Abstract: This paper presents a hybrid automata approach for modeling and analyzing the robotic motion planning and control. Robotic systems acquire data from perceptive sensors and respond to the perceptions through decision and control process. The hybrid perceptive framework is built based on continuous and discrete perceptive references. The robot can plan and modifying the original path through switching the continuous controller at discrete level. The hybrid system is stable in switching. The evolution of the hybrid references can be guaranteed during the unexpected events that can block the continuous or discrete references. Copyright © 2005 IFAC.

Keywords: Hybrid Automata, Perceptive Reference, Unexpected Events.

1. INTRODUCTION

As autonomous systems, robots process perception, the environmental information obtained from onboard sensors, and respond to the perception by changing the original path planning and control schemes. A perceptive frame can be built for robotic systems modeling and analysis. The perceptive framework introduces a concept of a perceptive reference $s$, a parameter that is directly relevant to measured sensory outputs and the task. The $s$ can be, for example, the distance the robot system traveled. Thus, instead of time, the control input is parameterized by the new reference, which is a function of the real time measurement. The planner is triggered by the new reference and generates the desired values of the system, according to the on-line computed reference $s$. The perceptive frame has been applied to the path planning and control for a single manipulator and multiple manipulators coordination with a continuous perceptive reference (Xi et al., 1996) A perceptive scheme was developed for integration of task scheduling, action planning and control of robotic manufacturing systems in (Song et al., 2000). The perceptive approach guarantees the stability of the robot system in the presence of unexpected events. However, unexpected events become serious problem, as they can block the evolution of the reference, for example, an obstacle. When the reference is blocked by an unexpected obstacle, the robot will stop until the blocked reference is released and then resume evolution. It is crucial for the reference to keep evolving. A hybrid perceptive framework is considered to solve the break in evolution of a single reference.

Hybrid systems have been attracting attention from control engineers recently and extensively investigated in (Lygeros et al., 2003), (Braniky, 1995), (Pettersson and Lennartson, 1993) and a variety of hybrid system models and frameworks have been proposed (Grossman et al., 1993) (Antsaklis et al., 1993) (Braniky, 1995).
expressed using hybrid languages. Carrying commands and perceptive references, is tasks and actions to execute. The information, framework, the discrete level references trigger the model are modeled by hybrid automata. in this major parts, a hierarchical command execution and a hybrid perceptive reference. All the units of the system governs the system based on perceptive information such that the system can keep the reference evolving through modifying original plans.

In this paper, using a hybrid framework, the hybrid perceptive references enable both the continuous and discrete part of the system to deal with multiple interruptions. In Egerstedt’s research, a complexity problem of a multiple obstacle avoidance is analyzed by using motion description language of Brockett’s hybrid model (Egerstedt, 2001). However, both the continuous part and the discrete part of the system use time reference.

In this work, a hybrid framework, the hybrid perceptive references enable both the continuous and discrete part of the system to deal with unexpected events. The discrete part of the system governs the system based on perceptive information such that the system can keep the reference evolving through modifying original plans.

2. HYBRID PERCEPTIVE FRAMEWORK

As described in (Sun et al., 2003), the task set $T$, action set $A$, corresponding reference sets $R_{Task}$ and $R_{Action}$, and continuous perceptive reference $s$, are defined in perceptive frame.

The formal languages $L_A, L_T, L_{RA}$ and $L_{RT}$ can be built by the alphabets $\Sigma_A, \Sigma_T, \Sigma_{RA}$, and $\Sigma_{RT}$ that are generated by sets $A, T, R_{Action}, R_{Task}$, respectively.

A Hybrid Language (HL) is generated from an original alphabet $\Sigma$, consisting of discrete atoms, and numbers $N \in R^n$. The alphabet of the HL is an extended alphabet of a formal language. An HL is defined as the set of all the strings over the extended alphabet $\Sigma^H$, a string is concatenation of the elements lying in the product set of the original alphabet and the Euclidean spaces. An HL derived from the extended alphabet $\Sigma^H$ is denoted $L^H$. A word in a hybrid language $L^H_T$ can be “$T_1(x, y)$” meaning “going to point(x,y)”. The alphabet gives the qualitative description, the numeric part is the quantization of the linguistic expressions.

