Toward Autonomous Robotic Containment Booms

Visual Servoing for Robust Inter-vehicle Docking of Surface Vehicles

Young-Ho Kim[*], Sang-Wook Lee[**],
Hyun S. Yang[***], and Dylan A. Shell[*]

[*]Department of Computer Science and Engineering,
Texas A&M University, College Station,Texas,USA,
{yhkim,dshell}@cse.tamu.edu
[**]Robotics Program,
Korea Advanced Institute of Science and Technology, Daejeon, S.Korea,
{sangwook}@paradise.kaist.ac.kr
[***]Department of Computer Science,
Korea Advanced Institute of Science and Technology, Daejeon, S.Korea,
{hsyang}@paradise.kaist.ac.kr

Abstract

Inter-vehicle docking is the problem of coordinating multiple robots to actively form and maintain physical contact. It is an important capability for Autonomous Surface Vehicles (ASV) and is an essential part of a wide class of missions. This article considers one such mission: the emergency response and environmental protection problem of containing a floating pollutant. We propose a solution in which multiple robots autonomously navigate so as to surround the surface matter. Before doing so, the robots dock with one another in order to secure specialized attachments designed to ensnare the contaminant. We describe the prototypical physical robot system developed to perform this task, and we detail the system architecture, sensing and computational hardware, control system, and visual processing pipeline.

While employing multiple ASVs maximizes spatial reconfigurability, it depends on the inter-robot docking capabilities being particularly reliable. But achieving robust docking is a significant technical challenge because the water continually induces external disturbances on the control system. These disturbances are non-stationary and almost impossible to predict for unknown environments. Our system relies primarily on visual servoing within a control framework in which a variety of sensors are fused. Accurate disturbance measurements are obtained through traditional sensor modeling and filtering techniques. As the environment is *a priori* unknown, varies from trial to trial, and has proven difficult to model, we apply a model-free reinforcement learning algorithm, SARSA(λ), along with specialized initial conditions which ensure stable operation, and an exploration guidance approach that increases the speed of convergence. We adopt a two-loop control scheme for visual servoing to successfully make use of feature descriptors with various (and variable) computational times. We demonstrate this approach to the docking problem with autonomous ground vehicles and autonomous surface vehicles. The results from several situations are compared, showing that disturbance rejection coupled with SARSA(λ) is an effective approach.

1 Introduction and Motivation

Since the burgeoning of the research area in the 1990's, Autonomous Surface Vehicles (ASVs) have been studied extensively for use in industry, military, and scientific research settings [Bertram, 2008]. Recent examples of missions for surface vehicles capable of autonomous navigation and way-point following include work in environmental monitoring [Wang *et al.*, 2009], investigation of hazardous environments [Murphy *et al.*, 2011], and rescue applications [Clauss *et al.*, 2009]. This paper describes a containment problem which we believe to be a unique, novel, and important application domain for ASVs. Two recent catastrophic events brought this area to light and motivated the present research:

Hebei Spirit oil spill On December 9th 2007, a collision between a barge and heavy tanker *Hebei Spirit* led to spill of crude oil off the coast of Taean, Chungcheonam Province, South Korea. A total of 10 800 tonnes of oil were spilled devastating coastal farms, fisheries, natural wetland areas, and public beaches [BBC, 2007]. In all, a total of more than one million volunteers assisted in the clean-up effort. The clean-up was complicated by abnormal weather in the critical, early stages of disaster recovery effort. This meant that the oil's spread along the west coast of the country took longer halt than otherwise would have been the case.

Deepwater Horizon oil spill An explosion on the *Deepwater Horizon* drilling platform killed 11 men and injured 17 others on April 20th 2010. The uncapped sea-floor oil well resulted in between 492 000 and 627 000 tonnes of oil spilling into the Gulf of Mexico [NOAA, 2010]. During the four months following the explosion a variety of short-term efforts were attempted in order to cap the well. (Although sometimes referred to as the 2010 oil spill, there were actually eight other oil spills that same year; the total number of oil spill disasters is astonishing.)

In the days immediately following both disasters, oil containment booms were deployed to corral the floating oil. These booms are passive devices, usually consisting of multiple connected inflatable sections, which float on the water surface in order to form a physical barrier to contain floating contaminants. Although the booms are not without limitations (*cf.*, Grier [2010] and Doerffer [1992]), it has been claimed that deployment logistics—rather than quantity of boom—was the primary limiting factor in the Deepwater Horizon spill containment effort [Sullivan, 2010]. In addition to the established and documented best practices (see Chapter 6 of Fingas and Charles [2000]), the importance of such containment booms has spurred academic interest from researchers (*e.g.*, Fang and Wong [2001], and Fang and Wong [2006]).

This paper describes a prototypical multi-robot system for deploying floating containment booms. We propose a distributed approach in which several ASVs incrementally form a chain around the floating contaminant. Figure 1 is an artist's rendering of the concept. Individual robotic units are deployed in parallel and, potentially, in a number of different ways (*e.g.,* either launched from a single vessel, thrown from several separate manned craft, released from the shore, or a combination of these options). Irrespective of their initial arrangement, the self-propelled units maneuver in order to position themselves for maximum advantage. Next, a process of chain construction occurs. Pairs of robots dock with one another in an operation that causes their boom connectors to interlock. As the units move apart, the connecting booms unreel to their full length. This chain formation process starts from one end and proceeds incrementally until the whole chain has been formed. The units, physically coupled in this way, maintain the chain and their propulsion still permits collective control of the barrier.

Effective distributed deployment of floating booms in the manner envisioned depends critically on the inter-robot (or inter-vehicle) docking capability. In this work we demonstrate that robust autonomous docking between two ASVs via visual servoing is indeed practicable, but that several techniques must be integrated to address issues that arise due to image noise, dynamic and unpredictable environments, and the distinct timescales of different feature detectors and the control loop. Feature-independent Image-Based Visual Servoing (IBVS) is employed to effectively estimate the image Jacobian matrix and the target robot's pose while maximizing generality. Robustly obtaining this information, including the target's distance, permits local path planning to best pick the most effective docking direction and can also be employed for following behaviors.

The disturbances caused by wind, wave, and current represent a significant challenge to precise control of the small sized ASVs considered here. To address this, our approach employs two-loop control system with a disturbance rejection method operating within the inner loop. However, one issue is that these external disturbances occur more or less continually and are non-stationary so that prediction remains challenging, especially if the environment is not known beforehand. We employ an on-line reinforcement learning method to adapt to the unknown environment, account for variable conditions, and achieve reliable performance. To achieve adequate operational performance, we employ initial conditions which give satisfactory starting performance and introduce an action exploration strategy that is directed in order to speed convergence.

The controller scheme and methods proposed in this paper are applied to real vehicles. The docking approach is validated through demonstrations with both Autonomous Ground Vehicles (AGVs) and ASVs. The AGVs represent a form of control-case because they permit evaluation without water induced disturbances and they allow random noise to be introduced artificially in a parameterizable way. The ASVs employed were custom designed prototypes for the containment boom scenario and illustrate the effectiveness of the robust visual servoing approach in unpredictable, dynamic environments, with thrusters for actuation, and under more realistic marine conditions. But the docking capabilities and the methods introduced for robust disturbance rejection are applicable more widely than the deployment and containment scenario we consider here. Both are important for many of the task domains already mentioned. Moreover, in the coming years, additional focus will likely be placed on problems which involve tightly coupled coordination, and the importance of these techniques is expected to grow.

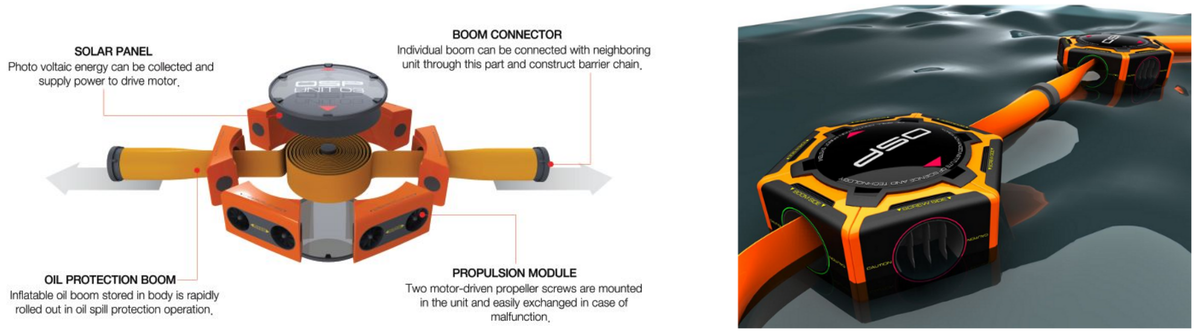
SOLAR PANEL
Photo voltaic energy can be collected and supply power to drive motor.
BOOM CONNECTOR
Individual boom can be connected with neighboring unit through this part and construct barrier chain.
OIL PROTECTION BOOM
Inflatable oil boom stored in body is rapidly rolled out in oil spill protection operation.
PROPULSION MODULE
Two motor-driven propeller screws are mounted in the unit and easily exchanged in case of malfunction.

Figure 1: An artist's depiction of the robotic containment boom. Multiple individual robots dock with one another in order to incrementally unfurl the inflatable boom in order to surround and corral the floating pollutant.

The remainder of this paper is organized as follows: Section 2 describes the relationship between this and existing work. Section 3 briefly describes the physical and logical system architecture. Section 4 explains the integrated visual servoing and control scheme. The experiments in Section 5 demonstrate the performance with both ASV and AGV robots; the data are discussed in that section too. The final section, Section 6, concludes the paper.

2 Related Work

We are not aware of any prior work that considers autonomous deployment of booms, surface pollutant containment, or incremental formation of tethered chains by surface vehicles. The concept of distributed oil clean-up was recently proposed by Kakalis and Ventikos [2008]; while that work includes conceptual pictures of "vacuum cleaner" like robots for collecting oil; we are not able to assess the viability of that concept for use with physical robots. The approach we propose is based on current oil spill management practice and is applicable to general containment of a variety of matter (*e.g.*, floating ice). Additionally, it can be sensible to deploy these boom devices as a precautionary measure: their long-term effectiveness does not depend on significant movement or ensuring wide-spread coverage.

The broad advantages that multi-robot systems have over single robot systems are directly applicable to ASVs: *viz.* robustness through redundancy, performance speed-up through parallelism, expanded collective capability through synergy, and ability to exploit the distributed nature inherent to certain tasks [Parker, 2008]. But within the literature there remain comparatively few demonstrations of multiple autonomous surface platforms. By way of examples: Benjamin *et al.* [2006] implement and demonstrate US Coast Guard mandated rules for prevention of collisions, and evaluate their implementation on multiple ASVs. Curcio *et al.* [2005] developed the SCOUT (Surface Craft for Undersea and Oceanographic Testing) surface vehicle as a platform for multiple vehicle operations algorithm testing, including underwater craft. The majority of multi-ASV work does not consider tightly-coupled coordination tasks where physical contact is necessary.

