Energy-Efficient Aquatic Environment Monitoring Using Smartphone-Based Robots

YU WANG, Michigan State University
RUI TAN, Nanyang Technological University
GUOLIANG XING, JIANXUN WANG, XIAOBO TAN, and XIAOMING LIU,
Michigan State University

Monitoring aquatic environment is of great interest to the ecosystem, marine life, and human health. This article presents the design and implementation of Samba—an aquatic surveillance robot that integrates an off-the-shelf Android smartphone and a robotic fish to monitor harmful aquatic processes such as oil spills and harmful algal blooms. Using the built-in camera of the smartphone, Samba can detect spatially dispersed aquatic processes in dynamic and complex environment. To reduce the excessive false alarms caused by the nonwater area (e.g., trees on the shore), Samba segments the captured images and performs target detection in the identified water area only. However, a major challenge in the design of Samba is the high energy consumption resulted from continuous image segmentation. We propose a novel approach that leverages the power-efficient inertial sensors on smartphones to assist image processing. In particular, based on the learned mapping models between inertial and visual features, Samba uses real-time inertial sensor readings to estimate the visual features that guide image segmentation, significantly reducing the energy consumption and computation overhead. Samba also features a set of lightweight and robust computer vision algorithms, which detect harmful aquatic processes based on their distinctive color features. Last, Samba employs a feedback-based rotation control algorithm to adapt to spatiotemporal development of the target aquatic process. We have implemented a Samba prototype and evaluated it through extensive field experiments, lab experiments, and trace-driven simulations. The results show that Samba can achieve a 94% detection rate, a 5% false alarm rate, and a lifetime up to nearly 2 months.

Categories and Subject Descriptors: C.3 [**Special-purpose and Application-based Systems**]: Signal Processing Systems; C.4 [**Performance of Systems**]: Modeling Techniques; I.4 [**Image Processing and Computer Vision**]: Scene Analysis—*Object recognition*

General Terms: Measurement, Performance

Additional Key Words and Phrases: Robotic sensor, smartphone, computer vision, object detection, inertial sensing

---

A preliminary version of this work was published in the following conference paper:
Yu Wang, Rui Tan, Guoliang Xing, Xiaobo Tan, Jianxun Wang, and Xiaoming Liu. 2015. Samba: a smartphone-based robot system for energy-efficient aquatic environment monitoring. In *Proceedings of the 14th ACM/IEEE Conference on Information Processing in Sensor Networks (IPSN)*. 262–273.
This work was supported in part by the U.S. National Science Foundation under grants CCF-1331852, CNS-1218475, and ECCS-1446793, and in part by a Start-up grant at Nanyang Technological University.
Yu Wang is now with Samsung Electronics America, Inc.
Jianxun Wang is now with Apple Inc.
Authors' addresses: Y. Wang, G. Xing, and X. Liu, Department of Computer Science and Engineering, Michigan State University, 428 South Shaw Lane, East Lansing, MI 48824, USA; emails: {wangyu3, glxing, liuxm}@msu.edu; R. Tan, School of Computer Science and Engineering, Nanyang Technological University, Block N4 Nanyang Avenue, 639798, Singapore; email: tanrui@ntu.edu.sg; J. Wang and X. Tan, Department of Electrical and Computer Engineering, Michigan State University, 428 South Shaw Lane, East Lansing, MI 48824, USA; emails: {wangji19, xbtan}@msu.edu.

DOI: http://dx.doi.org/10.1145/2932190

ACM Reference Format:
Yu Wang, Rui Tan, Guoliang Xing, Jianxun Wang, Xiaobo Tan, and Xiaoming Liu. 2016. Energy-efficient aquatic environment monitoring using smartphone-based robots. ACM Trans. Sen. Netw. 12, 3, Article 25 (June 2016), 28 pages.
DOI: http://dx.doi.org/10.1145/2932190

1. INTRODUCTION

The aquatic environment is facing increasing threats from climate change, industrial pollution, and improper waste disposal. The last 4 decades have witnessed more than a dozen major oil spills with each releasing over 30 million gallons of oil [Oil Spills and Disasters 2010]. Harmful algal blooms (HABs) have been observed in more locations than ever before [National Ocean Service 2014]. Figure 1(a) shows a recent proliferation of HABs in the Gulf of Mexico [Southern Regional Water Program 2011]. Such harmful aquatic processes greatly endanger the marine biology, ecosystem sustainability, and public health. For example, HABs contaminated Ohio's inland lakes that supply drinking water, resulting in 41 confirmed cases of health impact to humans in 2010 [Ohio Environmental Protection Angency 2010]. It is thus imperative to detect these harmful aquatic processes, monitor their development, and alarm the authorities to take preventive actions.

Although manual opportunistic spotting may be applied to monitor small-scale harmful aquatic processes, it is often labor-intensive and unreliable. An alternative method is *in situ* visual survey with patrol boats [Ammerman and Glover 2000]. However, this method is costly and can only cover a limited period of time. More advanced methods employ remote sensing technologies, for example, balloon [Kako et al. 2012], aircraft [Hu et al. 2003], and satellite imaging [Brekke and Solberg 2005]. The balloon-based monitoring is effective only for one-off and short-term surveillance of highly concentrated aquatic processes that have already been detected. The monitoring approaches using aircraft and satellite imaging often incur high operational overhead and cannot achieve fine monitoring resolution. Recently, autonomous underwater vehicles (AUVs) [Rudnick et al. 2004] have been used for various underwater sensing tasks. However, the manufacturing costs of AUV platforms are often high (over $\$50{,}000$ per unit [Rudnick et al. 2004]). In summary, these limitations make remote sensing and AUV-based approaches ill-suited for monitoring spatially scattered and temporally developing aquatic processes.

This article presents *Samba* (Smartphone-based aquatic monitoring robotic platform), an energy-efficient and low-cost robot system that integrates an off-the-shelf Android smartphone and a robotic fish to monitor phenomena on the water surface. Figure 1(b) shows a prototype of Samba that is built with a Samsung Galaxy Nexus phone. The integrated smartphone and robotic fish assemble a promising platform for aquatic monitoring due to the following salient advantages. The robotic fish developed in our previous work [Gliding Robotic Fish 2013] is a low-cost (about $\$3{,}000$ per unit) aquatic mobile platform with high maneuverability in rotation and orientation maintenance, enabling Samba to adapt to the dynamic aquatic environment. Moreover, it has stronger resistance to capsizing than robotic boats (e.g., Dunbabin and Grinham [2010]) due to the enhanced stability from its bionic design. The on-board cameras of smartphones provide an inexpensive yet high-resolution sensing modality for detecting the harmful aquatic processes. For example, HABs, as shown in Figure 1(a), can be detected using the phone's built-in camera based on their distinctive colors. Moreover, in addition to the camera, a few other sensors such as accelerometer and gyroscope, which are commonly available on smartphones, can assist the navigation and control of robotic fish. Second, compared with traditional chemical sensors that measure only one location at a time, the camera has a wider sensing range and provides richer information

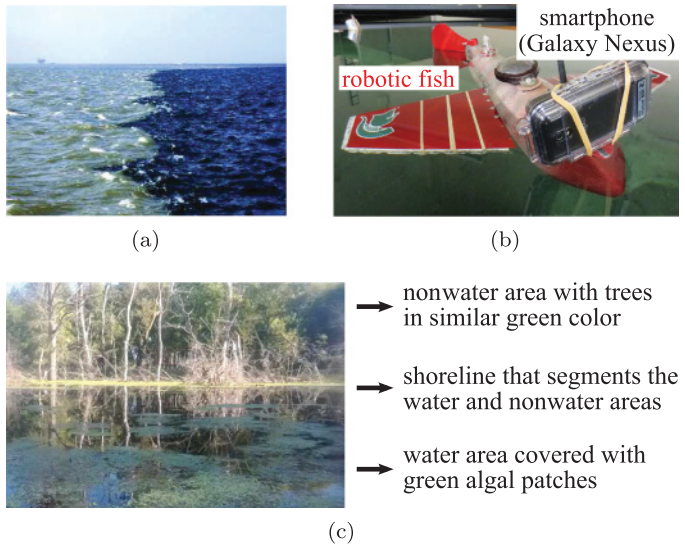


Fig. 1. (a) Algal patches in the Gulf of Mexico where the water area on the left side is covered by HABs and exhibits murky green color, 2011 (Photo Credit: Louisiana Universities Marine Consortium [Southern Regional Water Program 2011]); (b) A Samba prototype integrating a Samsung Galaxy Nexus smartphone with a robotic fish; (c) A sample image captured by the Samba prototype in an inland lake, where the water and nonwater areas are separated by a shoreline; the trees exhibit similar color with the algal patches in the water area.

about the aquatic process such as color and spatial distribution. Such information is often important for the authorities to conduct hazard analysis and take contingency measures. Third, current smartphones are capable of running advanced computer vision (CV) algorithms for real-time image processing. Last, the price of smartphones has been dropping drastically in the past few years, making it economically possible to deploy many Samba nodes to monitor a large affected area. Owing to these features, Samba represents an *energy-efficient*, *low-cost*, yet *intelligent* mobile sensing platform for aquatic monitoring.

Despite the aforementioned advantages, we need to address several major challenges in the design of Samba. First, aquatic processes are often scattered as patches over large geographic areas and spatiotemporally developing [Brekke and Solberg 2005; Platt et al. 2003], which create challenges for fine-grained monitoring. Continuous visual sensing can track their development, which, however, imposes significant energy and computation overhead to the smartphone. Second, the phones' built-in cameras have limited field of view. Although the controllable mobility of robot can help improve the sensing coverage, aquatic locomotion may incur high energy consumption. Last, existing vision-based detection algorithms, which often use background subtraction [Shapiro and Stockman 2001], perform poorly on the images captured in the aquatic environment. For example, the patches of the target aquatic process present in the camera view often block the background, making it difficult to learn the background model. Moreover, the target detection may be greatly affected by various dynamics such as blurred images caused by waves, highly variable illuminance, and complex nonwater area in the image. These uncertainties can lead to excessive false alarms. Figure 1(c) shows a sample image captured by a Samba prototype in an inland lake, where the trees on the shore exhibit similar green color with the algal patches on the water, potentially resulting in false detections.

In this article, we make the following contributions to address the above challenges:

1) We propose an inertial-sensing-assisted image segmentation approach to identifying the water area in the image. By focusing on the water area, Samba can reduce the false alarms caused by the nonwater area. A key novelty of this approach is to leverage the energy-efficient inertial sensing to replace the compute-intensive algorithms for visual feature extraction used in image segmentation. Specifically, Samba first learns the mapping models that project inertial sensor readings to visual features. It then uses these models and real-time inertial sensor readings to estimate the visual features (e.g., the shoreline in Figure 1(a)) for image segmentation without actually extracting them from the images.
2) We propose several lightweight and robust CV algorithms to detect harmful aquatic processes in dynamic environment. Our vision-based detection algorithms, consisting of back projection and patch identification, detect the existence of a target process based on its distinctive color features. The algorithms are specifically designed to address the dynamics in aquatic monitoring such as highly variable illuminance and the camera's internal noises.
3) We design a feedback-based robot rotation control algorithm that increases coverage and maintains a desired level of monitoring resolution on the developing aquatic process (e.g., the area expansion of an algal patch between two consecutive observations during the rotation). Based on the dynamics of the target process, the control algorithm adapts the rotation speed of Samba to meet the user's requirement on monitoring resolution.
4) We implement a prototype of Samba and evaluate it through real field experiments in a small inland lake and extensive lab experiments. The results show that Samba can accurately detect harmful aquatic processes, maintain the desired monitoring resolution, and achieve a system lifetime up to nearly 2 months.

