2005 American Control Conference
June 8-10, 2005. Portland, OR, USA

Language-measure-based Supervisory Control of a Mobile Robot

Xi Wang xxw117@psu.edu
Goutham Mallapragada goutham@psu.edu
Asok Ray axr2@psu.edu
Department of Mechanical Engineering
The Pennsylvania State University
University Park, PA 16802

Keywords: *Behavioral Robotics; Supervisory Control; Quantitative Design; Discrete-event System; Learning*

Abstract— **This paper presents the design, modelling, and supervisory control of a mobile robot based on a signed real measure of its automaton (i.e., discrete-event behavior) language. While the robot's dynamic behavior is manipulated in the continuous-time domain via motion control and visual servoing, the mission planning is performed in the discrete-event supervisory control setting. However, unlike the conventional qualitative framework of supervisory control following the Ramadge-Wonham approach that is based on a set of specified constraints, a quantitative approach has been adopted for synthesis of optimal supervisory controllers in robotic scenarios with a language measure being the performance index. The parameters of the language measure are identified via both experimental observations and simulation runs; the results are consistent with each other as well as with other measures. This approach complements the Q-learning method that has been widely used in robotics research to learn primitive behaviors.**

I. Introduction

The control of a mobile robot involves continuous time domain motion control as well as higher level mission planning. Two typical approaches towards mission planning are top-down and bottom-up approaches. A world model is used by the planner to decide the most appropriate sequence of actions for the agent in the top-down *planner-based* or *deliberative* approach. This approach may suffer from poor scalability to real world applications and infeasibility of making real-time responses to large abrupt changes in the environment [7] [2] because frequent replanning is required due to uncertainties in sensing and action as well as in the environment. On the other hand, the bottom-up *reactive-behavior-based* approach transforms the agent's control strategy into a collection of preprogrammed parallel condition-action pairs with a minimal set of internal states and no search requirements; hence, it is suitable for real-time applications [7].

The modularly designed component behaviors are activated in parallel, producing various commands to corresponding actuators. While these behaviors are suitable for representing low-level continuous-control functions and are usually successful in making local incremental decisions, higher level tasks have been traditionally handled by artificial intelligence (AI) symbolic planning because global states of the world may not be available. This issue has been addressed within the subsumption architecture [1], where behaviors are represented as finite state machines augmented with a set of input and output channels, providing a communication link between the higher and lower levels. For example, the lower level behaviors (e.g., *walking, attraction to a goal, repulsion from an obstacle*)that are mostly active trigger or deactivate the high level behaviors (e.g., *wandering around, avoidance)* and the higher level behaviors are allowed to "subsume" or override the output of the lower level. However, once the layered network architecture is completed, it remains fixed since the arbitration scheme is hardwired.

The discrete event approach was proposed to robotics research community by Košecká [5] for navigation of mobile robots with various tasks defined as discrete states (e.g., *moving, steer_away, path_following*). Similarly, Feddema et al [3] use the discrete states (e.g., *Search, Rotate, Backup*, and *Track*) for the line following problem. Discrete event systems are appropriate for modelling complex systems that usually consist of many interacting components operating in a dynamically changing environment driven by abrupt changes (also known as discrete events). In this sense, the discrete-event approach is similar to AI symbolic planning in which the robotic system is hierarchically structured into three levels: *symbolic planning, reactive behaviors, and low level servoing*. On the other hand, the discrete event approach provides a sound theoretical background and a formal treatment in the design of autonomous mobile robots. Considering the inherently discrete nature of the robot's behavior, various types of interruptions from the environment, and the discrete features extracted from sensors, it is more reasonable to model a mobile robot in the discrete-event framework. Supervisory control of discrete-event systems [8] has been adopted in this paper for the high-level mission planning.

Maximal permissiveness [14] is a qualitative method for synthesizing supervisory control policies. However, there are several issues that cannot be resolved by the qualitative design approach: 1) It may not be possible to comparatively evaluate the performance of individual supervisors, whose generating languages are not totally ordered; 2) The maximally permissive supervisor does not necessarily imply the best performance from the operation and mission fulfilling perspectives of the physical plant. In general, there is no direct relationship between permissiveness of the generating language and the plant performance goals; 3) The importance of either reaching certain states (*good*

0-7803-9098-9/05/$25.00 ©2005 AACC

marked states) or avoiding some other states (*bad marked states*), as much as possible is not addressed; 4) In many engineering application domains, the purpose of supervisory control is to make decisions where the supervisory decision is to keep exactly one controllable event enabled at a given instant. The conventional supervisory control [8] does not address which controllable event will give better performance in the synthesis procedure.

