VISION-BASED FROZEN SURFACE EGRESS: A DOCKING ALGORITHM FOR THE ENDURANCE AUV

Aniket Murarka
Department of Computer Science
The University of Texas at Austin, USA
aniket@cs.utexas.edu

Gregory Kuhlmann
Department of Computer Science
The University of Texas at Austin, USA
kuhlmann@cs.utexas.edu

Shilpa Gulati
Department of Mechanical Engineering
The University of Texas at Austin, USA
gulati@mail.utexas.edu

Mohan Sridharan
Department of Computer Science
Texas Tech University, USA
mohan.sridharan@ttu.edu

Chris Flesher
Stone Aerospace, USA
chris.flesher@gmail.com

William C. Stone
Stone Aerospace, USA
bill.stone@stoneaerospace.com

Abstract

This paper presents a vision-based docking algorithm for an autonomous underwater vehicle (AUV). The algorithm allows the AUV to egress through a melthole in the frozen surface of a lake after the AUV's dead-reckoning system brings the vehicle in the vicinity of the melthole. A blinking light source is used to guide the robot towards the melthole and through it. A light detection and tracking algorithm performs a temporal analysis of images captured from an upward-facing camera to detect sources of illumination and identify and track the blinking target light source. The vehicle first moves in a spiral pattern to search for the target using the light-detection algorithm. On finding the light, the AUV ascends while keeping the light centered in the camera's field of view.

The vision-based docking algorithm was implemented on the ENDURANCE AUV and tested during a four-week-long scientific mission to explore West Lake Bonney in Antarctica in December 2008. The algorithm was used to ascend in 10 missions and to descend in 8 missions through a three-meter-deep melthole only slightly larger than the vehicle itself. In each instance, the vehicle was able to safely ascend or descend without coming into contact with the walls. Quantitative analysis of mission data confirmed that the tracking algorithm and ascent controller were robust and precise.

1 Introduction

Scientists across several disciplines have become increasingly interested in exploring and mapping lakes frozen over with ice. These environments are too dangerous for divers and too sensitive for large manned submersibles, making them prime candidates for autonomous vehicle exploration. One of the many challenges faced by autonomous underwater vehicles in exploring such environments is the recovery of the vehicle at the end of a mission. This is a particularly difficult problem because the navigation systems onboard these vehicles are typically not accurate enough to return the vehicle to its precise home position. At the same time, the maneuver has little room for error as entrance and egress through the ice requires the vehicle to pass through a narrow, confined region.

In this paper we describe a robust vision-based docking algorithm for safely recovering an AUV through a melthole in the ice layer. The algorithm was motivated by and designed for the ENDURANCE AUV for its four-week scientific mission to explore West Lake Bonney, Antarctica in December 2008 [12]. The vehicle entered the lake via a three meter vertical hole melted through the surface ice that was only slightly larger than the vehicle's diameter. During its daily scientific missions, ENDURANCE traveled over many hun-

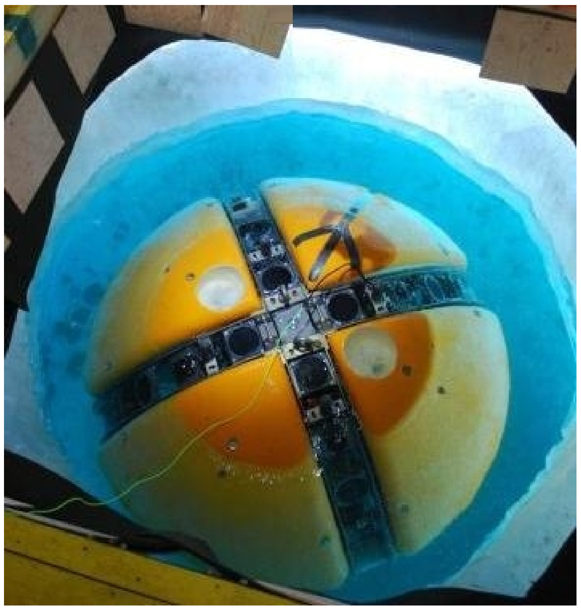


Figure 1: ENDURANCE AUV being deployed through the melthole in West Lake Bonney. The melthole diameter ($\approx$ 2 m) is only slightly larger than the vehicle's diameter ($\approx$ 1.8 m). A blinking light source was suspended approximately 1.5 m above the melthole.

dreds of meters before returning to the melthole and surfacing. The vehicle's dead-reckoning navigation system was sufficient to bring the AUV within a few of meters of the melthole but could not precisely position the AUV under the melthole for safe egress. We therefore had to develop a homing/docking method that not only had to center the AUV with respect to the melthole but had to detect the melthole in the first place. Once the melthole was detected and the AUV centered, the AUV could rise to the surface while avoiding contact with the melthole walls. Figure 1 shows the ENDURANCE AUV and the actual melthole.

Our vision based docking solution comprised both hardware and software components. For hardware, we suspended a downward-facing blinking light centered above the melthole and an upward-facing camera mounted on top of the vehicle. The software component consisted of a light detection module, a spiral search behavior, and an ascent controller to align and surface the vehicle. When the vehicle reached close to the melthole, as per its dead-reckoning, it transitioned to a spiral search routine – the vehicle spiraled outward from the starting position while looking for the blinking light in its camera. While executing its spiral path, the vehicle detected and tracked all light sources in the camera's field of view. On confirmed identification of the blinking target light, the vehicle transitioned from the search routine to the ascent controller. The ascent controller commanded the robot to rise while maintaining the target light in the center of the image and stopping the robot when it reached the water surface.

