Samba: A Smartphone-Based Robot System for Energy-Efficient Aquatic Environment Monitoring

Yu Wang[1], Rui Tan[2], Guoliang Xing[1], Jianxun Wang[3], Xiaobo Tan[3], and Xiaoming Liu[1]
[1]Department of Computer Science and Engineering, Michigan State University, USA
[2]Advanced Digital Sciences Center, Illinois at Singapore
[3]Department of Electrical and Computer Engineering, Michigan State University, USA

ABSTRACT

Monitoring aquatic environment is of great interest to the ecosystem, marine life, and human health. This paper 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 spill and harmful algal blooms. Using the built-in camera of on-board smartphone, Samba can detect spatially dispersed aquatic processes in dynamic and complex environments. To reduce the excessive false alarms caused by the non-water 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 the continuous image segmentation. We propose a novel approach that leverages the power-efficient inertial sensors on smartphone to assist the 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 the image segmentation, significantly reducing 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. Lastly, Samba employs a feedback-based rotation control algorithm to adapt to spatiotemporal evolution 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 94% detection rate, 5% false alarm rate, and a lifetime up to nearly two 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*

IPSN'15, April 14 – 16, 2015, Seattle, WA, USA.

http://dx.doi.org/10.1145/2737095.2737100.

Keywords

Robotic sensor, smartphone, computer vision, object detection

1. INTRODUCTION

Aquatic environment is facing increasing threats from climate change, industrial pollution, and improper waste disposal. The last four decades have witnessed more than a dozen major oil spills with each releasing over 30 million gallons of oil [1]. Harmful algal blooms (HABs) have been observed in more locations than ever before [2]. Fig. 1(a) shows a recent proliferation of HABs in the Gulf of Mexico [3]. 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 [4]. It is thus imperative to detect these harmful aquatic processes, monitor their evolution, 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 [7]. However, this method is costly and can only cover a limited period of time. More advanced methods employ remote sensing technologies, e.g., balloon-board [21], aircraft [19], and satellite imaging [10]. The balloon-board monitoring is effective only for one-off and short-term surveillance of highly concentrated aquatic processes that have already been detected. The monitoring approaches based on aircraft and satellite imaging often incur high operational overhead and cannot achieve fine monitoring resolution. Recently, autonomous underwater vehicles (AUVs) [28] have been used for various underwater sensing tasks. However, the manufacturing costs of AUV platforms are often high (over $ 50,000 per unit [28]). In summary, these limitations make remote sensing and AUV-based approaches ill-suited for monitoring spatially scattered and temporally evolving aquatic processes.

This paper 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. Fig. 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 [15] is a low-cost (about $ 3,000 per unit) aquatic mobile platform

(a) (b) (c)

Figure 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 [3]); (b) A Samba prototype integrating a Samsung Galaxy Nexus phone with a robotic fish; (c) A sample image captured by the Samba prototype in an inland lake, where the water and non-water areas are separated by a shoreline; the trees exhibit similar color with the algal patches in the water area.

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., [12]) due to the enhanced stability from its bionic design. The on-board cameras of smartphone provide an inexpensive yet high-resolution sensing modality for detecting the harmful aquatic processes. For example, HABs, as shown in Fig. 1(a), can be detected using the phone's built-in cameras based on their distinctive colors. Moreover, in addition to camera, a few other sensors such as accelerometer and gyroscope, which are commonly available on smartphone, can assist the navigation and control of robotic fish. Second, compared with traditional chemical sensors that measure only one location at a time, camera has a wider sensing range and provides richer information 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. Lastly, the price of smartphone has been dropping drastically in the past few years. 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 regions and spatiotemporally evolving [10, 27], which create challenges for fine-grained monitoring. Continuous visual sensing can track their evolution, which, however, imposes significant energy and computation overhead to smartphone. Second, phone's 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. Lastly, existing vision-based detection algorithms using background subtraction [29] 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 differentiate between the foreground and the real background. Moreover, the target detection may be greatly affected by various dynamics such as blurred images caused by waves, highly variable illuminance, and complex non-water area in the image. These uncertainties can lead to excessive false alarms. Fig. 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 paper, we make the following contributions:

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 non-water 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 Fig. 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 environments. 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 the highly variable illuminance and camera noise.
3) We design a feedback-based robot rotation control algorithm that increases coverage and maintains a desired level of monitoring resolution on the evolving 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 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 two months.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 provides an overview of Samba. Sections 4 – 6 present the hybrid image segmentation, aquatic process detection, and adaptive rotation control algorithms, respectively. Section 7 describes the prototype implementation. Section 8 evaluates Samba. Section 9 concludes this paper.

