Viability-based computation of spatially constrained minimum time trajectories for an autonomous underwater vehicle: implementation and experiments

A. Tinka*[1], S. Diemer[2], L. Madureira[3] , E. B. Marques[3], J. Borges de Sousa[3], R. Martins[3], J. Pinto[3], J. Estrela da Silva[4], A. Sousa[3], P. Saint-Pierre[5], A. M. Bayen[1]

***Abstract*— A viability algorithm is developed to compute the constrained minimum time function for general dynamical systems. The algorithm is instantiated for a specific dynamics (Dubin's vehicle with current constraints), in order to solve numerically the minimum time problem. With the specific dynamics considered, the framework of hybrid systems enable to solve the problem efficiently. The algorithm is implemented in C using epigraphical techniques to reduce the dimension of the problem. The feasibility of this optimum trajectory algorithm is tested in an experiment with a Light Autonomous Underwater Vehicle (LAUV) system. The hydrodynamics of the LAUV are analyzed in order to develop a low-dimension vehicle model. Deployment results from experiments performed in the Sacramento River in California are presented, which show good performance of the algorithm.**

I. Introduction

Reachability analysis seeks to find the set of points that can be reached by trajectories of a dynamical system in the presence of some non-deterministic features. In the context of system verification, the non-determinism may include both control and external uncertainties; in the context of optimal control, only the control input is included. The techniques of reachable set computation can be applied to the problem of optimum trajectory generation, by constructing a value function that can then be used with standard dynamic programming methods. While finding the minimum time to reach function in the absence of constraints is a standard problem, the addition of state constraints makes this problem significantly harder, theoretically and numerically. In this article, we present a method for generating optimum trajectories for a vehicle with a heading-specific planar dynamic, driven by a non-linear, non-parametric field, subject to spatial constraints. The spatial constraints are key, since they make this work significantly different from standard work in optimal control.

The theoretical foundations of the applicability of reachability to non-parametric optimal control problems are in the viscosity solutions to the *Hamilton-Jacobi-Bellman* (HJB) equation [1], [2], which opened the door to numerical approximation of non-differentiable value functions for continuous dynamical systems. However, it is very difficult to incorporate general constraints, such as spatial obstacles, into the Hamilton-Jacobi-Bellman framework. In particular, the implications of the constraints or the nature of the solution (lack of continuity of the value function) prevent the straightforward use of Hamilton-Jacobi tools [3].

In contrast with representational methods, such as levee set methods or Hamilton-Jacobi based methods, viability based approaches do not simplify the representation of reachable sets, but instead approximate them over a regular grid, in a way which can be shown to be mathematically equivalent to Hamilton-Jacobi formulations [1], [2], [4] in some cases, see for example [5] and [6]. In the present work, we present an instantiation of the algorithm described in [7] specifically applied to minimum time. We use a hybrid formulation for the system dynamics as a way of modifying the discretization of the continuous dynamics to the discrete dynamics required by the viability algorithm [8]. While the model of the system dynamics used here is not hybrid in nature, the hybrid formulation of the algorithm using this framework is a convenient way to handle the anisotropy inherent in the Dubins car model we use.

Fig. 1. The Light Autonomous Underwater Vehicle from Porto University used for the implementation of the algorithm

Our approach is applied to the problem of finding minimal-time trajectories for a *Light Autonomous Underwater Vehicle* (LAUV) in a non-parametric velocity field. This example problem features a model derived from the hydrodynamic

*Corresponding author: tinka@berkeley.edu .

[1]Systems Engineering, Department of Civil and Environmental Engineering, University of California at Berkeley.

[2]Ecole Nationale Suprieure des Mines de Paris.

[3]Electrical and Computer Engineering Department, Faculty of Engineering, Porto University.

[4]Electrical Engineering Department, Porto Polytechnic Institute.

[5]Department of Mathematics, Université Paris-Dauphine, Paris.

The Portuguese team was partially funded by the Luso-American Foundation (FLAD).

This research was funded by NSF under grant CNS-0615299.

parameters of a LAUV (shown in Figure 1) and a velocity field calculated for a junction of the Sacramento River in California. An experiment was performed in which we used the optimal trajectories generated by this algorithm to control the LAUV in the Sacramento River. Our modelling approach involves the reduction of a non-linear 6-DOF model of the LAUV to a 2D Dubins car constrained by the water dynamics.

In Section II, we introduce the HJB formulation for minimum time problems and the utility of viability kernels for problems with constraints. In particular, we underline the difference between the viability solution used in this article and the classical viscosity solution to the HJB equation. In Section III we present the LAUV model and discuss simplifications for planar motion, then adapt the model for the hybrid viability kernel formulation of the problem. Section IV describes the LAUV system with special emphasis on the control system. In Section V we describe the experimental setup and the results of the field deployment with the LAUV.

II. Viability Formulation of Minimum Time

A. Viability based formulation of the reachability problem

We consider the following dynamical system:

$$\dot{z}(t) = f(z(t), u(t)) \tag{1}$$

with $u(t) \in \mathcal{U} = \{u : [0, +\infty[\to U, \text{measurable}\}$, and U is a compact metric space. This system has the equivalent set-valued formulation [9]:

$$\dot{z}(t) \in F(z(t)) = \{f(z(t), u)\}_{u \in U} \tag{2}$$

We use the following assumptions:

- f is continuous in u,
- f is Lipschitz continuous in z,
- f is constrained by a linear bound: $\exists c : \max_{u \in U} \|f(z,u)\| \le c(\|z\|, +1)$,
- F is upper semicontinuous.