We chose a camera and a blinking light source for the homing operations after evaluating several alternatives, such as sonar. A vision-based approach is able to sample the environment more frequently and more richly than a sonar-based approach. In addition, it eliminates the need to address the challenge of overcoming the crosstalk between different sonar sensors that could occur in a narrow melthole. Other kinds of lights do not provide as strong a signal as a blinking one. A colored light doesn't work as the color signal dissipates quickly with distance. Arranging several lights in a pattern also does not work because flaring causes the distinct lights to appear as a single source. Likewise, lights of specific size and shape are almost impossible to discern at significant distances. A blinking light source on the other hand provides a powerful signature as the intensity signal penetrates to a good depth and because there are few light sources that blink brightly in natural environments. In Section 6 we further discuss some of the alternate solutions employed by other researchers in the literature.

Tracking a blinking light is, however, a challenging vision problem because it requires temporal analysis in addition to static analysis of sequential frames. Our approach uses contours to identify sources of illumination, and then applies a novel matching algorithm to register current light sources with past sources. The algorithm can distinguish between: an existing moving light source, a new light source entering the image, and an old light source that no longer needs to be tracked. Light sources that pass a series of filters on their size and shape are evaluated for the degree to which they blink. For flexibility, we do not assume that the light source is synchronized with the camera and thus the blinking may appear irregular. We provide a metric that is robust to these conditions, and that returns the most likely target in the image. The position of the target light, if found, is then passed to the ascent controller. The details of our light tracking approach are discussed in Section 4.

Our visual homing system was successfully deployed on the ENDURANCE mission in Antartica. In all missions where it was employed, it successfully found the light source and surfaced the vehicle through the melthole without collision. We present quantitative and qualitative results from the missions evaluating the ascent controller and the light detection module in Section 5. We begin with a description of the overall architecture of the system followed by the spiral and ascent behaviors and the light detection algorithm.

2 Docking System Architecture

The software architecture for visual homing consists of three main components: the system executive, the navigator and the light detection module. The system executive accepts commands from the user through a user interface. It can also execute scripted plans. The navigator can execute various behaviors such as homing, station-keeping etc. by instantiating an appropriate controller. The controller sends velocity commands to lower-level modules which are eventually converted into thrust values for the vehicle's six thrusters. The vehicle's control enables it to move independently in the lateral (x, y) plane and the vertical direction (z). The light detection module detects light sources in the images taken by the upward-facing camera, and identifies a blinking light source as the target. The coordinates of the light source center are expressed as angles $(\theta_x^t,\ \theta_y^t)$ in the image frame (Figure 2). These angles are computed as

$$\theta_x^t = \tan^{-1}\left((L_x - \frac{W_x}{2})\frac{2\tan(\alpha_x/2)}{W_x}\right) \tag{1}$$

$$\theta_y^t = \tan^{-1}\left((L_y - \frac{W_y}{2})\frac{2\tan(\alpha_y/2)}{W_y}\right) \tag{2}$$

where (L_x, L_y) are the image coordinates (in pixels) of the light center, (α_x, α_y) is the field of view of the camera in radians, and (W_x, W_y) is the width of the image in pixels (Figure 2).

As mentioned, the AUV's dead reckoning system is accurate enough to move the vehicle within a few meters of its home position under the melthole. At Lake Bonney, a 50% circular error probable (CEP) drift error of about 0.1% of distance traveled was observed [10]. Although this error is relatively small, a more accurate estimate of the melthole position is required for safe egress. The vehicle searches for the light source positioned over the melthole, centers it in its field of view, and ascends upward while avoiding collisions with the melthole walls.

The system executive initiates the visual homing routine by sending a message to the navigator when the dead-reckoning pose estimate indicates that the vehicle has arrived at its nominal home position. Upon receiving this message, the navigator initiates a spiraling behavior. In this behavior, the navigator sends a message to the light detection module to start detecting a target light source. At the same time, the vehicle starts to move outward in a spiral pattern while maintaining constant depth to search for a light source. If the light detection module is able to identify a target source, it starts tracking this source and sends a message to the navigator. The navigator then transitions to an ascent behavior. The light detection module continues to track the target while the vehicle keeps the light centered in its field of view using a proportional-derivative (PD) controller in the lateral plane. At the same time, it ascends vertically using a proportional-integral-derivative (PID) control law.

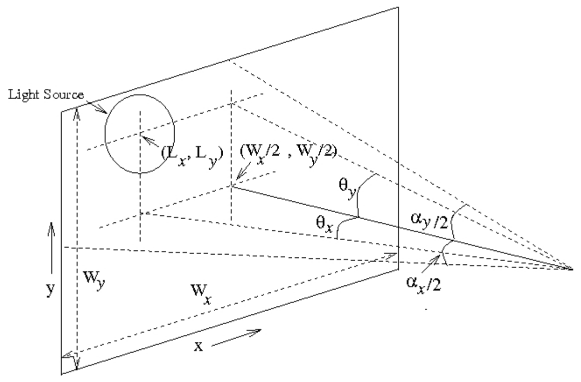


Figure 2: Various parameters associated with the image from the upward-facing camera. The light source is not necessarily centered in the image (see text for details).

When visual homing is used for descent, the spiraling behavior is not executed. This is because it is assumed that, during descent, the vehicle is directly beneath the target light source and hence no search is necessary. Instead, the navigator directly transitions to a descent behavior. This behavior is identical to the ascent behavior except that the PID control law for descent is used instead.

3 Navigator Behaviors

The visual homing system consists of two component behaviors: spiraling and ascent/descent. This section describes these two behaviors in detail.

