4

Real-Time Optimal Guidance and Obstacle Avoidance for UMVs

Oleg A. Yakimenko and Sean P. Kragelund
Naval Postgraduate School Monterey, CA
USA

1. Introduction

The single most important near-term technical challenge of developing an autonomous capability for unmanned vehicles is to assess and respond appropriately to near-field objects in the path of travel. For unmanned aerial vehicles (UAVs), that near field may extend to several nautical miles in all directions, whereas for unmanned ground and maritime vehicles, the near field may only encompass a few dozen yards directly ahead of the vehicle. Nevertheless, when developing obstacle avoidance (OA) manoeuvres it is often necessary to implement a degree of deliberative planning beyond simply altering the vehicle's trajectory in a reactive fashion. For unmanned maritime vehicles (UMVs) the ability to generate near-optimal OA trajectories in real time is especially important when conducting sidescan sonar surveys in cluttered environments (e.g., a kelp forest or coral reef), operations in restricted waterways (e.g., rivers or harbours), or performing feature-based, terrain-relative navigation, to name a few. For example, a primary objective of sidescan sonar surveys is 100% area coverage while avoiding damage to the survey vehicle. Ideally, a real-time trajectory generator should minimize deviations from the pre-planned survey geometry yet also allow the vehicle to retarget areas missed due to previous OA manoeuvres. Similarly, for operations in restricted waterways, effective OA trajectories should incorporate all known information about the environment including terrain, bathymetry, water currents, etc.

In the general case, this OA capability should be incorporated into an onboard planner or trajectory generator computing optimal (or near-optimal) feasible trajectories faster than in real time. For unmanned undersea vehicles (UUVs) the planner should be capable of generating full, three-dimensional (3D) trajectories, however some applications may require limiting the planner's output to two-dimensions (2D) for vertical-plane or horizontal-plane operating modes. For unmanned surface vehicles (USVs) the latter case is the only mode of operations.

Consider a typical hardware setup consisting of a UUV augmented with an autopilot (Fig.1). The autopilot not only stabilizes the overall system, but also enables vehicle control at a higher hierarchical level than simply changing a throttle setting $\delta_T(t)$, or deflecting stern plane $\delta_s(t)$ or rudder $\delta_r(t)$ angles.

In Fig.1, $\mathbf{x}^{WP}$, $\mathbf{y}^{WP}$, $\mathbf{z}^{WP}$ are the vectors defining x, y, and z coordinates of some points in the local tangent (North-East-Down (NED)) plane for waypoint navigation. Alternatively a

typical autopilot may also accept some reference flight-path angle $\gamma(t)$ (or altitude/depth) command and heading $\Psi(t)$ (or yaw angle $\psi(t)$), respectively. The motion sensors, accelerometers, and rate gyros measure the components of inertial acceleration, $\ddot{x}_I(t)$, $\ddot{y}_I(t)$ and $\ddot{z}_I(t)$, and angular velocity – roll rate $p(t)$, pitch rate $q(t)$, and yaw rate $r(t)$.

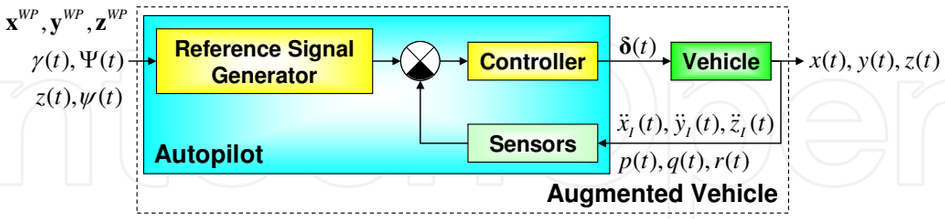


Fig. 1. A. UUV augmented with an autopilot

A trajectory generator would consider an augmented UUV as a new plant (Fig.2) and provide this plant with the necessary inputs based on the mission objectives (final destination, time of arrival, measure of performance, etc.). Moreover, the reference signals, $\gamma(t)$ and $\Psi(t)$, are to be computed dynamically (once every few seconds) to account for disturbances (currents, etc.) and newly detected obstacles.

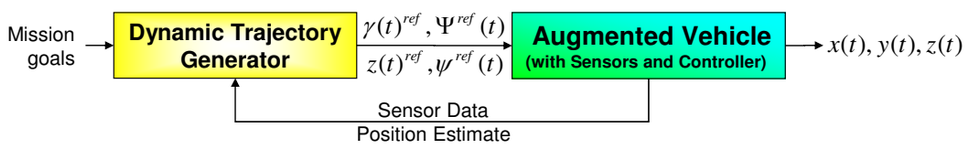


Fig. 2. Providing an augmented UUV with a reference trajectory

Ideally, the trajectory generator software should also produce the control inputs $\boldsymbol{\delta}^{ref}(t)$ corresponding to the feasible reference trajectory (Fig.3) (Basset et al., 2008). This enhanced setup assures that the inner-loop controller deals only with small errors. (Of course this setup is only viable if the autopilot accepts these direct actuator inputs.)

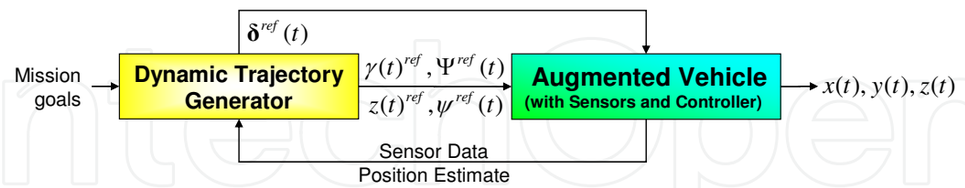


Fig. 3. Providing an augmented UUV with the reference trajectory and reference controls

The goal of this chapter is to present the dynamic trajectory generator developed at the Naval Postgraduate School (NPS) for the UMVs of the Center for Autonomous Vehicle Research (CAVR) and show how the OA framework is built upon it. Specifically, Section 2 formulates a general feasible trajectory generation problem, followed by Section 3, which introduces the general ideas behind the proposed framework for solving this problem that utilizes the inverse dynamics in the virtual domain (IDVD) method. Section 4 considers simplifications that follow from reducing the general spatial problem to two planar subcases. Section 5 describes the REMUS UUV and SeaFox USV and their forward looking

sonar (FLS) systems employed for OA research at NPS. Section 6 addresses path-following considerations and practical implementation details for tracking nonlinear trajectories with conventional vehicle autopilots. Section 7 presents results from computer simulations and field experiments for several different scenarios which benefit from faster-than-real-time computation of near-optimal trajectories.

2. Problem formulation

Let us consider the most general case and formulate an optimization problem for computing collision-free trajectories in 3D (it can always be reduced to a 2D problem by eliminating two states). We will be searching within a set of admissible trajectories described by the state vector

$$\mathbf{z}(t) = \left[x(t), y(t), z(t), u(t), v(t), w(t)\right]^T \in S,\ S = \left\{\mathbf{z}(t) \in Z^6 \subset E^6\right\},\ t \in \left[t_0, t_f\right] \tag{1}$$

where the components of the velocity vector – surge u, sway v, and heave w, defined in the body frame $\{b\}$ – are added to the UUV NED coordinates x, y and z ($z = 0$ at the surface and increases in magnitude with depth). While many UUVs are typically programmed to operate at a constant altitude above the ocean floor, it is still preferable to generate vertical trajectories in the NED local tangent plane because the water surface is a more reliable absolute reference datum than a possibly uneven sea floor. In general, however, it is a trivial matter to convert the resulting depth trajectory *z(t)* to an altitude trajectory *h(t)* for vehicles equipped with both altitude and depth sensors. Section 6.2 describes such practical considerations in detail.

The admissible trajectories should satisfy the set of ordinary differential equations describing the UUV kinematics

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix} = {}^{u}_{b}R \begin{bmatrix} u(t) \\ v(t) \\ w(t) \end{bmatrix} \tag{2}$$

In (2) ${}^{u}_{b}R$ is the rotation matrix from the body frame $\{b\}$ to the NED frame $\{u\}$, defined using two Euler angles, pitch $\theta(t)$ and yaw $\psi(t)$, and neglecting a roll angle as

$${}^{u}_{b}R(t) = \begin{bmatrix} \cos\psi(t)\cos\theta(t) & -\sin\psi(t) & \cos\psi(t)\sin\theta(t) \\ \sin\psi(t)\cos\theta(t) & \cos\psi(t) & \sin\psi(t)\sin\theta(t) \\ -\sin\theta(t) & 0 & \cos\theta(t) \end{bmatrix} \tag{3}$$

Although we are not going to exploit it in this study, admissible trajectories should also obey UUV dynamic equations describing translational and rotational motion. This means that the following linearized system holds for the vector $\boldsymbol{\varsigma}(t)$, which includes speed components u, v, w (being a part of our state vector $\mathbf{z}(t)$) and angular rates p, q, r:

$$\dot{\boldsymbol{\varsigma}}(t) = \mathbf{A}\boldsymbol{\varsigma}(t) + \mathbf{B}\boldsymbol{\delta}(t) \tag{4}$$

Here $\mathbf{A}$ and $\mathbf{B}$ are the state and control matrices and $\boldsymbol{\delta} = [\delta_T, \delta_s, \delta_r]^T$ is the control vector (Healey, 2004).

Next, the admissible trajectories (1) should satisfy the initial and terminal conditions

$$\mathbf{z}(t_0) = \mathbf{z}_0\,,\ \mathbf{z}(t_f) = \mathbf{z}_f \tag{5}$$

Finally, certain constraints should be obeyed by the state variables, controls and their derivatives. For example, in the case of a UUV these can include obvious constraints on the UUV depth:

$$z_{\min} \le z(t) \le z_{\max} \tag{6}$$

where $z_{\max}(x,y)$ describes a programmed operational depth limit. For vehicles programmed to operate at some nominal altitude above the sea floor, the $z_{\max}(x,y)$ constraint can be converted into a minimum altitude $h_{\min}(x,y)$ constraint as described in Section 6.2.

A 3D OA requirement can be formulated as

$$[x(t); y(t); z(t)] \cap \Re = 0 \tag{7}$$

where $\Re$ is the set of all known obstacle locations. The constraints are usually imposed not only on the controls themselves $|\boldsymbol{\delta}| \le \boldsymbol{\delta}_{\max}$ but on their time derivatives as well $|\dot{\boldsymbol{\delta}}| \le \dot{\boldsymbol{\delta}}_{\max}$ to account for actuator dynamics. Knowing the system's dynamics (4) (or simply complying with the autopilot specifications), these latter constraints can be elevated to the level of the reference signals, for instance

$$|\theta(t)| \le \theta_{\max} \quad \text{and} \quad |\dot{\psi}(t)| \le \dot{\psi}_{\max} \tag{8}$$

The objective is to find the best trajectory and corresponding control inputs that minimize some performance index *J*. Typical performance index specifications include: i) minimizing time of the manoeuvre $t_f - t_0$, ii) minimizing the distance travelled to avoid the obstacle(s), and iii) minimizing control effort or energy consumption. In addition, the performance index may include some "exotic" constraints dictated by a sensor payload. For example, a UUV may require vehicle trajectories which point a fixed FLS at specified terrain features or minimize vehicle pitch motion in order to maintain level, horizontal flight along a survey track line for accurate synthetic aperture sonar imagery (Horner et al., 2009).

Before we proceed with the development of the control algorithm, it should be noted that quite often the UUV surge velocity is assumed to be constant, $u(t) \equiv U_0$, to provide enough control authority in two other channels. This uniquely defines a throttle setting $\delta_T(t)$, and leaves only two control inputs, $\delta_s(t)$ and $\delta_r(t)$, for altering the vehicle's trajectory. It also allows us to consider matrices **A** and **B** in (4) to be constant (time- and states-independent). If this assumption is not required, inverting kinematic and dynamic equations will differ slightly from the examples presented in the next section. However, the general ideas of the proposed approach remain unchanged.