The rest of this article is organized as follows. Section 2 reviews the related work. Section 3 provides an overview of Samba. Sections 4 through 6 present the hybrid image segmentation, aquatic process detection, and adaptive rotation control algorithms, respectively. Section 7 discusses a variant rotation control approach and other use cases of Samba. Section 8 describes the prototype implementation. Section 9 evaluates the performance of Samba. Section 10 concludes this article.

2. RELATED WORK

Current approaches to monitoring harmful aquatic processes fall into four categories: manual spotting; patrol boat–assisted survey [Ammerman and Glover 2000]; AUV-based autonomous sensing [Rudnick et al. 2004]; and remote sensing by balloon [Kako et al. 2012], aircraft [Hu et al. 2003], and satellite imaging [Brekke and Solberg 2005]. Manual spotting, although viable for small-scale monitoring, is labor intensive and unreliable. Approaches using patrol boats and remote sensing are prohibitively expensive for long-term monitoring, especially when the target process is scattered over vast geographic areas. Likewise, the adoption of AUV-based monitoring is limited by the high manufacturing and operational costs of the platform.

Several research efforts have explored the integration of cameras with low-power sensing motes. SensEye [Kulkarni et al. 2005] incorporates a Cyclops camera into a Stargate platform [Stargate User's Manual 2014] to detect objects at a 128×128 resolution. In Teixeira et al. [2006], a camera module is installed on an XYZ node [Lymberopoulos and Savvides 2005] for gesture recognition at a resolution of 256×64. These camera-based platforms can only conduct simple image processing tasks at low resolutions due to their limited computation capabilities. Recently, mobile sensing using a smartphone receives increasing interest due to its rich computation, sensing, and storage resources. For example, a face recognition method using sparse representation

is designed for smartphones [Shen et al. 2014]. Different from existing vision-based systems, Samba has to address several unique challenges in aquatic monitoring, such as highly variable illuminance and dynamic aquatic process development. Moreover, we need to make the image processing pipeline of Samba highly energy-efficient to enable long-term autonomous monitoring.

In our previous work [Wang et al. 2014, 2016], we developed a smartphone-based robotic platform named SOAR to detect arriving debris objects. Samba fundamentally differs from SOAR in four aspects. First, Samba achieves significantly higher energy efficiency by integrating inertial sensing with visual sensing through a learning-based scheme. According to our experiments, Samba consumes 97% less energy than SOAR in processing an image frame. Second, the generic design of Samba enables the monitoring of either moving or static aquatic processes such as oil spills and HABs. In contrast, SOAR is designed to detect moving debris objects only while treating all the static targets as background. Moreover, SOAR identifies debris objects using a pixel-wise approach, while Samba detects target aquatic processes based on their color features without modeling each pixel. As a result, compared with SOAR, Samba drastically decreases the overhead of robot mobility control, since it does not need to maintain the correspondence between the pixels in different frames during the rotation movement in dynamic aquatic environment. Last, Samba adopts a feedback-control–based algorithm that adapts the robot's movement based on the detected dynamics of the target, while the movement of SOAR is primarily driven by prior knowledge such as an *a priori* arrival model of the debris objects.

Inertial sensing has recently been explored in various mobile sensing applications. In Hemminki et al. [2013], a set of inertial features are extracted from a phone's built-in accelerometer and used to identify the user's transportation mode. In Wang et al. [2013], accelerometer and gyroscope readings are employed to model vehicle dynamics and estimate the device's position inside the vehicle. In Faulkner et al. [2011], smartphones collaboratively detect an earthquake using their on-board accelerometers. The FOCUS system [Jain et al. 2013] mitigates shaken and blurred views during video recording based on measured acceleration. The V-Sense system [Chen et al. 2015] exploits measurements from various inertial sensors to detect dangerous maneuvers of vehicle steering. Different from the aforementioned inertial sensing applications, Samba employs inertial sensing as an energy-efficient alternative to visual sensing in compute-intensive image processing tasks.

Detecting targets of interest from captured images is a fundamental CV task. The traditional approach usually adopts background subtraction, which constructs a background model for each pixel according to historical observations [Shapiro and Stockman 2001]. However, background subtraction cannot be readily applied to monitoring aquatic processes such as oil spills and HABs. This is because the camera's view can be dominated by the target process for a long period of time when the camera is deployed in the affected area, making it difficult and even impossible to build the background model [Shapiro and Stockman 2001]. The latest approaches [Erhan et al. 2014; He et al. 2015] detect targets of interest without explicitly modeling the background. However, they usually require a significant amount of training data and incur heavy computation overhead. To address these issues, Samba features a set of detection algorithms that are specifically designed based on the distinctive color features of aquatic processes. The efficient design of these algorithms introduces affordable computation overhead for smartphones.

3. OVERVIEW OF SAMBA

Samba integrates an off-the-shelf Android smartphone with a robotic fish. The phone loads an app that implements the image processing and mobility control algorithms, including *hybrid image segmentation*, *aquatic process detection*, and *adaptive rotation*

control. The robotic fish receives a rotation schedule from the phone via wireline or wireless communication and propels Samba by beating its tail. Samba is designed to operate on the surface of relatively static waters with no or little water currents and monitor harmful aquatic processes such as oil spills and HABs. These processes typically disperse as patches in the aquatic environment and exhibit distinctive colors [Brekke and Solberg 2005; Platt et al. 2003]. To monitor a large affected area, multiple Samba nodes can be deployed to form a surveillance network. Their local observations can be sent back to a central server via the long-range communication interface on smartphones and stitched into a global map. In this article, we focus on the design of a single Samba node.

Due to the limited angular view of the phone's camera, it is challenging to monitor the scattered patches of the target process. Although extra optical components like fisheye lens can be used to broaden the camera view, their integration will complicate the system design and introduce additional computation overhead due to the distorted images [Shapiro and Stockman 2001]. Instead, we leverage the robotic fish's mobility to increase the sensing coverage. Specifically, Samba conducts orientation adjustment (i.e., rotation) by performing carefully controlled tail beats. The rotation mobility is much more energy-efficient than moving forward that requires continuous tail beats.

Before deployment, Samba is trained by sample images of the target aquatic process (TAP). The sample images are close-up photos of the TAP and can be provided by domain scientists. Samba can monitor multiple TAPs simultaneously when provided with their sample images. Samba conducts aquatic monitoring at a set of orientations, which are selected to ensure a full coverage of surrounding region given the angular view of Samba's on-board camera. After the initial deployment, it begins a TAP surveillance process consisting of multiple rounds. In each round, Samba keeps monitoring toward an orientation for a certain time interval. At the beginning of a round, Samba rotates to a new orientation and starts to capture images at a certain rate. For each image, it identifies the water area and executes several CV algorithms to detect the existence of TAP in real time. At the end of this round, Samba estimates the dynamics of TAP (e.g., development rate) and computes the monitoring interval for the next round. For example, if the TAP develops rapidly, Samba will shorten the monitoring interval such that it can rotate back to the current orientation sooner, keeping smooth track of the TAP development, and vice versa. When drastic development of the TAP is detected, Samba can alert the user by using the communication interface (cellular/Wi-Fi) on a smartphone. A primary design objective of Samba is to achieve long-term (up to a few months) autonomous monitoring given certain battery capacity. Between two image captures, Samba sleeps to reduce energy consumption. A novel feature of Samba is the inertial sensing–assisted image processing scheme that can lead to significant energy savings. In particular, Samba uses the learned models that partition an image based on inertial sensor readings, without actually extracting the visual features from the images that is often compute intensive. Specifically, Samba comprises the following three major components.

Hybrid image segmentation: By partitioning an image into water and nonwater areas, Samba performs aquatic process detection in the water area only. This approach reduces detection false alarms and computation energy consumption. Samba adopts a novel hybrid image segmentation framework, as illustrated in Figure 2(a). Specifically, it uses both vision-based and inertia-based segmentation modes. This hybrid approach learns regression-based mapping models that project the inertial sensor readings to the visual features for image segmentation. Therefore, Samba can partition an image based on the visual features estimated from the inertial sensor readings, avoiding executing the compute-intensive CV algorithms to actually extract the visual features. Samba

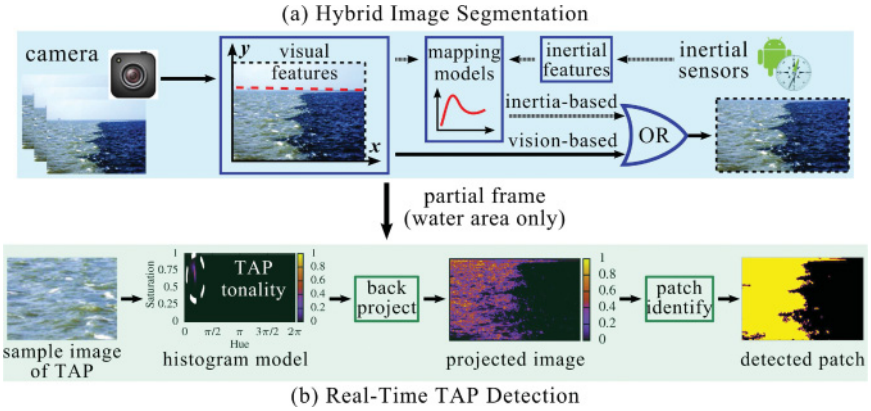


Fig. 2. The aquatic environment monitoring pipeline when Samba keeps an orientation. Samba periodically adjusts its orientation to adapt to the dynamics of surrounding target aquatic process (TAP) patches.

switches to vision-based segmentation if the mapping models need to be updated, for example, when the accuracy of inertia-based segmentation drops or Samba rotates to a new orientation.

Real-time TAP detection: This component detects TAP in the segmented images. As illustrated in Figure 2(b), it consists of two lightweight image processing modules (i.e., *back projection* and *patch identification*). First, Samba extracts robust color features of TAP in HSV color space, and performs back projection to identify the candidate pixels of TAP in each segmented image. The projected image is then passed to patch identification for probability thresholding, noise removal, and patch recognition. The patch identification can effectively deal with various environment and system dynamics such as highly variable illuminance and the camera's internal noises.

Adaptive rotation control: Samba monitors the surrounding TAP with full *circular coverage* and fine *temporal granularity* while meeting energy consumption constraints. Full circular coverage means that Samba rotates to fully cover the circular region around it such that newly developed patches within Samba's sensing radius will not be missed. As Samba needs to rotate to achieve full circular coverage while the TAP is developing or moving, fine temporal granularity means that the change of the TAP's property (e.g., area) between two Samba's consecutive observations toward a particular orientation is small. Fine temporal granularity enables the user of Samba to capture fine-grained details about the TAP development. For example, domain scientists can measure the growth rate of HABs seamlessly. To adapt to the dynamics of TAP that is primarily affected by environmental conditions, we develop a feedback control algorithm to maintain the desired temporal granularity. On the completion of a round, Samba estimates the dynamics of TAP (e.g., diffusion of oil slick and growth of HABs) based on detection results and then calculates a new rotation speed such that the expected temporal granularity in the next round can be maintained at the desired level.

