Bioinspiration & Biomimetics

NOTE

Human-in-the-loop active electrosense

RECEIVED
17 May 2016

REVISED
2 November 2016

ACCEPTED FOR PUBLICATION
15 November 2016

PUBLISHED
20 December 2016

Sandra Fang[1], **Michael Peshkin**[1] **and Malcolm A MacIver**[1,2,3]

[1] Department of Mechanical Engineering, Northwestern University, Evanston, IL, USA
[2] Department of Biomedical Engineering and Department of Neurobiology and Physiology, Northwestern University, Evanston, IL, USA
[3] Author to whom any correspondence should be addressed.

E-mail: maciver@northwestern.edu

Keywords: active electrosense, virtual reality, teleoperation, human–computer interaction, augmented reality

Abstract

Active electrosense is a non-visual, short range sensing system used by weakly electric fish, enabling such fish to locate and identify objects in total darkness. Here we report initial findings from the use of active electrosense for object localization during underwater teleoperation with a virtual reality (VR) head-mounted display (HMD). The advantage of electrolocating with a VR system is that it naturally allows for aspects of the task that are difficult for a person to perform to be allocated to the computer. However, interpreting weak and incomplete patterns in the incoming data is something that people are typically far better at than computers. To achieve human–computer synergy, we integrated an active electrosense underwater robot with the Oculus Rift HMD. The virtual environment contains a visualization of the electric images of the objects surrounding the robot as well as various virtual fixtures that guide users to regions of higher information value. Initial user testing shows that these fixtures significantly reduce the time taken to localize an object, but may not increase the accuracy of the position estimate. Our results highlight the advantages of translating the unintuitive physics of electrolocation to an intuitive visual representation for accomplishing tasks in environments where imaging systems fail, such as in dark or turbid water.

1. Introduction

Underwater environments are highly unfavorable to visually-guided behavior even under ideal conditions of being near the surface on a clear day [1]. Vision fails with turbidity or darkness, and with this failure comes the cost of slowed or stopped work for underwater tasks such as repair work or disaster recovery. For example, the Deepwater Horizon oil spill in 2010 demonstrated the crucial need for teleoperation systems that are robust to video 'brown outs'—where propellers of remotely operated vehicles disturb sediment halting work for hours at a time—or the presence of unrefined oil in the water that obscures vision. This hindered teleoperators on the surface from effectively interacting with objects and performing repairs. In environments similar to these, long range non-visual sensing approaches such as sonar are ineffective due to scattering and clutter, while zero range sensing such as touch may not be possible or sufficiently rich to guide the work.

We take inspiration from biological systems and use active electrosense, a near-range sensing modality used by weakly electric fish, as a possible solution for remote object manipulation in environments where vision fails. Nocturnal weakly electric fish inhabit the turbid waters of South American and African rivers and generate a small AC electric field. Objects in the field cause perturbations that can be sensed using highly sensitive electroreceptors distributed over their bodies. Any object with a conductivity or capacitance differing from that of water is electrically visible to these fish and can be distinguished based on its material or dielectric properties [2].

In this paper, we use the SensorPod, a capsule-shaped underwater robot capable of artificial electrosense (figure 1) [3–5]. The SensorPod emits an oscillating electric field with emitters at the front and rear of the capsule, and senses voltage perturbations in the field using two rows of electrodes along the midline on the left and right side of the robot. The SensorPod measures perturbations of an object by taking the

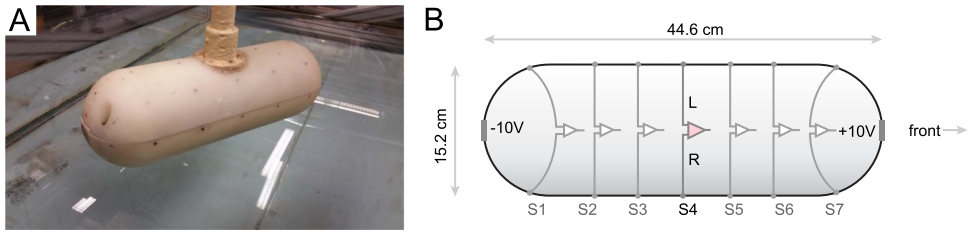

Figure 1. (a) Image of SensorPod over tank. (b) A schematic of the SensorPod viewed from above. The SensorPod has seven differential voltage sensors on its midline that measure the voltage across its left (L) and right (R) side. Emitters at each end of the robot emit an alternating ± 10 V field. The front end of the SensorPod is defined to be facing the positive x-axis when the SensorPod is aligned at 0° with respect to the x-axis. For our study only the sensor pair labeled S4 was used.

difference in voltage measured across left and right side pairs.

1.1. Prior work

There has been significant work on the problem of object characterization and localization with active electrosense for robotic systems [5–7]. Alamir attempted to solve the inverse problem by using graphical signatures to alleviate the computational burden [8]. Lebastard *et al* [9, 10] determined the size of a sphere by navigating an electrosense robot around the sphere, Ammari *et al* [11] developed shape-classification algorithms based on comparing features of the electric image that are invariant under rigid motion or scaling to a collection of learned shapes, and Bai *et al* [3] classified the aspect ratio, size, distance and orientation of spheroids based on signals acquired during algorithmically prescribed motions around detected object. Bai and Snyder also identified object properties by varying frequency and phase [12, 13]. Solberg *et al* [14] used a probabilistic model to perform localization based on a preexisting 2D map of the space surrounding the object. Nguyen *et al* [15] performed sparse beamforming with an object of known electric signature in 2D space to localize 0-2 objects. Miller *et al* [4] calculated an ergodic trajectory that maximized the amount of time spent in information-rich areas to localize an object or distinguish it from a distractor object, and Silverman *et al* [16] performed a modified range-only SLAM for the SensorPod itself using an incomplete map. Electrosense for pretouch and grasping has also been studied by Smith *et al* [17].