3. Real-time near-optimal guidance

For the dynamic trajectory generator shown in Figs. 2 and 3, it is advocated to use the direct-method-based IDVD (Yakimenko, 2000). The primary rationale is that this approach features

several important properties required for real-time implementations: i) the boundary conditions including high-order derivatives are satisfied *a priori*; ii) the resulting control commands are smooth and physically realizable, iii) the method is very robust and is not sensitive to small variations in input parameters, iv) any compound performance index can be used during optimization. Moreover, this specific method uses only a few variable parameters, thus ensuring that the iterative process during optimization converges very fast compared to other direct methods. The IDVD-based trajectory generator consists of several blocks. The goal of the first block, to be discussed next, is to produce a candidate trajectory, which satisfies the boundary conditions.

3.1 Generating a candidate trajectory

Again, consider the most general case of a UUV operating in 3D (as opposed to a USV). Suppose that each coordinate x_i, $i = 1,2,3$ of the candidate UUV trajectory is represented as a polynomial of degree M of some abstract argument τ, the virtual arc

$$x_i(\tau) = \sum_{k=0}^{M} a_{ik}\tau^k , \tag{9}$$

(for simplicity of notation we assume $x_1(\tau) \equiv x(\tau)$, $x_2(\tau) \equiv y(\tau)$ and $x_3(\tau) \equiv z(\tau)$). In general, analytic expressions for the trajectory coordinates can be constructed from any combination of basis functions to produce a rich variety of candidate trajectories. For example, a combination of monomials and trigonometric functions was utilized in (Yakimenko & Slegers, 2009).

As discussed in (Yakimenko, 2000; Horner & Yakimenko, 2007) the degree M is determined by the number of boundary conditions that must be satisfied. Specifically, it should be greater or equal to the number of preset boundary conditions but one. In general the desired trajectory includes constraints on the initial and final position, velocity and acceleration: x_{i0}, x_{if}, x'_{i0}, x'_{if}, x''_{i0}, x''_{if}. In this case the minimal order of polynomials (9) is 5, because all coefficients in (9) will be uniquely defined by these boundary conditions, leaving the "length" of the virtual arc τ_f as the only varied parameter. For more flexibility in the candidate trajectory, additional varied parameters can be obtained by increasing the order of the polynomials (9). For instance, using seventh-order polynomials will introduce two more varied parameters for each coordinate expression. Rather than varying two coefficients in these extended polynomials directly, we will vary the initial and final jerk, x'''_{i0} and x'''_{if}, respectively. In this case, coefficients a_{ik} in (9) can be determined by solving the obvious system of linear algebraic equations equating polynomials (9) to x_{i0}, x_{if}, x'_{i0}, x'_{if}, x''_{i0}, x''_{if}, x'''_{i0} and x'''_{if} at two endpoints ($\tau = 0$ and $\tau = \tau_f$) (Yakimenko, 2000, 2008).

By construction, the boundary conditions (5) will be satisfied unconditionally for any value of the final arc τ_f. However, varying τ_f will alter the shape of the candidate trajectory. Figure 4 demonstrates a simple example whereby a UUV operating 2m above the sea floor at 1.5m/s must perform a pop-up manoeuvre to avoid some obstacle. Even with a single varied parameter, changing the value of τ_f allows the UUV to avoid obstacles of different heights. Similar trajectories could be produced solely in the horizontal plane or in all three dimensions. It should be pointed out that even at this stage infeasible candidate trajectories

will be ruled out. (In Fig.4 the trajectory requiring the UUV to jump out of the water is infeasible because it violates the constraint (6).)

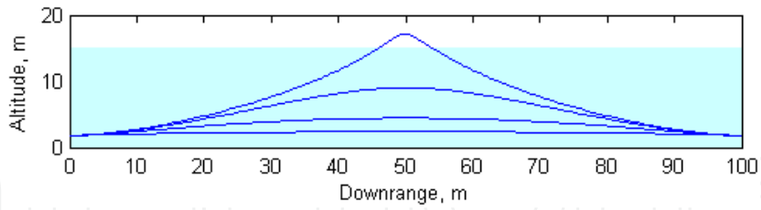


Fig. 4. Varying the candidate trajectory while changing the value of τ_f

With six free parameters, which in our case are components of the initial and final jerk (x_{i0}''' and x_{if}''', $i=1,2,3$) the trajectory generator can change the overall shape of the trajectory even further. To this end, Fig.5 illustrates candidate trajectories for a UUV avoiding a 10m obstacle located between its initial and final points. These trajectories were generated by varying just two components of the jerk, x_{30}''' and x_{3f}''', and minimizing τ_f. This additional flexibility can produce trajectories which satisfy operational constraints (6), as well as OA constraints (7).

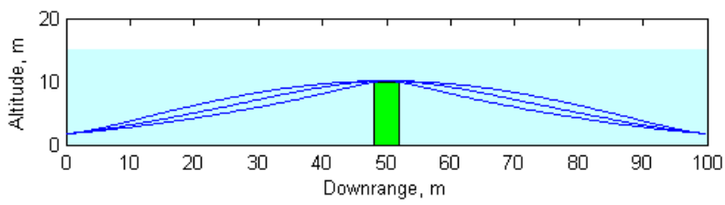


Fig. 5. Candidate trajectories obtained by varying the terminal jerks

The selection of a specific trajectory will be based upon whether the trajectory is feasible (satisfies constraints (8)) and if so, whether it assures the minimum value of the performance index calculated using the values of the vehicle states (and controls) along that trajectory. As an example, Fig.6 presents collision-free solutions for two different locations of a 10m-tall obstacle when five varied parameters, x_{10}''', x_{1f}''', x_{30}''', x_{3f}''' and τ_f, are optimized to assure feasible minimum-path-length trajectories

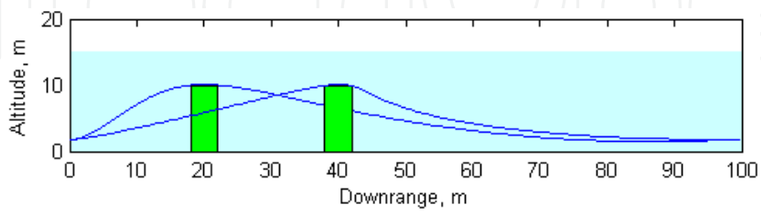


Fig. 6. Examples of minimum-path-length trajectories

Now, let us address the reason for choosing some abstract parameter τ as an argument for the reference functions (9) rather than time or path length, which are commonly used. Assume for a moment that $\tau \equiv t$. In this case, once we determine the trajectory we unambiguously define a speed profile along this trajectory as well, since

$$V(t) = \sqrt{u(t)^2 + v(t)^2 + w(t)^2} = \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2 + \dot{z}(t)^2} \tag{10}$$

Obviously, we cannot allow this to happen because we want to vary the speed profile independently. With the abstract argument τ this becomes possible via introduction of a speed factor λ such that

$$\lambda(\tau) = \frac{d\tau}{dt}. \tag{11}$$

Now instead of (10) we have

$$V(\tau) = \lambda(\tau)\sqrt{x'(\tau)^2 + y'(\tau)^2 + z'(\tau)^2} \tag{12}$$

and by varying $\lambda(\tau)$ we can achieve any desired speed profile.

The capability to satisfy higher-order derivatives at the trajectory endpoints, specifically at the initial point, allows continuous regeneration of the trajectory to accommodate sudden changes like newly discovered obstacles. As an example, Fig.7 demonstrates a scenario whereby a UUV executing an OA manoeuvre discovers a second obstacle and must generate a new trajectory beginning with the current vehicle states and control values (up to the second-order derivatives of the states). The suggested approach enables this type of continuous trajectory generation and ensures smooth, non-shock transitions.

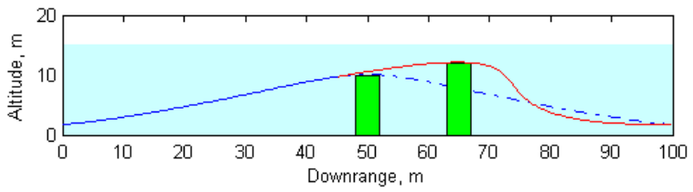


Fig. 7. Example of dynamic trajectory reconfiguration

3.2 Inverse dynamics

The second key block inside the dynamic trajectory generator in Figs. 2 and 3 accepts the candidate trajectory as an input and computes the components of the state vector and control signals required to follow it. In this way we can ensure that each candidate trajectory does not violate any constraints (including those of (8)).

First, using the following relation for any parameter ζ,

$$\dot{\zeta}(\tau) = \frac{d\zeta}{d\tau}\frac{d\tau}{dt} = \zeta'(\tau)\lambda(\tau) \tag{13}$$

we convert kinematic equations (2) into the τ domain

$$\lambda(\tau)\begin{bmatrix} x'(\tau) \\ y'(\tau) \\ z'(\tau) \end{bmatrix} = {}^{u}_{b}R \begin{bmatrix} U_0 \\ v(\tau) \\ w(\tau) \end{bmatrix} \tag{14}$$

Next, we assume the pitch angle to be small enough to let $\sin\theta(t) \approx 0$ and $\cos\theta(t) \approx 1$, so that the rotation matrix (3) becomes

$$ {}^{u}_{b}R(\tau) = \begin{bmatrix} \cos\psi(\tau) & -\sin\psi(\tau) & 0 \\ \sin\psi(\tau) & \cos\psi(\tau) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{15} $$

While this step is not required, it simplifies the expressions in the following development. Inverting (14) via the rotation matrix (15) yields

$$ \begin{bmatrix} U_0 \\ v \\ w \end{bmatrix} = \lambda \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \tag{16} $$

Hereafter each variable's explicit dependence on τ will be omitted from the notation.
Now the three equations of system (16) must be resolved with respect to three unknown parameters, v, w and ψ. While the last one readily yields

$$ w = \lambda z' \tag{17} $$

the first two require more rigorous analysis.
Consider Fig.8. Geometrically, a scalar product of two vectors on the right-hand-side of the first equation in (16) represents the length of the longest side of the shaded rectangle. Similarly, the second equation expresses the length of the shortest side of this rectangle. From here it follows that the square of the length of the diagonal vector can be expressed in two ways: $v^2\lambda^{-2} + U_0^2\lambda^{-2} = x'^2 + y'^2$. This yields

$$ v = \sqrt{\lambda^2(x'^2 + y'^2) - U_0^2} \tag{18} $$

From the same figure it follows that

$$ \psi = \Psi - \tan^{-1}\frac{v\lambda^{-1}}{U_0\lambda^{-1}} = \Psi - \tan^{-1}\frac{v}{U_0}, \quad \Psi = \tan^{-1}\frac{y'}{x'} \tag{19} $$

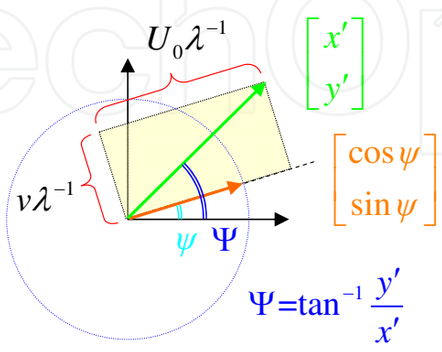


Fig. 8. Kinematics of horizontal plane parameters

Now, using these inverted kinematic equations, we can check whether each candidate trajectory obeys the constraints imposed on it (constraints (8)).

3.3 Discretization

We proceed with computing the remaining states along the reference trajectory over a fixed set of N points (for instance, N=100) spaced evenly along the virtual arc $[0;\tau_f]$ with the interval

$$\Delta\tau = \tau_f (N-1)^{-1} \tag{20}$$

so that

$$\tau_j = \tau_{j-1} + \Delta\tau\,,\quad j = 2,...,N\,,\ (\tau_1 = 0) \tag{21}$$

In order to determine coefficients for polynomials (9) we will have to guess on the values of the varied parameters τ_f, x'''_{i0}, x'''_{i0}, x'''_{if}, and x'''_{if}. These guesses will be used along with the known or desired boundary conditions x_{i0}, x'_{i0}, x''_{i0}, x_{if}, x'_{if}, and x''_{if}. The boundary conditions on coordinates x_{i0} and x_{if} come directly from (5). According to (14), the given boundary conditions on surge, sway, and heave velocities define the first-order time derivatives of the coordinates as

$$\begin{bmatrix} x'_{0;f} \\ y'_{0;f} \\ z'_{0;f} \end{bmatrix} = \lambda_{0;f}^{-1}\, {}^{u}_{b}R_{0;f} \begin{bmatrix} U_0 \\ v_{0;f} \\ w_{0;f} \end{bmatrix} \tag{22}$$

They also define the initial and final pitch and yaw angles used to compute ${}^{u}_{b}R_{0;f}$ in (22) as

$$\theta_{0;f} = \gamma_0 + \tan^{-1}\frac{-w_{0;f}}{\sqrt{U_0^2 + v_{0;f}^2}} \text{ and } \psi_{0;f} = \Psi_0 - \tan^{-1}\frac{v_{0;f}}{U_0} \tag{23}$$

In equation (22) we may use any value for the initial and final speed factor λ, for example, $\lambda_{0;f} = 1$. This value simply scales the virtual domain; the higher the values for λ, the larger the values for τ_f. This follows directly from equations (11) and (12) that $\lambda_{0;f}\tau_f^{-1} = U_0 s_f^{-1}$, where s_f is the physical path length.