4. HYBRID IMAGE SEGMENTATION

Image segmentation is the process of partitioning an image into multiple parts [Shapiro and Stockman 2001]. In aquatic monitoring, we adopt image segmentation to remove the nonwater area from the captured images, and thus perform the detection of TAP in the water area only. This can avoid the false alarms that occur in the nonwater area

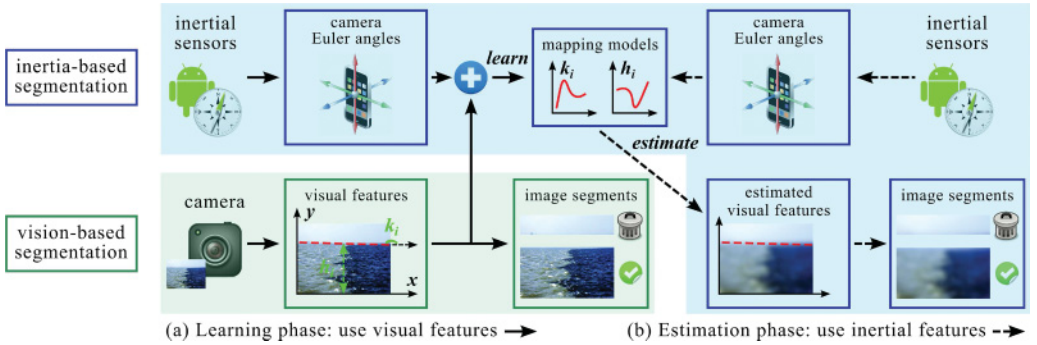


Fig. 3. The overall structure of hybrid image segmentation where Samba switches between the vision-based and inertia-based segmentation modes. In the online learning phase, Samba jointly uses inertial and visual sensing to learn regression-based mapping models, which project the camera Euler angles (obtained from the inertial sensors) to the extracted visual features (obtained from the camera). In the estimation phase, Samba utilizes the learned mapping models to estimate the visual features and conducts the image segmentation accordingly.

(e.g., those caused by trees in Figure 1(c)) and reduce computation in subsequent image processing tasks. Image segmentation is usually conducted based on visual features. For example, the shoreline can be used to identify the water area in inland lakes. However, most visual feature extraction CV algorithms are compute-intensive. According to our experiments (see Section 9.2.1), the Hough transform [Shapiro and Stockman 2001] for line extraction consumes more than 95% of the energy for processing an image. Moreover, the vision-based segmentation may fail due to the lack of differentiating color features and blurred images caused by waves. In this section, we propose a robust approach to overcoming the above limitations of vision-based segmentation.

4.1. Overview

In this article, we develop a novel hybrid image segmentation approach that utilizes both camera and inertial sensors on the phone, as illustrated in Figure 3. Inertial sensors provide the camera's transient pose, which can be used to guide the segmentation of captured images. To characterize a camera's projection, the visual sensing approach is susceptible to the quality of captured image and blocked line of sight, while inertial sensing will not be affected by these factors. Moreover, the energy consumption of inertial sensing is much lower than that of visual sensing. Our experiments show that the computation delay of inertia-based segmentation is only 2% of that of vision-based segmentation (see Section 9.2.1). The proposed hybrid approach aims to leverage these two heterogeneous sensing modalities. Specifically, it consists of an online learning phase and an estimation phase. The vision-based image segmentation is executed in the learning phase, and the inertia-based image segmentation is executed in the estimation phase. In our design, Samba switches between the two modes to save system energy while maintaining segmentation accuracy.

In the learning phase, images are segmented based on the extracted visual features, as shown in Figure 3(a). Meanwhile, the inertial sensor readings are converted to Euler angles, which describe the camera's orientation with respect to the world coordinate system. These angles, along with the corresponding visual features, are then used to learn the mapping models via regression. In the estimation phase, the visual features are estimated using the inertial sensor readings and the learned mapping models. These estimated visual features are used to guide the image segmentation, as shown in Figure 3(b). Therefore, the execution of the compute-intensive visual feature extraction

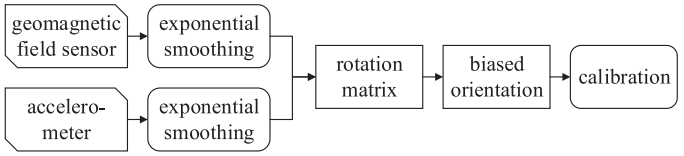


Fig. 4. Workflow of virtual orientation sensor.

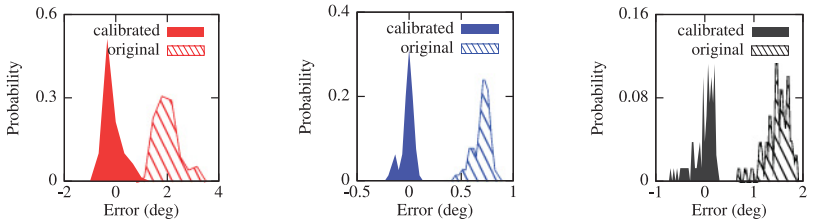


Fig. 5. PDF of orientation measurement errors before and after calibration.

CV algorithms is avoided. The hybrid approach periodically calibrates the inertial sensors and updates the mapping models in the learning phase, thus adapting to the possible environmental condition changes.

4.2. Measuring Camera Orientation

A key step in the hybrid image segmentation is to relate the inertial sensing results with the corresponding visual sensing results represented using the camera projection model. The camera's orientation characterizes its projection. Therefore, it is critical to accurately measure the camera's orientation. There are typically no built-in physical orientation sensors on Android smartphones.[1] Hence, we need to implement our own virtual orientation sensor based on other available inertial sensors such as accelerometer and geomagnetic field sensor. However, inertial sensor readings are often inaccurate and volatile. Studies have shown that the mean error for orientation measurements computed based on inertial sensor readings can be up to 30° [Blum et al. 2013]. To deal with the bias and random noise, we calibrate and denoise the sensor readings. Figure 4 shows the workflow of our virtual orientation sensor. It first uses inertial sensor readings to compute the rotation matrix that bridges the device coordinate system to the world coordinate system and then obtains the phone's orientation. Figure 5 compares the distributions of the orientation measurement errors before and after a calibration process on a Google Nexus 4 phone. The ground truth is measured using a protractor. We can see that the virtual orientation sensor is subjected to both biases and random noises, and the biases can be effectively removed by the calibration process. After the calibration process, the orientation measurement errors along all the three axes are smaller than 1°.

4.3. Feature Extraction

In order to establish the mapping models, Samba needs to extract features from both the camera's image frames and the inertial sensor readings obtained at the same time instants. In this article, we focus on the lake environment where the shoreline can

[1]The orientation sensor API has been deprecated in Android since version 2.2 [Android 2.2 Platform Highlights 2010].

help identify the water area. We assume that the shoreline is the longest visible line to Samba. This is reasonable as the shoreline usually crosses the captured images horizontally, as shown in Figure 1(c). Note that the shoreline is not static to Samba because of waves and Samba's rotational movements. Our approach can be easily extended to address other scenarios with different visual references for image segmentation. For image frame i, let k_i and h_i denote the shoreline slope and average vertical coordinate in the image plane, as illustrated in Figure 3(a). The phone can extract the shoreline using Hough transform [Shapiro and Stockman 2001] and thus obtain k_i and h_i. With these two parameters, frame i can be partitioned accordingly. Therefore, we define (k_i, h_i) as the *visual feature*s of frame i.

The camera's orientation, represented by Euler angles, is measured by the virtual orientation sensor. At runtime, we synchronize the orientation measurements with visual features obtained from the camera using the timestamps. We now derive their relationship based on the camera's perspective projection model [Shapiro and Stockman 2001]. We set the world coordinate system originated at the camera. We adopt α_i, β_i, and γ_i to denote the yaw, pitch, and roll angles of the camera when frame i is taken, and $\mathbf{P} = (x, y, z)$ to denote the coordinates of an observable point on the shoreline in the world coordinate system. We first project $\mathbf{P}$ to the Cartesian coordinate system that is originated at the camera and rotated by $(\alpha_i, \beta_i, \gamma_i)$ with respect to the world coordinate system. The projected coordinates, denoted by $\mathbf{P}' = (x', y', z')$, are given by

$$\mathbf{P}' = \mathbf{P} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma_i & \sin\gamma_i \\ 0 & -\sin\gamma_i & \cos\gamma_i \end{bmatrix} \cdot \begin{bmatrix} \cos\alpha_i & 0 & -\sin\alpha_i \\ 0 & 1 & 0 \\ \sin\alpha_i & 0 & \cos\alpha_i \end{bmatrix} \cdot \begin{bmatrix} \cos\beta_i & \sin\beta_i & 0 \\ -\sin\beta_i & \cos\beta_i & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Based on the similar triangles [Shapiro and Stockman 2001], we then project $\mathbf{P}'$ to the 2D image plane using the following formula:

$$\begin{bmatrix} x'' & y'' \end{bmatrix} = \begin{bmatrix} x' & y' \end{bmatrix} \cdot \begin{bmatrix} f_x \cdot g(z') & 0 \\ 0 & f_y \cdot g(z') \end{bmatrix},$$

where (x'', y'') denote the corresponding coordinates in the image plane, f_x and f_y are two hardware-dependent scaling parameters, and $g(z')$ is a scaling factor determined by z'.

Our following analysis shows that, under the above transforms, we can relate the camera's orientation $(\alpha_i, \beta_i, \gamma_i)$ to the visual features (h_i, k_i). We first derive k_i that represents the slope of the shoreline. Suppose (x''_1, y''_1) and (x''_2, y''_2) are two points on the extracted shoreline in frame i, and (x_1, y_1, z_1) and (x_2, y_2, z_2) are the corresponding points in the world plane. The shoreline slope in the image plane, that is, k_i, can be computed as

$$k_i = \frac{y''_1 - y''_2}{x''_1 - x''_2} = \frac{f_y}{f_x} \cdot \frac{(x_1 f_{i,3} - y_1 f_{i,4}) - (x_2 f_{i,3} - y_2 f_{i,4})}{(x_1 f_{i,1} - y_1 f_{i,2}) - (x_2 f_{i,1} - y_2 f_{i,2})} = \omega_1 \cdot \frac{f_{i,3} - \omega_2 f_{i,4}}{f_{i,1} - \omega_2 f_{i,2}}, \tag{1}$$

where $\omega_1 = f_y/f_x$ and $\omega_2 = (y_1 - y_2)/(x_1 - x_2)$ are two unknown but constant coefficients that are determined by the camera hardware and the shoreline slope in the world coordinate system. The 4-tuple $(f_{i,1}, f_{i,2}, f_{i,3}, f_{i,4})$ can be computed based on the camera's orientation by

$$\begin{aligned} f_{i,1} &= \cos(\alpha_i) \cdot \cos(\beta_i), \\ f_{i,2} &= \cos(\alpha_i) \cdot \sin(\beta_i), \\ f_{i,3} &= \sin(\alpha_i) \cdot \cos(\beta_i) \cdot \sin(\gamma_i) - \sin(\beta_i) \cdot \cos(\gamma_i), \\ f_{i,4} &= \sin(\alpha_i) \cdot \sin(\beta_i) \cdot \sin(\gamma_i) + \cos(\beta_i) \cdot \cos(\gamma_i). \end{aligned}$$

They interpret the camera projection model using orientation measurements. The above 4-tuple are defined as *inertial feature*s, which are related to the visual feature through the *mapping model* given by Equation (1). Similarly, we can derive the mapping model between another set of inertial features (with 3 unknown coefficients) and the vertical position of the shoreline in the image plane (i.e., h_i). With the mapping models, we can estimate the shoreline purely based on the inertial sensor readings and conduct image segmentation.

4.4. Learning Mapping Models

With the extracted visual and inertial features, Samba learns the mapping models that project the latter to the former. Based on Equation (1), we adopt a regression-based approach to estimating the unknown coefficients ω_1 and ω_2. Specifically, the training data instance from frame i can be expressed as $(k_i, f_{i,1}, f_{i,2}, f_{i,3}, f_{i,4})$, in which k_i is obtained from the camera and $(f_{i,1}, f_{i,2}, f_{i,3}, f_{i,4})$ are computed based on inertial sensor readings. Suppose the training dataset consists of N frames. We define the *feature set*, denoted by $\mathbf{F}$, as an $N \times 4$ matrix that contains the inertial features extracted from the N frames

$$\mathbf{F} = \begin{bmatrix} f_{1,1} & f_{1,2} & f_{1,3} & f_{1,4} \\ \vdots & \vdots & \vdots & \vdots \\ f_{N,1} & f_{N,2} & f_{N,3} & f_{N,4} \end{bmatrix}_{N \times 4}.$$

Moreover, we define the *observation vector*, denoted by $\mathbf{K}$, as an $N \times 1$ vector that contains the visual features (i.e., the shoreline slope), that is, $\mathbf{K} = [k_1, \ldots, k_N]^\intercal$. We then employ multivariate nonlinear regression to learn ω_1 and ω_2 with $\mathbf{F}$ and $\mathbf{K}$. Using the estimated ω_1 and ω_2, we can infer the shoreline slope from the inertial sensor readings based on the mapping model given by Equation (1).

