QUASI-RANDOM, MANOEUVRE-BASED MOTION PLANNING ALGORITHM FOR AUTONOMOUS UNDERWATER VEHICLES

Chiew Seon Tan, [1] **Robert Sutton, John Chudley**
Marine and Industrial Dynamics Analysis Group
School of Engineering
The University of Plymouth,
PL4 8AA, UK

Abstract: This paper presents an approach using a hybrid modelling technique known as Manoeuvre Automaton (MA) to capture the key dynamics of a nonlinear autonomous underwater vehicle (AUV) in such a way that high-level tasks such as optimal motion planning can be computationally simplified, while still allowing it to perform complicated manoeuvres when the situation arises. With respect to motion planning in an obstacle filled environment, an incremental stochastic technique derived from the Rapid-exploring Random Tree (RRT) algorithm is applied. This paper proposes a multiple nested node version of RRT and also addresses the case of a time varying final state. Simulation results as presented, using a 3 degree-of-freedom (DOF) nonlinear AUV model in order to prove the viability of the concept.

Keywords: Hybrid system, Manoeuvre Automaton, Dynamics quantisation, Optimal motion planning, Motion primitives

1. INTRODUCTION

In the last few years, AUVs are frequently being employed for sea bottom exploration, mine-hunting, scientific data gathering and reconnaissance missions. The requirement for the successful accomplishment of all the above tasks has manifested itself into an urgent demand for an increase in AUV autonomy. In fact, one area that needs particular attention is collision avoidance. Due to its obvious complexity and the limited length, this paper shall concentrate on addressing only the motion planning issues of an AUV.

Lately, there has been a sudden paradigm shift by the scientific communities from AUV deep sea exploration missions to deployment in littoral waters. The littoral zone is important for scientific research since it houses the bulk of ocean based organisms. Likewise, the navies have demonstrated a keen interest in exploiting AUV technology as a potential force multiplier to complement their amphibious power projection plans. Obviously, one can envisage numerous important military missions such as port infiltration and mine-hunting, where it is crucial for an AUV to be able to navigate in an unknown and hostile terrain.

Several inherent characteristics of an AUV particularly its highly nonlinear coupled dynamics, underactuated, non-driftless, non-minimum phase behaviour and its subjection to unpredictable exogenous disturbances makes controller design nontrivial. The last attribute is especially relevant for small, lightweight AUVs such as *Remus* (Prestero, 2001). Typically, a few linearised models are utilised for the AUV controller design, thus artificial operational constraints must be imposed to avoid violation of the linearity assumption. Consequently, this introduces additional restric-

[1] The authors would like to sincerely thank Dr. E.Frazzoli, for his various enlightening explanations and inputs. C. S. Tan is partly supported by IMarEST Stanley Gray Fellowship Award.

tion to the system performance envelope. Alternatively, a more preferable approach is to "quantise" the AUV dynamics, transforming a continuous dynamic model governed by complex nonlinear differential equations into a hybrid model which not only possesses higher levels of abstraction but is also more beneficial computationally. With an added advantage, this approach also conveniently permits the incorporation of complex and aggressive manoeuvres into the AUV. This process is achieved via the Manoeuvre Automaton (MA) representation.

Most path planning techniques introduced to date are based firmly on deterministic methods and graph searches. Unfortunately, due to their sample space discretisation issues, the generated trajectories need extra smoothing and interpolation. Notwithstanding this, the system dynamics are also neglected to avoid state-explosion effect. Collectively, these factors ensue a very conservative system performance. Randomisation methods are becoming popular as they are inherently more robust to the state explosion effect. One version is the Rapid-exploring Random Tree (RRT), which this paper extends and integrates with the MA.

This paper begins with a brief outline of the MA concept and its merits in Section 2. A brief theoretical foundation on how MA can be extended to solve optimal motion planning problem for cases without obstacles is provided. Section 3 discusses the AUV dynamic model and the generation of motion primitives. Implicitly, the latter process is critical as it will dictate the achievable behaviour of the AUV, *per se*. Section 4 is devoted to the integration of the RRT algorithm with the MA for the motion planning problem in the case of an obstacle filled environment. Discussion of the simulation results are contained in Section 5. Finally, Section 6 contains concluding remarks and future work.