2. RELATED WORK

Current approaches to monitoring harmful aquatic processes fall into four categories, i.e., manual spotting, patrol-boat-assisted survey [7], remote sensing (e.g., balloon-board [21], aircraft [19], and satellite imaging [10]), and AUV-based autonomous sensing [28]. 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 regions. 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 [22] incorporates a Cyclops camera into a Stargate platform [32] to detect objects at a 128×128 resolution. In [33], a camera module is installed on an XYZ node [24] 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 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 smartphone [30]. Different from existing vision-based systems, Samba has to address several unique challenges in aquatic monitoring, such as the highly variable illuminance and dynamic aquatic process evolution. Moreover, we need to make the image processing pipeline of Samba highly energy-efficient to enable long-term autonomous monitoring.

In our previous work [35], 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 much 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 spill and HABs; however, SOAR detects moving debris objects only while treating all the static targets as background. Moreover, SOAR identifies debris objects using a pixel-based 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, as it does not rely on robot mobility to maintain the pixel-level correspondence across frames in dynamic aquatic environment. Lastly, 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 the arrival model of the debris objects.

Inertial sensing has recently been explored in various mobile sensing applications. In [18], a set of inertial features are extracted from a phone's built-in accelerometer and used to identify user's transportation mode. In [36], accelerometer and gyroscope readings are employed to model vehicle dynamics and estimate the device's position inside the vehicle. In [14], smartphones collaboratively detect an earthquake using onboard accelerometer. The FOCUS system [20] detects shaken and blurred views during video recording based on measured acceleration. Different from the above 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. Traditional approach usually adopts background subtraction, which constructs a background model for each pixel according to historical observations [29]. However, background subtraction cannot be readily applied to monitoring aquatic processes such as oil spill and HABs. This is because the camera may have a view dominated by the target process when deployed in the affected region, making it impossible to build the background model [29]. The state-of-the-art approach [13, 16] detects targets of interest without explicitly segmenting the background. However, it usually requires significant amount of training data and incurs 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, which leads to affordable computation overhead for smartphone.

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 propels Samba by beating its tail, and communicates with the phone via a USB cable. Samba is designed to operate on water surface and monitor harmful aquatic processes such as oil spill and HABs. These processes typically disperse as patches in the aquatic environment and exhibit distinctive colors [10, 27]. To monitor a large affected region, multiple Samba robots 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 paper, we focus on the design of a single Samba robot. Due to the limited angular view of the phone's cameras, 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 to Samba will complicate the system design by introducing additional computation overhead due to the distorted images [29]. We leverage mobility to increase the sensing coverage. Specifically, Samba can rotate by performing discrete tail beats. We focus on the rotation mobility, because it 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 coverage 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 exe-

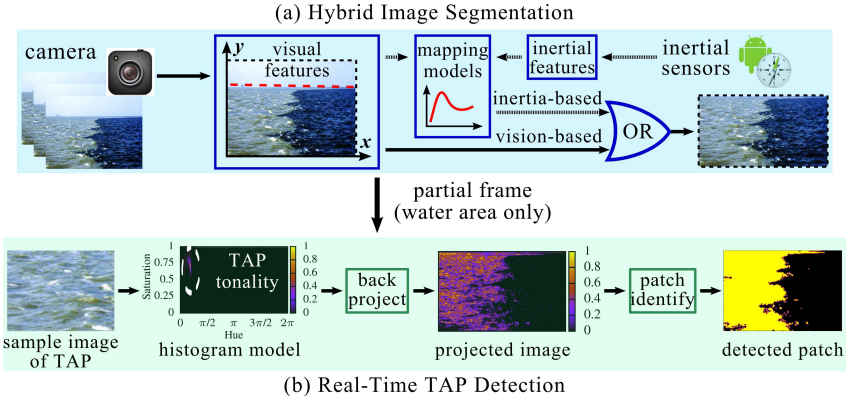


Figure 2: The aquatic environment monitoring pipeline when Samba keeps an orientation. Samba periodically adjusts its orientation to adapt to the dynamics of surrounding target aquatic process (TAP) patches.

cutes 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., evolution rate) and computes the monitoring interval for the next round. For example, if the TAP evolves rapidly, Samba will shorten the monitoring interval such that it can rotate back to the current orientation sooner, keeping smooth track of the TAP evolution; and vice versa. When drastic evolution of the TAP is detected, Samba can alert the user by using the communication interface (cellular/WiFi) on smartphone. A primary design objective of Samba is to realize long-term (up to a few months) autonomous monitoring. Between two image captures, Samba sleeps to reduce energy consumption. One 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 using compute-intensive algorithms. Specifically, Samba comprises the following three major components.

Hybrid image segmentation: By partitioning an image into water and non-water areas, Samba performs aquatic process detection in the water area only, thus reducing detection false alarms and computation energy consumption. Samba adopts a novel hybrid image segmentation framework as illustrated in Fig. 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 mapped from the inertial sensor readings, avoiding executing the compute-intensive CV algorithms continuously. Samba switches to vision-based segmentation if the mapping models need to be updated, e.g., 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 Fig. 2(b), it consists of two lightweight image processing modules, i.e., *back projection* and *patch identification*. First, Samba extracts the 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, which effectively deal with various environment and system dynamics such as the highly variable illuminance and camera's internal noise.