We now discuss the impact of training dataset size (i.e., N) on system performance. The N training frames can be evenly selected from an updating period. Therefore, the value of N defines the frequency at which Samba switches between the vision-based and inertia-based segmentation modes. In order to perform regression, N should be no smaller than the number of coefficients to be learned (i.e., 2). In aquatic monitoring, N is expected to be larger than this lower bound to account for the dynamics resulted from camera shaking and sensor noises. Intuitively, a larger training dataset with diverse training data can improve the regression accuracy. However, it also imposes additional computation overhead for the phone. Specifically, with a larger N, visual feature extraction has to be conducted more frequently to generate training data. Moreover, the computation overhead of the regression also increases with N. In Section 9.2.5, we will evaluate the trade-off between regression accuracy and system overhead through experiments. At runtime, Samba can assess inertia-based segmentation accuracy in the learning phase by comparing it with vision-based segmentation, and adaptively configure N.

5. AQUATIC PROCESS DETECTION

The real-time TAP detection pipeline of Samba is illustrated in Figure 2(b). Although our approach is based on elementary CV algorithms, it is nontrivial to customize them for detecting TAP and efficiently implement them on a smartphone. Samba consists of two major image processing modules (i.e., *back projection* and *patch identification*). The back projection models the TAP by extracting its color histogram, which is built using selected channels in HSV. The HSV representation is more robust to highly variable illuminance than the RGB color space [Maree et al. 2005]. It then detects candidate pixels of TAP in the image segment of water area. The patch identification removes

the salt-and-pepper noises from the projected segmented image and then identifies the TAP patches.

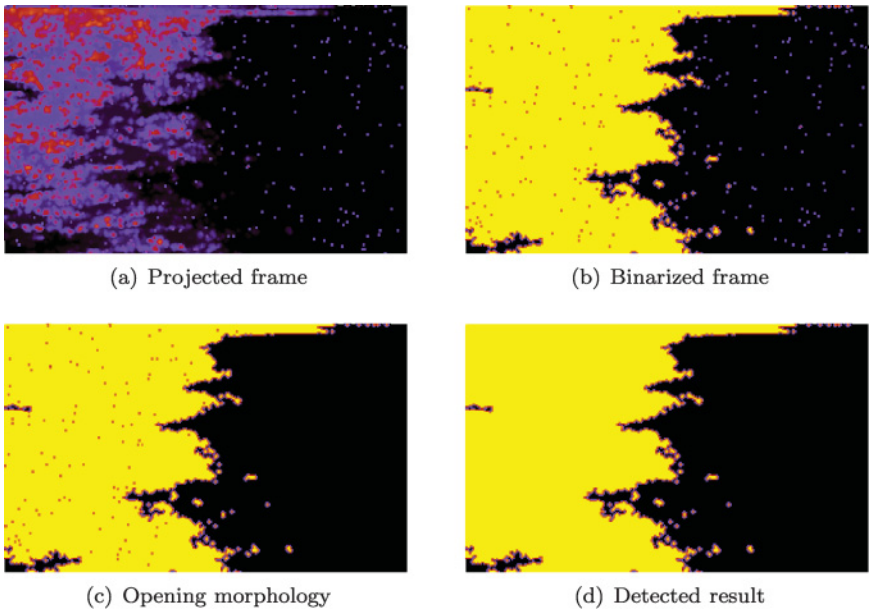
5.1. Back Projection

Back projection is a CV technique that characterizes how well the pixels of a given image fit a histogram model [Shapiro and Stockman 2001]. In this article, we adopt back projection to identify the existence of TAP in the captured frames given its unique features. The patches of TAP are often scattered over large water area. For example, the HABs occurred at Lake Mendota in Wisconsin produce algal patches spread over a 15km^2 water area [HABs and Lake Mendota 2007]. Therefore, Samba may have a camera view dominated by the TAP patches when deployed in an affected area. The widely used target detection approach based on background subtraction is thus not applicable, as it needs to build a background model without the TAP. Moreover, this approach requires all the captured images to be aligned and thus has limited feasibility in wavy waters. Our proposed approach is motivated by the fact that a TAP usually has featured and nearly constant colors. For example, oil slicks often appear to be brighter than the surrounding water surface [Brekke and Solberg 2005], and the typical color of HABs is green or red [Platt et al. 2003]. Therefore, to perform detection, we can match the pixels in a new frame with the histogram model constructed with offline sample images of the TAP.

Back projection constructs the histogram model based on sample images as follows. Let I_0 denote a sample image of the TAP. We first convert the representation of I_0 to HSV (Hue, Saturation, and Value) model. In HSV, the hue channel represents the color, the saturation channel is the dominance of hue, and the value channel indicates the lightness of the color. The hue and saturation channels are robust to illumination changes [Maree et al. 2005] and hence effective in interpreting color features in the presence of reflection on water surface. Thus, we adopt the hue-saturation histogram and calculate it from I_0. We equally divide the range of hue $[0, 360)$ and the range of saturation $[0, 1]$ into $[0, h_2, h_3, \ldots, h_{p-1}, 360)$ and $[0, s_2, s_3, \ldots, s_{q-1}, 1]$, respectively, to compute the color histogram $\mathbf{M}_{q\times p}$. Specifically, the $(i, j)^{\text{th}}$ entry of $\mathbf{M}$ is the frequency of pixels in I_0 with saturation within $[s_i, s_{i+1})$ and hue within $[h_j, h_{j+1})$. Note that the color histogram can be obtained based on multiple images by repeating the above process. Let $\widetilde{\mathbf{M}}$ denote the normalized histogram where each element characterizes the probability of the corresponding color representing the TAP. Therefore, $\widetilde{\mathbf{M}}$ depicts the tonality of the TAP, which is used as its histogram model. Figure 2(b) shows a sample image of HABs that occurred in the Gulf of Mexico [Southern Regional Water Program 2011] and the extracted hue-saturation histogram model.

When a new segmented frame (denoted by I_t) is available, we leverage the histogram model $\widetilde{\mathbf{M}}$ to detect the existence of TAP. For each pixel $\mathbf{p}_{m,n}$ in I_t, we first extract its hue $h_{m,n}$ and saturation $s_{m,n}$, and locate the corresponding element in $\widetilde{\mathbf{M}}$, that is, $\widetilde{\mathbf{M}}(s_{m,n}, h_{m,n})$. We then construct a projected frame I_1' that is in the same size of I_t but replaces each pixel $\mathbf{p}_{m,n}$ with $\widetilde{\mathbf{M}}(s_{m,n}, h_{m,n})$. Note that $\widetilde{\mathbf{M}}(s_{m,n}, h_{m,n})$ characterizes the probability that $\mathbf{p}_{m,n}$ in I_t represents a pixel of TAP. Visually, the brighter a pixel in I_1' is, the corresponding pixel in I_t is more likely to be from the TAP. An example of the projected frame I_1' is shown in Figure 6(a).

In back projection, each pixel in the new frame is classified based on how well it matches the color histogram obtained from TAP sample images. Therefore, the similarity in color between the TAP and water area affects the detection performance. In this article, we adopt a *color similarity* [Smith and Chang 1997] metric to measure the color resemblance. It quantifies the similarity between any two colors based on their proximity in the cylindrical HSV model. For two colors indexed by (h_i, s_i, v_i) and



(a) Projected frame (b) Binarized frame

(c) Opening morphology (d) Detected result

Fig. 6. Sample patch identification process.

(h_j, s_j, v_j), the color similarity, denoted by η, is given by

$$\eta = 1 - \sqrt{\frac{(v_i - v_j)^2 + (s_i \cos h_i - s_j \cos h_j)^2 + (s_i \sin h_i - s_j \sin h_j)^2}{5}}.$$

The η ranges from 0 to 1, where the lower bound 0 indicates the highest level of color contrast and the upper bound 1 represents the same color. Therefore, a larger value of η suggests a stronger color resemblance between the TAP candidate and water area, and hence the TAP candidate is less likely to be identified by the patch identification module in Section 5.2. In Section 9.2.6, we will evaluate the impact of η on detection performance through experiments.

5.2. Patch Identification

As illustrated in Figure 6, patch identification works on the projected image and identifies the TAP patches. In the projected frame shown in Figure 6(a), each pixel maintains a value within [0, 1], which represents the probability that it is from the TAP. We ignore the pixels with low probabilities of being TAP. The probability for a pixel being TAP is referred to as *matching probability*. In this section, we present a robust approach to identifying TAP pixels from the projected frame. Specifically, we model the distribution of the matching probabilities in the projected frame using a Gaussian mixture model (GMM) [Shapiro and Stockman 2001], which is a widely used model to characterize complex distributions. The choice of GMM is mainly due to the observations that, the water pixels have similar colors and low matching probabilities that can be described by a single Gaussian distribution, while the matching probabilities of TAP pixels often exhibit an irregular and complex distribution. Such a mixed pattern can be well captured by a GMM model. We adopt OpenCV's implementation of the Expectation Maximization (EM) clustering algorithm [Moon 1996] to learn the GMM. To determine the number of Gaussians, we try a range of settings and select the value that minimizes the fitting error. Figure 7 shows the distribution of the matching probabilities and the

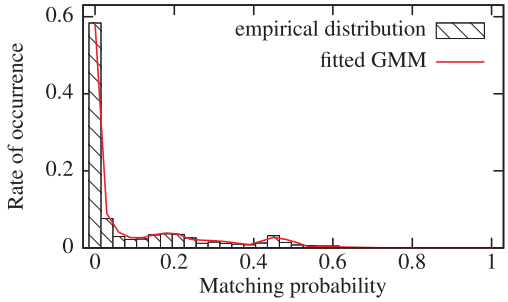


Fig. 7. The distribution of the matching probabilities and the fitted GMM.

fitted GMM for the projected frame shown in Figure 6(a). The fitted GMM comprises three one-dimensional Gaussians. In particular, the Gaussian with the smallest mean corresponds to the water area. This is from the said fact that water pixels have similar colors of low matching probabilities. We define a threshold to be the sum of the mean value of this Gaussian and 2.5 times of its standard deviation. For any pixel with a matching probability higher than this threshold, we consider it as a candidate TAP pixel and round its matching probability to 1; otherwise, we classify it as a water area pixel and round its matching probability to 0. A key advantage of this GMM-based approach is that it can capture the irregular and complex pattern of the matching probabilities of TAP pixels and accurately model the matching probabilities of water pixels. Based on the accurate model for water pixels, this approach can reliably detect the TAP pixels. At runtime, Samba runs the EM algorithm based on a captured frame and updates the TAP pixel detection threshold when rotating to a new orientation.

The aforementioned GMM approach will binarize the projected frame. The binarized frame, as shown in Figure 6(b), often contains randomly distributed noise pixels. This is because the back projection is conducted in a pixel-wise manner where the labeling of candidate TAP pixel can be affected by the camera's internal noises. To remove these false alarms, we apply the opening morphology operation [Shapiro and Stockman 2001]. It consists of an erosion operation followed by a dilation operation, where the former eliminates the noise pixels and the latter preserves the shape of true TAP area. Figure 6(c) depicts the resulted frame after noise removal. We then perform region growing [Shapiro and Stockman 2001] to identify the TAP patches. In particular, it uses the detected pixels as initial seed points and merges connected regions to obtain the candidate TAP patches. Finally, we apply another threshold to exclude the small patches. The patch with a small area in the frame usually indicates a false alarm. Figure 6(d) depicts the final detection result. We note that the detected TAP patches can be larger than the ground truth due to the reflection of trees on the water surface. One possible approach to addressing this is to improve the histogram model accuracy by using more selective sample images of TAP.