To address the above issues, a quantitative performance measure, called *language measure*, has been proposed by Wang and Ray [11], Ray and Phoha [9]. Optimal and robust control is proposed by Fu et al. [4] based on the language measure.

The paper is organized in six sections. Section I gave the motivation for using Discrete Event setting for robot control. Section II presents the design of a mobile robotic system. Section III describes the interface design between continuous-time domain and discrete-event domain. Section IV briefly reviews optimal controller synthesis using the quantitative language measure [11]. Section V presents optimal controller synthesis using language measure in both experiments and simulation, and the results are compared with other standard measures. The paper is summarized and concluded in Section VI along with recommendation for future research.

II. Supervisory Control Architecture

Figure 1 shows an architecture of the proposed discrete-event supervisory (DES) control for a multi-layer robotic system that integrates the event-driven dynamics modelled by a finite-state automaton (FSA) and the time-driven continuous dynamics modeled by a set of ordinary differential equations interconnected by appropriately defined continuous/discrete (C/D) and discrete/continuous (D/C) interfaces. The DES control module (DCSM) communicates with the

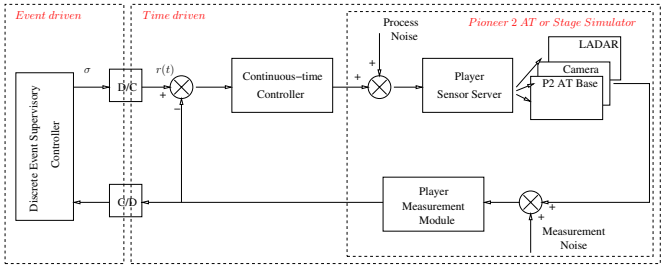

Fig. 1. DES behavior based robotic system architecture

robot's continuous time-varying control module (CTCM) by sending and receiving a set of discrete events (see Table I). DCSM is designed to interact with CTCM, independently of the underlying physical process. The DCSM is allowed to plug and play whenever a new DES controller is loaded or the format is changed. CTCM connects to the Player via a standard TCP socket for sending and receiving formatted messages that encode up-to-date sensor readings and continuously varying commands of reference signals at 10 Hz. The control strategy is event-driven, in which CTCM (in particular, the C/D block) generates discrete events based on the continuous sensor data received from the Player. The discrete events are transmitted to DCSM in a symbolic form. DCSM reacts immediately by sending out a discrete-event command back to CTCM generated by the supervisor. After receiving a particular command event from the DES controller, CTCM sends out a set of continuous reference signals to the Player to maneuver the robot accordingly. The robot continues executing this behavior until the sensor readings trigger the occurrence of a new event. In this architecture, DES control strategies for both the real robot and a simulator were designed and exercised.

III. Design of Robot Behavior Interface

A behavior is a set of processes involving sensing and action against the environment. Behaviors can be designed at a variety of levels of abstraction. In general, they are made to be higher than the robot's atomic actions (e.g., *turn left by 30 degree*), and they extend in time and space. Some commonly implemented behaviors include: *go home, search object, avoid obstacle, pick up object*, etc. In the discrete event setting, a behavior is a controllable event. The union of these behaviors is the controllable event set Σ_c. The set of uncontrollable events Σ_u is the set of all possible response of the robot during the interaction with its environment. It consists of the results of the robot's behaviors, including a successful or unsuccessful completion of the controllable events (e.g., *approach a target, grab an object*) and an interruption of the robot's current behaviors (e.g., *find an object, detect obstacle*).

