Usability Squared: Principles for doing good systems research in robotics

Soham Sankaran
Cornell University
Email: soham@soh.am

Ross A. Knepper
Cornell University
Email: rak@cs.cornell.edu

***Abstract*—Despite recent major advances in robotics research, massive injections of capital into robotics startups, and significant market appetite for robotic solutions, large-scale real-world deployments of robotic systems remain relatively scarce outside of heavy industry and (recently) warehouse logistics. In this paper, we posit that this scarcity comes from the difficulty of building even merely functional, first-pass robotic applications without a dizzying breadth and depth of expertise, in contrast to the relative ease with which non-experts in cloud computing can build complex distributed applications that function reasonably well. We trace this difficulty in application building to the paucity of good systems research in robotics, and lay out a path toward enabling application building by centering usability in systems research in two different ways: privileging the usability of the abstractions defined in systems research, and ensuring that the research itself is usable by application developers in the context of evaluating it for its applicability to their target domain by following principles of realism, empiricism, and exhaustive explication. In addition, we make some suggestions for community-level changes, incentives, and initiatives to create a better environment for systems work in robotics.**

I. Introduction

A. The deployment gap in robotics

Robotics is a fast-growing, multidisciplinary field with applications that are quickly leaping off the pages of science fiction into present day reality. The rapid spread of cheap, powerful mobile phones and their attendant sensor hardware, as well as the meteoric rise in the efficacy of machine learning techniques for perception, planning, control, and related problems, have left us on the cusp of an unprecedented golden age in robotics work, both in academia and industry.

We remain, however, *just on the cusp* of that golden age. Self-driving cars are being tested all over the US but remain controversial and continually delayed, major equipment manufacturers and startups alike make loud noises about agricultural robots coming any day now, and people expectantly wait for their next package to be delivered by drone. Meanwhile, nothing much has really changed out in the field. The successful robots of the past three decades — the industrial arms, the KIVA/Amazon Robotics warehouse robots, the iRobot vacuum cleaners — continue to chug along, but successful new deployments, in particular those that create inarguable value, have been elusive [6].

To some degree, this slow growth is to be expected. Robotics is hard. The physical world is riddled with inherent complexities, and many disparate strands of knowledge must be woven together to form a robotic system that does anything interesting, to say nothing of the exponential complexity of settings involving multiple robots coordinating with each other.

Despite the difficulty, the last decade has seen an unprecedented boom in both the founding and funding [18, 50] of companies intending to bring robotics into wider use across a dizzying array of industries. Heartbreakingly, the vast majority of these companies have failed, including a number of high-profile examples run by luminaries from the robotics research community [47]. In particular, most of these companies appear to have failed not due to an inability to build the technology, but due to a failure to achieve product-market fit [15, 41] — they were selling something that people didn't want.

The founders of these companies were, by and large, roboticists rather than experts in the domains their companies were targeting. Their mode of failure suggests that the people who are best equipped to drive innovative new use cases for robotic technology are domain experts (in application domains like agriculture, healthcare, or construction, for example), since they are most able to anticipate the needs of their particular market.

B. Application building

In order to allow experts in application domains to drive the real-world adoption of robotics, we must enable *fast application building*. An application here is a working end-to-end robot system artifact that can perform a task reliably and repeatedly in a real-world domain. The application does not have to be optimal or even very performant: it merely needs to work well enough to function as practical evidence for the utility of a robotic solution in the target domain.

Currently, application developers in robotics need to be deeply steeped in some mishmash of kinematics, dynamics, path planning, distributed systems, queuing theory, electrical engineering, and more. While one can certainly attempt to distribute the burden of knowledge across a team, at least one individual must have a working background in enough of the pieces to keep the whole thing together. Indeed, without such an individual it is nigh-impossible to figure out what kinds of problems are even tractable with current robotic technology.

There are very few of these full-stack roboticists, and their scarcity limits the total number of attempts at building real-world robotic applications per unit time to a very small number. More attempts in a diverse set of application domains

using distinct approaches, especially led by domain experts, would likely lead to more successes in real-world deployment. As such, it is incumbent on us as a field to prioritize enabling application building by technically sophisticated engineers who are not full-stack roboticists. This is where systems research comes in.

C. The role of systems research

The purpose of systems research, broadly defined, is to wrap complex machinery in human-usable abstractions that enable non-experts to build performant applications without having to understand every detail of the implementations underneath the interfaces they build on top of.

This is accomplished in a two-step process: first, the design and specification of usable abstractions that provide easy to reason about interfaces and guarantees for specific tasks, and second, the iterative refinement of implementations underneath these abstractions that provide better and better performance on whatever metrics the user cares about without breaking the abstraction's contract (though in practice these steps can be ordered in the opposite way, and they usually bleed into each other). Good systems work bridges the gap between abstract insight and real-world use cases by presenting just-right "Goldilocks" abstractions that are simultaneously simple enough to understand and use, powerful enough that real world applications can be built on top of them, and only loosely (if at all) tied to the vagaries of a specific implementation such that different implementations can be swapped in and swapped out under the abstraction layer.