Finally, initial values for the second-order derivatives are provided by the UUV motion sensors (see Figs. 1-3) (after conversion to the τ domain), while final values for the second-order derivatives are usually set to zero for a smooth arrival at the final point. Having an analytical representation of the candidate trajectory (9) defines the values of x_{ij}, and x'_{ij}, $i = 1,2,3$, $j = 1,...,N$.

Now, for each node $j = 1,...,N$ we compute

$$\lambda_j = \Delta\tau\Delta t_{j-1}^{-1} \tag{24}$$

where

$$\Delta t_{j-1} = \frac{\sqrt{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2 + (z_j - z_{j-1})^2}}{\sqrt{U_0^2 + v_{j-1}^2 + w_{j-1}^2}} \tag{25}$$

and then use (17)-(19) to compute w, v, ψ and Ψ at each timestamp. The vertical plane parameters, flight path angle γ and pitch angle θ, can be computed using the following relations:

$$\gamma_j = \tan^{-1}\frac{-z'_j}{\sqrt{x'^2_j + y'^2_j}}, \quad \theta_j \approx \gamma_j + \tan^{-1}\frac{-w_j}{\sqrt{U_0^2 + v_j^2}} \tag{26}$$

In order to check the yaw rate constraints (8) we must first numerically differentiate the expression for Ψ in (19).

3.4 Optimization

When all parameters (states and controls) are computed in each of N points, we can compute the performance index J and the penalty function. For example, we can combine constraints (6) and (8) into the joint penalty

$$\Delta = \left[k^{z_{\min}}, k^{z_{\max}}, k^{\theta}, k^{\dot{\psi}}\right] \begin{bmatrix} \min\limits_j (0; z_j - z_{\min})^2 \\ \max\limits_j (0; z_j - z_{\max})^2 \\ \max\limits_j (0; |\theta_j| - \theta_{\max})^2 \\ \max\limits_j (0; |\dot{\psi}_j| - \dot{\psi}_{\max})^2 \end{bmatrix} \tag{27}$$

with $k^{z_{\min}}$, $k^{z_{\max}}$, k^{θ} and $k^{\dot{\psi}}$ being scaling (weighting) coefficients. Now the problem can be solved using numerical methods such as the built-in *fmincon* function in the Mathworks' MATLAB development environment. Alternatively, by combining the performance index J with the joint penalty Δ we may exploit MATLAB's non-gradient *fminsearch* function. For real-time applications, however, the authors prefer to use a more robust optimization routine based on the gradient-free Hooke-Jeeves pattern search algorithm (Yakimenko, 2011).

4. Planar cases

This section presents two simplified cases for a vehicle manoeuvring exclusively in either the horizontal or vertical plane.

4.1 Horizontal plane guidance

For the case of a UUV manoeuvring in the horizontal plane or a USV, the computational procedure is simplified. The trajectory is represented by only two reference polynomials for coordinates x_1 and x_2. Hence, we end up having only five varied parameters, which are: τ_f, x'''_{10}, x'''_{20}, x'''_{1f} and x'''_{2f}. The remaining kinematic formulas are identical to those presented above with $z \equiv 0$, $z' \equiv 0$ and $\gamma \equiv 0$. Figure 9 shows an example of a planar scenario in which a USV has to compute a new trajectory twice. First, after detecting an obstacle blocking its original path, a new trajectory is generated to steer right and pass safely in front of the object (dotted line). Second, while executing the first avoidance manoeuvre the USV detects that the object has moved south into its path. It therefore

generates a new trajectory to steer left and pass safely behind the object's stern. The complete trajectory is shown as a solid line.

4.2 Vertical plane guidance

For the case of a UUV manoeuvring in the vertical plane, the 3D algorithm can be reduced to the 2D case in a manner similar to the horizontal case. Specifically, using five varied parameters, τ_f, x'''_{10}, x'''_{30}, x'''_{1f} and x'''_{3f}, we can develop reference trajectories for x_1 and x_3, and then use the same general equations developed in Section 3, assuming $y \equiv 0$, $y' \equiv 0$, and $\Psi \equiv 0$.

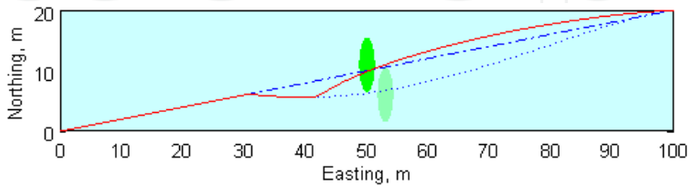


Fig. 9. Moving obstacle avoidance in a horizontal plane

Alternatively, we can use a single reference polynomial to approximate just x_3 and then integrate the third equation of (4) to get the heave velocity w. That allows computation of the time period Δt_{j-1} using

$$\Delta t_{j-1} = (z_j - z_{j-1}) w_{j-1}^{-1} \tag{28}$$

instead of (25).

Another way of dealing with vertical plane manoeuvres is to invert the dynamic equations (4) (Horner & Yakimenko, 2007). After developing the reference functions for two coordinates, x_1 and x_3, the stern plane δ_s control input is computed subject to five variable parameters: τ_f, x'''_{10}, x'''_{30}, x'''_{1f}, and x'''_{3f}.

In this case, the corresponding time period Δt_{j-1} is computed similarly to (28):

$$\Delta t_{j-1} = t_j - t_{j-1} = \frac{z_j - z_{j-1}}{w_{j-1}\cos\theta_{j-1} + u_0 \sin\theta_{j-1}} \approx \frac{z_j - z_{j-1}}{w_{j-1}} \tag{29}$$

and the heave velocity is calculated using the third equation of system (4) as

$$\begin{aligned} x'_{j-1} &= \lambda_{j-1}^{-1}(w_{j-1}\sin\theta_{j-1} + U_0\cos\theta_{j-1}) & x_j &= x_{j-1} + \Delta\tau x'_{j-1} \\ w'_{j-1} &= \lambda_{j-1}^{-1}\left(\mathrm{A}_{33}w_{j-1} + \mathrm{A}_{35}q_{j-1} + \mathrm{B}_{32}\delta_{s;j-1}\right) & w_j &= w_{j-1} + \Delta\tau w'_{j-1} \end{aligned} \tag{30}$$

The next step involves computing the pitch angle, pitch rate and pitch acceleration as

$$\theta_j = \cos^{-1}\left(\lambda_j \frac{u_0 x'_j + w_j z'_j}{w_j^2 + U_0^2}\right),\ q_j = \dot{\theta}_j \approx \frac{\theta_j - \theta_{j-1}}{\Delta t_{j-1}},\ \dot{q}_j = \ddot{\theta}_j \approx \frac{q_j - q_{j-1}}{\Delta t_{j-1}} \tag{31}$$

Finally, we can compute the dive plane deflection required to follow the trajectory using the 5th equation of system (4) as

$$\delta_{s;j} = (\dot{q}_j - A_{53} w_j - A_{55} q_j) B_{52}^{-1} \tag{32}$$

In this case the last two terms in the joint penalty Δ, similar to that of (27) but developed for the new controls, enforce $|\delta_s| \le \delta_{s\max}$ and $|\dot{\delta}_s| \le \dot{\delta}_{s\max}$.

5. Test vehicles and sensing architecture

The preceding trajectory generation framework has been implemented on several UMVs. Before presenting simulated and experimental results with specific vehicle platforms at sea, we first introduce two such vehicles in use at CAVR - the REMUS UUV and SeaFox USV. Both vehicles utilize FLS to detect and localize obstacles in their environment and employ the suggested direct method to generate real-time OA trajectories. This section provides system-level descriptions of both platforms including their sensors, and proposes a way of building the OA framework on top of the trajectory generation framework, i.e. enhancing the architecture of Figs. 2 and 3 even further.

5.1 REMUS UUV and SeaFox USV

Remote Environmental Monitoring UnitS (REMUS) are UUVs developed by Woods Hole Oceanographic Institute and sold commercially by Hydroid, LLC (Hydroid, 2011). The NPS CAVR owns and operates two REMUS 100 vehicles in support of various navy-sponsored research programs. The REMUS 100 is a modular, 0.2m diameter UUV designed for operations in coastal environments up to 100m deep. Typical configurations measure less than 1.6m in length and weigh less than 45kg, allowing the entire system to be easily transported worldwide and deployed by a two-man team (Fig.10a). Designed primarily for hydrographic surveys, REMUS comes equipped with sidescan sonar and sensors for collecting oceanographic data such as conductivity, temperature, depth or optical backscatter. The REMUS 100 system navigates using a pair of external transponders for long baseline acoustic localization or ultra-short baseline terminal homing, as well as an Acoustic Doppler Current Profiler/Doppler Velocity Log (ADCP/DVL). The ADCP/DVL measures vehicle altitude, ground- or water-relative vehicle velocity, and current velocity profiles in body-fixed coordinates.

a)  b) 

Fig. 10. NPS REMUS 100 UUV (a) and FLS arrays (b)

To support ongoing CAVR research into sonar-based OA, terrain-relative navigation, and multi-vehicle operations in cluttered environments, each NPS REMUS vehicle has been modified to incorporate a FLS, multi-beam bathymetric sonar, acoustic communications modem, navigation-grade inertial measurement system, and fore/aft horizontal/vertical cross-body thrusters for hovering or precise manoeuvring. (Figure 10b provides a close up of the NPS REMUS FLS arrays with nose cap removed.) To maximize the REMUS system's utility as a research platform, Hydroid developed the RECON communications interface so that sensor and computer payloads can interact with the REMUS autopilot. Using this interface, NPS payloads receive vehicle sensor data and generate autopilot commands based on NPS sonar processing, trajectory generation, and path-following algorithms.

The SeaFox USV was designed and manufactured by Northwind Marine (Seattle, WA) as a remote-controlled platform for intelligence, surveillance, reconnaissance, anti-terrorist force protection, and maritime interdiction operations (Northwind Marine, 2011). SeaFox is a 4.88m long, aluminium, rigid-hulled inflatable boat with a 1.75m beam; 0.25m draft; fold-down communications mast; and fully-enclosed electronics and engine compartments. SeaFox's water jet propulsion system is powered by a JP5-fueled, 185-HP V-6 Mercury Racing engine, and can deliver a top speed of 74km/h. Standard sensing systems include three daylight and three low light navigation cameras for remote operation, as well as twin daylight and infrared gyro-stabilized camera turrets for video surveillance. All video is accessible over a wireless network via two onboard video servers.

The NPS SeaFox was modified to enable fully-autonomous operations by integrating a payload computer with the primary autopilot (Fig.11). Meanwhile, the original remote control link was retained to provide an emergency stop function. NPS algorithms running on the payload computer generate rudder and throttle commands that are sent directly to the SeaFox autopilot. Recent navigational upgrades include a satellite compass that uses differential Global Positioning System (GPS) navigation service for accurate heading information, a tactical-grade inertial measurement unit for precise attitude estimation, and an optional ADCP/DVL for water velocity measurements. To support ongoing CAVR research into autonomous riverine navigation, the NPS SeaFox was further upgraded to deploy a retractable, pole-mounted FLS system for underwater obstacle detection and avoidance (Gadre et al., 2009). Figure 12 shows the SeaFox USV operating on a river with its sonar system deployed below the waterline.