Despite this progress, algorithms developed for automatic electrosense are relatively fragile and rely heavily on simplifications or assumptions such as having a map of the space beforehand and knowing the electrical signature of the target object. Not only is the discipline of artificial electrosense (or 'machine electrosense' in analogy to machine vision) in its early stages of development, the inverse problem of reconstructing an object's position and properties given its electric image is severely ill-posed [18]. The advantage of teleoperation that is augmented with information from electrosense algorithms is that it naturally allows for *human-assisted computation*, in which certain tasks that are computationally difficult or impossible are outsourced to the human. Adding a human in the loop creates robustness in algorithms as we are adept at extrapolating from incomplete information and making decisions quickly in unpredictable environments. A barrier to integrating a human in an electrosensory system, however, is that electrosensory signals are unintuitive for people to interpret. Thus, we endeavored to use a virtual environment (VE), rendered with the Oculus Rift head-mounted display (HMD) to assist the human operator.

1.2. VEs for teleoperation

It is worth spending some time defining the term *virtual reality* (VR) more precisely. We borrow heavily from Chalmers [19] by using the following definition: VR is the general term for the technology that sustains a VE, or for the environment itself. This environment typically has two or three of the following characteristics: it is (partially or fully) computer generated, it is immersive, and it is interactive. In recent times, VR has become more synonymous with using HMDs to generate a 3D virtual world, although non-immersive VR can also be achieved through the use of 2D screens.

When the VE is partially generated—such as by overlaying virtual models over a live video feed—the result is *augmented reality* (AR). It may be more appropriate to say that we are proposing to perform teleoperation with electrosense using AR. Although we are forced to render a complete VE due to the real underwater environment being dark or opaque, the environment and model of our robot correspond closely to their physical counterparts—inasmuch as they are known. For example, the movement of the robot as seen in the head mounted display is as it occurs in the real world. For our laboratory prototype, we also render a portion of the known geometry of the tank containing the workspace for the electrolocation trial. More or less of the geometry of the world can be rendered according to the application and what is known by the system. The only features that are purely virtual are the hints and the visual representation of the electrosensory signals.

VR has been shown to be beneficial for aiding physical object manipulation and interaction tasks,

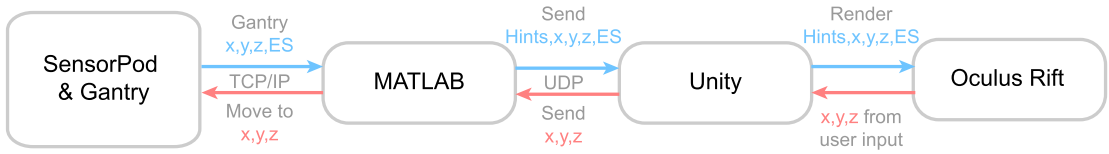

Figure 2. Communication pathways between the four major components. The SensorPod and Gantry sends its position (x, y, z), orientation (θ), and voltage perturbation values (ES, or electrosense data) to MATLAB, which runs a simple algorithm to determine the relevant pieces of information to send to Unity. Unity then renders this information for the Oculus Rift. Control input is passed all the way back to the SensorPod and gantry, which will move to act out the command. Information between the gantry and MATLAB is sent via TCP/IP, whereas information between MATLAB and Unity is sent via UDP.

especially for tasks such as computer-aided design or during teleoperation [20–23]. In addition, rendering *virtual fixtures*, or abstract sensorial data overlaid on the workspace, can reduce the cognitive load for the human operator [24]. We choose a 3D display over a 3D screen as it allows the user to orient and move the camera naturally (by simply tilting their head) and also offers better depth perception than a 2D screen. Thus if we consider that electrosense teleoperation is likely to be integrated with robotic manipulators, there is a strong incentive to develop a 3D VR application.

We rely on the idea that one can use a predefined electric image of an object to localize its position even in an unknown environment with sparse information. The computer will 'translate' electrosense readings into visual information by rendering what a user would perceive if he or she were to be capable of electrosense, and also render virtual fixtures that indicate potentially information-rich locations.

2. Setup

2.1. Hardware and communications

The setup for augmenting human sensing with active electrosense consists of four distinct components (figure 2): the SensorPod which is attached to a 4-DoF gantry (detailed in [3]), as well as MATLAB and Unity applications, and the Oculus Rift HMD.

For this application only the middle sensing pair labeled S4 on the SensorPod is used (figure 1). The gantry talks to MATLAB via TCP/IP. MATLAB determines which virtual fixtures to render, and sends this information to Unity via UDP. Unity then renders the appropriate information in Oculus Rift. Users can respond to the information and give a control input (using arrow keys) for the gantry position; this information is passed back to the gantry to update the physical position of the SensorPod.