An exemplar of a field where this is done right is distributed systems, which forms the academic foundation for cloud computing. Despite the dizzying array of hardware, software, and even fundamental physics concepts involved, anyone with basic computer science background can quickly learn to build and deploy a fairly complex distributed web application that scales to hundreds of thousands of users out of the box. It's as simple as writing an HTTP application on top of your favourite backend framework (Flask, Express, Revel) in your favourite language (Python, NodeJS, Go), interfacing it with a newly spun-up instance of the appropriate type of (relational, key-value) datastore (PostgreSQL, Redis), placing it behind a server (NGINX, Apache), and then just letting it run on a virtual machine. If you want to scale, you can replicate and/or shard your database, stick it behind a cache, automate the spinning up of more VMs for the application, stick that behind a load balancer, and so on. Every choice mentioned here turns on a small set of important tradeoffs, for example consistency vs. availability in the presence of network partitions and the richness of the query model vs latency for datastores, and can be made based on desired properties and workload assumptions for the system. What if you get it wrong? You need only to measure where you're deviating from your assumptions, revisit your tradeoffs, make some different choices, and redeploy.

This process is not trivial, though the proliferation of battle-tested hosted versions of all of the pieces involved by Google, Amazon, et al. has taken a lot of the pain out of it, but it isn't rocket science. Is it going to produce the **optimal** solution? No. Is it going to produce something **usable**? Very likely yes. Does it enable the creation of real-world applications that, but for the existence of systems that can compose in this usable way, would never have existed? Unquestionably yes. Done and working is better than vacuously perfect.

Aside from some reasonably healthy pieces of the ROS ecosystem (primarily authored and maintained outside of academia at places like Willow Garage, Clearpath, and OSRF) this sort of application building is very, very difficult, if not nearly impossible, in robotics today. Someone who wants to build their own serviceable (not even close to optimal or state-of-the-art) Amazon-style warehouse logistics application, for example, would likely not be able to do so without expert advice up and down the stack, despite the individual pieces of technology to do so being broadly within reach.

D. A way forward: Usability Squared

We believe that in order to enable application building, we must center usability in robotics research in two different ways:

1) Usable abstractions: Systems research in robotics must prioritize designing abstractions that are intuitive for non-expert application developers to reason about and straightforward to use in building applications.

2) Usable research papers: While providing usable abstractions is essential, the research paper itself must also be usable in the sense of being accurately evaluable for its utility in a given target domain by a non-expert application developer. This involves using research methodology that privileges realism, empiricism, and exhaustive explication to demonstrate that the design choices made in the work and the tradeoffs exposed by the abstractions specified are the right ones for the domain or domains the paper is aimed at.

In other words, we believe that the job of good systems research is to design and specify usable abstractions that are both *intuitive* and *powerful*, and the job of a good systems research paper is to validate the design choices made in specifying the abstraction and building its underlying implementation, thus making the research itself usable to an application developer. Both kinds of usability are essential in good systems work — without either, the utility of the work in the real world is compromised.

In the next two sections, we justify and elaborate on these two forms of usability.

II. Usable abstractions

If forced to choose, privilege the usability of the abstraction, in particular the intuitiveness of the interface to application developers and compositionality with other systems, over squeezing out the last drops of performance from the system or proving optimality. While it is often possible to squeeze greater performance out of less simple and intuitive abstractions, the increased complexity and resultant cognitive load generated often massively reduces the ability of application developers to

easily make use of and compose them, thus preventing building applications that work correctly or, indeed, exist at all.

For an example of this phenomenon in action, we turn to distributed datastores, an area which recently witnessed the rise [4, 26] and decline [33, 36] of eventual consistency. Eventual consistency [48] promises better performance in the form of lower query latency and higher availability for distributed datastores via the mechanism of reducing the coordination required for each query. The tradeoff here is that the global state of the datastore is not guaranteed to be consistent — this roughly means that if you write something to it, it will eventually be visible globally, but that is only guaranteed to happen as time goes to infinity. In the meantime, you may see inconsistent state in the system, with different reads returning conflicting values. This model is a significant departure from the strongly-consistent ACID (Atomicity, Consistency, Isolation, and Durability) semantics [24] of classical databases.

ACID semantics and strong consistency roughly align with what human programmers intuitively expect from a datastore, and they simplify writing correct applications on top of systems that guarantee them [19]. Eventual consistency sacrifices that abstraction simplicity for performance. While eventual consistency datastores became quite popular in the late 2000s and early 2010s, they fell from grace because the tradeoff eventually came to be seen as not worth it [54]. Here's a representative quote from Google's paper about F1 [42], their strongly-consistent high performance datastore for ads:

> "We [also] have a lot of experience with eventual consistency systems at Google. In all such systems, we find developers spend a significant fraction of their time building extremely complex and error-prone mechanisms to cope with eventual consistency and handle data that may be out of date. We think this is an unacceptable burden to place on developers and that consistency problems should be solved at the database level."

Robotics is still a young field. When there are many more real-world deployments and application developers experienced in the basics of building robotic applications, we can start profitably experimenting with increasing the complexity of our abstractions, but until people are able to reliably use the simple stuff, this will be actively harmful. Indeed, without first optimizing for usability, we may never have enough data from application domains to even know what the quantitative metrics and use cases we should optimize for even are.

III. Usable systems research papers

An abstraction specified in new research work, no matter how usable in design and performant in implementation, can only be used by an application developer if they can confirm its applicability to their target domain by validating that the assumptions and design choices made correspond with the ground realities of a real-world deployment in their target domain, and that the tradeoffs exposed are the appropriate ones for said domain.

