the environment. The reward function is a scalar function that summarizes performance and is subject to maximization. The objective (J) of the agent is to find a parameterized policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$ that maps states to actions in a way that maximizes the expected sum of future discounted reward from any given state.

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{i=0}^{\infty} \gamma^t r_{t_i}\right] \tag{3}$$

The state observations given to the agent is a relatively simple set of values based on the transforms presented in Figure 1. At each time step, the agent is presented with the current measurements of T_G^B , T_E^B , and T_G^E . Each transform is represented as a flattened $\mathbb{R}^{3\times3}$ rotation matrix and an $\mathbb{R}^3$ translation vector. We also include scalars with the current distance (Δp) and angular ($\Delta\alpha$) error norms, meaning that the complete observation is an $\mathbb{R}^{38}$ vector.

The only sensor data needed to obtain these observations are live tracking of vessel V_b relative to vessel V_a and forward kinematics through the robotic arm. To fully describe the environment, one might also want information such as the robot's joint configuration and time derivatives. In this sense, the problem could be considered partially observable. However, in the interest of simplicity, we do not make any attempts at explicitly measuring or modeling belief over these variables. Instead, we rely on a recurrent policy architecture as explained later.

The action space of the RL policy consists of the set of rigid transforms T_H^G , which we break up into a translational ($\mathbf{p}_H^G$), and an orientational (R_H^G) component. The translational component is directly predicted by the policy as an $\mathbb{R}^3$ vector. The orientational component is represented as a rotation vector where the direction gives the axis of rotation and the magnitude gives the angle. That is, the network predicts another $\mathbb{R}^3$ vector which is then converted into a rotation matrix. This representation has a known problem with discontinuities at $\pm\pi$ rotations (Zhou et al. 2019). However, we do not consider that an issue here since the policy should only issue small corrective rotations.

The reward chosen directly reflects the objectives presented in Equation 1. We sum the positional error Δp and the orientational error $\Delta\alpha$, and negate the result (Equation 4). This results in a dense reward that provides the agent with feedback at every time step. It is possible to weight the terms to trade off the two objectives, but we found a simple sum to be sufficient since the errors (meters, radians) tended to have roughly the same magnitude.

$$\text{reward} = -(\Delta p\ + \Delta\alpha\ ) \tag{4}$$

Neural Network Architecture

The RL Agent consists of two recurrent neural networks, a policy π_θ that is used to sample actions and a value function v_ϕ that predicts the state value, i.e. Equation 3 conditioned on the current state. Both networks have a similar architecture (see Figure 3), but do not share any parameters. Normalized observations are initially fed through a two-layer Multilayer Perceptron (MLP) encoder with tanh activation function. The RNN module takes the MLP output and feeds it though a single LSTM layer (Hochreiter and Schmidhuber 1997). The output of the RNN is added back to the output of the MLP with a skip connection which we found to help with stability during training. Lastly, the sum is forwarded to the head, which is implemented differently for the two networks.

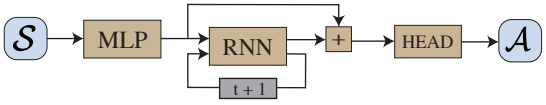

Figure 3: Network architecture for the policy π_θ. Except for $\mathcal{S}$ and $\mathcal{A}$ all arrows represent $\mathbb{R}^{128}$ vectors. The value function has equivalent architecture except for the head.

The head of the policy network performs a linear mapping from the model dimension and down to the number of actions ($\mathbb{R}^d \to \mathbb{R}^{3+3}$). The result is used as a mean, along with trainable standard deviations, to parameterize a Gaussian distribution ($\pi_\theta\ (a|o)\ = \mathcal{N}(\mu_\theta(o), \sigma_\theta)$). Meanwhile,

the head of the value network linearly maps its input down to a single scalar ($\mathbb{R}^d \rightarrow \mathbb{R}$) that represents the value estimate.

Optimization

The policy network is trained on-policy using the surrogate clipped likelihood ratio objective from PPO (Schulman et al. 2017) and Generalized Advantage Estimation (GAE) (Schulman et al. 2015). Additionally, an entropy regularizer is applied to the Gaussian distributions at the network's head to encourage exploratory behavior. The value network is trained to predict the realized discounted Monte Carlo return through a Huber loss (Huber 1964). Both networks are optimized with the Adam optimizer (Kingma and Ba 2014).