The gantry is placed over a tank of water with dimensions of 234 cm wide $\times$ 175 cm long $\times$ 90 cm high. Programmed position limits restrict the motion of the SensorPod to a smaller 153 cm $\times$ 85 cm workspace to maintain distance from the walls and bottom of the tank. The water height is 46 cm and the conductivity of the water is 0.04 S m^{-1}. At run time, the SensorPod moves at a speed of 2 cm s^{-1} to minimize surface water waves that distort the electrosense signal. The SensorPod is able to freely move in the xy-plane within the soft limits but is constrained so that its orientation is always 0° with respect to the x-axis, and such that the center of the SensorPod is at a depth roughly 17 cm from the bottom of the tank.

2.2. VR interface

The user interface (SensorPodVR) consists of a model of the SensorPod, a scene of a plane indicating the reachable space, and various fixtures rendered on the plane that help visualize the relevant properties of the electrosensory scene (figure 3). Users control the SensorPod using the arrow keys and the SensorPod then automatically gathers electrosensory data at each position. The rendering is done at 75 FPS (frames per second), although MATLAB updates the position and electrosense information at 40 Hz.

In figure 3, the dark blue surface, or *tank area*, corresponds to the workspace in the physical tank. Movement of the SensorPod is confined to this surface. On top of the surface, a colored trail consisting of square segments varying from blue (0 mV) to red (3+ mV) is rendered as the center of the SensorPod passes over the corresponding position in reality. The redder the segment, the greater the absolute value differential voltage signal is at that position. Trail segments correspond to a 1 cm $\times$ 1 cm square of the workspace in reality, and thus an electrosense reading will be associated with a unique position in the tank. The tank area is composed of 153 such segments in the x direction, and 83 segments in the y direction. This granularity was found to be most suitable as it provided enough detail while simultaneously maintaining performance.

The SensorPod is rendered as a transparent capsule-like object. The camera position is locked slightly behind the SensorPod but can freely rotate depending on the orientation of the Oculus Rift. As the user navigates around the tank area, Unity will render two types of virtual fixtures. Cylindrical objects indicate the computer's current target position estimate(s). A greater amount of transparency corresponds to a greater degree of uncertainty. Green units hint at locations where the information on target location is expected to be higher, which in our case correspond to regions of high voltage perturbation (unlike Fisher

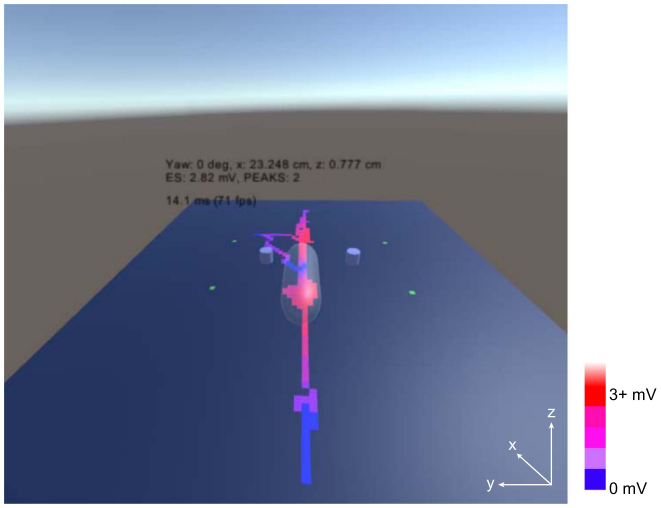

Figure 3. Sample of application view in SensorPodVR. Users control the transparent capsule like object representing the SensorPod in the physical world, and trace out a path where the redder the path, the bigger the voltage perturbation is at that point. Various dots and cylinders (virtual fixtures) describe the potential positions of large voltage perturbations and target objects, respectively.

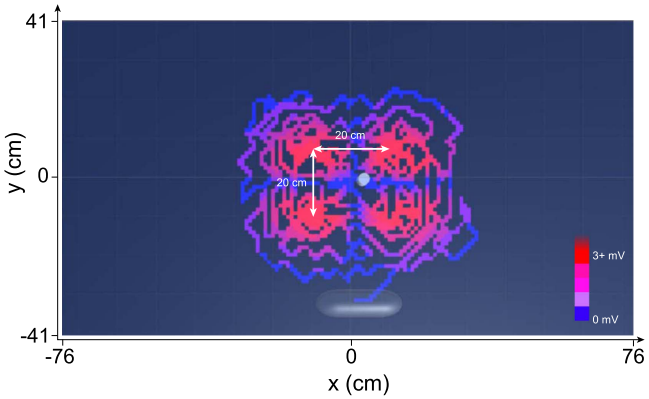

Figure 4. Electric image of target object in virtual reality application, as seen from a top-down view. We sample the absolute value of the voltage perturbations. The center of the four regions of high voltage perturbation (red) are spaced roughly 20 cm apart from each other in the x and y directions. The cylindrical target is in the center of the electric image, at a coordinate of little voltage perturbation (blue). A model of the SensorPod (transparent capsule on bottom) is shown for scale in relation to the target in virtual reality.

information maximization as used in [4]). Fixtures disappear and update according to the data gathered.

We fix the orientation of the SensorPod to be 0° with respect to the x-axis as the electric image of the target is affected by orientation. Only one target is localized at a time. The target is an aluminum cylinder with a height and diameter of 7.62 cm and is placed in a smaller zone $-50 \leqslant x \leqslant 50$ cm, $-30 \leqslant y \leqslant 30$ cm around the origin. The robot is restricted to moving in a plane above the cylinder to avoid collisions.

3. Algorithm for object localization

3.1. Electric image of the target object

The electric image of the aluminum cylinder target in 2D space is shown in figure 4. The target (center, white) is surrounded by four reddish regions, or *peaks* of high voltage differential. These regions are generated when the SensorPod moves over each grid and records the differential voltage reading at that (x, y) position in the tank area. Note that the target object, however, is located in an area of low differential voltage reading. With the SensorPod directly over the object, the electric field on both sides of the SensorPod is equally perturbed, and the difference in voltage is therefore 0. This is one aspect of differential mode electrosense that is unintuitive for humans to visualize —it conflicts with our implicit assumption that moving our sense organs closer to a target will increase the quality of information. We use the geometry of the electric image extensively to compute the target's likely positions.

3.2. Overview of algorithm for peak localization

Algorithm 1 uses the position and number of the peaks to locate the object. While the program SensorPodVR is running, the algorithm in MATLAB repeatedly

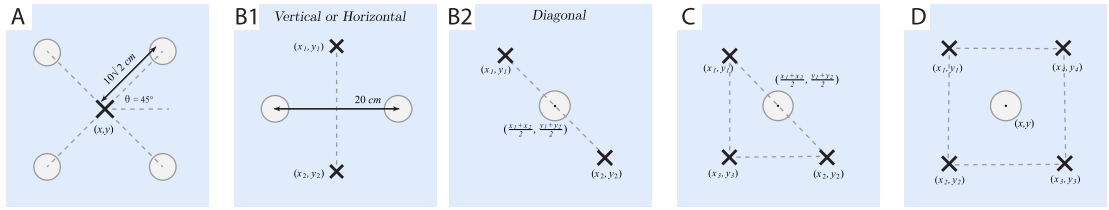

Figure 5. Possible positions of the target object (white circles) based on the information from the number of peaks discovered (crosses). (a) When one peak has been discovered, there are four possible positions for the target object. (b) With two peaks discovered, the algorithm will first determine whether the alignment is horizontal or vertical (b1), or diagonal (b2). (c) With three peaks found, the algorithm estimates the target position by taking the midpoint of the two peaks with the longest distance between them. (d) With four peaks, the algorithm calculates the convex hull of the peaks and estimates the target position at the centroid.

interpolates the existing information at each visited (x, y) point using natural neighbor interpolation (MATLAB's `scatteredInterpolant` function). The more points the SensorPod has visited and the more data it collected, the more accurate the interpolation will be. The algorithm looks for potential peaks at a position (x, y) at the visited sites by noting all grid units with the following properties:

(i) Absolute value of voltage at $(x, y) \geqslant 3$ mV.

(ii) Surrounded by at least 3 visited grid-units with voltage magnitudes smaller than or equal to the potential peak's units.

(iii) (x, y) is not on the border of the tank area. That is, 2 cm $< x <$ 152 cm and 2 cm $< y <$ 82 cm. Hereafter, all spatial units are cm unless otherwise noted.

The potential peak values and their positions are stored in an array and clustered using agglomerative hierarchical clustering (MATLAB's `cluster` function). Clusters are formed based on the Euclidean distance between the points. The computer then designates the maximum in each cluster as the true peak. The next section details the switch cases used to determine target positions based on the number of peaks found.

Algorithm 1. Locate objects and peaks.

function LOCATEOBJECTSANDPEAKS(*TankMatrix*)
  Interpolate sparse map of *TankMatrix*
  Locate peaks in interpolated map using hierarchical clustering
  if number of peaks > 0 **then**
    switch number of peaks **do**
      case 1
        Generate four possible positions symmetrically about peak
      case 2
        Determine alignment of peaks (horizontal, vertical, or diagonal)
        Generate positions depending on alignment
      case 3
        Find midpoint of peaks associated with longest distance
      case 4
        Get convex hull of peaks
        Find centroid of convex hull
(Continued.)
      case *default*
        No action
  return estimated positions, number of peaks

3.3. Switch cases

Figure 5 depict the various peaks (crosses) found and how they are used in the `switch` case. The number of peaks found determine the number of generated cylinders (circles) at estimated positions.

3.3.1. One peak

When the user has discovered one peak, there are four possible locations for the 7.62 cm diameter, 7.62 cm tall aluminum cylinder.

$$(x_{\text{target}}, y_{\text{target}}) = \begin{cases} (x - 10, y - 10), \\ (x + 10, y - 10), \\ (x + 10, y + 10), \\ (x - 10, y + 10). \end{cases} \tag{3.1}$$

3.3.2. Two peaks

When the user has discovered two peaks, there are a few scenarios depending on how the peaks are aligned. Alignment is determined by the ratio of x and y distances between the peaks. A deviation of roughly 11° from horizontal or vertical will be counted as horizontal or vertical, respectively, and a deviation of only ±1.4° from diagonal will be counted as diagonal. This strict adherence to 45° was created because too many pairs of peaks would otherwise be counted as diagonal.

$$\begin{cases} \dfrac{|x_1 - x_2|}{|y_1 - y_2|} < 0.2, & \text{horizontal}, \\ \dfrac{|y_1 - y_2|}{|x_1 - x_2|} < 0.2, & \text{vertical}, \\ 0.95 < \dfrac{|y_1 - y_2|}{|x_1 - x_2|} < 1.05, & \text{diagonal}. \end{cases} \tag{3.2}$$

When the alignment is horizontal or vertical, the algorithm first finds the midpoint of the two peaks. When the alignment is diagonal, the midpoint is a good approximation for the position.

$$(x_{\text{target}}, y_{\text{target}}) = \begin{cases} (x_{\text{mp}} + 10, y_{\text{mp}}, x_{\text{mp}} - 10, y_{\text{mp}}), & \text{horizontal,} \\ (x_{\text{mp}} + 10, y_{\text{mp}}, x_{\text{mp}} - 10, y_{\text{mp}}), & \text{vertical,} \\ (x_{\text{mp}}, y_{\text{mp}}) = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}\right), & \text{diagonal.} \end{cases} \quad (3.3)$$

3.3.3. Three and four peaks

When the user has discovered three or four peaks, it is likely that the true position of the target is not more than a few centimeters from the estimated position. With three peaks, the algorithm will search for the longest distance between two of the three peaks and take the midpoint of the points involved in the longest distance.

When the user has discovered four peaks, the computer will find the convex hull using MATLAB's `convhull` function, and take the estimated target position to be the centroid of the convex hull. Formally, the convex hull is described as the set of points $\boldsymbol{p_i}$ in the set S such that:

$$S = \{\sum_i \lambda_i \, \boldsymbol{p_i} \, | \sum \lambda_i = 1, \, \lambda_i \geqslant 0\}.$$

The convex hull can be thought of the enclosure created by stretching a rubber band around the set of points S. The convex hull function orders the points properly so that the centroid (C_x, C_y) of the quadrilateral can be calculated:

$$C_x = \frac{1}{6A} \sum_{i=0}^{n-1} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i), \quad (3.4)$$

$$C_y = \frac{1}{6A} \sum_{i=0}^{n-1} (y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i), \quad (3.5)$$

$$A = \frac{1}{2} \sum_{i=0}^{n-1} (x_i y_{i+1} - x_{i+1} y_i), \quad (3.6)$$

where A is the signed area.

3.4. Indicating regions of high perturbation

Originally virtual fixtures consisted only of cylinders rendered at their estimated positions. However, it was found that this was confusing for users because navigating to the position of the cylinder did not yield any additional useful information. In order to aid intuition we also render green squares on the floor of the tank area to denote possible peak locations. These squares serve as hints that guide the user to a region of high information density, which will improve the algorithm's estimate of the target position. If the target's estimated position is centered at (x, y), units are rendered at $(x - 10, y - 10)$, $(x + 10, y - 10)$, $(x + 10, y + 10)$, $(x - 10, y + 10)$.

4. System–user interaction

We perform hypothesis testing to determine the usefulness of virtual fixtures (green square hints) in aiding users to localize objects. *Usefulness* is measured by two metrics: how long it takes users to localize the object, and how accurate the localization is. The null hypothesis assumes that there is no difference in performance with the use of virtual fixtures.

4.1. Experimental setup

Ten students from our laboratory tested SensorPodVR. Users were given a set of instructions, briefed on the basics of electrosense, and shown the electric image in figure 4. This was done to reduce the variance in performance, as some users were more familiar with active electrosense than others. All users attempted to localize the target using two versions of the application —one with hints, and one without (control case). In the control group, users attempted to localize the object by using only the colored trail, while in the experimental group, users were able to rely on both virtual fixtures and the colored trail. The order chosen for the two versions for each user was randomized. This was because we did not want familiarity with the system to shift the mean of the time taken for localization. Half of the users ran the no-hints version of the application first, and half of the users ran the hints version first.

The SensorPod starts at the origin (0, 0), and the target is placed in a random position in the zone $-50 \leqslant x \leqslant 50$ cm, $-30 \leqslant y \leqslant 30$ cm around the origin. The timer starts when the user begins to move. When the user has determined that he or she has localized the target, the user would position the center of the SensorPod directly over the guessed coordinates, and the timer would stop. Because there is no visual grid overlaid with the tank area, to compare estimated position with the actual coordinates, we moved the SensorPod over the object by looking at the physical position of the object and took the difference. The value obtained is the accuracy measurement (error in distance).

4.2. Results and discussion

A total of 10 trials were taken for each version. We looked at the distribution of the data using a Lilliefors test and if the distribution was close to normal, we ran a two-sample t-test to test the hypothesis. If the distribution was not normal, a ranked-sum test was used. Results showed that virtual fixtures helped reduce the overall time it took to localize an object down by 44.5% on average but that there was no difference in accuracy (see tables 1 and 2). Target position estimates were within 4 cm of the physical position when excluding outliers, which is about half the body length of the object (7.62 cm diameter). This is an acceptable error, as it indicates that the majority

Table 1. Table of mean, standard deviations for time versus accuracy data in figure 6, with outliers. We can see that with outliers, the accuracy with hints appear to be, on average, worse than the accuracy without hints. A ranked sum test was computed for accuracy as the outliers cause the distribution to have a heavy tail. This is one drawback of using hints—users can be misled by the computer's estimate.

	Time (min)		Accuracy (cm)	
	Hints	No hints	Hints	No hints
μ	2.28	4.11	3.91	2.22
σ	0.76	1.37	4.01	1.06
p	0.0016 (t-test)		0.52 (ranked sum)	

Table 2. Table of mean, standard deviations for time versus accuracy data in figure 6, with no outliers. Once we remove the outliers we can see that using hints or no hints does not change the accuracy.

	Time (min)		Accuracy (cm)	
	Hints	No hints	Hints	No hints
μ	2.21	4.08	2.11	2.10
σ	0.74	1.51	0.88	1.11
p	0.0072 (t-test)		0.98 (t-test)	

of attempts at centering the SensorPod over the face of the cylinder are successful. Figure 6 shows how time taken does not correlate with the distance error for fixture and no fixture versions. This may be because users feel more certain of their own estimates when guided by the computer's estimate, or because virtual fixtures outline the possible regions of high information density so that users do not have to guess their locations.

However, if the user has collected an insufficient amount of data, the computer's interpolation will be off and may mislead the user by giving a wrong estimate of position. In the figure, there are two outliers that have a relatively low accuracy, and it is likely that during these trials the users prematurely determined the position of the target after collecting only a small set of data. This is one disadvantage of using guides—the error of the interpolation is strongly tied to the number of coordinates visited, so users can be misled if they are too hasty in determining the object's location. Removing the outliers does not change the aforementioned conclusions.

Table 3. Some sources of lag in SensorPodVR.

Estimated task durations during runtime (ms)	
Unity frame refresh (FPS^{-1})	13
Data gathering rate (voltage and position)	30
Communications (UDP)	4
Interpolation during Localization algorithm	$O(n^2)$, max 200
Input–output delay	1000

5. System performance

5.1. Latency

Table 3 lists the various sources of lag. Delays due to the data gathering rate and communications were calculated by averaging the time taken to loop the same task 1000 times. Note that we gather voltage data every other loop, and interpolate every 20th loop so the calculated duration is only the average. Calculations for the lag due to interpolation during the object localization algorithm is described in the next paragraph. Input–output lag was calculated by filming a slow-motion capture of an operator controlling the SensorPod and recording the time between user input and SensorPod movement. Of all of these sources of lag, aside from the input–output lag, it was found that interpolation during the object localization algorithm

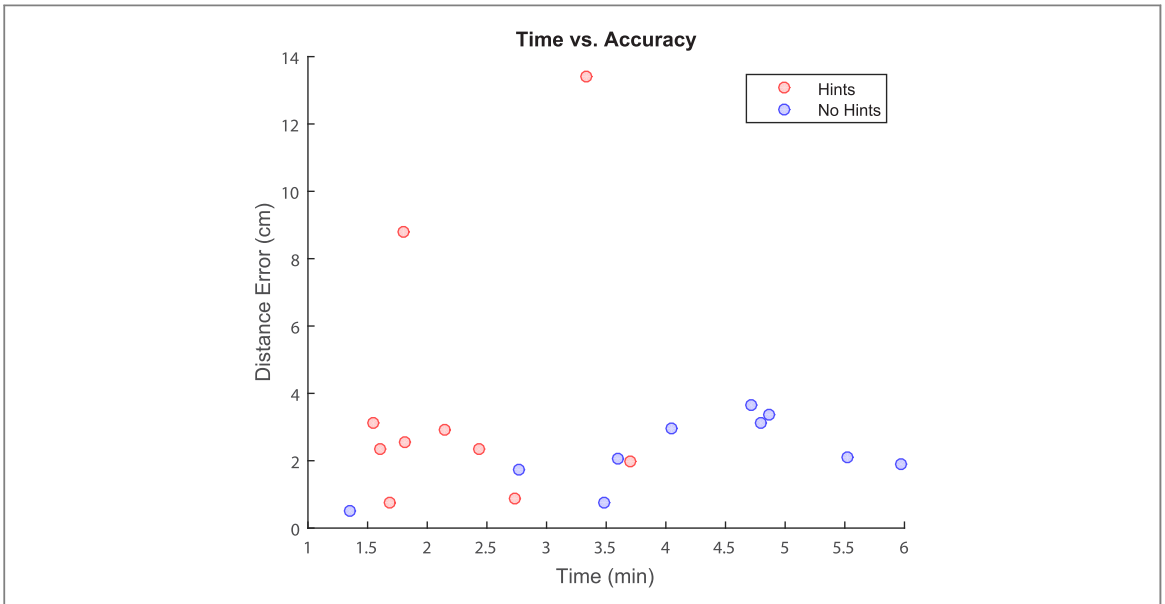

Figure 6. Plot of the amount of time users took to localize an object compared to how close they were to its physical position. Note that most guesses are within 4 cm of error, which is a little more than the radius of the cylinder (3.81 cm). Thus most attempts are successful in centering the SensorPod over the cylinder.

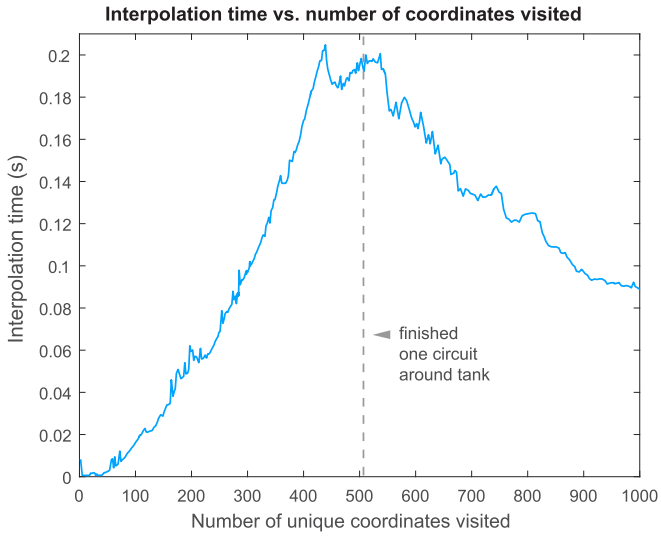

Figure 7. Interpolation is only performed once every 20 iterations of the program's main `while` loop. However, as the number of visited positions grows, the time to interpolate increases sharply and causes a noticeable amount of lag in the application. Note that the interpolation time grows as the square of the number of coordinates visited, but then decreases as the SensorPod begins to fill in the area enclosed by its circuit.

was the culprit for the biggest percentage of time taken.

For this interpolation, the worst case time complexity is $O(n^2)$, which occurs when the SensorPod travels around the perimeter of tank area (figure 7). This is because the computer needs to interpolate over the convex hull of the visited points, and so both the size of the convex hull and the number of visited coordinates influence the time needed. During a typical run in which an object is localized within 500 unique visited coordinates, the main `while` loop in MATLAB runs on average at 40 Hz. Other sources of lag do not tend to increase the longer the application runs.

Although the lag between user input and SensorPod movement is large, users did not report feeling disoriented. This may be because this lag does not affect Unity's frame refresh rate so the VE still responds to the user's head orientation sufficiently fast. Some users commented that the lag made it difficult to center the SensorPod correctly over the target towards the end of the trial, which may have resulted in positioning errors of 1–2 cm. However, this error is well below the acceptable error of 4 cm and thus do not affect the conclusion that users benefit from the use of virtual fixtures.

6. Noise and distortion of the electric field

In a real system there will always be some sort of noise or non-ideal factor that decreases the precision of the instrument. This is especially true for the SensorPod as it relies on a differential voltage reading. Any noise or distortion that causes asymmetry in the electric field can be picked up as a voltage difference by the differential op-amps across the sensor pair. Various sources of noise include: electrical noise due to the sensor circuitry, voltage offsets due to surface waves at the air–water boundary, noise due to turbulence and inhomogeneous resistivity of water, and distortions in the electric field due to tank walls.

All noise factors come together and result in a drifting voltage offset. This offset is the *baseline*, or the electric image of the tank at a fixed depth without any objects in the water. To reduce offsets due to the baseline, the SensorPod periodically sweeps through a series of 10 random straight-line trajectories in the tank and samples the current position and the perceived voltage differential at each point in the trajectory. This information is interpolated to create a map of the baseline at each point in the workspace (figure 8). Corner points of the reachable workspace are always included in these trajectories. This baseline voltage is subtracted from the readings we obtain during the runtime of SensorPodVR and is recalculated every 2–3 h to counteract the effects of drift. As the baseline cannot be calculated while running the experiment, we also use the VR application to compare the trail color of various points in the workspace before and after each localization attempt to ensure that the offset has not drastically changed.

Finally, as mentioned earlier, we restrict the placement of the target object in a smaller zone ($-50 \leqslant x \leqslant 50$ cm, $-30 \leqslant y \leqslant 30$ cm) centered around the origin of the tank as our algorithm does not yet address the effects of electric field distortion near tank walls [3, 16].

7. Conclusions

We have demonstrated the feasibility of operating the SensorPod using a VE to localize a target object with a

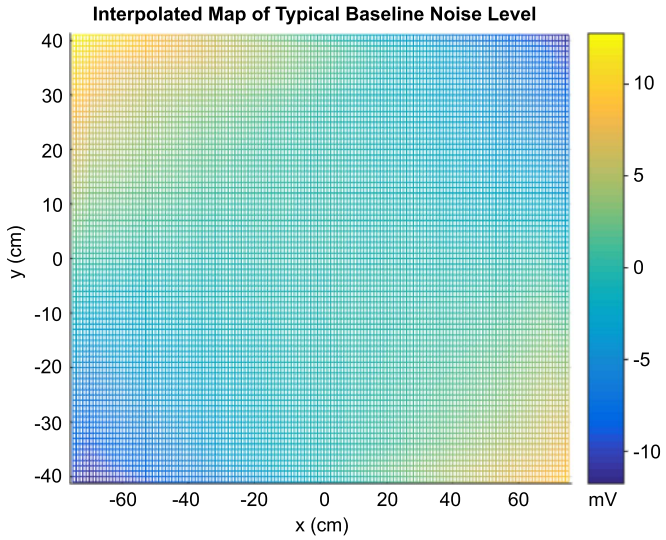

Figure 8. The interpolated map of the baseline (no object differential voltage readings). The corners have significantly larger voltage differences than the center of the tank. For this reason, we restrict object localization to the center.

specific set of properties. By visualizing electrosense information from the target as well as virtual fixtures for guidance, SensorPodVR helps users effectively respond to information from unintuitive types of sensing modalities and decrease the time needed for localization via human-based computation. By combining user intuition with machine computation, SensorPodVR has proven to be an effective early-stage teleoperation system for localization using electrosense.

