Design and Control of Platoons of Cooperating Autonomous Surface Vessels

Bradley E. Bishop
United States Naval Academy
105 Maryland Ave (Stop 14a)
Annapolis, MD 21401
Tel: (410)-293-6117
bishop@usna.edu

Abstract: This work focuses on the development of a provable, reliable and efficient strategy for coordination of groups, or platoons, of small, autonomous surface vessels for use in maritime security operations. This work builds on successful techniques that have been developed for coordination of platoons of holonomic vehicles as well as technology that has utilized in the development of an autonomous surface vessel test bed.

I. Introduction

Platoons or swarms of cooperating robotic vehicles provide the capability for highly efficient, fault tolerant operations. Additionally, autonomous surface vessels (ASVs), unlike autonomous underwater vehicles, admit the use of high bandwidth communications and full spectrum sensing. On the other hand, ASVs often require sophisticated control algorithms, as they are principally nonholonomic systems. Such systems have boundary constraints on achievable velocities, which provide substantial challenges for controller design. The most straightforward example of the difficulties associated with nonholonomic systems is the parallel parking problem. The ASV problem is further complicated by system drift and additional, unactuated degrees of freedom.

In this work, we present a basic coordination scheme for controlling platoons of cooperating surface vehicles. We use techniques from redundant manipulator control to generate a low-computation, real-time planning and control scheme. Platoon tasks are encoded as desired values of functions of the platoon state. Secondary platoon and individual unit objectives are embedded in the controller and are carried out using available platoon resources without affecting the primary tasks. The effects of nonholonomicity on the control are investigated with regards to the performance objectives and secondary tasks. Results are demonstrated using simulations of simplified craft and environmental conditions.

In the forward-looking portion of this work, we consider the most likely form of implementation of platoons of ASVs. It is noted that ASVs that are to be used for security operations will most likely operate in the planing state during transitions and (potentially) while carrying out missions. Control of planing ASVs relies on either a firm understanding of the dynamics of planing craft or an extremely robust control algorithm. As planing dynamics are quite complex and numerically challenging, robust control methods are discussed, including techniques such as genetic algorithms. Finally, we discuss the implications of planing craft dynamics as they inform and direct implementation of the developed control.

The remainder of the paper is organized as follows. In Section II, the fundamental control methodology for platoon coordination is discussed. Section III contains a discussion of the application of this methodology to nonholonomic vessels. Implementation issues are discussed in Section IV, and conclusions and future work are outlined in Section IV.

* This work partially supported by the Office of Naval Research, grant N0001402WR20357 as well as the ONR National Naval Responsibility in Naval Engineering (NNR-NE) program, grant N0014-03-1-0160.
II. Basic Platoon Control Algorithm

We begin the discussion of the basic platoon controller by making a few simplifying assumptions. We assume that the units of the platoon are holonomic, possessing two directly controllable degrees of freedom each (x and y position in a plane). The control method, referred to as statistical platoon control, regulates desired platoon-level performance (e.g., desired variance) while still allowing the individual units some degree of autonomy. Statistical controls can be run in real time (without explicit path planning), admitting the use of reactive methods.

Statistical methods control platoon level functions, such as mean and variance of positions of the units, which in turn restrict the movements of a platoon as a whole [1]. This approach involves using a special matrix, known as a Jacobian, that allows the calculation of the best velocity profile for the units according to the set limits of the performance of the platoon. Jacobian matrices are typically applied in traditional robotics to relate joint variable velocity to end-effector (tool) velocity [2]. A Jacobian matrix is defined by the state \( q \) of the system and some task function \( f(q) \). Denoted \( J(q) \), the Jacobian is given by:

\[
J(q) = \begin{bmatrix}
\frac{\partial f_1(q)}{\partial q_1} & \cdots & \frac{\partial f_1(q)}{\partial q_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial f_m(q)}{\partial q_1} & \cdots & \frac{\partial f_m(q)}{\partial q_n}
\end{bmatrix}
\]

where the state \( q \) is \( n \) dimensional and \( \dot{f}(q) = J(q)\dot{q} \). For a platoon with \( r \) units, \( q \) is given by \( q = [x_1, x_2, \ldots, x_r, y_1, y_2, \ldots, y_r]^T \), the concatenation of the 2D coordinates of the units.