2. THE MANOEUVRE AUTOMATON (MA)

The MA, a form of finite state machine, is proposed by Frazzoli *et al.* (1999) as a unified framework for formalising the control of nonlinear systems with symmetries. In essence, the main idea is to generate a complete trajectory via sequential combination of the copies of motion primitives from a library set. These motion primitives are extracted from the vehicle in an open-loop mode.

The approach relies primarily on two different types of motion primitives: trim trajectories and manoeuvres. Trim trajectories (relative equilibria) correspond to steady state behaviour in situation when the velocities in body-axes and inputs are constants. On the other hand, manoeuvres can be seen as finite time motion primitives that interconnect two trim trajectories together.

Similar to a differential or difference equation, a MA transcription, describes a dynamic system, differing only in that it has hybrid elements in both its control inputs (τ, p), and state vector $(\mathbf{x}, q)$. MA evolves in so-called "dense time" by either continuous flows or discrete transitions. Consequently, at each particular moment, the system is constrained to be either in a trim condition q or performing a manoeuvre p. Thus the system behaviour can also be explicitly formulated as below.

- An MA system H starting at state vector $(\mathbf{x}_i, q_i)$ in trim trajectories, evolve according to $f_q(\cdot)$ as determined by the length of the τ_k, which can be infinite. Where $f_q(\cdot)$ is the governing differential equation at the specific discrete state q_k. The hybrid state then evolves as:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \tau_k \dot{\mathbf{x}}_q \quad (1)$$
$$q_{k+1} = q_k \quad (2)$$
$$t_{k+1} = t_k + \tau_k \quad (3)$$

  where $\dot{\mathbf{x}}_q$ is the time rate of change of the vehicle's continuous state variables and k is the "stage" number.
- In the case of performing a manoeuvre p, the vehicle leaves the trim trajectory q_1 for a finite length of time before settling to the trim trajectory q_2. Mathematically, the manoeuvre is initiated by the control action p, which is discrete, and is described by a fixed duration Δt_p and displacement $\Delta \mathbf{x}_p$ in the continuous state space, as illustrated in Fig. 1 for a $SE(2)$ case. In reality, the control history of the continuous state-space system is implicitly encoded in the control action p. As such, when manoeuvering, the hybrid state evolves as:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta \mathbf{x}_p \quad (4)$$
$$q_{k+1} = q_2 \quad (5)$$
$$t_{k+1} = t_k + \Delta t_p \quad (6)$$

Although, the hybrid control input at instant k can be described by a vector $(\tau, p)_k$, however only one input, either τ or p can be active at any moment.

By having the AUV continuous behaviour encoded as a discrete state q, its configuration can be described by an element of the Lie group G of rigid motions in $\mathbb{R}^2$ or $\mathbb{R}^3$, called $SE(2)$ or $SE(3)$, respectively. In planar situations, where the altitude is constant, $SE(2)$ will be employed. The reason for restricting the formulation to $SE(2)$ has real practical significance, and will be elaborated

in Section 3. The group $SE(2)$ can be expressed using the homogenous coordinates as follows:

$$g = \begin{bmatrix} \cos\psi & -\sin\psi & x \\ \sin\psi & \cos\psi & y \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

The Lie algebra elements $\xi \in SE(2)$ are represented as matrices in $\mathbb{R}^{3\times3}$ and for a special case of $\omega = 0$, one yields Equation 8 to describe the configuration change after τ length of time in trim trajectory. The subscript k is omitted for clarity.

$$\mathrm{e}^{\xi\tau} = \begin{bmatrix} 1 & 0 & \upsilon_x\tau \\ 0 & 1 & \upsilon_y\tau \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

One can also describe the configuration change resulted of a manoeuvre p using Equation 7. As a result of applying the MA representation, one can now mathematically describe the system configuration by concatenating these motion primitives as expressed below:

$$g_f = g_0 \left[\prod_{k=1}^{N} e^{(\xi_k,\tau_k)} g_k \right] e^{\xi_{k+1}\tau_{k+1}} \tag{9}$$

Where g_0 and g_f is the initial and final configuration. $e^{(\xi_k,\tau_k)}$ and g_k represent the transformation of applying the k-th trim trajectory and the k-th manoeuvre, respectively.

The MA representation allows one to express easily the optimal motion planning problem. This is true, for certain cost functions that share the symmetry properties of the system, such as minimum time, minimum length and minimum control effort. For the special case of the minimum time cost functional, one can formulate it as Equation 10.

