A Model for Controlled Planning of Object Manipulation Using Multiple Agents: An Overview

Shahram Payandeh
Experimental Robotics Laboratory (ERL)
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
8888 University Drive
Canada, V5A 1S6

Abstract

In implementation of intelligent mechanisms such as dexterous mechanical hands, reconfigurable fixtures or part feeders, it is always required to plan to manipulate object in order to gain its proper configuration within the work-space of the manipulating agents. Most of the proposed planners for coordinated control have been based on ad-hoc approaches using search strategies embedded within various search trees. What has been lacking in these algorithms is a formal framework for analysis, synthesis and evaluation of their performances. This paper presents an overview of application of such possible controlled planner architecture *for manipulating objects using multiple agents. The controller is based on discrete-event control theory. The paper extends the application of such theory to the case of multiple agents manipulating a common object. Here for example, a set of events for lifting and rotating an object is described as labeled alphabets. Then it is shown how the manipulation of object can be described as finite-state automaton. The set of uncontrollable events is constructed. The supervisor (planner) can then be constructed by first defining the details of its legal behaviors.*

1 Introduction

Object manipulation is one of the key requirements for example in autonomous assembly environment. In general the work-environment of multiple agents consists of a number of low degrees of freedom mechanisms which can be employed and coordinated to accomplish a given object manipulation sub-task. Examples of such multi agents system are active part-feeders, reconfigurable fixtures, dexterous end-effectors and corporative mobile platforms handling a common object.

In the design of active(intelligent) part feeders[1], a series of active fences (e.g. having one degree of freedom) are located along a conveyor belt where objects are begin fed to the assembly area. The objective is to orient the objects as they arrive at the work-area. Here each fence is equipped with a contact sensor and a single active degree of freedom. It is shown that through interpretation of the contact forces, sequences of discontinuous motions (contact and non-contact) of the fences can be planned and coordinated such that eventually the object can occupy a desired configuration.

In the application of reconfigurable fixtures[2], a number of low degree of freedom moving constraints can be employed where through the coordinated movements of such constraining agents, the object can be held in a force-closure configuration. Here also a basic sensing modality can be integrated in the design of each agent which can be used to determine the contact status between the object and the agents.

In the application dexterous end-effector[3],[4] it is required to coordinate the motions of multiple agents (in this case the contacting fingers) to grasp an object and then manipulate the object between the fingers in order to gain proper configuration of the object within the end-effector.

The task of object manipulation in general is accomplished by first determining its initial and desired configurations. Then by incorporating the nature of constraints between the agents and the object, sequences of pushing forces and motions of constraining mechanisms can be determined. These basic motion and force primitives are coordinated such that in accordance with sequences of loosing and gaining of contact between the agents and object, the desired configuration of object can be obtained.

There are various issues related to grasping and manipulation of object using multiple agents such as contact force interpretation and planning, robust position controllers for each agents, robust contact transition controller (discontinuous controller) and force controller. These issues are being addressed in one way or another in the literature. However, one key element which needs attention is the development of a formal framework where one can study the controlled performance of the planner and the coordinator of multiple agents such as the ones described above. This work similar to [5] explores and present theoretical framework for the coordination of *controlled discontinuities.* These discontinuities are natural part of the object manipulation using multiple agents[7]. As described, the configuration of the object is altered through series of non-contact and contact motions of the agents. Describing a task as a series of distinct segments is not novel to dexterous manipulation; however, there is a need for a sound theoretical approach to provide not only a high-level coordinator, but also to provide some insight into the task organization and help to determine when, where and if sensors should be included with signal interpretation as a part of manipulation operation.

In this paper we apply the discrete-event control theory[8] to high-level description and coordination of dexterous manipulation. We give some insight on how this theory can be extended to such tasks and the challenges and promises of its implementation. Discrete-event control theory has been used to control a variety of robotic applications including manufacturing and assembly tasks[9],[10], [6] and a grasping task[11],[12]. We present here a control-theoretic approach based on [8] which can then be used to explicitly synthesize a supervisor.