The C/D and D/C blocks in Figure 1 are the interfaces between the discrete event dynamics and continuous time dynamics of the robot system. The D/C block is a map $\phi : \Sigma \to \mathbb{R}^m$ that converts each controllable event into the continuous reference signal as follows.

$$r(t) = \phi(\sigma_k) \qquad t_k \le t < t_{k+1} \tag{1}$$

where $\sigma_k \in \Sigma$ is the most recent controllable event and t_k is the time of the k-th discrete event occurrence. The C/D block is a map ψ that converts the state space $\mathbf{x}$ of the continuous time system into the set of discrete events Σ. An event σ_τ is generated at time $t = \tau$ if there exist $\epsilon, \delta > 0$, such that $\forall 0 < \epsilon < \delta$

$$h(\mathbf{x}, \mathbf{u}; \tau) = 0, \text{ and } h(\mathbf{x}, \mathbf{u}; \tau - \epsilon) \neq 0 \tag{2}$$

$$\sigma_\tau = \psi(\mathbf{x}) \tag{3}$$

Let Eq. (2) define a closed set Ω. An event is generated as the state trajectory enters Ω from outside for the first time. If there is a segment of trajectory $\mathbf{x}$ during $[t_1, t_2]$ that satisfies Eq. (2), then the event is defined to occur at t_1. The C/D block, also known as the event generator, is fully specified by the pair (h, ψ).

Based on all possible behaviors Σ_c and all possible responses Σ_u of the robot during the interaction with its environment, the alphabet Σ of discrete events is obtained

TABLE I

THE EVENT SET Σ FOR P2AT ROBOT

Σ	Description	$\in \Sigma_c$?
a	approach the object	$\surd$
A	avoid obstacle successfully	
c	reach goal with an object	
C	find an object but gripper full	
d	drop an object	$\surd$
g	grab an object	$\surd$
h	return to home	$\surd$
i	ignore the current observed target	$\surd$
l	lost the target	
o	obstacle ahead	
p	drop an object successfully	
P	fail to drop an object	
q	grab an object successfully	
Q	fail to grab an object	
T	find goal with an target	
s	search recognizable target	$\surd$
v	avoid obstacle	$\surd$
w	reach target without an object	
W	find object 1	
x	lost the goal	
X	find target 2	
y	lost an object	

as the union of the set Σ_c and the set Σ_u and is listed in Table I, where Σ_c and Σ_u are given by:

$$\begin{aligned}\Sigma_c &= \{a, d, g, h, i, s, v\}\\ \Sigma_u &= \{A, c, C, l, o, p, P, q, Q, T, w, W, x, X, y\}\end{aligned}$$

It should be noted that no two events are triggered at the same time instant. As an example, Figure 2 shows the performance of visual servoing, readers are referred to [6] for details, for approaching a detected object. It is possible that the approaching behavior is interrupted by "found obstacle" event generated by another sensor module such as the laser range finder that measures the distance.

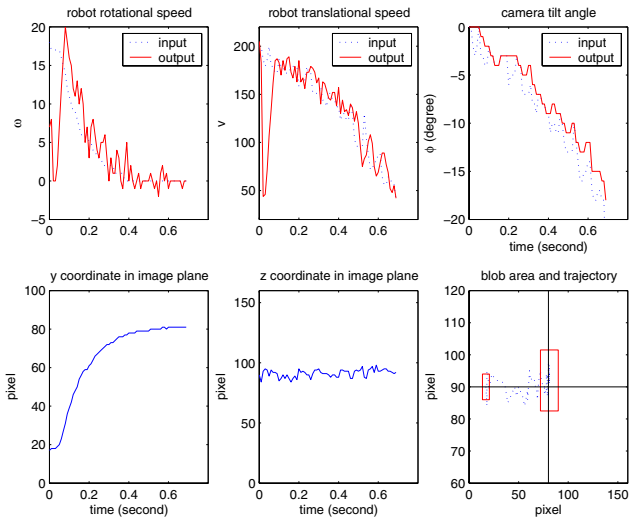

Fig. 2. Performance of robot visual servoing

IV. OPTIMAL SUPERVISOR SYNTHESIS USING LANGUAGE MEASURE

For a formal treatment of the language measure, the readers are refered to [12] and [9]. The pertinent assumptions in the language-measure-based optimal control synthesis are delineated below.

A1 (Cost redistribution) The probabilities of occurrence of *controllable* events in a controlled sublanguage $L(S/G) \subseteq L(G)$ are proportional to those in $L(G)$. For all $q \in Q_S$, where Q_S is the state space of the supervisor automaton S, and $\sigma \in \Sigma_S(q)$, where $\Sigma_S(q)$ is the set of events defined at $q \in Q_S$.

$$\tilde{\pi}_S[q,\sigma] = \frac{\tilde{\pi}_G[q,\sigma]}{\sum_{\sigma\in\Sigma_S(q)} \tilde{\pi}_G[q,\sigma]} \tag{4}$$

A2 (Event controllability) Any transition $\delta(q,\sigma)$, defined in the plant automaton G such that $\sigma \in \Sigma_{uc}(G)$ and $q \in Q$, is kept enabled in a supervisor S.