Definition 2.1: A function $L_{\mathcal{T}}(z_0)$ is the *minimum-time-to-reach* (MTTR) *function* for the dynamical system (1) and a closed, compact target set $\mathcal{T}$ if it satisfies the following condition:

$$L_{\mathcal{T}}(z_0) = \min_{u(\cdot) \in \mathcal{U}} \left\{ \begin{matrix} T : \exists z(\cdot) : \dot{z}(\cdot) = f(z(\cdot), u(\cdot)), \\ z(0) = z_0, z(T) \in \mathcal{T} \end{matrix} \right\} \tag{3}$$

where $\mathcal{U}$ is the proper class of measurable feedbacks.

Definition 2.2: A function $L_{\mathcal{T}}^K(z_0)$ is the *minimum-time-to-reach function with constraints* for the dynamical system (1), a closed, compact constraint set K, and a closed, compact target set $\mathcal{T}$ if it satisfies the following condition:

$$L_{\mathcal{T}}^K(z_0) = \min_{u(\cdot) \in \mathcal{U}} \left\{ \begin{matrix} T : \exists z(\cdot) : \dot{z}(\cdot) = f(z(\cdot), u(\cdot)), \\ z(0) = z_0, z(T) \in \mathcal{T}, \\ \forall \delta \in [0, T], z(\delta) \in K \end{matrix} \right\} \tag{4}$$

From [2] it is known that $L_{\mathcal{T}}(z_0)$ can be characterized as the viscosity solution of the Hamilton-Jacobi-Bellman equation

$$\begin{cases} H(\nabla L_{\mathcal{T}}, z) = -1 \\ \forall z \in \mathcal{T} : L_{\mathcal{T}}(z) = 0 \end{cases} \tag{5}$$

However, if we are interested in the MTTR function *with constraints*, the viscosity solution of the HJB equation is inadequate. In particular, the viscosity solution characterization of the solution (5) no longer holds. In particular, discontinuities inherent to problem (4) force the use of a more general concept of solution, called viability solutions (which are lower semi-continuous). By the definition of $L_{\mathcal{T}}^K(z)$ as $L_{\mathcal{T}}(z)$ with additional constraints, it is easy to see that:

$$L_{\mathcal{T}}(z) \le L_{\mathcal{T}}^K(z) \tag{6}$$

but other than providing this lower bound, the viscosity solution to the HJB equation is not helpful for our problem. One of the goals of this article is to use a characterization of the solution to (2.2) based on viability theory and to implement it algorithmically.

B. Instantiation of the viability technique used

We formalize the technique used, for our specific problem. Let $z := (x, y, \psi) \in Z := \mathbb{R}^2 \times [-\pi, \pi]$ denote the state variable, constrained to remain in a closed subset $K \subset Z$. The target is any closed subset of K denoted $\mathcal{T}$. Let $u \in U(z)$ be a control variable which ranges in a convex set (that may depend on the state z) and let

$$\dot{z}(t) \in F(z(t)) := \{f(z(t), u),\ u \in U(z(t))\} \tag{7}$$

be the set-valued representation of the dynamics. We assume that K is compact and that the set-valued map $F : Z \rightsquigarrow Z$ is convex, compact valued, and has closed graph. ($\rightsquigarrow$ denotes a set-valued function).

We denote by $\mathcal{S}_{F,K,\mathcal{T}}(z_0)$ the set of all solutions to (7) starting from z_0, viable in K, and reaching $\mathcal{T}$ in finite time. Compactness properties of this set can be found in [10].

The capture basin associated with the triple $(F, K, \mathcal{T})$ is defined by:

$$\text{Capt}_F(K, \mathcal{T}) := \{z_0 \in K,\ \exists z(\cdot) \in \mathcal{S}_{F,K,\mathcal{T}}(z_0)\} \tag{8}$$

The objective of this work is to determine a feedback control map $\widetilde{U}(z)$ such that, starting from an initial position $z_0 = (x_0, y_0, \psi_0) \in \text{Capt}_F(K, \mathcal{T})$, any trajectory associated with a selected $\widetilde{u}(z) \in \widetilde{U}(z)$ solution to the system (7) reaches the target in minimal time while remaining in the constraint set K.

C. Hybrid dynamical systems notations used

Hybrid dynamical systems are systems in which the evolutions may either follow a continuous dynamic or jump following an impulse rule when the state reaches a given closed reset set $\mathcal{R} \subset K$.

- The continuous evolutions are governed by the differential inclusion
$$\dot{z}(t) \in F_c(z(t)), \text{ for almost all } \ t \in \mathbb{R}^+ \tag{9}$$
- The impulsive evolutions are governed by the recursive inclusion
$$z^+ \in \Phi_d(z^-) \tag{10}$$

where the set-valued map Φ_d is defined on $\mathcal{R}$ and has compact values with closed graph.

A hybrid system is characterized by the pair (F_c, Φ_d).

Definition 2.3: A *hybrid* solution - or an *impulsive* solution or a *run* - of a system defined by (F_c, Φ_d) is any map $t \to z(t)$ starting from x_0 at the time $t = 0$ and defined on an interval $[0, T]$, (T can be zero, finite or infinite) such that there exists an integer N (N can be zero, finite or infinite), an increasing sequence $(t_n)_{n\in\{0\cdots N\}}$, and a sequence of positions $(z_n)_{n\in\{0\cdots N\}}$ such that $\forall n \in \{0 \cdots N\}$

- either $z_n \in \mathcal{R}$, $z_{n+1} \in \Phi_d(z_n)$, and $t_{n+1} = t_n$,
- or $z(\cdot)$ is a solution to (9) on $[t_n, t_{n+1}[$ such that $z(t_n) = z_n$ and, if $t_{n+1} < +\infty$, $z(t_{n+1}) = z_{n+1} \in K$.

