Experimental Studies of Integrated Cognitive Systems

Pat Langley
Computational Learning Laboratory
Center for the Study of Language & Information
Stanford University, Stanford, CA 94305 USA
langley@csli.stanford.edu

Elena Messina
National Institute of Standards & Technology
100 Bureau Drive, Mail Stop 8230
Gaithersburg, MD 20899 USA
elena.messina@nist.gov

Abstract— In this paper, we examine the issues that arise in the experimental study of integrated cognitive systems. We review the reasons why such artifacts are difficult to evaluate, then consider some dependent measures that can be used to characterize their behavior. Next we discuss independent variables that can influence this behavior, in particular features of the domain and characteristics of the system, including its knowledge and experience. We then turn to domains and testbeds that support experiments with such systems, giving examples of some promising candidates. We conclude with a discussion of the scientific goals of experimentation, which involve understanding the mapping from domain and system characteristics onto behavior.

I. Introduction and Motivation

For more than a decade, research in artificial intelligence has relied on experimentation as a key element in evaluation. Machine learning was perhaps the first subdiscipline to adopt systematic experiments (e.g., Kibler & Langley, 1988), but their use has spread throughout the broader community (e.g., Cohen, 1995). Today, experiments are the primary means by which AI researchers evaluate their methods, and the experimental techniques as mature and well understood.

However, the experimental study of integrated cognitive systems is less well established and clearly needs more development. The reasons should be clear from the phrase itself, which reflects the nature of the intelligent artifacts being constructed. First, it is inherently more difficult to evaluate systems than component algorithms, since they are harder to construct and analyze. Second, it is more challenging to run experiments with cognitive systems, since they rely on complex, multi-step reasoning rather than simple classification or reactive control. Finally, evaluating claims about integrated systems is problematic because it involves the examination of interactions among their components. Together, these distinctive factors have slowed the development of an experimental method for such complex entities.

In this paper, we propose an experimental framework that is appropriate for the study of integrated cognitive systems. In the next section, we discuss basic and higher-order dependent measures that can arise in such experiments. After this, we consider three main classes of independent factors that can influence system behavior, then turn to domains and testbeds that would support the experimental evaluation of such systems. In closing, we discuss the broader scientific goals of experimentation, which aim not to show superiority but to identify reasons for observed behaviors.

II. Dependent Measures of System Behavior

As scientists, we are concerned with understanding the behavior of integrated cognitive systems, which in turn means that we require ways to observe and characterize this behavior. In this context, it is important to distinguish between between metrics and dependent measures. These terms are closely related, but the first is typically associated with prescriptive benchmarks that are used to determine one system's superiority other another, whereas the second is generally associated with systematic experiments that aim at scientific understanding. The comments that follow are relevant to both approaches to evaluation, but our focus here is on the latter, which we think is far more appropriate for the current stage of the field. We organize our treatment into three broad categories: basic measures, averaged metrics, and higher-order variables.

A. Basic Measures of System Behavior

The existing literature reports a variety of basic measures that are relevant to integrated cognitive systems. These provide the simplest ways to describe the observed behavior of an intelligent construct. We should clarify that behavior always occurs in the context of some task, whether provided externally or generated by the agent itself, and some situation, whether it involves the agent's physical environment or its mental state. We will refer to this context informally as the problem that the agent is attempting to solve.

Perhaps the most straightforward behavioral measure concerns whether the agent succeeds or fails at handling a given problem. For example, a cognitive system may prove or fail to prove a geometry theorem, it may or may not solve a novel puzzle, it may or may not deliver a package to a specified address, and it may win or lose a given game. This measure offers only one bit of information, but it may still be valuable when combined with other results, as we will see shortly.

However, other problem-related measures provide more detail. One such metric is the efficiency or speed with which the cognitive system handles a given problem.

For instance, one can count the number of states in a problem space considered during a geometry proof, the time it takes a UPS driver to deliver a package, and the number of moves until checkmate in a chess game. Such a dependent variable gives information about the cognitive or physical efficiency with which the agent handles a particular problem.

Of course, some paths to success are more desirable than others, so we may also want to measure the quality of the cognitive system's solution to a problem. For example, a geometry proof may have few or many steps and thus be more or less elegant, a package deliverer may drive safely and politely or dangerously and impolitely on his way to an address, and a chess player may lose only a few unimportant pieces or many important ones in defeating an opponent. Metrics of this sort offer details about the desirability of the cognitive agent's behavior in accomplishing a given task.