Under assumption A1, the sum of event costs defined at the state q of a supervisor S is equal to that of state q of the plant G, i.e.,

$$\sum_{\sigma\in\Sigma_S(q)} \tilde{\pi}_S[q,\sigma] = \sum_{\sigma\in\Sigma_G(q)} \tilde{\pi}_G[q,\sigma] \tag{5}$$

Below is a procedure for optimal supervisor synthesis using the language measure. The proof of this algorithm can be found in [10].

1) Initialization. $\mathbf{\Pi}^0 = \mathbf{\Pi}_p$, then $\boldsymbol{\mu}^{\mathbf{0}}$=$(\mathbf{I} - \mathbf{\Pi}^0)^{-1}\mathbf{X}$.
2) Recursion. k-th iteration, where $k \geq 1$.
   1) For every $q \in Q_c(G)$, find out the event $\sigma^* \equiv \sigma^*(q) \in \Sigma_c(G,q)$ such that $q^* = \delta(q,\sigma^*)$ and

$$\mu(L(S^k(\tilde{\mathbf{\Pi}}^k), q^*)) = \max_{\substack{\sigma\in\Sigma_c(G,q)\\ q'=\delta(q,\sigma)}} \mu(L(S^k(\tilde{\mathbf{\Pi}}^k), q')) \tag{6}$$

   where $S^k(\tilde{\mathbf{\Pi}}^k)$ is the intermediate supervisor at k-th iteration whose transition is determined by $\tilde{\mathbf{\Pi}}^k$. Let $\boldsymbol{\sigma}^* = [\sigma_1^* \ \sigma_2^* \ \cdots \ \sigma_n^*]^T$, where the i-th element in $\boldsymbol{\sigma}^*$ is the controllable event left-enabled at state q_i of the plant G according to Eq. (6).
   2) Disable the event set $\Sigma_c(G,q) - \{\boldsymbol{\sigma}^*(q)\}$ for every $q \in Q_c(G)$ and redistribute event cost according to Eq. (4). This results in a new $\tilde{\mathbf{\Pi}}^{k+1}$-matrix which consequently produces a new $\mathbf{\Pi}^{k+1}$-matrix.

$$\mathbf{\Pi}^{k+1} = \mathbf{\Pi}^k + \Delta^k \tag{7}$$

   where Δ^k records the difference between $\mathbf{\Pi}^{k+1}$ and $\mathbf{\Pi}^{k+1}$, consisting positive and negative event costs corresponding to those kept and disabled controllable events, respectively. The resulting intermediate supervisor is $S^{k+1}(\tilde{\mathbf{\Pi}}^{k+1})$.
   3) Compute $\boldsymbol{\mu}^{k+1}$ =$(\mathbf{I} - \mathbf{\Pi}^{k+1})^{-1}\mathbf{X}$

3. Termination. If $\tilde{\mathbf{\Pi}}^{k+1} = \tilde{\mathbf{\Pi}}^{k}$, then stop.

Identification of the parameters in above algorithm can be found in [13].

V. Object Collection Scenario

The robot being used in the experiments is a Pioneer 2 AT robot equipped with a SICK LMS200 laser range finder, a SONY EVI-D30 camera with a Sensoray 311 PC104 frame grabber, and a Lucent WaveLan 11 Mbps wireless card. The on-board computing features a Transmeta Crusoe Processor TM5400 500MHz CPU, 310MB memory, and a 40GB hard disk. As presented in Section III, the robot can perform a set of behaviors, including *search, home, avoid, ignore, approach, grab*, and *drop*. The experimental scenario consists of a robot collecting colored objects to a pre-specified destination. A new colored object (*green* or *pink*) appears randomly in the field every time the robot collects and drops one of the objects back to the destination. Without any DES control, the robot starts with random *search*. Upon detecting an object, it has two options: *approach* or *ignore* the object. If the robot ignores the object, the random search is continued; otherwise, the robot approaches the object to grab. Upon reaching the object, the robot grabs the object and begins the random search to find the destination provided that the object grabbing is successful. If the destination is found, the robot can either *approach* or *ignore*. Once the robot decides to approach the destination and succeeds to reach the destination, it drops the object. The robot repeats the same procedure until stopped by the human operator.