A perceptive automaton can be defined as a tuple

$$M_e = (Q, \Sigma, \Sigma_R, \delta, Q_0, \eta, O),$$

where $Q$ is the set of the discrete states of the automaton and $\Sigma$ is the discrete control input set.

The following properties describe the state transitions and the output action triggered by the control command input and the reference inputs.

$$q_j = \delta(q_i, \sigma_j), O_k = \eta(q_i, \sigma_k)$$

where $q_i, q_j \in Q, O_k \in \Sigma, \sigma_j \in \Sigma, \sigma_k \in \Sigma_R$. Corresponding different inputs the automaton can have different responses including state switching and command issuing. A hybrid automaton is capable of bridging the two kinds of variables. Based on (Lygeros et al., 2003) it can be considered to be a tuple

$$M^H = (Q, X, \Sigma, \delta, Q_0, \eta, 0),$$

where $X$ is the set of continuous variables. The automaton can be described as

$$q_j = \delta(q_i, \sigma_j, X), \sigma_j \in \Sigma$$

$$O_k = \eta(q_i, \sigma_j, X)$$

A hierarchical automaton consists of several nodes, some of which are other automata. The first automaton is an upper automaton, the other automata denoting the nodes of the upper automaton are called embedded automata. This architecture can be described as a tuple, $M_h = (Q_U, Q_{EM}, \Sigma_U, \Sigma_{EM}, \delta_U, \delta_{EM}, Q_{U0}, Q_{EM0}, O)$.
\( \Sigma_U \) is the input set for the upper automaton, \( \Sigma_{EM} \) is the input set of embedded automaton, \( Q_U \) is a set of the states (the nodes of the upper automaton), \( Q_{EM} \) is a set of the states of the automata embedded into nodes of \( Q_U \), \( Q_{U0} \) is the set of the starting states of \( Q_U \), \( Q_{EM0} \) is the set of the starting states of \( Q_{EM} \), \( \delta_U \) and \( \delta_{EM} \) describe the transition functions of \( Q_U \) and \( Q_{EM} \), respectively.

Perceptive, hybrid and hierarchical automata refer to three basic prototypes of automata in the framework.

2.2 Hierarchical Task Execution Architecture

As shown in Fig. 1, the task execution includes three parts: task scheduler, action and motion planner, which can be modeled using automata.

The task scheduler sequentially generates tasks according to the commands and discrete task level references. It is a perceptive automaton hybrid language. It has perceptive reference inputs in hybrid language. The inputs from the Task Level Reference are mechanism to trigger the next level references. The reference generation can be issued based on the configuration sensors. The environmental sensory measurements are used by the higher levels for discrete perceptive reference generation, and the motion reference can be issued based on the configuration sensor measurements. Reference generation can be described as automata. Task and action level references are mechanism to trigger the next task or action based on the environmental sensor measurements and unexpected events.

The system has configuration sensors and environmental sensors. The environmental sensory measurements are used by the higher levels for discrete perceptive reference generation, and the motion reference can be issued based on the configuration sensor measurements. Reference generation can be described as automata. Task and action level references are mechanism to trigger the next task or action based on the environmental sensor measurements and unexpected events.

The detailed description of reference generation can be found in (Sun et al., 2003).

2.3 Reference Generation

The motion planner in Fig. 1 can generate a continuous trajectory for the desired motion, based on the actions from the action planner. The motion planner is a hybrid automaton triggered by the perceptive reference. We describe it as a tuple:

\[ M_{MotPlan} = (Q, \Sigma_U, \delta_U, \sigma_U, \eta_{MotPlan}) \]

The output is parameterized by reference \( s \). Another input of the automaton is in hybrid language \( L_A^H \) from the alphabet \( \Sigma_A^H \). The discrete part of the input \( \sigma \) in \( L_A^H \) make the automaton switch to the appropriate node. Each node of the automaton is a dynamic system (or a controller) which can issue the planned continuous trajectory.