3.1 Spiraling Behavior

As mentioned earlier, a search for the light source is performed by having the robot spiral outward from its present location while maintaining constant depth. A spiral search allows the vehicle to completely scan the under-surface of the ice above it.

The equation of the spiral (an Archimedean spiral) is

$$r = a + \frac{b}{2\pi}\theta \tag{3}$$

where a is an offset parameter which determines where the spiral starts. The value a is set to zero because we want the vehicle to start searching from its current location. The value b is the distance between the arms of the spiral. The parameter b is determined from the depth at which the spiral search is performed and the field of view of the camera. This value is chosen to ensure that the under-surface of the ice is completely scanned by the upward-facing camera for the light source, and that the overlap between search areas is not excessive. The θ parameter is varied from zero to θ_{max} such that a maximum radius of r_{max} is reached. The search radius itself is calculated from the lateral distance traveled by the vehicle and its dead-reckoning accuracy. For example, if lateral distance traveled is D and dead-reckoning accuracy is $p\%$ of D, then the search radius required is $pD/100$.

The position (x, y) of the vehicle in the lateral plane can be determined from the spiral coordinates (r, θ) as

$$\begin{aligned} x &= x_0 + r\cos\theta \\ y &= y_0 + r\sin\theta, \end{aligned} \tag{4}$$

where (x_0, y_0, z_0) is the initial position of the vehicle.

The (x, y) locations on the spiral are reached by using a PID controller that computes the appropriate velocity commands. This PID controller is similar to the z controller in Algorithm 1.

3.2 Ascent/Descent Behavior

Algorithm 1 outlines the controller used by the navigator to ascend up the melthole. The controller uses the coordinates of the center of the light source in the image as an error signal to center the vehicle on the light.

A PD controller is used to compute the vehicle's velocity in the lateral plane based on the light center coordinates. The controller tries to center the light source in the image thereby centering the robot on the light source. If the vehicle is already well-centered with respect to the light source in any one direction (x or y), then the commanded velocity in that direction is zero.

A non-zero z-velocity v_z^t is commanded only when the light is approximately centered in the image. A PID control law is used for the z-direction. This law uses an estimate of the vehicle's depth, z_{obs}, at the current and previous times to compute v_z^t as given in Algorithm 1. Note that positive z axis points in the downward direction.

Thus, the output of the controller is a vector of commanded velocities $(v_x^t,\ v_y^t,\ v_z^t)^T$. The controller stops when the vehicle reaches the water surface as indicated by z_{obs}. For descent, the vehicle uses the same controller as for ascent except that control in the z-direction is disabled. The vehicle is slightly weighed down and gradually sinks down the melthole while being controlled in the x and y directions.

```
Require: Center of light source in the image
        $(\theta_x^t, \theta_y^t)$, estimate of the vehicle's depth
        $z_{obs}^t, z_{obs}^{t-1}$, and estimate of vehicle's
        velocity $v_{z_{obs}}^t$
// Smoothed error where $\alpha = 0.75$
// $e_x^{t-1}$ is a static variable
$e_x^t \leftarrow \alpha\, \theta_x^t + (1-\alpha)\, e_x^{t-1}$
$e_y^t \leftarrow \alpha\, \theta_y^t + (1-\alpha)\, e_y^{t-1}$

// Derivative of error
$\dot{e}_x^t \leftarrow e_x^t - e_x^{t-1}$
$\dot{e}_y^t \leftarrow e_y^t - e_y^{t-1}$

// If error is greater than a
// threshold use PD controller,
// else set velocity to zero
if $|e_x^t| > \theta_T$ then
    $v_x^t \leftarrow \min(v^{max}, -K_p e_x^t - K_d \dot{e}_x^t)$
else
    // For low error, commanded
    // velocity is zero
    $v_x^t \leftarrow 0$
end

if $|e_y^t| > \theta_T$ then
    $v_y^t \leftarrow \min(v^{max}, -K_p e_y^t - K_d \dot{e}_y^t)$
else
    $v_y^t \leftarrow 0$
end

// Ascend/descend only when
// vehicle is approximately
// centered
if $|e_x^t| < \theta_R$ and $|e_y^t| < \theta_R$ then
    // Error in z
    $e_z^t \leftarrow z_{obs}^t - z_{obs}^{t-1}$
    // PID Controller
    $v_z^t \leftarrow -K_{pz}\, e_z^t - K_{dz}\, v_{z_{obs}}^t$
        $- K_{iz} \min(e_z^t\, \Delta t, \Delta e_z^{max})$
else
    $v_z^t \leftarrow 0$
end

// Store previous error estimate to
// compute smoothed errors
$e_x^{t-1} \leftarrow e_x^t$
$e_y^{t-1} \leftarrow e_y^t$

return $(v_x^t,\ v_y^t,\ v_z^t)^T$
```

Algorithm 1: Ascent Controller: Returns command velocities $(v_x^t,\ v_y^t,\ v_z^t)$ at time t.

4 Light Detection Module

The controller described in the previous section requires accurate detection and tracking of the target light source in the camera image. At the camera frame rate of around 6Hz, frames are captured and then processed by our low-level vision routines (built on OpenCV) to identify high-contrast contours as candidates for the target light. These contours are then filtered based on their roundness and size to eliminate obvious false positives. For each candidate contour that passes our filters, we pass the center and radius of the bounding circle to our light tracking algorithm.

Unlike the typical light tracking problem, which assumes that a target light is always illuminated, our problem is significantly more challenging. We must track a light even during the frames when it is not illuminated and distinguish it from persistent light sources in the image, all while the camera is moving. To tackle this problem we make a couple of assumptions. First, we assume that, although there maybe be many persistent light sources in the image, there is only one blinking light at any given time. And second, we assume that the camera does not move too quickly. The definition of “too quickly” is determined by constants in the algorithm that trade off false positives and false negatives.