Following [11], [8], we can generalize the viability kernel to hybrid systems:

Definition 2.4: The *hybrid kernel* of K for the hybrid dynamic (F_c, Φ_d) is the subset of initial states belonging to K from which there starts at least one viable hybrid solution. We denote these sets $\mathrm{Hyb}_{(F_c,\Phi_d)}(K)$.

Using an extension of the capture basin algorithm [8], a hybrid kernel can be approximated by a converging sequence of closed sets that are discrete viability kernels of discrete dynamical systems. The extension of viability kernel and capture basin algorithms to hybrid systems, and their convergence, can be found in [8] and [12].

III. Model of LAUV Dynamics

We treat the LAUV as a 3-DOF planar vehicle. The vehicle has a propeller for longitudinal acceleration, and fins for lateral and vertical actuation. However, we decouple the actuators and use the vertical actuation only to maintain a constant depth. We characterize the state of the system with three variables: x, y for earth-fixed East and North coordinates, and ψ for earth-fixed heading. For this application it is acceptable to assume that the effect of currents can be captured by superposition; in other words, the velocity of the surrounding water can simply be added to the velocity of the vehicle due to actuation.

The LAUV is modeled as a Dubins' car [13] with limited turn rate in a non-parametric velocity field:

$$\begin{aligned} \dot{x} &= V\cos\psi + v_{\text{cx}}(x, y) \\ \dot{y} &= V\sin\psi + v_{\text{cy}}(x, y) \\ \dot{\psi} &\in [-r_{\max}, r_{\max}] \end{aligned} \tag{11}$$

To find the turn rate limit, a more general 6-DOF model of the LAUV [14] was used, with the known parameters of the fin surface area and angular range. The fin servo dynamics are significantly faster than the vehicle dynamics and so were disregarded. The maximum speed of the LAUV in still water was experimentally determined to be $2m/s$, and at that speed, the maximum turn rate was $0.5\text{rad}/s$. By fitting the hydrodynamic model to these values, the turn rate/speed relationship could be estimated for lower speeds. The value $2m/s$ was judged to be too fast for the planned experiments. We used an intermediate value of $V = 1.0m/s$ and $r_{\max} = 0.2\text{rad}/s$.

A. Implemented model

The model (11) is a continuous-time, continuous-space, differential inclusion. In order to apply the viability algorithm, we must first choose an appropriate discretization, creating a discrete-time, discrete-space viability kernel problem while respecting the consistency bounds defined in [15] and [7]. The standard viability notation for the step size is h for the step size in the spatial dimensions and ρ for the time step. There is a consistency bound on h and ρ:

$$\rho \geq \sqrt{\frac{2h}{ML}} \tag{12}$$

where $M = \sup_{z\in K}\sup_{y\in F(z)} \|y\|$ is the norm of fastest system velocity, and L is the Lipschitz constant of the dynamics satisfying $\forall z, z' \in K, F(z') \subset F(z) + L\|z - z'\|\mathcal{B}_0$, $\mathcal{B}_0$ denoting the unit ball in the state space.

The formulations of the viability approximation algorithm in [15] and [7] assume an isotropic problem. In this problem, however, there is a significant difference between the spatial position dimensions (x, y) and the heading dimension ψ. The isotropic formulation uses a single step size h; there is no guideline to suggest what the relationship between the step sizes of (x, y), in meters, and the step size of ψ, in radians, should be. In order to treat these two categories of dimension separately, we adopted a hybrid formulation, essentially "breaking out" the ψ dimension and discretizing it separately. We are using this formulation as a means of achieving greater control over the discretization, not because the system is fundamentally hybrid.

In our hybrid formulation, we use h for the step size in (x, y), ρ for the time step, and $d\psi = \frac{2\pi}{N_\psi}$ for the heading step size. Note that $N_\psi \in \mathbb{Z}^+$. In the hybrid case, the M and L constants are evaluated on the continuous part associated with the set-valued right hand side map F_c. Since $\dot{\psi} = 0$, this amounts to evaluating them from the dynamics of the only state variable (x, y). To this standard bound we add an additional condition: that for any pairs of distinct headings ψ_1 and ψ_2, the one-time-step reachable sets are distinct. If

$$\Gamma_h(x, y, \psi) := \begin{bmatrix} \rho(v\cos\psi + v_{\text{cx}}) \cap X_h \\ \rho(v\sin\psi + v_{\text{cy}}) \cap Y_h \end{bmatrix}$$

then $\psi_1 \neq \psi_2 \Rightarrow d(\Gamma_h(x, y, \psi_1), \Gamma_h(x, y, \psi_2)) \geq h$, where $\mathcal{B}_0$ is the ball of radius 1 in (x, y), and X_h and Y_h are the points forming the grid in (x, y) with spacing h.