5.2 Sonar system

The NPS REMUS and SeaFox vehicles rely on FLS to detect and localize obstacles in their environment. Both platforms utilize commercial blazed array sonar systems manufactured by BlueView Technologies (BlueView Technologies, 2011). These sonar systems comprise one or more pairs of arrays grouped into sonar "heads." Each sonar head generates a 2D cross-sectional image of the water column in polar coordinates, typically plotted as the image plane's field of view angle vs. range. Due to the sonar arrays' beam width, the resulting FLS imagery has a 12-degree out-of-plane ambiguity. The REMUS FLS system consists of two fixed sonar heads, which provide a 90-degree horizontal field of view (FOV) and a 45-degree vertical FOV. Similarly, the SeaFox FLS system is comprised of twin sonar heads mounted on port and starboard pan/tilt actuators, providing each side with a 45-degree FOV image at an adjustable mounting orientation that can be swept through the water column for increased sensor coverage.

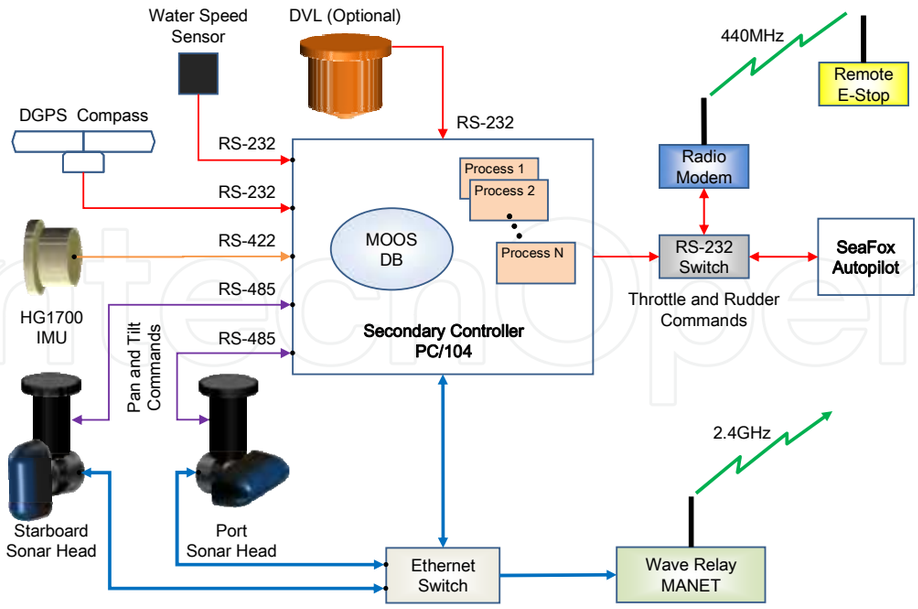


Fig. 11. SeaFox sensors and control architecture

Fig. 12. SeaFox USV navigating on the Sacramento River near Rio Vista, CA

5.3 Obstacle avoidance framework

The proposed OA framework built into the architecture of Figs. 2 and 3 is shown in Fig.13. It consists of an environmental map, a planning module, a localization module, sensors and actuators (Horner & Yakimenko, 2007). The environmental map can include *a priori* knowledge, such as the positions of charted underwater obstacles, but also incorporate unexpected threats discovered by sonar. The positions of all obstacles are eventually resolved in the vehicle-centred coordinate frame with the help of the localization module. The planning module is responsible for generating collision-free trajectories the vehicle should follow. This reference trajectory, possibly with reference controls, is then used to excite actuators.

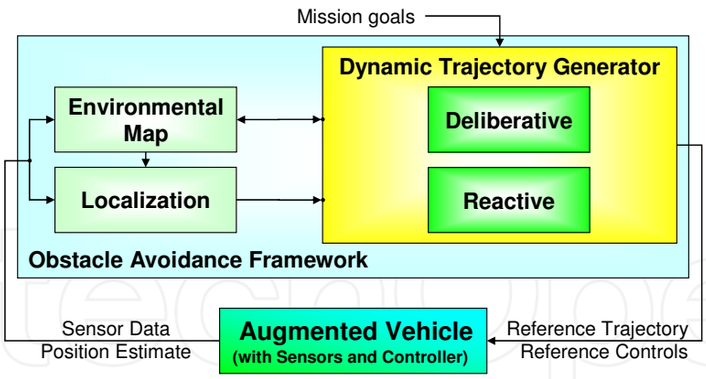


Fig. 13. Components of the NPS OA framework

The proposed OA framework supports both deliberative and reactive obstacle avoidance behaviours. Deliberative OA involves the ability to generate and follow a trajectory that avoids all known obstacles between an arbitrary start location and some desired goal location, whereas reactive OA involves the ability to avoid any previously unknown obstacles detected while following this trajectory. Since the sonar system continuously resamples the environment, this reactive behaviour can be achieved by a deliberative planner as long as i) it executes fast enough to incorporate all new obstacle information from the sonar, and ii) it generates feasible trajectories which begin with the vehicle's current state vector. Specifically, since the REMUS and SeaFox FLS have limited range and limited fields of view in both image planes, new trajectories must be generated continuously (e.g. on some fixed time interval or upon detection of a new obstacle) during execution of the current manoeuvre to ensure reactive avoidance of new obstacles.

As an example of deliberative OA, assume a REMUS vehicle is mapping a minefield with sidescan sonar prior to a mine clearance operation. For this mission, the goal locations are provided by the sequence of waypoints making up a typical lawn-mowing survey pattern. If an obstacle is detected along a specified track line, the preferred OA manoeuvre for this mission would be one that also minimizes the cumulative deviation from this track line, since we desire 100% sensor coverage of the survey area. Hence, deliberative OA implies the optimization of some performance index. Likewise, while digital nautical charts or previous vehicle surveys can be used to identify some obstacles *a priori*, this data is usually incomplete or outdated. Vehicles should be capable of storing in memory the locations of any uncharted obstacles discovered during their mission so that subsequent trajectories can avoid them—even when they are no longer in the sonar's current field of view. Deliberative OA, therefore, also entails the creation and maintenance of obstacle maps.

5.4 Obstacle detection and mapping

Detecting obstacles from sonar imagery is challenging because several factors affect the intensity of sonar reflections off objects in the water column. These factors include the size, material, and geometry of an object relative to the sonar head; interference from other acoustic sensors; and the composition of the acoustic background (e.g. bottom type, amount of sediment, etc.) to name a few (Masek, 2008). Once an obstacle has been detected, other image processing algorithms must measure its size and compute its location within the navigational reference frame. While localizing obstacles via the range and bearing data

embedded in the sonar imagery is straightforward, computing their true size is very difficult. First, for the REMUS FLS, an obstacle's height and width can be measured directly by both sonar heads only when it is located within a narrow 12-degree by 12-degree "window" directly ahead of the vehicle. Due to this narrow beam width, most obstacles are not imaged by both the horizontal and vertical sonar at the same time. Moreover, FLS images do not contain information in the region behind an obstacle's ensonified leading edge; this portion of the image is occluded. Therefore, the true horizontal and vertical extent of each obstacle must be deduced from multiple views of the same object. For a vehicle with a fixed sensor like the REMUS, this may be accomplished by deliberately inducing vehicle motion to vary the sonar angle (Furukawa, 2006) or by generating trajectories that will image the object from a different location at a later time. For these scenarios, it is desirable to balance OA behaviours with exploration behaviours in order to maximize sensor coverage and generate more complete obstacle maps. In this way, the proposed trajectory generation framework can be adapted to produce exploratory trajectories which more accurately measure the size and extent of detected obstacles (Horner et al., 2009). Nevertheless, due to the uncertainty in sonar images arising from environmental factors, sensor geometry, or obstacle occlusion, it is prudent to make conservative assumptions about an obstacle's boundaries until other information becomes available.

For the remainder of this section, we highlight different representations for incorporating obstacle size, location, and uncertainty into an obstacle map for efficient collision detection during the trajectory optimization phase. These representations can be tailored to the working environment. For operations in a kelp forest, for example, kelp stalks often appear as point-like features in horizontal-plane sonar imagery (Fig.14) but seldom appear in vertical-plane images. By making the reasonable assumption (for this environment) that these obstacles extend vertically from the sea floor to the surface, it may be simpler to perform horizontal-plane OA through this type of obstacle field. Nevertheless, when building an obstacle map comprised primarily of point features, mapping algorithms must account for the uncertainty inherent in sonar imagery. One simple but effective technique adds spherical (3D) or circular (2D) uncertainty bounds to each point feature stored in the obstacle map. Candidate OA trajectories which penetrate these boundaries violate constraint (7). Under this construct, collision detection calculations are reduced to a simple test to determine whether line segments in a discretized trajectory intersect with the uncertainty circle (2D) or sphere (3D) for each obstacle in the map. In general, when checking for line segment intersections with a circle or sphere there are five different test cases to consider (Bourke, 1992). Our application, however, requires only two computationally efficient tests to determine: i) which line segment along a discretized trajectory contains the closest point of approach (CPA) to an obstacle, and ii) whether this CPA is located inside the obstacle's uncertainty bound.

Most objects appear in sonar imagery not as point features, but as complex shapes. Unlike point features, it is difficult and computationally expensive to determine exhaustively whether or not a candidate vehicle trajectory will collide with these shapes. Instead, we can bound an arbitrary shape with a minimal area rectangle (or box, in 3D) aligned with the shape's principle axes (Fig.15). This type of object, called an oriented bounding box, is widely used in collision-detection algorithms for video games. One technique, based on the Separating Axis Theorem from complex geometry, results in an extremely fast test for line segment intersections with an oriented bounding box (Kreuzer, 2006). With slight

modification, this test can also be used to detect when a trajectory passes directly above a bounding box. In our application, we use the OpenCV computer vision library to generate a bounding box around each object detected in the horizontal image plane. For each box, we then compute its centre point, length extent, and angle relative to the vehicle's navigation frame. Due to occlusion, the width extent produced from this rectangle does not accurately convey the true size of the obstacle, so we assume a constant value for this parameter. To create a 3D (actually 2.5D) bounding box around the object, we compute its vertical extents from vertical sonar imagery. At this time, the assumption is that obstacles extend from the ocean floor to its measured height above bottom, but this method can be generalized to obstacles suspended in the water column or extending from the surface to a measured depth (i.e. ships in a harbour).

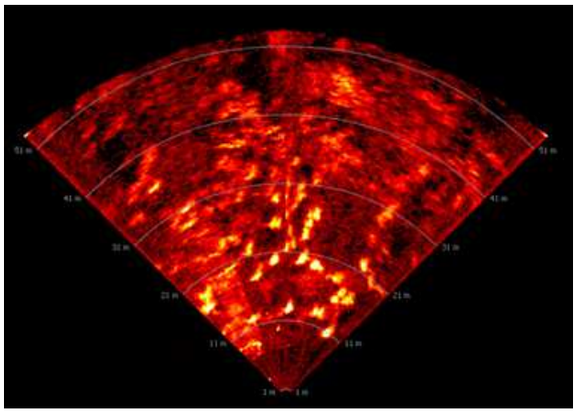


Fig. 14. Horizontal FLS image of a kelp forest

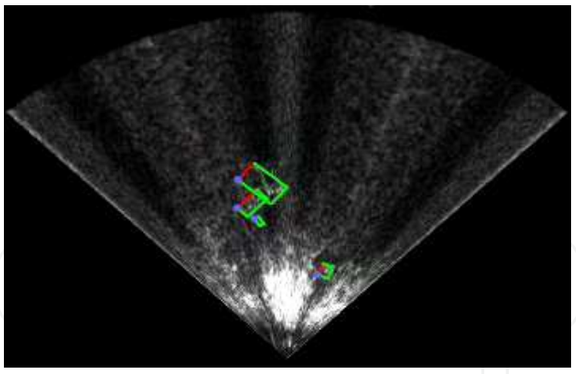


Fig. 15. Example of the bounding boxes used in conservative collision detection calculations

While oriented bounding boxes work well for mapping discrete obstacles in open-water environments, they require an additional image processing step and are not easily adapted to operations in restricted waterways. For these environments, a probabilistic occupancy grid is preferable for robustly mapping large continuous obstacles (e.g. harbour breakwaters) or natural terrain (e.g., a river's banks). Occupancy grids divide the environment into a grid of cells and assign each cell a probability of being occupied by an obstacle. Given a probabilistic sensor model, Bayes' Theorem is used to compute the probability that a given cell is occupied, based upon current sensor data. By extension, an

estimate for the occupancy state of each cell can be continually updated using an iterative technique that incorporates all previous measurements (Elfes, 1989). Figure 16a shows an occupancy grid map of a river generated by the SeaFox FLS system. In this image, each pixel corresponds to a 1-metre square grid cell whose colour represents the cell's probability of being occupied (red) or empty (green). For comparison, the inset portion of the occupancy grid map has been overlaid with an obstacle map of oriented bounding boxes in Fig.16b. Clearly, using discrete bounding boxes to represent a long, continuous shoreline quickly becomes intractable as more and more boxes are required. The occupancy grid framework is a much more efficient obstacle map representation for wide area operations in restricted waterways.