Adaptive rotation control: Samba monitors the surrounding TAP patches at fine spatiotemporal granularity while meeting energy consumption constraints. To adapt to the dynamics of TAP that is primarily affected by environmental conditions, we develop a feedback control algorithm to maintain the desired monitoring 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 monitoring 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 [29]. In aquatic monitoring, we adopt image segmentation to remove the non-water 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 non-water area (e.g., those caused by trees in Fig. 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 8.2.1), the Hough transform [29] 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 paper, we develop a novel hybrid image segmentation approach that utilizes both camera and inertial sensors on the phone, as illustrated in Fig. 3. Inertial sensors provide 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

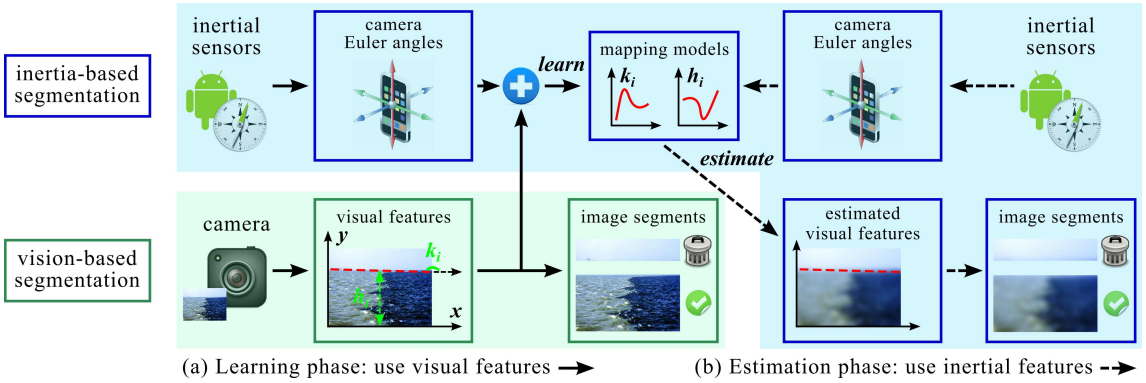


Figure 3: The overall structure of hybrid image segmentation where Samba switches between the vision-based and inertia-based segmentation modes. In the online learning phase, Samba jointly uses inertial and visual sensing to learn regression-based mapping models, which project the camera Euler angles (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.

the computation delay of inertia-based segmentation is only 2% of that of vision-based segmentation (see Section 8.2.1). 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 Fig. 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 learned mapping models estimate the visual features based on the inertial sensor readings and guide the image segmentation, as shown in Fig. 3(b). Therefore, the compute-intensive visual feature extraction CV algorithms are 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, which relies on 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 smartphone[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 of orientation measurements computed based on inertial sensor readings by a simple algorithm can be up to 30° [8]. According to our experiments, the inaccuracy in orientation measurements mainly results from bias. To deal with the bias and random noise, we smooth and calibrate the sensor readings. Fig. 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.

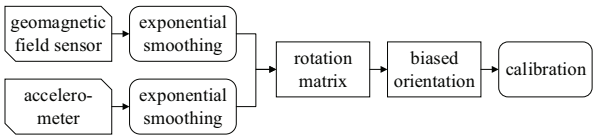


Figure 4: Workflow of virtual orientation sensor.

4.3 Feature Extraction

In order to establish the mapping models, Samba needs to extract features from both camera observations and inertial sensor readings for the same frame. In this paper, we focus on the lake environment where the shoreline can 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 Fig. 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 frame i, let k_i and h_i denote the shoreline slope and average vertical coordinate in the image plane, as illustrated in Fig. 3(a). The phone can extract the shoreline using Hough transform [29] 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 features* of frame i.

The camera's orientation, including yaw α_i, pitch β_i, and roll γ_i, is measured by the virtual orientation sensor. At runtime, we synchronize the orientation measurements with visual features obtained from the camera using the timestamps. By corresponding the world coordinate system with the 2D image plane, we can relate the camera's orientation

[1]The orientation sensor API has been deprecated in Android since version 2.2 [5].

$(\alpha_i, \beta_i, \gamma_i)$ to the visual features (h_i, k_i). Due to space limitation, the details of this coordinate mapping process is omitted here and can be found in [34]. We then 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, i.e., k_i, can be computed as

$$k_i = \frac{y_1''-y_2''}{x_1''-x_2''} = \omega_1 \cdot \frac{(x_1 f_{i,3}-y_1 f_{i,4})-(x_2 f_{i,3}-y_2 f_{i,4})}{(x_1 f_{i,1}-y_1 f_{i,2})-(x_2 f_{i,1}-y_2 f_{i,2})} = \omega_1 \cdot \frac{f_{i,3}-\omega_2 f_{i,4}}{f_{i,1}-\omega_2 f_{i,2}}, \quad (1)$$

where ω_1 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

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

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 Eq. (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). Due to space limitation, we omit the details here. 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 can learn the mapping models that project the latter to the former. Based on Eq. (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), i.e., $\mathbf{K} = [k_1, \ldots, k_N]^\intercal$. 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 Eq. (1).