B. Combined Measures of Behavior

The field of statistics tells us we should not draw conclusions from individual cases, but rather that we should rely on multiple samples. We can then combine the results from these samples and calculate a more robust dependent variable. Taking the average of sampled measurements is the most common and obvious combination scheme, but calculating cumulative scores is another possibility. The important thing is that, by combining measures for different samples, we can partly cancel out variation due to unknown or unavailable factors, and thus increase the chance of meaningful results.

Naturally, this approach requires some population from which to draw samples, typically different problems from within a single domain, although sampling from across domains is also possible. For instance, we might present the cognitive system with different geometry theorems to prove, ask it to deliver packages to distinct addresses or even in different cities, and confront it with different chess opponents or even chess-like games with alternative rules. The population from which one draws samples determines the generality of one's conclusions about the cognitive system's behavior. We may suspect that the agent can prove theorems not only in geometry but also in algebra, but sampling from the former domain provides no evidence for the latter. An empirical study should state clearly the population being sampled, ideally in formal terms but always in enough detail that others can replicate the sampling process.

We should note that combined measures of behavior offer more than guards against unknown factors and random noise. This approach also lets one convert qualitative measures, such as success or failure on a problem, into quantitative ones, such as the percentage or total number of problems solved. This makes them especially useful for researchers who want to make claims about new functionality, which at first glance appear to involve only qualitative evidence, but which can be handled in quantitative terms with averaged, cumulative, or other combined measurements of system performance.

C. Higher-Order Measures of Behavior

Although combined measures guard against unknown influences and offer quantitative variables, they still present only a small window into often complex behavior. Metrics that average across domains improve the situation, since they provide information about a cognitive system's broader generality, but more sophisticated responses are certainly possible.

For instance, we might plot the dependent measure for a novel system against the same measure for a baseline or control system, with each point summarizing the two systems' behaviors on a distinct problem. We can then use regression to fit a line to the points, which gives both a slope and an intercept as higher-order measures. A positive intercept means the novel system does better than the control even on easy problems, whereas a slope greater than one means it scales to difficulty better than the baseline system.

Another example, which we will discuss more later, involves learning curves, in which one plots a behavioral measure like efficiency or quality against the number of training cases a learning system has encountered. Such curves typically have either an exponential or sigmoid shape, so that linear regression is not appropriate, but we can fit them with other parametric forms. These produce higher-order measures for the system's performance at the outset, its rate of improvement as a function of experience, and its asymptotic performance.

Both of these examples involve some form of variation, though this need not be systematic. In general, whenever one collects simple measures of a cognitive system's behavior under a number of distinct conditions, these can be used to calculate higher-order measures that summarize its behavioral characteristics across the conditions from which the samples were taken.

III. Influences on System Behavior

A scientific experiment should do more than measure a system's behavior under one or more condition. The goal of experimentation is to understand the factors that influence the behavior, which means one should measure the dependent variables in multiple situations that differ along some dimension. Such a factor is often referred to as an independent variable, since one can typically vary it independently of others. As with dependent measures, different independent variables can reveal different facets of the system under study. In this section, we examine three broad classes of controllable factors that are appropriate for the experimental evaluation of integrated cognitive systems.

A. Characteristics of the Task and Domain

One important type of independent variable concerns aspects of the problem domain and the tasks which occur within it. The simplest version of this idea involves collecting multiple samples for an experimental condition, which we have already discussed above. For studies with an intelligent system, this means running the system multiple times on different problems from a domain, and then combining the results in some fashion. For this purpose, one draws sample tasks from some distribution over the problem domain. This may involve specifying a fixed set of problems or tasks, but another strategy involves creating a generator that can produce sample problems. In either case, one should state the relation between these samples and the broader class of problems over which one hopes to generalize.

An important variation on this idea involves running the system on problems from different domains to ensure its generality. If we are interested in this central issue, then it is essential to demonstrate successful behavior not only across different tasks within the same domain, but across a variety of distinct domains. For instance, most AI work on game playing has focused on a single game like chess, which Pell (1996) argues has produced systems that are optimized for that domain but do not demonstrate general intelligence. Instead, he defined an entire class of chess-like games and developed a system that plays reasonably when given information about their board, pieces, and rules.