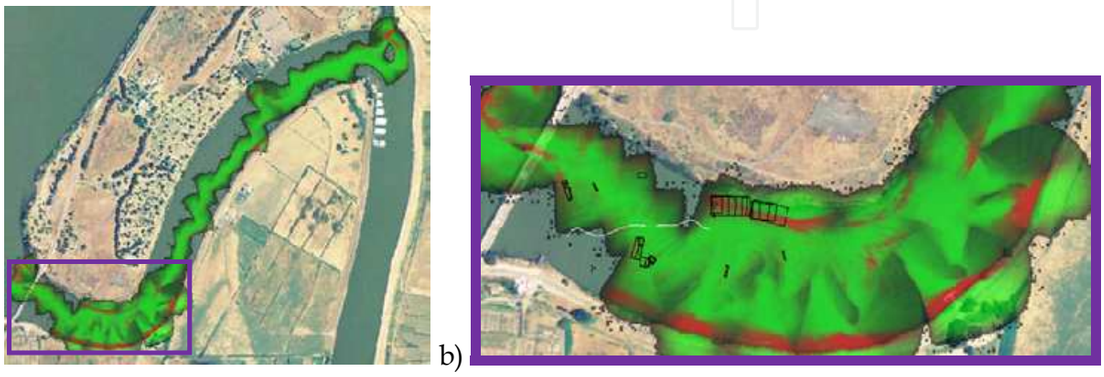


Fig. 16. Occupancy grid for a river as generated by the SeaFox FLS system

NPS has developed probabilistic sonar models for the BlueView FLS and has successfully combined separate 2D occupancy grids in order to reconstruct the 3D geometry of an obstacle imaged by the REMUS UUV's horizontal and vertical sonar arrays (Horner et al., 2009). Using this occupancy grid framework, each candidate trajectory's risk of obstacle collision is computed using the occupancy probabilities (a direct lookup operation) of the grid cells it traverses. Trajectory optimization for OA entails minimizing the cumulative risk of collision along the entire trajectory.

6. Path following

While the REMUS UUV and SeaFox USV are both commercial vehicles with proprietary autopilots, both provide communications interfaces that allow experimental sensor and computer payloads to override the primary autopilot via high-level commands. The REMUS RECON interface, for example, closely resembles the augmented autopilot depicted in Figs. 2 and 3 (although direct actuator inputs are only available for propeller and cross-body thrusters settings). For full overriding control, a payload module must periodically send valid commands containing all of the following: i) desired depth or altitude, ii) desired vehicle or propeller speed, and iii) desired heading, turn rate, or waypoint location. The developed trajectory generator (described in Section 3) outputs reference trajectories as parameterized expressions for each coordinate in a spatial curve plus a speed factor to use while traversing that curve. Using these expressions as reference trajectories, the 3D path following controller developed earlier (Kaminer et al., 2007) can compute turn rate and pitch rate commands required to drive a vehicle onto (and along) the desired trajectory. The

RECON interface, however, does not accept pitch rate commands (for vehicle safety reasons). Therefore, in order to use the aforementioned path following controller to track 3D trajectories with the REMUS UUV, controller outputs must be partitioned into horizontal (turn rate) and vertical (depth or altitude) commands as described in the following section (obviously, the SeaFox USV only uses the turn rate commands).

6.1 Horizontal plane

Consider the 2D problem geometry depicted in Fig.17, which defines an inertial {*I*} frame, Serret-Frenet {*F*} error frame and body-fixed reference frame {*b*}. The kinematic model of the vehicle (2)-(3) reduces to

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \end{bmatrix} = \begin{bmatrix} U_0 \cos\psi(t) \\ U_0 \sin\psi(t) \end{bmatrix} \tag{33}$$

with dynamics described by

$$\dot{\psi} = r \tag{34}$$

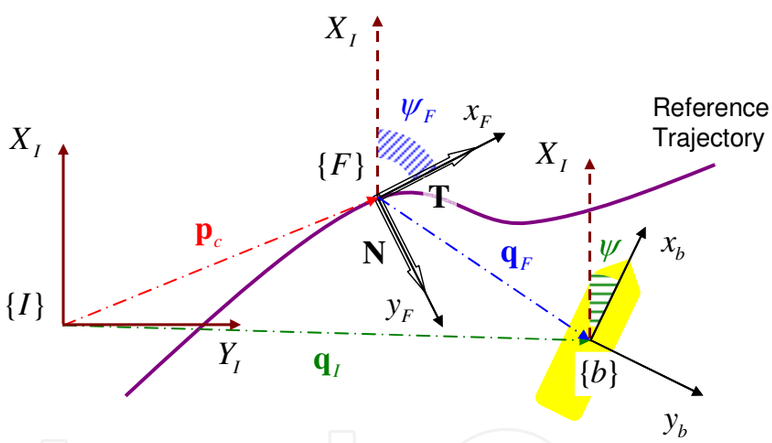


Fig. 17. Horizontal path-following kinematics

By construction, the local trajectory planner produces an analytic expression for each component of the spatial trajectory as a function of virtual arc length, $\mathbf{p}_c(\tau)$. We can also compute analytic expressions for $\mathbf{p}_c'(\tau)$ and $\mathbf{p}_c''(\tau)$, the first and second derivatives, respectively, of the spatial trajectory. Using the relationships in Fig.17, the errors can be expressed in the Serret-Frenet frame {*F*} as

$$\mathbf{q}_F = \begin{bmatrix} x_F \\ y_F \end{bmatrix} = {}^F_I R(\mathbf{q}_I - \mathbf{p}_c) \tag{35}$$

where ${}^F_I R = [\mathbf{T}, \mathbf{N}, \mathbf{B}]^T$ is a rotation matrix constructed from the tangent, normal, and binormal vectors of the Serret-Frenet error frame {*F*}. The tangent vector is computed from the expression for the trajectory's first derivative as:

$$\mathbf{T} = \frac{\mathbf{p}_c'(\tau)}{\|\mathbf{p}_c'(\tau)\|} \tag{36}$$

For the 2D problem, the normal vector components can be computed directly from the tangent vector components: $N_x = -T_y$ and $N_y = T_x$. Additionally, the signed curvature of the trajectory can be computed using the expressions for the trajectory's first and second derivatives as:

$$\kappa(\tau) = \frac{p_{cy}''(\tau)p_{cx}'(\tau) - p_{cx}''(\tau)p_{cy}'(\tau)}{\|\mathbf{p}_c'(\tau)\|^3} \tag{37}$$

Taking the time derivative of $\mathbf{q}_F$, we obtain the following state space representation for the error kinematics (i.e. the position and heading of the vehicle's body-fixed frame {*b*} relative to the Serret-Frenet frame {*F*}, which follows the desired trajectory):

$$\begin{aligned} \dot{x}_F &= -\dot{l}(1 - \kappa y_F) + U_0 \cos\psi_e \\ \dot{y}_F &= -\dot{l}(\kappa x_F) + U_0 \sin\psi_e \\ \dot{\psi}_e &= \dot{\psi} - \dot{\psi}_F = u_\psi - \dot{\psi}_F = u_\psi - \kappa\dot{l} \end{aligned} \tag{38}$$

where l is the path length of the desired spatial curve and $\dot{l}$ describes the speed at which a virtual target travels along this curve.

The goal is to drive the vehicle's position error ($\mathbf{q}_F$) and heading error (Ψ_e) to zero. This will drive the vehicle to the commanded trajectory location ($\mathbf{p}_c$) and align its velocity vector with the trajectory's tangent vector (**T**). The control signal u_ψ must now be chosen to asymptotically drive the vehicle position and velocity vectors onto the commanded trajectory. We choose the candidate Lyapunov function

$$V = \frac{1}{2}\left(x_F^2 + y_F^2 + (\psi_e - \delta_\psi)^2\right) \tag{39}$$

where δ_ψ is a shaping function that controls the manner in which the vehicle approaches the path

$$\delta_\psi = \sin^{-1}\left(\frac{-y_F}{|y_F| + d}\right) \tag{40}$$

with $d > 0$ an arbitrary constant.

Using some algebra, we choose the following control laws to ensure that $\dot{V} < 0$:

$$\begin{aligned} \dot{l} &= K_1 x_F + U_0 \cos\psi_e \\ u_\psi &= \kappa\dot{l} + \dot{\delta}_\psi + K_2(\psi_e - \delta_\psi) - \frac{\sin\psi_e - \sin\delta_\psi}{\psi_e - \delta_\psi} U_0 y_F \end{aligned} \tag{41}$$

In these expressions K_1, K_2, and d can be used as gains to tune the closed-loop performance of the path following controller.

6.2 Vertical plane

Now consider the REMUS UUV manoeuvring in the vertical plane using altitude commands. For survey operations, REMUS is typically programmed to follow a lawnmower pattern at a constant altitude above the sea floor determined by the desired sidescan sonar range. Since the ADCP/DVL sensor continuously measures vehicle altitude above the bottom, this operating mode ensures safe operation over undulating bottoms with slopes of up to 45 degrees (Healey, 2004). Obstacle avoidance manoeuvres are required to safely negotiate steeper slopes, step-like terrain features (e.g. sand bars or coral heads), or objects proud of the ocean floor. As described earlier, since the REMUS FLS is mounted in a fixed orientation, it may detect new obstacles while executing a manoeuvre to avoid the current obstacle threat. Periodic or detection-based replanning can handle these situations. This scenario was illustrated conceptually in Fig.7.

When negotiating a step-up ridge or sand bar whose extent is not known due to sonar occlusion at the time of detection, it may not be desirable to follow the planned vertical trajectory to its completion. Between planning iterations, a simple yet safer approach is to revert back to constant altitude control once the vehicle reaches a position directly above the detected object boundaries. This condition can be checked using a 2.5D version of the 3D bounding box intersection test described above. Figure 18 illustrates a simulation whereby the REMUS FLS detects the leading edge of a ridge at maximum sonar range. Image processing algorithms compute range to the object (80m) as well as its width (W, into the page) and its height above the seafloor (5.5m) but cannot determine the length of the ridge since it is occluded by its own leading edge. Therefore, the obstacle detection algorithm generates a 3D bounding box measuring W m wide x 1.0m long (assumed) x 5.5m high.

While the IDVD-method planner generates a vertical trajectory in NED coordinates, in shallow water it is safer to operate the vehicle in an altitude control mode. Therefore, the vertical coordinate trajectory is converted from a depth plan into an altitude plan by assuming constant water depth over the planning horizon and using the relationships

$$\begin{aligned} D &= h_{nom} + z(0) \qquad & h_{plan}(\tau) &= D - z_{plan}(\tau) \\ \Delta h_{plan}(\tau) &= h_{plan}(\tau) - h_{nom} \qquad & h_{cmd}(\tau) &= h_{nom} + \Delta h_{plan}(\tau) \end{aligned} \tag{42}$$

where D is the water depth computed at planner initialization time, h_{nom} is the nominal altitude set point, and $h_{cmd}(\tau)$ is the altitude command sent to the autopilot via the RECON interface. The resulting altitude plan $\Delta h_{plan}(\tau)$ is shown in Fig.18 as a deviation from the nominal mission segment altitude. As seen, sending this altitude command directly to the vehicle autopilot would cause an undesirable jump in the altitude profile once the ADCP/DVL sensor measures the vehicle's true altitude above the ridge. Instead, switching the altitude command to the nominal mission segment altitude once the vehicle reaches the ridge will produce the desired altitude profile). Note that even though Fig.18 depicts a sudden drop in altitude when the vehicle passes the ridge while commanding the nominal mission segment altitude, in practice the vehicle dynamics will ensure that the UUV executes a smooth transition back to its nominal survey altitude.