For a large platoon of robots, the number of state variables (x and y for each unit) is typically greater than the number of task variables (platoon mean position and variance in each dimension, for example). This redundancy creates an infinite number of possible configurations for the platoon while still achieving the desired task profile. In Equation (2), we see a simple and useful task function \( f(q) \), defined by the mean and variance of the x and y components of the state of the platoon. The mean determines the platoon position while variance dictates the spread of the elements without explicitly defining each unit’s position. The Jacobian for (2) is defined by Equation (1), and can be found in [1,3,7].

\[
f(q) = \begin{bmatrix}
\mu_x \\
\sigma_x \\
\mu_y \\
\sigma_y
\end{bmatrix} = \begin{bmatrix}
\frac{1}{r} \sum_{i=1}^{r} q_i \\
\frac{1}{r-1} \sum_{i=1}^{r} (q_i - \mu_x)^2 \\
\frac{1}{r} \sum_{i=r+1}^{2r} q_i \\
\frac{1}{r-1} \sum_{i=r+1}^{2r} (q_i - \mu_x)^2
\end{bmatrix}
\]

Although each unit is constrained by the platoon objectives, there is still a great deal of flexibility and autonomy. This flexibility can be characterized by the dimension of the null space of the Jacobian. Given a platoon with a total of \( 2r \) degrees of freedom and an \( m \)-dimensional task function \( f(q) \), there are effectively \( 2r - m \) degrees of freedom left over after the main task is achieved. This number is the dimension of the null space of the system, which is a manifold of positions on which the main task \( f(q) \) is achieved. In simple terms, it is easiest to see
the concept of null space when considering two cooperating units regulating their average position. Given a desired center point, there are an infinite variety of ways to position the units. However, if one unit moves, the other unit must also move in a predictable manner to maintain the desired center point.

Gradient projection controllers allow systems with more degrees of freedom than task variables to coordinate motions in a way that maintains the task variable regulation while using the additional degrees of freedom to carry out secondary objectives. The basic controller is given by:

\[
\dot{q} = J^+(K(f_d(q) - f(q)) + \dot{f}_d(q)) + (I - J^+J)v
\]

where \( J^+ \) is the pseudoinverse of \( J \) given by \( J^T(JJ^T)^{-1} \), \( K \) is a controller gain matrix, \( f_d \) is a desired task function trajectory, \( (I - J^+J) \) is the projection operator that guarantees coordination, and \( v \) is an encoded secondary task. The secondary task is carried out on the null space of the primary task. That is, it is achieved using extra DOFs with no effect on the primary task \( f(q) \). The overall control architecture is shown in Figure 1. A more complete overview of this control method can be found in [1,3].

![Figure 1: Basic Platoon Control Architecture](image)

The velocity-based secondary task \( v \) can be anything that can be cast in the form of differential equations that describe tasks that are also velocity based. A variety of different objectives can be carried out using this method [1,3,4,8].

Unfortunately, the controller developed assumes that the vehicles are holonomic in nature. In the following section, we investigate methods for extending this control to nonholonomic vehicles such as an autonomous surface vessel.

### III. Control of Platoons of ASVs

In order to study a more realistic problem, we have chosen to address the control of nonholonomic platoons. The model that we will use, while not reflecting the full richness of a surface vessel, contains the crucial kinematic velocity constraints that are the crux of the difficulties with the scheme outlined in Section II.