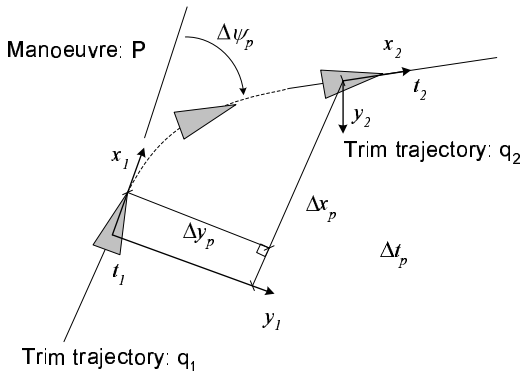

Fig. 1. Displacement of configuration variables and its time duration for manoeuvre p

$$\min_{p_k,\tau_k} \sum_{k=1}^{N} (\Delta t_{p(k)} + \tau_k) \tag{10}$$

such that Equation 9 is satisfied and $\tau \geq 0$.

The above optimisation problem can be solved using a dynamic programming (DP) technique (Frazzoli, 2001; Schouwenaars *et al.*, 2003). For the unique case when $\dot{\psi} = 0$, such that when all the trim trajectories are translations, the cost is linear with respect to the coasting variables τ, hence one can employ the linear programming (LP) method instead (Frazzoli, 2002*a*).

3. AUV DYNAMIC MODEL AND MA IMPLEMENTATION

In this section, the AUV model and techniques for synthesising the motion primitives are presented. This is done, in order to convert the continuous system model into the MA representation. The AUV model was supplied by QinetiQ, based on the *Autosub* vehicle (Millard *et al.*, 1998), which has a torpedo shaped hull. Dimensionally, the vehicle is 7 m long, and approximately 1 m in diameter and has a nominal displacement of 3600 kgs. In this paper, the model is restricted to only latitude dynamics and yaw control limited to the locked bow rudders. The vehicle has a maximum rudder deflection of $\pm25.2°$ and a rudder rate limit of $9.9°/s$. Including these two components into the model resulted in a nonlinear system. The pitch and roll effects are neglected. The main reason for concentrating only on a latitude model is due to the limitation imposed by a forward looking sonar. To elaborate, most commercial forward looking sonars are only capable of providing a projection of 2D image of the terrain, hence determination of object depth is extremely difficult. Accordingly, to mitigate any risk of collision, the AUV must avoid all the perceived obstacles.

Figure 2 shows a possible MA representation of the *Autosub* dynamics. Both 2 m/s and 5 m/s of cruising speeds are illustrated. Unfortunately, the above model lacks propulsion dynamics, here the forward velocity was held constant by an Proportional-Integral (PI) controller during the experiments. Therefore, this constrains the following simulations to only one speed regime which was selected to be 5 m/s. Normally, for the purpose of generating trim trajectories, a velocity augmentation loop must first be designed into the system. Nonetheless, this process is redundant since the AUV is already assumed to be cruising at a constant velocity. The *Autosub* model is discretised using the zero-hold method and a sampling frequency set to 10 Hz.Referring to Figure 2 again, it shows clearly a library that constitutes manoeuvres of 15°, 30°, 60°, 120°. Since the manoeuvres

are symmetry, the opposite direction manoeuvres are not shown. The manoeuvres should encompass the important performance envelope of the AUV, and their generation can be attained via a human operator or a controller input. The latter method is selected for the following study. A Proportional-Derivative (PD) autopilot is designed so that one can extract the manoeuvres by input step inputs. Obviously, a more advanced controller can also be applied to extract better performance out of the AUV. The input and the state histories are recorded. Table 1 shows a few manoeuvres with their associated execution time duration and displacements.

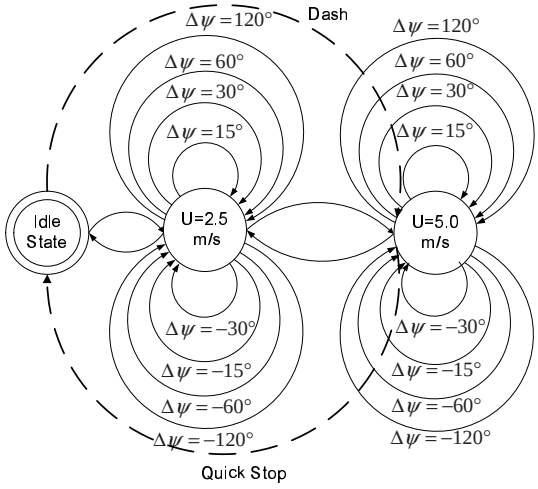

Fig. 2. AUV dynamics in MA representation

Table 1. Manoeuvre Library, $q = 5\ m/s$

P(index)	$\Delta T(s)$	$\Delta x(m)$	$\Delta y(m)$	$\Delta\psi(\circ)$
P_{15}	5.5	27.2	3.3	15
p_{30}	8.2	38.5	12.2	30
p_{60}	6.6	26.8	15.8	60
p_{120}	23	-8.4	93.4	120