Therefore, it can be formulated as:

\[ q_{MotPlan} = \delta(q_{MotPlan}, \sigma_j), \sigma_j \in \Sigma_A \] (10)
\[ O_{MotPlank} = \eta(q_{MotPlan}, s) \] (11)

It can be seen that the state transition and output are triggered separately by two different inputs. For motion actions, the output to the robot controller can be a vector, \( O_{MotPlank} = [X(s), Y(s), X(s), Y(s)]^T \), for a complete motion trajectory.

3. HYBRID PERCEPTIVE CONTROL FOR MANIPULATION AND TELE-OPERATION

A typical task is to pick up an object and transport it to a desired destination with a mobile manipulator. In the route of transporting the object, an unexpected obstacle occurs. In order to complete the task, the mobile manipulator will execute an obstacle avoidance task. The mobile manipulator will then resume its original task and reach the destination.

To build the hybrid languages from task set and action set, some continuous variables can be involved to the atoms. In the extended task set,
Similarly, the action space can be defined as “transporting an object from point \((m_1,n_1)\) to point \((m_2,n_2)\).”

\[ T2(m,n,l_o) \text{ denotes “to avoid an obstacle at}\,(m,n)\text{ at the distance } l_o. \]

In the extended action set, \(A1(m,n,l)\) denotes “following a straight line with directional cosines \((m,n)\) and length \(l.\)” \(A2(m,n,r)\) denotes “go through an arc centered at \((m,n)\) with length \(r.\)” \(A3\) and \(A4\) denote “close the gripper” and “open the gripper”, respectively.

In manipulation, the units behaves as follows:

**Task Scheduler:** The output of the task scheduler is \(T1(m1,n1)(m2,n2)\) until the obstacle has been detected. **Action Planner:** Based on \(T1(m1,n1)(m2,n2)\), the transition and the outputs are \(q_{AP1} = (q_0,T1)\) and \(o_{AP1} = \eta(q_{AP1},\text{reset}) = A1\). When the planner has the reference input \(A1\) meaning that the \(A1\) action is finished, the output action triggered by \(A1\) is a vector \(o_{AP1} = \eta(q_{AP1},A1) = [A2,A3]^T\). The output triggered by action reference \(A1\) is another action \(o_{AP1} = \eta(q_{AP2},A11) = A4\). The directional cosines and the length in \(A1\) and \(A11\) can be found with the parameters in \(T1\).

**Motion Planner:** The inputs \(A1(m1,n1,11), A11(m11,n11,11) \in \Sigma_{AR}^H\) cause the states transition of the motion planner. \(q_{MotPlan1} = \delta(q_0,A1), q_{MotPlan11} = \delta(q_0,A11).\) The states denote different vector fields, i.e., trajectories, namely, straight line path and arc path.

## 4. ANALYSIS ON SWITCHING SYSTEM

During switching between the tasks, which can be described in subspaces, stability is a very crucial issue. The values of the vector in each subspace represent the parameter disturbance of the low level controllers.

### 4.1 Mapping from high level task to low level controllers and regions

Task space can be defined as a N-dimensional linear vector space.

\[
T^s = [T_1T_2...T_n]^T
\]

\[
T^s = \text{diag}[0...I_p...0]
\]

\[
T^s = [0...x_{i1},x_{i2}...x_{ip},0]^T
\]

Where \(I_p\) is a diagonal matrix with order \(p\). Therefore, for each task, we have a multidimensional vector space. For several independent tasks, the combined task space can be generated and, each task is defined in a subspace. For example, the \(T_i\) is a task in \(p\) dimensional space. The dimension of the subspace depends on the specific task. Similarly, the action space can be defined as:

\[
A^s = [A_1A_2...A_n]^T
\]

\[
A_j = \text{diag}[0...D_j...0]A^s
\]

Where \(D_j\) is a diagonal matrix with order \(q\). The transform would result in mapping the task from a task subspace into an action subspace. The mapping \(T^s \rightarrow A^s\), therefore, can be described as a linear mapping.