The theory is based on expressing the plant (the agents and the object) as a finite-state machine. In addition the planner (the controller) is also expressed as a finite-state machine where both the plant and the controller have a set of marked (desired) states. The plant can be partitioned into a set of controllable and uncontrollable events. The controller (planner) is also equipped with a feedback map which can be enabling or disabling the controllable events. The feedback map can be interpreted as an intelligent ruled-based subsystem for disablement of the events and ensuring that uncontrollable events are never enabled. The behaviors of the plant when it is constraint by the planner is called the supervised discrete-event system.

In the following section we present and overview of such control formalism and then highlight how it can be applied to the case of object manipulation followed by some discussions regarding its implementation.

2 Discrete-Event System

Let the plant be modeled by an automaton, called *plant* in the following form:[8]

$$G = (Q, \Sigma, \delta, q_0, Q_m)$$

where Q is a set of *states*; Σ is a non-empty set of event labels called *alphabet*; δ is the *transition function*, a partial function $\delta : \Sigma \times Q \to Q$; $q_0 \in Q$ is the initial state; and $Q_m \subseteq Q$ is the set of *marker* (terminal) *states.* When Q is finite, G can be described as a finite-state automaton and can be represented as a directed graph where the nodes of the graph are the states in Q, the arcs of the graph are the transitions defined by the function δ, and the set of labels for the arcs are the events in Σ. Thus for any event $\sigma \in \Sigma$ and an initial state $q_0 \in Q$, $\delta(\sigma, q_0)$ is defined (written $\delta(\sigma, q)!$) if there is an arc from q_0 to some other state labeled by σ.

The set Σ^* contains all possible finite sequences, or *strings*, over Σ plus the null string ϵ. The definition for δ can be extended to Σ^*

$$\delta(\epsilon, q) := q,$$

$$(\forall \sigma \in \Sigma)(\forall s \in \Sigma^*)\delta(s\sigma, q) := \delta(\sigma, \delta(s, q)).$$

The language generated by G, also called the *closed behavior* of G, described all possible event sequences that the discrete-event system can undergo

$$L(G) := \{s : s \in \Sigma^* \text{ and } \delta(s, q_0)!\}.$$

The language $L_m(G)$, or the *marked behavior* of G, describes all possible event that represent completed tasks

$$L_m(G) := \{s : s \in \Sigma^* \text{ and } \delta(s, q_0) \in Q_m\}$$

By definition, $L_m(G) \subseteq L(G)$.

For any string $s \in \Sigma^*$, we say that $t \in \Sigma^*$ is a *prefix* of s if $s = tu$ for some $u \in \Sigma^*$. Thus every string $s \in \Sigma^*$ has at least two prefixes, ϵ and s.

If $L \subseteq \Sigma^*$, the *prefix closure* of L is a language denoted by $\bar{L}$, consisting of all prefixes of strings of L,

$$\bar{L} := \{t \in \Sigma^* \mid t \text{ is a prefix of } s, \text{ for some } s \in L\}$$

Let us partition set of events Σ into the disjoint sets Σ_c, *controllable* events, and Σ_u, *uncontrollable* events. Controllable events are those events whose occurrence is either preventable (i.e. may be "disabled") or allowable (i.e. are said to be "enabled"). Uncontrollable events are those events which cannot be prevented and are deemed permanently enabled. A *supervisor* (or controller) may enable or disable controllable events at any time during its observation of a sequence of events generated by G. Thus supervisor allows only subset of $L(G)$ to be generated.