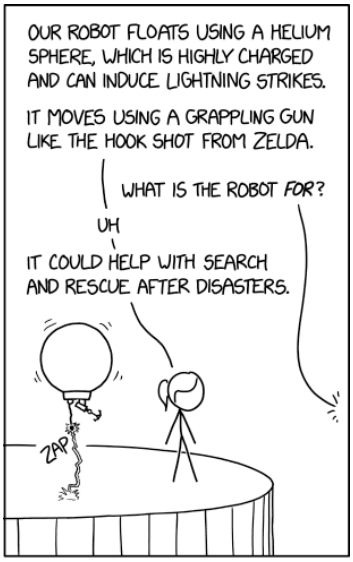

Fig. 1: xkcd: New Robot by Randall Munroe (license: CC BY-NC 2.5) [34]

In the spirit of Butler Lampson's classic *Hints for Computer System Design* [30], we propose a few methodological principles in service of ensuring this second kind of usability in systems research.

A. Principle 1: Target at least one specific real-world domain

Good systems work comes from real-world problems. There is a great deal of work in robotics that seems to tack on an application domain as an afterthought, as the classic XKCD comic in Figure 1 [35] illustrates.

Having at least one real-world application domain ensures that at least application developers targeting that domain can use the work. In addition, having a concrete domain to compare against allows application developers targeting other domains to more easily evaluate the utility of the work for their domain.

B. Principle 2: Make realistic assumptions and avoid unnecessary, unrealistic, and fanciful assumptions

Good systems work is informed by real-world constraints. The assumptions that underlie systems research must align with the circumstances of (realized or hypothetical) real deployments of the application domain or domains the research is targeted at.

1) Necessary, realistic assumptions: In order for research work to be applicable to a real-world domain, it must make assumptions that are fundamental to the operation of that domain, without which the work would be unrealistic and inapplicable.

An important example of a necessary realistic assumption is that real robotic applications are *always on and never stop*. A substantial portion of the power of robotic autonomy comes from its ability to facilitate the smooth, uninterrupted running

of processes 24x7, and many existing and potential robotic applications don't (or won't) have a neatly-defined end state — they would ideally just keep going, moving packages, building cars, and tending to fields until the end of time. Stopping, even for a few seconds, can be disastrously expensive. As such, it would behoove systems research that targets always-on domains to optimize for this assumption when possible.

Consider path planning. Classical path planning algorithms like A* do single-query one-shot planning of the whole path. In practice, real robots in always-on domains perform an iterative process of planning and replanning toward as they are given new goals within a somewhat but not maximally dynamic environment. Performing planning from scratch at every update cycle discards potentially useful state from the computation of prior plans. In the mid-to-late 2000s, there was a burst of research work on iterative multi-query path planning and "anytime" planning [43, 46] that sought to exploit this potentially useful state for faster and more optimal planning. It would make sense for systems that do path planning in quasi-dynamic environments to use these algorithms to, say, harness redundancies between the iterations to reduce average-case plan-update latency, which is the more important metric than worst-case cold-start plan-creation latency for always-on domains. For whatever reason, systems research in robotics tends to ignore this work, instead sticking with one-shot path planning techniques.

Perhaps in part due to this lack of uptake in systems research and real deployments, there is disproportionately little new work in this area relative to offline planning.

2) Fanciful assumptions: Fanciful assumptions are assumptions about the target domain, usually taking the form of very specific constraints, that are not supported by the ground realities of that domain.

In multirobot systems work, there are a huge number of papers that focus on coordination given some specific, often unique unreliable communication model [11, 20, 53]. In almost all domains we care about, it is either possible to get quite reliable communication, for example by combining services from two consumer mobile broadband providers to get 99.999% connection availability [5], or it is not possible to get communication at all, for example in RF-denied nuclear disaster zones or in the deep sea. Non-military use cases requiring the use of some kind of ad-hoc mesh networking are largely limited to the exploration of caves and space, which collectively comprise a relatively small proportion of the domains that exist today. There is still a lot of work to do be done in domains where communication is reliable — we have by no means solved multirobot coordination under those models — but these much more realistic problems are often ignored. Assumptions like these should be strongly avoided.

3) Beguiling assumptions that seem necessary but aren't: There is a class of beguiling assumptions that are simple, intuitive, and seemingly useful, but in practice, at best, unnecessary and, at worst, actively harmful. Consider the assumption of deadlock-freedom in multi-agent path planning for warehouse domains. While it may seem entirely reasonable to want to guarantee that agents never deadlock, this guarantee is almost impossible without using totally centralized global planning, which severely limits scalability, and, crucially, is almost never a problem in practice — companies using systems with no deadlock freedom guarantee see deadlock on the order of a few times a year even in very large deployments [40], and at that rate of occurrence it is better to simply have humans reset one of the robots after a timeout.

C. Principle 3: Avoid irrelevant proofs and guarantees that are useless in practice

Roboticists have a distinct affinity for theoretical proofs, even within systems work. While proofs of useful properties can certainly be beneficial in providing guarantees that make systems abstractions more usable, this fixation on proofs can be harmful in two ways:

1) If it slows down or prevents the publication of a practical contribution that can be empirically validated
2) If a provable guarantee that is actually irrelevant clouds understanding of what metrics really matter and thus prevents the exploration of potentially profitable research directions

A good example of the second phenomenon can be found in the literature around probabilistic, sampling-based planners such as the Probabilistic Roadmap (PRM) [28] and the Rapidly-exploring Random Tree (RRT) [31] planning algorithms. These motion planners rely on proofs of eventual probabilistic completeness that guarantee that some solution will be found as time goes to infinity. In practice, no robot has infinite time to wait, so it is common tradecraft to run RRT, for example, with a series of timeout-based restarts with the hope that different samples will produce a plan quicker. These restarts are seldom included in evaluations of systems using RRT, as noted by Wedge and Branicky [51] in their excellent analysis of plan time distributions and restarts, and if they're mentioned at all it's perfunctory and not particularly well-explained, such as in this quote from the Forage-RRT paper [29]:

> "Moreover, any RRT reaching 10,000 nodes was restarted to improve the average planning time of all planners (empirically when an RRT grows too large, it will have trouble connecting to the goal, so it is better to restart)."