According to the control model of action planning,

\[
O_k^A = L(q_1,\sigma_i,\sigma_k) = L'(q_i,\sigma_k, T^s)
\]

Where \(D_i\) is a diagonal matrix. The switch between the task level inputs is equivalent to the switch of the Linear mappings. Given a different task level input, there exist a linear mapping to generate the the vectors in Action Space.

\[
O^M = L_M(q_i^M, \sigma_i, \sigma) = L''(q_i, s, A^s)
\]

Where the \(D_j\) is a diagonal matrix. \(L'\) and \(L''\) are linear mappings.

Fig. 1 describes the model with hybrid perceptive references and states. A hybrid perceptual trajectory is a finite or infinite sequence of reference interval \(e = \{S_{TRi}S_{ARj}\}S_K\), where \(S_K \in [S_i, s_i^*], S_{TRi} \in \Sigma_{RT}^H, S_{ARj} \in \Sigma_{RA}^H\). Furthermore, an execution of a hybrid perceptual automaton describes a collection \(\chi = \{e, q, x\}\), where \(e\) is a hybrid perceptual reference trajectory, \(q\) is a map from \(S_{TR}S_{ARj}\) to \(q, x\) maps the continuous reference \(s\) to the continuous state space.

### 4.2 switching conditions for low level controllers

For the system \(dx/ds = f(x)\), we say that \(V(s)\) is a candidate Lyapunov function if \(V(s)\) is a continuous positive definite function (about the origin) with continuous partial derivatives. Note this assumes \(V(0) = 0\).

Considering switched hybrid systems, the systems operate in a hybrid metric space. The stability property of such systems can be described in a hybrid state space of the perceptive frame. Therefore, multiple Lyapunov function can be used to discuss the stability issue.

Given a dynamical system in the perceptive frame, if there exist Lyapunov functions \(V_1(s^h), V_2(s^h)\) with hybrid perceptual references, corresponding to the hybrid perceptual reference trajectory and different segments of executions, for a given strictly increasing sequence of \(S_{TRi}S_{ARj}\) (discrete reference values), denoted by \(I = S_{TRi}S_{ARj}\) in perceptive reference, we say that \(V\) is a Lyapunov-like function for function \(f\) and execution \(\chi = \{e, q, x\}\), if
1. Given $s \in (s_i, s_{i+1})$ where $dV(x(s^h))/ds \leq 0$
2. $V$ is monotonically non-increasing on ordered set $I$.

Theorem 1 (Stability): Suppose we have candidate Lyapunov functions $V_i, i = 1 \ldots N$, and vector field $dz/ds = f_i(x)$ with $f_i(0) = 0$. Let $E$ be the set of all switching sequences associated with the system. If for each $s^h \in E$ we have for all $i$, $V_i$ is Lyapunov-like for $f_i$, then the system is stable in the sense of Lyapunov.

Theorem 2 (Exponential Stability): Suppose we have candidate Lyapunov functions $V_i, i = 1 \ldots N$, and vector field $dz/ds = f_i(x)$ with $f_i(0) = 0$. Let $E$ be the set of all switching sequences associated with the system. In the perceptive frame, if for each $s^h \in E$ we have for all $i$, $V_i$ is Lyapunov-like for $f_i$, then the system is exponentially stable in the sense of Lyapunov. If there exist constants $\alpha_i, \beta_i, \gamma_i, g = 1, \ldots, l$, such that:

\[ -\forall x \in \Omega, \alpha_i(\|x\|) \leq V_i(x) \leq \beta_i(\|x\|) \]
\[ -\forall x \in \Omega, \]
\[ -\forall x \in E_{gr}, V_i(x) \leq V_i(x) \]

for switching from $q$ to $r$.

4.3 continuity of input-output mapping

The offset on the continuous variables of the task level linguistic inputs. For the tasks, which have the same discrete task but different continuous variables, i.e., two task controls in the same task subspace, if the continuous parts approximate sufficiently, then the states of the system are sufficiently close to each other.