5. AQUATIC PROCESS DETECTION

The real-time TAP detection pipeline of Samba is illustrated in Fig. 2(b). Although our approach is based on elementary CV algorithms, it is non-trivial to customize them for detecting TAP and efficiently implement on 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 the highly variable illuminance than the RGB color space [25]. 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 [29]. In this paper, 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 15 square kilometers water area [6]. Therefore, the robot may have a camera view dominated by the TAP patches when deployed in an affected region. 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 uncalm waters. Our proposed approach is motivated by the fact that a TAP usually maintains featured and nearly constant colors. For example, oil slicks often appear to be brighter than the surrounding water surface [10], and the typical color of HABs is green or red [27]. 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 [25] 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 quantifies the probability that the corresponding color represents the TAP. Therefore, $\widetilde{\mathbf{M}}$ depicts the tonality of the TAP, which is used as its histogram model. Fig. 2(b) shows a sample image of HABs occurred in the Gulf of Mexico [3] 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 $\widetilde{\mathbf{M}}$, i.e., $\widetilde{\mathbf{M}}(s_{m,n}, h_{m,n})$. 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 larger probability that the corresponding pixel in I_t is the TAP. An example of the projected frame I'_1 is shown in Fig. 5(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 paper, we adopt a *color similarity* [31] 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 (h_j, s_j, v_j), the color similarity, denoted by η, is given by

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

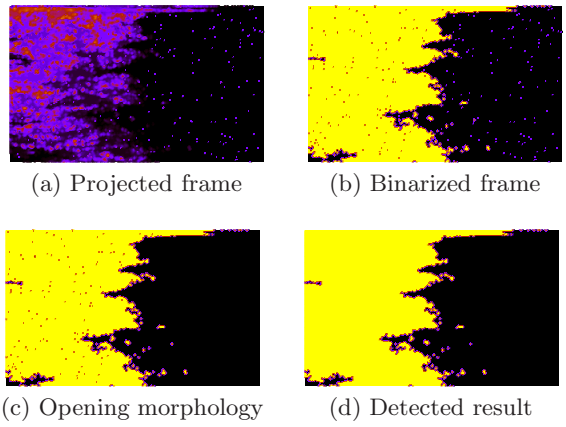
The η ranges from 0 to 1, where the lower bound 0 indicates the highest level of color contrast and the upper bound 1 represents the same color. Therefore, a larger value of η suggests a stronger color resemblance between the TAP candidate and water area, and hence the TAP candidate is less likely to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

5.2 Patch Identification

As illustrated in Fig. 5, patch identification works on the projected image and identifies the TAP patches. In the projected frame shown in Fig. 5(a), each pixel maintains a value within $[0, 1]$, which represents the probability that it belongs to the TAP. We adopt a threshold-based approach to removing the pixels with extremely low probabilities. Specifically, for any pixel with a value higher than a threshold, we consider this pixel as a candidate TAP pixel and round its value to 1; otherwise, it is classified as a pixel that represents the water area and set to 0. To determine this threshold, we try various settings and find that, with a setting of 0.1, the detected TAP boundary is most similar to visual observation. The binarized frame, as shown in Fig. 5(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 camera's internal noise. To remove these false alarms, we apply the opening morphology operation [29]. It consists of an erosion and a dilation, in which the former eliminates the noise pixels and the latter preserves the shape of true TAP area. Fig. 5(c) depicts the resulted frame after noise removal. We then perform region growing [29] 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. Fig. 5(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

To monitor the surrounding TAP patches at fine spatiotemporal granularity, Samba needs to periodically adjust its orientation. However, this is challenging because the evolution 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 [23], and the growth of HABs is sensitive to local nutrient availability [17]. To adapt to such dynamics, we propose a rotation control algorithm, which maintains the monitoring granularity (e.g., the area expansion of an algal patch between two consecutive observations during robot rotation) at the desired level by adjusting the monitoring interval in each round.



(a) Projected frame (b) Binarized frame

(c) Opening morphology (d) Detected result

Figure 5: Sample patch identification process.

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 [29]. FOV is typically modeled as a sector 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 region [10,27], 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. As the TAP evolves (e.g., diffusion of oil slick), it has 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 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 following the user-specified sleep/wake duty cycle. For each frame, the severeness of TAP can be measured by the total area of detected patches in the frame. The TAP and robot will drift at a similar speed and therefore remain in the same inertial system. Thus, the expansion of TAP patches characterizes the its dynamics. Using frames captured at different time instants, Samba can estimate the dynamics of TAP under current environmental conditions by computing the change in severeness.