The model that we will use bears a strong similarity to the kinematics of a tricycle-drive robot [5], but also loosely represents the basic velocity kinematics of a surface vessel that has a
thruster and a single control surface (rudder). The dynamics are given in Equation (4) and illustrated in Figure 2, where the vehicle speed is \( u \) and the rudder angle is \( \delta \). Note that there is a factor of \( \frac{1}{2} \) in the dynamics of this system that is not present in standard tricycle-drive systems, to represent in a limited way the effects of viscous friction and drag for ASVs.

\[
\begin{align*}
\dot{x} &= u \cos(\theta) \\
\dot{y} &= u \sin(\theta) \\
\dot{\theta} &= \frac{u}{2L} \sin(\delta)
\end{align*}
\tag{4}
\]

![Figure 2: Nonholonomic Surface Vessel Kinematic Model](image)

The controller shown in Figure 1 assumes that the desired velocity for each vehicle (as shown by \( \dot{q} \)) is exactly achieved. The control is precisely the derivative of the state \((x, y)\). In Equation (4), we see that the two controls (speed \( u \) and rudder angle \( \delta \)) actually control three variables \((x, y, \theta)\) through a nonlinear relationship. This is an underactuated system, where the dimension of the control space (two, in this case) is less than the dimension of the state space (three, in this example). The result of this underactuation is a nonholonomic constraint, limiting the achievable velocities of the system.

To accommodate this type of actuation, we write a single-unit controller that takes a desired velocity \((\dot{x}^d, \dot{y}^d)\) for a given unit and tracks it using the two available actuations. The controller is outlined in Equation (5), where \( K_u \) is a control gain.

\[
\begin{align*}
\dot{u} &= \sqrt{(\dot{x}^d)^2 + (\dot{y}^d)^2} \\
\dot{\theta}^d &= K_u \left( \tan^{-1}\left( \frac{\dot{y}^d}{\dot{x}^d} \right) - \theta \right) \\
\delta &= \begin{cases} 
\text{sgn}\left( \frac{2L\dot{\theta}^d}{u} \right) \frac{\pi}{4} & \text{when } \left| \frac{2L\dot{\theta}^d}{u} \right| > \sin\left( \frac{\pi}{4} \right) \\
\sin^{-1}\left( \frac{2L\dot{\theta}^d}{u} \right) & \text{when } \left| \frac{2L\dot{\theta}^d}{u} \right| \leq \sin\left( \frac{\pi}{4} \right)
\end{cases}
\end{align*}
\tag{5}
\]

Equations (4) and (5) are used in the ‘Platoon’ block of Figure 1, taking desired until velocities as inputs and providing the actual unit motions as outputs. The secondary objective \( v \) is defined by an obstacle avoidance vector which effectively ‘pushes’ a unit away from nearby obstacles and other units. This ‘push’ is projected onto the null space of the Jacobian so that other units react appropriately when one unit moves to avoid an obstacle. The mathematics for this secondary objective can be found in [1], and are excluded here for brevity.

Because the control is limited, the underactuated, nonholonomic system will not act in the same manner as it would under fully holonomic control. The two cases are illustrated in the following examples, where the platoon transitions from an initial deployment of \( q = [x_1 \ x_2 \ x_3 \ x_4 \ y_1 \ y_2 \ y_3 \ y_4]^T \) = [0.7, 0.5, 0.0, 0.3, 0.0, 0.2, 0.5, 0.7]^T to \([\mu_x \ \sigma_x^2 \ \mu_y \ \sigma_y^2]^T = [4 \ 1 \ 1 \ 1]^T\) in 10 seconds,
and then holds this set of task values for 5 seconds, avoiding the circular obstacle at location [3,1]. The desired task trajectory of the platoon is generated using a cubic polynomial interpolation, which is common for robotic tasks [6]. The controllers assume $K = 3$ for mean and 6 for variance, $L = 0.125m$, $K_u = 25$ and there is a fixed rudder limit of $|\delta| \leq \pm \pi/4$. The fully holonomic case is shown in Figure 3 and Figure 4, while the application of the basic controller coupled with (5) is shown in Figure 5 and Figure 6. Note that the units all initially have a heading of $\theta = \pi/2$, aimed along the y-axis. This deployment highlights the problem with nonholonomic units, as the desired initial motion is primarily perpendicular to the units’ initial heading.

