Autonomous River Navigation using the Hamilton-Jacobi Framework for Underactuated Vehicles

Kevin Weekly, Leah Anderson, Andrew Tinka and Alexandre M. Bayen

Abstract—Motorized floating sensors have distinct advantages over their non-actuated counterparts. A motorized unit can prevent the sensor from washing ashore or heading into dangerous areas, expanding the mission regions in which they can be feasibly operated. In this article, we present a control framework and describe the physically realized system used to prove its effectiveness. The controller uses two minimum-time-to-reach (MTTR) functions—one giving the time to reach the center of the river and one giving the time to reach the shoreline. The MTTR functions are constructed from solutions to Hamilton-Jacobi-Bellman-Isaacs (HJBI) Equations. Contours along these functions are used to define the state transition thresholds for an on-off controller. The first MTTR function is also used to construct the optimal bearing to travel back to the center of the river. We investigate the effectiveness of the controller using a software-in-the-loop (SIL) simulator. Using prototypes built at UC Berkeley, results from a field operational test in the Sacramento–San Joaquin River Delta are then presented to validate the simulation results.

I. INTRODUCTION
For several years, the Floating Sensor Network project at UC Berkeley[1] has been operating floating sensor units, called drifters, in various California waterways. Lately, the third generation of such drifters has been enhanced by a buoyancy control and dual motor system for autonomy. Each drifter contains a Global Positioning System (GPS) receiver and Global System for Mobile Communications (GSM) cell-phone modem, allowing telemetry readings to be reported to an Internet server in real-time. The readings are interpreted as a remote indication of local flow velocity and are integrated with existing forward mathematical models to improve upon flow estimations in real-time. Ultimately, it is hoped that this system can be used to populate a map of water conditions to provide interested parties with up-to-date information on tidal conditions or the spread of a contaminant in a region being monitored. This vision requires long term operation of the drifters, which in turn requires some way of avoiding obstacles such as debris or shallow regions. Hence, our most recent redesign of the sensors introduced a propulsion system intended for collision avoidance, consisting of two propellers in “differential drive” configuration. However, the mechanical motivations of building a motorized vessel are diametrically opposed to those of building a Lagrangian sensor. A small asymmetric profile is desired to reduce drag in the direction of motion; on the other hand, a symmetric and large profile is desired so the unit quickly settles to the local water velocity. We designed the drifters to have a large cylindrical profile in the water. This satisfies their primary role of water tracking at the cost of reduced control authority.

Figure 1 shows a module-level view of the drifter hardware. The main computation unit is a Gumstix Overo single-board computer, containing a Texas Instruments OMAP3530 applications processor running at 720 MHz and 256 MB of RAM. The Overo runs an embedded Linux kernel. Position and velocity information is gathered by a Magellan AC-12 GPS module. Communication with an Internet server are handled through a Motorola G24 GSM module, and short-range communications with the field team are handled through a Digi XBee 802.15.4 radio. Hard real-time tasks, such as processing water
quality sensor signals and controlling the propulsion systems, are performed by an Atmel Atmega128L microcontroller.

In this article, we address the problem of obstacle avoidance in an environmental setting. Controlling in the presence of obstacles is linked to path planning problems [2][3], which sometimes rely on the same theory as this article [4]. Two features of our problem distinguish it from the traditional path-planning problem:

- The drifter is an underactuated system. That is, the unit is in the presence of a river current which is more powerful than the propulsion of the unit. Thus, a successful algorithm must account for the river current and act preemptively to avoid being pushed into an obstacle.
- Our goal is not to reach a single target way-point, rather, the goal of the drifter is to not run into obstacles.

Several approaches can be used to solve these types of problems. Viability-based approaches compute regions of the state-space such as “all points guaranteed to be safe” [5][6][7]. Another approach is to use the level and sub-level sets of solutions to Hamilton-Jacobi-Bellman-Isaacs (HJBI) equations [8][9][10][11][12][13][14]. We chose to use the HJBI framework, which can be used to compute the same sets, in order to use an existing mathematical toolbox [15] to solve these equations numerically.