Such studies ensure generality, but they do not by themselves reveal the reasons for variations in system behavior. For this, we must examine the relation between problem difficulty and response. We can order problems by the results they produce on some behavioral measure like problems solved or efficiency of solutions, but this does not provide much insight. Ideally, one should vary experimentally the problem difficulty and examine its effects on system behavior. This in turn requires an analysis of the domain that suggests what factors influence the difficulty of problems.

Kibler and Langley (1998) provide an early domain analysis for machine learning. They propose a number of factors that affect the difficulty of induction tasks, including the complexity of the target concept, the number of irrelevant features, and the amount of noise in the training data. Their analysis focused on classification, but they mention analogous difficulty factors for other areas, such as the regularity of problem spaces and the structure of target grammars. One factor they overlooked was the rate of environmental change, which can pose a challenge for any learning system.

Studies that vary problem difficulty typically rely on synthetic domains to control this factor, but Langley (1996) warns against their casual use. Synthetic problems give one fine-grained control over domain characteristics, which can let one determine how these factors influence behavior. But one must be careful to ensure that these problems are sufficiently similar to ones which arise in natural domains that they remain relevant. Nor should one utilize synthetic problems except to support the systematic variation of domain features. In general, a well-balanced experimental program includes studies with both synthetic domains, to provide insight, and natural ones, to ensure relevance.[1]

B. Characteristics of the System

If we want to understand why a cognitive system behaves well or poorly, then we must vary characteristics of that system. The simplest version of this idea involves replacing the entire system with another, as typically occurs in competitions. Unfortunately, even when one system behaves uniformly better than another, which seldom happens, such comparisons provide no insight into the reasons for their behavioral differences.

One form of finer-grained study involves varying the parameters associated with the cognitive system and measuring the effect on its behavior. For instance, one might alter the depth to which search occurs in a system that proves geometry theorems, the utility function used to guide a driving system's choices, and the relative values of pieces in a chess player. Such experiments can lead to conclusions about the importance of a parameter to system behavior, which may be unchanged across a wide range of parameter values, change slowly as the parameter varies, or produce sudden shifts at certain threshold values. Parametric studies may also detect interactions among settings that indicate nonlinear effects.

Another experimental approach compares the basic system's behavior with that when one or more of its modules has been removed. For example, one might compare a driving agent with and without a component for planning routes. Similarly, one might examine a geometry theorem prover with and without a module that learns from previous proofs or a chess player that can or cannot analyze its opponent's strategy. Such lesion studies let one draw conclusions about the contribution of the removed components to the system's overall behavior. They can be especially useful in understanding integrated cognitive systems, since they can reveal interactions among modules. For instance, inclusion of planning and learning abilities in a driving system may provide benefits greater than their sum when used alone.

C. Knowledge and Experience of the System

Cognitive systems rely centrally on knowledge about a domain to make inferences and generate candidate solutions to the problems they encounter. Knowledge is just as important a determinant of behavior as the domain and system characteristics. However, the precise impact of

[1] Unfortunately, this mixture is quite rare in the literature, presumably because it requires extra effort from experimenters, but this does not reduce its importance for the study of intelligent systems.

knowledge on a specific intelligent system is an open issue that can be studied experimentally.

The methodology of lesion studies, which we discussed above in the context of system components, can be adapted easily to knowledge. We can run a geometry theorem prover with and without access to lemmas, we can ask a driver to deliver packages with and without a cognitive map of the city, and we can provide or not provide a chess player with a library of opening moves. In some cases, such lesion studies are equivalent to experiments with system modules, since certain components may be included only to utilize a specific type of knowledge. But the modules of many cognitive systems have more general abilities, so that running them with and without access to knowledge can uncover its importance independent of the component processes themselves.

Of course, the knowledge utilized by a cognitive system does not usually come in large packages, but rather in small, modular knowledge elements. As a result, one can also vary systematically the amount of knowledge available to the agent of a given type. For instance, a theorem prover may have access to many or few lemmas, a driver responsible for delivering packages may have a more or less complete cognitive map, and a chess player may know about different numbers of opening moves. Experiments that treat knowledge in this manner produce graphs that plot behavioral measures like efficiency and quality against knowledge. These can also provide higher-order metrics that describe the rate of improvement per knowledge element, as we discussed earlier.

