Roboat: A Novel Autonomous Surface Vessel for Urban Transportation

Wei Wang1,2, David Fernández-Gutiérrez1, Rens Doornbusch3, Joshua Jordan3, Tixiao Shan2, Pietro Leoni2, Niklas Hagemann2, Jonathan Klein Schiphorst3, Fabio Duarte2, Carlo Ratti2, and Daniela Rus1

Abstract—In this paper, we present our novel autonomous surface vessel (ASV) platform, a full-scale Roboat for urban transportation. This 4-meter-long Roboat is designed with six seats and can carry a payload of up to 1000 kg. Roboat has two main thrusters for cruising and two tunnel thrusters to accommodate docking and inter-connectivity between Roboats. With accurate receding horizon tracking control, path planning, obstacle tracking, and avoidance capabilities, Roboat can navigate in urban canals to perform transportation tasks. We build a nonlinear dynamic model for Roboat and implement an online nonlinear model predictive controller (NMPC) for the trajectory tracking while transporting passengers. We use a sparse directed graph to represent the canal topological map, and find the most time-efficient global path in a city-scale environment using the A∗ algorithm. We then employ a multi-objective algorithm’s lexicographic search to generate an obstacle-free path using a point cloud projected 2D occupancy grid map. Extensive experiments in Amsterdam waterways demonstrate that Roboat can 1) successfully track the optimal trajectories generated by the planner with varying numbers of passengers on board; 2) execute an autonomous water taxi task where it docks to pick up passengers, drive passengers to the destination while planning its path to avoid obstacles and finally dock to drop off passengers.

I. INTRODUCTION

Autonomous Surface Vessels (ASVs) has become increasingly popular for marine exploration and industrial applications, such as search and rescue, environmental monitoring, and hydrology surveying [1]–[6]. As the road traffic is becoming more and more congested, ASVs could be an attractive alternative for the transportation system in many coastal and riverside cities such as Amsterdam and Venice. The deployment of a self-driving surface fleet can shift parts of transportation from roads to waterways, which may reduce the road traffic congestion in these water-related cities.

We focus on designing and developing of autonomous surface vessels for urban transportation and also constructing dynamic floating infrastructure (such as bridges and stages) with these ASVs [7]–[14]. After reviewing more than 50 ASV prototypes in the literature [15], we find that most of the existing ASVs are scaled models with either a single hull or a double hull. The hulls do not seem to have been carefully designed for realistic transportation operations. Current ASV hulls are typically not designed into regular shapes to maximize the payload space and facilitate the docking and inter-connectivity between ASVs. Moreover, many ASVs do not have the docking/latching module. To address the above mentioned problems, in this work, we present a new design of Roboat which has a rectangular floor plan and six latching modules. We also introduce new features to enhance the stability and performance of the ASV, while easing its manufacturability.

Besides hardware design, we need to develop the autonomy package of Roboat in urban waterways. There have been a lot of progresses on ASV autonomy over the last several decades [6], [15], such as localization [1], [4], [5], trajectory tracking control [16], [17], object detection [18], and path planning [5], [19], [20]. However, the vast majority of existing ASVs are designed for open water applications [6], [17], [21] and thus will not work in confined and crowded urban water environments such as Amsterdam canals.

This paper focuses on developing an autonomous navigation package for ASVs like Roboat that operate in urban waterways. Trajectory tracking control is one of the essential capacities for ASVs. Many trajectory tracking strategies such as adaptive control [22], integrator back-stepping method [16], [23], and sliding mode method [24], [25], have been designed for ASVs. Nevertheless, most of the ASV controllers are only validated in the simulation. Moreover, a number of ASV controllers employ a kinematic model instead of a dynamic model, which will degrade the control performance due to the highly non-linearity of the water environment and the persistent disturbances from the water environment. Furthermore, the tracking control of Roboat is more challenging since Roboat is a non-traditional real-sized boat that is more prone to turn. In this work, we build an nonlinear dynamic model and implement an nonlinear model predictive controller for Roboat to achieve stable trajectory tracking in various payload conditions. To address the navigation challenge in the canals, we design a hierarchical planning system that includes global and local planners. The global path is obtained by searching a sparse graph reflecting the topology of canals at the city-scale level. The local planner generates obstacle-free paths by considering the detected obstacles from onboard perceptual sensors.