This represents, in some way, a matching condition between the "granularity" of ψ and (x, y): if it is not met, we could say that computational effort is being wasted on redundant ψ modes. We address this with the following new bound, which relates the h, ρ, and N_ψ steps to the farthest possible point reachable in one step (disregarding the $(v_{\text{cx}}, v_{\text{cy}})$ terms, because they do not rotate):

$$2\pi\frac{\rho v}{h} \geq N_\psi \tag{13}$$

By discretizing ψ separately, we freed it from the standard consistency bound in (12). The new bound (13) is the constraint that keeps the overall discretization consistent.

In order to respect the limited turn rate required by (11), our hybrid model must include a restriction on the rate at which mode transitions happen. One common method [8] is to add a "dwell variable" that counts down as time moves forward, and only permit mode transitions when the dwell variable is smaller than zero. Adding variables, of course, increases the dimension of the problem. If the "time steps per mode change" is non-integer, another source of error appears, since rounding the dwell variable will result in either losing or gaining some turn rate performance in the trajectories. We could eliminate this error by manipulating the time step so that the number of mode transitions per time step is an integer; we could eliminate the problem of the dwell variable entirely if we set the steps so that one transition were permitted per time step.

$$r_{\max}\rho\frac{N_\psi}{2\pi} \approx 1 \tag{14}$$

Conditions (13) and (14), when combined to form a bound on ρ, result in

$$\rho \geq \sqrt{\frac{h}{r_{\max}v}} \tag{15}$$

Depending on v, $r_{\max}$, and L, only one of conditions (12) and (15) will be necessary. In this problem setup, it is condition (12). Our final selections for step sizes are $h = 0.2\text{m}, \rho = 1.3\text{s}, N_\psi = 24$.

Once the step sizes have been chosen, the discretization of (11) follows the standard viability kernel algorithm [15]:

Continuous dynamics:

$$\begin{cases} x^{n+1} \in \big(x^n + \rho v \cos\psi^n + v_{\text{cx}}(x^n, y^n) + h\mathcal{B}_0\big) \cap X_h \\ y^{n+1} \in \big(y^n + \rho v \sin\psi^n + v_{\text{cy}}(x^n, y^n) + h\mathcal{B}_0\big) \cap Y_h \\ \psi^{n+1} = \psi^n \\ \tau^{n+1} = \tau^n - \rho \end{cases}$$

Discrete dynamics:

$$\begin{cases} x^+ = x^- \\ y^+ = y^- \\ \psi^+ \in \big(\psi^- + \{-1, 0, 1\}\big) \mod N_\psi \\ \tau^+ = \tau^- \end{cases} \tag{16}$$

This is the model implemented in the viability algorithm described as Algorithm 1. Note that in the "continuous" (non-hybrid) dynamics, we now consider each of the state variables as a sequence instead of a function of time ($x^1, x^2, x^3, \cdots$ instead of $x(t)$, etc). Also note the dilation of the x and y successors by a ball of radius h; this is a standard feature of the viability kernel algorithm.

B. Algorithm

Algorithm 1 below describes the principal viability loop, which iterates over a discrete grid until a fixed point of the value function is reached. Points in the grid are labelled "Target", "Viable", or "NonViable". Initially all the points are marked either Target or NonViable; the algorithm progressively changes some of the NonViable points to Viable. At the end of the procedure, the code converges to a fixed set of labels.

Algorithm 1 The hybridized, suspended viability algorithm for minimal time to reach target

```
mark points in K as NonViable
mark points in 𝒯 as Target
repeat
  set numChanges to 0
  for all point p = (x, y, ψ) in K do
    if hybrid switches allowed for (x, y, ψ) then
      set S to list of successors of p in mode ψ as well
      as in possible switched modes ψ1, ψ2, ···
    else
      set S to list of successors of p in mode ψ
    end if
    find pB, the best of the successors of p in S (see
    Algorithm 2)
    if pB is marked Target then
      mark p as Viable
      set BestTime(p) to ρ
      set BestSucc(p) to pB
    else if pB is marked Viable then
      mark p as Viable
      set BestTime(p) to BestTime(pB)+ρ
      set BestSucc(p) to pB
    else
      mark p as NonViable
      set BestTime(p) to Null
      set BestSucc(p) to Null
    end if
    if p was changed then
      increment numChanges
    end if
  end for
until numChanges is 0
```

Algorithm 2 describes the subroutine used to find the best successor point p_B given a starting point p and a list of candidate successors. Each candidate point p_C is checked against the "best point so far" using the following criteria:

1) Target points are better than Viable points are better than NonViable points
2) Smaller time-to-target is better
3) Smaller heading change is better

Compared to the original viability kernel algorithm presented in [15], this algorithm is focused on computing viability kernels that are epigraphs of a value function, using the functional approach given in [5] for the minimal time-to-reach problem. This approach is also used in numerical finance, to compute the evaluation function for financial instruments [16].

Algorithm 2 Selection algorithm for "best successor"

Input: point $p = (x, y, \psi)$, successor set S
Output: best successor point p_B
  set $p_B = (x_B, y_B, \psi_B)$ to Null
  for all point $p_C = (x_C, y_C, \psi_C)$ in S **do**
    if p_C is marked Target **then**
      if p_B is marked Target **then**
        if $\left|\psi_C - \psi\right| < \left|\psi_B - \psi\right|$ **then**
          set p_B to p_C
        end if
      else
        set p_B to p_C
      end if
    else if p_C is marked Viable **then**
      if p_B is not marked Target **then**
        if BestTime(p_C)<BestTime(p_B) **then**
          set p_B to p_C
        else if BestTime(p_C)=BestTime(p_B) **then**
          if $\left|\psi_C - \psi\right| < \left|\psi_B - \psi\right|$ **then**
            set p_B to p_C
          end if
        end if
      end if
    end if
  end for

IV. Light Autonomous Underwater Vehicle

The LAUV is a small (110*cm* $\times$ 16*cm*) low-cost submarine with a maximum operating depth of 50m for oceanographic and environmental surveys designed and built at Porto University (Figure 1). It has one propeller and three control fins. The onboard navigation suite includes a Microstrain low-cost inertial measurement unit, a Honeywell depth sensor, a GPS unit and a transponder for acoustic positioning (GPS does not work underwater). This transponder is also used for recieving basic commands from the operator. The LAUV has a WiFi interface and GSM module for communications at the surface. The LAUV has a miniaturized computer system running modular controllers on a real-time Linux kernel.