For cognitive systems that learn, we can examine the effects of experience in a similar manner. Here one relates the number of problems solved, the time spent by the agent, or other measures of experience to the standard behavioral variables. For example, one can graph the percentage of geometry theorems proved as a function of the number of previous efforts, the efficiency of package delivery against the number of earlier trips, and the number of chess pieces lost against the number of games played. As mentioned earlier, such learning curves also provide higher-order information about the rate of improvement and asymptotic behavior.

IV. Repositories for Cognitive Systems

As we have noted, experimental studies of intelligent systems require some class of problems on which to measure behavior, but developing such tasks can be time consuming and expensive. The natural response is to develop a common repository of domains and problems for use by the research community. The earliest example was the UCI Machine Learning Repository (Blake & Merz, 1998), launched by David Aha in the late 1980s. This provided a variety of well-documented data sets for the evaluation of supervised learning systems, and within a few years it became so popular that most papers on machine learning utilized it in their experimental studies. Another model came from computational linguistics, where the annual TREC competitions came to drive many research efforts and has been imitated by other fields, such as the AI planning community.

Unfortunately, despite their advantages, repositories and competitions also have negative aspects. Their very ease of use can encourage a community to focus only on the technical issues they represent. For example, the UCI repository encouraged increased learning research on classification domains at the expense of work on problem-solving tasks. Moreover, many learning researchers have adopted a 'bake-off' mentality that is concerned only with improving performance scores over earlier systems, and competitions like TREC have much the same effect. To the extent that the contents of repositories come to be viewed as benchmark problems, they lose their usefulness for genuine scientific studies.

A. Desirable Characteristics of Testbeds

Nevertheless, a common repository is an obvious means to encourage and support research on integrated cognitive systems, so we should consider what characteristics would make it most useful. Like the UCI repository, it must include a variety of distinct domains to ensure the generality of experimental results. Moreover, its contents must be well documented and it must be easy for researchers to use, with a standardized format or interface to simplify interaction with different cognitive systems. These are key characteristics of existing repositories that are well worth replicating in new ones.

However, the repository should support experiments with integrated cognitive systems in ways that previous ones have not. For example, it should not contain data sets like the UCI site or the TREC competitions, or even sets of problems, like the planning competitions. Instead, it should provide the community with environments or testbeds in which researchers can evaluate their creations. Unlike many component AI algorithms, a cognitive system exists over time and requires some environment in which to operate. This environment need not be a physical one, but embodied cognitive systems are perhaps the most interesting variety, so the repository should contain some testbeds that support the study of physical agents.

A testbed provides supporting or enabling infrastructure for work on a given problem domain. Each testbed must include a definition of the tasks or missions that arise in its domain, stated in terms of initial situations and the desired states or objectives. Each domain should support a range of such tasks and, ideally, come with a problem generator that researchers can use to produce novel ones. A testbed provides infrastructure that facilitates experimentation by the community and thus can lead to insights about alternative approaches. Examples of infrastructural support include: external databases, such as geographic information systems, and the means to connecting to

these resources; the controlled capture, replay, halting, and restart of scenarios; and methods for capturing relevant performance measures via application programming interfaces, access to variables and parameters, and external physical instrumentation.

A well-designed testbed for cognitive systems eases their experimental evaluation, which follows naturally from certain desirable attributes of the infrastructure and problem set. To assist researchers in evaluating high-level behavior, it should provide an environment that has little or no dependence on actuation or sensor processing. In addition, the infrastructure and problem domain should offer a rich operating environment, with the ability to model and control various entities. The testbed should let researchers vary, in quantifiable ways, the difficulty or complexity of the environment or mission. Moreover, although the study of integrated systems is crucial, a testbed should also support evaluation of component subsystems, such as reasoning and learning methods, through parametric and lesion studies.

For domains that involve an external setting, one can certainly create a physical testbed to support evaluation, but another option is to develop a realistic simulated environment that can be used by many more research groups at much lower cost. For example, Jacoff, Messina, and Evans (2001) describe a physical testbed for evaluating robot search and rescue, whereas Balakirsky and Messina (2002) report a simulated environment to support research on the same problem. Simulated testbeds have an additional advantage in that they allow easy variation of domain parameters, ranging from details of the environmental layout to noise in the agent's sensors. Moreover, they let one record detailed traces of the intelligent system's physical behavior and its mapping onto cognitive state, which in turn supports detailed analyses and replay starting from any point along the agent's behavioral trajectory.

