Optimal trajectory determination of a single USV for Underwater Sensors Localization based on Range-Free Scheme

Haoran Zhao Aaron T. Becker

Abstract— **Many applications of underwater wireless sensor network require the sensor location. The main idea in most localization algorithm has been that localizing the unknown sensor nodes location with the help of landmarks with know coordinates (e.g GPS equipped beacons). A promising method to reduce the cost of localization process is to replace bunch of GPS equipped beacons with one mobile beacon. In this case, considering the coverage, time, accuracy and energy cost, path planning of mobile beacon becomes the fundamental research issue.**

This paper presents a strategy called "compass" for optimizing trajectory of a moving Unmanned Surface Vehicle (USV) to complete localization. The present strategy involves three major components: a) underwater localization algorithm based on centralized Range-Free scheme and Monte Carlo Localization algorithm. b) calculation of an expected information density (EID) map based on prior information. c) R-search mode based on pseudo formation. The "compass" strategy does not rely on high accuracy sensing model, and is well-posed for coverage, time and cost. We successfully simulate localization for targets in two-dimensional work-space using directional sensor under different conditions, and compare performance to traditional trajectories.

I. Introduction

Underwater wireless sensor network (UWSN) is an emerging research area that benefits ocean environmental monitoring, offshore exploration, military surveillance, *etc* [**?**]. It is now more common for oil and gas companies, fishing industry, militaries, and marine researchers to deploy underwater wireless sensor network to obtain submarine data. However, the fundamental problem is how to efficiently complete underwater localization.

currently, many underwater localization algorithms have been proposed, and they can be classified into three categories [**?**]: (1) In stationary localization algorithms, all sensors have fixed locations. (2) In mobile localization algorithms, all sensors are mobile. (3) In hybrid localization algorithms, fixed and mobile sensors are coexist. these three categories can be further compared and summarized into subcategories presented in [**?**], using following aspects:

1) range method: Range-based schemes and range-free schemes
2) time synchronization: Time synchronization is an important role in range-based schemes. in underwater localization environment, it is difficult to achieve precise time synchronization. Thus, in some localization algorithm based on range-based schemes, whole processes are assumed to be perfect time synchronization.
3) localization coverage: Maximizing exploration coverage is a fundamental requirement for underwater localization algorithm. one goal of localization is to balance coverage and cost.
4) localization time: In any localization algorithm, localization time can not be too long. especially in UWSN, sensors can be passively drifted by currents. The localization accuracy will decrease with increasing of time.
5) localization accuracy: Localization accuracy is the most important evaluation criterion. A good localization algorithm or path planning is aim to achieve high localization accuracy.
6) energy consumption: energy consumption can be considered into two parts. firstly, mobile beacon consumption, which means electricity, fuel, *etc*. secondly, wireless sensor consumption.

considering above aspects, accurate time synchronization in real-time will introduces more energy consumption and hardware cost. and the range-based schemes uses ToA (Time of Arrival) and TDoA (Time Difference of Arrival). thus, the accuracy of time synchronization will significantly influence the localization results.additional, unlike the speed of light propagation in air is constant, acoustic speed propagates in water varies with temperature, salinity and water pressure, which is a nonlinear speed model [**?**], [**?**]. considering these reasons, we use range-free schemes to avoid time synchronization and reduce hardware cost. because computation complexity will also influence the energy consumption, we transfer 3D work space into 2D work space by simply equipping a pressure sensor to obtain depth data [**?**] to reduce the computation complexity. also, all computation is centralized, which occur on the USV. the "compass" strategy is inspired by [**?**] [**?**]. the main idea of "compass" strategy is to use prior information to construct EID map map into 360 degree, which is similar as compass. compass will instruct USV the orientation to trade off coverage and cost. after result accuracy achieve specific level, USV enters R-search mode, which is inspired by citeOptimal trajectory determination of a single moving beacon for efficient localization in a mobile ad-hoc network, to refine localization results.

this paper is organized as follows: In Section 2, we discuss the related work of optimizing trajectory algorithm of underwater localization with mobile beacons. Section 3 presents the localization algorithm for a single USV. Section 4 presents the EID algorithm, coverage problem. and pseudo formation.we discuss how "compass" strategy works

s.fekete@tu-bs.de, H. Zhao and A. Becker are with the Dept. of Electrical and Computer Engineering, University of Houston, Houston, TX 70004, USA atbecker@uh.edu.

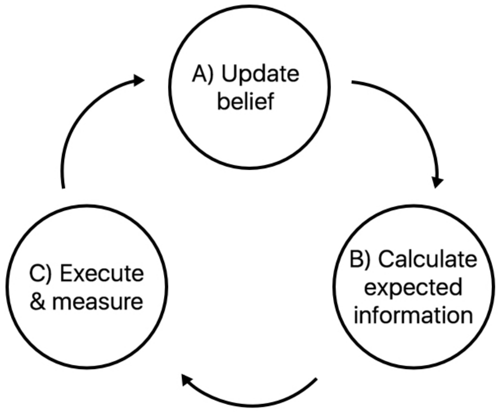

Fig. 1. illustration of the necessary component for the closed-loop localization process. Section 3 discusses step A. Section 4 presents step B. We will talk about step C in Section 5.

to optimizing the trajectory of USV in Section 5. Section 6 shows the simulation results under different conditions compare with traditional strategy. In Section 7. we conclude with discussion on performance and future work.