Formally, a supervisor $\mathcal{S}$ is a pair (S, Ψ) in which S is a automaton

$$S = (X, \Sigma, \xi, x_0, X_m)$$

where X is a set of *state* for the supervisor; Σ is the alphabet used by G; ξ is the *transition function*, a partial function $\xi : \Sigma \times X \to X$; x_0 is the *initial state* for the supervisor; X_m is the set of *marker states*; and Ψ, called *feedback map*, is given by $\Psi : \Sigma \times X \to \{0, 1\}$ satisfying:

$$\begin{array}{lll} \Psi(\sigma, x) = 1 & if & \sigma \in \Sigma_u, x \in X, \\ \Psi(\sigma, x) \in \{0, 1\} & if & \sigma \in \Sigma_c, x \in X \end{array}$$

The number 0 is interpreted as the command "disable" and the number 1 as "enable". The automaton $\mathcal{S}$ monitors the behavior of G and changes state according to the events generated by G. The control rule $\Psi(\sigma, x)$ dictates whether σ should be enabled or disabled at the corresponding state in G.

We interpret the operation of S as follows: The supervisor is initialized in the state x_0 and thereafter undergoes state transitions in response to the outputs of the plant G. At each state of S the feedback map Ψ selects a new set of enabled events for Σ_c and thus exerts some control over the future evolution of the plant.

The behavior of G when it is constrained by $\mathcal{S}$ is described by the automaton $\mathcal{S}/G$, called the *supervised discrete-event system*:

$$\mathcal{S}/G = (\Sigma, Q \times X, (\delta \times \xi)^{\Psi}, (q_0, x_0), Q_m \times X_m).$$

The behavior of $\mathcal{S}/G$ is described by $L(\mathcal{S}/G)$ and $L_m(\mathcal{S}/G)$. The modified transition function $(delta \times \xi)^{\Psi}$ is defined as a mapping $\Sigma \times Q \times X \to Q \times X$:

$$(\delta \times \xi)^{\psi}(\sigma, (q, x)) := \begin{cases} (\delta(\sigma, q), \xi(\delta, x)) \text{ if } \delta(\sigma, q)!, \\ \xi(\delta, x)!, \text{ and } \Psi(\sigma, x) = 1; \\ \text{undefined otherwise }. \end{cases}$$

A supervisor $\mathcal{S} = (S, \Psi)$ is *nonblocking* for G if:

$$L(\mathcal{S}/G) = \overline{L_m(\mathcal{S}/G)}$$

That is nonblocking supervisor ensures that, in closed-loop, any sequences s that is started (i.e. $s \in L(\mathcal{S}/G)$) can be completed to a marked sequence (i.e. $s \in L_m(\mathcal{S}, G)$).

The control problem can be stated as: given a plant G over an alphabet Σ (with controllable events Σ_c) and given some non-empty languages A and E where $A \subseteq E \subseteq L(G)$ find a nonblocking supervisor $\mathcal{S}$ such that:

$$A \subseteq L(\mathcal{S}/G) \subseteq E$$

What this formalism captures is problems where some process (say G) that can be described as a finite state machine is given , and some set of desirable (or legal) sequences is given (say E) and a controller is sought to inhibit process behavior so that only desirable sequences are generated. The language A describes the minimally acceptable set of sequences that any closed-loop solution must contain.

To describe a solution to the above problem, it is convenient to use the notion of *controllability.* Given G over an alphabet Σ, for a language $K \subseteq L(G)$, K is *controllable* with respect to G if:

$$\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$$

where $\overline{K}\Sigma_u := \{s\sigma \mid s \in \overline{K} \text{ and } \sigma \in \Sigma_u\}$. The controllability property may be paraphrased by saying, K is controllable if and only if no $L(G)$-string that is already a prefix of K, when followed by an uncontrollable event $\underline{\text{in}}$ G exits from the prefixes of K *the prefix language $\overline{K}$ is invariant under the occurrence of uncontrollable events in G.* If E is not controllable, a largest (or supremal) controllable sublanguage of E, denoted by $\sup\underline{C}(E, G)$ can always be found[8]. The standard solution to the control problem produces a supervisor that acts on G to generate $\sup\underline{C}(E, G)$

The following section highlights through example how some of the above descriptions can be applied to the case of multiple agents manipulating an object.