However, as we noted above, testbeds that rely on synthetic domains also come with the danger of irrelevance. Whenever possible, they should be based closely on a physical testbed and provide simulations of sufficiently high fidelity. Wang (2003) describes one such simulated domain that incorporates models, based on a gaming engine that supports kinematics and dynamics, of the physical NIST arenas for urban search and rescue. To further ensure relevance for intelligent systems that sense their environment, a testbed may provide data sets collected from real sensors in analogous locations (e.g., Shneier, 2003). Such additions can help retain the advantages of physical environments while offering the affordability and ease of simulated ones.

B. Promising Domains and Testbeds

We can clarify the desirable features of testbeds with some examples. We have already mentioned the search and rescue domain, for which NIST has developed both physical and simulated testbeds. The primary task involves searching for survivors in an urban area after an earthquake or similar disaster. This domain requires the combination of sensing, planning, and action in an integrated cognitive system that can recognize humans, find routes through dangerous areas, and execute its plans successfully. The testbeds have been in place for a number of years and have been used effectively in a number of international competitions.

Another candidate domain involves flying a simulated aircraft in a military setting. Keeping an airplane aloft can be a challenging control task, but by itself this does not require much cognitive activity or integration of different capabilities. However, Jones et al. (1999) report a complex environment in which an agent must fly a jet fighter, distinguish friendly from enemy aircraft, respond according to established doctrine, and communicate with other pilots. Their intelligent agent operated within the ModSAF environment, which was populated by other aircraft, some controlled by programs and others by humans. A related set of problems would involve flying an unmanned reconnaissance vehicle over enemy territory to gather information while avoiding dangerous areas.

A third challenging domain involves in-city driving. This raises few problems at the control level, since keeping a car upright, on the road, and within its lane does not require much intelligence. But the presence of buildings, sidewalks, traffic signs and signals, moving and parked vehicles, and pedestrians make for a very rich environment that requires the allocation of perceptual attention and other resources. Moreover, driving can support many distinct high-level tasks, such as delivering packages, tailing another car unobtrusively, and pulling over vehicles for moving violations. These all require the integration of cognitive, perceptual, and motor components in a complex dynamical setting.

There already exist many simulated driving environments, but few have been developed with the intention of evaluating intelligent systems. Moriarty and Langley (1998) report a simulator for highway driving, but this environment had low fidelity and agents had limited options. More recently, Choi et al. (2004) describe an in-city driving environment, which they have used to evaluate a cognitive driving agent, that includes many more objects and a broader range of activities. Balakirsky, Scrapper, and Messina (in press) are developing another infrastructure, Mobility Open Architecture Simulation and Tools, that provides well-defined interfaces to the various driving subsystems and rich visualization at various levels of resolution. Several organizations are using this system to test subsystems for vehicle control, but it remains to be seen whether the environment meets all the requirements for evaluating an integrated cognitive system.

Both driving and flying involve control of an individual agent, but an equally important class of domains involve managing a large set of other agents. Commanding troops in a battlefield scenario is one example that requires capa-

bilities like monitoring, situation assessment, planning and scheduling of activities, and allocation of resources. However, interactive strategy games like Civilization have similar characteristics and complexity, and they are familiar to more people. Aha and Molineaux (2004) are constructing a framework that simplifies the interface to such games, and thus will provide a set of related testbeds for the experimental study of integrated cognitive systems. Michael Genesereth (personal communication, 2004) is developing a different infrastructure to support an annual competition in generalized game playing (http://games.stanford.edu/), with the intent of fostering research efforts on flexible approaches to intelligent behavior.

V. Concluding Remarks

In the preceding pages, we have considered the dependent measures and independent factors that arise in studying integrated cognitive systems, along with characteristics of repositories and testbeds to support such experiments. Before closing, we should situate these comments in the broader context of scientific experimentation. As in other fields, the aim of systematic experiments is not to show that one approach is superior to another but rather to increase our understanding of complex systems. Such understanding may also lead to improved artifacts, but the overriding goal is to produce replicable and interpretable results that add to our scientific knowledge about intelligent behavior.