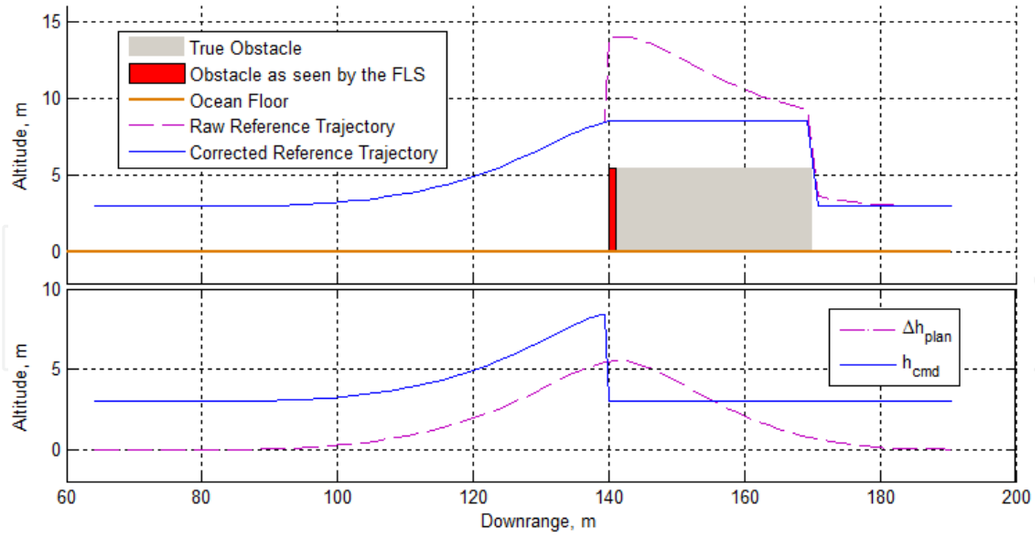


Fig. 18. Simulation results for vertical OA using UUV altitude control mode

7. Computer simulations and sea trials

The proposed path-planning method can be tailored to a specific vehicle or operational domain by modifying the performance index *J* to incorporate vehicle or actuator dynamics (feasibility constraints) and mission objectives such as OA or underwater rendezvous. This section presents simulated and in-water experimental results for four different applications which use the proposed trajectory optimization framework for UMV guidance: i) underwater docking of a UUV with a mobile underwater recovery system (MURS); ii) optimal exploitation of a terrain-relative feature map to improve UUV self-localization accuracy; iii) 2D or 3D OA in cluttered environments; and iv) specific USV implementations for sonar-based OA in riverine operations.

7.1 Underwater recovery

The goal of underwater recovery (Yakimenko et al., 2008) is to be able to compute a rendezvous trajectory from any point on the UUV holding pattern to any point on the MURS holding pattern as shown in Fig.19 (note hereinafter that depth values are shown as negative numbers).

While the stochastic simulation shown in Fig.19 employs circular race tracks, in practice the MURS would establish a race track that allowed it to travel back and forth along two long track legs (see Fig.20). These legs are needed to allow sufficient time to contact the UUV (which is assumed to be in its holding pattern somewhere within communication range) and allow it to transit from its holding pattern to the rendezvous point. The proposed sequence of events is to have the MURS (at position 1 in Fig.20) signal the UUV (at position 2) and command it to proceed to a rendezvous point by a certain time. The UUV computes the trajectory required to comply with the command. If the commanded rendezvous is feasible, the UUV sends an acknowledgement message. Otherwise (i.e. the request violated some constraint) the UUV sends a denial message (stage A in Fig.20) and requests that the MURS command a different rendezvous point or time. The final point of the trajectory is located in the approximate location of the MURS docking station at a given time. Knowing the

geometry of the MURS allows the planner to construct a "keep out" zone corresponding to the MURS propeller and aft control surfaces. The UUV rendezvous trajectory must avoid this area. Once the rendezvous plan has been agreed upon and acknowledged, both the UUV and the MURS proceed to position 3 for rendezvous (stage B). Finally, at position 4 the recovery operation (stage C) is completed.

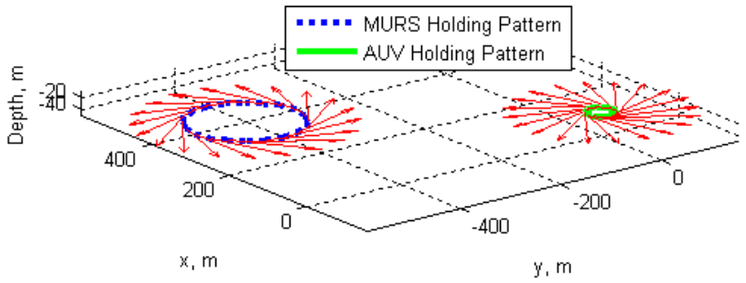


Fig. 19. Manifold of initial and final conditions

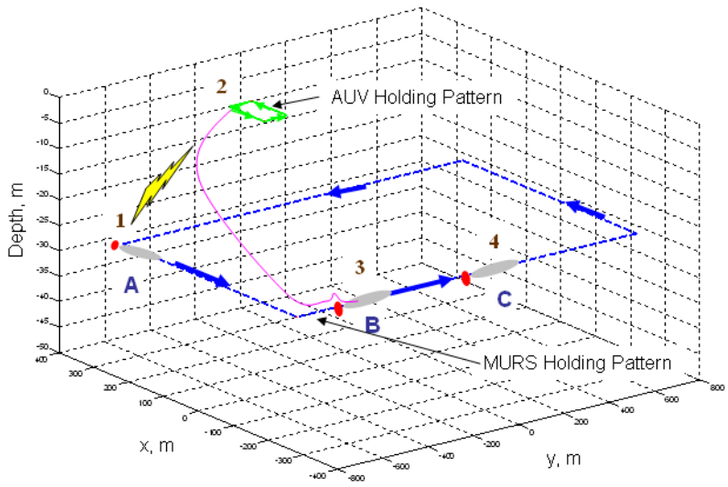


Fig. 20. Proposed rendezvous scenario

The simulated rendezvous scenario assumes three stages: communication (A), execution (B), and recovery (C), respectively. From the trajectory generation standpoint we are primarily concerned with optimizing the path that would bring the UUV from its current position (point 2) to a certain rendezvous state (point 3) in the preset time T_r proposed by the MURS, while obeying all possible real-life constraints and avoiding the MURS keep out zone.

Figures 21 and 22 present a computer simulation in which a MURS is moving due east at $1m/s$ ($1.94kn$) with the docking station at a depth of 15m. A UUV is located 800 meters away. The MURS wishes to conduct a rendezvous operation T_r minutes later and sends the corresponding information to the UUV. This information includes the proposed final position x_f, y_f, z_f rendezvous course, speed, and time. Figure 21 shows several generated trajectories, which meet the desired objectives for this scenario and also avoid an obstacle located along the desired path to MURS. These trajectories differ by the arrival time T_r.

During handshaking communications with the MURS, the UUV determines whether the suggested T_r is feasible. Of the four trajectories shown, the trajectory generated for

$T_r = 450s$ happens to be infeasible (the constraints on controls are violated). The solution of the minimum-time problem for this scenario yielded 488 seconds as the soonest possible rendezvous time.

The other three trajectories shown in Fig.21 are feasible. That means that the boundary conditions are met (by construction) and all constraints including OA are satisfied (via optimization). As an example, Fig.22 shows the time histories for the yaw rate $\dot{\psi}_c$ and flight path angle γ_c vehicle control parameters as well as the UUV's speed as it followed the trajectory for $T_r = 600s$.

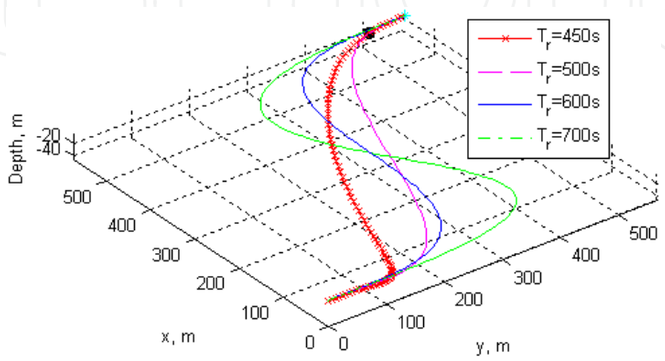


Fig. 21. Examples of rendezvous trajectories

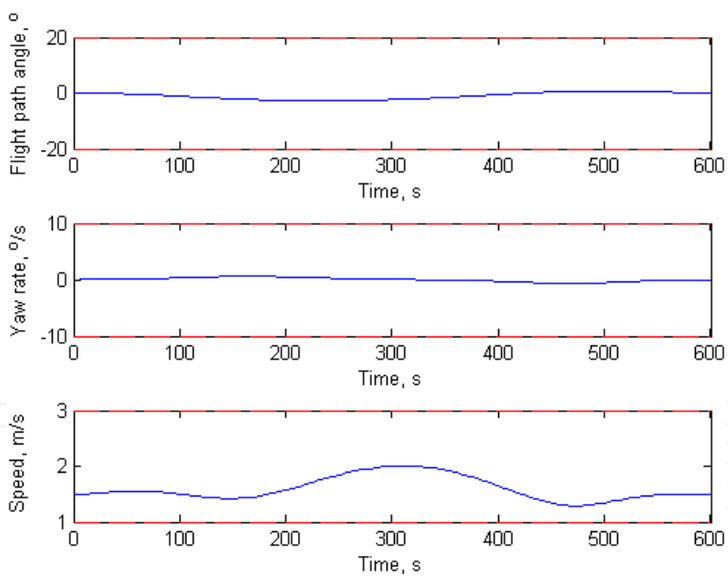


Fig. 22. Constrained vehicle parameters for $T_r = 600s$

Stochastic simulations of the manifolds shown in Fig.21 illustrate that a successful rendezvous can take place in all cases as long as T_r is greater than a certain value. Furthermore, they show that minimization of the performance index using the IDVD method ensures that a smooth, realizable trajectory is calculated in just a few seconds, regardless of the initial guess. Converting code to an executable file in lieu of using an interpretative programming language reduces execution time down to a fraction of a second.

7.2 Feature-based navigation

In the last decade, several different UUVs have been developed to perform a variety of underwater missions. Survey-class vehicles carry highly accurate navigational and sonar payloads for mapping the ocean floor, but these payloads make such vehicles very expensive. Vehicles which lack these payloads can perform many useful missions at a fraction of the cost, but their performance will degrade over time from inaccurate self-localization unless external navigation aids are available. Therefore, it is interesting to consider collaborative operations via a team of vehicles for maximum utility at reasonable cost. The NPS CAVR has been investigating one such concept of operations called feature-based navigation. This technique allows vehicles equipped only with a GPS receiver and low cost imaging sonar to exploit an accurate sonar map generated by a survey vehicle. This map is comprised of terrain or bottom object features that have utility as future navigational references. This sonar map is downloaded to the low-cost follow-on vehicles before launch. Starting from an initial GPS position fix obtained at the surface, these vehicles then navigate underwater by correlating current sonar imagery with the sonar features from the survey vehicle's map. The localization accuracy of vehicles performing feature-based navigation can be improved by maximizing the number of times navigational references are detected with the imaging sonar. The following simulation demonstrates how the IDVD trajectory generation framework can be tailored to this application. By incorporating a simple geometric model of an FLS having a range of 60m, 30-degree horizontal FOV and operating at a nominal ping rate of 1Hz, a new performance index was designed to favour candidate trajectories, which point the sonar toward navigational references in the *a priori* feature map. For this example, we sought trajectories that could obtain at least three sonar images of each feature in the map. Figure 23 shows results of a computer simulation in which the number of times each target was imaged by the sonar has been annotated. The resulting trajectory is feasible (i.e. satisfies turn rate constraints) and yields three or more sonar images of all but two targets.

