Computational Resources of Miniature Robots: Classification and Implications

Stefan M. Trenkwalder

Computational Resources of Miniature Robots: Classification & Implications

Stefan M. Trenkwalder[1]

***Abstract*—When it comes to describing robots, many roboticists choose to focus on the size, types of actuators or other physical capabilities. As most areas of robotics deploy robots with large memory and processing power, the question "how computational resources limit what a robot can do" is often overlooked. However, the capabilities of many miniature robots are limited by significantly less memory and processing power. At present, there is no systematic approach to comparing and quantifying the computational resources as a whole and their implications.**

This paper proposes computational indices that systematically quantify computational resources—individually and as a whole. Then, by comparing 31 state-of-the-art miniature robots, a computational classification ranging from non-computing to minimally-constrained robots is introduced. Finally, the implications of computational constraints on robotic software are discussed.

***Index Terms*—Performance Evaluation and Benchmarking; Software, Middleware and Programming Environments; Control Architectures and Programming; Swarms.**

I. Introduction

ROBOTIC Systems are machines that interact with their environment [1]. These systems contain sensors and actuators to interact with and a computational system to coordinate and manage interactions. Typically, robotic systems are categorised by their physical properties (e.g., size), operation environment (e.g., grounded, airborne), the field of application (e.g., medical, industrial) or the number of robots (e.g., single, multi-robot systems) as shown in [1]–[3]. While the robots depend on their physical properties to perform actions, a minimal amount of computational resources is required to control the desired sequence of actions.

The computation is commonly performed by an embedded system that operates one or multiple microprocessor units (MPU). A single MPU can provide anything from a few billions of Instructions Per Second (IPS) with gigabytes of memory (e.g., Intel i7-9700 with $1.6 \cdot 10^{11}$ (floating-point) IPS and $\leq$ 128 GB of RAM) to only a few million IPS with a few kilobytes of memory (e.g., ATMega 328 with $\leq 2.0 \cdot 10^{7}$ (integer) IPS and 2 kB of RAM). By decreasing the computational resources (i.e., IPS and memory), less complex software can be deployed. Until now, this relationship has not been investigated.

This paper provides the following contributions:

1) A list and comparison of 31 miniature robots.
2) Computational indices systematically quantifying memory and processing power.
3) A classification of robots based on the proposed indices.
4) Requirements analysis of common robotic system software[1] and tasks for miniature robots.

In the next section, computational indices are proposed allowing a classification of robots. Section III discusses current robotic system software with a focus on their computational requirements. Robotic tasks are presented in Section IV. Finally, conclusions are presented in Section V.

II. Computational Quantification & Classification

This work focuses on devices based on a random-access machine (i.e., a type of Turing machine) as defined in [4]. A device must include a processor that executes instructions on registers and can access any memory element at any time. While this covers the majority of devices (in particular robots), other systems—such as quantum computers or biological systems (e.g., animal brains)—are not considered in this work.

Until now, computational resources have only been investigated in [5] which classifies devices used in the Internet-of-Things and wireless sensor networks. [5] uses solely a device's memory (RAM and ROM) to classify devices into Class 0 ($\ll$ 10 kB and $\ll$ 100 kB), Class 1 ($\sim$ 10 kB and $\sim$ 100 kB), and Class 2 ($\sim$ 50 kB and $\sim$ 250 kB) of constrained devices—referred to as CCD 0, 1, and 2. When applying this classification to robots as shown in Table I, it can be seen that many robots exceed the classification; therefore, the classification is insufficient for robotics.

A. Computational Indices

As computational devices are primarily defined by processing power and memory, this paper proposes a memory, M_I, and a processing index, P_I, as

$$M_I = \log\left(1 + m\right), \tag{1}$$

$$P_I = \log\left(1 + \sum_i \underbrace{n_i\, f_i\, e_i}_{\text{IPS per MPU}}\right), \tag{2}$$

Manuscript received: January 09, 2019; Revised April 02, 2019; Accepted May 16, 2019.

This paper was recommended for publication by Editor Dezhen Song upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the DOC Fellowship of the Austrian Academy of Sciences.

[1]S. M. Trenkwalder is with the Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, UK stefan@trenkwalder.tech

Digital Object Identifier (DOI): 10.1109/LRA.2019.2917395.

[1]System software is software that manages/operates the device without implementing a specific application or behaviour (see Section III for more details).

where m is the available primary memory[2] in bytes and n_i, f_i, and e_i are the number of cores, the clock frequency, and the average instructions per clock cycle[3] of an MPU, i.

While each index classifies the magnitude of a computational resource, an individual resource is not sufficient to classify a system for two reasons. Firstly, an increase in a single resource would not necessarily improve a system. For instance, an e-puck robot does not have enough memory to store a single image of its onboard camera. Increased processing power would not enable the robot to load and process a single frame. Secondly, a single algorithm can be implemented prioritising memory consumption or execution time while performing the same action. Ideally, a metric needs to be developed that incorporates the dependency between memory and processing power.

In cryptoanalysis, [6] introduced an execution-time-memory trade-off. It is a general solution to calculate a one-way function inverter[4]. This trade-off describes that an algorithm can speed up by using a larger look-up table with precalculated values. Due to the complexity of one-way function inverters, this method can be applied to a large variety of problems in the complexity class NP, even outside of cryptoanalysis. [6] estimates this trade-off as

$$m\,t^2 = k = const., \tag{3}$$

where m and t are the used memory and processing time for a given implementation. Note that k is algorithm-specific and composed of multiple parameters (see [6] for more details).

To investigate the impact of the robots resources, let m be set to the maximum available memory, m_{max}, and the processing power changed from p' to p. This results in

$$m_{max}\,(t'_{min})^2 \;=\; m_{max}\left(t_{min}\,\frac{p}{p'}\right)^2 \;=\; k', \tag{4}$$

$$m_{max}\,p^2 \;=\; \frac{k'\,(p')^2}{t_{min}^2} \;=\; const., \tag{5}$$

where the memory and the square of the processing power are constant for a given algorithm and response time, t_{min}. In other words, any algorithm satisfying (3) can be implemented on any system satisfying (5) in such a way that the implementations have the same processing time. As a result, these systems can be seen as equally computationally powerful.

By combining (1), (2), and the logarithm of (5), this work proposes the computational index,

$$C_I \;=\; M_I \;+\; 2\,P_I. \tag{6}$$

The computational index was applied to 31 state-of-the-art robots, which are listed in Table I.

[2]Note that m is the size of the primary memory (i.e., RAM), where the processor has direct random access as described by a random-access machine. Data on the secondary (e.g., FLASH, hard drives) or tertiary memory (e.g., SD Cards, cloud storage) requires transfer to the primary memory before accessing it. Therefore, the primary memory is the defining factor.

[3]Note that e_i is not always available or documented. Commonly, high-performance MPUs offer high values (e.g., AMD Ryzen 7 1800x: $e_i = 10.6$) while microcontrollers have values around 1 (e.g., DSPic30F: $e_i = 1.05$). If e_i cannot be determined, it is set to 1 as the worst-case value.