LAUV's onboard software system is called DUNE, is written in C++ and was designed for autonomous vehicle control. Device drivers, controllers and services are implemented as dedicated tasks. The task manager has the responsibility of creating new tasks, assigning priority to tasks, and terminating running tasks in a safe and ordered manner. The acoustic navigation system uses the well known *long baseline navigation* (LBL) technique. LBL navigation requires the deployment of a least two transponders in the water in the area of operation. The vehicle interrogates each transponder with a given frequency, and each transponder replies with another frequency. The time elapsed between the interrogation and the reply is proportional to the distance to each transponder. The position of the vehicle is computed from the distances to the two transponders, and from the depth measurements.

The command and control system consists of one or more, laptop computers running the *Neptus* command and control framework [17] on top of the Seaware publish/subscribe framework [18]. In its basic version, we operate with one laptop connected to a wireless router and to an acoustic transponder through a serial cable. The transponder is deployed in the water to listen to the acoustic exchanges between the LAUV and the other two transponders; this information makes it possible to track the position of the LAUV. In addition, it can send an *abort* command to the LAUV. The LAUV answers by surfacing.

The LAUV control algorithms assume decoupled modes of operation [19]. We consider two types of control: lateral control and longitudinal control. This is acceptable in practice if we avoid changing the vehicle's heading while changing its depth. The input to the lateral control PID is a heading reference while the input to the longitudinal control is a depth reference. The longitudinal velocity command (surge) is assumed to be piecewise constant during mission execution. The low-level control system is composed of a periodic discrete-time control law for the heading controller and for the depth controller. The current implementation of the control system depends solely on the earth-fixed coordinates, which are provided by the navigation algorithm. The LAUV's navigation algorithm is based on an *Extended Kalman Filter* (EKF), to take into account the non-linearities in the model [20].

V. Implementation

Once the viability algorithm has returned the data, finding the optimum trajectory for a particular starting point is done by sequentially connecting the current point to its optimum successor. The trajectory is the list of these optimum successor points. Planning the trajectory of the LAUV during the experiment was done by selecting a starting point and heading, then finding the optimum trajectory using the precomputed results of the viability kernel algorithm, then converting the trajectory into a set of waypoints spaced $10m$ apart. The LAUV was then tasked to go to those waypoints using its waypoint tracking algorithm.

The general algorithm and hybrid model framework described above was instantiated for a specific problem: finding optimal trajectories for a LAUV in a river environment where the currents were significant compared to the forward velocity of the vehicle. The performance of the LAUV in terms of a simple current-driven Dubins car model was determined by developing a hydrodynamic model of the vehicle, matching it to known performance parameters, and inferring the parameters of the simplified model. The algorithm was implemented in C and used to generate optimal trajectories, which were then executed by the LAUV in the river.

The deployment area was a $400m \times 300m$ rectangle containing the junction of the Sacramento River and the Georgiana Slough in California, USA. Under normal conditions, water flows from North to South, at speeds ranging from $0.5m/s$ to $1.5m/s$. Bathymetric data for the region is available from the USGS. The channel depth drops steeply from the bank, and is deeper than $2m$ at all points away from the shore, so operations can be safely conducted as long as the LAUV does not come within $5m$ of the shore.

Figure 2 presents the *Neptus* mission planning GUI for the experiment. *lsts1* and *lsts2* represent the two transponders used for acoustic navigation, and *basestation* represents the command and control system, which consisted of 3 laptops. *lsts1* and *lsts2* were deployed at approximately 2ms from the bottom of the channel at depths of respectively 8m and 10m. The third transponder was deployed in the water, in the proximity of the *basestation* (the transponder is connected to one laptop with a serial cable). This transponder monitors the acoustic signals in the water.

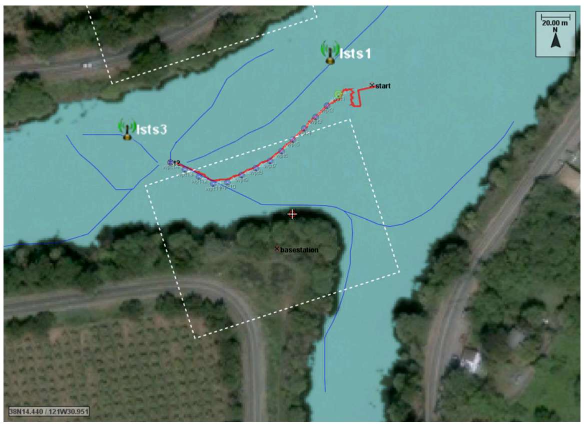

Fig. 2. Optimal trajectory of the submarine using the proposed algorithm, in the Georgianna Slough.