Algorithm 2 outlines our blinking light detection and tracking algorithm. The algorithm maintains a set, L, which contains the current light sources being tracked, including their bounding circles and *histories*. A history is a list that contains whether or not the light source was seen or unseen for each frame going back to some upper bound on the history length (24 frames in our experiments). The algorithm updates this data structure by incorporating C, the set of candidate light sources seen in the current frame. The tracker must identify which candidates correspond to light sources that have already been seen and which candidates are new. We take a greedy edge-elimination approach, as described in Algorithm 2 and illustrated by example in Figure 3.

The algorithm begins by computing a matrix, Δ, containing the distances between all currently tracked light sources and the candidates in the frame. The list, λ, is a list of all possible pairs, as illustrated by the edges in the left-hand graph in Figure 3.

The first round of elimination, removes all edges with distance greater than the threshold Δ_{max} (50 pixels in our experiments). Essentially, if a candidate light is farther away than this value from a light in the tracker, they cannot be considered the same light source. We set this threshold large enough to allow the robot to move some significant amount between frames, but small enough to avoid clumping all light sources together. At the same time, we eliminate edges if they connect light sources

```
**Require**: Lights tracked in memory, $L^{t-1}$,
          and candidate lights returned by
          contour algorithm on current frame,
          $C^t$.
$\Delta \leftarrow |L^{t-1}| \times |C^t|$ matrix
$\lambda \leftarrow L^{t-1} \times C^t$
**foreach** $(l, c) \in \lambda$ **do**
    $\Delta_{l,c} \leftarrow ||l.center - c.center||$
**end**
// Cull large edges
**foreach** $(l, c) \in \lambda$ **do**
    **if** $\Delta_{l,c} > \Delta_{max}$ *or*
    $|\log(l.radius) - \log(c.radius)| >$
    $R_{max}$ **then**
        Remove $(l, c)$ from $\lambda$
    **end**
**end**
Sort $\lambda$ by descending $\Delta_{l,c}$
// Greedy edge removal
**foreach** $(l, c) \in \lambda$ **do**
    **if** $(l', c) \in \lambda$ *or* $(l, c') \in \lambda$ $[l' \neq l, c' \neq c]$
    **then**
        Remove $(l, c)$ from $\lambda$
    **end**
**end**
// Update histories
**foreach** $l \in L^{t-1}$ **do**
    **if** $(l, c) \in \lambda$ **then**
        Mark $L^t_l$ seen
    **else**
        Mark $L^t_l$ unseen
    **end**
**end**
// Add new candidates
**foreach** $c \in C^t$ **do**
    **if** $(l, c) \notin \lambda$ **then**
        Add $c$ to $L^t$
    **end**
**end**
// Find target by score
$l^* \leftarrow \arg\max_l score(L^t_l)$
**if** $score(l^*) \geq S_{min}$ **then**
    **return** $l^*.center$
**else**
    **return** $\perp$
**end**
```

Algorithm 2: Light Tracker: Returns position of blinking light (θ^t_x, θ^t_y) in image at time t, or $\perp$ if not found.

with drastically different sizes. If the logs of their radii differ by more than R_{max}, then the edge is eliminated. The primary motivation for this filter is to prevent specular glare on the light source glass from being recognized as the light when it is off, a problem we encountered during early experimentation.

After the first culling, the edges that remain appear as depicted in the middle graph in Figure 3. Notice that some nodes still have more than one edge, which means that there are multiple potential matches to be made. The next round of elimination determines which of those potential matches is best. The algorithm sorts the remaining edges by decreasing distance and then eliminates them one-by-one. An edge will not be eliminated if it is the only edge remaining for the nodes it connects, because that edge is treated as a correct match.

In the end, the graph that results from the greedy culling algorithm looks like the right-hand graph in Figure 3. No node has more than one edge. The edges that remain connect the candidates in the current frame with their corresponding light source histories. Orphaned nodes in L (one in the figure) are tracked light sources that were not seen in the current frame. Their histories are updated to reflect that they were unseen. Orphaned nodes in C (two in the figure) are newly-observed candidates, which are added to the list of tracked light sources with a fresh history.

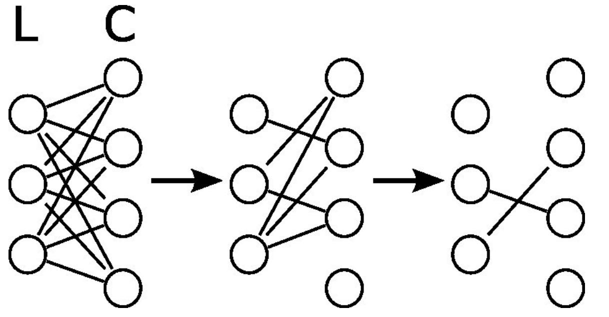


Figure 3: To process new candidate light sources, our algorithm begins with a bijection between those candidates (C) and the currently tracked light sources (L) [left]. Edges are removed if the distance between the sources is too large or if the disparity in their sizes is too high [middle]. Finally, edges are removed in order from greatest distance to smallest, while not removing edges (l, c) that would orphan both l and c [right].