One additional constraint for environmental monitoring purposes is that the drifter unit’s measurements are less useful when they are under actuation because they are not acting as Lagrangian particles. Thus, we seek to maximize the amount of time the motors are turned off. Off-on control, also known as bang-bang control, is therefore a natural choice. It then remains to choose the transitions and determine the on-state policy of the on-off controller. We show that the solution to a HJBI equation can be used to construct a minimum-time-to-reach (MTTR) function to a given target region. Two such MTTR functions, \( V_{\text{center}} \) and \( V_{\text{shore}} \), are used to determine the transitions of the on-off controller. It is also possible to find the optimal control policy, given the proper MTTR function. It is well known that because the HJBI equation is derived by application of Dynamic Programming (DP) techniques [10], synthesizing these MTTR functions suffers the same curse of dimensionality as other DP methods [16]. Fortunately, for low-dimensional systems the problem is tractable.

II. HAMILTON-JACOBI-BELLMAN-ISAACS BASED OPTIMAL CONTROL

A. Model and Problem Statement

We model the system dynamics of a single drifter as a 2-dimensional single-integrator:

\[
\begin{aligned}
   \dot{x} &= w(x) + a(t) + b(t), \\
   \|a(t)\|_2 &\leq \bar{a}, \\
   \|b(t)\|_2 &\leq \bar{b},
\end{aligned}
\tag{1}
\]

where \( x \in \mathbb{R}^2 \) is a two-dimensional state vector representing the position of the drifter in meters. Table 2 describes the terms we use in the dynamics and their typical maximum magnitudes. Note the absence of a yaw state variable, which is intentionally omitted to reduce the computational burden. We believe the term to be largely irrelevant for longer time scales as the vehicle is highly maneuverable around its vertical axis, owing to the differential drive configuration.

We also define the following sets of functions:

\[
\begin{aligned}
   A &\triangleq \{a(\cdot) : \forall t \quad \|a(t)\|_2 \leq \bar{a}\}, \\
   B &\triangleq \{b(\cdot) : \forall t \quad \|a(t)\|_2 \leq \bar{b}\},
\end{aligned}
\]

and parameterize the trajectory of the system in terms of time, initial condition, and the \( a(\cdot) \) and \( b(\cdot) \) inputs,

\[
x = x(t; x_0, a(\cdot), b(\cdot)).
\]

We are also given a set of undesirable positions, \( D \subset \mathbb{R}^2 \) for which the system is unsafe due to proximity to the banks. Conversely, the complement of this set, \( S \triangleq D^C \), gives the positions for which the system is safe.

Our goal is to find a control input \( a(\cdot) \) such that

\[
\forall t > 0, \forall x_0 \in D^C, \forall b(\cdot) \in B, x(t; x_0, a(\cdot), b(\cdot)) \in D^C,
\tag{2}
\]

which minimizes the time of actuation,

\[
t_{\text{act}} = \int_0^\infty \mathbb{1}_{|a(t)| \neq 0} \, dt.
\]

In this way, we have set up a differential game [17][9] in which the inputs \( a(\cdot) \) and \( b(\cdot) \) work against each other to either satisfy or attempt to violate (2), respectively. As we will see later, \( b(\cdot) \) will always act in the opposite direction of \( a(\cdot) \) with magnitude \( \bar{b} \). Therefore, in this case running this differential game with input constraints \( (a, \bar{b}) \) is equivalent to running a single-player game with input constraints \( (\bar{a} - \bar{b}, 0) \), although this is not true for general differential games [17].

B. Mathematical Solutions

In this section we describe the meaning of a MTTR in the context of HJBI equations. We begin with a target set, \( T \subset \mathbb{R}^n \), giving a set of states we are trying to reach. Consider the construction of a static cost function, \( V(x_0) \):

\[
V(x_0) = \inf_{a(\cdot) \in A} \sup_{b(\cdot) \in B} \left\{ \int_0^{t^*} l(x(t; x_0, a(\cdot), b(\cdot)), a(\cdot), b(\cdot)) \, dt \right\},
\tag{3}
\]

\[
t^*(x_0, a(\cdot), b(\cdot)) = \inf \{ t : x(t; x_0, a(\cdot), b(\cdot)) \in T \},
\tag{4}
\]

Table 2: Description of terms in (1).