To this end, researchers should not carry out unmotivated comparisons between different systems or environments. In most cases, one should have a clear question in mind or a specific hypothesis that one wants to test, and the experimental design should reflect this intention. Simple demonstrations of functionality and generality are reasonable when one first develops a cognitive system, but they should quickly give way to scaling studies that reveal its ability to handle complexity and to lesion studies that identify the roles that its components play in determining overall behavior.

Whenever possible, experimental results should be utilized to test such hypotheses. Because most studies involve averaging across samples, one should be careful about drawing conclusions. Statistical tests can be useful for this purpose, but they are overrated, in that one can sometimes obtain 'significant' differences between experimental conditions even when they are not substantial. Nor are statistical tests required when differences are large, although reporting confidence intervals is crucial for conditions with high variance.

Results that agree with an hypothesis lend it evidence, though they do not 'confirm' it; science can never draw final conclusions about any situation. Results that diverge from one's expectations count as evidence against a claim, and thus require additional explanation. Negative results need not imply failure, since they can lead one to alter assumptions about system behavior and suggest new ways to test them. The iterative loop of hypothesize and test is as central the study of intelligent systems as to other experimental disciplines.

Nevertheless, integrated cognitive systems pose special challenges that require creative adaptation of standard experimental methods. We must develop testbeds that exercise the full capabilities of such systems, rather than emphasizing tasks that can be handled by simple classification or reactive control. We must study behavior at the system level, rather than focusing on component algorithms. Finally, we must design experiments that illuminate the manner in which the modules of such systems interact to produce flexible and robust behavior. Taken together, these steps should let us transform the study of integrated cognitive systems into a dynamic and well-balanced experimental science.

Acknowledgements

This research was funded in part by Grant HR0011-04-1-0008 from Rome Laboratories. Discussions with David Aha, Michael Genesereth, and Barney Pell contributed to the ideas presented in this paper.

References

Aha, D. W., & Molineaux, M. (2004). Integrating learning in interactive gaming simulators. Proceedings of the AAAI-2004 Workshop on Challenges of Game AI. San Jose, CA: AAAI Press.

Balakirsky, S., & Messina, E. (2002). A simulation framework for evaluating mobile robots. Proceedings of the Performance Metrics for Intelligent Systems Workshop. Gaithersburg, MD.

Balakirsky, S., Scrapper, C., & Messina, E. (in press). Mobility Open Architecture Simulation and Tools Environment. Proceedings of the International Conference Integration of Knowledge Intensive Multi-Agent Systems. Boston.

Blake, C. L. & Merz, C. J. (1998). UCI repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

Choi, D., Kaufman, M., Langley, P., Nejati, N., & Shapiro, D. (2004). An architecture for persistent reactive behavior. Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (pp. 988–995). New York: ACM Press.

Cohen, P. R. (1995). Empirical methods for artificial intelligence. Cambridge, MA: The MIT Press.

Jacoff, A., Messina, E., & Evans, J. (2001). Experiences in deploying test arenas for autonomous mobile robots. Proceedings of the Performance Metrics for Intelligent Systems Workshop. Mexico City, Mexico.

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P. G., & Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. AI Magazine, 20, 27–41.

Kibler, D., & Langley, P. (1988). Machine learning as an experimental science. Proceedings of the Third European Working Session on Learning (pp. 81–92). Glasgow, Scotland: Pittman.

Langley, P. (October, 1996). Relevance and insight in experimental studies. IEEE Expert, 11–12.

Moriarty, D., & Langley, P. (1998). Learning cooperative lane selection strategies for highways. Proceedings of the Fifteenth National Conference on Artificial Intelligence (pp. 684–691). Madison, WI: AAAI Press.

Pell, B. (1996). A strategic Metagame player for general chess-like games. Computational Intelligence, 12, 177–198.

Shneier, M., Chang, T., Hong, T.H., Cheok, G., Scott, H., Legowik, S., & Lytle, A. (2003). A repository of sensor data for autonomous driving research. Proceedings of the SPIE Aerosense Conference. Orlando, FL.

Wang, J., Lewis, M., & Gennari, J. (2003). A game engine based simulation of the NIST USAR arenas. Proceedings of the 2003 Winter Simulation Conference (pp. 1039–1045). New Orleans, LA.