INVITED CONTENT

Buzz

A Programming Language for Robot Swarms

Carlo Pinciroli and Giovanni Beltrame

THE FIELD OF swarm robotics researches fully decentralized approaches for coordinating large-scale teams of robots. This research is ambitious: it envisions robot swarms for scenarios for which current solutions are impractical, too dangerous, or nonexistent.

From drones to self-driving cars, robot swarms are becoming pervasive and are used in many kinds of applications. Such applications include search and rescue, industrial and agricultural inspection, coordinated vehicle platooning, space exploration, and nanomedicine. We envision a world in which designers can specify the behavior of heterogeneous groups of robots and package it in an application that can be installed on multiple robotic systems. Swarm-based solutions will likely form the backbone for the upcoming self-driving-car infrastructure and will enable widespread consumer robotics.

Over the past 20 years, researchers have applied biological swarm models to robotics scenarios to propose new approaches or validate the models in fully controllable conditions. However, over the past decade, a research trend has emerged that parts from natural inspiration and focuses on the engineering aspects of designing robot swarm systems.

This trend is motivated by the need to consolidate current knowledge in the field, which still appears like a collection of scattered success stories. An important open problem in this context is the lack of general tools for experimentation. Remarkable progress is being made on the hardware side, with the creation of widely used swarm robotics platforms such as the e-puck and KiloBot. On the software side, significant research concentrates on single-robot scenarios, with frameworks such as the Robot Operating System. However, common “swarm libraries” do not yet exist, and reusing code is difficult owing to the lack of swarm-centric development platforms. To address these problems, we created Buzz, a programming language for heterogeneous robot swarms.

A Swarm as a Programmable Machine

Although it seems natural to deal with robot swarms as yet another instance of a classic distributed system, important aspects set robot swarms apart from distributed systems. Robot swarm dynamics are characterized by an inseparable mixture of spatial and network aspects. Spatial aspects include the fact that robots move and modify
Swarm robotics researchers have the ability to encode complex behaviors. However, there are several challenges in implementing swarm algorithms comfortably. Currently, a slow trial-and-error process is often necessary to fine-tune individual robots’ behaviors. Buzz is based on the idea that the developer should be offered both levels of abstraction and that the language’s syntax should allow for seamless mixing of bottom-up and top-down constructs.

In Buzz, the “machine” being programmed is a swarm seen as a discrete, heterogeneous collection of robots, each of which executes a Buzz Virtual Machine (BVM). Once a script has been compiled and transmitted to the robots, each robot executes a local copy of it.

One reason we developed the BVM is to facilitate implementing Buzz in resource-limited robots such as the e-puck and the Kilobot. The BVM’s compiled binary is smaller than 12 kilobytes. Despite the BVM’s tiny footprint, it provides automatic garbage collection and low-bandwidth, robust communication protocols. In particular, it uses situated communication whereby a robot, upon receiving a message, can estimate the message source’s relative location.

**Buzz Highlights**

Buzz’s syntax mixes imperative and functional constructs and is inspired by dynamic languages such as JavaScript, Lua, and Python. This ensures a short learning curve for newcomers and readable, familiar-looking scripts.

The language’s bottom-up primitives are analogous to those in traditional dynamic languages and include assignments, arithmetic operations, loops, branches, and function definitions.

FIGURE 1. Definition of a simple distance gradient across a robot swarm. (a) The swarm diagram, in which the star denotes a target, the nodes contain the value of the variable dist of a robot, and the arcs indicate the network or sensing topology, labeled with the distances. (b) The corresponding Buzz code.
Buzz’s main novelty is three top-down primitives that cover a wide range of scenarios and let developers compose complex, reusable behaviors starting from simple concepts. What before took hundreds of lines of platform-dependent C or Python code now can be expressed compactly and intuitively combining these primitives and can be reused for any kind of robot.

The first top-down primitive is the swarm, a collection of robots for which a certain predicate is true. Developers can perform set operations on swarms such as union, intersection, difference, and complement. They can assign to a swarm dedicated tasks, expressed through lambdas.

The second top-down primitive is the neighbors type. It stores positional information about a robot’s immediate neighbors—that is, those within communication range—and provides methods to exchange messages. Developers can use variables of the neighbors type to compute motion vectors and computational fields. For example, Figures 1 and 2 show how simple it is to create a distance gradient from a specific location by combining broadcast (neighbors.broadcast) and message-driven callbacks (neighbors.listen), which sets up a callback triggered whenever a neighbor broadcasts a certain key-value pair). In most other high-level languages, this would require complex synchronization and message passing. In addition, developers can create neighbors variables by filtering or mapping functions onto the stored data.

The third top-down primitive is virtual stigmergy,7 a tuple space that lets the robots share (key, value) pairs across the swarm. Stigmergy refers to a communication modality whereby social insects mark and modify the environment (rather than talking directly) to coordinate tasks such as food search and nest construction. The tuple space acts as a virtual environment in which all robots store information to achieve global consensus. Virtual stigmergy is instrumental to encode globally optimized behaviors such as task allocation and scheduling and to coordinate swarm-wide state transitions.

These top-down primitives do not cover the spectrum of possible swarm algorithms. Nevertheless, they constitute a simple yet powerful set of basic building blocks with which developers can express, combine, and compare a large class of swarm behaviors.

We are creating the first library of swarm behaviors and testing it on real robots of various types. We believe that Buzz and the library will enable future research on real-world, complex swarm robot systems. Currently, no standardized platform lets researchers compare, share, and reuse swarm behaviors. Inescapably, development involves recoding recurring swarm behaviors, such as flocking, reaching consensus, and creating gradients. Buzz’s design is motivated and nurtured by the need to overcome this state of affairs. We hope Buzz will have a lasting impact on the growth of the swarm robotics field.

We ultimately aim to create the first software ecosystem for swarm robotics applications, comprising a domain-specific programming language and the means to deploy, monitor, and debug swarm behaviors. This ecosystem would enable the sharing and comparison of swarm algorithms, letting practitioners lay the foundations of a principled “swarm engineering.” Buzz is distributed as open source software and is available at http://the.swarming.buzz.

References

Searching? Trust CiSE.

CISE is Computing in Science & Engineering, a journal that you’ll find often in your searches for the best in computational content. Trust CiSE. It’s peer-reviewed and appears in both the IEEE Xplore and AIP library packages.


CARLO PINCIROLI is a postdoctoral researcher in the MIST (Making Innovative Space Technology) laboratory at École Polytechnique de Montréal. Contact him at carlo@pinciroli.net.

GIOVANNI BELTRAME is an associate professor in the Computer Engineering Department at École Polytechnique de Montréal and directs the MIST (Making Innovative Space Technology) laboratory. Contact him at giovanni.beltrame@polymtl.ca.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.