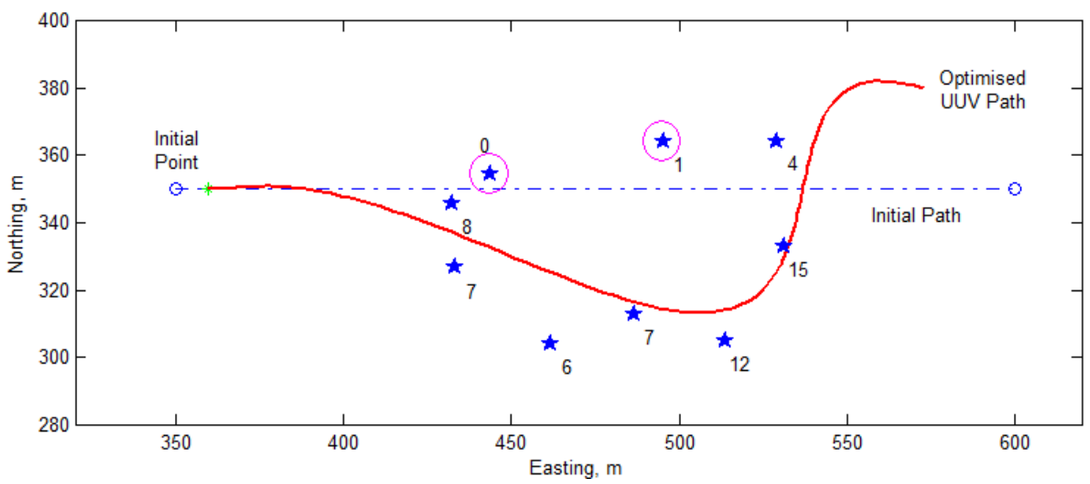


Fig. 23. Simulation results for a feature-based navigation application

7.3 Obstacle avoidance in cluttered environments

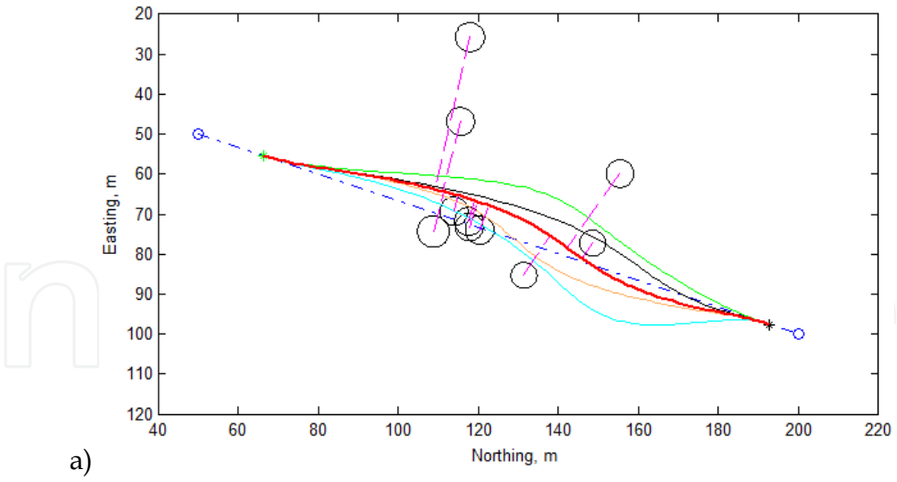
Another application which benefits from the aforementioned trajectory generation algorithm is real-time OA in a highly cluttered environment. Figure 24 illustrates simulated trajectories for avoiding a field of point-like objects in the 2D horizontal plane (e.g. a kelp forest) and in all three dimensions (e.g. a mine field). In both simulations, the performance index was designed to minimize deviations from a predefined survey track line while avoiding all randomly generated obstacles via a CPA calculation. Terminal boundary conditions for the OA manoeuvre were chosen to ensure the UUV rejoined the desired track line before reaching the next waypoint (i.e. the manoeuvre terminated at a position 95% along the track segment). Initial boundary conditions were chosen to simulate a random obstacle detection which triggers an avoidance manoeuvre after the UUV has completed about 10% of the predefined track segment. For illustration purposes, Fig.24 includes several candidate trajectories evaluated during the optimization process although the algorithm ultimately converged to the trajectory depicted with a thicker (red) line (CPA distances to each obstacle appear as dashed lines).

Figure 25 shows the results from an initial sea trial of 3D OA that took place in Monterey Bay on 9 December 2008. This experiment tested periodic trajectory generation and replanning on the REMUS UUV using a simulated obstacle map comprised of oriented bounding boxes. As seen in Fig.25, initially the REMUS UUV follows a predefined track segment (dash-dotted line) at 4 meters altitude. At some point the vehicle's FLS simulator "detects" an obstacle (i.e. the current REMUS position and orientation place the virtual obstacle within the range and aperture limits of the FLS). This activates the OA mode, and the planner generates an initial trajectory (green) from the current vehicle position to the final waypoint. REMUS follows this trajectory until the next planning cycle (4 seconds later) when the vehicle generates a new trajectory and continues this path planning-path following cycle.

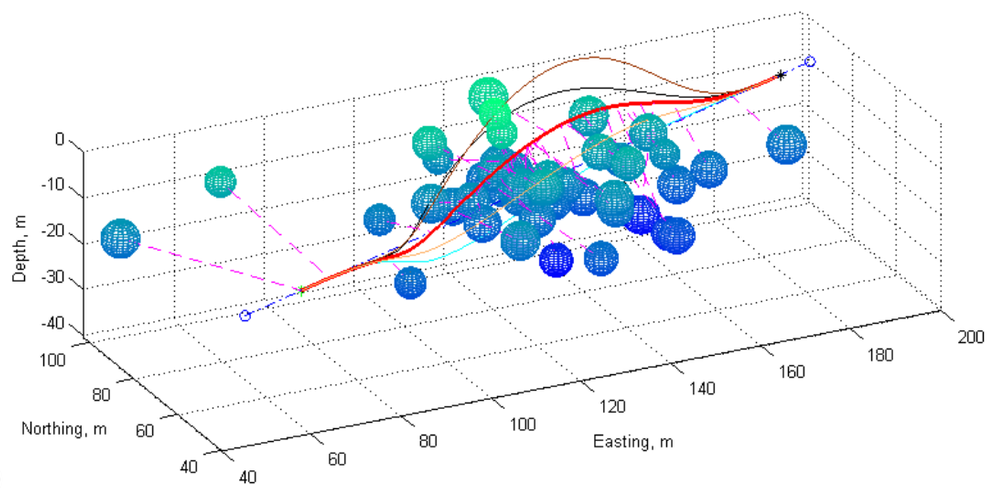
7.4 Obstacle avoidance in restricted waterways

The NPS CAVR in collaboration with Virginia Tech (VT) is developing technologies to enable safe, autonomous navigation by USVs operating in unknown riverine environments. This project involves both surface (laser) and subsurface (sonar) sensing for obstacle detection, localization, and mapping as well as global-scale (wide area) path planning, local-scale trajectory generation, and robust vehicle control. The developed approach includes a hybrid receding horizon control framework that integrates a globally optimal path planner with a local, near-optimal trajectory generator (Xu et al., 2009).

The VT global path planner uses a Fast Marching Method (Sethian, 1999) to compute the optimal path between a start location and a desired goal location based on all available map information. While resulting paths are globally optimal, they do not incorporate vehicle dynamics and thus cannot be followed accurately by the USV autopilot. Moreover, since level set calculations are computationally expensive, global plans are recomputed only when necessary and thus do not always incorporate recently detected obstacles. Therefore, a complimentary local path planner operating over a short time horizon is required to incorporate current sensor information and generate feasible OA trajectories. The IDVD-based trajectory generator described above is ideally suited for this purpose. VT has developed a set of matching conditions which guarantee the asymptotic stability of this



a)



b)

Fig. 24. Simulated 2D (a) and 3D (b) near-optimal OA trajectories

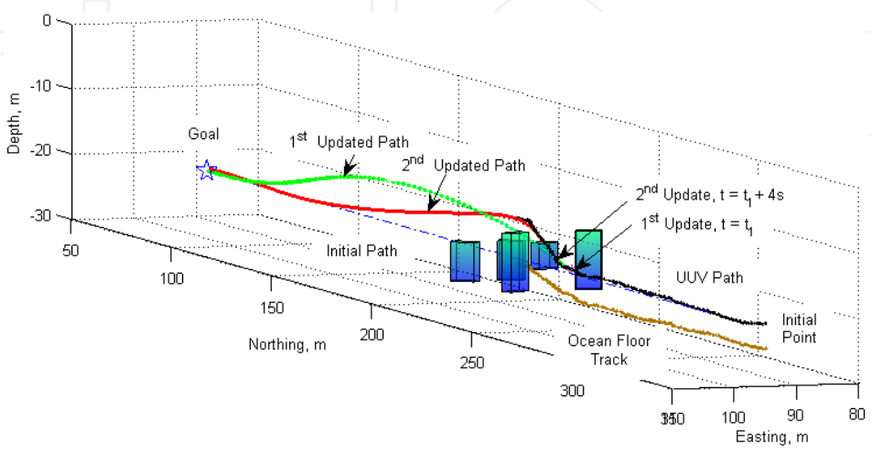
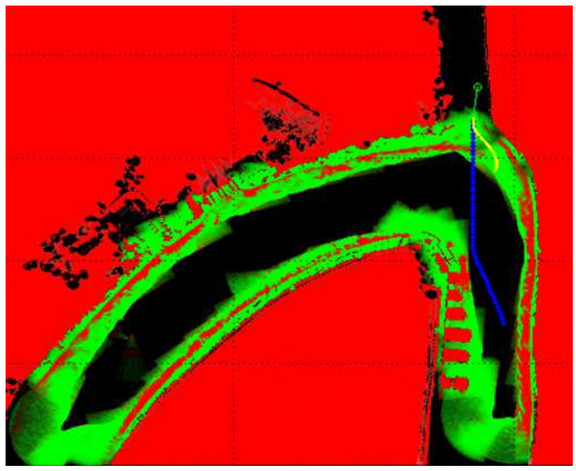


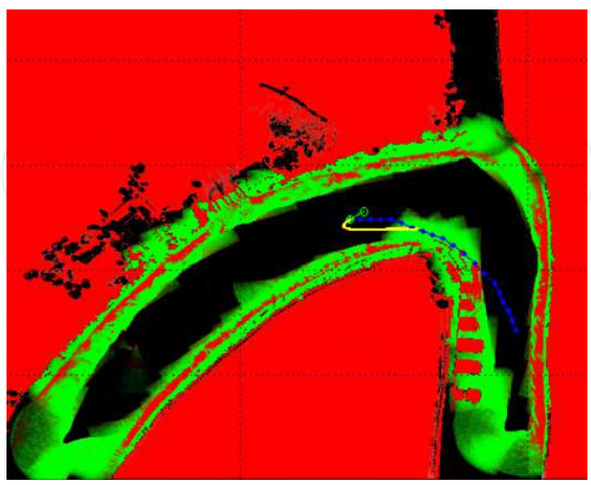
Fig. 25. REMUS sea trial results demonstrating periodic planning and path following

framework. When these matching conditions are satisfied, the sequence of local trajectories will converge to the global path's goal location. If the local trajectories no longer satisfy these conditions (usually because the global path is no longer compatible with recently detected obstacles), the global path is recomputed.

Simulation results demonstrate the need for local trajectories that incorporate vehicle dynamics and real-time sensor data (Fig.26). For this simulation, an initial level set map was computed using an occupancy grid created by masking land areas as occupied and water areas as unoccupied in an aerial image of the Sacramento River operating area. Performing gradient descent on the level set from the USV's initial position produces an optimal path shown in blue. To simulate local trajectory generation with a stale global plan, the initial level set map was not updated during the entire simulation. Meanwhile, to simulate access to real-time sensor data, the local planner was provided with a complete sonar map generated during a previous SeaFox survey of the area. In Fig.26, this sonar map has been overlaid on the *a priori*



a)



b)

Fig. 26. Simulated local OA trajectories

map with red and green colour channels representing the probability that a cell is occupied or unoccupied, respectively. Black pixels represent cells with unknown status. A short green line segment depicts the USV's orientation when the local planner is invoked, and the resulting trajectory is shown in yellow. The first simulation (Fig.26a) shows a local trajectory which deviates from the stale global plan to avoid a sand bar detected with sonar. In the second simulation (Fig.26b) the USV is initially heading in a direction opposite from the global path, but the local planner generates a dynamically feasible trajectory to turn around and rejoin the global path later.