To achieve fine spatiotemporal monitoring granularity 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 evolving dynamics. When the evolution 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 [11], i.e., $\Delta s = \varepsilon \cdot 2\pi/\rho$. 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. To save energy, Samba remains stationary as long as it can provide the required resolution. The setting of δ describes how smooth the user aims to keep track of the TAP. Fig. 6 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, 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 [26]. 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 [26], 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 Fig. 6. First, Samba has rotation errors. It may not head exactly toward the desired orientation due to complex fluid dynamics. Second, the detected severeness of TAP exhibits variance. We define the relative error for Δs as the absolute error to the ground truth of TAP area in the image. As the detection of severeness is based on CV algorithms that work in a pixel-wise manner, it can be affected by dynamics like camera noise. In light of these disturbances, we design the controller $G_c(z)$ to mitigate their impact. From control theory [26], 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 Fig. 6, 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 $\rho_k = 2\pi/\tau_k$.

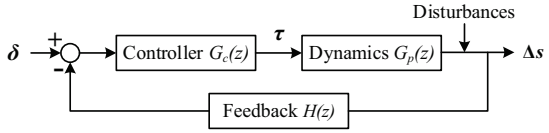


Figure 6: The closed loop for robot rotation control.

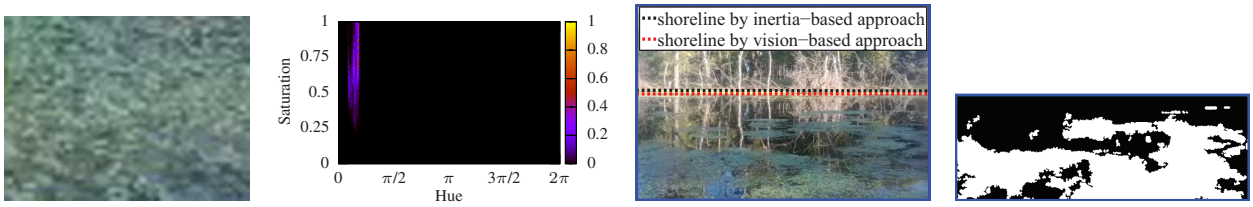
7. IMPLEMENTATION

We have built a proof-of-concept prototype of Samba for evaluation. The hybrid image segmentation, TAP detection, and adaptive rotation control algorithms presented in Sections 4 – 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 1.2 GHz dual-core processor and 1 GB memory, and Nexus4 is equipped with a 1.5 GHz quad-core processor and 2 GB memory. They are representative mid- and high-end Android smartphones. The app takes about 6.24 MB storage on the phone after installation, and requires about 10.8 MB RAM allocation while running on a frame resolution of 720× 480. To exploit the multi-core computation capability, the visual feature extraction, mapping models learning, and TAP detection are implemented in separate threads. In our current prototype, we use a host computer to relay the communication between the phone (via WiFi) and the fish (via ZigBee). In the future, we will establish direct connection between them. Fig. 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.

On initialization, Samba extracts the hue-saturation histogram of TAP based on 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. During the implementation, we find that the Hough transform, which is used to extract the shoreline, incurs excessive delay. To address this issue, we use OpenCV's implementation through Java Native Interface. We also examined the frame processing performance when the app uses the new runtime option ART, which is introduced in Android 4.4 and pre-complies the Java code into system-dependent binaries. According to our measurements, ART can reduce the system delay by about 20% over Dalvik.

8. 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



(a) Sample image (b) Histogram model (c) A typical frame (d) Detection result

Figure 7: Sample HABs detection in the field experiments.

fully integrated robot. 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 two months.

8.1 Field Experiments

To test the detection performance of Samba in real world, we deploy our prototype in an inland lake with an area of 200,000 square feet on September 22, 2014. Part of the lake is covered by patches of filamentous algae [9] that exhibit pale green color, as shown in Fig. 1(c). During the experiments, the average illuminance is around 1,082 lux. The impact of significant illuminance change is evaluated in Section 8.2.4. We set the frame resolution to be 720×480 and the frame rate to be 0.5 fps. 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 while disabling the robot rotation. 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 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 is executed offline using the captured frames. The purpose of our field experiments is three-fold. 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. Lastly, we validate that Samba can effectively reduce the false alarms caused by the non-water area through image segmentation.

8.1.1 Sample HABs Detection

Fig. 7 depicts a sample HABs detection in the field experiments. An image of the algal patch, as shown in Fig. 7(a), is used as the sample image for the detection pipeline. Fig. 7(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. Fig. 7(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 field experiments. In Section 8.2.3, we will evaluate the performance of hybrid image segmentation under more dynamic environments. On the segmented water area, Samba executes the TAP detection algorithms. Fig. 7(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.

8.1.2 Detection Performance

We now evaluate the detection performance of Samba quantitatively. For each frame, we define the *positive overlap ratio* as the ratio of the overlap area between detection and ground truth to the actual 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. Fig. 8 plots the detection rate versus positive overlap ratio for our approach (i.e., using hue-saturation histogram) and the RGB-based baseline, 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 rate than the baseline. 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 8.2.4, we will evaluate the impact of η on detection performance.