The robot plant model G is built upon the experimental results derived from the scenario in Figure 3. A non-blocking supervisor S is then synthesized according to the following specification language K.

1) Whenever an object is found, the robot approaches it;
2) The robot approaches the destination if it has grabbed an object and found the destination.

Figure 4 shows the state diagram of the supervisor S that is controllable with respect to the plant G and uncontrollable event set Σ_u.

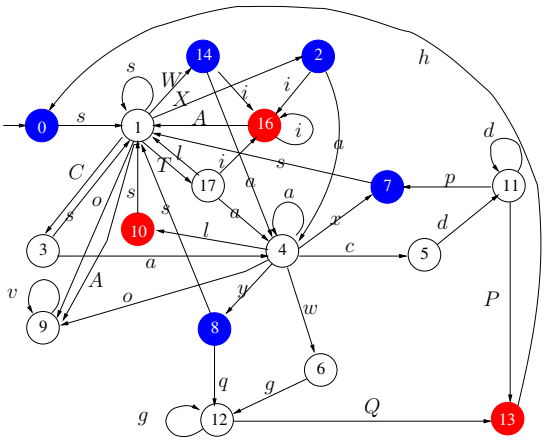

Fig. 3. DES plant model G for the experiment scenario

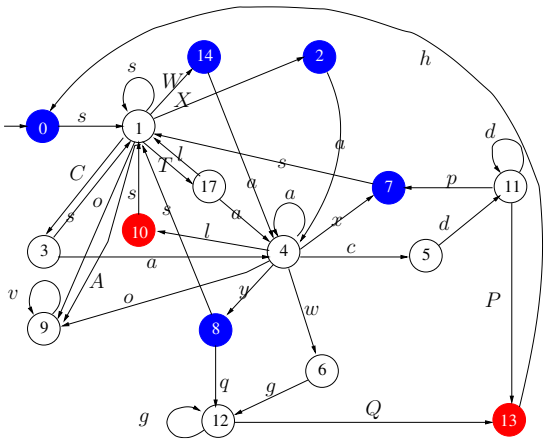

Fig. 4. DES supervisor S for the experiment scenario

TABLE II

The state set Q of the plant model G and its $\mathbf{X}$-vector

State	Description	χ value
0	robot ready for mission	0.0
1	searching for target	0.0
2	found object 2	0.1
3	fount an object but gripper is full	0.0
4	approaching target	0.0
5	ready to drop object at destination	0.0
6	ready to grab object	0.0
7	drop an object successfully	0.8
8	grab an object successfully	0.4
9	avoiding obstacle	0.0
10	lost target	-0.2
11	dropping an object at destination	0.0
12	grabbing an object	0.0
13	failed grabbing or dropping	-1
14	found object 1	0.3
17	found the destination	0.0
16	ignoring an object	-0.1

Next an optimal supervisor of the plant is synthesized using the language measure μ as the performance index and its performance is compared with that of a supervisor that is designed using the conventional procedure [8]. The first task is identification of the $\tilde{\mathbf{\Pi}}$-matrix of the language measure either experimentally or through simulation [13]. Figures 5 and 6 show the convergence of selected non-zero elements in $\tilde{\mathbf{\Pi}}$-matrix obtained via both robot experimentation and simulation. It is seen that the probability of finding object 2 at the supervisor state 1 in the robot experimentation is higher than that in the simulation. The rationale for this discrepancy is that realization of random positioning of the objects in simulation is much easier than in the experiment. In fact, due to the varying illumination conditions, the robot has a higher probability of losing an object. Therefore, experiments need to be designed by placing the objects randomly in a much smaller area of the available test field in order for the robot to find them.