The networks are trained in epochs that consist of data collection and on-policy training. Each epoch starts by rolling out the current policy and collecting a batch of trajectories. Predictions from the current value function are then combined with the empirical rewards to calculate advantage estimates with GAE. The policy and value functions are then trained for a fixed maximum number of gradient steps, with early stopping for the policy if the action distribution changes too much (as given by an estimate of KL divergence).

Results

The proposed method is numerically evaluated in simulation. We describe the simulator, show results for just the IK controller, and finally, investigate how the RL agent improves on it. Supplementary video clips can be found on YouTube[1] and the code is shared on Github[2].

Setup

We build a simulation environment on top of the Deep Mind Control Suite (DMCS) (Tunyasuvunakool et al. 2020) and the MuJoCo physics engine (Todorov, Erez, and Tassa 2012). The physics model consists of the two vessels and a manipulator arm as shown in Figure 1. We use a realistic model of the UR10e[3] arm, a 1.3-meter long arm that is suitable for future in-lab experiments. The IK controller runs at 50Hz and uses a numerical solver implementation of (Wampler 1986), provided by the DMCS framework. The PD controller is integrated with MuJoCo and runs, along with the physics simulation time step, at 500 Hz.

The vessels are constrained to follow trajectories given by data pre-generated by a high-fidelity simulator for marine operations (SIMO[4]). The trajectories are generated by simulating a large (25.5 meters long) aquaculture service vessel under various sea conditions, equivalently to the data used in (Brandt et al. 2022). The different sea states $\mathcal{W}$ are modeled with the Jonswap Spectrum (Hasselmann et al. 1973), varying the significant wave height H_s and typical wave period T_p (see Table 1).

[1]https://youtu.be/wj1YKecxQ7I

[2]https://github.com/sherilan/exposed-mcx

[3]https://www.universal-robots.com/products/ur10-robot/

[4]https://www.sima.sintef.no/simo/index.html

Sea State	$\mathcal{W}_a$	$\mathcal{W}_b$	$\mathcal{W}_c$	$\mathcal{W}_d$
H_s [m]	1.0	1.5	2.0	2.5
T_p [s]	6.0	8.0	9.0	10.0

Table 1: Jonswap spectrum parameters for the four (increasingly extreme) sea states used in the experiments.

Similarly to (Brandt et al. 2022), we scale down the vessel trajectories to match the size of the arm in a manner that preserves linear accelerations. This entails that if the trajectory positions are scaled down by a factor λ the sample rate of the trajectory is sped up by a factor of $\sqrt{\lambda}$. However, we use $\lambda = 10$ instead of the $\lambda = 4.84$ used in (Brandt et al. 2022), as it allows a theoretical full-scale version of the arm to comfortably reach up to 10 meters, an ideal range according to the referenced work. Note that this makes the problem slightly harder, as it leads to a relative 44% vessel motion speedup.

For each sea state we generate a dataset of approximately one hour of simulated vessel motion. The first two-thirds of each set is used for training and the last third is reserved for testing. Since the data is a time series with the position and orientation of a single vessel, two random temporal windows are sliced out for each episode to provide motion trajectories for vessels V_a and V_b . This is not entirely realistic, as the motion of two vessels in the same waves will typically be correlated up to a phase shift. However, we argue that it is still a reasonable way to evaluate the proposed method, since breaking this correlation should only make it harder for the RL policy to predict relative motion. In addition, we randomize the target frames G at the start of each episode by sampling a random pose uniformly from a 20x20x20 cm working area on vessel V_b .

We run five different configurations for the artificial delays, two explicit delays $\mathcal{D}_{E(k)}$, two Butterworth-based low-pass delays $\mathcal{D}_{B(N,f_c\ )}$, and one where there is no added delay. The two explicit delays use $k = 1$ and $k = 2$, leading to a delay of 20 and 40 milliseconds for the 50 Hz controller. The two low-pass delays use fourth-order Butterworth coefficients ($N = 4$) with differing cut-off frequencies f_c . The cut-off frequencies f_c are set according to the formulas derived by (Manal and Rose 2007) so that the resulting delay will approximately match their explicit counterparts. The no-added-delay configuration is implemented as an explicit delay with $k = 0$.

Type	$\mathcal{D}_{E(0)}$	$\mathcal{D}_{B(4,21)}$	$\mathcal{D}_{E(1)}$	$\mathcal{D}_{B(4,4)}$	$\mathcal{D}_{E(2)}$
Delay	0 ms	$\sim$ 20 ms	20 ms	$\sim$ 40 ms	40 ms

Table 2: Artificial delays considered in the experiments.