A. Model

This paper builds on the techniques for controlling many simple robots with uniform control inputs presented in [**?**], [**?**], [**?**], using the following rules:

1) A planar grid *workspace* W is filled with a number of unit-square robots (each occupying one cell of the grid) and some fixed unit-square blocks. Each unit square in the workspace is either *free*, which a robot may occupy or *obstacle* which a robot may not occupy. Each square in the grid can be referenced by its Cartesian coordinates $\boldsymbol{x} = (x, y)$.
2) All robots are commanded in unison: the valid commands are “Go Up” (u), “Go Right” (r), “Go Down” (d), or “Go Left” (l).
3) Robots all move in the commanded direction until they
    a) hit an obstacle
    b) hit a stationary robot.
    c) share an edge with a compatible robot

    If a robot shares an edge with a compatible robot the two robots bond and from then on move as a unit. A *command sequence* $\boldsymbol{m}$ consists of an ordered sequence of moves m_k, where each $m_k \in \{u, d, r, l\}$ A representative command sequence is $\langle u, r, d, l, d, r, u, \ldots\rangle$. We assume the area of W is finite and issue each command long enough for the robots to reach their maximum extent.

II. Related Work

Our efforts have similarities with *mechanical computers*, computers constructed from mechanical, not electrical components. For a fascinating nontechnical review, see [**?**]. These devices have a rich history, from the *Pascaline*, an adding machine invented in 1642 by a nineteen-year old Blaise Pascal; Herman Hollerith’s punch-card tabulator in 1890; to the mechanical devices of IBM culminating in the 1940s. These devices used precision gears, pulleys, or electric motors to carry out calculations. Though our GRID-WORLD implementations are rather basic, we require none of these precision elements—merely unit-size obstacles, and sliding particles sized 2×1 and 1×1 for achieving computational universality.

A. Collision-Based Computing

Collision-based computing has been defined as *“computation in a structureless medium populated with mobile objects”*. For a survey of this area, see the excellent collection [**?**]. Early examples include the billiard-ball computer proposed by Fredkin and Toffoli using only spherical balls and a frictionless environment composed of elastic collisions with other balls and with angled walls [**?**]. Another popular example is Conway’s *Game of Life*, a cellular automaton governed by four simple rules [**?**]. Cells live or die based on the number of neighbors. These rules have been examined in depth and used to design a Turing-complete computer [**?**]. Game of life scenarios and billiard-ball computers are fascinating, but lack a physical implementation. In this paper we present a collision-based system for computation and provide a physical implementation.

B. Sliding-Block Puzzles

Sliding-block puzzles use rectangular tiles that are constrained to move in a 2D workspace. The objective is to move one or more tiles to desired locations. They have a long history. Hearn [**?**] and Demaine [**?**] showed tiles can be arranged to create logic gates, and used this technique to prove PSPACE complexity for a variety of sliding-block puzzles. Hearn expressed the idea of building computers from the sliding blocks—many of the logic gates could be connected together, and the user could propagate a signal from one gate to the next by sliding intermediate tiles. This requires the user to know precisely which sequence of gates to enable/disable. In contrast to such a hands-on approach, with our architecture we can build circuits, store parameters in memory, and then actuate the entire system in parallel using a global control signal.

C. Other Related Work on Programmable Matter

Clearly there is a wide range of interesting scenarios for developing approaches to programmable matter. One such model is the *abstract Tile-Assembly Model* (aTAM) by Winfree [**?**], [**?**], [**?**], which has sparked a wide range of theoretical and practical research. In this model, unit-sized pixels (“tiles”) interact and bond with the help of differently labeled edges, eventually composing complex assemblies. Even though the operations and final objectives in this model are quite different from our particle computation with global inputs (e.g., key features of the aTAM are that tiles can have a wide range of different edge types, and that they keep sticking together after bonding), there is a remarkable geometric parallelism to a key result of our present paper: While it is widely believed that at the most basic level of interaction (called *temperature 1*), computational universality *cannot be achieved* [**?**], [**?**], [**?**] in the aTAM with only unit-sized pixels, very recent work [**?**] shows that computational universality *can be achieved* as soon as even slightly bigger tiles are used. This resembles the results of our paper, which shows that unit-size particles are insufficient for universal computation, while employing bigger particles suffices

III. Conclusion

In this paper we

This work, along with [**?**], [**?**], [**?**], introduces a new model for additive assembly. Interesting applications will aim at nanoscale and microfluidics work.