The other parameter of the language measure μ is the characteristic vector $\mathbf{X}$ that is selected based on the designer's perception of the relative importance of the marked

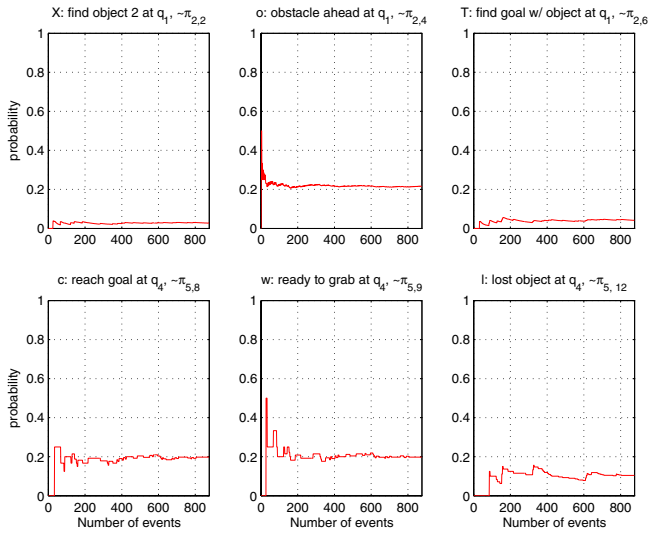

Fig. 5. Convergence of some non-zero $\tilde{\mathbf{\Pi}}$ elements in experiment

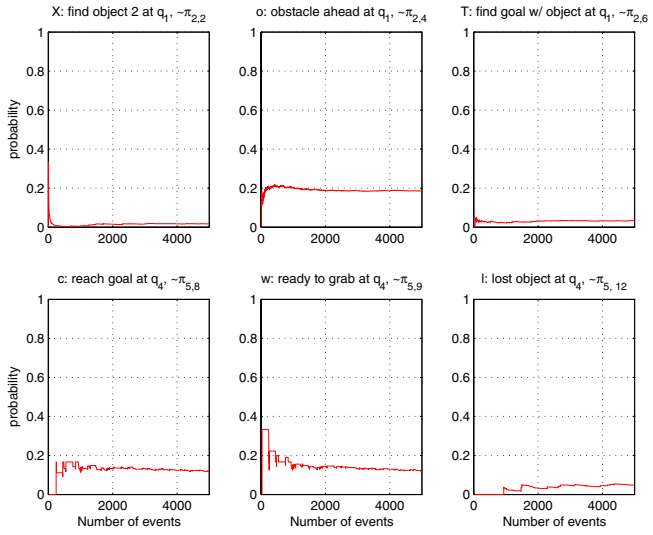

Fig. 6. Convergence of some non-zero $\tilde{\mathbf{\Pi}}$ elements in simulation

states [12] [9]. The states of the plant model G and the respective χ values are listed in Table II. For example, a value of 0.4 is assigned to the state 8 where the robot grabs an object successfully. The state 7, where the robot successfully drops an object at the destination representing the end of the current mission, is assigned a value of 0.8. Finding colored objects are given different level of importance (i.e., χ values) depending on the color as seen in Table II. Lost and ignored an object are given a penalty of -0.2 and -0.1, respectively.

An optimal supervisor is synthesized using the procedure described in Section 4. The algorithm converges after 2 iterations, as listed in Table IV. It is noted that $\boldsymbol{\mu}$ increases elementwise at each iteration [10]. In addition, both μ_3 and μ_{13} do not change at each iteration. By observing the corresponding elements of $\tilde{\mathbf{\Pi}}$-matrix, it is found that the