4. QUASI RANDOM RAPID-EXPLORING RANDOM TREE

The approximate cell-decomposition methods such as A^*, dynamic programming and breath-first search are highly susceptible to the curse of dimensionality. Therefore, it is reasonable for one to concentrate on randomised algorithms. These algorithms do not have the completeness [2] and optimality properties of the previous algorithms. However, their robustness to the "curse of dimensionality" tends to make them preferable in practical and real-time applications. One version of this algorithm is the RRT (Lavalle, 1998). It is a form of incremental stochastic search technique that has been devised to search efficiently nonconvex high-dimensional state space.

4.1 Quasi Random Generator

Here, the quasi random (sub-random) generator based on the Halton sequence (Halton, 1960) is utilised instead of a pseudo random generator. Theoretically, the former generator possess certain desirable properties such as low discrepancy and improved uniformity over the sampling space. The generation of an element of a one-dimensional Halton sequence within the interval $[0, 1]$ is calculated using Equation 11 and Equation 12. Different prime numbers starting from the smallest are used for the multi-dimensional sampling case.

$$x_i = \sum_{k=0}^{\infty} n_{k,i} p^{-k-1} \tag{11}$$

with $i > 0$,$p = 2$ and $n_{k,i}$ determined by the following equation:

$$i = \sum_{k=0}^{\infty} n_{k,i} p^k; \quad 0 \leq n_{k,i} \leq p; \ n_{k,i} \in \mathbb{N} \tag{12}$$

4.2 Motion Planning Algorithm

Frazzoli (2002*b*) advocates enhancing the RRT algorithm by fusing it with the MA to solve motion planning problem with obstacles. The algorithm assumes that one has an embedded planner, that can plan an optimal trajectory in an obstacle free environment between two arbitrary states (Equation 9). The approach in this paper is that of multiple nested nodes. Since every state in a trim trajectory can be considered as a starting point of a manoeuvre. This algorithm generates child nodes at every connection point between a trim trajectory and manoeuvre. This improves the RRT branching capability, thus increasing the probability of finding a solution. A brief explanation of the the algorithm with reference to figure 3 is outlined below:

(1) Generate a subgoal (R1) using the quasi-random generator and attempt to connect to it using the embedded planner based on a minimum time criterion.
(2) If there is no collision, then generate an edge with new vertices at all interconnecting points of trim trajectories and manoeuvres. For all the new vertices, attempt to connect directly to the goal (greedy algorithm).
(3) If failed, generate another random subgoal (R2). Sort the shortest time trajectories to R2 from all vertices in an ascending order and attempt to connect to it. Apply this to only the first few near-optimal trajectories to avoid vertices saturation.

[2] A property where the algorithm will return a solution if such a solution exists

(4) The whole process is repeated until a feasible trajectory to the final state is found, maximum vertices size or time limit is reached. Figure 3 shows that vertex (nc1) has connected successfully to the final state.

4.3 Error Mitigation

Few researchers have expressed their concern regarding the prescribed error generated by RRT algorithm. Due to the discretised nature of the inputs in the original RRT algorithm, when the input history is applied, there will exist some errors in the final state. Hence, Kim and Ostrowski (2003) attempt to circumvent the problem by introducing a subconnection process. Similarly, Cervern *et al.* (2004) introduces error mitigation scheme to reduce the error caused by the concatenation effects. The former approach relies on "integrating" the dynamic model using the acquired input history to ascertain the final state. One disadvantage, in this approach is the requirement of an accurate dynamic model of the system. This might not be true in practice, due to model complexity, or nonexistence of a mathematical model. In fact, this error can be considered as a form of disturbance, and a robust controller can be designed to track the nominal trajectory instead. The design of the tracking controller is non-trivial due to the multi-input-multi output (MIMO) and underactuated behaviour of the AUV, as such it will be addressed in the near future.