 Compared with our previous work on the quarter-scale and half-scale Roboat [7]–[11], [13], this paper design and develop a four-meter-long full-scale Roboat, which is ca-
able of autonomously navigating in canals with up to six passengers and providing water taxi service for the people in demand. In particular, our work makes the following contributions in designing and developing the new Roboat:

- Novel design and the building of a 4-meter-long Roboat for urban transportation;
- Model identification and revised NMPC control strategy for the full-scale Roboat with main thrusters and tunnel thrusters;
- City-scale path planning algorithm and high-resolution lidar-based obstacle detection and tracking algorithms;
- Extensive experiments to validate the NMPC and the autonomy package in different payload (passenger) conditions in the Amsterdam canals.

II. ROBOAT DESIGN

A. Hull design

Evolving from the quarter-scale and half-scale prototypes [8] [26], the final design of Roboat introduces new features that enhance its stability and performance. It still uses only flat and single-curved aluminum plates as shown in Fig. 1, keeping its ease of manufacturing.

![Mechanical design of the Roboat prototype.](image1)

To facilitate the docking and inter-connectivity between Roboats, our Roboat adopts a rectangular floor-plan that differs from usual ship designs, which is typically more slender. This choice of hull shape affects its dynamic response, making the boat more prone to turn. Therefore, it needs to be accounted for in the control model. A central keel is introduced in the new design to improve the stability of the boat. This allows us to raise the inner deck above the waterline while keeping the maximum expected roll angles below 10°. Overall, the hull shape has been streamlined from its predecessors, reducing the forward drag by 45%. The introduction of the keel also improves the directional stability of the boat reducing the steering needs, which directly benefits the robustness of the control system. While the thruster arrangement used in previous designs is maintained, the nature of the thrusters varies now. Instead of four equal units, we distinguish between two main propellers placed at starboard and port for forwards and backwards thrust and two tunnel propellers placed at the bow and stern ends that provide lateral thrust. The Roboat hull provides a stable floating platform on top of which different swappable top-decks can be installed. Fig. 2 demonstrates the passenger transportation application.

![Roboat prototype being tested at AMS Institute with passenger seating top-deck.](image2)

The hydrostatic properties of the Roboat depend on its hull geometry. The weight of the Roboat itself is ~1050 kg, including the top-deck, which ensures that the keel is always submerged and allows to carry the design payload of 1000 kg. In particular, for the passenger transportation use case, the Roboat can then carry safely 6 passengers.

B. Electronics and Sensors

Fig. 3 overviews the current Roboat hardware. In particular, Roboat can last about 12 hours with a battery that has a capacity of 12 kWh. Together with the batteries, the thrusters that propel the boat are a critical choice. Two Torqeedo POD Cruise 4.0 FP drives were selected as main units, delivering each a maximum propulsive power of 2.24 kW. For the lateral movement, two Vetus BOW A0301 tunnel thrusters were chosen to provide a maximum lateral thrust of 30 kgf each, completing the propulsion system.
An Intel NUC with the ROS framework is used for the onboard computation. The primary sensors of the boat are a dual RTK GPS, a 128-beam LiDAR, a camera, and an IMU. For physically connecting the boat to other boats or to a docking station, the latching system is used. The latching system consists of mechanical robotic arms that reach out to grab the target, a camera for target perception, and a Raspberry Pi 4 for image processing. We refer the readers to [27] for more details of the latching system.

III. BOAT DYNAMICS AND CONTROL

A. Dynamic Model

The dynamics of our Roboat is governed by the nonlinear differential equation below [7]

\[
\dot{x} = T(x)v
\]

(1)

\[
\dot{v} = M^{-1}(\tau + \tau_{\text{env}}) - M^{-1}(C(v) + D(v))v
\]