3 An Example

To demonstrate how this formalism can be apply to the object manipulation using multiple agents and to point-out the challenges and advantages of this formalism, let us consider the case where it is required to manipulate a held(grasped) object using four agents. Here the objective is to grasp the object and then change the orientation of it by using the relocating agents.

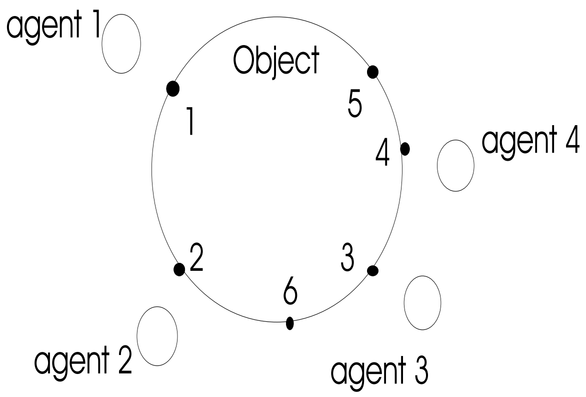

Figure 1: A schematic of agents and the object. The numbers represent he possible contact points.

The complete task can be described as: the *approach* phase where the agents move toward the object; the *grasping* phase where the motions of the agents changes from free to constrained motions, i.e. *discontinuity in motion.* The grasping phase can then be followed by manipulation phases. The manipulation phases can be a combination of agents pushing on the object, sliding along the object or relocating on the object.

As an example of the proposed frame-work and referring to Figure (1), let us describe the set of events be the set Σ:

$$\begin{aligned}\Sigma \;=\; &\{\text{approach(i) , retract-agent(i),}\\ &\text{contact-made(agent(i)), contact-break(agent(i))}\\ &\text{detect no force-closure, un-stable contact,}\\ &\text{constraint work-space, relocated agent(i)}\\ &\text{agent(i)/object misaligned}\\ &\text{relocate agent(i)}\\ &\text{constraint motion of agent(i),}\\ &\text{realigned agents, reorient the grasping forces,}\\ &\cdots\}\end{aligned}$$

The above present a sample of the list of events which can be generated by G. In the above, let us define the following:

- *approach(i)* is an approach of agent i towards the object.
- *retract-agent(i)* is the removal of agents i from the proposed contact location.
- *contact-made(agent(i))* is the detection of contact force between the agent and the object.
- *contact-break(agent(i))* is the detection of no contact force between the agent and the object.
- *no force-closure* is a presence of no static equilibrium condition in a quasi-static manipulation of th object. The condition can be simply stated that the net forces acting on the object should be end-up to be zero. This can be written as:

$$\mathbf{G}f = F_o$$

where $\mathbf{G} \in \mathbf{R}^{6\times p}$ where p is the number of contacting agents, $f \in \mathbf{R}^p$ is a vector of force magnitude of each contacting agents and $F_o \in \mathbf{R}^6$ is the external force vector acting on the object. In general all of the components in the above equation are described in th object coordinate frame system. The condition of the force-closure is equivalent to the rank of the matrix $\mathbf{G}$ to be equal to 6, or $rank(\mathbf{G} = 6)$. At each discrete contact configuration between the agents and the object, if the rank of $\mathbf{G}$ is less than 6, the condition will flag a non force-closure case, i.e. for a planar grasp, the rank condition is equal to 3.