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
<th>Bounds (2-norm ball)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w(x) )</td>
<td>River current</td>
<td>0.8m · s(^{-1})</td>
</tr>
<tr>
<td>( a(t) )</td>
<td>Control Input</td>
<td>0.2m · s(^{-1})</td>
</tr>
<tr>
<td>( b(t) )</td>
<td>Disturbance Input</td>
<td>0.05m · s(^{-1})</td>
</tr>
</tbody>
</table>

...
where \( l(\cdot, \cdot, \cdot) : (\mathbb{R}^n, \mathbb{R}^n) \rightarrow \mathbb{R} \) is a Lagrangian cost functional associating a cost for the system to be in a certain state and taking a certain action.

Thus, the interpretation of (4) is that it designates the first time the trajectory, \( x(t; a(\cdot), b(\cdot)) \), enters \( \mathcal{T} \).

Suppose we take \( l(\cdot, \cdot, \cdot) \equiv 1 \), representing a constant accrual of cost until the target set is reached. From (3), the solution becomes

\[
V(x_0) = \inf_{a(\cdot) \in \mathbf{A}} \sup_{b(\cdot) \in \mathbf{B}} \left\{ \int_0^{t^*} \, dt \right\}, \tag{5}
\]

or, more concisely

\[
V(x_0) = t^* (x_0, a^*(\cdot), b^*(\cdot)), \tag{6}
\]

\[
a^*(\cdot) = \arg \inf_{a(\cdot) \in \mathbf{A}} t^*, \tag{7}
\]

\[
b^*(\cdot) = \arg \sup_{b(\cdot) \in \mathbf{B}} t^*,
\]

where \( a^*(\cdot) \) is called the optimal control as it minimizes the accrued cost, and \( b^*(\cdot) \) is the worst case disturbance.

In this case, (6) simply gives the minimum time to reach the target set \( \mathcal{T} \) from \( x_0 \), so we call \( V \) a MTTR function for this system. Note that \( V(x_0) = +\infty \) in the case in which the target set is not reachable from the initial condition \( x_0 \).

We are interested in finding the optimal control, or disturbance, which satisfies (7) and achieves a minimum-time trajectory to, or from, \( \mathcal{T} \). Both can be computed explicitly as a function of the gradient of the MTTR function by the following relations:

\[
a^*(x) = -\overline{a} \frac{\nabla V(x)}{\|\nabla V(x)\|_2}, \quad b^*(x) = \overline{b} \frac{\nabla V(x)}{\|\nabla V(x)\|_2}. \tag{8}
\]

In general, \( V \) is difficult to compute especially for systems such as (1) which have an arbitrary forcing term, \( w \), and an arbitrary target set, \( \mathcal{T} \). We elect to extend the technique found in [15] for finding the MTTR function of a holonomic system: a time-dependent HJBI equation, for which there are known methods to solve [13], is constructed as follows:

\[
0 = \phi_t + \min \left[ 0, \bar{G}(x, \nabla \phi) \right], \quad t \in [0, h], \tag{9}
\]

\[
\bar{G}(x, p) \triangleq \max_{\|a\|_2 \leq \overline{a}} \min_{\|b\|_2 \leq \overline{b}} \left\{ p^T \cdot f(x, a, b) \right\}.
\]

\[
\phi(x, 0) = \begin{cases} -1 & x \in \mathcal{T} \\ 1 & \text{otherwise} \end{cases}, \tag{10}
\]

As shown in [9], \( \mathcal{T} \) can be reached in \( h \) time or less from the set of points

\[
\mathcal{G}(h) = \{ x : \phi(x, h) \leq 0 \}.
\]

The frontier of this set of points as it evolves through time is also the set of points from which \( \mathcal{T} \) can be reached in exactly \( h \) time. The set is related to \( V \) by

\[
\partial \mathcal{G}(h) = \{ x : \phi(x, h) = 0 \}.
\]

Consider a contour, \( \{ x : V(x) = h \} \) of the MTTR function, describing a set of points reachable in exactly \( h \) time units. Comparing this contour with \( \partial \mathcal{G}(h) \), we find that, for a given \( x \), \( V(x) \) is given by the first temporal zero crossing of \( \phi(x, t) \). If such a zero crossing does not exist, this means the system cannot navigate from \( x \) to \( \mathcal{T} \), therefore, \( V(x) = +\infty \).