(2)

where \(x = [x \ y \ \psi]^{T} \in \mathbb{R}^{3 \times 1}\) represents the pose (position and heading) of the vessel in the inertial frame; \(v = [u \ v \ r]^{T} \in \mathbb{R}^{3 \times 1}\) represents the velocity of the vessel in the body-fixed frame; \(T(x) \in \mathbb{R}^{3 \times 3}\) represents the rotation matrix expressing the transformation from body-fixed frame to inertial frame; \(M \in \mathbb{R}^{3 \times 3}\) represents the symmetric positive-definite added mass and inertia matrix; \(C(v) \in \mathbb{R}^{3 \times 3}\) represents the skew-symmetric matrix accounting for the Coriolis and centripetal forces; \(\tau_{\text{env}} \in \mathbb{R}^{3 \times 1}\) represents the environmental disturbances from the currents, waves and winds; \(D(v) \in \mathbb{R}^{3 \times 3}\) represents the positive-semi-definite drag matrix; \(\tau = [\tau_{h} \ \tau_{r} \ \tau_{u}]^{T} \in \mathbb{R}^{3 \times 1}\) represents the total force and torque exerted from the thrusters, which is defined as follow

\[
\tau = Bu = \begin{bmatrix}
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
\frac{1}{2} & 2 & -\frac{1}{2} & -2
\end{bmatrix}
\begin{bmatrix}
f_1 \\
f_2 \\
f_3 \\
f_4
\end{bmatrix}
\]

(3)

where \(B \in \mathbb{R}^{4 \times 3}\) denotes the control matrix which defines the thruster configuration, \(u = [f_1 \ f_2 \ f_3 \ f_4]^{T} \in \mathbb{R}^{4 \times 1}\) denotes the control vector where \(f_1, f_2, f_3\) and \(f_4\) represent respectively the forces generated by left, right, anterior, and rear thrusters; \(a\) is the distance between the main thrusters and \(b\) is the distance between the tunnel thrusters. We refer the readers to [7] for more details of \(M, C(v)\) and \(D(v)\).

By reformulating (1) and (2), the complete dynamic model of the vessel can be obtained as follow

\[
\dot{q}(t) = f(q(t), u(t))
\]

(4)

where \(q = [x \ y \ \psi \ u \ v \ r]^{T} \in \mathbb{R}^{6 \times 1}\) denotes the state of the vessel, and \(f(\cdot, \cdot) : \mathbb{R}^{n_{q}} \times \mathbb{R}^{n_{u}} \rightarrow \mathbb{R}^{n_{q}}\) denotes the continuously differentiable state update function. The system model (4) defines how the state \(q\) evolves with the control input \(u \in \mathbb{R}^{4 \times 1}\). Moreover, a nonlinear measurement model \(h(t)\) can be defined as follow

\[
z(t) = h(q(t), u(t))
\]

(5)

where \(z = [x \ y \ \psi \ r \ f_1 \ f_2 \ f_3 \ f_4] \in \mathbb{R}^{8 \times 1}\) represents the measurement vector, and \(h(\cdot, \cdot) : \mathbb{R}^{n_{q}} \times \mathbb{R}^{n_{u}} \rightarrow \mathbb{R}^{n_{z}}\) represents measurement function.

The dynamic model has an unknown hydrodynamic parameter vector, \(\xi = [m_{11} \ m_{22} \ m_{33} \ n_{x} \ n_{y} \ n_{z}]^{T}\), that needs to be identified before running the controller. The estimation of \(\xi\) is a grey-box identification problem, and can be formulated as follow

\[
\min_{\xi} \sum_{t=0}^{T_e} \varepsilon(t)^{T}w\varepsilon(t),
\]

(6a)

s.t. \(\xi_{l} \leq \xi \leq \xi_{u}\),

(6b)

where \(\varepsilon(t) \in \mathbb{R}^{3 \times 1}\) denotes the deviation between the experimental velocity \(v'(t)\) and the simulated velocity \(v''(t)\) at time \(t\). \(\xi_{l}\) and \(\xi_{u}\) denote the lower and upper bounds of the unknown vector \(\xi\), respectively. \(w \in \mathbb{R}^{3 \times 3}\) denotes the weight matrix in the optimization. In this work, we employ the Sequential Quadratic Programming (SQP) method to solve (6) numerically since (6) is a small-scale optimization problem and the SQP can satisfy the bounds at all iterations.