With L updated, we compute a score for each tracked light source. After some experimentation with alternatives, the score that worked best was the simplest: number of transitions. Every time the light switches from seen to unseen or unseen to seen in its bounded history, that light source receives a point. The highest possible score, then, is half the history length. In practice, however, a true positive tends to score in the 3–7 range. Through informal experimentation, we arrived at a value of 4 for the minimum threshold value on the score, S_{min}. Thus, the light source with the highest score above threshold is returned as the target. If no light sources score above the threshold, then the algorithm returns a null value.

5 Experiments and Results

Visual homing was used for 10 missions in West Lake Bonney. It was used 10 times for ascent and 8 times for descent. Of the 10 times when it was used for ascent, spiraling behavior was executed 5 times. Two times, the vehicle was farther away form the melthole than could be reached with a single spiral search. Here, spiral search was executed 2 and 3 times respectively by manually issuing a visual homing command through the user-interface. The transit depth of the vehicle was 5 m and hence most ascents were initiated at a depth of 5 m. For the rest of this paper the ascents will be labeled as ascent-i where i is the mission number. The descents will be labeled in a similar manner.

The value of the parameter b for spiral search was chosen to be 0.25 m. The longest length mission performed at West Lake Bonney was approximately 1800 m. Since the dead reckoning accuracy for the vehicle is approximately 0.1% of total distance traveled, a search radius r_{max} of at least 1.8 m was necessary. To account for unexpected drift, we set $r_{max} =$ 4.8 m for Lake Bonney missions.

For all the 18 instances, visual homing was successful in guiding the vehicle up and down the melthole without collisions with the walls. Figure 4 shows some images taken by the upward-facing camera during an ascent. Figure 5 shows the path of the vehicle for two ascents and one descent. For the ascent, the vehicle's path during a spiral search for the light source is also shown. The spiral search proved to be a simple yet effective way to search for the light source.

5.1 Ascent/Descent Controller Precision

We analyze the precision of the ascent controller [1]. A sequence of time-stamped poses is available from the vehicle's sensors. We use the standard deviation in the vehicle's x and y coordinates as an estimate of the precision of the controller for an ascent (the standard deviations capture not only the ascent controller's precision but also the precision of the light detection module at

[1]Since we do not have ground truth pose information available to us, we cannot evaluate the accuracy of the controller.

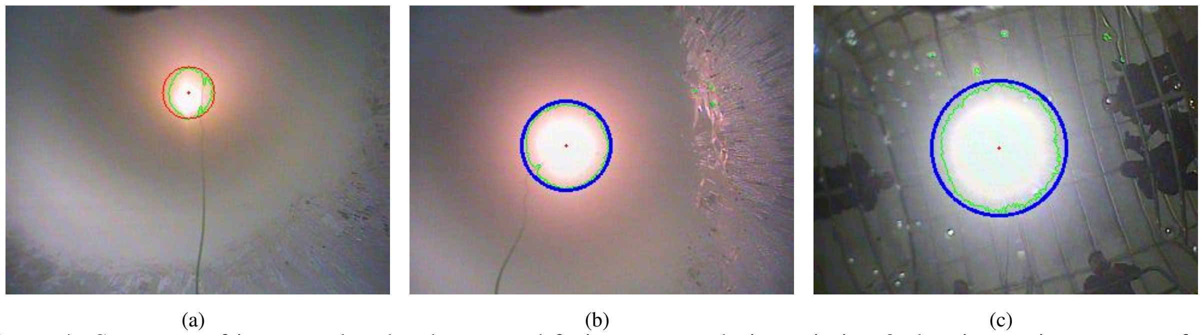


Figure 4: Sequence of images taken by the upward-facing camera during mission-3 showing various stages of the visual homing behavior. (a) During search, the light-detection module identifies a candidate light source (red circle). Note that the light source is not centered in the image. (b) After observing the light source blink a few times, the source is confirmed as the desired target (blue circle). The robot also starts centering the light source in the image. (c) The view from the camera when the robot reaches the top of the melthole using the ascent controller with the light source well centered.

locating the center of the light source). The standard deviation is computed for pose values obtained after the vehicle has approximately centered itself with respect to the light source. Since both image and pose data is available for only 7 missions, we analyze data for these missions. Table 1 lists the standard deviations for the 7 missions.

In addition to standard deviation values for each mission, we compute the overall standard deviation. To do this, we first normalize all x and y values for each ascent by their means. If $\overline{x}$ is the mean x pose for an ascent, then the normalized value $\widehat{x}$ for a particular x coordinate is given by $\widehat{x} = x - \overline{x}$. The normalized values are combined into a single vector and the overall precision is then given by the vector's standard deviation. The overall precision for the y coordinates is similarly computed.

As we can see from Table 1, the overall value for the standard deviation is 5.70 cm in the x direction and 4.60 cm in the y direction. The vehicle has a clearance of 10 cm on all sides when it is in the melthole implying that for about 92% of the time, the vehicle was fairly well centered in the melthole and not touching any walls. Qualitatively, the vehicle came up the melthole successfully every time without suffering any damage, suggesting that at worst the vehicle only lightly grazed the melthole walls. Thus, the controller and the light detection algorithm were able to successfully surface the robot. The precision for the descents is numerically similar to that for the ascents.

