Optimizing an AUV's path to Localize Underwater Sensors

Haoran Zhao Aaron T. Becker

***Abstract*— We present fundamental progress on localizing underwater sensors with a patrolling AUV**

I. Introduction

One of the exciting new directions of robotics is the design and development of micro- and nanorobot systems, with the goal of letting a massive swarm of robots perform complex operations in a complicated environment. Due to scaling issues, individual control of the involved robots becomes physically impossible: while energy storage capacity drops with the third power of robot size, medium resistance decreases much slower. As a consequence, current micro- and nanorobot systems with many robots are steered and directed by an external force that acts as a common control signal [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**]. These common control signals include global magnetic or electric fields, chemical gradients, and turning a light source on and off.

Clearly, having only one global signal that uniformly affects all robots at once poses a strong restriction on the ability of the swarm to perform complex operations. The only hope for breaking symmetry is to use interactions between the robot swarm and obstacles in the environment. The key challenge is to establish if interactions with obstacles are sufficient to perform complex operations, ideally by analyzing the complexity of possible logical operations. In previous work [**?**], [**?**], [**?**], we were able to demonstrate how a subset of logical functions can be implemented; however, devising a fan-out gate (and thus the ability to replicate and copy information) appeared to be prohibitively challenging. In this paper, we resolve this crucial question by showing that only using unit-sized robots is insufficient for achieving computational universality. Remarkably, adding a limited number of domino-shaped objects *is sufficient* to let a common control signal, mobile particles, and unit-sized obstacles simulate a computer. While this does not imply that large-scale computational tasks should be run on these particle computers instead of current electronic devices, it establishes that future nano-scale systems are able to perform arbitrarily complex operations *as part of the physical system*, instead of having to go through external computational devices.

A. Model

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

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.

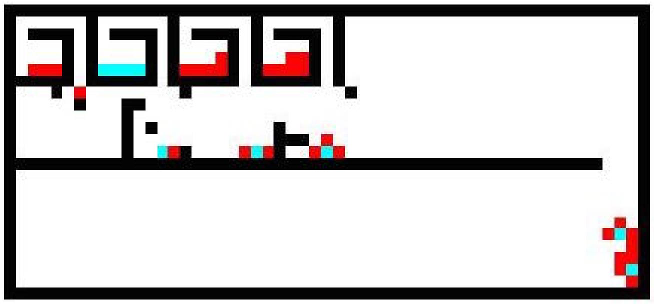

Fig. 1. Gravity-fed hardware implementation of particle computation. The reconfigurable prototype is setup as a FAN-OUT gate using a 2×1 robot (white). This paper proves that such a gate is impossible using only 1×1 robots.

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.