The experimental region is tidally forced, which gives it time-dependent behavior. This makes it an attractive region for hydrodynamic study, but adds complexity to the estimation of the currents. We performed our calculations and experiments for a stable portion of the tidal behavior, which gave us a two hour window where the currents could reasonably be modeled as time-independent. The velocity field used in the optimum trajectory calculations was generated by a forward simulation of the 2D Saint-Venant shallow water equations using FLUENT [21][22].

In the ideal case, the development of optimum trajectories based on the velocity field would happen in real-time. In the present implementation, the process of generating a FLUENT forward simulation from the boundary conditions, and then calculating optimum trajectories based on this current field, takes several hours. It was therefore necessary to develop the optimum trajectories off-line, and then perform the experiment when the tidal conditions were similar to the pre-calculated velocity field. By choosing a stable portion of the tidal behavior, the window of operations was 2 hours per day. Nevertheless, the currents during the experiments were never exactly the same as those in the pre-generated simulation. This is an unavoidable source of error, which can only be mitigated by real-time boundary condition gathering, and much faster computation of the velocity field and optimum trajectories.

The optimal control algorithm takes as inputs the LAUV model parameters, the geometry of the junction and the flow field. The geometry of the junction is used to generate a 2D grid of points, labeled either as "river" or "land" points. The "river" points are used to build $\bar{K}$, the set of allowed points for the viability algorithm. As described in section II-A, $\bar{K}$ is a 4D set; all possible ψ values are permitted, and τ can be in $[0, +\infty[$ as mentioned. Any trajectory constructed from the viability algorithm will respect the constraint set $\bar{K}$, and therefore will be contained in the allowed river area.

The viability algorithm builds a minimum time function for the target, but due to the way in which this function is built, it is efficient to immediately output optimal trajectories. In other words, there is no need to perform a dynamic programming optimization on the minimum time function; these results are immediately accessible during the algorithm run. Therefore, the algorithm output is a set of trajectories; each trajectory is a series of (x, y, ψ) points. The optimal control can be inferred from the sequence of ψ values.

The algorithm was run using an x, y grid of 1m, a time step of $1.2s$, and a ψ spacing of $15°$. Trajectories were generated for starting points on a $10m \times 10m$ grid, for all 24 possible initial headings. Figure 5 shows a sample of the trajectories generated by the viability algorithm, including the one used for the experiment. The target is a 5m radius circle. Notice that some of the trajectories join into a common approach to the target. This figure only shows the (x, y) coordinates of the trajectories; the direction of travel is a combination of the thrust vector of the LAUV (in the heading ψ) and the action of the current.

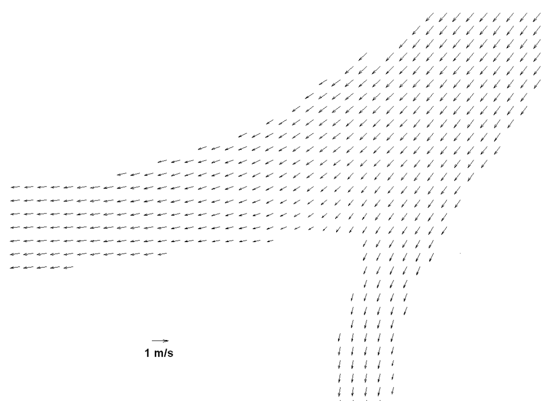

Fig. 3. Current map (decimated).

VI. Results

A. Computational results

Figures 4, 5, and 6 show the results of the minimal time computation using the viability algorithm for the LAUV model. The effect of the anisotropic current on the minimal time function is clearly visible in Figure 4, and the presence of discontinuities in the action map is apparent in Figure 6.

In Figure 5, the trajectories labelled "A" show how a change in initial heading ($270°$ versus $255°$) can result in

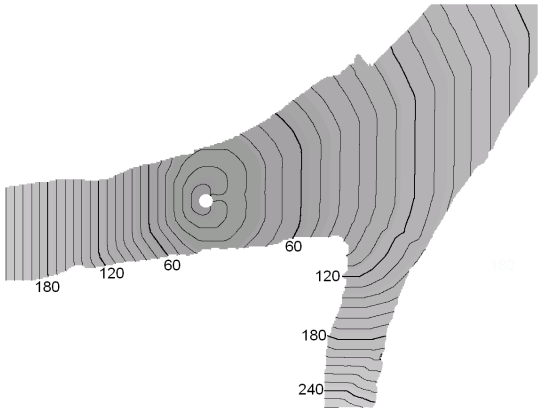

Fig. 4. Isochrones of the minimal time to target for various starting positions, and initial heading East. Numbered contours give time to target in seconds.

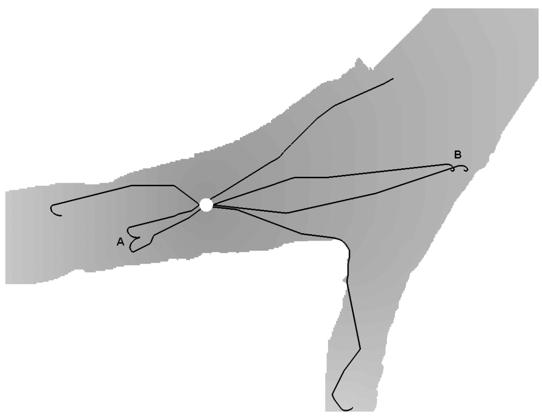

Fig. 5. Sample of generated trajectories. Pair "A" have the same starting position, headings differ by 15°. Pair "B" have the same starting heading, positions separated by 10m.