Baseline

We begin by testing just the IK controller (no RL adjustments) and observe how it performs under different sea states and actuation delays. This is implemented by swapping out the RL module for a dummy agent that returns the

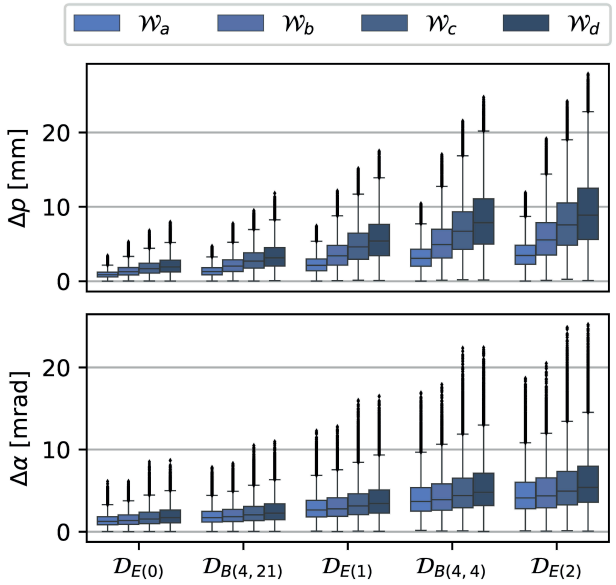

Figure 4: Distribution of positional (Δp) and orientational ($\Delta\alpha$) errors for the baseline (IK + PD) controller under various sea states and actuation delays.

identity transform for T_H^G . For each considered sea state $\mathcal{W}$ and actuation delay $\mathcal{D}$, we sample 50 test trajectories, each 1000 samples long (20 simulated seconds). We look at the positional and orientational error at each time step, except for the first 50 steps (1 sec) to give the controller and simulator time to stabilize.

Figure 4 visualizes the distribution of errors for the pure IK controller under various sea states and actuation delays. Consistently with (Brandt et al. 2022), we observe that more intense sea states result in larger tracking errors. Moreover, we see that larger waves' impact on tracking errors is intensified as we increase the delay. The errors are generally a bit lower for the Butterworth-based delays than their explicit counterparts, despite being calibrated to have the same theoretical delay. We conjecture that this could be because the smooth output of the filter is more suited as input for the underlying PD controller.

RL Corrections

Next, we train our RL agent and compare its performance to the pure IK controller. Each training episode consists of 1000 steps and sample data from one of the four wave datasets (chosen randomly). We train for a total of 2500 epochs, each consisting of 8 episode trajectories, resulting in a total of 20 million environment interactions ($\sim$ 111 hours).

An independent RL agent is trained for each of the 5 delay configurations, the motivation for this being that we assume the delay to be a constant property of the arm. The procedure is repeated for 5 different random seeds, while keeping all other hyperparameters equal, resulting in a total of 25 agents. In general, we found the algorithm to converge robustly for all considered delays and random seeds, as long as we were careful not to train on a handful of episodes where the randomized waves put the target pose outside of the arm's dexterous workspace.

Each trained RL agent is evaluated under the same conditions as the baseline presented above. We also make sure to match the random seed used in the environment, meaning that they are subjected to the exact same vessel motions and sampled positions for G . The results are summarized in Table 3.

From the results, it is clear that the RL-based corrections have a positive impact on the arm's ability to compensate for vessel motion. The first column group in Table 3 shows the average errors of the proposed method along with their standard deviations. We see a similar trend as in the baseline experiments (Figure 4), but with smaller magnitudes. Positional errors lies in a range starting at 0.39 millimeter ($\mathcal{D}_{E(2)}$, $\mathcal{W}_a$) and ending at 1.81 ($\mathcal{D}_{E(2)}$, $\mathcal{W}_d$). The orientational errors follow a similar pattern, starting at 0.5 milliradians ($\mathcal{D}_{B(4,21)}$, $\mathcal{W}_a$) and maxing out at 1.8 ($\mathcal{D}_{E(2)}$, $\mathcal{W}_d$). The second column group shows the relative improvement over the baseline. Even for the least challenging wave configuration ($\mathcal{D}_{E(0)}$, $\mathcal{W}_a$), the RL method reduce average error by 43%. From there, longer delays appear to provide the RL agent with more opportunities to add value. A very encouraging result is that of $\mathcal{D}_{B(4,4)}$, where we observe an almost 90% reduction in average Δp for the three most challenging sea states.