8.1.3 Effectiveness of Image Segmentation

In TAP detection, we define the *negative overlap ratio* as the area with false detections to the actual TAP area in each frame. 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 detections mainly result from the captured shore area. In particular, the trees on the shore, sharing a color similarity η with the target filamentous algae of up to 0.97, are the major contributor. Fig. 9 plots the false alarm rate versus negative overlap ratio for our approach and the RGB-based baseline, respectively. We find that our approach achieves consistently lower false alarm rate than the baseline, as it can characterize the TAP color features more effectively. We then validate the effectiveness of image segmentation by evaluating the reduction in false alarms. Image segmentation allows Samba to perform HABs detection in the water area only. Fig. 10 compares the false alarm rates for our approach and the RGB-based baseline, respectively, with and without image segmentation. The reported values are calculated by upper-bounding the negative overlap ratio at 0.5. We can see that by applying image segmentation, the false alarm

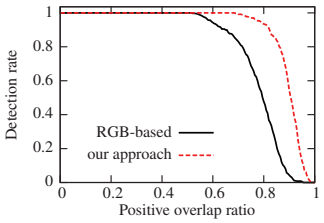


Figure 8: Detection performance under various positive overlap ratios.

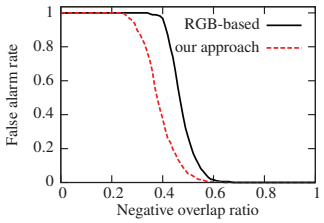


Figure 9: False alarm rate under various negative overlap ratios.

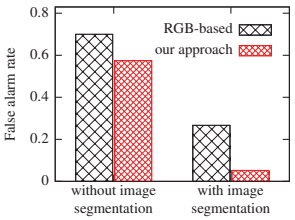


Figure 10: Reduction in false alarm rate after image segmentation.

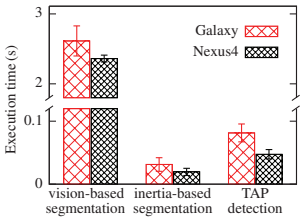


Figure 11: Execution time on representative smartphone platforms.

rate of our approach is reduced by over 90%. Moreover, the baseline also benefits from image segmentation, but still yields a higher false alarm rate than our approach.

8.2 Lab Experiments

The objective of lab experiments is to evaluate Samba under more dynamic environments. The experiments were conducted in a 15 feet × 10 feet water tank in our lab. We vertically place a white foam board along one side of the tank to produce an artificial shoreline. The settings of Samba are consistent with those in the field experiments, 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 wavy water and illuminance change.

8.2.1 Computation Overhead

We first evaluate the overhead of image segmentation and TAP detection algorithms on phone. Specifically, we measure the computation delay of each module, i.e., 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 Fig. 11. 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 drastically efficient, achieving over 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 80 and 50 milliseconds on Galaxy and Nexus4, respectively. Therefore, our aquatic monitoring solution is efficient on mainstream smartphones and can well meet the real-time requirement. Compare with SOAR [35] which typically takes more than 3 seconds to process a frame, Samba reduces the computation delay by about 97%.

8.2.2 Projected Lifetime

We now evaluate the lifetime of Samba based on its power consumption profile as shown in Table 1. 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

Table 1: 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

to the scheduled orientation within 15 seconds. We can thus upper-bound the daily consumed energy for fish rotation by $(12 \times 60/5) \times (15/3600) \times p_r$ W·h, where p_r is the motor power consumption for beating the tail. Fig. 12 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. Fig. 12 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 W·h battery capacity, including a backup 13.5 W·h and two main 75 W·h batteries on the fish, and a 6.48 W·h 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 Fig. 12. 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. The lifetime of Samba can be further extended if the phone is powered off during the sleep periods.

8.2.3 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 of visual features as $|k_i - \widetilde{k}_i|$ and $|h_i - \widetilde{h}_i|$, where $\widetilde{k}_i$ and $\widetilde{h}_i$ are the estimated visual features by the inertia-based approach, and k_i and h_i are the ground truth measured by the camera. Fig. 13 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, Fig. 14 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 hybrid image segmentation can achieve satisfactory performance under wavy water environment.

8.2.4 Impact of Illuminance

As discussed in Section 5.1, we adopt two designs to enhance Samba's robustness to illuminance change. First, the

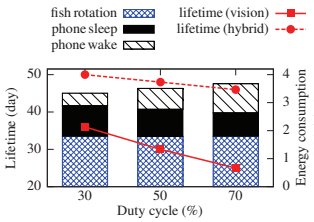


Figure 12: Projected lifetime and daily energy consumption breakdown.