[4]A one-way function is a function of at most polynomial (P) complexity, where its inversion is NP-hard (i.e., nondeterministic polynomial).

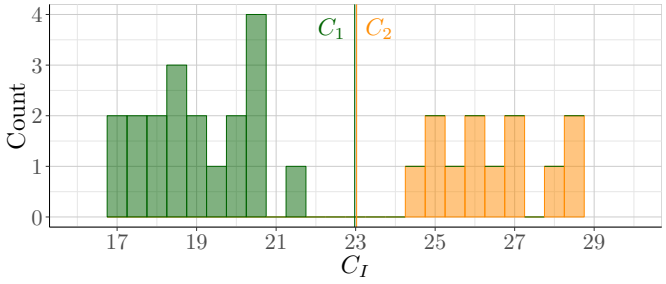

Fig. 1. Histogram of the computational index values, C_I, based on 31 miniature robots of Table 1. The values populate two sets highlighted by colour. Note that C_I is logarithmic, and the gap between the two sets indicate magnitudes higher computational resources.

B. Robot Classification

Table I compares 31 state-of-the-art mobile miniature robots, where a robot is considered miniature if it is millimetre- to decimetre-sized as used in literature (e.g., [22], [33], [40]). Note that nano- and micro-robotics also use the term miniature (e.g., [41]). As these robots do not compute similarly to a random-access machine and only react to introduced environmental changes, they are considered non-computational devices.

When applying (1), (2), and (6) to Table I, it shows that the values of C_I populate two regions, 17–22 and 24–29, as illustrated in Fig. 1. The existence of two distinct regions suggests that robots can be grouped[5] into two sets referred to as severely-constrained ($C_I \leq 23$) and weakly-constrained robots ($C_I > 23$). As described before, robots that do not compute (e.g., [41], [42]) are referred to as non-computational robots. Similarly, robotic systems accessing infrastructure that exceed the capabilities of any individual computer system by magnitudes (e.g., [43]) are referred to as minimally-constrained robots. To simplify the referencing, let class C_0, C_1, C_2, and C_∞ refer to non-computational, severely-constrained, weakly-constrained, and minimally-constrained robots respectively. Note that this classification is shown in Table I.

C. Discussion

When comparing the proposed classification to [5], it can be seen that the classes of [5] consistently fall into C_1, ergo, the classifications are compatible. However, [5] defines fuzzy ranges[6] which makes this classification ambiguous and difficult to apply consistently. In comparison, the proposed classification uses ranges that were chosen based on empirical data of 31 robots.

When plotting the computational resources of Table I with non-miniature robots, Fig. 2 reveals that miniature robots tend to provide fewer resources than non-miniature robots used in research (e.g., iCub) or commercially (e.g., Parrot AR.Drone 2.0). Furthermore, a gap between the two sets of robots is shown in Fig. 1 and 2. This gap can be explained by the fact that many C_1 robots use microcontrollers providing integrated

[5]Note that the threshold is the middle of the gap between the two sets.

[6]For instance, it is not clear into which class a PIC18F67K40 with 3.5 kB of RAM and 128 kB of ROM falls.

TABLE I
COMMON MINIATURE ROBOTS[A], THEIR COMPUTATIONAL RESOURCES, AND CLASSIFICATION. NOTE THAT *Entert.*, *Educ.*, *Med.*, AND *Reconf.* STAND FOR ENTERTAINMENT, EDUCATIONAL, MEDICAL, AND RECONFIGURABLE, RESPECTIVELY.