Good average motion compensation is of limited utility if the edge cases are bad, as that is where accidents are likely to happen. Therefore, we investigate whether we have been effective at reducing the worst-case errors. The third and fourth column group in Table 3 calculates the same ratio as the second, except only across the datapoints with the top 5 and 1 percent errors. We observe that the RL-agent still robustly provides utility and that the improvements are not restricted to just the average case, particularly for the positional error. The last column group shows the same ratio, except calculated with only the maximum error. The improvements for Δp still hold up fairly well here, but $\Delta\alpha$ a bit less so. For $\mathcal{D}_{E(0)}$ and $\mathcal{D}_{B(4,21)}$ in sea state $\mathcal{W}_d$, the largest measurement of $\Delta\alpha$ was about 20% higher for the RL agent than for the pure IK controller.

Discussion and Future Work

The results paint a promising picture of what is possible to achieve with RL for vessel-to-vessel motion compensation. Even though the problem considered here was restricted to keeping the TCP stationary, a stable platform localized to vessel V_b can be useful in several ways. One can, for instance, imagine a situation where an attached gripper must be held still while closing its grasp around an object. Nevertheless, future work may want to extend the problem to following trajectories localized to the deck of vessel V_b .

In this work, we considered a small-scale arm simulation, in part because we had an accurate model of the UR10e arm, but also because it sets the stage for future in-lab experiments similar to that of (Brandt et al. 2022). More research is needed before the method is ready for real-world deployment.

		Error ($\mu \pm \sigma$)		Imp (all)		Imp (top 5%)		Imp (top 1%)		Imp (max)	
		Δp [mm]	$\Delta\alpha$ [mrad]	Δp	$\Delta\alpha$	Δp	$\Delta\alpha$	Δp	$\Delta\alpha$	Δp	$\Delta\alpha$
$\mathcal{D}_{E(0)}$	$\mathcal{W}_a$	0.39 ± 0.19	0.50 ± 0.25	0.58	0.65	0.59	0.67	0.60	0.64	0.54	0.50
	$\mathcal{W}_b$	0.42 ± 0.22	0.57 ± 0.32	0.70	0.64	0.71	0.62	0.71	0.57	0.63	0.22
	$\mathcal{W}_c$	0.51 ± 0.28	0.70 ± 0.43	0.72	0.61	0.70	0.58	0.69	0.55	0.55	0.33
	$\mathcal{W}_d$	0.61 ± 0.34	0.81 ± 0.52	0.72	0.59	0.70	0.53	0.67	0.47	0.36	-0.18
$\mathcal{D}_{B(4,21)}$	$\mathcal{W}_a$	0.39 ± 0.19	0.51 ± 0.26	0.72	0.74	0.72	0.74	0.72	0.73	0.52	0.57
	$\mathcal{W}_b$	0.44 ± 0.23	0.59 ± 0.35	0.80	0.72	0.79	0.69	0.77	0.65	0.64	0.35
	$\mathcal{W}_c$	0.58 ± 0.34	0.75 ± 0.48	0.80	0.69	0.76	0.65	0.72	0.62	0.50	0.38
	$\mathcal{W}_d$	0.72 ± 0.45	0.88 ± 0.58	0.79	0.66	0.74	0.61	0.69	0.56	0.52	-0.21
$\mathcal{D}_{E(1)}$	$\mathcal{W}_a$	0.62 ± 0.30	0.71 ± 0.36	0.73	0.76	0.73	0.77	0.73	0.76	0.60	0.67
	$\mathcal{W}_b$	0.70 ± 0.35	0.81 ± 0.45	0.81	0.75	0.81	0.74	0.79	0.71	0.67	0.58
	$\mathcal{W}_c$	0.91 ± 0.51	1.03 ± 0.62	0.81	0.71	0.78	0.69	0.76	0.67	0.57	0.42
	$\mathcal{W}_d$	1.12 ± 0.66	1.22 ± 0.77	0.80	0.69	0.77	0.65	0.73	0.60	0.55	0.13
$\mathcal{D}_{B(4,4)}$	$\mathcal{W}_a$	0.51 ± 0.26	0.52 ± 0.29	0.84	0.87	0.84	0.87	0.83	0.86	0.75	0.79
	$\mathcal{W}_b$	0.58 ± 0.30	0.62 ± 0.38	0.89	0.86	0.88	0.84	0.87	0.82	0.78	0.69
	$\mathcal{W}_c$	0.77 ± 0.43	0.81 ± 0.53	0.89	0.84	0.87	0.81	0.85	0.79	0.75	0.66
	$\mathcal{W}_d$	0.97 ± 0.57	0.97 ± 0.67	0.88	0.82	0.86	0.79	0.83	0.75	0.71	0.19
$\mathcal{D}_{E(2)}$	$\mathcal{W}_a$	0.95 ± 0.47	1.10 ± 0.53	0.74	0.76	0.74	0.79	0.73	0.79	0.62	0.72
	$\mathcal{W}_b$	1.10 ± 0.57	1.19 ± 0.60	0.82	0.76	0.81	0.77	0.80	0.75	0.58	0.60
	$\mathcal{W}_c$	1.46 ± 0.84	1.52 ± 0.87	0.82	0.73	0.78	0.72	0.75	0.70	0.44	0.43
	$\mathcal{W}_d$	1.81 ± 1.07	1.80 ± 1.08	0.81	0.71	0.77	0.68	0.73	0.64	0.36	0.30

Table 3: Summary of positional (Δp) and orientational errors ($\Delta\alpha$) for the RL-based controller under all considered sea states and actuation delays. The first column group gives the average errors with standard deviation. The second shows the decrease in average error compared to the IK controller (0.8 means 80% reduction). The third and forth groups also show the decrease in error, but calculated across the top 5 and 1 percent of errors respectively. The last shows reduction of maximum error.

Autonomous marine operations require a system that can accurately sense its surroundings. Specifying a complete set of sensor requirements is beyond the scope of this work. However, we note that the solution presented here is relatively lightweight, as it only requires tracking the relative pose of vessel V_b, information that is typically available in these settings (see e.g. Tørdal and Hovland (2017)). Delays caused by measurement processing and filtering could potentially also be absorbed by learned delay compensation.

Full-scale arms differ in dynamics from the UR10e arm used here. As noted by (Brandt et al. 2022), longer links are likely to make vibrations a more prominent concern. High-fidelity simulations or real-world experiments with a full-scale arm are needed to explore how this affects the control problem. It would also be interesting to investigate whether residual policy learning can mitigate the impact of vibrations. Since the motion from vibrations is relatively small, small corrections might be sufficient to counteract them.

Unless a very high-fidelity simulator of the full-scale arm and vessel dynamics is created, there is likely going to be a so-called sim-to-real gap that must be bridged. Alternatively, the system can be trained or fine-tuned online. In this case, sample complexity becomes a very relevant concern. Preliminary experiments suggest that it is possible to cut training time down substantially with simple measures such as using a more aggressive learning rate or swapping out the LSTM core with GRU (Cho et al. 2014). It could also be worthwhile to experiment with more data-efficient off-policy RL algorithms.

Conclusion

In this paper, we study the impact actuation delays has on vessel-to-vessel motion compensation with a robotic arm and try to mitigate the resulting errors. We design a hybrid solution where an RL agent learns to issue small corrections to an underlying IK controller. A clear advantage of this approach, as opposed to e.g. giving full control to the RL agent, is that it limits the hazard potential of the black-box NN policy. The RL agent only requires modest sensory input and is agnostic to the underlying controller, making it a flexible add-on that can be included in diverse setups.

Numerical experiments in simulation showed that increasing actuation delays and more intense sea states has a compounding effect on the tracking errors, especially for the positional component of the error. The corrections predicted by the RL agent lead to a substantial decrease in errors ($\sim$50-80%), with the largest reductions being observed while under the influence of larger actuation delays. Potential future work includes motion compensation while following trajectories that are localized on another vessel, running simulations of full-scale arms, addressing the issue of sample complexity, and in-lab experiments.

Acknowledgements

This work was funded by The Research Council of Norway (RCN) through the Center for Research-based Innovation Exposed Aquaculture Operations (RCN project number 237790). We thank Halgeir Ludvigsen (SINTEF Ocean AS) for providing the ship motion data used in the simulation.

References

Andersen, T. T.; Amor, H. B.; Andersen, N. A.; and Ravn, O. 2015. Measuring and Modelling Delays in Robot Manipulators for Temporally Precise Control Using Machine Learning. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 168–175.

Bjelland, H. V.; Føre, M.; Lader, P.; Kristiansen, D.; Holmen, I. M.; Fredheim, A.; Grøtli, E. I.; Fathi, D. E.; Oppedal, F.; Utne, I. B.; and Schjølberg, I. 2015. Exposed Aquaculture in Norway. In *OCEANS 2015 - MTS/IEEE Washington*, 1–10.

Brandt, M. A.; Herland, S.; Gutsch, M.; Ludvigsen, H.; and Grøtli, E. I. 2022. Towards Autonomous Contact-Free Operations in Aquaculture. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4176492. Forthcoming.

Cao, Y.; and Li, T. 2020. Review of antiswing control of shipboard cranes. *IEEE/CAA Journal of Automatica Sinica*, 7(2): 346–354.

Cho, K.; van Merrienboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. arXiv:1409.1259.

From, P. J.; Duindam, V.; Gravdahl, J. T.; and Sastry, S. 2009. Modeling and motion planning for mechanisms on a non-inertial base. In *2009 IEEE International Conference on Robotics and Automation*, 3320–3326.

From, P. J.; Gravdahl, J. T.; Lillehagen, T.; and Abbeel, P. 2011. Motion planning and control of robotic manipulators on seaborne platforms. *Control Engineering Practice*, 19(8): 809–819.

Hasselmann, K.; Barnett, T.; Bouws, E.; Carlson, H.; Cartwright, D.; Enke, K.; Ewing, J.; Gienapp, H.; Hasselmann, D.; Kruseman, P.; Meerburg, A.; Muller, P.; Olbers, D.; Richter, K.; Sell, W.; and Walden, H. 1973. Measurements of wind-wave growth and swell decay during the Joint North Sea Wave Project (JONSWAP). *Deut. Hydrogr. Z.*, 8: 1–95.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-term Memory. *Neural computation*, 9: 1735–80.

Huber, P. J. 1964. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1): 73 – 101.

Jha, A.; Subedi, D.; Løvsland, P.-O.; Tyapin, I.; Reddy Cenkeramaddi, L.; Lozano, B.; and Hovland, G. 2020. Autonomous Mooring towards Autonomous Maritime Navigation and Offshore Operations. In *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 1171–1175.

Johannink, T.; Bahl, S.; Nair, A.; Luo, J.; Kumar, A.; Loskyll, M.; Ojea, J. A.; Solowjow, E.; and Levine, S. 2019. Residual Reinforcement Learning for Robot Control. In *2019 International Conference on Robotics and Automation (ICRA)*, 6023–6029.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement Learning: A Survey. *J. Artif. Int. Res.*, 4(1): 237–285.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Kober, J.; Bagnell, J. A.; and Peters, J. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11): 1238–1274.

Manal, K.; and Rose, W. 2007. A general solution for the time delay introduced by a low-pass Butterworth digital filter: An application to musculoskeletal modeling. *Journal of Biomechanics*, 40(3): 678–681.

Nguyen, H.; and La, H. 2019. Review of Deep Reinforcement Learning for Robot Manipulation. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 590–595.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015. High-Dimensional Continuous Control Using Generalized Advantage Estimation. arXiv:1506.02438.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Silver, T.; Allen, K.; Tenenbaum, J.; and Kaelbling, L. 2018. Residual Policy Learning. arXiv:1812.06298.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033.

Tunyasuvunakool, S.; Muldal, A.; Doron, Y.; Liu, S.; Bohez, S.; Merel, J.; Erez, T.; Lillicrap, T.; Heess, N.; and Tassa, Y. 2020. dm_control: Software and tasks for continuous control. *Software Impacts*, 6: 100022.

Tørdal, S. S.; and Hovland, G. 2017. Inverse kinematic control of an industrial robot used in Vessel-to-Vessel Motion Compensation. In *2017 25th Mediterranean Conference on Control and Automation (MED)*, 1392–1397.

Wampler, C. W. 1986. Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1): 93–101.

Zhang, R.; Hou, J.; Chen, G.; Li, Z.; Chen, J.; and Knoll, A. 2022. Residual Policy Learning Facilitates Efficient Model-Free Autonomous Racing. *IEEE Robotics and Automation Letters*, 1–8.

Zhou, Y.; Barnes, C.; Lu, J.; Yang, J.; and Li, H. 2019. On the Continuity of Rotation Representations in Neural Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5738–5746.