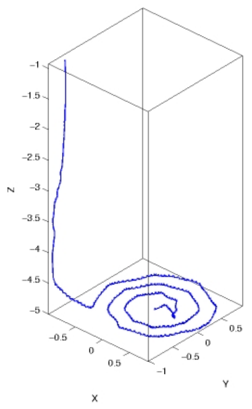
5.2 Light Detection Accuracy

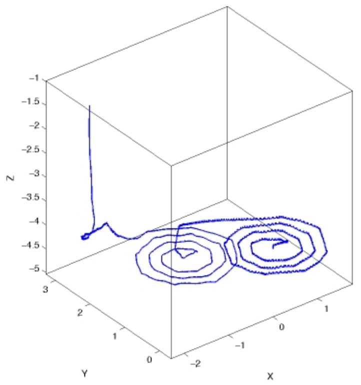
The performance of the light detection algorithm is best evaluated by its impact on the controller. However, to provide more insight into its strengths and weaknesses, we present some statistics about the light tracking output. First, to help gain a qualitative sense of algorithm performance, Figure 6 shows a few key frames taken by the camera during a field ascent. The images are annotated with shapes to visualize the internal state of the light tracking algorithm through the course of the ascent. A red circle indicates a candidate light source. A blue circle indicates a light source determined to be the target. From Figure 6(e), we see that misclassifications are possible, but Figure 6(f) shows that the algorithm is able to recover.

To evaluate algorithm performance, we are primarily interested in four criteria. First, the algorithm should acquire and detect the blinking light quickly. Second, it should not lose track of the light once it has been acquired, producing false negatives. Third, it should not produce false positives by misclassifying non-target light sources. And fourth, it should maintain an accurate estimate of the target center.

For seven separate ascents of the vehicle, we evaluated the light tracking algorithm against these criteria. Our performance metrics require a comparison with the true location of the target light source. Without access to ground truth, we approximated the location of the target light source by hand-labeling the images in our data set. For each frame in which the true target light source was present, we manually recorded the pixel location that appeared to be closest to the center of the target. We then ran the same images through the light-detection algorithm and computed the error in this estimate along with several other statistics. The complete results are shown in Table 5.2.

The first statistic, "Total frames", is the number of frames stored by the camera from the time the first candidate light source is identified until the vehicle completes ascent. Due to real-time requirements, we did not store every image used by the algorithm while the vehicle was


(a) ascent-10


(b) ascent-9

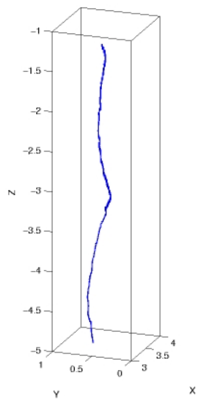
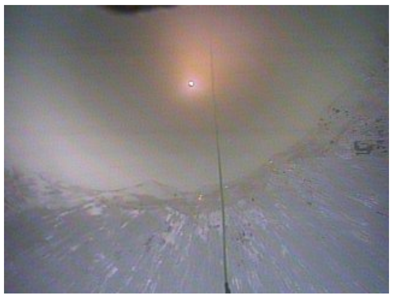
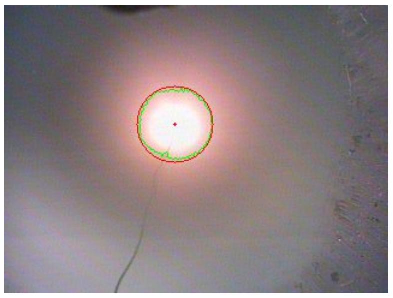

(c) descent-6

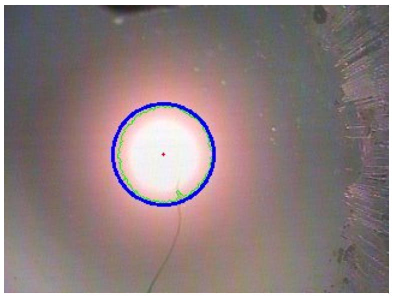
Figure 5: Plots showing the vehicle's path once the visual homing algorithm starts. Shown are two ascents and one descent. (a) In ascent-10, the robot searches spirally for the light source and once it locks onto the light source, the robot begins to rise while staying centered on the light. (b) In ascent-9, the vehicle fails to find a light source the first time it spirals. It then drifts for a short distance before the visual homing algorithm is restarted. This time the vehicle finds the light source and rises. (c) A descent – the controller stops when the vehicle reaches a depth of 5 m.

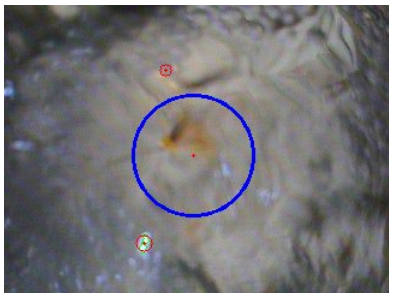
Ascent No.	3	5	8	9	10	12	13	Overall
σ_x (cm)	8.77	6.20	5.32	2.54	2.65	6.27	5.54	5.70
σ_y (cm)	1.71	3.09	3.15	7.78	5.94	5.48	2.05	4.60

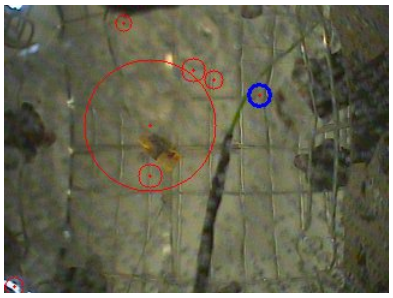
Table 1: Standard deviations in the x and y coordinates of vehicle's pose during various ascents. Also shown are the overvall standard deviations computed for all ascents combined.


(a)


(b)


(c)


(d)


(e)

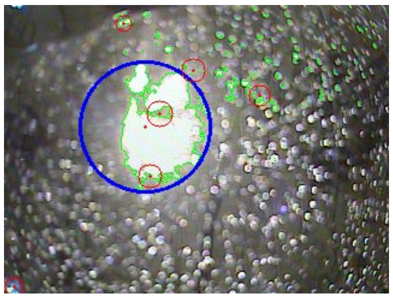

(f)

Figure 6: Images taken by the upward-facing camera. (a) Light too small to be recognized. (b) Light recognized as candidate. (c) Target identified as blinking. (d) Blinking light tracked while unlit. (e) Misclassification. Blue circle is false positive. Large red circle is true target. (f) Recovery after misclassification.