Within the broader autonomous marine vehicle literature, docking is an important instance in which a direct physical coupling is sought. Often underwater craft that explore the ocean floor, for example, may only recharge their batteries and transmit data when docked—either to a seabed station or an ASV. Murarka *et al.* [2009] and Park *et al.* [2009] both use structured light for docking their Autonomous Underwater Vehicle (AUV). The light serves as a reference signal which helps the robots approach the docking target. This is a particularly shrewd choice of positioning signal because the water's filtering and attenuation can reduce the effect of external elements such as sunlight, illumination changes, and artificial light, which cause interference on the surface. The present study employs vision-based methods for inter-robot docking; we focus on the relationship to work employing such methods in the remainder of this section. It is worth noting, however, that alternative sensors have been used for docking (*e.g.,* Kim *et al.* [2009] use RF signals), but this leads to a new set of challenges.

Docking via Visual Servoing

Vision-based systems have been previously been shown to be effective for docking an ASV with an AUV. Martins *et al.* [2007] and Dunbabin *et al.* [2008] are good examples of coordinated ASV to AUV docking. That work makes creative use of their particular hardware: a vision camera is fixed to the roof of the ASV and special angles are used to maximize detection of the AUV's salient color. In this work, we are unable to use the same configuration since the robots we employ are homogeneous and the focus on inter-robot docking means that no particular robot is privileged so as to form a special drifting target. The solution employed in the present study—detailed in the following section—is to employ visual feature descriptors and feature matching. This proved to be effective at accurately determining the distance to a target object and it was less sensitive to lighting changes and more robust to scale variance than color segmentation (see Section 5.1). The majority of examples in the literature that successfully employ black and white, and color segmentation methods are for submerged operations (*e.g.,* underwater line-following [Bibuli *et al.*, 2008; El-Fakdi and Carreras, 2008]). The fundamental trade-off is that feature-based techniques are more computationally expensive; the control architecture we employ, however, overcomes the latency which arises from the slower detection and tracking.

Researchers attempting to perform visual servoing must address the issues of measurement noise, the limited computational time, and imperfect system models. There are two broad types of approach: Position-Based Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS) [Hutchinson *et al.*, 1996]. Generally speaking, PBVS is robust to image noise but object information is required to predict the object pose; PBVS is particularly sensitive to camera calibration and usually requires a sophisticated model of the target (*e.g.*, a CAD model). In contrast, IBVS does away with those requirements. It is largely robust to camera calibration error, but the image Jacobian matrix depends on both the camera and the 3-D coordinates of the feature points. (Section 4.4 describes how these elements are computed in the IBVS method we use.) SIFT [Lowe, 2004] based trackers, like those employed in this work, have been used in dynamic multi-vehicle settings before. Most closely related is the system that Giesbrecht *et al.* [2009] developed as a step toward autonomous ground vehicle conveys. They address the problem of autonomously following a leader and demonstrate this with on- and off-road vehicles. They face the same problem of comparatively slow feature tracking, recognizing that obtaining data at the proper frequency is critical for the system to work well. Their approach is to reduced the search space so as to save computational time. Instead, this work uses an enhanced SIFT detector within a two-loop control architecture for controlling the ASV (see Section 4.2).

Modeling Environmental Disturbances via Reinforcement Learning

Prior work on ASV-to-AUV docking presumes that the latter is drifting on the water, and the water itself is assumed to be calm. In this work we are concerned with docking quickly with both a poor *a priori* understanding of the environment and potentially changing ocean conditions. Thus, the proposed method applies an on-line learning approach to the ASV control problem. We illustrate how tabular SARSA(λ) [Sutton and Barto, 1998] will achieved convergence. Important related work is the visual servoing control method of Martinez-Marin and Duckett [2005], who apply IBVS and reinforcement learning in a grasp positioning scenario. We believe Gaskett *et al.* [1999], who employed Q-learning and a neural network, were the first to apply reinforcement learning to AUVs. In contrast to our model-free approach, Pereira *et al.* [2008] employ a simplified model of ASV dynamics in order to perform station keeping with an under-actuated system. Fahimi [2007] develops sliding-mode control for ASVs, but relies of assuming disturbances are always of some maximum amplitude. A learned model, like that which we employ, will likely result in better performance in a particular environment because the model can adapt to the structure of that environment by learning that structure *in situ.*

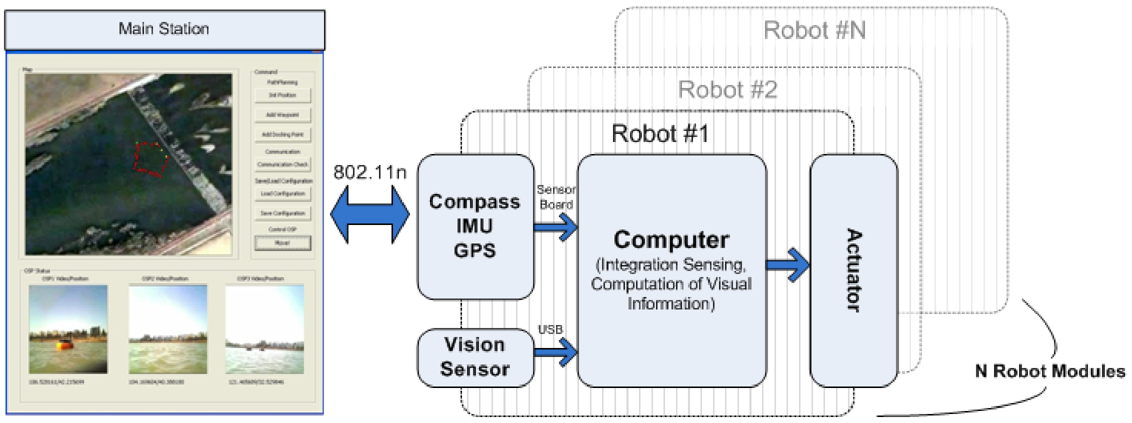

Figure 2: Physical and logical architecture of the overall containment boom deployment system.

3 Physical System Architecture

We describe the overall architecture of an approach we envision for autonomous containment boom deployment. Physically, the system consists of a single monitoring station and multiple autonomous robots as shown in Figure 2. The monitoring station receives information about the spill configuration and transmits target positions to each of the robots via wireless communication. These way-points (in GPS coordinates) are used both during initial positioning before the chain has been formed and after booms are deployed to actively corral the floating oil. Each robot is autonomous, locally filtering and fusing several sensors, estimating properties of the environment, and capable of docking with other units without any external low-level guidance.

Each robot periodically transmits its estimated location back to the monitoring station so as to update the visual display; this information is used by the operator to assess the navigation progress of each unit with respect to the oil. The window in Figure 2 shows a simple interface in which a region is selected with the mouse via rubber-band-like process. Target way-point locations are computed by dividing the circumference appropriately. Once satisfied with the relative position of robots and contaminant, the operator sends a chain formation command, which kicks off a process of docking around the perimeter. The robots dock in the following manner. An initially selected robot docks with the robot nearest to it along the boundary. Those two robots separate and the boom unfurls between them. Those two robots then hold their positions in order to become docking targets for their neighboring robots. The process continues until all the booms have been unfurled. Each robot adapts to its local environment, predicts its own position, and maintains its own status. When the docking operation is triggered, the robot which still has no boom unrolled is given only the GPS coordinate of the other unit. Although the error in this coordinate is too large for docking directly, it is precise enough to allow the robot to get the target within its visual field of view. Visual servoing is used to obtain the precision necessary to complete the docking procedure.

The monitoring station manages information about the containment area but relies on each robot to autonomously perform its docking. Apart from the high-level directives, the station serves to keep track of the overall scenario and watch the development of the chain. This paper demonstrates that this architecture is logical and feasible because the robots can receive high-level docking commands and autonomously perform the necessary low-level controls to carry them out.

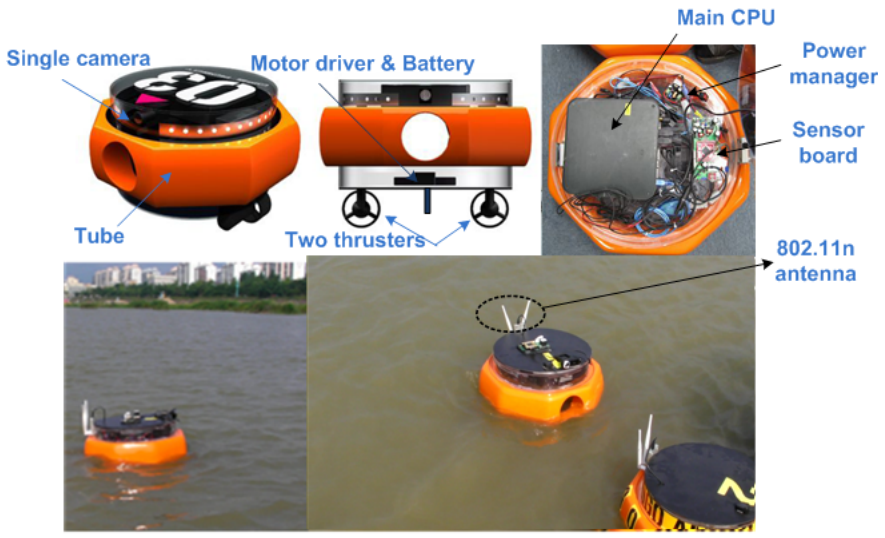

Figure 3: Autonomous Surface Vehicle designed as a prototype system in order to assess the feasibility of a robotic containment boom deployment. These units are the primary platform used for implementation, development and validation of the robust visual servoing and docking behaviors.

Consequently, the bulk of the paper concentrates on visual servoing in order to dock in dynamic, unpredictable environments.

3.1 Prototype Hardware Units

The prototype ASV we developed is shown in Figure 3. This vehicle is the principal experimental platform for the development and testing of robust docking. Each ASV has three parts: an upper body, a middle body and lower body. Upper body includes the main CPU, sensor board, and power manager. The middle body add buoyancy to ensure that the ASV stays afloat. The lower body houses the motor driver and battery; these heavy components ensure the proper center of gravity. The single camera is positioned at the midpoint of the front side pointing forward. The two thrusters are positioned so that when both have equal power the unit moves forward, *i.e.,* in

Figure 4: Test navigation of a single unit on a calm day.

the same direction as the camera is pointing. Figure 4 shows the wake pattern of the device while navigating. The hardware provides a selection of sensors which are ultimately fused in software to order deal with unpredictable dynamic environments; environmental state estimation is managed by two micro-controllers (Atmega128, TMS320F28335) which mange sensors at a variety of sampling rates (a) GPS (66ch, DGPS): 500ms, (b) Compass (HMC6352, 2-axis): 250ms, (c) IMU (ADIS16350 with 3-axis accelerometer and gyro): 100μs. The main CPU, an Intel core 2 duo at 1.8GHz, performs the vision processing and manages high-level control.