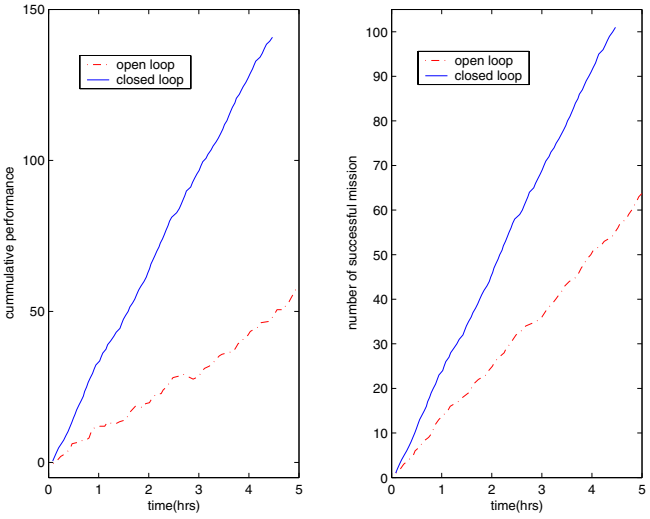

Fig. 7. Cumulative performance comparison in simulation

probabilities of visiting these two states are zero, i.e., they were never visited. Consequently, the 3^{rd} and 13^{th} rows of $(\mathbf{I}-\mathbf{\Pi}^k)^{-1}$ contain one and only one none zero element at the 3^{rd} and 13^{th} element, respectively. In other words, the discrete-event plant model could have been further refined by eliminating state 3 and state 13 from the plant model G. In this case, the optimal supervisor happens to be the same as the conventionally designed supervisor.

The next performance comparison between the open loop and closed loop system behavior under both robot experiments and simulation is given by

$$\begin{aligned} \mu_{\text{exp}}(L(G)) &= 0.1640, \quad \mu_{\text{exp}}(L(S/G)) = 0.1987 \\ \mu_{\text{sim}}(L(G)) &= 0.0799, \quad \mu_{\text{sim}}(L(S/G)) = 0.1349 \end{aligned}$$

It is seen that the results based on real experiments and simulation are consistent. That is, the performance of the supervised system S/G is superio to that of the unsupervised plant G. The cumulative performance comparison in simulation is shown in Figure 7. In about 5 hours mission of collecting objects, the robot under DES control performs much better in both language measure and number of successful missions. As expected, whenever the robot finds an object, the DES controller always lets the robot go and grab it, while without DES control, the robot may ignore it and keep searching.

VI. Summary

This paper presents the synthesis of optimal discrete-event supervisory control using a language-measure-based quantitative approach. The concept is implemented on a mobile robot for controlling its behavior and the results are validated by both experimentation and simulation. Based on an experiment scenario, the discrete event plant model G and a DES controller S are designed. The $\tilde{\mathbf{\Pi}}$-matrix is identified according to the procedure in [13] in both real experiment and simulation. While the cost matrix parameters of the language measure are identified via experimentation

TABLE III

$\mathbf{\Pi}$-MATRIX (17×17) FOR THE DISCRETE-EVENT MODEL G

π_{ij}	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}	q_{11}	q_{12}	q_{13}	q_{14}	q_{17}	q_{16}
q_0	0	.9500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q_1	0	.4087	.0254	0	0	0	0	0	0	.4087	0	0	0	0	.0288	.0785	0
q_2	0	0	0	0	.4396	0	0	0	0	0	0	0	0	0	0	0	.5104
q_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q_4	0	0	0	0	0	.3760	0.3760	0	0	0	.1979	0	0	0	0	0	0
q_5	0	0	0	0	0	0	0	0	0	0	0	.95	0	0	0	0	0
q_6	0	0	0	0	0	0	0	0	0	0	0	0	.95	0	0	0	0
q_7	0	.9500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q_8	0	.9500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q_9	0	.4750	0	0	0	0	0	0	0	.4750	0	0	0	0	0	0	0
q_{10}	0	.9500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q_{11}	0	0	0	0	0	0	0	.95	0	0	0	0	0	0	0	0	0
q_{12}	0	0	0	0	0	0	0	0	.95	0	0	0	0	0	0	0	0
q_{13}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
q_{14}	0	0	0	0	.4495	0	0	0	0	0	0	0	0	0	0	0	.5005
q_{17}	0	0	0	0	.5200	0	0	0	0	0	0	0	0	0	0	0	.4300
q_{16}	0	.3484	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.6016

TABLE IV

ITERATION OF μ SYNTHESIS WITH $(\tilde{\mathbf{\Pi}}, \mathbf{X})$ OBTAINED IN SIMULATION

$\boldsymbol{\mu}$ Elements	$\boldsymbol{\mu}^0$	$\boldsymbol{\mu}^1$	$\boldsymbol{\mu}^2$
μ_0	**0.0799**	**0.1349**	**0.1349**
μ_1	0.0840	0.1420	0.1420
μ_2	0.1578	0.2781	0.2781
μ_3	0	0	0
μ_4	0.3667	0.4071	0.4071
μ_5	0.7218	0.7670	0.7670
μ_6	0.3937	0.4389	0.4389
μ_7	0.8798	0.9349	0.9349
μ_8	0.4798	0.5349	0.5349
μ_9	0.0694	0.1173	0.1173
μ_{10}	-0.1202	-0.0651	-0.0651
μ_{11}	0.7598	0.8074	0.8074
μ_{12}	0.4144	0.4620	0.4620
μ_{13}	-1.0000	-1.0000	-1.0000
μ_{14}	0.3514	0.4735	0.4735
μ_{17}	0.0664	0.1843	0.1843
μ_{16}	-0.2002	-0.1501	-0.1501