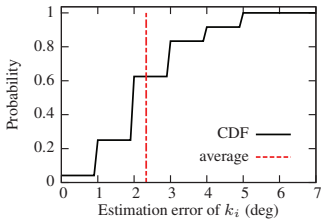


Figure 13: CDF and average of estimation error for shoreline slope k_i.

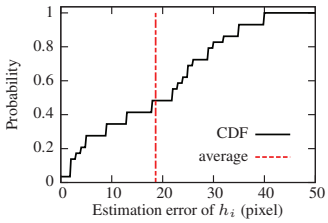


Figure 14: CDF and average of estimation error for shoreline position h_i.

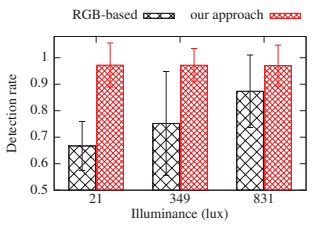


Figure 15: Impact of illuminance on detection performance.

color histogram is built in the HSV representation and excludes the value channel that is sensitive to lighting condition. Second, we normalize the color histogram before applying it to back projection. In the experiments, we create various lighting conditions by using different combinations of three compact fluorescent lamps and a Power Tech LED light. We use the patch with $\eta = 0.64$ and conduct 10 runs of experiments for each lighting condition. Fig. 15 plots the detection rate under various lighting 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 illuminance levels, while the RGB-based baseline is sensitive to illuminance change.

8.2.5 *Integrated Evaluation*

In this set of experiments, all modules of Samba, i.e., 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 the evolution of TAP 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 can be fully covered when Samba slides 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 deg/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 detected severeness change at the desired level, which is set to be 595 pixels, until it is parallel to the tank's side again. Throughout the experiments, Samba achieves the detection rate of around 97% and false alarm rate of about 5%. Fig. 16 plots the detected severeness changes in the first 5 rounds, which are well maintained at the desired level. Moreover, Fig. 16 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°.

8.3 Trace-driven Simulations

We evaluate the adaptive rotation control of Samba through trace-driven simulations, given the difficulty in accessing an actively evolving TAP. To simulate realistic settings, we collect Samba rotation errors and real chemical diffusion process. 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 evolution of diffusion process is characterized by the expansion of the red area over time. The traces contain 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 $c = 0.85$.

Fig. 17 plots the detected severeness change Δs in the first 10 rounds. We can see that Δs quickly converges to the desired severeness resolution δ. Fig. 17 also shows the scheduled rotation speed, which is scheduled based on the current dynamics of TAP and δ. We further evaluate the response of our algorithm to the sudden changes in TAP evolution. Specifically, we artificially reduce the severeness measurements by 30% at the 7^{th} rotation (i.e., the left arrow in Fig. 18) and continuously since the 14^{th} rotation (i.e., the right arrow in Fig. 18). 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 evolving dynamics of TAP.

9. CONCLUSION AND FUTURE WORK

This paper presents Samba – a novel aquatic robot designed for monitoring harmful aquatic processes such as oil spill 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. 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 two months.

In our future work, we will study several issues such as the impact of the number of sample images, robust thresholding in patch identification, and adaptive image scaling. Moreover, we plan to exploit the use of both front- and rear-facing

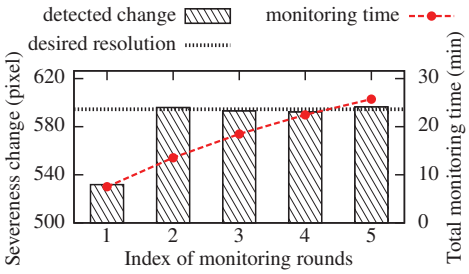


Figure 16: Detected severeness change and total monitoring time versus index of rotation.

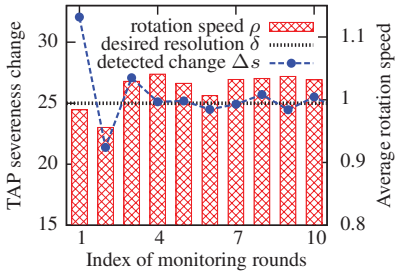


Figure 17: Detected severeness change and Samba rotation speed versus index of rotation.

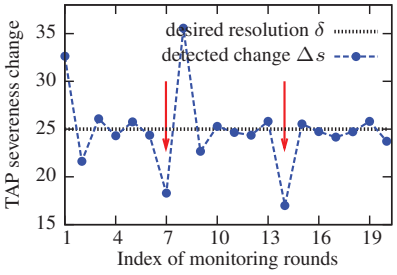


Figure 18: Impulsive and step responses (at the 7^{th} and 14^{th} rounds) of our approach.

cameras on smartphone to increase the surveillance coverage and reduce the system energy consumption due to rotation.

10. ACKNOWLEDGMENTS

This work was supported in part by the U.S. National Science Foundation under grants CCF-1331852, CNS-1218475, and EECS-1446793. We thank our shepherd Dr. Polly Huang and the anonymous reviewers for the valuable feedback.