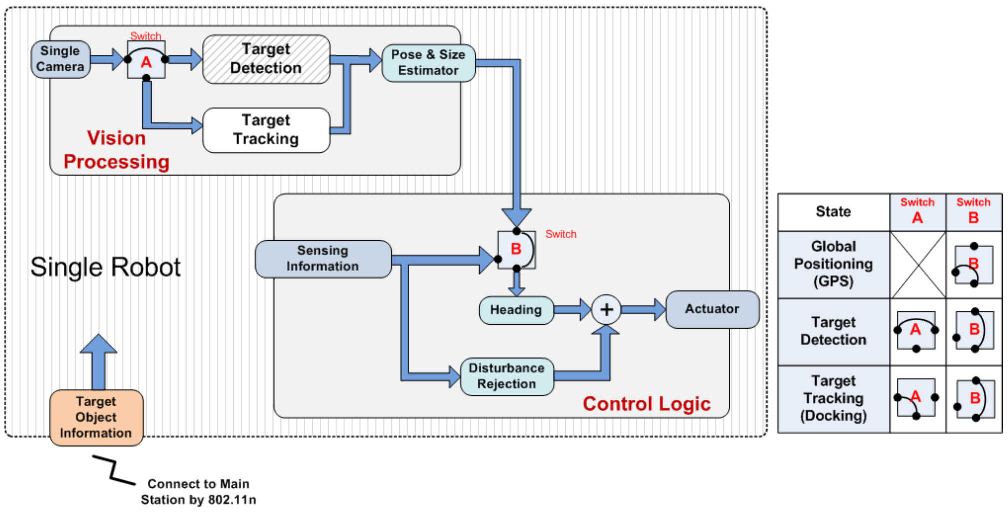

Figure 5: Each robot's controller software consists of two modules: the *Control Logic Module* and the *Vision Processing Module*. The two switches labelled *A* and *B* permit separate activation and deactivation of the modules. A sequence of switch settings allows robust docking with neighbor robots to be performed.

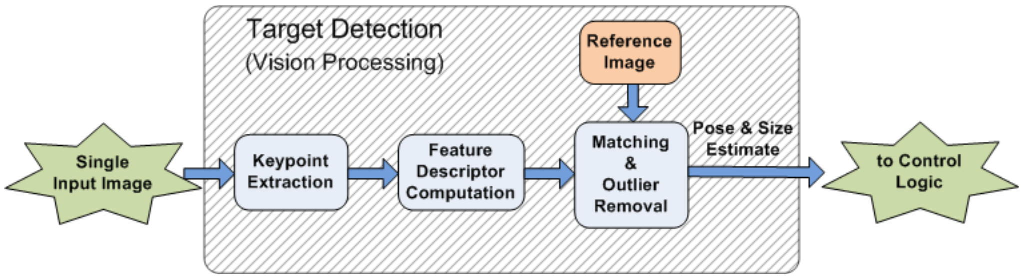

Figure 6: The Target Detection procedure within the Vision Processing Module follows a three stage pipeline, with each stage performed sequentially: (1) keypoints are extracted from the current image, (2) feature descriptors are computed for each keypoint, and (3) a match to the previous image is found and outliers discarded.

4 Integrated Visual Servoing Control Scheme

Each robot runs controller software that is cleanly decomposed into two modules. Figure 5 shows the two modules: the *Control Logic Module*, the *Vision Processing Module*, and their interactions. Three separate modes of behavior are realized by activating the modules in different ways via the two switches labelled *A* and *B*. The Control Logic Module is the primary procedure for controlling navigation of the ASV. Its internal logic and novel integration of several components are described in Section 4.2. The Visual Processing Module is responsible detecting and tracking other robots.

Initially the robot attempts to navigate toward the target as assigned to it by the monitoring station. The Control Logic Module will operate in a global positioning mode, wherein input from the Vision Processing Module is unnecessary. Uncertainty due to imprecision in the GPS signal means that the robot cannot dock with that position information alone. Consequently, Vision Processing Module must detect a target robot. After successful detection, the Vision Processing Module provides tracking information to the Control Logic Module, which it uses to navigate toward the target. While doing this, other aspects of the control system (like disturbance compensation, for example) are operating concurrently.

4.1 The Vision Processing Module

The Vision Processing Module operates first in Target Detection mode and then transitions to the Target Tracking mode during the final phase of tightly coupled docking. Figure 6 shows the sequential pipeline involving three main procedures for the detection mode of operation: (1) keypoint extraction; (2) feature descriptor computation, and (3) matching to a reference along with outlier removal. Particular experiences and implementation choices are described for each of these next, along with the details of operation within the tracking mode.

Keypoint Extraction Given an input frame from the camera, the first operation is to extract features for subsequent processing. Three requirements must be met: we must be able to identify and extract features reliably, they must persist (or reoccur) across frames at least for some short time-window, and processing must be efficient. These requirements become most important when addressing the challenge of target detection under conditions arising from simultaneous target and camera motion in a dynamic environment. We implemented Rosen and Drummond's FAST keypoint extractor. As shown in Rosten and Drummond [2006], the method has the benefit of computational efficiency which enables high-speed corner detection.

Feature Descriptors The keypoints identified in the preceding step are particular points in the input image. The property of invariance to changes in scale and orientation ensures that similar views of the same scene will produce overlapping keypoint sets. A richer descriptor for each keypoint must be computed in order to reliably associate the points between frames. Ideal feature descriptors will be maximally distinctive while being invariant to artefacts arising from the camera and environment. In this work, robustness to changes in illumination turned out to be especially important. We employed SIFT [Lowe, 2004] descriptors, which are invariant to image translation, scaling and rotation, and partially invariant to illumination changes and robust to local geometric distortion.

Matching As a final process of target detection, a reference image's feature points and an input frame's feature points must be correctly matched. This process is necessary in order to estimate the target object's position and to remove outliers. Iterative matching for each point, while capable of finding the best possible matching, is computationally costly and would adversely affect the performance of the entire vision pipeline. Computational savings result from selecting PROSAC [Ondřej Chum and Jiri Matas, 2005] over RANSAC (Random Sample Consensus) [Fischler and Bolles, 1981]. Significant speed results from exploiting a linear ordering structure for the matching, which requires assumptions that are reasonably met in operation (a similarity measure must predict correctness of a match better than random guessing).

Tracking The detection mode seeks to recognize, identify, and locate the target object by processing a single image in isolation. Since the detection mode considers all the pixels within the image, significant processing cost (in terms of time) is incurred, which motivated our development of the tracking module. The tracking module is designed to exploit spatio-temporal locality: once the target object has been detected, the next target position is likely to be near the previous one. The system architecture includes a mechanism to permit smooth transition from detection mode to tracking mode and back again. Figure 5 shows this with the symbol marked Switch A; it obeys the following logic: when the target detection module succeeds in determining the target object's position, that information is provided to the tracking code for processing subsequent frames; if the target object is missed in target tracking mode, the vision processing is switched back to detection behavior as and when needed. Thus, the tracking mode serves to complement the detection mode. Our implementation uses a simple Mean-Shift Matching [Comaniciu *et al.*, 2002] to track a target object. This method uses the color histogram of the object in the previous image to find the target object near the object's past position. The simplicity of the method made it an attractive candidate for implementation and it proved to be adequate for the salient orange robots.

4.2 The Control Logic Module

The Control Logic Module is responsible for controlling the navigation of the ASV and is composed of the most critical elements of the control system. Figure 7 gives a graphical representation the controller. The figure shows three control loops: (1) the inner control loop, which is always active; (2) outer control loop-A, which is responsible for visual servo control and only active during that mode of operation ; and (3) outer control loop-B, which is the global positioning control logic and is active while the robot is approaching the target from afar. Since exactly two control loops are active at any one time, we term this a two-loop switching architecture. The two control loops are switched when the robot is ready for docking.

The use of two loops permits the control system to minimize latency even though the time required for processing vision data can be significant. The visual servo control (marked as "outer control loop-A" reproduced in Figure 7) detects and recognizes a target objects and estimates an object's pose. It does this at comparatively slow speeds considering the dynamics of the robot. However, simultaneously, the tracker in the inner control loop ("inner control loop" in Figure 7) tracks the object at a high speed. A primary objective of the architecture was to abstract away the particular choice of feature tracker and detector. This abstraction is only possible when the other responses of the system are not affected. Although not a perfectly clean abstraction, our approach enlarges the visual servo system's feature independence and does permit multiple features to be used. The result is that the logic ultimately improves docking the target object because we were

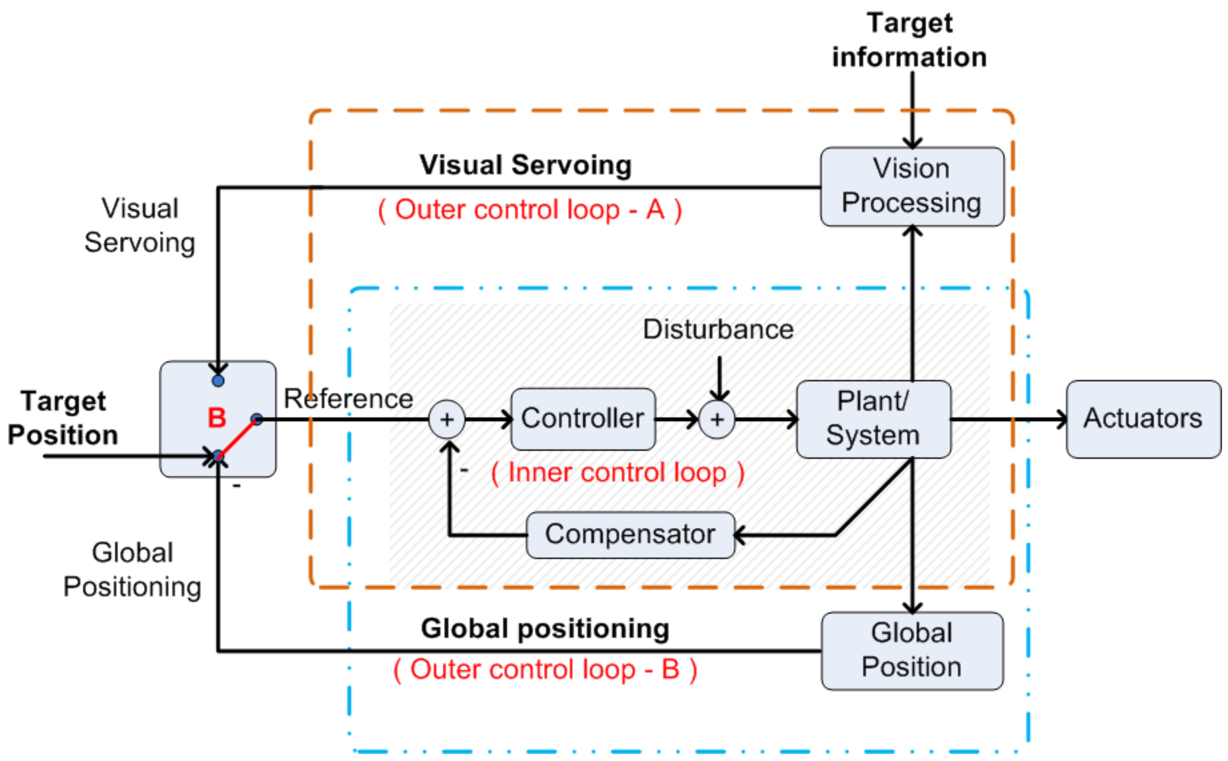

Figure 7: A block diagrammatic overview of the Control Logic Module: three control loops are visible. The inner loop is common and always active, but only one of the outer control loops is ever active at one time.

able to explore various feature choices.

On the other hand, in global positioning mode (marked as “outer control loop-B” in Figure 7) is applicable when approaching a target object from afar. Global position from the DGPS (Differential Global Positioning System) sensor has an uncertainty of almost ± 3m in real environments. So the control logic also helps in approaching the target object. The target’s position is communicated from main station, and the outer control loop computes heading updates only at the GPS sampling frequency (every 0.5s). The inner control loop (“inner control loop” in Figure 7) is able to apply disturbance rejection (along with reinforcement learning, as is explained at the next subsection) concurrently.

Thus, the inner control loop ensures accurate docking and way-point following. It manages to do this because of the box labelled “Local Position” in the block diagram, which includes a disturbance rejection mechanism (Compensator) to help overcome environmental dynamics. The “Compensator” employs high-frequency IMU data in order to address the effects of environmental disturbances, rather than relying on visual or GPS-based sensory feedback.

4.3 Disturbance Rejection

Figure 7 shows the inner control loop with a block labelled “compensator” which reduces the effect of disturbances in an effort to maximize the range of circumstances in which the system may robustly navigate. Navigation behavior is implemented through a combination of a traditional control scheme and Reinforcement Learning. The traditional control scheme involves a filtering approach that is similar to that employed by Caccia *et al.* [2008]: we apply a state-space model of the Inertial Measurement Unit (IMU) and Kalman filter to estimate accelerations and deal with measurements' noise characteristics since the measured accelerations contain errors such as drift. Even in relatively benign environments, prior work using Image-Based Visual Servoing for tracking (*e.g.,* Aull [2009]) has emphasized the need for filtering of target locations. Ultimately, the approach we employ relies on the use of linear accelerations to measure environmental disturbances.

In order to measure disturbance exerted by dynamic environments, a low-cost, strap-down, solid-state IMU is used in our system. The IMU consists of three accelerometers and three angular rate gyros. The three accelerometers measure linear accelerations in three perpendicular directions and three gyros measure angular velocities in the same directions. The gyros are used to estimate the robot's attitude and the estimated attitude is applied to compensate for the local gravity in the measured accelerations.

The local gravity can be calculated if its initial value or the robot's initial orientation is known. In case of the former, using the given initial gravity and rotational angles obtained by the integration of the angular velocities, the tri-axial components of the local gravity can be computed. In the latter case, the attitude of the robot system is dependent on the initial orientation because the angular rates from the gyros are integrated to get the orientation angles of the robot system. Thus, the initial orientation of the robot system is needed to estimate accelerations from dynamic environments more correctly. Therefore, we adopt the former approach in our system.

4.3.1 Inertial Measurement Unit Model

To define the IMU model for detecting and hence rejecting disturbances, we choose a state vector

$$\mathbf{x} = \begin{bmatrix} {\mathbf{x}_x}^\mathsf{T} & {\mathbf{x}_y}^\mathsf{T} & {\mathbf{x}_z}^\mathsf{T} \end{bmatrix}^\mathsf{T} \tag{1}$$

where $\mathbf{x}_m = [\mathbf{v}_m \quad \mathbf{a}_m \quad \mathbf{b}_{am} \quad \omega_m \quad \mathbf{b}_{\omega m}]^\mathsf{T}$ is a state vector and $\mathbf{v}_m$, $\mathbf{a}_m$, $\mathbf{b}_{am}$, ω_m and $\mathbf{b}_{\omega m}$ are velocity, acceleration, acceleration bias, angular velocity, and angular velocity bias of the robot with respect to the m-axis, where $m \in \{x, y, z\}$.

Using the state vector defined above, state-space equations for the IMU are given by [Farrel and Barth, 1998; Grewal *et al.*, 1991]:

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \mathbf{G}\mathbf{u}(k) + \mathbf{n}(k), \tag{2}$$

$$\mathbf{y}(k) = \mathbf{H}\mathbf{x}(k) + \mathbf{v}(k). \tag{3}$$

(The details of these equations appear in Appendix A.)

In the above equations, we assume all the process noises are uncorrelated with one another, and accelerations and angular velocities along x-, y-, and z-axes can be observed independently. In order to overcome the biased and non-stationary behavior of the IMU and to estimate the magnitude of errors, the bias terms, b_{am} and $b_{\omega m}$ related to the accelerometers and gyros, are modeled as undergoing a random walk and considered as states.

As the state-space equations show, the resulting process and measurement equations are all linear, and the error states (b_{am} and $b_{\omega m}$) and their relevant true states (a_m and ω_m) are coupled only through the measurement equations but not process equations.

The accelerations and angular velocities estimated from these equations are used to compute the robot's orientation and to estimate the environmental disturbances as measured by accelerations. As the noise associated with the accelerometers and the gyros is spatially Gaussian and temporally white, and the overall system equations are linear, in our approach we use Kalman filtering in order to smooth noise.

4.3.2 IMU Kalman Filter

Kalman filtering [Welch and Bishop, 2006; Grewal and Andrews, 2001] is a Bayesian filtering approach and the optimal (minimum variance) filter for linear discrete systems that consist of difference equations like Equations (2) and (3).

Through Kalman filtering, the tri-axial accelerations and angular velocities are estimated. Then rotational angles are obtained by integration of the estimated angular velocities are used to compute three perpendicular components of the local gravity about the robot-attached coordinate frame. These estimated accelerations are used in order to measure environmental disturbance.

4.3.3 Initial Gravity

For calculating tri-axial components of the local gravity about the robot-attached coordinate frame, gravity at the robot's initial attitude, which is called the initial gravity, is required. To do this, we follow the idea of Joshi *et al.* [2010]. They assumed that if the robot initially has no external forces on it other than gravity, then the measured accelerations are Gaussian distributed about the constant force of gravity such that the mean direction of acceleration is an initial direction of gravity:

$$\mathbf{g}_0 = \frac{1}{N} \sum_{i=1}^{N} \mathbf{a}(n) \tag{4}$$

where $\mathbf{g}_0$ is the initial gravity, N is the number of measurements, and $\mathbf{a}(n)$ is the n-th acceleration measurement vector.

4.4 Controller

This section deals with the block labelled "Controller" in Figure 7.

4.4.1 Traditional Controller

While the controller is receiving input from the camera, it requires a processing step to transform the target position from image-based coordinates to robot heading coordinates. This step in unnecessary for the other forms of input (*e.g.*, from GPS).

$$\dot{\mathbf{x}} = \mathbf{L}_x \mathbf{v}_c, \tag{5}$$

$$\mathbf{L}_x = \begin{bmatrix} -\frac{1}{z} & 0 & \frac{x}{z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{z} & \frac{y}{z} & 1+y^2 & -xy & -x \end{bmatrix}, \tag{6}$$

$$\dot{\mathbf{v}}_c = \mathbf{K}\mathbf{L}_x^{+} \begin{bmatrix} x_c - x_d \\ y_c - y_d \end{bmatrix}. \tag{7}$$

Equation (5) relates the image plane feature velocity $\dot{\mathbf{x}}$ to the fixed camera velocity $\mathbf{v}_c$. The relationship between the two relies on $\mathbf{L}_x$, the image Jacobian which is shown in Equation (6), where z is the target object's depth. Since feature distance is computed from vision information, the $\mathbf{L}_x$ can be calculated directly from known variables. The image plane feature position is (x_c, y_c). The desired feature position is (x_d, y_d). The traditional PID controller is constructed in Equation (7) where $\mathbf{L}_x^+$ is Pseudo-inverse of $\mathbf{L}_x$.

4.4.2 Reinforcement Learning

A variety of different learning methods exist within the reinforcement learning literature. The present task requirements dictate that we address a control problem without a complete prior model of the environment dynamics. Two additional (and inter-related) constraints direct the choice of method: firstly, the requirement for reasonable actuator controls from the very beginning, and, secondly, a preference for rapid convergence. SARSA, named because it learns from the variables abbreviated as $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$, is an algorithm which estimates a state-action–value function (or Q-function) for a given policy. SARSA is a popular application of the temporal difference approach to address control problems. The algorithm forms an intermediate method between TD(0) and the Monte-Carlo method being both model-free and capable of boot-strapping. Consideration of a_{t+1} is what distinguishes SARSA from standard Q-Learning: the Q-values in the former represent the values for the policy the learning agent follows including the consequences of exploration, while the latter considers only the values from following a exploitation policy. Put another way, in SARSA a policy describes the action selected in a particular state (s_t) with the action performed in the next state (s_{t+1}) drawn from the same policy; thus, it is called an on-policy method.

One advantage of SARSA is the ease with which it can be extended to allow eligibility traces in the form of a method called SARSA(λ). An eligibility trace is additional information associated with each Q-value which records the visits to the associated state. The eligibility trace does this in a time-decaying manner (we employ the notation $e(s, a)$ below for this information). The trace value is interpreted as the degree to which a particular Q-value is suited for learning updates, which means the temporal difference information can update many values rather than a single one. Thus, SARSA(λ) has the advantage that learning time can be greatly accelerated over the standard SARSA method.

In the implementation and evaluation described below, we have introduced a new action selection policy which augments a basic policy by imposing additional structure on the non-exploitative actions. We term this "exploration guidance" because it uses the current Q-value maximum to influence exploration choices. In particular, we found that the vast majority of exploratory actions can be drawn from boundary conditions that are essentially useless because they ignore the structure already identified via the maximum Q-value. When appropriate initial conditions are used (as described below) exploration guidance tempers the actions that are selected so that reasonable control is achieved throughout the learning process. Essentially, the method enforces a bound on the difference between the exploration and the exploitation actions, via a parameter σ_e. So doing adds the additional requirement that there be a measure of difference between actions, which is quite natural in our control context. The algorithm, from Sutton and Barto [1998], is shown in Algorithm 1 along with the modifications that add exploration guidance. The pseudo-code shows how an action selection based on the ϵ-greedy policy can use the σ_e guidance; treatment for ϵ-soft, or softmax variations are analogous.

Algorithm 1 SARSA(λ) with exploration guidance action selection.

```
Initialize $\mathbf{Q}(s,a) = \mathbf{H}(s,a)$ and $e(s,a) = 0$, for all $s, a$
repeat for each episode do
  Initialize $s, a$
  repeat for each step of episode do
    Take action $a$, observe $r$,$s'$
    Compute action $a^\star$ which maximizes $\mathbf{Q}$-value from $s'$
    with probability $\epsilon$ do
      Guided exploration: randomly choose $a'$ such that $|a' - a^\star| \leq \sigma_e$
    otherwise
      Exploitation: choose $a' = a^\star$
    end
    $\delta \leftarrow r + \gamma\ \mathbf{Q}(s', a') - \mathbf{Q}(s,a)$
    $e(s,a) \leftarrow e(s,a) + 1$
    for all $s, a$ do
      $\mathbf{Q}(s,a) \leftarrow \mathbf{Q}(s,a) + \alpha\delta e(s,a)$
      $e(s,a) \leftarrow \gamma\lambda e(s,a)$
    end for
    $s \leftarrow s'; a \leftarrow a'$
  end repeat
end repeat
```

State variables (50 states)	$(x, y) = [-200, -192, -184, \cdots, 192, 200]$
Target state	$(x, y) = [-10 : 10]$
Control variable (50 actions)	$\theta = [-1.0, -0.96, -0.92, \cdots, 0.92, 0.96, 1.0]$
Guidance deviation	$\sigma_e = 20$
Initialization deviation	$\sigma_h = 5$
Sampling time	50ms
Reward	$R = \begin{cases} 50 \text{ if goal,} \\ -5 \text{ if break,} \\ -\text{error otherwise.} \end{cases}$

Table 1: Parameters of Reinforcement Learning algorithm on the physical robot.

Table 1 summarizes the particulars of the reinforcement learning problem used for the physical robots. Note that (x, y) denotes an image coordinate position. Also, the basic sampling interval is 50ms, but it can be increased by a vision processing delay up to 500ms. This order of magnitude increase in time is non-deterministic. The control variables represent the power of two thrusters: so the heading between -1.0 to 1.0 is represented by discretizing into 50 actions. The reward is constructed to that arriving in the reference position results in a score of 50. Otherwise, -5 is obtained when an image processing failure occurs (*e.g.*, when the target leaves the camera frame).

We have assumed that the camera is located and aligned so the central region of the image coincides with a straight heading. Using this information, an initial state-action table with reasonable starting values is constructed. This is done via a 50×50 matrix $\mathbf{H}$ composed to have values for treatment of each control variable ($-1.0 < \mathbf{A} < 1.0$, a total of 50 actions) in each possible state ($-200 < \mathbf{S} < 200$, a total of 50 states). We impose the following natural relationship between actions and states: alignment means that the indices can be directly associated. For each state (a column in $\mathbf{H}$) we assign values in the form of a Gaussian probability distribution across the actions (the row entries). The distribution has a constant standard deviation given by σ_h, but is shifted so that the peak in column i falls in the ith position. Thus, the maximum values fall along the diagonal of $\mathbf{H}$. As the number of episodes is increased and additional empirical data is obtained, the $\mathbf{Q}$ matrix adapts to better reflect the dynamic, unpredictable environment. The purpose of $\mathbf{H}$ is to help achieve operational robustness in the early stages of learning.

To sum up, we have employed two approaches to apply reinforcement learning techniques to our problem: one is using guidance exploration SARSA(λ) to help speed convergence, the other is designing an initial $\mathbf{H}$ matrix for stability.

4.5 Disturbance Compensation

Algorithm 2 describes the simple logic that implements the disturbance compensation method in software: essentially it consists of two concurrently executing control loops, each operating at a time-scale dictated by their inputs, which are coupled through heading variables. $D_m = \begin{bmatrix}\mathbf{y}_x{}^\top & \mathbf{y}_y{}^\top & \mathbf{y}_z{}^\top\end{bmatrix}^\top$ is the output of the disturbance model (in m/s^2) as is described in Section 4.3 and detailed in Appendix A. Since the IMU coordinate system is aligned with the image frame coordinates in our system, the model of environmental disturbances can be simplified by considering the accelerations along the first two axes (*viz.* $\begin{bmatrix}\mathbf{y}_x{}^\top, \mathbf{y}_y{}^\top\end{bmatrix}$). In the algorithm, R_{pos} denotes the horizontal reference position in camera coordinates (*i.e.,* the center of image, in our configuration). T_{pos} represents the current estimate of the target object's position and is produced by the vision processing pipeline as outlined in Section 4.1. From the reference position and the target's position we directly compute a heading which we denote with θ_t. Also θ_d represents the robot's desired heading which is computed to offset estimated disturbances. Finally, $\dot{x}_d$ is the robot's desired speed which is similarly adjusted.

The control loops are shown as each of the two main functions: The OUTER_CONTROLLER which updates θ_t by using the measured target object position. The other is INNER_CONTROLLER which counteracts the disturbance by using $D_m(\mathbf{y}_x{}^\top, \mathbf{y}_y{}^\top)$. The disturbance elements $\mathbf{y}_x{}^\top$ and $\mathbf{y}_y{}^\top$ are used because they have the primary influence on the heading and speed of the robot. The gain g_x is used in computing θ_d as described in the algorithm, and it serves to normalize the output to between -1 and $+1$ radians. The robot's speed, $\dot{x}_d$ is adjusted by g_y similarly.

5 Experiments

The environment used for the marine experiments that follow was a flowing river. The experimental scenario is shown in Figure 8. The global positioning mode is employed to have the ASV approach the target object and get it within range before detection of the target object and visual servoing commences. The docking target was passive: floating on the surface not executing any motor commands, which we feel can help in evaluating the proposed method. Once visual servoing mode

Algorithm 2 Disturbance compensation via two loop controller

```
Inputs:
  $D_m$ = Disturbance measurement $\begin{bmatrix}\mathbf{y}_x{}^\top & \mathbf{y}_y{}^\top & \mathbf{y}_z{}^\top\end{bmatrix}^\top$ (acceleration along each axis)
  $R_{pos}$ = Reference position $[x]$ in camera coordinates
  $T_{pos}$ = Target object position $[x, y]$ in camera coordinates
Parameters:
  $g_{x,y}$ = The gain parameter of computing $\theta_d$ and $\dot{x}$
  $\dot{s}$ = The robot's pre-adjusted speed
Computed Values:
  $\theta_t$ = The direction to the target object by image processing
  $\theta_d$ = The robot's desired heading
  $\dot{x}_d$ = The robot's desired speed

  Function OUTER_CONTROLLER($T_{pos}, R_{pos}$)        (update $\theta_t$ when the vision is processed)
    $\theta_t = R_{pos} - T_{pos}(x)$

  Function INNER_CONTROLLER($\theta_t, D_m$)        (update $\theta_d$ and $\dot{x}_d$ when the IMU is processed)
    $\theta_d = \theta_t - g_x D_m(X)$
    $\dot{x}_d = \dot{s} - g_y D_m(Y)$
```

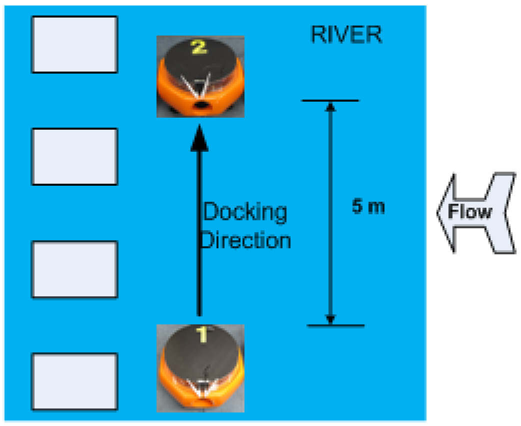

Figure 8: The evaluation scenario: flowing water with wind typical of unpredictable dynamic environments which are challenging to model.

is enabled the target object was reliably. ASV operates autonomously employing the controller described. Measurements were made of whether the ASVs docked; final docking was deemed to have been achieved only when contact was made between robots. Since we were unable to instantaneously measure the flowing river, ground truth (*e.g.,* with different degrees of disturbance and noise) for evaluation purposes remained problematic. Thus, we also used the same controllers on ground vehicles: with the AGVs we assume that ground is static easily characterizable environment compare to a choppy, flowing water surface. With the ground vehicles, synthetic noise was added for comparison to the no noise case. Although only an approximation, we believe this to be a good preliminary evaluation of the impact of the disturbance compensation mechanism. The final we test we carried out was with ASVs in a realistic marine environment, and the results show that the proposed disturbance rejection, reinforcement learning, and two-loop controller permit docking in unpredictable and dynamic environments. Moreover, comparison to the traditional controller results show the effectiveness of the disturbance rejection method.

5.1 Depth Estimation in Visual Servoing

Most image features have different processing times. Particularly, SIFT [Lowe, 2004] performs well in detecting and recognizing a target object in real environment. However, because of slow computation time, it is insufficient for uses that require real-time performance. In order to use it in real environments, Giesbrecht *et al.* [2009] uses a SIFT tracker to track a moving object. To do so, the authors measured the processing time for the SIFT computation and calibrated their control loop so as to use this as the underlying control period. A reduction in search space was necessary in order to maximize the control frequency.

In this paper, we employ a two-loop control logic in order to have a feature-independent approach. Thus, our system can use feature descriptors such as SIFT in addition to a robust matching method. Generally feature descriptors are comparatively slow, but they can successfully estimate a target object's pose and depth both of which are useful for dynamic docking as well as following. In the following experiments, the scale of the target object (as a function of distance) is given by a scale table. Consequently, depth information is produced from scale. Figure 9 shows the detected image height versus ground truth distance for both SIFT and our matching method. While both methods have agreement over their common range, SIFT can detect up to 200cm while the tracker can track the target object up to 500cm. When both run concurrently, SIFT takes 1000ms per frame while the tracker uses only 100ms on our hardware.

It can support the estimated scales of the target robots. SIFT can detect up to 200cm within 1000ms, while simultaneously, the tracker can track the target object up to 500cm within 100ms with our hardware. Thus, various features can be used for target object detection and tracking in the proposed control architecture. To overcome the feature descriptors' slow speed, several trackers can be used within the inner controller, which helps to compensate for the detection speed and assists in tracking long distances.

The results of the tests on the detection and tracking of target objects in real environments are shown in Figure 10. The scale size represents robot's depth which is from produced from the scale table.

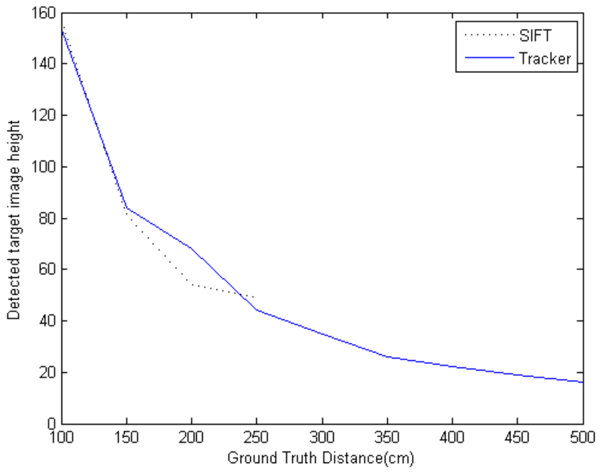

Figure 9: Scale Estimation: detected target image height versus ground truth distance for the slow SIFT features compared and our implementation of a tracking method.

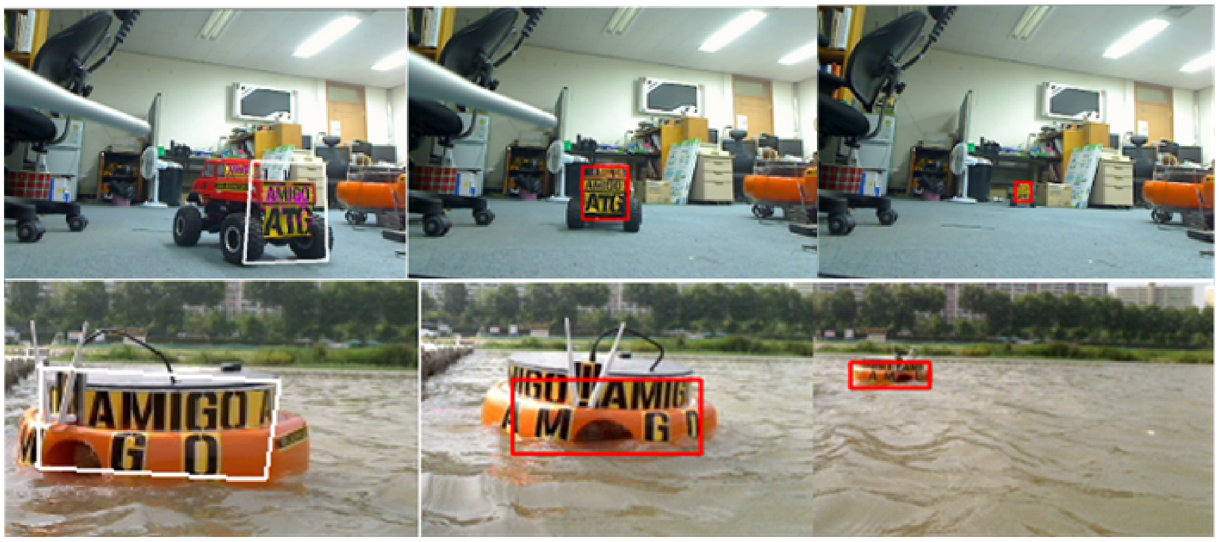

Figure 10: Target detection (white boxes) and tracking (red boxes) for the AGV and ASVs. Left: Target Detection at 1m. Middle: Target Tracking at 2m. Right: Target Tracking at 4m.

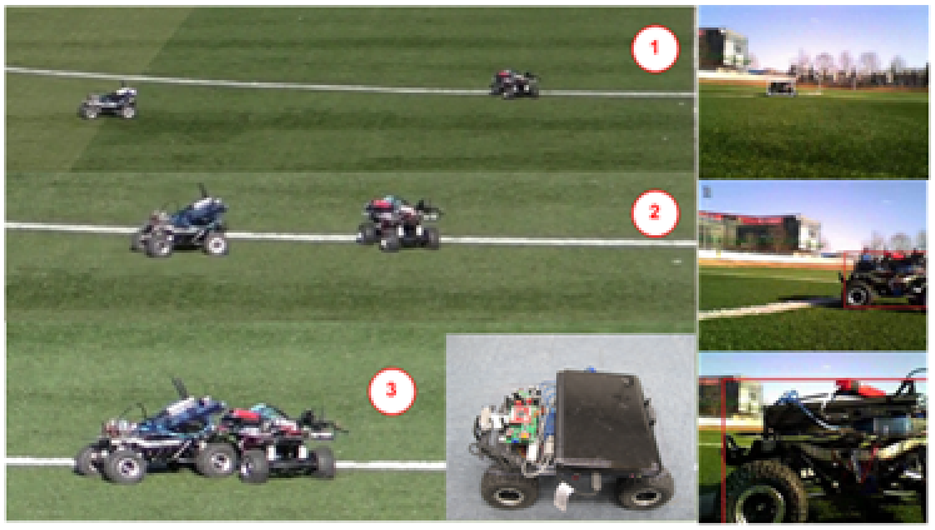

Figure 11: A "docking" test with autonomous ground vehicles. This scenario permits controlled evaluation of the role of environmental disturbances since the AGVs do not contend with the effects of choppy, flowing water or wind.

5.2 Results of Autonomous Ground Vehicle

To evaluate the proposed method, one target AGV was placed 4m ahead of another AGV. The target AGV was detected AGV. Figure 11 shows the approach of the docking AGV along with its tracking on the right. The utility of disturbance rejection was evaluated by comparing a traditional PID (with Reinforcement Learning) controller with disturbance rejection to one without. On the ground, there is not enough noise to reject disturbance. So, injected Gaussian noise is used to capture elements of a dynamic, unpredictable environment. In Figure 11, number 1 shows completion of approach control, and number 2 shows detecting and tracking the target robot with disturbance rejection. Finally, number 3 is docked with the target robot.

As a result, in Figure 12, the left figure shows the results of Reinforcement Learning without disturbance rejection which does not have irregular rewards at any time. The right figure shows those of Reinforcement Learning with disturbance rejection: observe both the increased mean reward function, the reduction in variance, and the significantly improved lower envelope. This suggests the disturbance rejection improves the learning process itself.

Moreover, in Figure 13, we present two types of noise conditions for both the PID and RL controllers (left and right, respectively). The "No Added Noise" condition reflects the natural noise within the robot system, *i.e.,* but without additional noise being injected. The "Random Noise" condition includes the addition of Gaussian random noise (horizontal position error is sampled from a Normal distribution with $\sigma = 20$ pixels.) The plots show the response of the system under each of these conditions along with a third line illustrating the effectiveness of the controllers under the "Random Noise" condition but with the disturbance rejection functionality enabled. By comparing the relative amplitude of the oscillations it is straightforward to observe that the Reinforcement Learning controller follows the target more smoothly than the traditional PID controller. Additionally, disturbance rejection helps to follow a target object more smoothly in both cases. Finally,

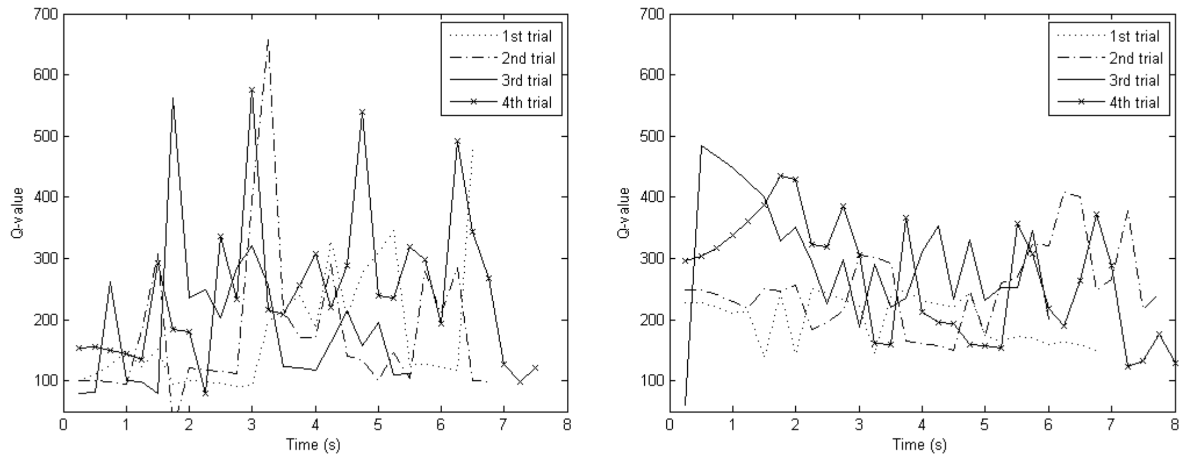

Figure 12: An illustration of the effectiveness of disturbance rejection on the learned state-action map (Q-values): the Q-value function landscape is flattened and smoothed. The left figure shows the Q-values of the executed policy versus time in four separate trials for Reinforcement Learning without disturbance rejection on docking ground vehicles. The right figure is identical except with disturbance rejection enabled. Significant variations both along individual trials and between trials is readily observed. Nevertheless the disturbance rejection reduces the amplitude of the Q-value variation. (Since SARSA is an on-policy method, the Q-values represent estimates of returned reward including both exploration and exploitation aspects.)

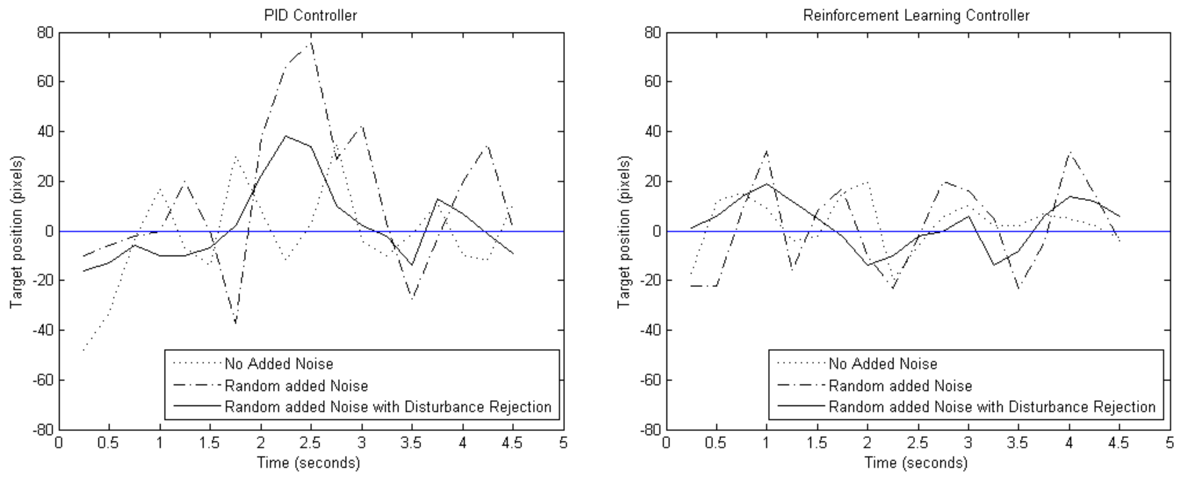

Figure 13: Evidence of increased stability in visual servoing control on ground vehicles. The vertical axis represents the distance of the centroid of the target from the central line in the image (in number of pixels); the horizontal axis denotes time in seconds. Smoothness of the target following behavior is reflected in the relative amplitudes. (Left: PID controller, Right: Reinforcement Learning controller controller).