6. ADAPTIVE ROTATION CONTROL

Samba periodically adjusts its orientation to monitor the surrounding TAP patches with full circular coverage and fine temporal granularity. Feedback motion control for the fish tail beating can maintain the orientation of Samba in relatively static waters. Thus, Samba can remain stationary to the TAP patches and detect their area expansions. Taking HABs monitoring as an example, the temporal granularity can be characterized by the area expansion of an algal patch between two consecutive observations toward the patch during the robot rotation. However, this is challenging because the development of an aquatic process is heavily affected by changeable and

even unpredictable environmental conditions. For example, the diffusion of oil slick can be affected by water salinity and temperature [LaRoche et al. 1970], and the growth of HABs is sensitive to local nutrient availability [Hecky and Kilham 1988]. To address such uncertainties, we propose a rotation control algorithm, which maintains the temporal granularity at the desired level by adjusting the monitoring interval in each round.

6.1. Dynamics of Aquatic Process

The sensing coverage of a camera is described by its *field of view* (FOV) where any target with dimensions greater than some thresholds can be reliably identified [Shapiro and Stockman 2001]. FOV is typically modeled as a sector originated at the camera with an angular view θ and a radius R, in which θ depends on lens and R can be measured for a particular application. Since a TAP is likely to be scattered over a large area [Brekke and Solberg 2005; Platt et al. 2003], we define the *surveillance region* of Samba as the circular area originated at the robot with a radius R. Limited by θ, Samba needs to change its orientation to cover all the TAP patches within its surveillance region. To address both the TAP development and the camera's limited angular view, Samba needs to continuously rotate to achieve the desired temporal coverage of each orientation.

The robot rotation initiates a monitoring round. Each round has a camera orientation and an associated monitoring interval. In our design, we equally divide the circular surveillance region into several sectors based on the camera's angular view θ such that the surveillance region can be fully covered by these discrete orientations. During a round, Samba remains stationary toward an orientation and conducts hybrid image segmentation and TAP detection at the user-specified sleep/wake duty cycle rate. For each frame, the *severeness* of TAP can be measured by the total area of detected patches in the frame. Thus, the area expansion of TAP patches characterizes the dynamics. Using frames captured at different time instants, Samba can estimate the dynamics of TAP by computing the change in severeness.

To achieve full circular coverage and fine temporal granularity of monitoring with limited energy budget, the rotation of Samba needs to be carefully scheduled. Samba controls the monitoring interval of each round to adapt to the dynamics of TAP. We define *rotation speed*, denoted by ρ, as the rotated angle over a monitoring interval. Note that the rotated angle for each round is fixed. For simplicity of exposition, the following discussion focuses on rotation speed. If the TAP spreads fast, ρ is expected to be large to capture the fast dynamics. When the development of TAP slows down, Samba may decrease ρ accordingly to save energy. Let ε denote the dynamics of TAP measured by the camera. We define the monitoring resolution Δs as the change in severeness between two consecutive observations toward a certain orientation. Therefore, Δs depends on the dynamics of TAP and ρ. For example, the spread of diffusion process is approximately linear with time [Crank 1979], that is, $\Delta s = \varepsilon \cdot 2\pi/\rho$. In Section 6.2, a robot rotation control algorithm is designed based on this model, and it can be easily extended to address other models of TAP dynamics.

6.2. Robot Rotation Control

Our objective is to maintain a desired resolution on severeness change, denoted by δ, under various environment and system uncertainties. Specifically, δ is the desired change of the total TAP area between two consecutive observations toward a certain orientation. To save energy, Samba rotates at a low speed as long as it can provide the required resolution. The setting of δ describes how fine the user aims to keep track of the TAP development. Figure 8 shows the block diagram of feedback control loop, where $G_c(z)$, $G_p(z)$, and $H(z)$ represent the transfer functions of the rotation scheduling algorithm, the dynamics model of TAP, and the feedback, respectively. In particular,

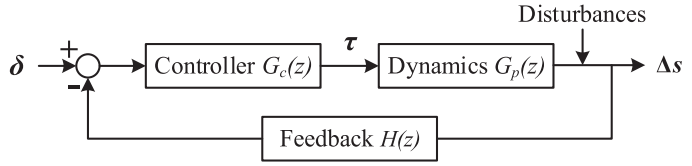


Fig. 8. The closed loop for robot rotation control.

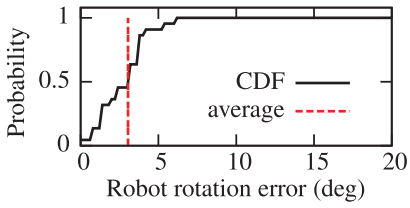


Fig. 9. CDF and average of errors for Samba orientation adjustment.

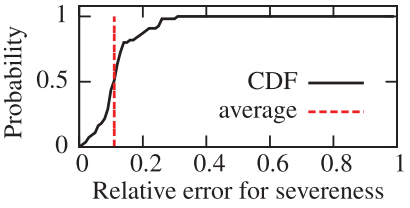


Fig. 10. CDF and average of relative errors for detected TAP area.

the desired resolution δ is the *reference*, and the actually detected severeness change Δs is the *controlled variable*. Because Δs is a nonlinear function of ρ, we define $\tau = 2\pi/\rho$ as the *control input* to simplify the controller design. Thus, $\Delta s = \varepsilon \cdot \tau$, and its z-transform is $G_p(z) = \varepsilon$. In each round, Samba updates τ for the next round and sets ρ accordingly. As the feedback control will take effect in the next round, $H(z) = z^{-1}$, representing the delay of one orientation adjustment. Given that the system is of zero order, it is sufficient to adopt a first-order controller to maintain the stability and convergence of the control loop [Ogata 1995]. Therefore, we set $G_c(z) = \frac{a}{1-b\cdot z^{-1}}$, where $a > 0$ and $b > 0$. Following the standard method for analyzing stability and convergence [Ogata 1995], the stability and convergence condition can be obtained as $b = 1$ and $0 < a < 2/\varepsilon$.

The uncertainties are modeled as disturbances in the control loop shown in Figure 8. First, Samba suffers rotation errors since the robot may not head exactly toward the desired orientation due to complex fluid dynamics. Such errors reflect the uncertainties in motion control. Figure 9 evaluates this type of error by the discrepancies between desired and actual orientations, where we collect the data using our Samba prototype (see Section 8) in wavy water. Second, the detected severeness of TAP exhibits variance. As the detection of severeness is conducted in a pixel-wise manner, it can be affected by dynamics like the camera's internal noises. Such errors reflect the uncertainties in visual sensing. We define the relative error for Δs as the absolute error to the ground truth of TAP area in the image. For each captured image, we manually pinpoint the boundary of algal patches to provide the ground truth. Figure 10 plots the CDF and average of this relative error when Samba fixes its orientation and monitors a still TAP. Without special care of the aforementioned uncertainties, they may significantly increase the volatility of the feedback-based rotation scheduling. For example, Samba may rotate at fast changing speeds. To mitigate the impact of such considerable uncertainties, we specifically design the controller $G_c(z)$ as follows. From control theory [Ogata 1995], the effects of injected disturbances on the controlled variable Δs can be minimized if the gain of $G_c(z)G_p(z)H(z)$ is set as large as possible. By jointly considering the stability and convergence, we set $a = 2c/\varepsilon$, where c is a relatively large value within $[0, 1]$. In the experiments, we set c to be 0.85.

Implementing $G_c(z)$ in the time domain gives the robot rotation scheduling algorithm. According to Figure 8, we have $G_c(z) = \tau(z)/(\delta - H(z)\Delta s)$. From $H(z)$ and $G_c(z)$, the control input can be expressed as $\tau(z) = z^{-1}\tau(z) + 2b\varepsilon^{-1}(\delta - z^{-1}\Delta s)$, and its time-domain

implementation is given by $\tau_k = \tau_{k-1} + 2b\varepsilon^{-1}(\delta - \Delta s_{k-1})$, where k is the count of orientation adjustments. The average rotation speed to be set for the k^{th} round is thus given by $\rho_k = 2\pi/\tau_k$.

7. DISCUSSIONS

7.1. Irregularly Distributed TAP Monitoring

In this section, we discuss an extension to our rotation control approach to monitoring the TAP that is irregularly distributed around Samba. In previous sections, we assume that Samba is surrounded by TAP patches. Thus, our feedback control algorithm schedules the monitoring interval toward an orientation per control cycle. Specifically, it determines the monitoring interval toward the next orientation based on the detected area expansion of TAP patches in the current orientation. This approach can be easily extended to monitor irregularly distributed TAP. In this case, we can define the controlled variable as the total TAP area expansion detected during the previous full round rotation. Accordingly, based on the detected TAP area expansion, a feedback control algorithm schedules the total monitoring interval for the next full round rotation to adapt to the TAP development. The new total monitoring interval can be allocated proportionally to all orientations based on the detected area expansions in individual orientations in the last round. In this approach, the orientations with faster area expansions will be allocated with longer monitoring intervals. Nevertheless, it is still important for Samba to conduct full round rotation to not to miss newly developed TAPs, where an orientation without TAP can be allocated with a minimum monitoring interval.

7.2. Extended Use Cases

We now discuss other potential use cases of Samba. Although our discussion has been mainly focused on HABs, Samba can be used to monitor other TAPs such as aquatic debris and oil spills. These distinctive phenomena on the water surface are observable to Samba if the vision-based detection module is trained with proper sample images. Moreover, Samba can also be used to monitor the status of a fish farming pond for an extended period of time. In pisciculture, infectious diseases and lack of dissolved oxygen in water due to various reasons are two major threats. Samba can be deployed to detect fish kills with floating heads, tails, or bellies up on the water surface. As the fish has distinctive colors from the water area, Samba can effectively identify abnormal events and send immediate warnings to the farmer. Last but not least, Samba can be used for various security monitoring applications. For example, for water systems in closed fishing seasons, Samba can be deployed to detect fishing boats and fishnet buoys. Pictures of suspicious objects can be transmitted using cellular networks to the authorities to examine and take actions.

8. IMPLEMENTATION

We have built a proof-of-concept prototype of Samba for evaluation. Figure 1(b) shows the Samba prototype that integrates a Galaxy in a water-proof enclosure with a robotic fish swimming in a water tank in our lab. The hybrid image segmentation, TAP detection, and adaptive rotation control algorithms presented in Sections 4 through 6 are implemented on Android. System evaluation is mainly conducted on two phones, including a Samsung Galaxy Nexus (referred to as *Galaxy*) and a Google Nexus 4 (referred to as *Nexus4*) running Android 4.3 Jelly Bean and 4.4 KitKat, respectively. Galaxy is equipped with a 1.2GHz dual-core processor and 1GB memory, and Nexus4 is equipped with a 1.5GHz quad-core processor and 2GB memory. They are representative low- and mid-end Android smartphones. Moreover, Android 4.4 introduces the

new runtime environment Android Runtime (ART) that features ahead-of-time compilation [Android Runtime 2014]. We will investigate the performance improvement in app execution of ART over the Dalvik process virtual machine used in Android 4.3. The app takes about 6.24MB of storage on the phone after installation and requires about 10.8MB RAM allocation while running on a frame resolution of 720×480. To exploit the multicore computation capability, the visual feature extraction, mapping models learning, and TAP detection are implemented in separate threads.

The communication between the phone and the fish can be established in different ways. For example, wireless connections such as Wi-Fi, Bluetooth, ZigBee and Z-Wave can be used. An alternative solution is wireline communication through USB or an audio jack, which can be more energy-efficient since the phone does not need to keep the communication processor awake. Moreover, the USB connection enables power transfer to the phone. In our prototype, we use a host computer to relay the wireless communications between the phone (via Wi-Fi) and the robotic fish (via ZigBee). Thus, under our current implementation, the lifetime of Samba presented in Section 9.2.3 is underestimated. In the future, we plan to set up direct wireline connection between the phone and the fish.

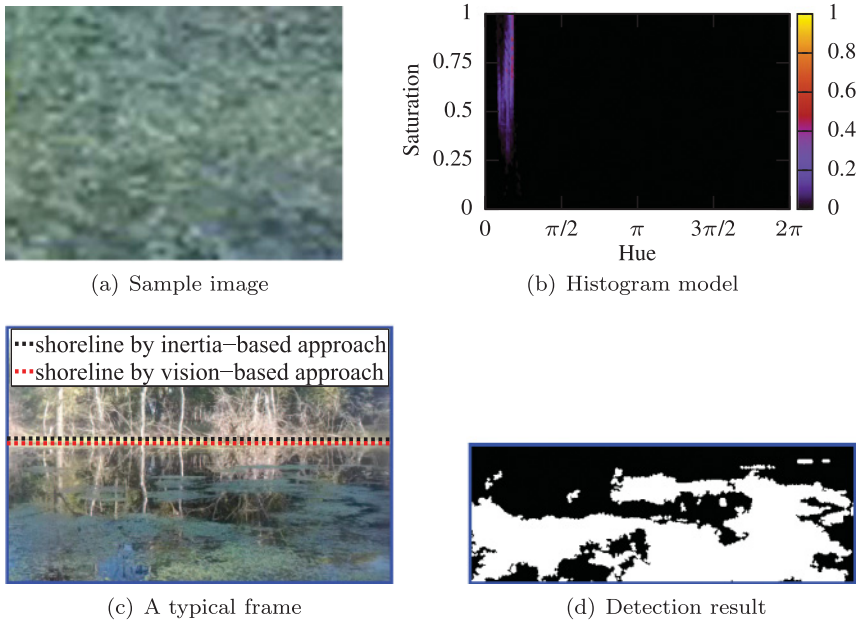
On initialization, Samba extracts the hue-saturation histogram of TAP based on provided sample images. When a new frame is available, it first conducts image segmentation to identify the water area. After the mapping models are learned, Samba switches between the vision-based and inertia-based segmentation modes according to the frequency specified by N (See Section 4.4). Then, the robot executes the TAP detection algorithms on the segmented frame of water area. During the implementation, we find that the Hough transform, which is used to extract the shoreline, incurs excessive computation overhead. Note that we use OpenCV's implementation of the Hough transform through Java Native Interface. We also examined the frame processing performance when the app uses ART. According to our measurements, ART can reduce the system delay by about 20% over Dalvik.

9. PERFORMANCE EVALUATION

We evaluate Samba through field experiments, lab experiments, and trace-driven simulations. The field experiments thoroughly test Samba's performance in detecting real HABs in an inland lake. The lab experiments evaluate system overhead, monitoring effectiveness under a wide range of environmental conditions, as well as the overall performance of a fully integrated Samba node. The trace-driven simulations examine the performance of adaptive rotation control with varied settings. Our results show that Samba can achieve reliable detection performance in the presence of various dynamics while maintaining a lifetime up to nearly 2 months.

9.1. Field Experiments

To test the detection performance of Samba in real world, we deployed our prototype in an inland lake with an area of 200,000ft^{2}on September 22, 2014. Part of the lake was covered by patches of filamentous algae [Boyd 2003] that exhibit pale green color, as shown in Figure 1(c). During the experiments, the average illuminance was around 1,082 lux, as measured by phone's built-in ambient light sensor. The impact of significant illuminance change on Samba will be evaluated in Section 9.2.7. We set the frame resolution to be 720×480 and the frame rate to be 0.5fps. The hybrid image segmentation updates the mapping models every 20 frames (i.e., $N = 20$). Because the HABs at the test site were in a stable stage, we focused on evaluating the detection performance without rotating the robot. Samba runs the hybrid image segmentation and TAP detection algorithms consecutively on each frame. A total of 5,211 frames were captured and processed. For each frame, we manually pinpoint the boundary of algal patches to



(a) Sample image (b) Histogram model

(c) A typical frame (d) Detection result

Fig. 11. Sample HABs detection in the field experiments.

provide the ground truth. For comparison, we adopt a baseline approach that uses the same sample images but constructs the color histogram model in the RGB color space. The RGB-based baseline approach is executed offline using the captured frames. The purpose of our field experiments is threefold. First, we test the detection performance of Samba in a real aquatic environment with complex background and colors. Second, we evaluate the real-time execution of hybrid image segmentation and TAP detection algorithms. Last, we validate that Samba can effectively reduce the false alarms caused by the nonwater area through image segmentation.

9.1.1. Sample HABs Detection. Figure 11 depicts a sample HABs detection in the field experiments. An image of the algal patch, as shown in Figure 11(a), is used as the sample image for the detection pipeline. Figure 11(b) shows the normalized hue-saturation histogram, in which each element characterizes the probability that the corresponding color represents the HABs. Therefore, a majority of colors in the histogram are of near-zero probability. For each frame, Samba first extracts the water area by locating the shoreline through hybrid image segmentation. Figure 11(c) shows a typical frame captured by Samba, where the red and black dashed lines represent the shorelines obtained by the vision-based and inertia-based image segmentation, respectively. As we can see, the inertia-based approach yields a fairly accurate estimation of shoreline, which is partially due to the calm water during the experiment. In Section 9.2.4, we will evaluate the performance of hybrid image segmentation under more dynamic environment. On the segmented water area, Samba executes the TAP detection algorithms. Figure 11(d) presents the detection result after back projection and patch identification. We can observe that our TAP detection algorithms effectively identify the algal patches in the segmented frame.

9.1.2. Detection Performance. We now evaluate the detection performance of Samba. For each frame, we define the *positive overlap ratio* as the ratio of the overlap area between the detected TAP area and the ground-truth TAP area to the ground-truth

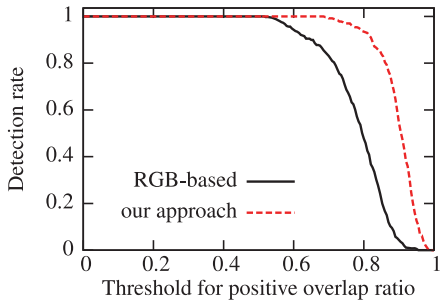


Fig. 12. Detection performance under various thresholds for positive overlap ratio.

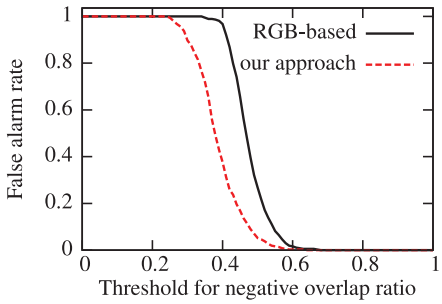


Fig. 13. False alarm rate under various thresholds for negative overlap ratio.

TAP area. Hence, the positive overlap ratio characterizes the effectiveness of detection algorithms in a frame. Given a threshold on positive overlap ratio, we calculate the *detection rate* as the ratio of the frames with positive overlap ratio larger than this threshold to the total frames with TAP patches. Figure 12 plots the detection rate versus the threshold for positive overlap ratio for our approach (i.e., using hue-saturation histogram) and the RGB-based baseline approach, respectively. When the positive overlap ratio is lower-bounded at 0.8, Samba can achieve detection rate up to 94%. Moreover, our approach yields consistently higher detection rates than the baseline approach. In the field experiments, the HABs and water area have a color similarity η of 0.88, which represents a rather strong color resemblance (see Section 5.1). In Section 9.2.7, we will evaluate the impact of η on detection performance.

9.1.3. Effectiveness of Image Segmentation. In TAP detection, we define the *negative overlap ratio* as the area of false TAP detection to the ground-truth TAP area in each frame. We say a false alarm occurs when the negative overlap ratio exceeds a given threshold. We calculate the *false alarm rate* as the ratio of the frames with false alarms to the frames without TAP. In the field experiments, the false TAP detections mainly result from the captured shore area. In particular, the trees on the shore, with a color similarity η of up to 0.97 with the target filamentous algae, are the major reason of the false TAP detections. Figure 13 plots the false alarm rate versus the threshold for negative overlap ratio for our approach and the RGB-based baseline approach, respectively. We find that our approach achieves consistently lower false alarm rates than the baseline approach. This is because our approach can characterize the TAP color features more effectively. We then validate the effectiveness of image segmentation in reducing false alarms. Image segmentation allows Samba to perform HABs detection in the water area only. Figure 14 compares the false alarm rates for our approach and the RGB-based baseline approach, respectively, with and without image segmentation. The reported results are obtained based on a threshold for the negative overlap ratio of 0.5. We can see that by applying image segmentation, the false alarm rate of our approach is reduced by over 90%. Moreover, the baseline approach also benefits from image segmentation but still yields a higher false alarm rate than our approach.

9.2. Lab Experiments

The objective of lab experiments is to evaluate Samba under more dynamic environment. The experiments were conducted in a 15 feet $\times$ 10 feet water tank in our lab. We vertically place a white foam board along a side of the tank to produce an artificial shoreline. The settings of Samba are consistent with those in the field experiments

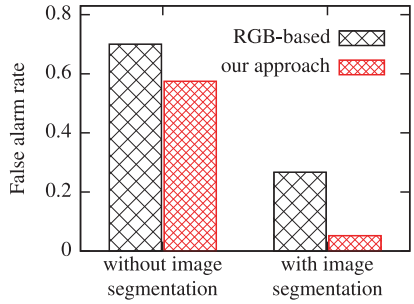


Fig. 14. Reduction in false alarm rate after image segmentation.

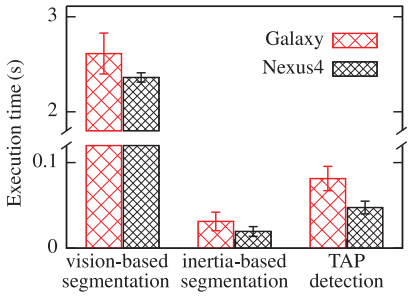


Fig. 15. Execution time on representative smartphone platforms.

unless otherwise stated. We test the performance of hybrid image segmentation and TAP detection algorithms under a wide range of environmental conditions and system settings such as training dataset size, wavy water, illuminance change, and color similarity between the TAP and water area.

9.2.1. Computation Overhead. We first evaluate the overhead of image segmentation and TAP detection algorithms, where the reduction in computation delay characterizes the improvement in energy efficiency on phone. In this set of experiments, we measure the computation delay of each module, that is, vision-based segmentation, inertia-based segmentation, and TAP detection, on Galaxy and Nexus4, respectively. The computation delay is measured as the elapsed time using Android API `System.nanoTime()`. The results are plotted in Figure 15. We can see that the vision-based segmentation incurs the longest delay, which is mainly due to the compute-intensive Hough transform. In contrast, the inertia-based segmentation is rather efficient, achieving 98% reduction in computation delay. Note that in the hybrid segmentation, Samba learns the mapping models every $N = 20$ frames. Thus, the measured overhead of inertia-based segmentation provides an overhead upper bound of the proposed hybrid approach. The TAP detection algorithms take about 80ms and 50ms on Galaxy and Nexus4, respectively. Therefore, our aquatic monitoring solution is efficient on mainstream smartphones and can well meet the real-time requirement. Compared with SOAR [Wang et al. 2014] which typically takes more than 3 seconds to process a frame, Samba reduces the computation delay by about 97%.

9.2.2. Impact of Image Scaling. Samba reduces the overhead of CV algorithms by leveraging the energy-efficient inertial sensing. There are other approaches to decreasing the computation overhead purely from a CV perspective, for example, image scaling. Image downsampling can reduce the number of pixels to be processed. However, sensing performance will also decrease due to lost information during the downsampling process. In this set of experiments, we evaluate the trade-off between the TAP detection rate and the image processing delay under different downsampling settings. Specifically, we proportionally down scale the height and width of a captured frame and define the *downsampling rate* as the ratio of the remaining pixels. Figure 16 shows the detection rate and the execution time of the TAP detection algorithm under different settings of the downsampling rate. We can see that, with downsampled frames, the detection rate decreases. This is because the scaled image loses information such as small TAP patches. On the other hand, downsampling reduces the processing delay. Note that other parameters of visual sensing, e.g., frame rate, can be tuned to reduce computation overhead while losing sensing granularity and accuracy.

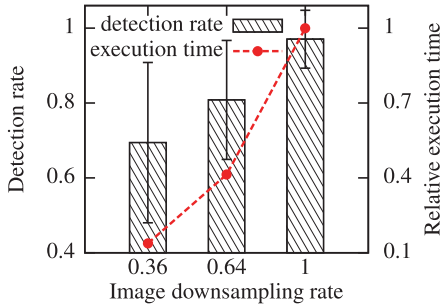


Fig. 16. Impact of image resolution on detection performance and computation delay.

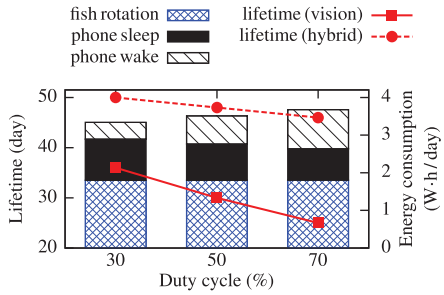


Fig. 17. Projected lifetime and daily energy consumption breakdown.

Table I. Samba Power Consumption Profile

	Voltage (V)	Current (A)	Power (W)
Galaxy wake	3.7	0.439	1.624
Galaxy sleep	3.7	0.014	0.518
Servo motor	6	0.5	3

9.2.3. Projected Lifetime. We now evaluate the lifetime of Samba based on its power consumption profile as shown in Table I. Smartphone computation, standby, and fish rotation consume the highest powers. The energy drain on phone can be calculated using offline measurements and duty cycle. Specifically, we measure the power consumption of Galaxy using an Agilent 34411A multimeter when the phone is executing the TAP detection algorithms with vision-based and hybrid segmentations. According to the integrated evaluation, a monitoring round lasts 5 minutes, on average, and Samba can rotate to the scheduled orientation within 15 seconds. We can thus upper-bound the daily consumed energy for fish rotation by $(12 \times 60/5) \times (15/3{,}600) \times p_r$ Wh, where p_r is the motor power consumption for beating the tail. Figure 17 shows the projected lifetime of Samba when running the vision-based and hybrid segmentations, respectively. We can see that the system lifetime is almost doubled by switching vision-based segmentation to hybrid approach. Figure 17 also evaluates the impact of duty cycle, which is defined as the ratio of wake time to the total time. As expected, the system lifetime decreases with duty cycle. In our current prototype, Samba has a total of 170 Wh battery capacity, including a backup 13.5 Wh and two main 75 Wh batteries on the fish, and a 6.48 Wh battery on the phone. Even with half of the battery capacity, Samba can achieve a lifetime of nearly a month with hybrid segmentation and 30% duty cycle. Moreover, the breakdown of daily energy consumption is plotted in Figure 17. We find that a majority of energy is actually consumed by the sleep periods and fish rotation. Owing to the high efficiency of hybrid image segmentation and TAP detection algorithms, the wake periods consume the least amount of energy. Therefore, the lifetime of Samba can be further extended if the phone is powered off during the sleep periods.

9.2.4. Image Segmentation Accuracy. We then evaluate the accuracy of hybrid image segmentation. To create dynamics, we generate waves by connecting a feed pump to the tank. As a result, the shoreline slope (i.e., k_i) varies up to 20°, and the average vertical coordinate (i.e., h_i) varies up to 170 pixels. We define the estimation errors for visual features as $|k_i - \tilde{k}_i|$ and $|h_i - \tilde{h}_i|$, where $\tilde{k}_i$ and $\tilde{h}_i$ are the estimated visual features by the inertia-based approach, and k_i and h_i are the ground truth measured by the

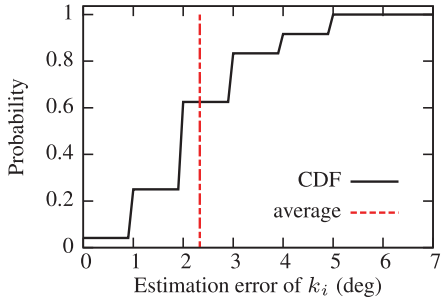


Fig. 18. CDF and average of estimation error for shoreline slope k_i.

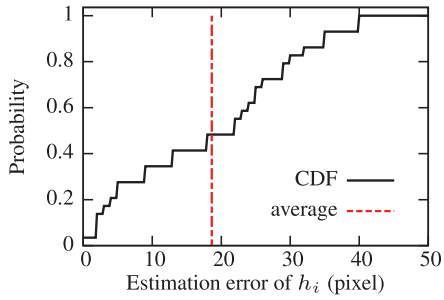


Fig. 19. CDF and average of estimation error for shoreline position h_i.

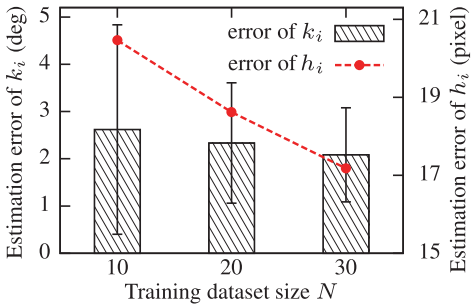


Fig. 20. Impact of training dataset size N on estimation accuracy.

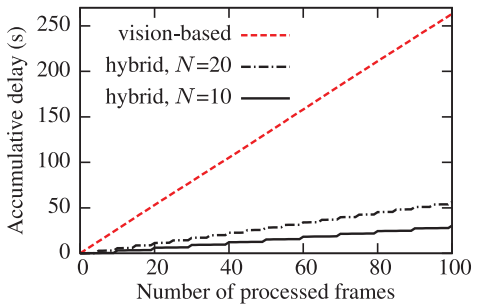


Fig. 21. Impact of training dataset size N on computation delay.

camera. Figure 18 plots the CDF and average of the estimation errors for k_i. We can see that the inertia-based segmentation can accurately estimate k_i, with an average estimation error of about 2.3°. Moreover, Figure 19 plots the CDF of the estimation errors for h_i. As shown in this figure, the average estimation error is around 18 pixels. This set of experiments validate that the proposed hybrid image segmentation can achieve satisfactory performance under wavy water environment.

9.2.5. Impact of Training Dataset Size. As discussed in Section 4.4, there is a trade-off between the estimation accuracy and system overhead caused by the training dataset size N. In this set of experiments, we evaluate the impact of N. Figure 20 plots the estimation errors for k_i and h_i under various settings of N. It can be observed that the average estimation accuracy of both k_i and h_i increases with N. In Samba, the setting of N determines the size of the buffer to store the training data for mapping models learning. Therefore, a larger N increases the diversity of training data, resulting in mapping models that account for a wider range of dynamics. Moreover, we find that the variance of estimation errors decreases with N. Meanwhile, the setting of N contributes to the system overhead. Figure 21 shows that the computation delay of hybrid segmentation increases with N. This is mainly because the visual features are extracted more frequently under a larger N. The computation overhead of mapping models learning also increases with N. According to our measurements, such delay is in the order of millisecond and thus has limited impact on computation delay. In practice, system designers can conduct similar experiments to choose a satisfactory setting of N. Figure 21 also compares the accumulative computation delay of

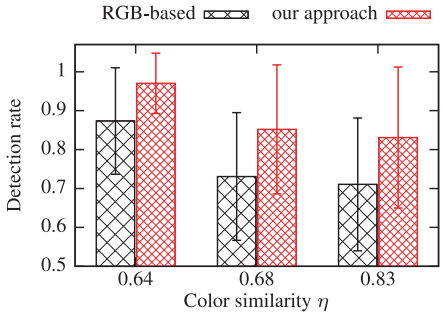


Fig. 22. Impact of color similarity η on detection performance.

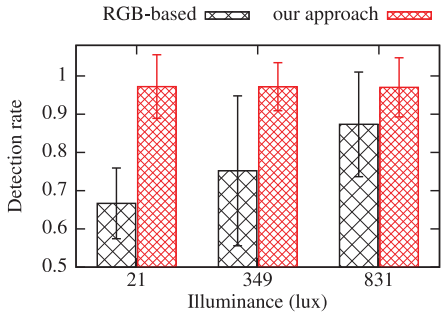


Fig. 23. Impact of illuminance on detection performance.

vision-based and hybrid approaches. We can see that the delay is notably shortened with the hybrid image segmentation.

9.2.6. Impact of Color Similarity. Samba identifies the TAP from the water area by distinguishing their color features (see Section 5.1). This set of experiments evaluate how the color similarity η between the two areas affects the TAP detection performance. We adopt three boards in different colors as the TAP patches. For each colored patch, we conduct 10 runs of experiments. For each run, we apply a threshold of 0.8 for the positive overlap ratio to determine the detection rate. Figure 22 shows the detection rate under various η, where the error bar represents standard deviation. We can see that the detection rate decreases with η. As η quantifies the proximity of two colors in the cylindrical HSV model, the increase in η reduces the likelihood of identifying the TAP patches from water. We can also find that our TAP detection algorithms achieve satisfactory detection performance under high color similarity. For example, when $\eta = 0.83$, the detection rate is about 84%. Moreover, our approach yields consistently better detection performance than the RGB-based baseline approach becaudse the HSV model differentiates similar colors more effectively.

9.2.7. Impact of Illumination. As discussed in Section 5.1, we adopt two designs to enhance Samba's robustness to illuminance change. First, the color histogram is built in the HSV model and excludes the value channel that is sensitive to illumination condition. Second, we normalize the color histogram before applying it to back projection. In the experiments, we create various illumination conditions by using different combinations of three compact fluorescent lamps and a Power Tech LED light [Power Tech LED 2014]. We use the patch with $\eta = 0.64$ and conduct 10 runs of experiments for each illumination condition. Figure 23 plots the detection rate under various illumination conditions, where the reported illuminance is measured by the phone's built-in light sensor. We can see that our TAP detection algorithms maintain consistently satisfactory performance under different illumination levels, while the RGB-based baseline approach is sensitive to illumination change.

9.2.8. Integrated Evaluation. In this set of experiments, all modules of Samba, that is, hybrid image segmentation, TAP detection, and adaptive rotation scheduling, are integrated and evaluated. Moreover, on the control board of robotic fish, we implement a closed-loop proportional-integral-derivative (PID) controller that adaptively controls the tail beats based on the discrepancy between the scheduled and actual orientations. We imitate TAP development by gradually uncovering the board's surface (with $\eta = 0.64$). Based on the angular view of Galaxy, we select the camera orientations as $\{0, \pi/4, \pi/2, 3\pi/4, \pi\}$ with respect to the tank's side to ensure that the semi-circle

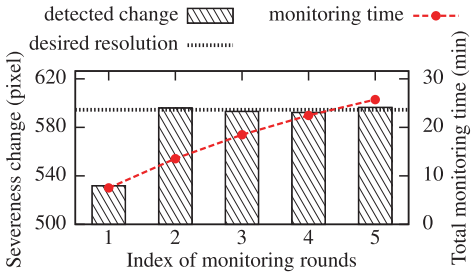


Fig. 24. Detected severeness change and total monitoring time versus index of rotation.