To track these local trajectories, the 2D controller described in Section 6.1 was implemented on the SeaFox USV by mapping the controller's turn rate commands into rudder commands understood by the SeaFox autopilot. After validating the turn rate controller design during sea trials on Monterey Bay, the direct method trajectory generator and closed-loop path following controller were tested on the Pearl River in Mississippi on 22 May 2010 (Fig.27). For this test, the local planner used a sonar map of the operating area to generate the trajectory (the cyan line) from an initial orientation (depicted by the yellow arrow) to a

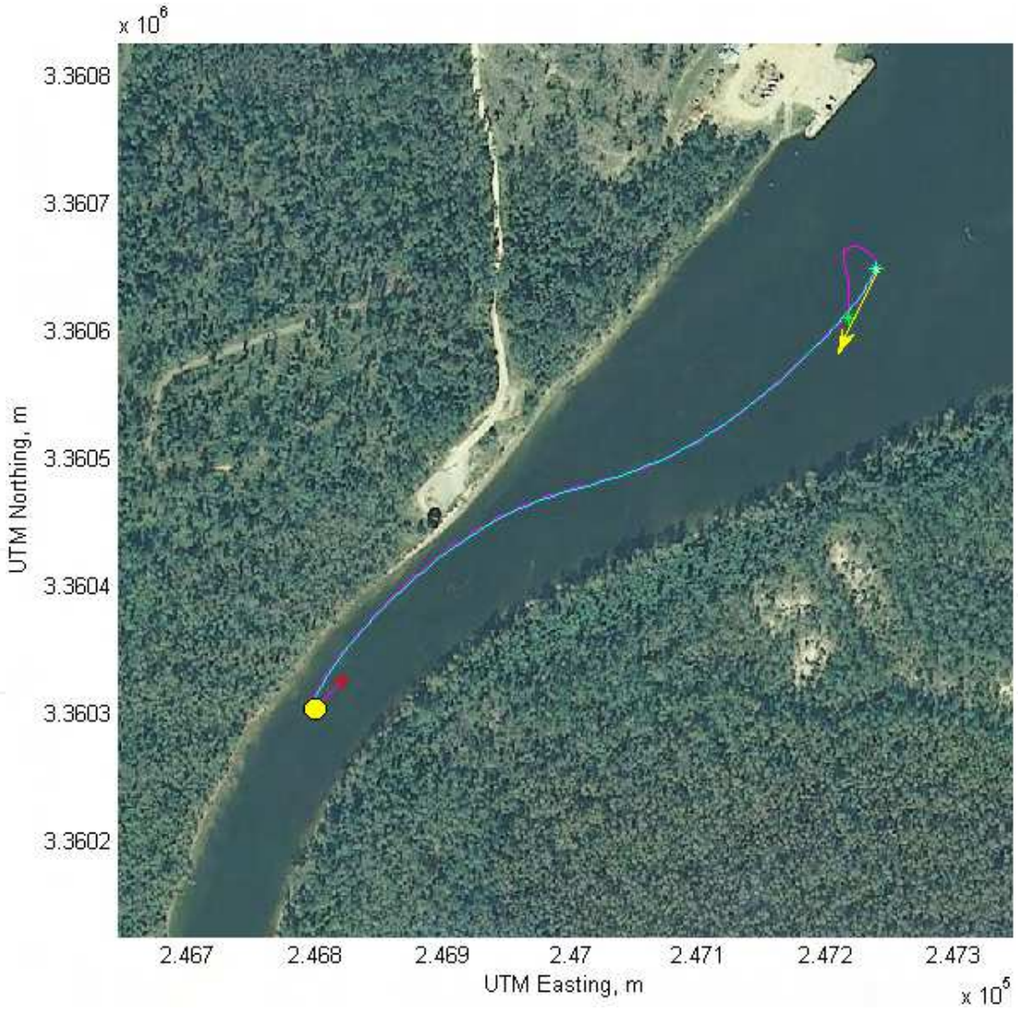


Fig. 27. Path-following controller test on the Pearl River

desired goal point (depicted by a circle). The SeaFox USV then followed it almost precisely (the magenta line). As seen from Fig.27 the trajectory generator was invoked at an arbitrary location while the USV was performing a clockwise turn. Since the USV was commanded to return to its start location upon completion of this manoeuvre, the magenta line includes a portion of this return trajectory as well (otherwise, the actual USV track would be nearly indistinguishable from the reference trajectory on this plot).

8. Conclusion

An onboard trajectory planner based on the Inverse Dynamics in the Virtual Domain direct method presented in this chapter is an effective means of augmenting an unmanned maritime vehicle's autopilot with smooth, feasible trajectories and corresponding controls. It also facilitates incorporation of sophisticated sensors such as forward-looking sonar for deliberative and reactive obstacle avoidance. This approach has been implemented on both unmanned undersea and surface vehicles and has demonstrated great potential. Beyond its ability to compute near-optimal collision-free trajectories much faster than in real time, the proposed approach supports the utilization of any practically-sound compound performance index. This makes the developed control architecture quite universal, yet simple to use in a variety of applied scenarios, as demonstrated in several simulations and preliminary sea trials. This chapter presented results from only a few preliminary sea trials. Future research will continue development of the suggested trajectory framework in support of other tactical scenarios.

9. Acknowledgements

The authors wish to gratefully acknowledge the support of Doug Horner, Co-Director of the CAVR and Principle Investigator for the REMUS UUV and SeaFox USV research programs at NPS. In addition, Sean Kragelund would like to thank his CAVR colleagues Tad Masek and Aurelio Monarrez. Mr. Masek's outstanding software development work to implement obstacle detection and mapping with forward looking sonar made possible the OA applications described herein. Likewise, the tireless efforts of Mr. Monarrez to continually upgrade, maintain, and operate CAVR vehicles in support of field experimentation have made a lasting contribution to this Center.

10. References

Basset, G., Xu, Y. & Yakimenko, O. (2010). Computing short-time aircraft maneuvers using direct methods," *Journal of Computer and Systems Sciences International*, 49(3), 145-176

BlueView Technologies, Inc. (2011). 2D Imaging sonar webpage. Available from: www.blueview.com/2d-Imaging-Sonar.html

Bourke, P. (1992). Intersection of a line and a sphere (or circle). Professional webpage. Available from: http://paulbourke.net/geometry/sphereline

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6), 46-57

Furukawa, T. (2006). *Reactive obstacle avoidance for the REMUS underwater autonomous vehicle using a forward looking sonar*. MS Thesis, NPS, Monterey, CA, USA

Gadre, A., Kragelund, S., Masek, T., Stilwell, D., Woolsey, C. & Horner, D. (2009). Subsurface and surface sensing for autonomous navigation in a riverine environment. In: *Proceedings of the Association of Unmanned Vehicle Systems International (AUVSI) Unmanned Systems North America convention*, *Washington, DC, USA*

Healey, A. J. (2004). Obstacle avoidance while bottom following for the REMUS autonomous underwater vehicle. In: *Proceedings of the IFAC conference*, *Lisbon*, *Portugal*

Horner, D. & Yakimenko, O. (2007). Recent developments for an obstacle avoidance system for a small AUV. In: *Proceedings of the IFAC conference on Control Applications in Marine Systems*, *Bol*, *Croatia*

Horner, D., McChesney, N., Kragelund, S. & Masek, T. (2009). 3D reconstruction with an AUV-mounted forward-looking sonar. In: *Proceedings of the International symposium on Unmanned Untethered Submersible Technology (UUST09), Durham, NH, USA*

Hydroid, Inc. (2011). REMUS 100 webpage. Available from: www.hydroidinc.com/remus100.html

Kaminer, I., Yakimenko, O., Dobrokhodov, V., Pascoal, A., Hovakimyan, N., Cao, C., Young, A. & Patel, V. (2007). Coordinated path following for time-critical missions of multiple UAVs via L_1 adaptive output feedback controllers. In: *Proceedings of the AIAA Guidance, Navigation, and Control conference*, *Hilton Head, SC, USA*

Kreuzer, J. (2006). 3D programming – weekly: Bounding boxes. Collision detection tutorial webpage. Available from: www.3dkingdoms.com/weekly/weekly.php?a=21

Masek, T. (2008). *Acoustic image mModels for navigation with forward-looking sonars*. MS Thesis, NPS, Monterey, CA, USA

Northwind Marine. (2011). SeaFox webPage. Available from: www.northwindmarine.com/military-boats

Sethian, J. (1999). Fast marching method. *SIAM Review*, 41(2), 199-235

Xu, B., Kurdila, A. J. & Stilwell, D. J. (2009). A hybrid receding horizon control method for path planning uncertain environments. In: *Proceedings of the IEEE/RSJ International conference on Intelligent Robots and Systems, St. Louis, MO, USA*

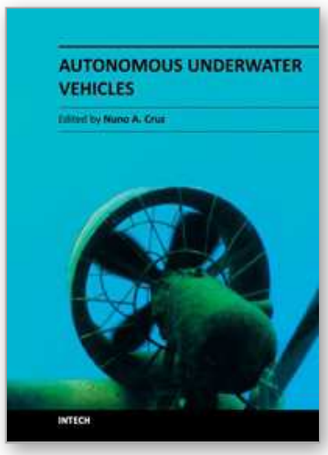
Yakimenko, O. & Slegers, N. (2009). Optimal control for terminal guidance of autonomous parafoils. In: *Proceedings of the 20th AIAA Aerodynamic Decelerator Systems Technology conference, Seattle, WA, USA*

Yakimenko, O. (2000). Direct method for rapid prototyping of near optimal aircraft trajectories. *Journal of Guidance, Control, and Dynamics*, 23(5), 865-875

Yakimenko, O. (2011). *Engineering computations and modeling in MATLAB/Simulink*. AIAA Education Series, ISBN 978-1-60086-781-1, Arlington, VA, USA

Yakimenko, O. A. (2008). Real-time computation of spatial and flat obstacle avoidance trajectories for AUVs. In: *Proceedings of the 2nd IFAC workshop on Navigation, Guidance and Control of Underwater Vehicles (NGCUV'08)*, *Killaloe*, *Ireland*

Yakimenko, O.A., Horner, D.P. & Pratt, D.G. (2008). AUV rendezvous trajectories generation for underwater recovery, In: *Proceedings of the 16th Mediterranean conference on Control and Automation, Corse, France*



Autonomous Underwater Vehicles

Edited by Mr. Nuno Cruz

ISBN 978-953-307-432-0

Hard cover, 258 pages

Publisher InTech

Published online 17, October, 2011

Published in print edition October, 2011

Autonomous Underwater Vehicles (AUVs) are remarkable machines that revolutionized the process of gathering ocean data. Their major breakthroughs resulted from successful developments of complementary technologies to overcome the challenges associated with autonomous operation in harsh environments. Most of these advances aimed at reaching new application scenarios and decreasing the cost of ocean data collection, by reducing ship time and automating the process of data gathering with accurate geo location. With the present capabilities, some novel paradigms are already being employed to further exploit the on board intelligence, by making decisions on line based on real time interpretation of sensor data. This book collects a set of self contained chapters covering different aspects of AUV technology and applications in more detail than is commonly found in journal and conference papers. They are divided into three main sections, addressing innovative vehicle design, navigation and control techniques, and mission preparation and analysis. The progress conveyed in these chapters is inspiring, providing glimpses into what might be the future for vehicle technology and applications.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Oleg A. Yakimenko and Sean P. Kragelund (2011). Real-Time Optimal Guidance and Obstacle Avoidance for UMVs, Autonomous Underwater Vehicles, Mr. Nuno Cruz (Ed.), ISBN: 978-953-307-432-0, InTech, Available from: http://www.intechopen.com/books/autonomous-underwater-vehicles/real-time-optimal-guidance-and-obstacle-avoidance-for-umvs

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen