Bat-Inspired Robot Navigation

Michael J. Kuhlman
Dept. of Electrical and
Computer Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY 12180
Email: kuhlmm@rpi.edu

Kate McRoberts
Dept. of Electrical and
Computer Engineering
Grove City College
Grove City, PA 16127
Email: mcrobertskm1@gcc.edu

Advisors:
T. K. Horiuchi & P. S. Krishnaprasad
Dept. of Electrical and Computer Engineering
Institute for Systems Research
University of Maryland
College Park, MD 20742

***Abstract*—A key objective of Robotics is the autonomous navigation of mobile robots through an obstacle field. Inspired by echolocating bats, we developed a two-part navigation system consisting of obstacle detection through echolocation and motion planning. The first part relies upon a binaural sonar system, which emits ultrasonic pulses and then determines the interaural level difference (ILD) of the returning echoes to infer obstacle locations. Next, the *Openspace* motion planner computes the best direction of travel based on the locations of the target and the detected obstacles. We implemented this navigation system on a mobile platform, which repeatedly computes the safest direction of travel and moves accordingly, ultimately generating a real-time path to the goal.**

I. Introduction

Bats' seemingly effortless navigation abilities have long fascinated scientists. Imagine swiftly navigating a dense forest at night with only mediocre vision. Echolocation makes this possible for bats. Inspired by this navigation method, we designed and implemented a system that enabled a robot to exhibit obstacle avoidance using echolocation. While we were provided with a sonar device to mimic echolocation, our challenge was to process the signal data to determine information about the environment (such as locations of obstacles), and use this in conjunction with the *Openspace* motion planning algorithm to direct a robot's movement, thereby developing a path to a goal in real time.

II. Echolocation and Sound Localization Cues

Using binaural sound localization to detect objects with echolocation, information about an object location lies in the differences between echoes in the left and right microphones. Absolute qualities can vary greatly based on the object size, material and geometry. Since none of this is of concern to navigate around obstacles (we do not care whether we detect a PVC pipe, or a wall, just as long as we can avoid it), we opt to cross-compare channels by studying contrast and time differences instead of absolute qualities.

The sonar system's speaker emits a 40kHz pulse that reflects off objects in the world, and similar but different echoes return to the two microphones, which are angled 45 degrees. Since the distance to the further ear is greater than that to the closer ear, there is a time difference in which the sonar signals arrive at the two different microphones, called the interaural time difference (ITD). Fig. 1 demonstrates geometrically how this difference arises.

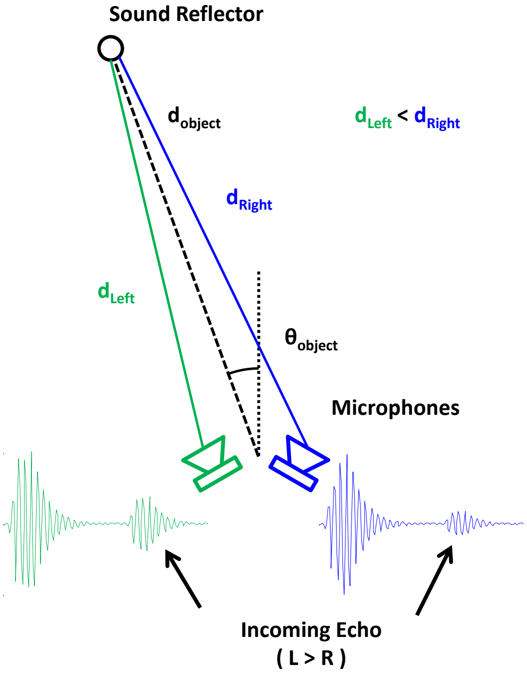


Fig. 1. Visualization of properties of the binaural sonar system which result in sound localization cues.

While ITD is a very reliable metric for computing azimuth angle for larger interaural distances (such as in human heads), the two microphones in our sonar system and the respective ears on bats are too close for this metric to be effective. The overwhelming majority of bats use the Interaural level difference (ILD) to compare channels as in (1). Three phenomena factor into ILD in varying amounts. Most microphones naturally have directional filtering (sounds in the center are often louder than sounds on the periphery). Acoustical shadowing also occurs when lossy material (such as the head) is placed between the two microphones/ears. This phenomenon is more pronounced at the higher frequencies used in bat echolocation. The aforementioned difference in distance from the microphones to the sound source also contributes to variations in ILD, since sound intensity falls off with an inverse squared

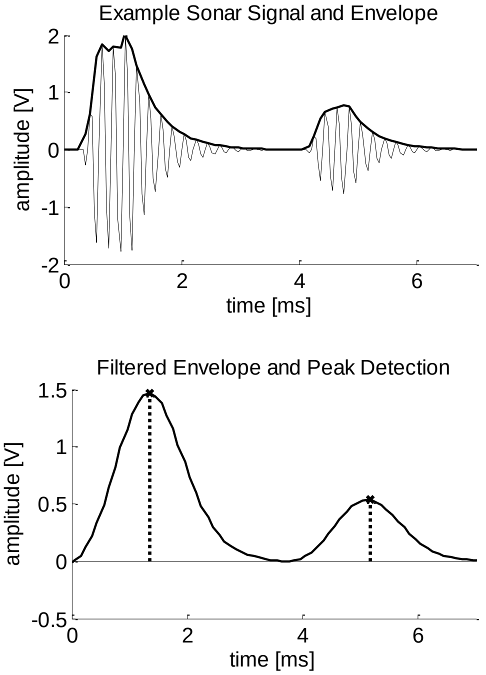


Fig. 2. Demonstration of signal processing on sample sonar signal. The sonar board detects the envelope of the signal using an Op-Amp based circuit. This is sent to MATLAB for filtering and peak detection for object detection.

relationship to distance of the target. The summation of these phenomena which factor into ILD make a direct mapping from ILD values to angles complicated, especially when using a constant frequency pulse. For further reading on how *E. Fucus* uses ILD for sound localization, please refer to [1].

$$ILD_{LR} = 20 \log \left(\frac{L}{R}\right) \tag{1}$$

III. Signal Processing and Obstacle Detection

The sonar signal envelope peaks are salient features that can be cross-compared across channels regardless of time distortion due to ITD. Therefore, peak detection is the most important feature extraction for object detection. The system uses a 35th order FIR filter with a cutoff frequency of 10% of the sampling frequency (i.e. normalized frequency), using a Hamming window. Different values for the order and the cutoff frequency of the filter do not significantly effect the accuracy of the obstacle detection method.

Next, the system takes the derivative of the filtered envelope and searches for sign changes (from positive to negative), which correspond to envelope peaks, recording both peak amplitudes and times for further analysis. This process is summarized visually in Fig. 2 and makes peak detection a simple, reliable metric for determining outgoing pulse and echo locations. If multiple peaks occur for the same object, these ambiguities will be resolved when solving for correspondence. A small absolute threshold for peak detection above the quantization error was also be used to remove peaks due to noise.

A. Correspondence and Distance Metrics

Given left and right channel signals from the microphones, one must deduce which echoes correspond to which real world objects. Depending on angle, echoes will be detected in one or two of the channels. We assume that echo peaks within 10 samples (0.3 ms at 25 kHz) of each other originate from the same real world object.

When determining an object's location, we decompose its location into distance and angle relative to the sonar system. Since the microphones pick up the initial outgoing pulse generated by the speaker, we are able to estimate object distance accurately; the time between the peaks of the outgoing pulse and object echo is the time in which sound traveled to and from the object.

B. Michelson Contrast and ILD

We can infer the azimuth angle of the object due of the difference in peak amplitudes in the left and right channels. This relationship is commonly encoded with (1), where L and R are the corresponding echo peak amplitudes in the left and right channel for a given object. However, (1) can become complex when the inhibitory channel $R = 0$, which is possible with quantization. The use of Michelson Contrast (2) as our metric for calibration alleviated the singularities possible with ILD. ILD and Michelson Contrast are similar in behavior. Using series expansion of $\ln(z)$, one can obtain the first order approximation (3) of ILD, which is similar to (2).

$$contrast_{LR} = \frac{L-R}{L+R} \tag{2}$$

$$ILD_{LR} = 20 \log \left(\frac{L}{R}\right) \approx \frac{80}{\ln(10)} \left(\frac{L-R}{LR+L+R+1}\right) \tag{3}$$

C. Calibration

Once we compute the angle metric for given object data, the next step is to determine the corresponding angle. Since the nature of sound propagation is very complex, we opt for one-dimensional calibration instead of modeling to map metric values to object angles. We thus constructed a calibration data set with 140 data points with a single object ranging from radii $r = 1$ft to $r = 2.5$ft in .25 ft intervals every 5° from $\pm 45^\circ$ from the center axis of the sonar system. After computing the Michelson Contrast for each trial, we mapped object angles to metric values, creating ordered pairs in the region of monotonicity as in Fig. 3. For experimental data, one could use (inverse) interpolation to map experimental metrics to estimated angles. To verify the accuracy of the calibration, we then estimated object angles in the calibration data set using Michelson Contrast and the calibration data. 76% of all trials estimated the object's angle within 5° (i.e. one point of resolution) of its actual angle.

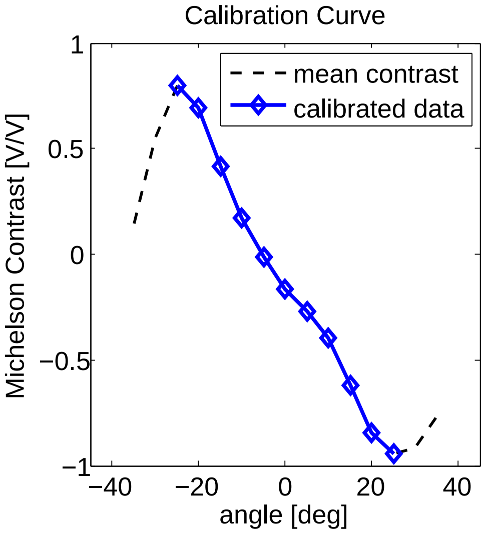


Fig. 3. Calibration data mapping Michelson Contrast values to angles. Only the region of monotonicity can be calibrated, so the dashed portion of the curve is truncated for calibration

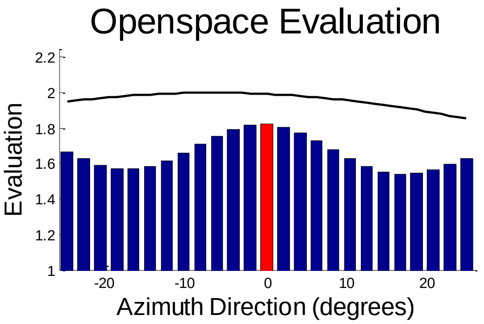


Fig. 4. Example evaluation function from the MATLAB simulation. Zero degrees is the determined winner-take-all direction for this time step. The evaluation before taking the obstacles into account is shown by the black line above the bars. From the figure, we can see that the goal is approximately located in the direction of $-6°$. We can also see that obstacles to the left and right of the bat suppress the evaluation.

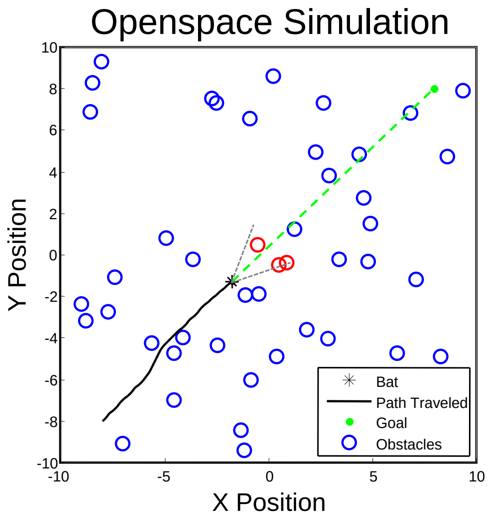


Fig. 5. MATLAB simulation of a bat navigating an obstacle field using the *Openspace* evaluation. The red obstacles represent those that are of concern to the bat and therefore affect the evaluation function.

IV. Openspace

The *Openspace* motion planning algorithm was used as the robot's navigation method. One distinguishing feature of this algorithm is that it does not require foreknowledge of the whole environment, so it works well in real time. The sonar based sensor system is suitable for *Openspace*, because *Openspace* only require information about impeding obstacles. The obstacles of concern are those that obstruct the robot's path according to the capabilities of the sonar device and requirements of the mobile platform.

The *Openspace* evaluation function, (4) from [2] evaluates the desirability of moving in any one direction based on the location of the destination and the locations of impeding obstacles. This evaluation is computed for each of a discrete set of directions, which, in real time, is determined by hardware limitations. An azimuth range and resolution define the direction set. For example, our sonar device can properly map ILD to azimuth angles within a range of $\pm 25°$(with $0°$ being straight ahead). Since our sonar system has a low resolution, we broke up this range into one-degree intervals for a total of 51 directions.

Three terms constitute (4). The first term is a constant bias so that all evaluations are positive. The second term is a Gaussian centered about the direction in which the destination is located (so the direction of the goal starts with a higher evaluation than any other direction). Since we did not have a global positioning system (GPS) and thus could not have cartesian coordinates for the goal, we opted to have the robot "wander," forcing the goal angle $\theta = 0$, or to drive straight until an impeding object forces the robot to turn. The third term is a summation of Gaussians. Each one of these Gaussians corresponds to an impeding obstacle. Because the third term is subtractive, each obstacle detracts from the desirability of moving in that direction. Each obstacle suppresses the evaluation with strength inversely proportional to its distance away, so closer objects create a deeper and wider suppression. Finally, the direction with the highest evaluation is selected as the best direction of travel, or is the winner-take-all (WTA) selection. A path develops over repeated *Openspace* evaluations and corresponding movements. Fig. 4 depicts an example evaluation function with its selected WTA in the environment depicted in Fig. 5.

We created a simulation of *Openspace* in MATLAB to understand the algorithm and how various parameters affect its performance. Furthermore, having a working simulation smoothed the transition to using real-time data and eventually a mobile platform. Fig. 5 shows a MATLAB simulation of a bat navigating an obstacle field, such as a forest of trees, to real its goal.

The version of the *Openspace* code suitable for real time is similar to the simulation code, but much more robust. Its inputs are the direction of the goal and the obstacle locations (given by distance and angle from the sonar device). If no goal is specified, it is assumed that the goal direction is straight ahead (zero degrees).

The evaluation function builds according to the information provided. It starts as just the constant bias term, and adds the goal steering Gaussian if a there is a goal direction (which there currently always is, since we assume zero degrees if none is specified, but future work could make use of the option) and adds the obstacle suppressions if there are obstacles. Finally, the real time *Openspace* code simply outputs the WTA angle, which is sent to the robot to change its direction accordingly.

$$E(\theta) = E_0 + g \cdot e^{\frac{-(\theta-\theta_g)^2}{{\sigma_g}^2}} - \sum_{i=1}^{N} \frac{1}{r_i} \cdot e^{\frac{-(\theta-\theta_g)^2}{\sigma(r_i)^2}} \qquad (4)$$

V. Robot Control System and Hardware

To ensure rapid development, we opted to keep the sonar signal processing and *Openspace* computation in MATLAB, which was run on a laptop which was attached to the robot. The sonar system connected to the laptop via USB, and the *Openspace*'s WTA direction was sent over a serial connection to the embedded system on the robot, a *Pioneer 3*. Lower level controls at the robot level were handled by Motion Description Language extended (MDLe), a framework suitable for hybrid controls systems and robotics [3].

Given both the nature of *Openspace* and the available functions in MDLe, we decompose the control of the robot about the world plane into forward velocity (i.e. translational velocity) and rotational velocity, also known as the unicycle model. MDLe has a built-in controller that will convert translational and turning rates into the needed wheel speeds. Since *Openspace* does not specify the translational speed of the robot, we set the robot's forward speed to a constant 0.05 m/s to ensure safe and reliable travel. When *Openspace* computes the evaluation function, the WTA direction, relative to the robot's current heading, specifies the angle error, θ_{error} in which the robot needs to turn to head in a safe direction. We thus used Proportional control, multiplying this θ_{error} by a constant K_p to become the turning rate ω_{turn} command sent to the wheel speed controller. Fig. 6 summarizes the system's overall structure.