Ascent No.	3	5	8	9	10	12	13
Total frames	190	278	243	150	193	179	153
Acquisition time	21	12	162	101	123	95	8
Drops	1	0	0	0	0	0	0
Total drop time	11	0	0	0	0	0	0
Misclassification	1	0	0	0	0	0	0
Misclassification time	15	0	0	0	0	0	0
Average distance	11.2	1.91	3.63	4.44	2.24	3.02	3.38
Avg. dist. w/o misclassified	2.83	1.91	3.63	4.44	2.24	3.02	3.38

Table 2: Statistics evaluating blinking light tracker algorithm across seven separate ascents at Lake Bonney (see Section 5.2 for details).

in operation. Although the camera operated at 30Hz, images were stored at about 6Hz, recording roughly every fifth frame. Thus, for example, for an ascent that lasts around 30 seconds, roughly 180 frames would be recorded.

The next metric, "Acquisition time", is the number of frames between the first candidate light and the first identified blinking target. If the target (blue circle) is lost after initial acquisition, then we consider that a "Drop". We record the number of drop occurrences and the total duration (in frames) over the occurrences. We do the same for "Misclassifications", which are instances when the algorithm locks on to the incorrect light source. Finally, we measure the Euclidean distance (in pixels) between the true target position and the algorithm output. Because misclassifications have a large impact on this metric, we record two versions of this statistic: one with misclassified instances included, and one without.

From Table 5.2 it is clear that the algorithm performed well during the Lake Bonney deployment. In only one of the seven ascents did the algorithm ever encounter any drops or misclassifications. Also, the low-level contour algorithm appears to have done a good job of finding the centers of the light sources, as is evidenced by the average distance values.

To achieve these strong results, the algorithm gives up quite a bit in terms of acquisition time. Although shorter acquisition times would certainly be preferable, for the Lake Bonney missions it was extremely important to have very few false positives. The misclassification of a distant target could have caused the vehicle to slam into the ice and cause damage. For that reason, the algorithm parameters were tuned to err on the side of safety.

6 Related Work

In our application, we are interested in finding the specific light source to enable the AUV to egress the melthole reliably. The core challenge addressed is that of tracking multiple hypotheses across time, and then estimating a specific target *state transition*. This is a well-researched problem in the AUV community, and similar challenges arise in several other computer vision and robotics applications. We review a few representative techniques here.

Feezor et al. [6] proposed an electromagnetic (EM) homing/docking system for an AUV. It consists of a transmitter located at the dock and a receiver located on the vehicle. The transmitter generates two orthogonal oscillatory magnetic fields from two coils, while three orthogonal coils (with co-located centers) mounted on the nose of the AUV provide the directional information required to guide the vehicle to the docking position. Stokey et al. [11], on the other hand, used an Ultra Short Base Line (USBL) rangefinder to determine the location of a transponder attached to the docking station.

For the purpose of docking, vision provides a few significant advantages. Given the recent developments in sensor technology, high-fidelity visual sensors are available at moderate costs. In addition, the information obtained from the visual sensors does not have a minimum detection range. Furthermore, it is possible to devise highly precise control algorithms using the visual input, leading to smaller docking areas in comparison to the other (non-visual) techniques.

The typical approach for using visual information for homing/docking an AUV involves a contrived arrangement of light sources and detectors. For instance, Cowen et al. [4] developed a vision-based approach for guiding a vehicle to an underwater dock. Their optical system uses a photo-diode split into four quadrants, with a photo-detector positioned behind a plano-convex lens that collects the incoming light. When the tracker is aligned with the light beacon, the light intensity is equal in all four quadrants of the detector. The light source is set up to oscillate at 40Hz in order to differentiate it from other possible sources (e.g. the sun), and appropriate band-pass filters are associated with each photodetector quadrant. This system has a very high precision ($\approx$ 1 cm) and is able to perform docking from a maximum range of $\approx$ 100 m, but found it difficult to differentiate sunlight from the light source under certain conditions.

Richburg et al. [9] have patented a method to guide a vehicle to a specific position using a light source that is divided into four panels. The vehicle can detect all four panels if that light source is within the primary field of view of the camera. Outside this specific region, the detected panels provide an indication of the desired direction of motion of the vehicle. In search mode, when the vehicle attempts to locate the light source, all panels oscillate at the same rate. Once the light source has been sighted, a signal is sent to the docking station, causing the panels to oscillate at different rates to distinguish between the panels while the vehicle performs its homing maneuver.

Deltheil et al. [5] presented an interesting comparison of acoustic, magnetic and optical sensing methods for recovery of an autonomous underwater vehicle. They showed that the optical system provided the best performance in terms of homing/guiding an underwater vehicle to a desired point reliably. There have hence been several attempts in the recent past to use visual information for similar applications. Hong et al. [7], for instance, proposed the use of camera images to estimate the relative pose of a AUV with respect to an underwater dock. Lights arranged on the rim of a circular dock were projected to an ellipse on the 2D (camera) image plane. The geometric shape of the ellipse provided an estimate of the relative distance and pose.