B. Nonlinear Model Predictive Control

We formulate the optimal trajectory tracking control problem for NMPC as a least square function which penalizes the deviations of state (\(q_{k}\)) and control (\(u_{k}\)) trajectories from their specified references, over the given horizon \(N_{e}\) for daily operations. To achieve this, we penalize the tunnel thrusters much heavier than that of the main thrusters as

\[
\min_{q_{k}, u_{k}} \sum_{k=0}^{N_{e}-1} \left( \|q_{k} - q_{k}^{\text{ref}}\|_{W_{q}}^{2} + \|u_{k} - u_{k}^{\text{ref}}\|_{W_{u}}^{2} \right)
\]

(7a)

\[
\|q_{N_{e}} - q_{N_{e}+1}^{\text{ref}}\|_{W_{q}}^{2}
\]

s.t. \(q_{j} \geq q_{j}^{\text{ref}}, j = 0, \ldots, N_{e}\),

(7b)

\(q_{k,\min} \leq q_{k} \leq q_{k,\max}, k = j, \ldots, j + N_{e}\),

(7c)

\(u_{k,\min} \leq u_{k} \leq u_{k,\max}, k = j, \ldots, j + N_{e} - 1\),

(7d)

where \(q_{j} \in \mathbb{R}^{n_{q}}\) is the current state estimate, \(q_{k}^{\text{ref}}\) and \(u_{k}^{\text{ref}}\) are respectively the time-varying state and control references; \(q_{N_{e}}^{\text{ref}}\) is the terminal state reference; \(W_{q} \in \mathbb{R}^{n_{q} \times n_{q}}\), \(W_{u} \in \mathbb{R}^{n_{u} \times n_{u}}\), and \(W_{N_{e}} \in \mathbb{R}^{n_{q} \times n_{q}}\) are the positive-definite weight matrices that penalize the deviations from the desired values. These weight matrices are constant in this study and need to be adjusted to appropriate values. Moreover, \(q_{k,\min}\) and \(q_{k,\max}\) are respectively the lower and upper bounds of the vessel states; \(u_{k,\min}\) and \(u_{k,\max}\) are respectively the lower and upper bounds of the control.

The weighting matrices \(W_{q}, W_{u}\) and \(W_{N_{e}}\) for the NMPC used in the experiments are chosen as

\[
W_{q} = \text{diag}\{6000, 6000, 7000, 2000, 2000, 1200\}
\]

(8)

\[
W_{u} = \text{diag}\{6000, 6000, 7000, 2000, 2000, 1200\}
\]

(9)

\[
W_{N_{e}} = \text{diag}\{1, 1, 4, 4\}
\]

(10)

Note that we would like to mainly rely on the main thrusters for daily operations. To achieve this, we penalize the tunnel thrusters much heavier than that of the main thrusters as listed in weighting matrix \(W_{u}\). The prediction horizon \(N_{e} = 4\) s, and the constraints on the control input \(u\) used in the
experiments are selected as follow: \(-1030 \leq f_1 \leq 1680\ N, -1030 \leq f_2 \leq 1680\ N, -450 \leq f_3 \leq 450\ N, -450 \leq f_4 \leq 450\ N.\)

IV. NAVIGATION

In this section, we introduce the navigation framework of our Roboat, as shown in Fig. 4. Roboat localizes itself by running the dual RTK GPS, which can achieve centimeter-level accuracy. The heading of Roboat is also calculated by the dual GPS. The NMPC will accurately and robustly track the reference trajectories from the planner during the whole task in an urban waterway. The planner will avoid obstacles from the obstacle occupancy map generated by the 3D LiDAR sensors. We will describe the object tracking and the path planning in the following.

A. Obstacle Detection and Tracking

We rely on the 3D point cloud received from the high-resolution Ouster OS1-128 lidar for obstacle detection and tracking. Because the lidar perceives more than 1.3 million points per second from its surrounding environment, we require a series of efficient algorithms to achieve real-time performance with limited computational resources.

1) Reduce point cloud size: we first voxelize the raw point cloud with a resolution of 0.2 meters. The downsampled point cloud is then transformed into the global frame using the current robot pose. We further reduce the point cloud size by cropping with point elevation. Because the robot only travels in a horizontal plane, points above or below a specific height are discarded. Finally, we perform euclidean distance clustering to filter objects under a certain size. The clusters with point numbers below a threshold are removed. This point number is chosen to allow us to recognize swimmers while ignoring smaller objects such as leaves. All the remaining clusters are considered as obstacles and passed into the following system for obstacle tracking.

2) Obstacle tracking: we perform obstacle tracking using the Kalman filter (KF) [28], which estimates the position and velocity of the point cloud clusters. From these clusters, we calculate the centroid for each cluster in the horizontal plane. At system initialization, we initialize a list of KF instances with these centroids for each cluster. Each cluster is assumed to have zero velocity at initialization. After system initialization, we use the Hungarian algorithm to associate the newly arrived cluster centroids with the predicted centroids from their corresponding KFs. If a new cluster can be associated with an existing cluster, the corresponding KF is updated with the new cluster centroid. Otherwise, a new KF is initialized for this new cluster. If an existing cluster hasn’t been observed for a certain time, we consider that it has left the scene and delete the relevant KF. The successfully tracked cluster information is passed to the path planning module to allow the planner to navigate around obstacles.

B. Path Planning

The path planning system of Roboat is divided into two parts: the global planner, which finds the most time-efficient path in a city-scale environment, the local planner, which returns the obstacle-free path in the local environment.

1) Global planner: we formulate the global planning task as solving a graph search problem. A sparse directed graph \( G = (V, E) \) can be built using the canal topological map, with vertices \( V \) indicates the intersection of canal segments and edges \( E \) represents the canal segment. Upon receiving a goal location, we search \( G \) using the \( A^* \) algorithm and return the shortest path between the robot and the goal. Note that the path from the global planner may not be obstacle-free. However, this path provides an instantaneous solution in a large-scale environment and serves as guidance for the local planner.

2) Local planner: the local planner produces an obstacle-free path using the global planner path and the tracked point cloud clusters (Sec. IV-A). We first build a 2D occupancy grid map which is centered at the robot. The tracked point cloud clusters are then projected onto the map, marking the regions occupied by the obstacles.

The goal for the local planner is selected as a point on the global path 10 seconds past the NMPC prediction horizon. This value was chosen to reduce the computational cost of the search algorithm while keeping the path stable enough for the NMPC prediction. Finally, we employ lexicographic search [12], a multi-objective search algorithm, to generate an obstacle-free path. Two costs, risk [29] and distance, are considered here for local planning. We refer the readers to [12] for more details of the configuration of two costs.

V. EXPERIMENTS AND RESULTS

We consider two tasks in Amsterdam canals to demonstrate the effectiveness of the developed autonomous system: TrajectoryTrackingTask and WaterTaxiTask. TrajectoryTrackingTask requires Roboat to follow a sinusoidal trajectory at a predefined speed while different numbers of passengers sitting at varied areas. This task aims at evaluating the NMPC tracking control in handling various passenger conditions. WaterTaxiTask requires Roboat to complete a taxi service on the water autonomously, which contains a sequence of actions including pick up passenger at a docking station.
navigate to the destination, and drop off passenger at a docking station. The objective of WaterTaxiTask is to demonstrate Roboat autonomy in a realistic task. We use C++ to implement our autonomous package in the ROS environment. We run all our experiments on an onboard NUC of Roboat. State estimation runs at 100 Hz and all the other algorithms run at a rate of 10 Hz.

A. Model Identification Results

The unknown parameters in dynamic model (4) of Roboat is experimentally identified as follows: Roboat was manually controlled to follow a sinusoidal trajectory in the canal for 150 seconds. The Roboat poses, velocities and thruster forces were recorded during the experiments. The experiment were repeated five times. The optimization algorithm described in Section III-A was used to identify the unknown parameters in Eqn. 4. Finally, we obtained the identified parameters $\xi$ as follows:

\[
\begin{align*}
    m_{11} &= 1169 \text{ kg}, \\
    m_{22} &= 1169 \text{ kg}, \\
    m_{33} &= 3800 \text{ kg}, \\
    X_u &= 126 \text{ kg/s}, \\
    Y_v &= 460 \text{ kg/s} \text{ and } N_r = 160 \text{ kg} \cdot \text{m}^2/\text{s}.
\end{align*}
\]

In addition, the thruster distance parameters are listed as follow:

\[
\begin{align*}
    a &= 1.4 \text{ m}, \\
    b &= 2.05 \text{ m}.
\end{align*}
\]

B. Tracking Control Results

We tested four conditions of passenger transport in the TrajectoryTrackingTask: two passengers sitting in the rear area, six passengers sitting in the central area, six passengers sitting in the port area and six passengers sitting in the starboard area. All the reference trajectory are the same but with a slight different start position and heading. The duration of the reference trajectory is 63 seconds. The weight of each passenger is around 80 kg. As the weight of Roboat is 1050 kg, the passengers increase the Roboat weight from 15.2% to 38.1%, which will affect the hydrodynamics to a certain degree.