For an application developer attempting to evaluate planners and planning systems, this sort of opacity around a crucial aspect of real-world deployment confounds their ability to make reasonable choices for their domain.

In addition, this sort of obfuscation may well harm future research in this area. There might be a motion planner that, for example, does not guarantee probabilistic completeness, but for all the domains we care about produces results faster than RRT (when tested empirically). In the current research paradigm, this planner's real-world superiority to RRT might not ever come to light.

In general, the existence of some pervasive tradecraft secret like planner restarts that goes mostly unmentioned or unevaluated in the literature is a good heuristic for detecting that some guarantee being provided is unrealistic or useless — there is usually an opening for good systems work to be found in these situations.

D. Principle 4: Explicitly justify design choices with reference to counterfactual designs

Given that the point of a systems paper is to justify the design choices made in the research described, it is essential to explicitly consider counterfactual options, the roads not taken, to justify why the choices made were the correct ones. This not only helps evaluate work in comparison with related work in the area, and in general better validate the reasoning behind design choices, but also helps the application developer distinguish design choices that are essential to ensure the proper functioning of the system from design choices that can be safely modified depending on the specifics of a particular domain or implementation. This can be the difference between an application developer incorrectly seeing some research as incompatible with their domain and that same developer profitably using the core ideas of the work while modifying things on the periphery to achieve compatibility with their domain.

Computer architecture papers often do a very good job of this kind of design space exploration and justification. Here's a quote from the abstract of the Q100 paper, which proposes an architecture for a specialized Database Processing Unit (DPU) (Wu et al. [52]):

> "This work explores a Q100 design space of 150 configurations, selecting three for further analysis: a small, power-conscious implementation, a high performance implementation, and a balanced design that maximizes performance per Watt. We then demonstrate that the power-conscious Q100 handles the TPC-H queries with three orders of magnitude less energy than a state of the art software DBMS, while the performance-oriented design outperforms the same DBMS by 70X."

The graphs in Figure 2, which are Figures 3, 4, and 5 from Wu et al. [52], detail some of the analyses they ran as part of their design space exploration in which they vary the number and connectivity of various component types.

The graph in Figure 3, which is Figure 6 from Wu et al. [52], charts the performance relative to power consumption of their 150 different configurations, highlighting the three they chose for further study.

E. Principle 5: Make your tradeoffs explicit and empirically explore the tradeoff space

The design choices made by systems will fix some set of parameters and expose other sets of parameters as tunable tradeoffs. These tradeoffs, which act as knobs that application developers can twiddle, need to be explicitly highlighted, motivated, and empirically explored in order to allow for evaluation with respect to real-world conditions in a target domain.

An example of a tradeoff exposed in robotics is discretization granularity. Discretizing space is a common strategy for path planning, especially in the multirobot domain [2, 27, 49]. Discretization represents a set of tradeoffs against plan-optimality in the real (continuous) world, including speed of computation, simplicity of algorithm, and ease of implementing occupancy-based safety guarantees. These tradeoffs are rarely explicated in direct ways, and the tradeoff space is almost never empirically explored in order to justify the design decisions made and parameters selected. How does finer and finer discretization affect compute time in various realistic settings? How close to optimal do you get with reasonably granular discretization? Is there some optimal point on the tradeoff graph where the computation is quick enough for use on real robots and the solutions generated are close to optimal, with diminishing returns for finer granularity? These questions are incredibly relevant for both real-world use and future research directions, but are almost never answered.

An example of this in action in database systems is the fundamental tradeoff between strength of consistency guarantees and query latency, as explicated by Dan Abadi in his PACELC principle [1], which is an extension of Eric Brewer's CAP theorem [9, 10] that famously limits a distributed datastore to two out of the three of strong Consistency, constant Availability, and Partition tolerance. Database systems such as Cassandra allow you to choose different levels of consistency [38], with higher levels resulting in higher latency, and these systems have been subject to comprehensive of studies of performance along the tradeoff axis [25].

F. Principle 6: Test till you break, scale till you fail

Don't just publish a graph demonstrating linear performance scaling for, say, a multirobot path planning system from 1 to 20 robots — graph your results past the point where you stop being able to scale. This is incredibly important for pointing to where future work needs to improve, and it helps people using your system bracket the range of reasonable performance for their needs. Database papers often do this particularly well, with Figure 4, which is Figure 7 from the SOSP 2013 Silo paper [45], serving as a good example.

The graph demonstrates that latency performance is roughly unchanged as you scale the number of worker threads until a massive spike past 28 threads. This result allows an application developer working in, for example, a target domain with a requirement for latency lower than 200ms to decide whether Silo is appropriate for them based on whether they need greater or fewer than 28 threads to satisfy their throughput requirements.