More recently, Park et al. [8] proposed a similar scheme that uses a CCD camera to detect a set of five lights arranged in a specific pattern around the rim of the dock entrance. The input image is binarized and the candidate light blobs are filtered with suitable masks and heuristics, based on the known characteristics of the light sources. Next, the detected light sources are matched against the expected pattern to determine the relative position of the dock entrance, and generate appropriate vehicle motion commands. Such methods can result in accurate docking by estimating the pose of the vehicle relative to the pattern of light sources, but the entire pattern of lights needs to be within the field of view.

While these previous approaches have performed successfully, they are too complex for our operational problem. We take the approach of using a single blinking light as the homing beacon – a solution that is simple but provides a unique signature nevertheless. The hardware required for this approach is available off-the-shelf and the software, as described in previous sections, requires very little calibration in the field. As a result, our approach eliminates the above-mentioned constraints of existing approaches.

In the computer vision literature, considerable work exists on tracking multiple hypotheses using established state estimation techniques such as Kalman Filters and Particle Filters [13]. Extensive research has been performed on the data association challenges that arise while tracking the multiple hypotheses [1, 3]. In addition, the desired high-level target (i.e. a blinking light) can be modeled as a *state transition* that is detected using techniques such as Hidden Markov Models [2]. Since computational efficiency is of great importance in our application and the application is simple enough, we do not require the use of heavy duty probabilistic techniques. Instead we devise and use a simple and efficient matching algorithm to detect the blinking light source and track it accurately over a period of time.

7 Conclusions

We have presented a novel vision-based docking algorithm for an autonomous underwater vehicle. The algorithm allows the AUV to exit through a hole in the ice over a frozen lake. The system uses a relatively unique homing signal in the form of a blinking light source. On reaching close to the melthole after a mission, the robot initiates the visual homing routine and enters a spiral search while scanning the surface of the ice for the blinking light source using an upward facing camera. A temporal analysis of the input images is used to identify sources of illumination, track their positions, sizes and transition (off-on-off) histories and then robustly identify the blinking light source amongst all light sources. Once the light source has been identified the robot centers itself with respect to the light and rises up through the melthole using an ascent controller.

The overall algorithm was tested successfully during a four-week scientific mission to explore West Lake Bonney, Antarctica in December 2008. The robot completed 10 homing runs (ascents) using the system as well 8 descents. Quantitatively, the precision of the ascent/descent controller was sufficient to keep the robot away from the melthole walls 92% of the time. Of the seven ascents on which the light detection module was evaluated on, in only one ascent was the incorrect light source identified, that too temporarily. Qualitatively, the robot successfully surfaced or descended in all runs without colliding with the melthole walls.

There are several directions in which the visual homing system can be extended. Currently, we do not make any assumptions about the blinking light source. Occasionally, this leads to incorrect detection. If we were to assume a specific blink frequency the algorithm could be made more robust since it is unlikely that other light sources will blink at the same frequency. Another way to extend the work would be to have a more intelligent search routine for the light. Currently, the robot simply spirals to search – it might be possible to combine the spiral search with a more greedy search that uses intensity gradients. This would be particularly applicable when a spiral search fails and the search has to be manually restarted. A gradient descent search might direct the robot automatically towards the light source.

References

[1] Y. Bar-Shalom and T.E. Fortmann. *Tracking and Data Association*. Academic Press, Boston, USA, 1988.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2008.

[3] R. Brooks and S. Iyengar. *Multi-sensor Fusion: Fundamentals and Application with Software*. Prentice Hall, 1998.

[4] S. Cowen, S. Briest, and J. Dombrowski. Underwater docking of autonomous undersea vehicles using optical terminal guidance. In *Oceans MTS/IEEE Conference*, pages 1143–1147, 1997.

[5] C. Deltheil, L. Didier, E. Hospital, and D. P. Brutzman. Simulating an optical guidance system for the recovery of an unmanned underwater vehicle. *IEEE Journal of Oceanic Engineering*, 25(4):568–574, 2000.

[6] M. D. Feezor, F. Yates Sorrell, P. R. Blankinship, and J. G. Bellingham. Autonomous underwater vehicle homing/docking via electromagnetic guidance. *IEEE Journal Of Oceanic Enigineering*, 26(4):1137–1142, 2001.

[7] Y. H. Hong, J. Y. Kim, P. M. Lee, B. H. Jeon, K. H. Oh, and J. H. Oh. Development of the homing and docking algorithm for AUV. In *Thirteenth International Offshore and Polar Engineering Conference*, pages 205–212, 2003.

[8] J.-Y. Park, P.-m. Lee B.-h. Jun, and J. Oh. Experiments on vision guided docking of an autonomous underwater vehicle using one camera. *Ocean Engineering*, 36(1):48–61, 2009.

[9] C. Richburg and D. D. Hobden. Method for guiding a vehicle to a position. In *United States Patent: US 6957132 B1*, October 2005.

[10] K. Richmond, S. Gulati, C. Flesher, B. P. Hogan, and W. C. Stone. Navigation, control, and recovery of the ENDURANCE under-ice hovering AUV. In *International Symposium on Unmanned Untethered Submersible Technology UUST*, 2009.

[11] R. Stokey, M. Purcell, N. Forrester, T. Austin, R. Goldsborough, B. Allen, and C. von Alt. A docking system for REMUS, an autonomous underwater vehicle. In *MTS/IEEE OCEANS Conference*, volume 2, pages 1132 – 1136, 1997.

[12] W. C. Stone, B. Hogan, C. Flesher, S. Gulati, K. Richmond, A. Murarka, G.Kuhlman, M. Sridharan, P. Doran, and J. Priscu. Sub-ice exploration of west lake bonney: ENDURANCE 2008 mission. In *International Symposium on Unmanned Untethered Submersible Technology UUST*, 2009.

[13] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, USA, 2005.