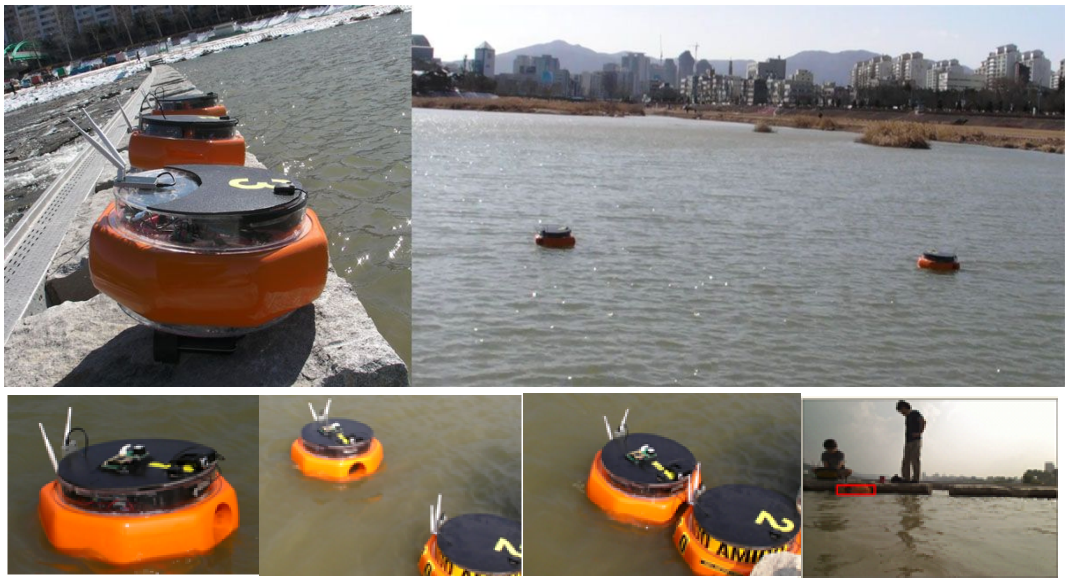

Figure 14: An example USV pair docking test. (Compare the water surface to the calm day shown in Figure 4.)

the combined use of disturbance rejection and Reinforcement Learning is beneficial. (Note that in Figure 13, the vertical axis represents object positions relative to the center of input images so zero is the target reference, permitting one to interpret the required robot heading.)

5.3 Results of Autonomous Surface Vehicles

The previous section demonstrated that the techniques integrated into the control system are effective for AGV docking. Next, the system was tested with our prototype ASVs in a real river environment. The scenario as described above. As shown in Figure 14, one ASV detect the other ASV and then docks with it. The fourth frame of Figure 14 is the view of ASV from the initial position.

Tests in which the ASV tries to dock to a drifting target robot at a distance of $4m$ away were executed for different initial poses but same distance and the results are reported in Figure 15. The dotted lines on both the graphs are results of PID & RL without disturbance rejection (two different initial positions). The solid lines are results of PID & RL with disturbance rejection (three different initial positions). In general, the particular initial positions give repeated fluctuation. The PID controller has more frequent and large oscillation compared to the Reinforcement Learning method. Moreover, disturbance rejection reduces these fluctuation, and reduces the time needed to approach the other robot. The shorter execution times suggest additional robustness to fluctuations, as reaching stability implied faster recover. As the number of time is increased which is shown DR1 to DR3, the docking time is reduced from around 7 seconds to 5 seconds. Moreover, based on the oscillation, the Reinforcement Learning with disturbance rejection shows the stable visual controller compared to the traditional approach.

Evidence of increased stability in visual servoing control on ground vehicles. The vertical axis

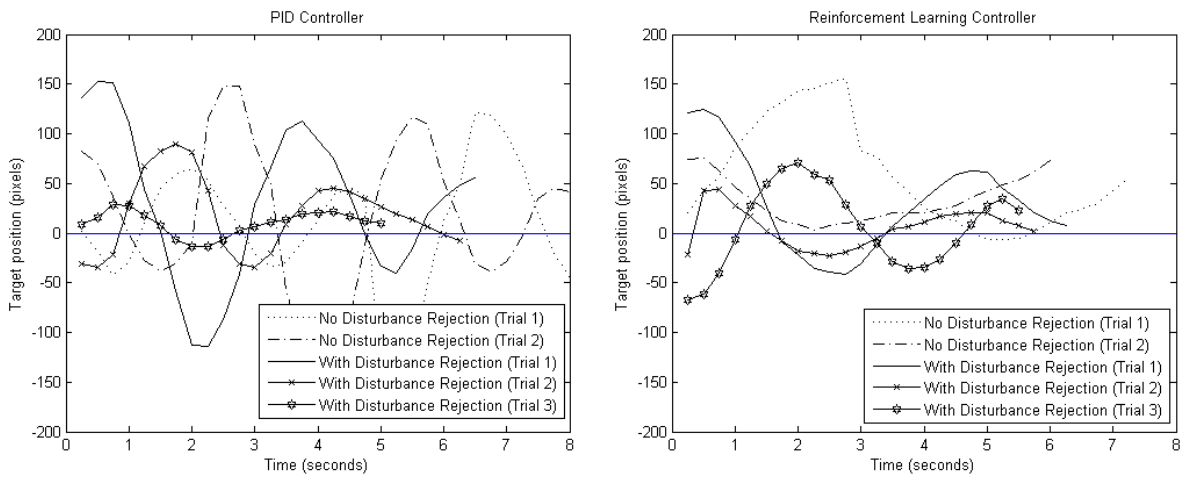

Figure 15: Visual servoing control of Autonomous Surface Vehicles at real environment. The vertical axis represents the distance of the centroid of the target from the central line in the image (in number of pixels); the horizontal axis denotes time in seconds. (Left: PID controller, Right: Reinforcement Learning controller controller).

represents the distance of the centroid of the target from the central line in the image (in number of pixels); the horizontal axis denotes time in seconds. Smoothness of the target following behavior is reflected in the relative amplitudes. (Left: PID controller, Right: Reinforcement Learning controller controller).

6 Conclusions

Ultimately, if ASVs are to fulfill important search, rescue, navigation, and inspection duties, these vehicles need to detect and recognize objects fast and reliably, and they need to be able to operate in unpredictable and dynamic environments. This paper has described an approach with several integrated elements toward that end:

- Visual servoing is generally considers a single control loop. However, this work used a two-loop system with a switching structure that increases feature generality. The visual servoing controller uses numerous features such as color, templates, and local descriptors, all of which have different extraction speeds, yet all of which can be used without serious restriction. Attempting to attain feature-independence in visual servo control is a useful principle in order to achieve robustness and a two-loop switched controller is an effective way to do this.
- Disturbance rejection-like stabilization at the shortest time-scale is beneficial in dynamic environments, but can require selection of computationally efficient techniques (like the mean-shift, in our case). Fusing data at the longest time-scale is likely to be problematic.
- Adaptability via on-line learning improves performance in non-stationary environments, but careful choice of learning algorithm and choice of sub-problem is important. In this work

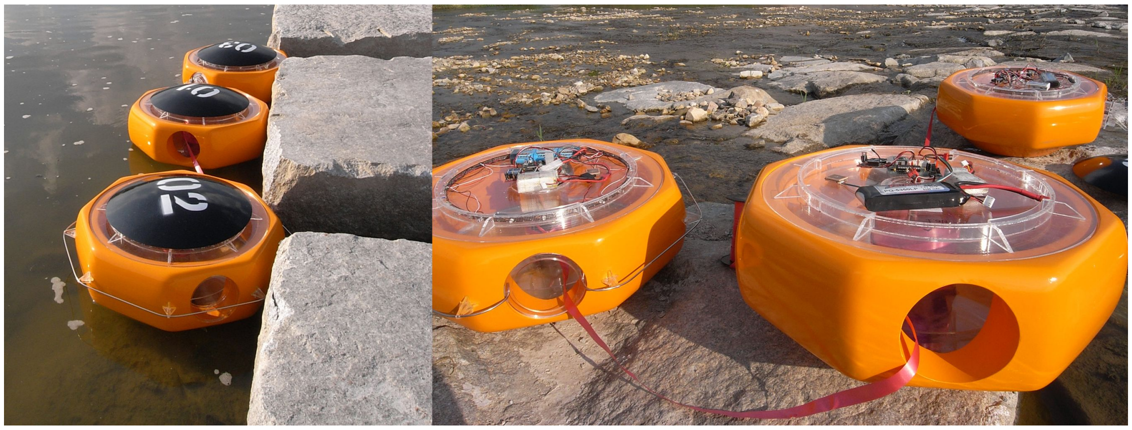

Figure 16: Tests with multiple tethered units.

it remains unclear how disturbance rejection affects the learning process. Both disturbance rejection and adaptation improve performance; their interplay is not yet clear.

This work has proposed an important multi-ASV application: that of autonomous robotic boom deployment in order to corral floating pollutants. Our experience was that wind, waves, and water currents would pose a particular challenge for tasks that require tightly coupled coordination. A prototype robotic system was constructed, and the plausibility of autonomous multi-ASV docking demonstrated, including evaluation with test-trials with controlled noise conditions on ground vehicles.

6.1 Outlook and Future Work

Further development is necessary before the robotic containment boom idea can be declared feasible. An important mechanical consideration is the mechanism by which booms should reliably fasten after the docking procedure. Possibly insights from the reconfigurable robotics community could contribute to that solution. Figure 16 shows some preliminary experiments that were conducted with the units that had been tethered together. Exactly how to maneuver large groups of such robots remains an open question.

Future work could consider ways in which the disturbance rejection scheme can be extended through a robust three-axis modeling that can help the system adapt to additional types of challenging environments. This would address limitations of the current approach and likely improve performance across a range of dynamic environments. Second, while currently employing two control loops helps ameliorate the effects of delay in sensor processing, faster feature extraction methods will naturally improve the quality of the control output and can simplify the controller itself. For example, the separation between target detection and tracking is critical to the success of the current system because detection on its own would be too slow. Third, important questions remain as to the rate of convergence when adaptive on-line learning methods are used in multi-vehicle scenarios. For example, when two robots are approaching one another in order to dock, it would be beneficial to have an estimate of their comparative stability as this could affect the servoing strategy employed.

A IMU Model

State-space equations for the IMU are given by

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k) + \mathbf{n}(k), \tag{8}$$

$$\mathbf{y}(k) = \mathbf{H}\mathbf{x}(k) + \mathbf{v}(k), \tag{9}$$

where $\mathbf{\Phi} = diag(\Phi_x, \Phi_y, \Phi_z)$ is the state transition matrix and is a block diagonal matrix that consists of the state transition matrices

$$\mathbf{\Phi}_m = \begin{bmatrix} 1 & T_s & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{10}$$

of each axis, where T_s is the sampling interval. The input matrix, $\mathbf{G} = diag(\mathbf{G}_x, \mathbf{G}_y, \mathbf{G}_z)$ is a block diagonal matrix with the input matrices, $\mathbf{G}_m = [T_s \quad 0 \quad 0 \quad 0 \quad 0]^\mathsf{T}$. The measurement matrix, $\mathbf{H} = diag(\mathbf{H}_x, \mathbf{H}_y, \mathbf{H}_z)$ is also a block diagonal matrix having three m-axis input matrices, $\mathbf{H}_m = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$.