4.4 Time-varying final condition

The case of a time-varying final condition is particularly interesting. This problem is commonly met when an AUV is conducting an interception mission such as docking with a moving mother submarine (Tan *et al.*, 2003). The problem of addressing the time-varying final condition using RRT was first pursued by Cervern *et al.* (2004). His approach is based on embedding time variable into the system state vector. Evidently, the immediate effect is the increase in the state vector dimension. A simpler solution proposed here is to adopt an iterative subroutine commonly known as the "false-position" method. Fundamentally, assuming that the target is moving at a constant velocity, the concept is to use a predict-correct process to converge within a tolerance of the final state. Nonetheless, there is no guarantee of convergence, thus an upper bound to the iteration count is needed to terminate the loop as a contingency.

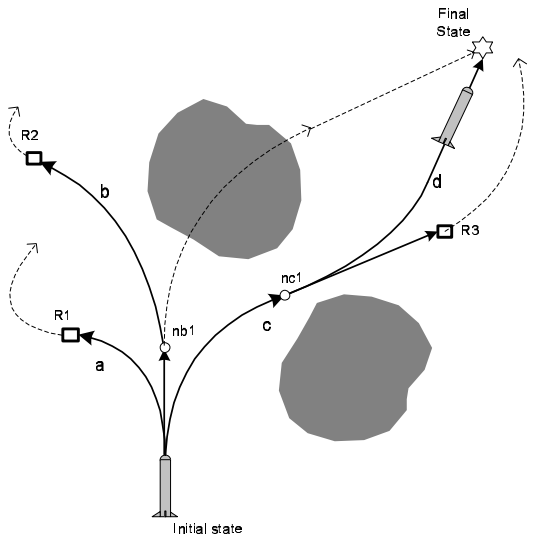

Fig. 3. Simplified illustration of the RRT Algorithm operation

5. SIMULATION RESULTS AND DISCUSSIONS

The algorithm is implemented in MATLAB with the GNU Linear Programming Kit ver. 4.4 (GLPK) [3] in an 2.1 GHz Pentium IV machine, with 512 MB of RAM and running Windows XP. The environment, based on the North-East-Down coordinate, is set to $300 \times 300m$ in dimension. The simulations assume an ideal case where *a priori* information of the environment is provided and there is no external disturbance from the environment. The simulations are run with 200 maximum nodes, 300 maximum iterations, and a 2 seconds time constraint, terminating when either criterion is reached or if a solution is found.

Pertaining to simulation 1, the AUV initial state is set to [1 1 0.1], while the final state is [170 145 κ] where κ denotes an unconstrained variable. Here the individual variables are [x y ψ], displacements in meters and heading in radians, the goal tolerance is defined as a $7m$ radius. Figure 4 illustrates one of the trajectories found by the algorithm. The dotted lines are candidate trajectories where the continuous line is the feasible trajectory. The triangles symbolise the AUV, enlarged twice for reason of clarity. The numeric values denote the vertices. For simulation 2 (Figure 5), the final state is set to [150 70 2.2] and moving at 2 m/s, simulating a moving submarine. Again, a trajectory is found as depicted in Figure 5.

Due to the probabilistic nature of the algorithm, a sample of 100 simulations are run to compile the statistics (Table 2). From the median statistic, it is observed that the majority of the solutions are less than a fraction of a second for both simulations. The failure rate of simulation 2 is higher due to the time varying state issue. Moreover, in certain

[3] Obtained from http://www.gnu.org /software/ glpk/ glpk.html

cases, the AUV will intercept the target in a head on position instead of a tail-chase fashion. This is prevented by wrapping the heading angle and constraining the final state heading.

6. CONCLUDING REMARKS

The primary objective of this paper is to verify the feasibility of employing the MA representation and the RRT algorithm to an AUV to solve the motion planning problem. The simulation results obtained are very encouraging, although additional detail studies are warranted. Its very short computational time makes it an ideal algorithm for real-time applications. Additionally, a simpler algorithm for solving time-varying final state has been proposed. As aforementioned, the algorithm is intrinsically a feedforward controller, therefore a robust low-level feedback controller will be designed in the near future to track the prescribed trajectory.