- *un-stable contact* is the case for the normal component of force at each contact area not being within the friction constraint. This condition can be described as an unstable contact. Parameterizing the components of the grasping forces at each contact point, the constraint equation can be written as:

$$f_x^2 + f_y^2 \leq \mu^2 f_z^2 \text{ where } f_z \leq 0$$

- *agent(i)/object misaligned* indicates the condition that agent(i) can not obtain a proper orientation with the surface normal at the contact point.
- *relocate agent(i)* is the relocation of the agent (i) to a new location on the object.
- *constraint motion of agent(i)* is the motion of the agents while in contact with the object. In this case the servo-controller of each agent should regulate the desired set-point controller of the grasping force f while following a trajectory. This can be the pure translation of the contact points while the agents are in contact with the object. Or, it can be the rolling motion of the agent having for example the following constraint on the trajectory of the contact point:

$$v_x = v_y = v_z = \omega_z = 0$$

where x, y are in tangent plane to the object at the contact point and z is perpendicular to the tangent plane. Or the motion can be pure sliding of the agent on the object at the local frame where the constraint can be describe as:

$$\omega_x = \omega_y = \omega_z = v_z = 0$$

- *Constraint work-space(i)* indicates that the desired motion of the agents is beyond the indicated work-space of contacting agent. For example let $\mathcal{T}_i$ be the set of all points such that for all $r_i \in \mathbf{R}^3$ we have:

$$\mathcal{T}_i = \{r_i \in \mathbf{R}^3 \mid ||r_i||_{min} \le ||r_i|| \le ||r_i||_{max}\}$$

- *re-orient agent* is the re-orientation of the contacting agents such that the normal force are within the friction cone at each contact point.

The above is only an example of a subset of the actual list of events that the alphabet may contained.

A possible list of uncontrollable events can be represented as:

$$\Sigma_u \quad = \quad \{ \text{ no force-closure, unstable contact } \}$$

To construct a supervisor for G, a description of the desired or legal behavior of G should be defined. For example, the following can be a sub-set of such behavior for this example:

(i) agents(i) where $i = 1, 2, 3$ should approach the object before agent(4).

(ii) only two consecutive attempts are allowed for relocating agent four.

(iii) if the local slip occurs after the forces are applied, two attempts of the agents are allowed to adjust for the grasping forces.

Figure (2) shows a partial expansion of the legal description (i). This partial expansion is in the form of a finite-state automata which is required by the formalism. All the possible legal behavior of G can be expanded similar to Figure (2).

The control problem we are interested in requires that the supervisor $\mathcal{S}$ must impose the legal behavior or the largest controllable subset of legal behavior on G. Thus we need an automaton that recognizes only the desirable sequences as described by E. In addition we need a set of control rules that will indicate whether or not a given event at the current state of the plant is enabled or disabled. A supervisor that will solve our problem can be constructed as follows: first, we compute sup $\underline{C}(E, G)$, the largest controllable sublanguage of E; then we define a supervisor that ensures that only those strings and all those strings of G

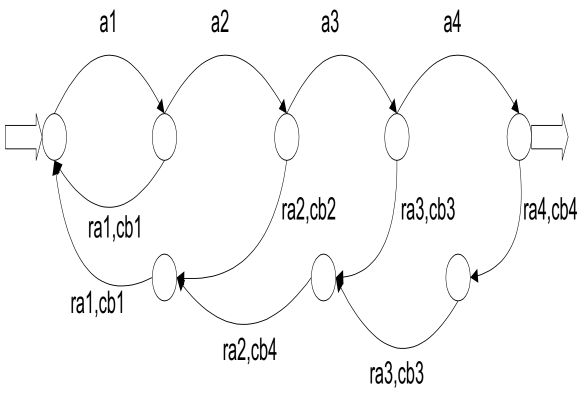

Figure 2: First specification of the legal behavior as a finite-state automata.

that are in this controllable sublanguage are permitted to occur. The minimally adequate language A will capture error-free executions of the task.

4 Discussions

This paper presented an overview of an extension of a modeling approach for controlling discrete-event systems. The main motivation for such extension is to point-out the lack of any formalism in the literature in the area of intelligent planning. Through the propose extension it may be possible for such formalism to be developed where the synthesis of *controlled planning* can be investigated.