III. IMPLEMENTATION AND SIMULATION

A. Flow Field Modeling

We use flow field estimates from REALM, a forward simulation model of the Sacramento–San Joaquin Delta [18], for the values of \( w(x) \) required for computation. Due to the tidal nature of the flows in the area, multiple flow field estimates are taken, corresponding to different times of day. The on-board controller will automatically cycle through the policies throughout the course of the day to account for varying water currents.

For simulation purposes, we implement a simple kinematic model of the drifter under the effects of viscous friction and random forces. A REALM flow field is integrated into the environment, and all other physical parameters of the model are designed to most closely resemble the behavior of the physical prototypes we built.
Level-Set
HJBI Solver
Flow Field
Estimate
Target Definition
Policy File
On-Board
Controller
GPS
SIL
Simulator
Plant
Motors
Compass
Heading-hold
controller
file
loaded
offline
region used for
SIL testing

Fig. 5: Controller Implementation Diagram – Orange box indicates online operations; Blue box indicates offline processes.

B. Computation of Control Feedback

The on-board controller requires three two-dimensional arrays to be computed offline and loaded before the experiment:

1) The MTTR function, shown on the left of Figure 4, towards the shore of the river is designated $V_{\text{shore}}$. The $V_{\text{shore}}$ MTTR satisfies (5), where $\bar{a} > 0$ is the maximum current speed which could push the drifter to the shore, no other force disturbs the drift ($\bar{b} = 0$), and $T$ is the left binary image from Figure 3. The function therefore describes how much time the vehicle would move, pushed at speed $\bar{a}$, before crashing on the shore.

2) The MTTR function, shown on right of Figure 4, towards the center of the river is designated as $V_{\text{center}}$. This function also satisfies (5), with $\bar{a} > 0$ is the maximal propulsion of the drifter, $\bar{b}$ is the maximal disturbance, and $T$ is the right binary image from 3. The function therefore describes how much time the vehicle would move, pushed at speed $\bar{a}$, before crashing on the shore.

3) The optimal bearing towards the center of the river, denoted by $\angle^*(x)$, is the angle component of the optimal control given (8), where $V$ in this relation is $V_{\text{center}}$. This function is shown in Figure 6.

These three arrays are combined into a policy file which is loaded onto the drifter prior to the experiment. If we repeat this process for different values of $w$ or $T$, we could generate several such policy files. The drifter is able to select with policy file is used in the on-board controller. For example, the on-board controller could automatically change the policy file over the course of the day to account for periodic tidal flows.

C. On-board Controller

Within this framework, our on-board on-off control system can be encoded by a hybrid automaton $H = (Q, X, R, f, \Sigma, U)$, where $Q$ is the set of discrete modes, $X$ is the domain of continuous states, $R : (Q, \Sigma, X) \rightarrow (Q, X)$ is the transition function, $f_q : X \rightarrow X$ are the continuous dynamics for each mode, $\Sigma$ is the set of discrete events, and $U$ is the set of continuous inputs [19][20]. The resulting automaton definition is:

- $Q = \{q_{\text{drift}}, q_{\text{actuate}}\}$
- $X = \mathbb{R}^2$
- $R$ is as documented in Figure 7
- $f_q$ are shown in Figure 7
- $\Sigma = \{x \in T, x \in D\}$
- $U = A = \{a(t) : \forall t \|a(t)\|_2 \leq \bar{a}\}$

D. Software-In-the-Loop Testing

To assist in development and verification of the drifter’s software systems, a Software-In-the-Loop (SIL) simulator was constructed. Our software architecture uses Unix Domain Sockets as an inter-process communication mechanism. Code is separated into individual modules according to natural functional divisions.

A feature of this modular design choice is that we can write virtual modules which emulate the functionality of physical systems. In this case, we wrote a virtual
GPS module providing coordinates from the dynamics simulator in Section III-A. The dynamics simulator includes a simulated heading-hold controller which exports a virtual motor interface and takes motor commands via this service.