VI. Obstacle Scenarios

Fig. 7(a): The robot is able to navigate between two obstacles. When the robot tries to go through gaps of this nature head on, oscillations occur since the echoes of the two obstacles interfere, generating a single accidentally perceived object directly ahead of the robot. This causes the robot to turn to avoid the obstacle, eventually breaking the accidental view. Note that the distance between the two objects is 0.61m.

Fig. 7(b): Two objects too close together for the robot to pass through generate enough overlapping suppression to cause the robot to turn, avoiding the gap. Note that the distance between the two objects is 0.46m and the width of the robot is 0.48m.

Fig. 7(c): Effects of multiple objects do not stack in the same manner as other motion planners. *Openspace* localizes the effects of objects in the evaluation function by using Gaussians with small enough variances. Even at the object detection level, the echoes from two objects that are close enough together are detected as a single object, further sidestepping this issue.

Fig. 7(d): The robot is able to find the largest gap and move through it. In other runs of the same obstacle setup, random perturbations in the robot's response will cause the robot to turn left first. It will then focus on the leftmost gap, which is the second largest, but just a few inches smaller than the robot's width, and it will attempt to navigate through, but will crash into obstacles. The robot does not intrinsically know if the gap is large enough to fit through. Parameter tuning or the addition of whisker-like sensors could solve this problem.

Fig. 7(e): *Openspace* can also handle random obstacle fields. If the obstacle field becomes dense, Openspace can be given maximum object distances ignoring obstacles outside a given range so that the evaluation function does not become too chaotic.

VII. Conclusion

Our results show that the robot successfully navigated many notable obstacle arrangements that highlight the capabilities of *Openspace*. We feel that *Openspace* and the sonar system are highly compatible navigation tools because each one plays to the strengths of the other. Also, we conclude that *Openspace* is an effective motion planner because of the flexibility all of the parameters provide.

One innovative discovery includes the use of the Michelson contrast, which avoids many singularity issues. Since ILD and Michelson evaluations are similar, this substitution provides practical benefits for the user. The highlight of the project was the integration of the entire system, starting with receiving raw echo envelopes and ultimately translating them into real-time robot instructions.

VIII. Future Work

Performance with the FIR filter suggests that it can be replaced with other filters. IIR filters can be lower order and thus built using fewer physical components and have a shorter time delay, suitable for harware implementation. In its current state, our system does not provide the robot with information about its location in the world frame. For this reason, future work will include incorporation of a GPS system. This would allow for goal-oriented movement instead of pure wander behavior.

A. Feedback

Currently, our only form of feedback in the system is the P controller which drives the angle error to zero. We wish to incorporate better forms of feedback to optimize motion plans generated by *Openspace*. Feedback can occur at three major levels: the obstacle detection level, the evaluation function level, and the robot controller level Object tracking could