$\mathbf{y} = \left[\mathbf{y}_x{}^\mathsf{T} \quad \mathbf{y}_y{}^\mathsf{T} \quad \mathbf{y}_z{}^\mathsf{T}\right]^\mathsf{T}$ is an output vector that is a compound of each axis's output vector, $\mathbf{y}_m = [a_m \; \omega_m]^\mathsf{T}$. Here a_m and ω_m mean measurements of the accelerometer and gyro with respect to the m-axis. A system's input vector, $\mathbf{u} = \left[g_x + a_{F_x} \quad g_y + a_{F_y} \quad g_z + a_{F_z}\right]^\mathsf{T}$. consists of components along each axis of the local gravity, g_m and of the acceleration, a_{F_m} exerted by the robot' driving force. $\mathbf{n} = \left[\mathbf{n}_x{}^\mathsf{T} \quad \mathbf{n}_y{}^\mathsf{T} \quad \mathbf{n}_z{}^\mathsf{T}\right]^\mathsf{T}$ means a process noise vector of unmodeled effects with the process noise vectors of m-axis, $\mathbf{n}_m = [n_{vm} \quad n_{am} \quad n_{b_{am}} \quad n_{\omega m} \quad n_{b_{\omega m}}]^\mathsf{T}$, each element of which means Gaussian white noise corresponding to each element of the state vector and $n_{vm} \sim N(0, Q_{vm})$, $n_{am} \sim N(0, Q_{am})$, $n_{b_{am}} \sim N(0, Q_{b_{am}})$, $n_{\omega m} \sim N(0, Q_{\omega m})$, and $n_{b_{\omega m}} \sim N(0, Q_{b_{\omega m}})$. A measurement noise vector, $\mathbf{v} = \left[\mathbf{v}_x{}^\mathsf{T} \quad \mathbf{v}_y{}^\mathsf{T} \quad \mathbf{v}_z{}^\mathsf{T}\right]^\mathsf{T}$ is made up of each measurement noise vector, $\mathbf{v}_m = [v_{am} \quad v_{\omega m}]^\mathsf{T}$ of axes where v_{am} and $v_{\omega m}$ are Gaussian white noises corresponding to each element of the output vector, $\vec{y}_m$, and therefore these noise terms are defined by $v_{am} \sim N(0, R_{am})$, and $v_{\omega m} \sim N(0, R_{\omega m})$.

References

[Aull, 2009] Mark Aull. Visual servoing for an autonomous rendezvous and capture system. *Intelligent Service Robotics*, 2(3):131–137, July 2009.

[BBC, 2007] BBC. South Korea fights huge oil spill, December, 10 2007. http://news.bbc.co.uk/2/hi/asia-pacific/7135896.stm.

[Benjamin *et al.*, 2006] Michael Benjamin, Joseph Curcio, John J. Leonard, and Paul Newman. Navigation of Unmanned Marine Vehicles in Accordance with the Rules of the Road. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'06)*, pages 3581–3587, Orlando, FL, May 2006.

[Bertram, 2008] Volker Bertram. Unmanned Surface Vehicles – A Survey. In *Proceedings of Skibsteknisk Selskab Meeting*, Copenhagen, Denmark, March 2008.

[Bibuli *et al.*, 2008] Marco Bibuli, Gabriele Bruzzone, Massimo Caccia, Giovanni Indiveri, and Alessandro A. Zizzari. Line following guidance control: Application to the Charlie unmanned surface vehicle. In *Proceeding of the IEEE International Conference on Intelligent Robots and Systems (IROS'08)*, pages 3641–3646, Nice, France, September 2008.

[Caccia *et al.*, 2008] Massimo Caccia, Marco Bibuli, Riccardo Bono, and Gabriele Bruzzone. Basic navigation, guidance and control of an Unmanned Surface Vehicle. *Autonomous Robots*, 25(4):349–365, November 2008.

[Clauss *et al.*, 2009] Günther F. Clauss, André Kauffeldt, and Nils Otten. AGaPaS — Autonomous Galileo-Supported Rescue Vessel for Persons Overboard. In *Proceedings of International Conference on Ocean, Offshore and Arctic Engineering (OMAE'09)*, Honolulu, Hawaii, May 2009.

[Comaniciu *et al.*, 2002] Dorin Comaniciu, Peter Meer, and Senior Member. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[Curcio *et al.*, 2005] Joseph Curcio, John Leonard, and Nicholas Patrikalakis. SCOUT—a low cost autonomous surface platform for research in cooperative autonomy. In *Proceedings of the MTS/IEEE Oceans Conference*, pages 725–729, Washington D.C.,, September 2005.

[Doerffer, 1992] Jerzy Doerffer. *Oil Spill Response in Marine Environment.* Pergamon Press, New York, NY, 1992.

[Dunbabin *et al.*, 2008] Matthew Dunbabin, Brenton Lang, and Brett Wood. Vision-based Docking Using an Autonomous Surface Vehicle. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'08)*, pages 26–32, Pasadena, CA, USA, May 2008.

[El-Fakdi and Carreras, 2008] Andres El-Fakdi and Marc Carreras. Policy gradient based Reinforcement Learning for real autonomous underwater cable tracking. In *Proceeding of the International Conference on Intelligent Robots and Systems (IROS'08)*, pages 3635–3640, Nice, France, September 2008.

[Fahimi, 2007] Farbod Fahimi. Sliding-Mode Formation Control for Underactuated Surface Vessels. *IEEE Transactions on Robotics*, 23(3):617–622, June 2007.

[Fang and Wong, 2001] Jianzhi Fang and Kau-Fui V. Wong. Optimization of an Oil Boom Arrangement. In *Proceedings of Biennial International Conference on Oil Spills*, Tampa, FL, March 2001.

[Fang and Wong, 2006] Jianzhi Fang and Kau-Fui V. Wong. An advanced VOF algorithm for oil boom design. *International Journal of Modelling and Simulation*, 26(1), January 2006.

[Farrel and Barth, 1998] Jay A. Farrel and Matthew Barth. *The Global Positioning System and Inertial Navigation.* McGraw-Hill, New York, USA, 1998.

[Fingas and Charles, 2000] Mervin F. Fingas and Jennifer Charles. *The basics of oil spill cleanup.* CRC Press, 2 edition, 2000.

[Fischler and Bolles, 1981] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Comm. of the ACM*, 24(6):381–395, June 1981.

[Gaskett *et al.*, 1999] Chris Gaskett, David Wettergreen, and Alexander Zelinsky. Reinforcement Learning applied to the control of an Autonomous Underwater Vehicle. In *Proceedings of the Australian Conference on Robotics and Automation (AUCRA99)*, pages 125–131, Brisbane, Australia, March 1999.

[Giesbrecht *et al.*, 2009] Jared L. Giesbrecht, Hien K. Goi, Timothy D. Barfoot, and Bruce A. Francis. A vision-based robotic follower vehicle. *Proceedings of the International Society for Optics and Photonics (SPIE)*, 7332:733210–1–733210–12, April 2009.

[Grewal and Andrews, 2001] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering: Theory and Practice Using MATLAB.* Wiley-Interscience, New York, USA, 2001.

[Grewal *et al.*, 1991] Mohinder S. Grewal, Vinson D. Henderson, and Randy S. Miyasako. Application of Kalman Filtering to the Calibration and Alignment of Inertial Navigation Systems. *IEEE Transactions on Automatic Control*, 36(1):4–13, January 1991.

[Grier, 2010] Peter Grier. Containment boom effort comes up short in BP oil spill. *The Christian Science Monitor*, June, 11 2010.

[Hutchinson *et al.*, 1996] Seth Hutchinson, Gregory D. Hage, and Peter I. Corke. A Tutorial on Visual Servo Contol. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, October 1996.

[Joshi *et al.*, 2010] Neel Joshi, Sing Bing Kang, C. Lawrence Zitnick, and Richard Szeliski. Image deblurring using inertial measurement sensors. *ACM Transactions on Graphics*, 29(4):1–9, July 2010.

[Kakalis and Ventikos, 2008] Nikolaos M.P Kakalis and Yiannis Ventikos. Robotic Swarm Concept for Efficient Oil Spill Confrontation. *Journal of Hazardous Materials*, 154(1–3):880–887, June 2008.

[Kim *et al.*, 2009] Myungsik Kim, Nak Young Chong, and Wonpil Yu. Robust DOA estimation and target docking for mobile robots. *Intelligent Service Robotics*, 2(1):41–51, January 2009.

[Lowe, 2004] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, January 2004.

[Martinez-Marin and Duckett, 2005] Tomás Martinez-Marin and Tom Duckett. Fast Reinforcement Learning for Vision-guided Mobile Robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'05)*, pages 4170–4175, Barcelona, Spain, April 2005.

[Martins *et al.*, 2007] Alfredo Martins, J. M. Almeida, H. Ferreira, H. Silva, N. Dias, A. Dias, Carlos Almeida, and E. P. Silva. Autonomous Surface Vehicle Docking Manoeuvre with Visual Information. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'07)*, pages 4994–4999, Roma, Italy, April 2007.

[Murarka *et al.*, 2009] Aniket Murarka, Gregory Kuhlmann, Shilpa Gulati, Mohan Sridharan, Christopher Flesher, and William C. Stone. Vision-based Frozen Surface Egress: A Docking Algorithm for the ENDURANCE AUV. In *Proceedings of the International Symposium on Unmanned Untethered Submersible Technology (UUST)*, New Hampshire, USA, August 2009.

[Murphy *et al.*, 2011] Robin R. Murphy, Eric Steimle, Michael Hall, Michael Lindemuth, David Trejo, Stefan Hurlebaus, Zenon Medina-Cetina, and Daryl Slocum. Robot-Assisted Bridge Inspection. *Journal of Intelligent & Robotic Systems*, pages 1–19, 2011.

[NOAA, 2010] NOAA. Deepwater Horizon MC252 Gulf Incident Oil Budget, August, 2 2010. http://www.noaanews.noaa.gov/stories2010/PDFs/DeepwaterHorizonOilBudget20100801.pdf.

[Ondřej Chum and Jiri Matas, 2005] Ondřej Chum and Jiri Matas. Matching with PROSAC—Progressive Sample Consensus. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 220–226, San Diego,USA, July 2005.

[Park *et al.*, 2009] Jin-Yeong Park, Bong huan Jun, Pan mook Lee, and Junho Oh. Experiments on vision guided docking of an autonomous underwater vehicle using one camera. *Ocean Engineering*, 36(1):48–61, January 2009.

[Parker, 2008] Lynne E. Parker. Multiple Mobile Robot Systems. In Bruno Siciliano and Oussama Khatib, editors, *Handbook of Robotics*, chapter 40. Springer, 2008.

[Pereira *et al.*, 2008] Avind Pereira, Jnaneshwar Das, and Gaurav S. Sukhatme. An Experimental Study of Station Keeping on an Underactuated ASV. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS'08)*, pages 3164–3171, Nice, France, September 2008.

[Rosten and Drummond, 2006] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision (ECCV'06)*, pages 430–443, Graz,Austria, May 2006.

[Sullivan, 2010] Gregory Sullivan. Miles of Oil Containment Boom Sit in Warehouse, Waiting for BP or U.S. to Use. *Pajamas Media*, June, 8 2010.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, USA, 1998.

[Wang *et al.*, 2009] Jianhua Wang, Wei Gu, and Jianxin Zhu. Design of an autonomous surface vehicle used for marine environment monitoring. In *Proceedings of the International Conference on Advanced Computer Control*, pages 405–0409, Los Alamitos, CA, USA, 2009.

[Welch and Bishop, 2006] Greg Welch and Gary Bishop. An introduction to the Kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill, July 2006.