G. Principle 7: Use or make open, public benchmarks and baselines, ideally based on real-world workloads.

The machine learning, computer systems, and databases communities make rapid progress due to the existence of open datasets (ImageNet [17]), load generation tools (YCSB

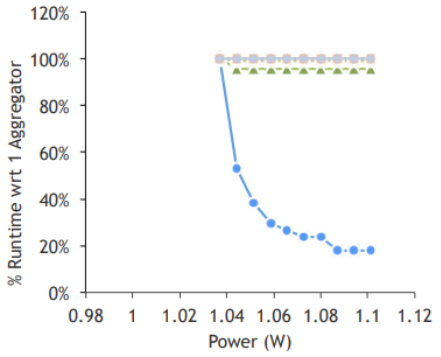

Figure 3. Aggregator sensitivity study shows that Q1 is the only query that is sensitive to number of aggregators, and its performance plateaus beyond 8 tiles.

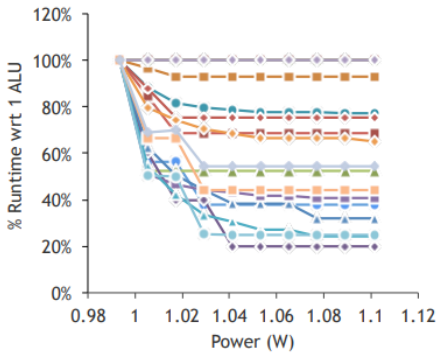

Figure 4. ALU tiles are more power hungry than aggregators, but adding more ALUs helps most query's performance. This tradeoff necessitates an exploration of the design space varying number of ALUs.

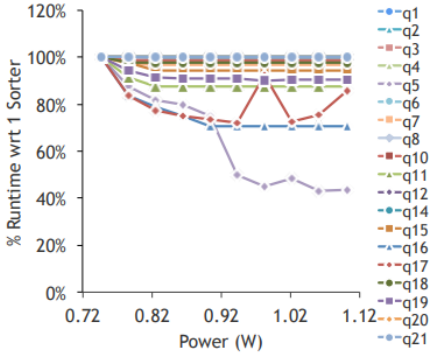

Figure 5. Sorter tiles are the most power hungry, dissipating almost 40 mW per tile. Q17 exhibits a corner case where the scheduler makes bad decisions causing performance to degrade as number of sorters increase.

Fig. 2: Design space exploration graphs from Wu et al. [52]

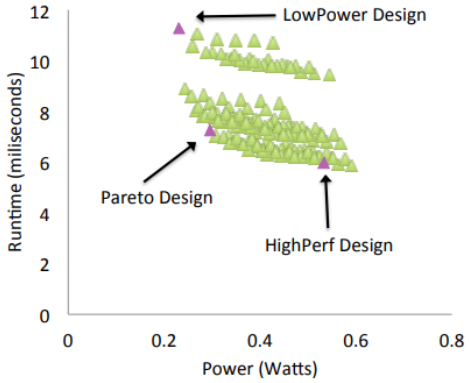

Figure 6. Out of 150 configurations, we selected three designs for further evaluation: LowPower for an energy-conscious configuration, HighPerf for a performance-conscious configuration, and Pareto for a design that maximizes performance per Watt.

Fig. 3: Performance relative to power consumption of 150 different configurations from Wu et al. [52]

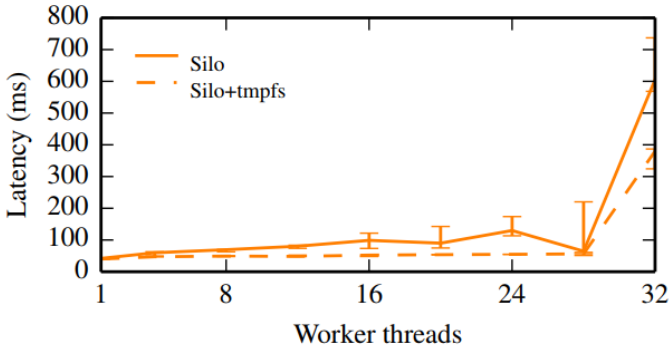

Figure 7: Silo transaction latency for TPC-C with logging to either durable storage or an in-memory file system.

Fig. 4: Transaction latency graph from Tu et al. [45]

[12]), and benchmarks (TPC-C [32]). In addition, in the latter two communities, it is common to try and model real-world workloads or replicate them exactly using traces sourced from industry [14, 39].

Systems work in robotics often relies on either *proof by video*, where a cherry-picked short sample of footage of the system in action is used as evidence of efficacy, at worst, or homespun ill-defined metrics at best. This is insufficient. Without common benchmarks and baselines, it is impossible to evaluate work relative to other work in a principled manner, and for systems work, this means that it is effectively impossible to divine the correctness of the design choices made in any specific paper. Aside from overlap with the machine learning and computer vision communities (from which datasets like KITTI [23] have emerged), robotics largely does not produce or utilize such benchmarks [16], with the exception of somewhat infrequent and inconsistently targeted competitions [3]. This is true even for problems where there should be strong incentives to make apples-to-apples comparisons, including path planning (single and multirobot) and task allocation for multirobot systems. There is some evidence that this state of affairs is changing, with the help of work like Nathan Sturtevant's planning benchmarks [44], but it is not changing quickly enough.

In particular, the use of end-to-end benchmarks based on simulated workloads that model real-world deployments, for example the Asprilo warehouse logistics problem generator

[21] developed by Gebser et al [22], would greatly improve the ability of application developers to judge the relevance of research to their target domain.