dramatically different actions. The trajectories labelled "B" show how two starting points, separated by 10m, with the same initial heading ($90°$), can take very different paths to the target. Once the step sizes have been chosen, the minimal time trajectory problem can be thought of as a shortest-path problem on a directed graph. Condition (14) effectively limits the number of edges leaving each node. The $\mathcal{B}_0$ dilation results in up to 4 possible successors for each heading, although these may overlap; the result is that every node has $4-12$ outbound edges. Roughly 38% of the points in the (x, y) grid are in the river; although not all of these will be in the viability kernel, we can approximate the number of points in the kernel by $38\% \times 2000 \times 1500 \times 24 \approx$ 27M points and 108M $-$ 234M edges. Future work will investigate how graph theoretic approaches could improve the efficiency of the viability algorithm.

B. Experimental results

We ran several experiments with the trajectory data sets provided by the optimal control algorithm. This took place in second week of November 07. The qualitative behavior of the LAUV did not change significantly across several experiments, and showed good agreement with predicted results. Figure 2 displays the results for the experiment concerning the optimal trajectory. In this experiment the LAUV was deployed at the location labeled *start*, where it drifted with the current while waiting for the *startmission* command (this means that the initial position and orientation are not exactly known in advance). The mission consisted of three phases: diving, moving to the first waypoint and executing the optimal trajectory. The optimal trajectory consisted of the sequence of 15 waypoints *wp1, ..., wp15*, depicted in the figure along with the trajectory of the LAUV.

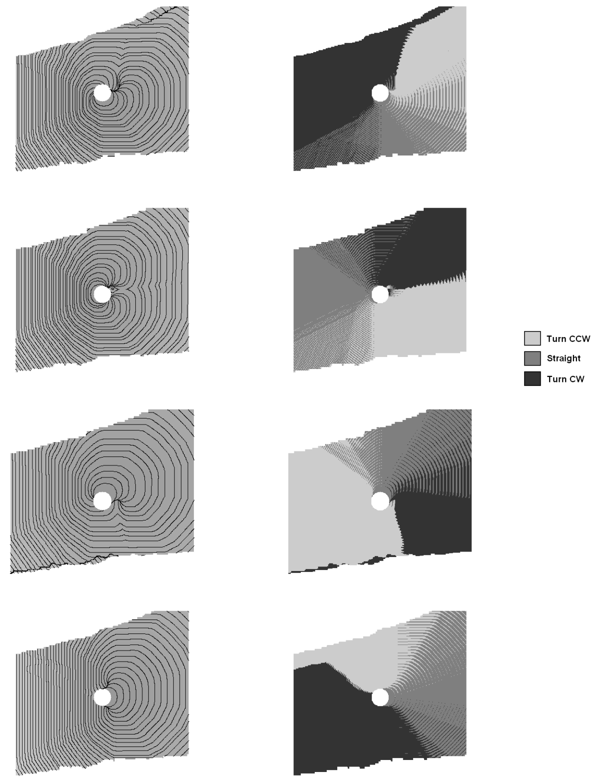

Fig. 6. Minimal time function (left) and feedback map (right) for a 100m $\times$ 100m region around the target, and initial heading (from top) North, East, South, West.

Figure 7 shows the deviation between the planned and actual trajectory. The first segment of the trajectory connects the starting point to the first waypoint. The large deviation results from the fact that the LAUV has to reach the first waypoint with the right orientation after diving from the starting point. The LAUV rapidly converges to the desired optimal trajectory starting at the first waypoint. The tracking error is less than $2m$. Observe that the typical navigation error for this type of navigation scheme is in the order of $1m$. The navigation scheme also explains the jumps in tracking error: with each new acoustic fix there is a, possibly discontinuous, correction to the position of the LAUV. The trend in the deviation may be explained by a mismatch between the predicted and actual currents.

Figure 8 depicts the heading reference and the heading

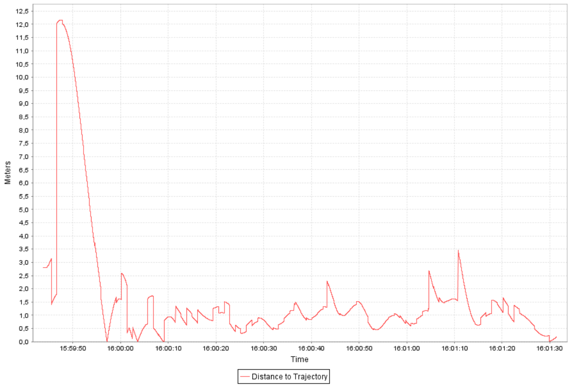

Fig. 7. Deviation between planned and actual trajectory.

estimate. The transients concerning the first segment of the overall trajectory are easily identified in the figure.

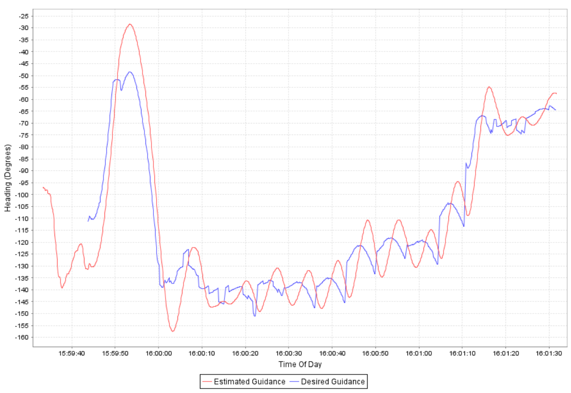

Fig. 8. Heading: reference and estimated.