11. REFERENCES

[1] `http://infoplease.com/ipa/A0001451.html`.
[2] `http://oceanservice.noaa.gov`.
[3] `http://srwqis.tamu.edu`.
[4] `http://epa.ohio.gov`.
[5] `http://bit.ly/1tsVf29`.
[6] `http://blooms.uwcfl.org/mendota`.
[7] J. Ammerman and W. Glover. Continuous underway measurement of microbial ectoenzyme activities in aquatic ecosystems. *Marine Ecology Progress Series*, 201:1–12, 2000.
[8] J. Blum, D. Greencorn, and J. Cooperstock. Smartphone sensor reliability for augmented reality applications. In *MobiQuitous*, 2013.
[9] B. Boyd. *Introduction to Algae*. Prentice Hall, 2003.
[10] C. Brekke and A. Solberg. Oil spill detection by satellite remote sensing. *Remote Sensing of Environment*, 95(1):1–13, 2005.
[11] J. Crank. *The Mathematics of Diffusion*. Oxford University Press, 1979.
[12] M. Dunbabin and A. Grinham. Experimental evaluation of an autonomous surface vehicle for water quality and greenhouse gas emission monitoring. In *ICRA*, 2010.
[13] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *CVPR*, 2014.
[14] M. Faulkner, M. Olson, R. Chandy, J. Krause, M. Chandy, and A. Krause. The next big one: detecting earthquakes and other rare events from community-based sensors. In *IPSN*, 2011.
[15] Robotic fish. `http://nbcnews.to/1fGAomj`.
[16] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2015.
[17] R. Hecky and P. Kilham. Nutrient limitation of phytoplankton in freshwater and marine environments: a review of recent evidence on the effects of enrichment. *Limnology and Oceanography*, 33(4):796–822, 1988.
[18] S. Hemminki, P. Nurmi, and S. Tarkoma. Accelerometer-based transportation mode detection on smartphones. In *SenSys*, 2013.
[19] C. Hu, F. Müller-Karger, C. Taylor, D. Myhre, B. Murch, A. Odriozola, and G. Godoy. MODIS detects oil spills in Lake Maracaibo, Venezuela. *Eos, Transactions, American Geophysical Union*, 84(33):313–319, 2003.
[20] P. Jain, J. Manweiler, A. Achary, and K. Beaty. FOCUS: clustering crowdsourced videos by line-of-sight. In *SenSys*, 2013.
[21] S. Kako, A. Isobe, and S. Magome. 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):1156–1162, 2012.
[22] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu. SensEye: a multi-tier camera sensor network. In *MM*, 2005.
[23] G. LaRoche, R. Eisler, and C. Tarzwell. Bioassay procedures for oil and oil dispersant toxicity evaluation. *Water Pollution Control Federation*, pages 1982–1989, 1970.
[24] D. Lymberopoulos and A. Savvides. XYZ: a motion-enabled, power aware sensor node platform for distributed sensor network applications. In *IPSN*, 2005.
[25] R. Maree, P. Geurts, J. Piater, and L. Wehenkel. Random subwindows for robust image classification. In *CVPR*, 2005.
[26] K. Ogata. *Discrete-Time Control Systems*. Prentice Hall, 1995.
[27] T. Platt, F.-Y. Csar, and F. Kenneth. Marine ecology: spring algal bloom and larval fish survival. *Nature*, 423(6938):398–399, 2003.
[28] D. Rudnick, R. Davis, C. Eriksen, D. Fratantoni, and M. Perry. Underwater gliders for ocean research. *Mar. Technol. Soc. J.*, 38(2):73–84, 2004.
[29] L. Shapiro and G. Stockman. *Computer Vision*. Prentice Hall, 2001.
[30] Y. Shen, W. Hu, M. Yang, B. Wei, S. Lucey, and C.-T. Chou. Face recognition on smartphones via optimised sparse representation classification. In *IPSN*, 2014.
[31] J. Smith and S.-F. Chang. VisualSEEk: a fully automated content-based image query system. In *MM*, 1997.
[32] Stargate platform. `http://xbow.com`.
[33] T. Teixeira, E. Culurciello, J. Park, D. Lymberopoulos, A. Barton-Sweeney, and A. Savvides. Address-event imagers for sensor networks: evaluation and modeling. In *IPSN*, 2006.
[34] Y. Wang, R. Tan, G. Xing, J. Wang, X. Tan, and X. Liu. Samba: a smartphone-based robot system for energy-efficient aquatic environment monitoring. Technical Report MSU-CSE-15-4, Computer Science and Engineering, Michigan State University, 2015.
[35] Y. Wang, R. Tan, G. Xing, J. Wang, X. Tan, X. Liu, and X. Chang. Aquatic debris monitoring using smartphone-based robotic sensors. In *IPSN*, 2014.
[36] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, and R. Martin. Sensing vehicle dynamics for determining driver phone use. In *MobiSys*, 2013.