H. Principle 8: Exhaustive explication — no tricks up your sleeve — to enable replication

As noted above, robotics research is plagued by the problem of proofs by video. It is often impossible to replicate these videos in academic settings, where some system flakiness is tolerable, much less in the high-reliability low-error-tolerance world of industry. This lack of replicability stems from the many undocumented patches, hacks, and simplifying assumptions used to get a robotic system to run that are passed down only via intra-lab oral tradition, if at all, and it fatally destroys the value of the research by preventing both further academic exploration and validation by independent groups as well as any kind of real-world deployment. It is thus incumbent on systems researchers to ensure that any work they produce is reproducible. There has been much recent discussion of the reproducibility crisis in robotics and methodological principles for conducting reproducible research in our field [7, 8], and the first (though by no means the last) step towards reproducibility is to document every detail of how the system was made to work, from physical specifications to operating system versions to lighting conditions. While conference papers have space constraints, it is easy to put a supplementary document on arXiv or a similar archival service.

This principle very much does not supplant or subsume all the work being done on reproducibility in the sciences and engineering in general, and in robotics in particular. We recommend anyone working on robotic systems keep abreast of the latest best practices in this area.

IV. Community-level suggestions for promoting good systems work

While we certainly hope that individual research papers following the principles outlined will lead to better work overall, the overall paucity of good systems research in robotics can only be rectified by consistent community-level acceptance and encouragement of this work. While the major robotics conferences pay lip service to wanting more systems work, explicitly welcoming and endowing awards for submissions of this type, word on the street is that the work is often not seen as core research, and the standards for inclusion are often inconsistent. People have very different and often conflicting opinions of what a systems research project or a systems paper even is. If we want to enable application building, we must make space within the community for the systems work that forms the foundation for it.

In the hope of beginning that process, we present a set of community-level suggestions for creating a better environment for good systems work.

A. Suggestion 1: Recognize that finding a useful new way of structuring a problem is a first-class research contribution

The bread and butter of systems work is reorganizing a problem around some key insight or set of insights such that it becomes more tractable. Once this is accomplished, the actual solution may seem easy to conceptualize and implement. This should not serve to diminish the validity of the research contribution of the work — indeed, one hallmark of great systems work is that it repurposes preexisting abstractions and components to solve new problems, reducing redundant work and exposing fundamental shared structure. It is easy to dismiss the value of restructuring a problem domain, and this impulse must be fought vigorously.

B. Suggestion 2: Value "incremental" systems work

Work that builds on pre-existing research in ways that are not revolutionary but still substantive, improving performance on the same benchmarks and baselines or providing some new useful feature, is necessary and important, both for ensuring that exciting work from academia reaches the threshold of real-world acceptable performance and for ensuring that new work that purports to be a revolutionary is not worse than previous generation work with incremental modifications. If the latter situation is not detected because no one ever bothered to build the optimizations, then researchers might go down suboptimal research paths due to the incorrect belief that the prior pathway could never reach the performance that the new one does.

In computer systems and databases, there is a strong tradition of work that preserves the same programmer-facing abstraction (or very close to it) while changing the implementation in intelligent ways to significantly improve performance. The methods behind these sorts of improvements must be recognized as first-class research contributions to ensure that work like this can exist in robotics.

C. Suggestion 3: Incentivize the creation of open, public benchmarks and workload datasets

As noted in the principles, it is essential that researchers test their systems on public benchmarks. As a community, we should incentivize the creation of such benchmarks and the publication of workload datasets with guaranteed publication slots, special awards, and cold, hard cash via the creation of some kind of benchmark fund or prize. These carrots should be complemented by the stick of subjecting to strict scrutiny and possible publication denial papers that either don't use appropriate common benchmarks, don't release their own benchmarks, or both.

D. Suggestion 4: Incentivize frontier-illuminating papers — systems papers that try to build some arbitrary application in a principled way using state-of-the-art research

A combination of the fractal nature of robotics as a field and the preponderance of work that doesn't follow the principles outlined above makes it hard to see the frontier of our ability to build systems that solve any specific real-world problem. Competitions [3] like the Amazon Picking Challenge [13] have been the primary method of illuminating these frontiers, but have the downside of being infrequent and industry-controlled. As a result, we believe that we should build a program of practical frontier papers where groups are funded to solve a

rotating set of real-world problems — for example clearing a cluttered home or the monitoring and harvesting of a specific crop in a greenhouse — using existing research, then report on their methods and results on a set of predefined open benchmarks. This sort of work will also serve to illuminate the practical frontiers of the constituent subproblems of these real-world problems, tying performance to real-world workloads.

The Integrated Intelligence for Human-Robot Teams paper by Oh et al. [37] is a good model for this kind of work.

V. Conclusion

We believe that making space for more and better systems research that follows the principles we've outlined will not just help us make progress toward the goal of seeing robots widely deployed in the real world, but also benefit academia immensely by providing new problems motivated by experiences that application developers have in their particular domains, opening up a large pool of potential new sources for funding, and motivating a larger and more diverse set of people to work in robotics after seeing it in their daily lives. This has been the experience of research communities in other computer science and engineering fields, most prominently of late in machine learning.

There are millions of potential application developers out there waiting for us to unlock their ability to reimagine the world we live in. We are in the pre-Apple II days of robotics, tooling away with our expensive research toys with only baroque industrial deployments to prove the worth of our field. It is our duty to bring the power of robotics to the people, and we can scarcely imagine the depth of the ingenuity that doing so will reveal.

Acknowledgments

Thanks to Wil Thomason and Dylan A. Shell for their participation in initial discussions leading up to this paper, as well as to Natacha Crooks, Adrian Sampson, Chris De Sa, and A. Feder Cooper for suggesting systems papers worth referencing.