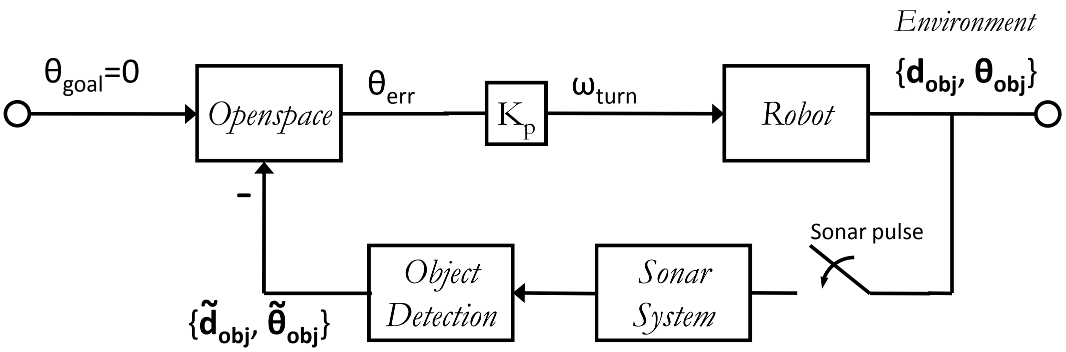
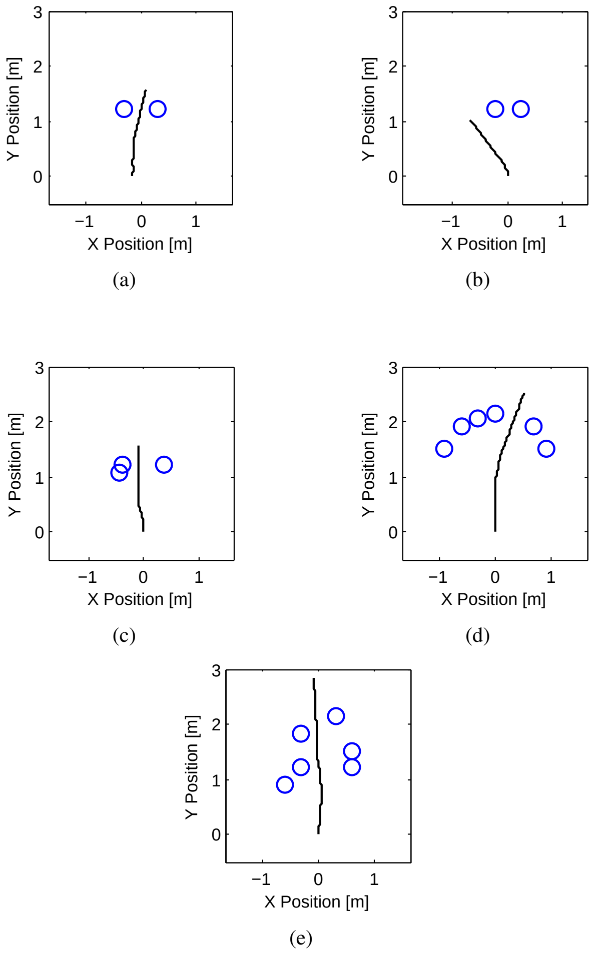


Fig. 6. System overview demonstrating the relation of the P Controller to the rest of the system. Note that the system is sampled due to active sonar, and an approximation of object locations is used in *Openspace*.



(a)

(b)

(c)

(d)

(e)

Fig. 7. Five experimental runs using the robotic platform. 7a: Robot can safely fit through obstacles. 7b: Robot correctly determines that the gap is too narrow and moves towards the outside. 7c: Additional obstacles do not affect *Openspace*. 7d: Robot finds and travels through the largest gap. 7e: Robot maneuvers random obstacle field

expand the viewing angle of the sonar system (memory of out of sight obstacles), and resolve accidental views. In other words, we expand upon the assumption of inferring the minimum number of obstacles to reconstruct the sonar envelope peaks to the minimum number of obstacles that reconstruct both previous and current sonar data. At the *Openspace* level, we also have the option of adding additional additive Gaussians centered at previous headings to dampen the system response. Classical (dynamical system) control techniques can be applied at the robot controller level, changing how the control system drives the angle error to zero, by means of adding compensators, PID control, etc. Hybrid control systems could also be of interest in handling different scenarios, such as turning on and off dampening when needed. MDLe's design encourages the use of hybrid control systems and should be of further interest.

ACKNOWLEDGMENT

The authors would like to thank Graham Alldredge, Chetan Bansal, and Matteo Mischiati for their technical advice and assistance in the laboratory.

REFERENCES

[1] Shi, R.Z. and Horiuchi, T.K., *Neuromorphic VLSI Model of Bat Interaural Level Difference Processing for Azimuthal Echolocation*,IEEE Transactions on Circuits and Systems I: Regular Papers, vol 54-1 pp74-88, 2007

[2] T.K. Horiuchi, *A Spike-Latency Model for Sonar-Based Navigation in Obstacle Fields*, Institute for Systems Research (preprint), 2008

[3] D. Hirstu-Varsakelis et. al. *A Motion Description Language for Hybrid System Programming*, Institute for Systems Research (preprint), 2003