The intent is that the same production code which runs on the actual drifter hardware can be connected to a simulated drifter. This enables running certain experiments without the overhead of a field operation. Part of the motivation for future field tests is to gather control data so that the simulator parameters may be tuned to produce the same trajectories as the physical world.

Several interesting scenarios were simulated to observe the performance of the controller. We are particularly interested in the cases in which the original REALM flow field estimates have a some level of inaccuracy, as may be the case for areas in which flows are poorly known.

For the following simulations, we introduced a viscous force on the drifter towards the east. This could, for example, be due to wind. This was necessary to force the drifter into dangerous regions and cause our control to be activated.

Figure 8 illustrates two trajectories of the simulated drifter. The left is the ideal situation in which REALM provides a reasonably accurate flow field estimate. In the event that we know REALM will be inaccurate for a region, a compromise is to not use any flow field. This case is shown on the right, where the river flow is set to 0 at all points. As apparent from the results shown here, our algorithm was found to be robust against potential inaccuracies in MTTR calculations when desired actuation is nearly orthogonal to external input.

IV. FIELD OPERATIONAL TESTS

A. Operation Layout

A field operational test was carried out targeting the Sacramento - San Joaquin River Delta in California (approximately Latitude 38.03 N, Longitude 121.58 W. See Figure 6). In addition to determining the efficacy of the controller presented in this article, we were also actively logging GPS traces from 9 drifter units. These traces will be used in an inverse modeling algorithm to increase the accuracy of REALM flow field estimates in the future.

The controller described earlier was tested for approximately 5 hours in the river. Two boat teams were responsible for monitoring the drifters and retrieving trapped units if necessary. Retrieved drifters were placed back in the river at safe locations to continue their mission. One goal of the experiment was to determine if the controller presented in this article effectively prevented the drifters from heading into dangerous areas, therefore reducing the number of necessary retrievals.

B. Results

Figure 10 shows data from the field deployment that was gathered by one of the units. It is plotted in a similar fashion to the simulated results presented in Section III-D, though lacking flow field arrows as the true flow field experienced during the experiment is unknown.

This result demonstrates behavior similar to that predicted by the simulation. During deployment, an easterly wind threatened to beach the drifters. Here, this drifter floats north with the river current, but is also being pushed towards the eastern shoreline. Upon crossing the $V_{shore}$ threshold (red contour), it begins to maneuver back to safety. Once it reaches the $V_{center}$ threshold (green contour), it transitions back to drifting without actuation. This is the behavior we expect given the simulator results. Small discrepancies are likely due to inaccuracies in flow magnitude in the REALM flow field used during simulation.

Primarily, the field test validated the use of the simulator as a verification tool. The test also demonstrated that HJBI-based controllers are tractable and effective for physical deployments using the drifter platform.
actively avoid running against the bank of a river. The controller itself should also benefit from the accurate measurement of these parameters due to their influence on a mission. This technique used the solutions of time-dependent Hamilton-Jacobi-Bellman-Isaacs equations to construct minimum time-to-reach functions, used by an on-board on-off controller. We showed the efficacy of the algorithm both in software-in-the-loop simulation and in a field test for the scenario in which the unit must actively avoid running against the bank of a river. Future work will focus on tuning the algorithm to reliably solve general obstacle avoidance scenarios and applications of the theory to multi-vehicle control. We also seek to reduce the computational burden of the technique and move towards a on-line integrated solution.

V. CONCLUSION

In this article, we have described a successful technique for controlling our autonomous floating sensor platforms so that they avoid obstacles and the shoreline during a mission. This technique used the solutions of time-dependent Hamilton-Jacobi-Bellman-Isaacs equations to construct minimum time-to-reach functions, used by an on-board on-off controller. We showed the efficacy of the algorithm both in software-in-the-loop simulation and in a field test for the scenario in which the unit must actively avoid running against the bank of a river.

ACKNOWLEDGEMENTS

Thanks to Ian Mitchell for his Level Set Toolbox and advice on HJBI theory and also Qingfang Wu, Peter Schwartz, and Eli Ateljevich for their work with the REALM simulator.

REFERENCES