Robot	MPU/MCU	$\frac{\text{Cores}}{\text{MPU}}$	Arch.	Freq.	RAM	ROM	Application Environment	Type	Network	Group Size	CCD[b]	$\frac{M_I}{P_I}$	C_I	Class
AIBO ERS-7 [7]	MIPS R7000	1	64 bit	576 MHz	64 MB	4 MB	Ground	Entert. Quad-Pedal	-	Single	> 2	7.8 8.8	25.3	C_2
Colias [8]	ATMega 168 ATMega 644	1 1	8 bit 8 bit	20 MHz 20 MHz	1 kB 4 kB	16 kB 64 kB	Ground	Wheeled Educ.	Infra-Red	Multi-Robot	0	3.7 7.9	19.5	C_1
CrazyFly 2.0 [9]	STM32F405RG	1	32 bit	168 MHz	196 kB	1 MB	Air	Quatcopter	Bluetooth	Single Multi-Robot	> 2	5.3 8.2	21.7	C_1
Droplet [10]	Xmega128A3U	1	8 bit	32 MHz	8 kB	128 kB	Ground	Wheeled Research	Infra-Red	Multi-Robot	1	3.9 7.5	18.9	C_1
e-puck [11]	dsPic30	1	16 bit	7 MHz	8 kB	144 kB	Ground	Wheeled	Bluetooth Infra-Red	Swarm	1	3.9 6.8	17.6	C_1
Elmenreich's robot [12]	ATmega328p	1	8 bit	8 MHz	2 kB	32 kB	Ground	Hexapedal	Infra-Red	Multi-Robot	0	3.3 6.9	17.1	C_1
Evo-bot [13]	PIC24	1	16 bit	16 MHz	8 kB	128 kB	Floating	Reconf.	Wired (CAN)	Multi-Robot	1	3.9 7.2	18.3	C_1
GRITSBot [14]	Atmega 328 Atmega 168	1 1	8 bit 8 bit	8 MHz 20 MHz	2 kB 1 kB	32 kB 16 kB	Ground	Wheeled	ANT Infra-Red	Swarm	0	3.5 7.4	18.4	C_1
GoPiGo [15]	Rasberry Pi 3	4	64 bit	1.2 GHz	1 GB	8 GB	Ground	Educ. Wheeled	-	Single Multi-Robot	> 2	9.0 9.7	28.4	C_2
HyMod [16]	ARM Cortex M4	1	32 bit	72 MHz	64 kB	256 kB	Ground	Modular Wheeled	Wired (CAN)	Multi-Robot	2	4.8 7.9	20.5	C_1
I-Swarm[c] [17]	Synopsys 8051	1	8 bit	12 MHz	2 kB	8 kB	Ground	Wheeled	Wired	Modular Swarm	0	3.3 7.1	17.5	C_1
Jasmine[d]	ATMega168	1	8 bit	20 MHz	1 kB	16 kB	Ground	Wheeled	Infra-Red	Modular Swarm	0	3.0 7.3	17.6	C_1
Khepera IV [18]	ARM Cortex-A8	1	32 bit	800 MHz	512 MB	4 GB	Ground	Wheeled	WiFi Bluetooth	Multi-Robot	> 2	8.7 9.2	26.5	C_2
Kilobot [19]	ATMega328	1	8 bit	8 MHz	2 kB	32 kB	Ground	Wheeled	Infra-Red	Swarm	0	3.3 6.9	17.1	C_1
Lego Mindstorms NXT [20]	ATMEL AT91 ATMega48	1 1	32 bit 8 bit	48 MHz 8 MHz	64 kB 512 B	256 kB 4 kB	Main Unit	Educ.	Bluetooth USB	Single	2 0	4.8 7.7	20.3	C_1
M-Block [21]	STM32F051 ARM Cortex-M0	1	32 bit	48 MHz	8 kB	64 kB	Ground	Reconf. Jumping	ANT Bluetooth	Multi-Robot	1	3.9 7.7	19.3	C_1
marXbot [22]	i.MX31 (ARM 11)	1	32 bit	533 MHz	128 MB	-[e]	Ground	Wheeled	WiFi Bluetooth	Multi-Robot	> 2	8.1 8.7	25.6	C_2
MHP [23]	ARM Cortex M4	1	32 bit	72 MHz	64 kB	256 kB	Underwater	Modular	Infra-Red	Multi-Robot	2	4.8 7.9	20.5	C_1
Micro Quadrotor [24]	ARM Cortex-M3	1	32 bit	72 MHz	64 kB	128 kB	Air	Quadcopter	ZigBee	Swarm	> 2	4.8 7.9	20.5	C_2
Monsun II [25]	Blackfin BF537	1	16 bit	500 MHz	32 MB	40 MB	Underwater	UAV	Bluetooth Wired	Multi-Rrobot	> 2	7.5 8.7	24.9	C_2
mROBerTO [26]	Nordic nRF51422	4	32 bit	16 MHz	32 kB	256 kB	Ground	Wheeled Educ.	Bluetooth ANT	Multi-Robot	2	4.5 7.8	20.1	C_1
Pheeno [27]	ARM Cortex-A7 ATmega328p	4 1	32 bit 8 bit	900 MHz 8 MHz	1 GB 2 kB	- 32 kB	Ground	Wheeled Educ.	WiFi Bluetooth	Multi-Robot	> 2	9.0 9.6	28.1	C_2
r-one [28]	TI LM3S8962	1	32 bit	50 MHz	64 kB	256 kB	Ground	Wheeled	Infra-Red ZigBee	Multi-Robot Swarm	2	4.8 7.7	20.2	C_1
s-bot [29]	Intel XScale	1	32 bit	400 MHz	64 MB	32 MB	Ground	Wheeled	WiFi	Swarm	> 2	7.8 8.6	25.0	C_2
Soft Robotic Fish [30]	ATMega 644	1	8 bit	20 MHz	4 kB	64 kB	Underwater	Research Soft	ZigBee	Single	0	3.6 7.3	18.2	C_1
Thymio II [31]	PIC24	1	16 bit	8 MHz	16 kB	128 kB	Ground	Wheeled	IEEE 802.15.4	Swarm	1	4.2 6.9	18.0	C_1
TurtleBot 3 (Burger) [32]	Rasberry Pi 3 ARM Cortex-M7	4 1	64 bit 32 bit	1.2 GHz 216 MHz	1 GB 320 kB	- 1 MB	Ground	Wheeled Educ.	USB Ethernet	Single	> 2	9.0 9.7	28.4	C_2
UltraSwarm [33]	Intel XScale PXA255	1	32 bit	200 MHz	64 MB	-[e]	Air	Helicopter	Wifi Bluetooth	Multi-Robot	> 2	7.8 8.3	24.4	C_2
Wanda [34]	TI LM3S1960 Bluetechnix CM-BF561	1 2	32 bit 16 bit	50 MHz 600 MHz	64 kB 64 MB	256 kB 8 MB	Ground	Wheeled Educ.	Infra-Red ZigBee	Multi-Robot	> 2	7.8 9.1	26.0	C_2
WolfBot [35]	ARM Cortex-A8	1	32 bit	1 GHz	512 MB	4 GB	Ground	Wheeled Educ.	WiFi ZigBee	Multi-Robot	> 2	8.7 9.0	26.7	C_2
X4-MaG [36]	dsPic33F ARM Cortex-A8	1 1	16 bit 32 bit	40 MHz 1.2 GHz	8 kB 512 MB	128 kB -	Air	Quadcopter	RS232 WiFi	Swarm	> 2	8.7 9.1	26.9	C_2

[a] Robots has been obtained from various sources—including [1], [37]–[39]—and it has been included in this table if it is miniature and mobile and details of their computational capabilities could be obrained from publications, datasheets, manuals, or project websites. If the robot has not been used in publication within the last 10 years (e.g., Alice) or technical specification is not available (e.g., Anki Vector, Cubelets, Dash, Root, and Sphero), the robot has not been included.
[b] CCD is the class of constrained devices as proposed in [5].
[c] All details were taken from the I-Swarm project homepage (http://www.i-swarm.org/MainPage/Robots/R_Description1.htm).
[d] All details were taken from the Jasmine project homepage (http://www.swarmrobot.org/GeneralDesign.html).
[e] Available onboard secondary memory was not specified.

memory, and C_2 robots tend to provide MPUs with discrete memory chips.

While integrated memory is often limited by the die size and production methods, discrete memory chips provide a large amount of memory relatively cheaply. In other words, when moving from a microcontroller to an MPU, magnitudes larger amounts of memory are available causing the gap in Fig. 2 (right). In contrast, the processing power transitions from low (i.e., microcontrollers) to high values (i.e., MPUs) less distinctively (see Fig. 2 top). Note that these characteristics can also be seen when comparing 5227 computer systems (see supplementary material website [44]). As this results

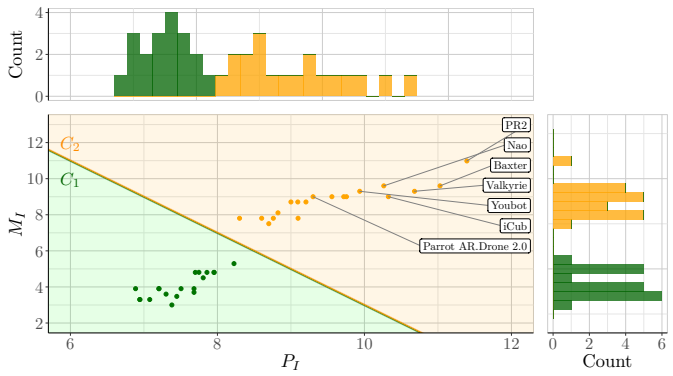

Fig. 2. Memory indices, M_I, and processing indices, P_I, of Table I and common non-miniature platforms (with labels). The top and right plot show a histogram for M_I and P_I, respectively. Green and orange indicated severely- and weakly-constrained robots, respectively.

from deploying two types of systems (i.e., microcontrollers and MPUs), it is likely that future systems will show similar characteristics even though the threshold (23) might shift with further technological advances.

Interestingly, C_1 robots are almost exclusively used in research environments. This potentially stems from the complexity of real-world tasks and the technical hurdles faced (i.e., reality gap). Constrained computational resources represent such a hurdle. This argument is supported by recent work, such as [45], where computational extensions[7] are built for an existing miniature robot to specifically overcome the reality gap in evolutionary robotics.