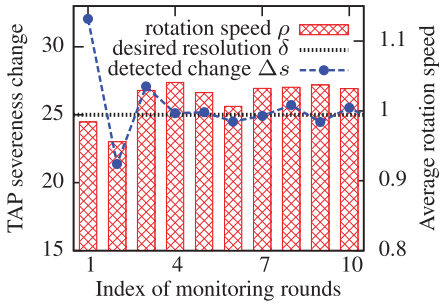


Fig. 25. Detected severeness change and Samba rotation speed versus index of rotation.

can be fully covered when Samba rotates over these orientations. At $t = 0$, Samba is deployed in parallel to the tank's side and starts the aquatic monitoring. We set the initial average rotation speed as 6°/min. Therefore, the first monitoring round has an interval of 7.5 minutes. After the first round, Samba adaptively schedules its rotation speed to maintain the desired severeness change, which is set to be 595 pixels, until it is parallel to the tank's side again. Throughout the experiments, Samba achieves a detection rate of around 97% and a false alarm rate of about 5%. Figure 24 plots the detected severeness changes in the first five rounds, which are well maintained at the desired level. Moreover, Figure 24 shows the total monitoring time versus index of robot rotation. We can see that the monitoring time varies across rounds, and it has a 5-minute length, on average. During the experiment, we also find that our PID controller can generally direct Samba to the desired camera orientation, with an error lower than 7°.

9.3. Trace-Driven Simulations

We evaluate the adaptive rotation control of Samba through trace-driven simulations, given the difficulty in accessing an actively developing TAP. To improve realism, we collect data traces, including Samba rotation errors and real chemical diffusion process, and use them in our simulations. First, the error traces of rotation are collected using our prototype in the water tank. We measure the rotation error by the discrepancy between the desired orientation and actual heading direction of Samba. Second, we record the diffusion traces of Rhodamine-B, which is a solution featuring red color, in saline water using a camera. Hence, the development of diffusion process is characterized by the expansion of the red area over time. The diffusion traces include the detected Rhodamine-B area and corresponding timestamp.

In the simulations, Samba monitors TAP within its circular surveillance region. According to our measurements, the camera on Galaxy has an angular view of $5\pi/18$. Hence, we partition the surveillance region into 8 sectors such that a full coverage can be achieved by a complete scan. For each rotation, the actual direction is set based on the collected error traces and thus is subjected to errors. For each orientation, the monitoring interval is determined by the scheduled rotation speed. We use the collected diffusion traces as severeness measurements, based on which the robot estimates the dynamics of TAP and schedules the rotation speed. Other settings include the desired resolution $\delta = 25$ and controller coefficient $c = 0.85$.

Figure 25 plots the detected severeness change Δs in the first 10 rounds. We can see that Δs quickly converges to the desired severeness resolution δ. Figure 25 also shows the scheduled rotation speed, which is scheduled based on the current dynamics of

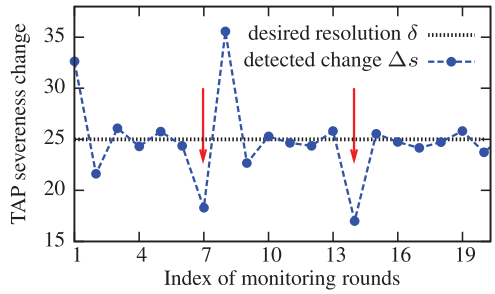


Fig. 26. Impulsive and step responses (at the 7^{th} and 14^{th} rounds) of our control algorithm.

TAP and δ. We further evaluate the response of our algorithm to the sudden changes in the TAP development. Specifically, we artificially reduce the severeness measurements by 30% at the 7^{th} rotation (i.e., the left arrow in Figure 26) and continuously since the 14^{th} rotation (i.e., the right arrow in Figure 26). For both types of changes, our algorithm converges within a few rounds of rotation. Therefore, the proposed algorithm can effectively adapt the rotation of Samba to the developing dynamics of TAP.

10. CONCLUSION AND FUTURE WORK

This article presents Samba—a novel aquatic robot designed for monitoring harmful aquatic processes such as oil spills and HABs. Samba integrates an off-the-shelf Android smartphone and a robotic fish. Samba features hybrid image segmentation, TAP detection, and adaptive rotation control algorithms. The hybrid image segmentation effectively reduces the false TAP detections by excluding the nonwater area in TAP detection. It also significantly decreases the overhead of continuous visual sensing by integrating inertial sensing via a learning-based approach. The TAP detection algorithms are lightweight and robust to various environment and system dynamics such as illuminance change and the camera's internal noises. The adaptive rotation control enables Samba to achieve full circular coverage and fine temporal granularity in monitoring the spatiotemporally developing TAP by adjusting the rotation speed. Field experiments, lab experiments, and trace-driven simulations show that Samba can achieve reliable and real-time TAP detection with a system lifetime up to nearly 2 months. In our future work, we plan to deploy Samba to more lakes/ponds to evaluate its effectiveness under diverse environments.

REFERENCES

James Ammerman and William Glover. 2000. Continuous underway measurement of microbial ectoenzyme activities in aquatic ecosystems. *Marine Ecology Progress Series* 201 (2000), 1–12.

Android 2.2 Platform Highlights. 2010. Homepage. Available at http://developer.android.com/guide/topics/sensors/sensors_position.html.

Android Runtime. 2014. Homepage. Available at http://source.android.com/devices/tech/dalvik/art.html.

Jeffrey Blum, Daniel Greencorn, and Jeremy Cooperstock. 2013. Smartphone sensor reliability for augmented reality applications. In *Proceedings of the 10th International Conference on Mobile and Ubiquitous Systems*. 127–138.

Bryan Boyd. 2003. *Introduction to Algae*. Englewood Cliffs.

Camilla Brekke and Anne Solberg. 2005. Oil spill detection by satellite remote sensing. *Remote Sensing of Environment* 95, 1 (2005), 1–13.

Dongyao Chen, Kyong-Tak Cho, Sihui Han, Zhizhuo Jin, and Kang Shin. 2015. Invisible sensing of vehicle steering with smartphones. In *Proceedings of the 13th International Conference on Mobile Systems, Applications, and Services (MobiSys)*. 1–13.

John Crank. 1979. *The Mathematics of Diffusion*. Oxford University Press.

Matthew Dunbabin and Alistair Grinham. 2010. Experimental evaluation of an autonomous surface vehicle for water quality and greenhouse gas emission monitoring. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 5268–5274.

Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. 2014. Scalable object detection using deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2155–2162.

Matthew Faulkner, Michael Olson, Rishi Chandy, Jonathan Krause, K. Mani Chandy, and Andreas Krause. 2011. The next big one: Detecting earthquakes and other rare events from community-based sensors. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 13–24.

Gliding Robotic Fish. 2013. Homepage. Available at http://nbcnews.to/1fGAomj.

HABs and Lake Mendota. 2007. Homepage. Available at http://blooms.uwcfl.org/mendota.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 9 (2015), 1904–1916.

Robert Hecky and Peter Kilham. 1988. Nutrient limitation of phytoplankton in freshwater and marine environments: A review of recent evidence on the effects of enrichment. *Limnology and Oceanography* 33, 4 (1988), 796–822.

Samuli Hemminki, Petteri Nurmi, and Sasu Tarkoma. 2013. Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

Chuanmin Hu, Frank Müller-Karger, Charles Taylor, Douglas Myhre, Brock Murch, Ana Odriozola, and Gonzalo Godoy. 2003. MODIS detects oil spills in Lake Maracaibo, Venezuela. *Eos, Transactions, American Geophysical Union* 84, 33 (2003), 313–319.

Puneet Jain, Justin Manweiler, Arup Achary, and Kirk Beaty. 2013. FOCUS: Clustering crowdsourced videos by line-of-sight. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

Shinichiro Kako, Atsuhiko Isobe, and Shinya Magome. 2012. Low altitude remote-sensing method to monitor marine and beach litter of various colors using a balloon equipped with a digital camera. *Marine Pollution Bulletin* 64, 6 (2012), 1156–1162.

Purushottam Kulkarni, Deepak Ganesan, Prashant Shenoy, and Qifeng Lu. 2005. SensEye: A multi-tier camera sensor network. In *Proceedings of the 13th ACM International Conference on Multimedia (MM)*. 229–238.

Gilles LaRoche, Ronald Eisler, and Clarence Tarzwell. 1970. Bioassay procedures for oil and oil dispersant toxicity evaluation. *Water Pollution Control Federation* (1970), 1982–1989.

Dimitrios Lymberopoulos and Andreas Savvides. 2005. XYZ: A motion-enabled, power aware sensor node platform for distributed sensor network applications. In *Proceedings of the 4th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 449–454.

Raphael Maree, Pierre Geurts, Justus Piater, and Louis Wehenkel. 2005. Random subwindows for robust image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 34–40.

Tood Moon. 1996. The expectation-maximization algorithm. *IEEE Signal Processing Magazine* 13, 6 (1996), 47–60.

National Ocean Service. 2014. Homepage. Available at http://oceanservice.noaa.gov.

Katsuhiko Ogata. 1995. *Discrete-Time Control Systems*. Prentice Hall.

Ohio Environmental Protection Agency. 2010. Homepage. Available at http://epa.ohio.gov.

Oil Spills and Disasters. 2010. Available at http://infoplease.com/ipa/A0001451.html.

Trevor Platt, Fuentes-Yaco Csar, and Frank Kenneth. 2003. Marine ecology: Spring algal bloom and larval fish survival. *Nature* 423, 6938 (2003), 398–399.

Power Tech LED. 2014. Homepage. Available at http://swflashlights.com.

Daniel Rudnick, Russ Davis, Charles Eriksen, David Fratantoni, and Mary Perry. 2004. Underwater gliders for ocean research. *Journal of the Marine Technology Society* 38, 2 (2004), 73–84.

Linda Shapiro and George Stockman. 2001. *Computer Vision*. Prentice Hall.

Yiran Shen, Wen Hu, Mingrui Yang, Bo Wei, Simon Lucey, and Chun-Tung Chou. 2014. Face recognition on smartphones via optimised sparse representation classification. In *Proceedings of the 13th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 237–248.

John Smith and Shih-Fu Chang. 1997. VisualSEEk: A fully automated content-based image query system. In *Proceedings of the 5th ACM International Conference on Multimedia (MM)*. 87–98.

Southern Regional Water Program. 2011. Homepage. Available at http://srwqis.tamu.edu.

Stargate User's Manual. 2014. Homepage. Available at http://xbow.com.

Thiago Teixeira, Eugenio Culurciello, Joon Park, Dimitrios Lymberopoulos, Andrew Barton-Sweeney, and Andreas Savvides. 2006. Address-event imagers for sensor networks: Evaluation and modeling. In *Proceedings of the 5th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 458–466.

Yu Wang, Rui Tan, Guoliang Xing, Jianxun Wang, Xiaobo Tan, Xiaoming Liu, and Xiangmao Chang. 2014. Aquatic debris monitoring using smartphone-based robotic sensors. In *Proceedings of the 13th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 13–24.

Yu Wang, Rui Tan, Guoliang Xing, Jianxun Wang, Xiaobo Tan, Xiaoming Liu, and Xiangmao Chang. 2016. Monitoring aquatic debris using smartphone-based robots. *IEEE Transactions on Mobile Computing* 15, 6 (2016), 1412–1426.

Yan Wang, Jie Yang, Hongbo Liu, Yingying Chen, Marco Gruteser, and Richard Martin. 2013. Sensing vehicle dynamics for determining driver phone use. In *Proceedings of the 11th International Conference on Mobile Systems, Applications, and Services (MobiSys)*. 41–54.

Received August 2015; revised March 2016; accepted April 2016