In addition, we'd like to thank Christopher Leet for his input on the structure of the paper, as well as Wil Thomason, Tom Magrino, Sowmya Dharanipragada, Danny Adams, Alexa VanHattum, Kate Donahue, Gregory Yauney, and A. Feder Cooper for reading drafts of it.

This paper is based on work partly supported by the National Science Foundation under Grant No. 1646417. We are grateful for this support.

References

[1] Daniel Abadi. Consistency tradeoffs in modern distributed database system design: Cap is only part of the story. *Computer*, 45(2):37–42, 2012.

[2] Rachid Alami, Sara Fleury, Matthieu Herrb, Félix Ingrand, and Frédéric Robert. Multi-robot cooperation in the martha project. *IEEE Robotics & Automation Magazine*, 5(1):36–47, 1998.

[3] John Anderson, Jacky Baltes, and Chi tai Cheng. Review: Robotics competitions as benchmarks for ai research. *Knowl. Eng. Rev.*, 26(1):11–17, February 2011. ISSN 0269-8889. doi: 10.1017/S0269888910000354. URL http://dx.doi.org/10.1017/S0269888910000354.

[4] Peter Bailis, Shivaram Venkataraman, Michael J Franklin, Joseph M Hellerstein, and Ion Stoica. Probabilistically bounded staleness for practical partial quorums. *Proceedings of the VLDB Endowment*, 5(8):776–787, 2012.

[5] Dziugas Baltrunas, Ahmed Elmokashfi, and Amund Kvalbein. Measuring the reliability of mobile broadband networks. In *Proceedings of the 2014 conference on internet measurement conference*, pages 45–58. ACM, 2014.

[6] Jason Bloomberg. Why you should think twice about robotic process automation, Nov 2018. URL https://www.forbes.com/sites/jasonbloomberg/2018/11/06/why-you-should-think-twice-about-robotic-process-automation/.

[7] F. Bonsignorio. A new kind of article for reproducible research in intelligent robotics [from the field]. *IEEE Robotics Automation Magazine*, 24(3):178–182, Sep. 2017. ISSN 1070-9932. doi: 10.1109/MRA.2017.2722918.

[8] F. Bonsignorio and A. P. del Pobil. Toward replicable and measurable robotics research [from the guest editors]. *IEEE Robotics Automation Magazine*, 22(3):32–35, Sep. 2015. ISSN 1070-9932. doi: 10.1109/MRA.2015.2452073.

[9] Eric Brewer. A certain freedom: thoughts on the cap theorem. In *Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 335–335. ACM, 2010.

[10] Eric A. Brewer. Towards robust distributed systems (abstract). In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '00, pages 7–, New York, NY, USA, 2000. ACM. ISBN 1-58113-183-6. doi: 10.1145/343477.343502. URL http://doi.acm.org/10.1145/343477.343502.

[11] Kyle Cesare, Ryan Skeele, Soo-Hyun Yoo, Yawei Zhang, and Geoffrey Hollinger. Multi-uav exploration with limited communication and battery. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 2230–2235. IEEE, 2015.

[12] Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 143–154. ACM, 2010.

[13] Nikolaus Correll, Kostas E Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M Romano, and Peter R Wurman. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, 15(1):172–188, 2016.

[14] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russi-

novich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 153–167, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5085-3. doi: 10.1145/3132747.3132772. URL http://doi.acm.org/10.1145/3132747.3132772.

[15] Steve Crowe. Inside the rethink robotics shutdown, Nov 2018. URL https://www.therobotreport.com/rethink-robotics-shutdown/.

[16] Angel P del Pobil, Rad Madhavan, and Elena Messina. Benchmarks in robotics research. In *IROS Workshop Notes*, 2006.

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[18] Kevin Dowd. Automation takes flight: A look at vc's soaring interest in robotics & drones, Mar 2019.

[19] Emily Drevets. Why acid transactions matter in an eventually consistent world, Aug 2016. URL https://www.oreilly.com/ideas/why-acid-transactions-matter-in-an-eventually-consistent-world.

[20] Yuan Fan, Lu Liu, Gang Feng, Cheng Song, and Yong Wang. Virtual neighbor based connectivity preserving of multi-agent systems with bounded control inputs in the presence of unreliable communication links. *Automatica*, 49(5):1261–1267, 2013.

[21] Martin Gebser, Philipp Obermeier, Thomas Otto, Torsten Schaub, Orkunt Sabuncu, Van Nguyen, and Tran Cao Son. asprilo, 2018. URL https://asprilo.github.io/.

[22] Martin Gebser, Philipp Obermeier, Thomas Otto, Torsten Schaub, Orkunt Sabuncu, Van Nguyen, and Tran Cao Son. Experimenting with robotic intra-logistics domains. *Theory and Practice of Logic Programming*, 18(3-4): 502–519, 2018.

[23] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11): 1231–1237, 2013.

[24] Theo Haerder and Andreas Reuter. Principles of transaction-oriented database recovery. *ACM computing surveys (CSUR)*, 15(4):287–317, 1983.

[25] Gerard Haughian, Rasha Osman, and William J Knottenbelt. Benchmarking replication in cassandra and mongodb nosql datastores. In *International Conference on Database and Expert Systems Applications*, pages 152–166. Springer, 2016.

[26] Pat Helland and David Campbell. Building on quicksand. In *CIDR 2009, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2009, Online Proceedings*, 2009. URL http://www-db.cs.wisc.edu/cidr/cidr2009/Paper_133.pdf.

[27] Wolfgang Hönig, Scott Kiesel, Andrew Tinka, Joseph W Durham, and Nora Ayanian. Persistent and robust execution of mapf schedules in warehouses. *IEEE Robotics and Automation Letters*, 4(2):1125–1131, 2019.