One can argue that, based on Moore's law, more robots will be developed with increased computational resources, thus reducing the latter technical difficulties. However, miniature robots, such as Robotbees [46] face challenges regarding size, weight, and power consumption. As microcontrollers tend to be smaller, lighter, and consume less power than high-performance multicore processors, it can be expected that further miniaturisation will continue to create severely-constrained robots.

III. Robotic System Software

Let us examine the implications of the computational resources of robots discussed above on their respective software. Software can be categorised into application software (also behaviour or behavioural software) and system software. System software is an umbrella term for any software that controls and manages the system itself without implementing a behaviour or application. Therefore, system software provides a platform for behavioural software. It commonly aims to reduce development time and complexity as well as to improve reusability and deployability. Typical examples are firmware, operating systems, middleware, and virtual machines. In robotics, system software is commonly a cloud-enabled system software, middleware, or a virtual machine.

[7]This work considers a robot with hardware alteration a different robot.

A. Cloud-enabled System Software

To overcome computational constraints, resources can be outsourced to external devices (e.g., clouds) via a reliable network. These cloud-enabled systems exist in two forms, (I) where the behaviour is implemented on a cloud, which remotely operates robots (i.e., Robot-as-a-Service (RaaS) [43], [47]) and (II) where the behaviour is implemented on the robot, which uses the cloud for computation or storage (i.e., cloud robotics [48], [49]). Examples are Robot Cloud Center [50] (RaaS), Rapyuta [51] (cloud robotics processing), and RoboEarth [52] (cloud robotics knowledgebase).

Performing remote computation and storage enables a more cost-efficient robot design due to reduced computational requirements. However, these systems require a reliable connection to the cloud. This is a significant limitation in many miniature robots as they often provide simple communication capabilities. As a result, these systems are not feasible for all cases.

As cloud robotics and RaaS extend the computational power of single robots, the entire system needs to be considered in their classification. Since clouds exceed the capabilities of any individual computer system by magnitudes, such a system is considered a minimally-constrained robotic system.

B. Robotic Middleware

When accessing another infrastructure is not feasible, local processing is often a more practical approach. As many operating systems, such as Windows or Linux, manage resources and communication, they often lack functionality needed in robots (e.g., representation in a physical space or actuation control). Consequently, a broad range of robotic *middleware*[8] has been developed to provide those missing features [53], [54].

This section discusses a selection of robotic middlewares—Miro, ORoCoS, Player, and ROS. Miro [55], ORoCoS [56], Player [57], and ROS [58] have been used widely on platforms for RoboCup, real-time and safety-critical environments, research, and industry and research, respectively. Overall, they share similar features:

- a layer (Miro and ORoCoS), a control thread (Player), or dedicated nodes (ROS) abstracting robotic hardware, which improves portability,
- modular[9] software development, which improves reusability and generality, and
- a large set of often-used algorithms and features of its domain reducing developmental time and efforts.

While robotic middlewares are widely used, one limitation of middleware (including any presented in [53], [54]) is its need for an operating system. Any middleware utilising Linux[10], Windows[11], or macOS[12] requires at least $C_I \geq 26.5$,

[8]A middleware is any software that is executed between an operating system and an application (i.e., behaviour).

[9]Miro, OroCoS, and ROS provide a well-defined interface between modules—CORBA (Miro and ORoCoS), RPC (ROS). This allows development with various programming languages and on various systems.

[10]The requirement is based on Ubuntu Mate 18.04 as it is required for the current version of ROS (Melodic Morenia).

[11]The requirement is based on Windows 10 IoT.

[12]The requirement is based on macOS Mojave 10.14.

25.6, or 28.6, respectively. This makes middleware unsuitable for many miniature robots.

C. Robotic Languages & Virtual Machines

Another form of system software gains popularity—Virtual Machines (VM) executing Domain Specific Languages (DSL) [59]. A DSL is a programming language providing frequently-used functions and a high level of abstraction, which decreases development-time, reduces lines of code and, consequently, improves software quality [60], [61]. In this section, ASEBA [62], Buzz [63], Urbi [7], and Supervisory control framework [64] are discussed as they have been used on mobile miniature robots.

VMs are system software that interpret source code (Urbi), bytecode (ASEBA and Buzz), or a generated finite-state machine table (supervisory control). While Urbi interprets scripts directly, the compilation to bytecode allows a higher execution efficiency[13] as repeated syntactic and semantic analysis can be avoided. While the use of a VM increases the portability of the code, it is also a limitation as interpreted code—except the calling of library functions—has a reduced execution efficiency as shown in [61], [65]. For example, [62] reports an average execution efficiency of $\frac{1}{70}$ for ASEBA scripts.

Overall, the VMs of ASEBA, Buzz[14] and supervisory control have been implemented directly on robots with a computational index of at least 17.7, 17.2 and 17.2 (i.e., C_1 robots), respectively. In comparison, the Urbi VM has been implemented on robots with a computational index of 25.3 (i.e., C_2 robots). Note, Urbi is also a middleware as it requires an operating system (APERIOS[15]).

D. Discussion

When comparing the robotic system software and their class of guaranteed deployability (CGD)[16] as shown in Table II, it can be seen that each type of system software focuses on a different group of robots. Cloud-enabled system software is executed on infrastructure that exceed the capabilities of any individual robot. As a result, robots can use a large set of features and models, can utilise knowledge databases, and can perform time-consuming calculations. Robotic middleware is designed for robots with operating systems (i.e., weakly-constrained devices). It utilises operating system features and often provide additional development tools, libraries, simulators, and other features. Robotic languages and virtual machines, on the other hand, are programming environments to describe a robot's behaviour. As the VMs only translates DSL commands to a robot's action, they can be deployed on many severely-constrained robots.

[13]In this work, execution efficiency describes how many instructions are needed for a single DSL instruction. As supervisory control uses events triggering the execute operations, execution efficiency cannot be applied.

[14]The considered Buzz implementation is based on BittyBuzz providing a smaller but limited version of the Buzz VM [66].

[15]APERIOS is a Proprietary Real-Time Operating system developed by Sony for the Aibo robot [7].

[16]The CGD is the class of the robot with the smallest C_I that was capable of executing the respected software based on an extensive literature search. Note that a specific CGD indicates that no evidence was found that the respected software can be deployed on a robot with less resources.

TABLE II
COMMON ROBOTIC SYSTEM SOFTWARE AND THEIR CLASS OF GUARANTEED DEPLOYABILITY (CGD).

System Software	CGD
A. Cloud-enabled System Software	
Robot Cloud Center [50]	C_∞
Rapyuta [51]	C_∞
RoboEarth [52]	C_∞
B. Robotic Middleware	
Miro [55]	C_2
ORoCoS [56]	C_2
Player [57]	C_2
ROS [58]	C_2
C. Robotic Languages & Virtual Machines	
ASEBA [62]	C_1
Buzz [63]	C_1
Urbi [67]	C_2
Supervisory Control [64]	C_1

Generally, each type of system commonly provides (I) abstraction of hardware allowing fast high-level development and (II) modular design capabilities allowing fast adaptation to changes (in particular, Miro, ORoCoS, ROS, ASEBA, Urbi, and Supervisory control). While system software for minimally-constrained and weakly-constrained robots offers large sets of libraries, system software for severely-constrained robots often does not. This could stem from a higher availability of weakly-constrained robots and, therefore, larger communities (e.g., ROS). Also, it is likely that reduced computational resources hinder the development and deployment of generic and less hardware-optimised libraries.