TABLE V

ITERATION OF μ SYNTHESIS WITH $(\tilde{\mathbf{\Pi}}, \mathbf{X})$ OBTAINED IN EXPERIMENT

$\boldsymbol{\mu}$ Elements	$\boldsymbol{\mu}^0$	$\boldsymbol{\mu}^1$	$\boldsymbol{\mu}^2$
μ_0	0.1640	0.1987	0.1987
μ_1	0.1726	0.2091	0.2091
μ_2	0.2737	0.3382	0.3382
μ_3	0	0	0
μ_4	0.5114	0.5418	0.5418
μ_5	0.8700	0.9013	0.9013
μ_6	0.5090	0.5403	0.5403
μ_7	0.9640	0.9987	0.9987
μ_8	0.5640	0.5987	0.5987
μ_9	0.1562	0.1892	0.1892
μ_{10}	-0.0360	-0.0013	-0.0013
μ_{11}	0.9158	0.9487	0.9487
μ_{12}	0.5358	0.5687	0.5687
μ_{13}	-1.0000	-1.0000	-1.0000
μ_{14}	0.4798	0.5436	0.5436
μ_{15}	0.2229	0.2818	0.2818
μ_{16}	-0.1000	-0.0681	-0.0681

and simulation, the characteristic (χ) values are assigned according to the mission objective of the robot operation scenario. Through both theoretical language measure computation and empirical cumulative performance, it is shown that the synthesized supervisor has better performance than the open loop system. While the language measure μ is proven to be a useful quantitative measure for the performance evaluation of DES controller, the robot simulator demonstrates its high fidelity with respect to the real system, and hence can be used for further study of more complex scenarios.

REFERENCES

[1] R. A. Brooks, *A robust layered control system for a mobile robot*, IEEE Journal of Robotics and Automation **2** (1986), no. 1, 14–23.

[2] ———, *Intelligence without representation*, Artifical Intelligence **47** (1991), 139–160.

[3] J. T. Feddema, R. D. Robinett, and B. J. Driesen, *Designing stable finite state machine behaviours using phase plane analysis and variable structure control*, May 1998, pp. 1134–1141.

[4] J. Fu, A. Ray, and C.M. Lagoa, *Unconstrained optimal control of regular languages*, Automatica **40** (2004), no. 4, 639–646.

[5] Jana Kosecka, *A framework for modeling and verifying visually giuded agents: Design, analysis and experiments*, Ph.D. thesis, University of Pennsylvania, March 1996.

[6] Y. Ma, *A differential geometric approach to computer vision and its applications in control*, Ph.D. thesis, UC Berkeley, August 2000.

[7] P. Maes and R. A. Brook, *Learning to coordinate behaviors*, AAAI (1990), 796–802.

[8] P. J. Ramadge and W. M. Wonham, *Supervisory control of a class of discrete event processes*, SIAM J. Control and Optimization **25** (1987), no. 1, 206–230.

[9] A. Ray and S. Phoha, *Signed real measure of regular languages for discrete-event automata*, Int. J. Control **76** (2003), no. 18, 1800–1808.

[10] X. Wang, J. Fu, P. Lee, and A. Ray, *Robot behavioral selection using discrete event language measure*, Preprints 2004 American Control Conference, Boston, MA (2004).

[11] X. Wang and A. Ray, *Signed real measure of regular languages*, Proceedings of the 2002 American Control Conference **5** (2002), 3937–3942.

[12] ———, *A language measure for performance evaluation of discrete-event supervisory control systems*, Applied Mathematical Modeling (2004, in press).

[13] X. Wang, A. Ray, and A. Khatkhate, *Online identification of language measure parameters for discrete event supervisory control*, Proc. of 42th IEEE Conf. on Decision and Control (2003).

[14] W. M. Wonham, *Discrete event system control theory and application*, University of Toronto, 2001.