Table 2. Statistics from 100 samples run

Statistics	Sim. 1	Sim. 2
Average time taken,s	0.41	1.22
Median,s	0.27	0.56
Standard deviation,s	0.25	0.41
Success rate	88/100	68/100

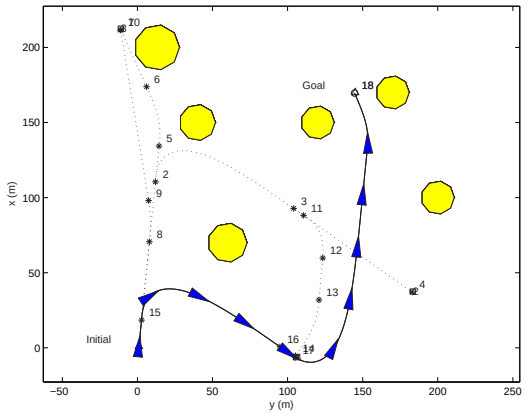

Fig. 4. Simulation 1. Environment with static obstacles

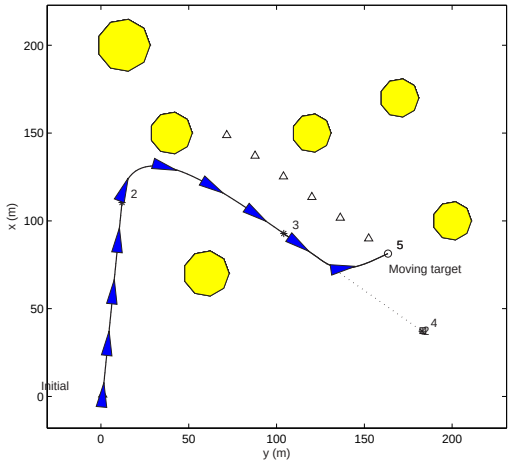

Fig. 5. Simulation 2. Environment with time-varying final state

REFERENCES

Cervern, W. T., F. Bullo and V. L. Coverstone (2004). Vehicle Motion Planning with Time-Varying Constraints. *Journal of Guidance, Control, and Dynamics* **27**, 506–509.

Frazzoli, E. (2001). Robust Hybrid Control for Autonomous Vehicle Motion Planning. *P.h.D Thesis, Massachusetts Institute of Technology, Cambridge, MA*.

Frazzoli, E. (2002*a*). Manoeuvre-Based Motion Planning And Coodination for Single and Multiple UAV's. *Proc. of AIAA/IEEE Digital Avionics Systems Conference* **2**, 8D31–8D312.

Frazzoli, E. (2002*b*). Quasi-Random Algorithms for Real-Time Spacecraft Motion Planning and Coordination. *Acta Astronautica* **53**, 485–495.

Frazzoli, E., Munther A. Dahleh and Eric Feron (1999). Hybrid Control Architecture for Aggressive Maneuvering of Autonomous Helicopters. *Pro. of The IEEE Conference on Decision and Control* **3**, 2471–2476.

Halton, J. H. (1960). On the Efficiency of Certain Quasi-random Sequences of Points in Evaluating Multi-dimensional Integrals. *Numerical Mathematics* **2**, 84–90.

Kim, J. and J. P. Ostrowski (2003). Motion Planning of Aerial Robot using Rapidly-exploring Random Trees with Dynamic Constraints. *IEEE Int. Conference on Robotics and Automation*.

Lavalle, S. M. (1998). Rapidly-exploring Random Trees: A New Tool for Path Planning. *TR 98-11 Computer Science Dept., Iowa State University*.

Millard, N.W., G. Griffiths, Finegan, S. D. McPhail, D. T. Meldrum, M. Pebody, J.R. Perrett, P. Stevenson and A.T. Webb (1998). Versatile Autonomous Submersibles-The Realising and Testing of A Practical Vehicle. *Underwater Technology* **23**(1), 7–17.

Prestero, T. (2001). *Verification of a Six-Degree of Freedom Simulation Model for the REMUS Autonomous Underwater Vehicle, Master Thesis*. MIT. Massachusetts, USA.

Schouwenaars, T., B. Mettler, E. Feron and P. Jonathan (2003). Robust Motion Planning Using a Manoeuvre Automaton with Built-In Uncertainties. *Proc. of the American Control Conference* **3**, 2211–2216.

Tan, C. S., R. Sutton, J. Chudley and S. Ahmad (2003). Autonomous Underwater Vehicle Retrieval Manoeuvre Using Artificial intelligence Strategy. In: *Proc. of GCUV' 2003 Conference*. IFAC. Newport, UK. pp. 143–148.