Generic features and libraries could be implemented in DSLs, which ensures portability. However, the reduced execution efficiency could impact the behaviour of a robot. One approach to improve the execution efficiency is to move from DSL VMs to DSL compilers allowing the code to be executed directly on the hardware. This would (I) provide a good level of abstraction by the DSL, (II) prevent the inefficient execution on a VM, and (III) allow the implementation of large sets of libraries and features. On the other hand, it might negatively impact development times due to repeated recompilation and increased difficulty of debugging.

The largest difference between severely- and weakly-constrained robots is that the majority of behaviours on severely-constrained robots are still implemented directly without system software. This is likely to be a consequence of the computational constraints of the robot and the subsequent need of efficient execution. As discussed in the previous paragraph, one approach to overcome this is the execution of compiled DSL code. Alternative approaches could involve other system software, such as embedded operating systems similar to sensor network operating systems [68] (e.g., TinyOS [69] or Contiki [70]). Overall, this indicates that more engineering and research efforts are required to improve software engineering methods on severely-constrained robots.

IV. ROBOTIC TASKS

Finally, in this section, the implications of computational resources on tasks performed by miniature robots are inves-

TABLE III
COMMON RESEARCH TASKS IN SWARM ROBOTICS AND THEIR CLASS OF GUARANTEED DEPLOYABILITY (CGD). A COMPREHENSIVE DESCRIPTION OF THE SWARM ROBOTICS TASKS CAN BE FOUND IN [37], [71], [72]. NOTE THAT THE CGD IS BASED ON THE CITED WORK.

Tasks	CGD
(I) Spatially-Organising Behaviours	
Aggregation [73]	C_1
Pattern Formation [74]	C_1
Object Clustering [75]	C_1
(II) Navigation Behaviours	
Collective Exploration/Mapping [76]	C_2
Collective Movement [77]	C_1
Collective Transport [78]	C_1
(III) Collective-Decision Making	
Consensus Achievement [79]	C_1
Task Allocation [80]	C_1
(IV) Other Collective Behaviours	
Collective Fault Detection [81]	C_1
Human-Swarm Interaction [82]	C_1
(V) Complex Multi-Task Behaviours	
Foraging [83]	C_1/C_2
Search and Rescue [84]	C_2
Surveillance [85]	C_2

tigated. Candidate tasks can be taken from reconfigurable or swarm robotics as the majority of miniature robots are deployed in these areas. While reconfigurable robotics mostly investigates the creation of shapes and operability of assembled robots, swarm robotics provides a large variety of tasks motivated by potential future applications and, hence, these are used in this section.

Generally, swarm robotics research investigates the solving of tasks with large numbers of robots, a lack of central infrastructure, and sole access to local information. An extensive list of swarm robotics tasks can be found in [37], [71], [72]. Based on [71], a task can be a (I) spatially-organising behaviour, (II) navigation behaviour, (III) collective decision-making algorithm, (IV) other collective behaviour, and (V) multi-task behaviour.

Table III shows swarm robotics tasks and their CGD. It can be seen that solutions for individual tasks, (I)–(IV), can be performed by C_1 robots, the only exception being collective exploration/mapping. This behaviour was implemented on robots with $C_I \geq 25.6$ (i.e., C_2 robots). Tasks composed of multiple individual tasks, (V), tend to require more powerful (i.e., C_2) robots.

One such task, foraging, is a canonical class of tasks that can combine exploration, mapping, navigation, path-planning, object-recognition, decision-making, and transport [86], [87]. It is notable that versions of foraging are suitable for C_1 robots (e.g., [83]). However, in these cases, the deployment of a system is limited to special environments allowing implicit path-planning/navigation (e.g., via pheromone tracks). Systems without such restrictions consistently use C_2 robots.

Surveillance, as well as search and rescue, are used in swarm robotics [84], [85]. However, both tasks are consistently performed by C_2 robots. In many cases, it can require the performing of simultaneous localisation and mapping (SLAM), object-recognition, and navigation, which themselves are non-trivial (e.g., SLAM requires a weakly-constrained robot [88]).

Overall, it has been demonstrated that severely-constrained robots are in many cases capable of performing individual tasks. However, when the complexity increases (i.e., multi-task behaviours), tasks are only performed within a simplified environment or by weakly-constrained (i.e., more powerful) robots. This suggests that severe computational constraints are a hurdle hindering a system from performing more complex behaviours in more realistic environments.

V. CONCLUSION

This work focuses on an often overlooked property of robots, the computational resources and their implications on software. Firstly, computational indices were proposed to systematically quantify the resources of a system. Then, a classification was introduced categorising the entire spectrum of robotics into non-computational, severely-constrained, weakly-constrained, and minimally-constrained robots. Based on the data of 31 state-of-the-art miniature robots, it was shown that miniaturisation tends to reduce a robot's computational resources. This results in a large proportion of miniature robots being severely-constrained. The severely-constrained resources restrict a robot to system software lacking extensive libraries and features and limit robots to perform single individual behaviours within a simplified environment. As miniaturisation will continue to produce severely-constrained robots, more research efforts are required to enable the robots to perform complex tasks in a complex environment.

Note that there are early research efforts toward computation-free[17] control [73]. However, it requires a specialised environment and many technical challenges still need to be overcome for it to be deployed in a real-world environment.

Alternative approaches to overcome the computational limitations could include (I) outsourcing computation to external infrastructures similar to cloud-enabled system software, (II) reducing computation by designing mechanisms similar to micro- and nanorobotics, or (III) distributing computation across multiple robots. As described in this work, cloud-enabled system software, (I), is only suitable for a small number of robots with robust communication to a central infrastructure. The design of hardware/physical mechanisms, (II), would allow actions without computation which is an efficient approach. On the other hand, it lacks the flexibility of software as it potentially requires redesigning of the system when a change to the task or environment occurs. Combining resources across robots, (III), is a general approach that can be used for a multitude of problems. However, it requires robust communication, adequate resource management, and new methods of designing behaviours. Consequently, further research on these aspects could enable (I) as well as (III) for miniature systems.

Potential future work includes: (I) extending the computational index to incorporate the benefits of parallel processing

[17]Note that computation-free in [73] only refers to "without arithmetic operations." As the controller is a Mealy automata it performs computation.

and floating-point units; (II) analysing more robots including non-miniature robots; and (III) statistically analysing the computational indices for each task, which could reveal more details on the requirements of different approaches.

ACKNOWLEDGMENT

The author thanks Izabela S. Stopinska, Dr Gabriel Kapellmann Zafra, and Dr Yuri Kaszubowski Lopes for their feedback and input.

REFERENCES

[1] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Germany: Springer, 2016.