As SensorPodVR is still in its early stages of development, there are a few directions one could take its development. An immediate next step would be to improve the electrolocation algorithm. For simplicity, this demonstration and the corresponding algorithm assumes a fixed target shape and material; however, our prior work on estimating shape, size, distance, and material of targets could be integrated to make the algorithm more generalized and robust [3, 13]. In addition to providing the user with hints for discrete locations where information is highest within an electrosensory scene, we could hint at the optimal path to sample the space proportional to its expected information density by integrating ergodicity [4].

Acknowledgments

This work was supported by NSF Grant IIS 1427419.

References

[1] Whitcomb L L, Yoerger D R, Singh H and Howland J 2000 Advances in underwater robot vehicles for deep ocean exploration: navigation, control, and survey operations *Robotics Research: The Ninth Int. Symp.* ed J Hollerbach and D Koditschek (London: Springer-Verlag) pp 439–48

[2] Lissmann H W and Machin K E 1958 The mechanism of object location in gymnarchus niloticus and similar fish *J. Exp. Biol.* **35** 451–86

[3] Bai Y, Snyder J B, Peshkin M A and MacIver M A 2015 Finding and identifying simple objects underwater with active electrosense *Int. J. Robot. Res.* **34** 1255–77

[4] Miller L M, Silverman Y, MacIver M A and Murphey T D 2016 Ergodic exploration of distributed information *Trans. Robot.* **32** 36–52

[5] Neveln I D, Bai Y, Snyder J B, Solberg J R, Curet O M, Lynch K M and MacIver M A 2013 Biomimetic and bio-inspired robotics in electric fish research *J. Exp. Biol.* **216** 2501–14

[6] MacIver M A and Nelson M E 2001 Towards a biorobotic electrosensory system *Auton. Robots* **11** 263–6

[7] Boyer F, Gossiaux P B, Jawad B, Lebastard V and Porez M 2012 Model for a sensor inspired by electric fish *IEEE Trans. Robot.* **28** 492–505

[8] Alamir M, Omar O, Servagent N, Girin A, Bellemain P, Lebastard V, Gossiaux P B, Boyer F and Bouvier S 2010 On solving inverse problems for electric fish like robots *2010 IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)* (Piscataway, NJ: IEEE) pp 1081–86

[9] Lebastard V, Chevallereau C, Amrouche A, Jawad B, Girin A, Boyer F and Gossiaux P B 2010 Underwater robot navigation around a sphere using electrolocation sense and Kalman filter *2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (Piscataway, NJ: IEEE) pp 4225–30

[10] Lebastard V, Chevallereau C, Girin A, Servagent N, Gossiaux P B and Boyer F 2013 Environment reconstruction and navigation with electric sense based on a kalman filter *Int. J. Robot. Res.* **32** 172–88

[11] Ammari H, Boulier T, Garnier J and Wang H 2014 Shape recognition and classification in electro-sensing *Proc. Natl Acad. Sci.* **111** 11652–7

[12] Bai Y, Snyder J, Silverman Y, Peshkin M A and MacIver M A 2012 Sensing capacitance of underwater objects in bio-inspired electrosense *2012 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (Piscataway, NJ: IEEE) pp 1467–72

[13] Bai Y, Neveln I D, Peshkin M and MacIver M A Enhanced detection performance in electrosense through capacitive sensing *Bioinspir. Biomim.* **11** 055001

[14] Solberg J R, Lynch K M and MacIver M A 2008 Active electrolocation for underwater target localization *Int. J. Robot. Res.* **27** 529–48

[15] Nguyen N, Wiegand I and Jones D L 2009 Sparse beamforming for active underwater electrolocation *IEEE Int. Conf. on*

Acoustics, Speech and Signal Processing, 2009, ICASSP 2009 (Piscataway, NJ: IEEE) pp 2033–36
[16] Silverman Y, Bai Y, Snyder J B and MacIver M A 2012 Location and orientation estimation with an electrosensing robot *2012 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (Piscataway, NJ: IEEE) pp 4218–23
[17] Smith J R, Garcia E, Wistort R and Krishnamoorthy G 2007 Electric field imaging pretouch for robotic graspers *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2007, IROS 2007* (Piscataway, NJ: IEEE) pp 676–83
[18] Uhlmann G 2009 Electrical impedance tomography and calderón's problem *Inverse Problems* **25** 123011
[19] Chalmers D J 2016 The virtual and the real *Disputatio* at press
[20] Burdea G C 1999 Invited review: the synergy between virtual reality and robotics *IEEE Trans. Robot. Autom.* **15** 400–10
[21] Jankowski J and Grabowski A 2015 Usability evaluation of vr interface for mobile robot teleoperation *Int. J. Hum.–Comput. Interact.* **31** 882–9
[22] Lin Q and Kuo C 1997 Virtual tele-operation of underwater robots *Proc., 1997 IEEE Int. Conf. on Robotics and Automation, 1997* vol 2 (Piscataway, NJ: IEEE) pp1022–27
[23] Bejczy A K, Kim W S and Venema S C 1990 The phantom robot: predictive displays for teleoperation with time delay *Proc.1990 IEEE Int. Conf. on Robotics and Automation 1990* (Piscataway, NJ: IEEE) pp 546–51
[24] Rosenberg L B 1993 Virtual fixtures: Perceptual tools for telerobotic manipulation *1993 IEEE Virtual Reality Annual Int. Symp., 1993* (Piscataway, NJ: IEEE) pp 76–82