One of the main challenges of such formalism is the description of any legal behavior of the multi-agent system in the form of finite-state automaton. In addition, as it was pointed-out the set Σ which can describe the possible events can be some-what arbitrary. Hence, part of the synthesis of the developing a supervisor is the determination of suitable set of events and the legal behaviors. Whenever verbal and/or mathematical specifications must be translated into a language (such as finite-state machines) used by the formalism, there is always the challenge of ensuring the error free translation.

A software tool called TCT[13] can be used to calculate all the description of the proposed extensions such as determining the sup $\underline{C}(E, G)$. Other techniques such as the approach proposed by [14] can be used to calculate the supervisory model proposed in this extension. For example, the work by [14] presents the control synthesis using technique of Binary Decision Diagrams. Future works are to study and analyze the supervisor proposed in this paper based on various definitions of legal behaviors, uncontrollable and controllable events defined in a given Σ.

References

[1] A. Salvarinov and S. Payandeh, *Motion Behavior of the Objects Manipulated by Active Fence (AF)*, proceedings of IROS'97, pp. 428-434

[2] A. Salvarinov and S. Payandeh, *Flexible Fixturing Tool for Prismatic Parts*, To be presented in IROS'98

[3] D. Chevallier and S. Payandeh, *On manipulation of Objects in Three-Fingered Grasp*, Transaction of ASME, Journal of Mechanical Design, Vol. 117, pp. 561-565, 1996

[4] M. Hong and S. Payandeh, *Design and Planning of a Novel Modular End-Effector for Agile Assembly*, Proceedings of 1997 IEEE International Conference on Robotics and Automation, pp. 1529-1535

[5] S.L. Ricker, N. Sarkar and K. Rudie, *A discrete-event Systems Approach to Modeling Dexterous Manipulation*, Robotica, Vol. 14, pp. 515-525, 96

[6] B.J. McCarragher, G. Hovland, P. Sikka, P. Aigner and D. Austin, *Hybrid Dynamic Modelling and Control of Constrained Manipulation Systems*, IEEE Robotics & Automation Magazine, June 1997, pp. 27-44

[7] S. Payandeh, *A Natural Framework for Designing Bounce-less Controller for Robotic Contact Tasks Problems*, Proceedings of IFAC, SYROCO Conference, 1997

[8] P.J. Ramadge and W.M. Wonham,*Supervisory Control of a Class of Discrete Event Processes*, SIAM Journal of Control and Optimization, Vol. 25(1), pp. 206-230, 1987

[9] B.A. Brandin, W.M. Wonham and B. Benhabib,*Manufacturing Cell Supervisory Control - A Modular Timed Discrete-Event System Approach*, Proceedings of IEEE International Conference on Robotics and Automation, 1993, pp. 846-851

[10] B.J. McCarragher and H. Asada,*A Discrete Event Approach to the Control of Robotic Assembly tasks*, Proceedings of IEEE International Conference on Robotics and Automation, 1993, pp. 331-336

[11] A. Ordean and S. Payandeh, *Design and Analysis of Enhanced Opportunistic System for Grasping Through Evolution*, Proceedings Third Conference on Advanced Robotics, Intelligent Automation and Active Systems, 1997, pp. 239-244

[12] T.M. Sobh and R. Bajcsy,*Autonomous Observation Under Uncertainty*, Proceedings of the IEEE International Conference on Robotics and Automation, 1992, pp. 1792-1798

[13] TCT,*Dept. Electrical Engineering, University of Toronto, Systems Control Group*, 1996

[14] S. Balemi, G.J. Hoffman, P. Gyugyi, H. Wong-Toi and G.F. Franklin, *Supervisory Control of a Rapid Thermal Multiprocessor*, IEEE Transaction on Automatic Control, Vol. 38, No. 7, 1993