We now present the trajectory tracking results of our Roboat while transporting passengers. Tracking performance and tracking errors for the four conditions of passenger transport are shown in Fig. 5 and Fig. 6. Our first observation from Fig. 5 is that our NMPC controller is able to track on the reference trajectory in all four conditions while transporting passengers in natural water environments. The average position tracking RMSE (Root Mean Square Error) value for the four conditions in Fig. 6 are respectively 0.3868 m, 0.6063 m, 0.7624 m, and 0.6544 m. The tracking errors indicate that both the payload (passengers) and payload distribution (passenger distribution) can degrade the trajectory tracking performance.

Moreover, Fig. 7 shows the corresponding thruster forces for the four conditions. It is clear that the main thrusters on the left and right side of Roboat contribute significantly as the system in on-track. If the system is not on-track, all the thrusters would contribute significantly (such as before 10 s) to help the system rapidly reach the reference trajectory. We calculate the propulsive power consumption of Roboat by using the following equation:

\[
p = \frac{(|f_1| + |f_2| + |f_3| + |f_4|) \times U}{\zeta} \tag{11}
\]

where $U$ is the forward speed of the robot and $\zeta$ is the thruster efficiency which is set to 0.8 in this study. The power consumption for each condition are respectively 147.8 W, 188.1 W, 260.8 W, and 214.1 W. We can see that the total passenger weight and weight distribution affects the power consumption. Moreover, it can be found that the power consumption is positively related to the tracking error. One possible reason for the large power consumption in situation (b), (c) and (d) is that the NMPC exerts greater efforts to eliminate the large tracking errors. The above results show that the NMPC is able to successfully track the trajectory reference while transporting passengers. But the results also
suggest that the current NMPC could be further improved by actively dealing with the payload weight and payload distribution in the near future.

C. Navigation Results

The WaterTaxiTask is executed in the Amsterdam canal close to the Amsterdam Institute for Advanced Metropolitan Solutions (AMS Institute). As illustrated in Fig. 8, the WaterTaxiTask is completed as follows. In the experiment, the boat started in random orientation at point ‘A’. It then moved to point ‘B’ which we call the "prelatching position" with a heading parallel to the dock marked in green. From point B it moves sideways to the dock at point ‘C’. Once arrived at ‘C’, a passenger enters the boat. The boat then moves back to the prelatching position at ‘B’ to then proceed to point ‘D’. Point ‘D’ was deliberately chosen to be in between the obstacles marked in red and the wall. The total length of this route was 99 m. The space between the obstacles and the wall closely resembles the available space in the Amsterdam canals. The obstacles are detected and a safe route to avoid the obstacles is planned. Finally, the boat is told to move to prelatching position ‘E’ with a southeast heading, again parallel to the dock. It then finishes the experiment by moving sideways to the dock at ‘F’ to drop-off the passenger.

We successfully performed three consecutive water taxi experiments with our developed Roboat platform to show the robustness of our water taxi navigation. This was a total of 300 m of autonomous sailing with all the obstacles shown in red in Fig. 8 correctly identified and avoided. We repeated the experiments three times. We used three cameras to record the experiments. Fig. 9 shows some snapshots of the water taxi experiments. Readers can review the taxi experiment more clearly by watching the video submitted with the paper.

VI. CONCLUSIONS

In this paper, we have developed a novel autonomous surface vessel (ASV) which can transport up to six passengers in urban cities. Roboat is capable of performing receding horizon tracking control, path planning and obstacle tracking and avoidance in urban waterways.

Our work will be extended in the following directions in the near future. First, will add traffic rules into the planner when we test Roboat in busy canals. Second, we will estimate disturbances such as currents and waves to further improve the tracking performance in more noisy waters. Third, we will develop algorithms for multi-robot formation control and self-assembly on the water, enabling the construction of on-demand large-scale infrastructure.
REFERENCES