[28] Lydia E Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, 12(4), 1996.

[29] Leo Keselman, Erik Verriest, and Patricio A Vela. Forage rrtan efficient approach to task-space goal planning for high dimensional systems. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1572–1577. IEEE, 2014.

[30] Butler W. Lampson. Hints for computer system design. In *Proceedings of the Ninth ACM Symposium on Operating Systems Principles*, SOSP '83, pages 33–48, New York, NY, USA, 1983. ACM. ISBN 0-89791-115-6. doi: 10.1145/800217.806614. URL http://doi.acm.org/10.1145/800217.806614.

[31] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11, Computer Science Dept., Iowa State University, 1998.

[32] Scott T. Leutenegger and Daniel Dias. A modeling study of the tpc-c benchmark. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, SIGMOD '93, pages 22–31, New York, NY, USA, 1993. ACM. ISBN 0-89791-592-5. doi: 10.1145/170035.170042. URL http://doi.acm.org/10.1145/170035.170042.

[33] Wyatt Lloyd, Michael J Freedman, Michael Kaminsky, and David G Andersen. Don't settle for eventual consistency. *Communications of the ACM*, 57(5):61–68, 2014.

[34] Randall Munroe. xkcd - a webcomic - license, 2012. URL https://xkcd.com/license.html.

[35] Randall Munroe. xkcd: New robot, Mar 2019. URL https://xkcd.com/2128/.

[36] Kannan Muthukkaruppan. The underlying technology of messages, Nov 2010. URL https://www.facebook.com/notes/facebook-engineering/the-underlying-technology-of-messages/454991608919/.

[37] Jean Oh, Thomas M Howard, Matthew R Walter, Daniel Barber, Menglong Zhu, Sangdon Park, Arne Suppe, Luis Navarro-Serment, Felix Duvallet, Abdeslam Boularias, et al. Integrated intelligence for human-robot teams. In *International Symposium on Experimental Robotics*, pages 309–322. Springer, 2016.

[38] Apache Cassandra Project. Apache cassandra 4.0 documentation: Tunable consistency, 2019. URL http://cassandra.apache.org/doc/4.0/architecture/dynamo.html#tunable-consistency.

[39] Alexander Pucher. Cloud traces and production workloads for your research, Jun 2015. URL https://alexpucher.com/blog/2015/06/29/cloud-traces-and-production-workloads-for-your-research/.

[40] Soham Sankaran and Ross A. Knepper. Interviews with engineers from two companies with large multirobot

deployments, 2019.

[41] Ron Schmelzer. Why are robotics companies dying?, Oct 2018. URL https://www.forbes.com/sites/cognitiveworld/2018/10/29/why-are-robotics-companies-dying/.

[42] Jeff Shute, Radek Vingralek, Bart Samwel, Ben Handy, Chad Whipkey, Eric Rollins, Mircea Oancea, Kyle Littlefield, David Menestrina, Stephan Ellner, et al. F1: A distributed sql database that scales. *Proceedings of the VLDB Endowment*, 6(11):1068–1079, 2013.

[43] Omar Souissi, Rabie Benatitallah, David Duvivier, AbedlHakim Artiba, Nicolas Belanger, and Pierre Feyzeau. Path planning: A 2013 survey. In *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM)*, pages 1–8. IEEE, 2013.

[44] N. Sturtevant. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games*, 4(2):144 – 148, 2012. URL http://web.cs.du.edu/~sturtevant/papers/benchmarks.pdf.

[45] Stephen Tu, Wenting Zheng, Eddie Kohler, Barbara Liskov, and Samuel Madden. Speedy transactions in multicore in-memory databases. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 18–32. ACM, 2013.

[46] Jur Van Den Berg, Dave Ferguson, and James Kuffner. Anytime path planning and replanning in dynamic environments. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2366–2371. IEEE, 2006.

[47] Bram Vanderborght. Robotic dreams, robotic realities: Why is it so hard to build profitable robot companies?, Mar 2019. URL https://spectrum.ieee.org/automaton/robotics/industrial-robots/robotic-dreams-robotic-realities.

[48] Werner Vogels. Eventually consistent. *Queue*, 6(6):14–19, 2008.

[49] Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015.

[50] Richard Waters. Rise of the robots is sparking an investment boom, May 2016. URL https://www.ft.com/content/5a352264-0e26-11e6-ad80-67655613c2d6.

[51] Nathan A Wedge and Michael S Branicky. On heavy-tailed runtimes and restarts in rapidly-exploring random trees. In *Twenty-third AAAI conference on artificial intelligence*, pages 127–133, 2008.

[52] Lisa Wu, Andrea Lottarini, Timothy K. Paine, Martha A. Kim, and Kenneth A. Ross. Q100: the architecture and design of a database processing unit. In *Architectural Support for Programming Languages and Operating Systems, ASPLOS '14, Salt Lake City, UT, USA, March 1-5, 2014*, pages 255–268, 2014. doi: 10.1145/2541940.2541961. URL https://doi.org/10.1145/2541940.2541961.

[53] Feng Xiao and Long Wang. Asynchronous consensus in continuous-time multi-agent systems with switching topology and time-varying delays. *IEEE Transactions on Automatic Control*, 53(8):1804–1816, 2008.

[54] Robert Yokota. Don't settle for eventual consistency, Jul 2017. URL https://yokota.blog/2017/02/17/dont-settle-for-eventual-consistency/.