VII. Conclusions

The algorithm described in this article was successfully used to compute feedback maps and trajectories that were executed by an LAUV in a series of experiments in the Sacramento River. The optimal trajectories were executed by the LAUV using waypoint control. The behavior of the LAUV under this control strategy closely matched the expected behavior. We have demonstrated the validity of the viability approach to planning trajectories for a vehicle in a non-parametric velocity field. The general formulation of the vehicle dynamics make this method applicable to many scenarios where an oriented vehicle must find trajectories in a plane while being driven by complex non-linear dynamics and while subject to constraints.

Acknowledgments

The authors gratefully acknowledge the contributions of Paulo Dias and Rui Gonçalves from Porto University to the development of the UAV software tools used in this experiment. The authors wish to thank Professor Jean-Pierre Aubin for his guidance on viability theory and Professor Ian Mitchell for his comments on the discretization conditions. The authors are indebted to Issam Strub, Qingfang Wu, Julie Percelay, and Jean-Severin Deckers for their work on the forward simulations of the shallow water equations that provided the water current data. The code used in this study relies on technologies and algorithms developed by the company VIMADES.

Special thanks to Dee Korbel, Jennifer Stone and Beata Najman for helping the UC Berkeley team get the LAUV through US customs.

References

[1] M. Crandall and P. Lions, "Viscosity solutions of Hamilton-Jacobi equations," *Transactions of the American Mathematical Society*, vol. 277, p. 1, 1983.

[2] M. Bardi and I. Capuzzo-Dolcetta, *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*. Boston, MA: Birkhäuser, 1997.

[3] R. Moitie and N. Seube, "Guidance algorithm for uuvs obstacle avoidance systems," in *OCEANS 2000 MTS/IEEE Conference and Exhibition*, vol. 3. IEEE, 2000, pp. 1853–1860.

[4] H. Frankowska, "Lower semicontinuous solutions of Hamilton-Jacobi-Bellman equations," in *SIAM Journal of Control and Optimization*, vol. 31, Jan. 1993, pp. 257–272.

[5] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre, "Optimal times for constrained nonlinear control problems without local controllability," *Applied Mathematics and Optimization*, vol. 36, pp. 21–42, Jan. 1997.

[6] I. Mitchell, A. Bayen, and C. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control*, vol. 50, no. 7, pp. 947–957, July 2005.

[7] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre, "Set-valued numerical analysis for optimal control and differential games," in *Stochastic and Differential Games*, M. Bardi, T. E. S. Raghavan, and T. Parthasarathy, Eds. Springer, 1999, pp. 177–247.

[8] P. Saint-Pierre, "Approximation of viability kernels and capture basins for hybrid systems," in *European Control Conf.*, Porto, Portugal, Sep. 2001, pp. 2776–2783.

[9] J.-P. Aubin, *Viability Theory*. Boston, MA: Birkhäuser, 1991.

[10] J.-P. Aubin and H. Frankowska, *Set-Valued Analysis*. Boston, MA: Birkhäuser, 1990.

[11] J.-P. Aubin, "Impulse differential inclusions and hybrid systems: A viability approach," Lecture Notes, University of California at Berkeley, 1999.

[12] E. Cruck and P. Saint-Pierre, "Nonlinear impulse target problems under state constraint: A numerical analysis based on viability theory," *Set-Valued Analysis*, vol. 12,4, pp. 383–416, 2004.

[13] L. E. Dubins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, pp. 497–516, 1957.

[14] T. Fossen, *Guidance and Control of Ocean Vehicles*. John Wiley and Sons, Inc., New York, 1994.

[15] P. Saint-Pierre, "Approximation of the viability kernel," *Applied Mathematics and Optimization*, vol. 29, pp. 187–209, Mar. 1994.

[16] D. Pujal and P. Saint-Pierre, "L'algorithme du bassin de capture appliqué pour évaluer des options européennes, américaines ou exotiques," *Revue de l'Association Française de Finance, Paris*, 2004.

[17] P. S. Dias, R. Gomes, J. Pinto, S. Fraga, G. Gonçalves, J. B. de Sousa, and F. L. Pereira, "Mission planning and specification in the neptus framework," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, IEEE, Ed., 2006, pp. 3220–3225.

[18] E. R. B. Marques, G. M. Gonçalves, and J. B. de Sousa, "The use of real-time publish-subscribe middleware in networked vehicle systems," in *Proceedings of the First IFAC Workshop on Multivehicle Systems (MVS'06)*, IFAC, Ed., 2006.

[19] A. R. Girard, J. B. de Sousa, and J. E. Silva, "Autopilots for underwater vehicles: Dynamics, configurations, and control," in *Proceedings of the IEEE Oceans 2007 Conference*, IEEE, Ed. IEEE, 2007, pp. 1–6.

[20] A. Matos, N. Cruz, A. Martins, and F. L. Pereira, "Development and implementation of a low-cost lbl navigation system for an auv," in *Proceedings of the MTS/IEEE Oceans 99 Conference*. IEEE, 1999.

[21] I. S. Strub and A. M. Bayen, "Lagrangian data assimilation algorithms for shallow water flows," in preparation.

[22] Fluent, *Fluent 6.2 UDF Manual*, Fluent Inc., Centerra Resource Park, 10 Cavendish Court, Lebanon, NH 03766, USA, 2005.