![Motion of Platoon Units, 0 Projections](image)

**Figure 3:** Platoon motion with holonomic units.

![Task tracking errors with holonomic units](image)

**Figure 4:** Task tracking errors with holonomic units. Maximum absolute value and integrated absolute error value are shown.
We can see from the examples that the nonholonomic nature of the vessels precludes exact matching of the desired trajectories in task space as generated by the system shown in Section II. Performance degradation is significant, and is expected to degrade further as the dynamic model shown in Equation (4) deviates from the actual system dynamics.

In order to overcome the difficulties associated with the nonholonomic units and improve performance, it was necessary for the control scheme to be refined. The key concept in developing the refined control is that, while the units are underactuated, the swarm as a whole
still possesses more degrees of freedom than task variables. As such, it is possible to somewhat mitigate the nonholonomicity through the use of null space projections.

The procedure developed is an iterative scheme that projects onto the null space of the Jacobian the error between desired and actual heading for a unit at each point in time. Essentially, this generates as a secondary objective the desire for the units to deviate as little as possible from their current state. The component added to the control (which produces the desired velocities) at each iteration is \((I - J^T J)(\dot{q} - \dot{q}^d)\).

The approach is outlined in Figure 7. The process is repeated for each unit whose heading error is above some threshold until a pre-determined number of iterations has been reached.

![Figure 7: Using redundancy to reduce tracking error](image)

Results for a sample case are shown in Figure 9 and Figure 9, where the limit on iterations is 10 and the heading error threshold is \(\pi/8\). Re-projections of error for a unit are not carried out when the unit is closer than three body lengths from an obstacle or another unit, to avoid collisions.

![Figure 8: Platoon motions before projections (a) and after 10 projections (b).](image)
Figure 9: Performance metrics for nonholonomic platoon control as a function of algorithmic iterations. (a) shows maximum absolute error for each task variable, while (b) shows the integrated absolute error.

Each error plot in Figure 9 shows an error metric as a function of the number of iterations of the algorithm. The developed approach works well in reducing overall error and maximum deviation from the desired task variables. Real-time, on-line reactive control is still possible under the iterative re-projection scheme, as the iterations can be coded in closed-form. It is important to note that the improvement in performance is a local phenomenon (as is typical for common controllers of this type), not guaranteed in a global sense. Nevertheless, the sample runs show that this approach is very promising.

Throughout testing, it was seen that the number of iterations is crucial, as is the limit on heading error for re-projection. There are many potential alternative methods for re-projection of error. Examples of these include re-projecting error for only the worst-case unit, re-projecting error only when the re-projection reduces overall system-wide error, etc. Many of these issues will be addressed in forthcoming work.

IV. Implementation Issues

Implementation of a platoon of cooperating surface vessels is a nontrivial task. The principle considerations for fielding such a system include communications and sensing (for localization as well as obstacle avoidance) as well as the hull form, operating envelope and unit-level control (replacing (4) with a more realistic systems).

Communications is a prime consideration for cooperative ASVs. Many strategies for control of such systems exist, from fully decentralized control [7] to low-bandwidth methods [8] to a fully-centralized explicit communications scheme represented in this work. Bandwidth limitation is important underwater, but for ASVs there is little need to limit communications. As such, the proposed scheme assumes that every unit provides its state to a centralized planner. That planner can be off-line or a member of the platoon. Communication among the vessels above water is a straightforward matter and will not be discussed further here.

The information provided by each unit to the planner is generated by sensors. GPS, sonar, computer vision, magnetic compasses, IMUs, etc. are all well suited to the task of localization (determining the location of a unit) and state estimation (for speed, heading, etc.) as well as
environmental feedback. An example of a sensing suite that is appropriate for an ASV is shown in [9].