[2] A. Dobra, "General classification of robots. Size criteria," in *Proc. 2014 IEEE Int. Conf. Robot. in Alpe-Adria-Danube Region (RAAD)*. Piscataway, NJ: IEEE, 2014, pp. 1–6.

[3] IEEE, "Robots grouped by IEEE," https://robots.ieee.org/learn/types-of-robots/, 03 2019, accessed on: 15-Mar-2019.

[4] B. Robič, *The foundations of computability theory*. Berlin, Germany: Springer, 2015.

[5] C. Bormann, M. Ersue, and A. Keranen, "Terminology for Constrained-Node Networks," https://tools.ietf.org/html/rfc7228, RFC 7228, Internet Engineering Task Force (IETF), 2014, accessed on: 09-Oct-2016.

[6] M. Hellman, "A cryptanalytic time-memory trade-off," *IEEE Trans. Inf. Theory*, vol. 26, no. 4, pp. 401–406, 1980.

[7] M. Fujita and R. Enteretainment, "Entertainment Robot: AIBO," *J. Inst. Image Inform. and Telev. Eng.*, vol. 54, no. 5, pp. 657–661, 2000.

[8] F. Arvin, J. Murray, C. Zhang, and S. Yue, "Colias: An autonomous micro robot for swarm robotic applications," *Int. J. Adv. Robot. Syst.*, vol. 11, no. 7, p. 113, 2014.

[9] W. Giernacki, M. Skwierczyński, W. Witwicki, P. Wroński, and P. Kozierski, "Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering," in *Proc. 2017 IEEE Int. Conf. Methods and Models in Autom. and Robot. (MMAR)*. Piscataway, NJ: IEEE, 2017, pp. 37–42.

[10] N. Farrow, J. Klingner, D. Reishus, and N. Correll, "Miniature six-channel range and bearing system: algorithm, analysis and experimental validation," in *Proc. 2014 IEEE Int. Conf. Robot. and Autom. (ICRA 2014)*. Piscataway, NJ: IEEE, 2014, pp. 6180–6185.

[11] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Proc. 9th Conf. Auton. Robot Syst. and Competitions*, vol. 1. Castelo Branco, Portugal: IPCB, 2009, pp. 59–65.

[12] W. Elmenreich, B. Heiden, G. Reiner, and S. Zhevzhyk, "A low-cost robot for multi-robot experiments." in *Proc. 2015 IEEE Int. Works. Intel. Solutions Embed. Syst. (WISES 2015)*. Piscataway, NJ: IEEE, 2015, pp. 127–132.

[13] J. A. Escalera, M. Doyle, F. Mondada, and R. Groß, "Evo-bots: A Simple, Stochastic Approach to Self-Assembling Artificial Organisms," in *Proc. 2016 Int. Symp. Distrib. Auton. Robot. Syst. (DARS 2016)*, no. EPFL-CONF-221161. Berlin, Germany: Springer, 2016, pp. 373–383.

[14] D. Pickem, M. Lee, and M. Egerstedt, "The GRITSBot in its natural habitat-a multi-robot testbed," in *Proc. 2015 IEEE Int. Conf. Robot. and Autom. (ICRA 2015)*. Piscataway, NJ: IEEE, 2015, pp. 4062–4067.

[15] A. Pierson, Z. Wang, and M. Schwager, "Intercepting Rogue Robots: An Algorithm for Capturing Multiple Evaders With Multiple Pursuers." *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 530–537, 2017.

[16] C. Parrott, T. J. Dodd, and R. Groß, "HyMod: A 3-DOF Hybrid Mobile and Self-Reconfigurable Modular Robot and its Extensions," in *Proc. 2016 Int. Symp. Distrib. Auton. Robot. Syst. (DARS 2016)*. Berlin, Germany: Springer, 2016, pp. 401–414.

[17] J. Seyfried, M. Szymanski, N. Bender, R. Estaña, M. Thiel, and H. Wörn, "The I-SWARM Project: Intelligent Small World Autonomous Robots for Micro-manipulation," in *Proc. 2004 Int. Conf. Swarm Robot. (SAB 2004)*. Berlin, Germany: Springer, 2005, pp. 70–83.

[18] J. M. Soares, I. Navarro, and A. Martinoli, "The Khepera IV mobile robot: performance evaluation, sensory data and software toolbox," in *Proc. 2nd Iberian Robot. Conf.* Berlin, Germany: Springer, 2016, pp. 767–781.

[19] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A Low Cost Scalable Robot System for Collective Behaviors," in *Proc. 2012 IEEE Int. Conf. Robotics and Autom. (ICRA 2012)*. Piscataway, NJ: IEEE, 2012, pp. 3293–3298.

[20] W. Grega and A. Pilat, "Real-time control teaching using LEGO® MINDSTORMS® NXT robot," in *Int. Multi-Conf. Comput Sci. and Inform. Techn. (IMCSIT 2008)*. Piscataway, NJ, USA: IEEE, 2008, pp. 625–628.

[21] J. W. Romanishin, K. Gilpin, S. Claici, and D. Rus, "3D M-Blocks: Self-reconfiguring robots capable of locomotion via pivoting in three dimensions," in *Proc. 2015 IEEE Int. Conf. Robot. and Autom. (ICRA 2015)*. Piscataway, NJ: IEEE, 2015, pp. 1925–1932.

[22] M. Bonani, V. Longchamp, S. Magnenat, P. Rtornaz, D. Burnier, G. Roulet, F. Vaussard, H. Bleuler, and F. Mondada, "The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research," in *Proc. 2010 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS 2010)*. Piscataway, NJ, USA: IEEE, 10 2010, pp. 4187–4193.

[23] M. J. Doyle, X. Xu, Y. Gu, F. Perez-Diaz, C. Parrott, and R. Groß, "Modular hydraulic propulsion: A robot that moves by routing fluid through itself," in *Proc. 2016 IEEE Int. Conf. Robotics and Autom. (ICRA 2016)*. Piscataway, NJ, USA: IEEE, 2016, pp. 5189–5196.

[24] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Auton. Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[25] C. Osterloh, T. Pionteck, and E. Maehle, "MONSUN II: A small and inexpensive AUV for underwater swarms," in *Proc. 7th German Conf. Robot. (ROBOTIK 2012)*. Frankfurt, Germany: VDE, 2012, pp. 1–6.

[26] J. Y. Kim, T. Colaco, Z. Kashino, G. Nejat, and B. Benhabib, "mROBerTO: A modular millirobot for swarm-behavior studies," in *Proc. 2016 IEEE/RSJ Int. Conf. Int. Robots and Syst. (IROS 2016)*. Piscataway, NJ: IEEE, 2016, pp. 2109–2114.

[27] S. Wilson, R. Gameros, M. Sheely, M. Lin, K. Dover, R. Gevorkyan, M. Haberland, A. Bertozzi, and S. Berman, "Pheeno, A Versatile Swarm Robotic Research and Education Platform," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 884–891, 2016.

[28] J. McLurkin, J. Rykowski, M. John, Q. Kaseman, and A. J. Lynch, "Using multi-robot systems for engineering education: teaching and outreach with large numbers of an advanced, low-cost robot," *IEEE Trans. Educ.*, vol. 56, no. 1, pp. 24–33, 2013.

[29] F. Mondada, G. C. Pettinaro, A. Guignard, I. W. Kwee, D. Floreano, J.-L. Deneubourg, S. Nolfi, L. M. Gambardella, and M. Dorigo, "SWARM-BOT: A new distributed robotic concept," *Auton. Robots*, vol. 17, no. 2-3, pp. 193–221, 2004.

[30] A. D. Marchese, C. D. Onal, and D. Rus, "Autonomous soft robotic fish capable of escape maneuvers using fluidic elastomer actuators," *Soft Robotics*, vol. 1, no. 1, pp. 75–87, 2014.

[31] F. Riedo, M. Chevalier, S. Magnenat, and F. Mondada, "Thymio II, a robot that grows wiser with children," in *Proc. IEEE Workshop on Adv. Robot. Soc. Impac. (ARSO 2013)*, Eidgenössische Technische Hochschule Zürich, Autonomous System Lab. Piscataway, NJ, USA: IEEE, 2013, pp. 187–193.

[32] E. Guizzo and E. Ackerman, "The TurtleBot3 Teacher," *IEEE Spectrum*, vol. 54, no. 8, pp. 19–20, 2017.

[33] R. Nardi and O. Holland, "UltraSwarm: A Further Step Towards a Flock of Miniature Helicopters," in *Swarm Robot.*, ser. Lecture Notes in Comput. Sci., E. Şahin, W. Spears, and A. Winfield, Eds. Berlin, Germany: Springer, 2007, vol. 4433, pp. 116–128.

[34] A. Kettler, M. Szymanski, J. Liedke, and H. Wörn, "Introducing wanda-a new robot for research, education, and arts," in *Proc. 2010 IEEE/RSJ Int. Conf. Int. Robots and Syst. (IROS 2010)*. Piscataway, NJ: IEEE, 2010, pp. 4181–4186.

[35] J. Betthauser, D. Benavides, J. Schornick, N. O'Hara, J. Patel, J. Cole, and E. Lobaton, "WolfBot: A distributed mobile sensing platform for research and education," in *Proc. Conf. American Soc. Eng. Edu. (ASEE 2014 Zone 1)*. Piscataway, NJ: IEEE, 2014, pp. 1–8.

[36] A. Manecy, N. Marchand, F. Ruffier, and S. Viollet, "X4-MaG: a low-cost open-source micro-quadrotor and its linux-based controller," *Int. J. Micro Air Veh.*, vol. 7, no. 2, pp. 89–109, 2015.

[37] H. Hamann, *Swarm Robotics: A Formal Approach*. Berlin, Germany: Springer, 2018.

[38] Wikimedia Foundation, "List of common swarm robots," https://en.wikipedia.org/wiki/Swarm_robotic_platforms, 2019, accessed on: 29-March-2019.

[39] P. Moubarak and P. Ben-Tzvi, "Modular and reconfigurable mobile robotics," *Robot. Auton. Sys.*, vol. 60, no. 12, pp. 1648–1663, 2012.

[40] S. Martel, M. Sherwood, C. Helm, W. G. De Quevedo, T. Fofonoff, R. Dyer, J. Bevilacqua, J. Kaufman, O. Roushdy, and I. Hunter, "Three-legged wireless miniature robots for mass-scale operations at the sub-atomic scale," in *Proc. 2001 IEEE Int. Conf. Robot. and Autom. (ICRA 2001)*. Piscataway, NJ: IEEE, 2001, pp. 3423–3428.

[41] S. Fusco, M. S. Sakar, S. Kennedy, C. Peters, S. Pane, D. Mooney, and B. J. Nelson, "Self-folding mobile microrobots for biomedical applications," in *Proc. 2014 IEEE Int. Conf. Robot. and Autom. (ICRA 2014)*. Piscataway, NJ, USA: IEEE, 2014, pp. 3777–3782.

[42] S. Lee, S. Kim, S. Kim, J.-Y. Kim, C. Moon, B. J. Nelson, and H. Choi, "A Capsule-Type Microrobot with Pick-and-Drop Motion for Targeted Drug and Cell Delivery," *Adv. Healthc. Mater.*, vol. 7, no. 9, p. 1700985, 2018.

[43] L. S. Terrissa, B. Radhia, and J.-F. Bretheé, "Towards a new approach of Robot as a Service (RaaS) in Cloud Computing paradigm," in *Proc. 5th. Int. Symp. ISKO-Maghreb*. Hammamet , Tunisia: International Society for Knowledge Organization, 11 2015.

[44] S. M. Trenkwalder, "Online supplementary material page," 5 2019, accessed on: 04-May-2019. [Online]. Available: https://trenkwalder.tech/pubs/robot-classification

[45] S. Jones, M. Studley, S. Hauert, and A. F. T. Winfield, "A Two Teraflop Swarm," *Frontiers in Robotics and AI*, vol. 5, p. 11, 2018.

[46] R. Wood, R. Nagpal, and G.-Y. Wei, "Flight of the Robobees," *Scientific American*, vol. 308, no. 3, pp. 60–65, 2013.

[47] Y. Chen, Z. Du, and M. García-Acosta, "Robot as a Service in Cloud Computing," in *Proc. 2010 IEEE Int. Symp. Serv. Orient. Syst. Eng. (SOSE 2010)*. Piscataway, NJ, USA: IEEE, 2010, pp. 151–158.

[48] K. Goldberg and B. Kehoe, "Cloud robotics and automation: A survey of related work," *Tech. Rep. UCB/EECS-2013-5*, 2013.

[49] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A Survey of Research on Cloud Robotics and Automation," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 398–409, 4 2015.

[50] Z. Du, W. Yang, Y. Chen, X. Sun, X. Wang, and C. Xu, "Design of a Robot Cloud Center," in *Proc. 2011 IEEE Int. Symp. Auton. Decentr. Syst. (ISADS 2011)*. Piscataway, NJ, USA: IEEE, 2011, pp. 269–275.

[51] G. Mohanarajah, D. Hunziker, R. D'Andrea, and M. Waibel, "Rapyuta: A Cloud Robotics Platform," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 481–493, 4 2015.

[52] M. Waibel, M. Beetz, J. Civera, R. d'Andrea, J. Elfring, D. Galvez-Lopez, K. Häussermann, R. Janssen, J. Montiel, A. Perzylo *et al.*, "RoboEarth," *IEEE Robot. Autom. Mag.*, vol. 18, no. 2, pp. 69–82, 2011.

[53] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "A review of middleware for networked robots," *Int. J. Computer Sci. and Netw. Security*, vol. 9, no. 5, pp. 139–148, 2009.

[54] A. Elkady and T. Sobh, "Robotics middleware: A comprehensive literature survey and attribute-based bibliography," *J. Robot.*, 2012.

[55] H. Utz, S. Sablatnog, S. Enderle, and G. Kraetzschmar, "Miro-middleware for mobile robot applications," *IEEE Trans. Robot. Autom.*, vol. 18, no. 4, pp. 493–497, 2002.

[56] H. Bruyninckx, P. Soetens, and B. Koninckx, "The real-time motion control core of the Orocos project," in *Proc. 2003 IEEE Int. Conf. Robot. and Autom. (ICRA 2003)*, vol. 2. Piscataway, NJ, USA: IEEE, 9 2003, pp. 2766–2771.

[57] T. H. Collett, B. A. MacDonald, and B. P. Gerkey, "Player 2.0: Toward a practical robot programming framework," in *Proc. Australasian Conf. Robot. and Autom. (ACRA 2005)*. Sydney, Australia: ARAA, 2005, p. 145.

[58] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, Kobe. Piscataway, NJ: IEEE, 2009.

[59] T. Kosar, P. E. Martı, P. A. Barrientos, M. Mernik *et al.*, "A preliminary study on various implementation approaches of domain-specific language," *Inform. Softw. Techn.*, vol. 50, no. 5, pp. 390–405, 2008.

[60] S. MacConnell, *Code complete: A practical handbook of software construction*. Redmont, WA, USA: Microsoft Press, 1993.

[61] L. Prechelt, "An empirical comparison of seven programming languages," *IEEE Computer*, no. 10, pp. 23–29, 2000.

[62] S. Magnenat, P. Rétornaz, M. Bonani, V. Longchamp, and F. Mondada, "ASEBA: A modular architecture for event-based control of complex robots," *IEEE/ASME Trans. Mechatronics*, vol. 16, no. 2, pp. 321–329, 2011.

[63] C. Pinciroli, A. Lee-Brown, and G. Beltrame, "Buzz: An extensible programming language for self-organizing heterogeneous robot swarms," *arXiv:1507.05946*, 2015.

[64] Y. K. Lopes, S. M. Trenkwalder, A. B. Leal, T. J. Dodd, and R. Groß, "Supervisory control theory applied to swarm robotics," *Swarm Intell.*, vol. 10, no. 1, pp. 65–97, 2016.

[65] M. A. Ertl and D. Gregg, "The structure and performance of efficient interpreters," *J. Instruction-Level Parallelism*, vol. 5, pp. 1–25, 2003.

[66] G. Beltrame and A. Dentinger, "BittyBuzz repository," https://github.com/MISTLab/BittyBuzz, 03 2019, accessed on: 31-March-2019.

[67] J. C. Baillie, "Urbi: A universal language for robotic control," *Int. J. Humanoid Robot.*, pp. 7–29, 2004.

[68] M. O. Farooq and T. Kunz, "Operating Systems for Wireless Sensor Networks: A Survey," *Sensors*, vol. 11, no. 6, pp. 5900–5930, 2011.

[69] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An Operating System for Sensor Networks," in *Proc. Ambient Intell.* Berlin, Germany: Springer, 2005, pp. 115–148.

[70] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Proc. 2004 IEEE Int. Conf. Local Comput. Netw.* Piscataway, NJ, USA: IEEE, 2004, pp. 455–462.

[71] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intell.*, vol. 7, no. 1, pp. 1–41, 2013.

[72] L. Bayındır, "A review of swarm robotics tasks," *Neurocomputing*, vol. 172, pp. 292–321, 2016.

[73] M. Gauci, J. Chen, W. Li, T. J. Dodd, and R. Groß, "Self-Organised Aggregation without Computation," *Int. J. Robot. Res.*, vol. 33, no. 8, pp. 1145–1161, 2014.

[74] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[75] M. Gauci, J. Chen, W. Li, T. J. Dodd, and R. Groß, "Clustering Objects with Robots That Do Not Compute," in *Proc. 12th Conf. Auton. Agents and Multi-Agent Syst. (AAMAS 2014)*. Richland, SC: IFAAMS, 2014, pp. 421–428.

[76] F. Ducatelle, G. A. Di Caro, C. Pinciroli, and L. M. Gambardella, "Self-organized cooperation between robotic swarms," *Swarm Intell.*, vol. 5, no. 2, p. 73, 2011.

[77] A. R. Shirazi and Y. Jin, "A Strategy for Self-Organized Coordinated Motion of a Swarm of Minimalist Robots," *IEEE Trans. Emer. Top. Comput. Intell.*, vol. 1, no. 5, pp. 326–338, 2017.

[78] J. Chen, M. Gauci, W. Li, A. Kolling, and R. Groß, "Occlusion-Based Cooperative Transport with a Swarm of Miniature Mobile Robots," *IEEE Trans. Robot.*, vol. 31, no. 2, pp. 307–321, 2015.

[79] G. Valentini, E. Ferrante, H. Hamann, and M. Dorigo, "Collective decision with 100 Kilobots: Speed versus accuracy in binary discrimination problems," *Auton. Agents and Multi-Agent Sys.*, vol. 30, no. 3, pp. 553–580, 2016.

[80] Q. Li, X. Yang, Y. Zhu, and J. Zhang, "Self-organized Task Allocation in a Swarm of E-puck Robots," in *Chin. Intell. Autom. Conf.* Berlin, Germany: Springer, 2017, pp. 153–160.

[81] D. Tarapore, A. L. Christensen, and J. Timmis, "Generic, scalable and decentralized fault detection for robot swarms," *PloS one*, vol. 12, no. 8, p. e0182058, 2017.

[82] G. Kapellmann-Zafra, "Human-Swarm Robot Interaction with Different Awareness Constraints," Ph.D. dissertation, The University of Sheffield, 2017.

[83] A. Reina, A. J. Cope, E. Nikolaidis, J. A. R. Marshall, and C. Sabo, "ARK: Augmented Reality for Kilobots," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1755–1761, 7 2017.

[84] M. S. Couceiro, "An overview of swarm robotics for search and rescue applications," in *Handbook of Research on Design, Control, and Modeling of Swarm Robotics*. IGI Global, 2016, pp. 345–382.

[85] M. Schwager, B. J. Julian, M. Angermann, and D. Rus, "Eyes in the sky: Decentralized control for the deployment of robotic camera networks," *Proc. IEEE*, vol. 99, no. 9, pp. 1541–1561, 2011.

[86] A. F. Winfield, "Foraging robots," in *Encyclopedia of complexity and systems science*. Berlin, Germany: Springer, 2009, pp. 3682–3700.

[87] E. Sakthivelmurugan, G. Senthilkumar, K. Prithiviraj, and K. T. Devraj, "Foraging behavior analysis of swarm robotics system," in *Proc. Int. Conf. Res. Mech. Eng. Sci. (RiMES 2017)*, vol. 144. Les Ulis, France: EDP Sciences, 2018, p. 01013.

[88] B. Steux and O. E. Hamzaoui, "tinySLAM: A SLAM algorithm in less than 200 lines C-language program," in *Proc. 2010 IEEE Int. Conf. Contr. Autom. Robot. Vision*. Piscataway, NJ, USA: IEEE, 12 2010, pp. 1975–1979.