While selecting sensors is a relatively straightforward matter, it is not so simple to perform the sensor fusion, filtering, estimation and error rejection necessary in a real marine environment. Fortunately, many robust sensing schemes exist for aerial and underwater vehicles. Unfortunately, some of these techniques rely on knowledge of the system dynamics, which is much more challenging for high-speed ASVs than for autonomous vehicles in those other domains, as will be discussed.

One of the most significant issues in the development of a fielded ASV involves selection of an appropriate hull form. The design requirements are much relaxed from standard hull forms, as no human factors are involved. On the other hand, the nature of the tasks that the device will be expected to carry out are also significantly different. Scaled sea states will reach much higher levels for small surface vessels than they will for full-sized craft.

Any hull form selection process must be informed by low-level control. The controller shown in (5) is designed for a simple set of velocity kinematics, but is not appropriate for craft with highly rich dynamics. For example, it is well-known that high-speed, low-weight craft tend to operate in the planing regime, where most of the support for the vehicle comes from dynamic impact forces as opposed to the buoyant force that is common among displacement vessels. Planing craft, such as the deep-V craft shown in Figure 10, make an excellent choice for small surface vessels, as these craft will need to make very long transitions (in terms of body lengths) to achieve any mission of significance. Planing can reduce the amount of power required to reach a target, and maximize speed for a given actuation. Unfortunately, the dynamics of these craft are difficult to model, and closed-form unit-level controllers are challenging to design.

**Figure 10: A deep-V planing hull used as an ASV at USNA.**

Years of research into the dynamics of planing craft, even in very low sea states, has not provided a closed-form set of dynamic equations [10,11]. As such, other tools must be investigated if a closed-form controller is to be designed. Genetic algorithms [12] and neural networks [13] are common methods for achieving modeling and control of complex systems whose dynamics are too rich or poorly understood for the application of traditional control
methods. Genetic algorithms (GAs) use evolutionary methods to develop models and controllers for systems through exhaustive testing and reproduction-based combination of large numbers of candidate ‘individuals.’ Neural networks use a computational intelligence representation of the brain to ‘learn’ appropriate responses and models using training sets of inputs and (desired) outputs.

Unfortunately, both of these methods require extensive testing (on the order of thousands of trials) and neither is guaranteed to converge to an appropriate control or model. Due to the number of required iterations, these methods are best used when a mathematical or simulation model exists for a system, but that model is either numerical in nature or possesses dynamics that are highly nonlinear and time-varying (both of which present challenges for traditional control [14,15]). Thus, using GAs and neural networks for control of planing craft requires a simulation model that is of moderate fidelity and yet low in computational complexity. Systems exist that offer high fidelity simulation of planing craft, but a few seconds of simulated time require many hours of CPU time. As such, these systems are ill-suited to the use of GAs and neural networks.

These facts indicate that control of ASVs that operate in the planing state requires a combination of simulation and testing. Simulations of low to moderate fidelity (if such can be found) can be used to generate candidate controllers using GAs or neural networks. These candidate controllers can be tuned and improved using real hardware experiments, the results of which can be fed back to (potentially) improve the simulations.

V. Conclusions

In this work, we have demonstrated a real-time platoon planning and control architecture suited to nonholonomic units. We have shown that prior work on holonomic systems can be applied to the nonholonomic case, and have demonstrated a local re-projection controller that iteratively improves performance of the nonholonomic platoon over the baseline controller. It is important to note that this result is local, and that improvements are not guaranteed over a complete run. However, the results shown are very promising.

We have shown that a number of considerations are important for implementation of a cooperative platoon of ASVs, but that the primary driving force in these decisions is the hull form and operating paradigm. Planing craft were seen to be one of the most likely choices for ASVs, and the issues associated with such craft for autonomous operation were discussed.

Future work on cooperative ASVs involves refinement of the re-projection algorithm, development of a moderate-fidelity dynamic simulator for planing craft, and application of genetic algorithms and neural networks to the control of planing craft. The platoon control will undoubtedly need to be refined based on the single-unit controllers developed to replace (4) and (5), but the presented algorithms provide an excellent baseline for real implementation.

References