It can be formally described by the following:

For a given task subspace $T_i$, the mapping $L$ is referred to as a continuous mapping, if there always exists an $\epsilon$, for given $\delta$ such that $\|A\| < \epsilon$, when $\|\Delta T\| < \delta$. The intuitive meaning is that both the task and the action will evolve continuously. The norm $\|A\|$ is referred to as the sum of all the action offset. I.e., $\|A\| = \Sigma \|A_i\|$. Thus, the input and output of the task controller in the linear space $T^s$ and $A^s$ are continuously dependent.

4.4 stability conditions for switched systems

The stability conditions include two parts. First, the switching between the task subspace with different dimension is stable under certain conditions of Lyapunov functions. Second, the continuous variable of the task input do not spoil the stability, if for a given value, the system is stable.

Assumption: The input and the output of the task controllers in linear space are continuous dependent.

Claim 3: If the vector fields corresponding to the switching task $T_1$ and $T_2$ are exponentially stable. Then given any continuous part $T_1$ and $T_2$, the switching tasks will be stable.

5. Modeling of Unexpected Event Processing

The ability to deal with unexpected events is crucial for a perceptive control system. $Q$ represents the set of the vector fields, $(q_1, \ldots, q_n)$, for a $m$-dimensional space, at least it has $m$-dimensional orthogonal vector fields. $f_1 \ldots f_m$. Therefore, for any 2 points $x_1$ and $x_2$, the system can start from $x_1, x_2$ within finite time of switches over the vector fields. The continuous reference $s$ should satisfies: $L_{s^h}S(q, x) > 0$, it guarantees that the continuous reference is increasing over time $t$.  

5.1 Unexpected Event (UE) Processing

Unexpected events can be described as follows:

For a given $e = \{S_{TRi}, S_{ARj}, K\}$, where $S_K \in [s_i, s_{i+1}], S_{TRi} \in \Omega^R, S_{ARj} \in \Omega^A$, and an execution of a hybrid perceptive automaton $\chi = (e, q, x)$. A unexpected event happens, when the following two condition hold:

\[ ds/dt, s_{u^h} = 0, s_{u^+} \in (s_{h}, s_{j}), \]
\[ s_{u^h} \notin S_{TRi}, S_{ARj}. \]

Furthermore, an unexpected event represents a convex region $U_A$ that is an open set of states, in which the above conditions hold, the boundary is a scalar function $\omega(x) = 0$, and $\omega(x) > 0$ for $x \in U_A$, when UE happens, using “Lie Derivative”, we have $L_{sf}(x) > 0, \omega(x) = 0$.

It is a local blocking event of Task $T$ if we can find a subset of the trajectory $\chi_1 = \{s_i, q, x\}$, where $s_t = (s_{h}, s_{j}) \notin U_A$.

For hybrid automata, the unexpected events can be described as a disturbance for a system. The system is desired to be able to return to the designed trajectory after the disturbance. Or, the unexpected events can be treated as a task executed before finishing the current task. The new task is to switch out of the blocking state then return to and resume the previous task. For the given model, the unexpected event will affect the states, causes a discrete state transition. A new vector field will be applied on the continuous states of the system. which is an orthogonal vector field to the old one, $< f_i(x), f_{i+1}(x) >= 0$, then it
can satisfy the following conditions: $L_{f_{i+1}}, \omega(x) < 0$, $L_{f_{i+1}}, S(q, x) > 0$, the former makes the system leave domain $U_A$ from the boundary $\omega(x) = 0$, the latter guarantees the evolution of the continuous reference.

Theorem 4 shows the existence of the solution at unexpected events in the perceptive frame.

**Theorem 4:** if the unexpected event $e_u$ is a local blocking event, there exists another finite automaton and with the same alphabet, corresponding to the new automaton, the automaton can go back to the original task trajectory $\chi = \{e, q, x\}$.

The detailed proofs of Theorem 1-4 can be found in (Sun, 2004).

### 5.2 UE Processing in Mobile Manipulation

After an UE that blocks the evolution of continuous reference is detected, the task reference generates the UE with the obstacle info $T2(m_o, n_o, l_o)$, the output of the task scheduler is $T2(m_o, n_o, l_o)$. It causes a discrete switch over discrete variables, an obstacle avoidance on the task execution to switch and resume the evolution of the continuous reference, which leads the robot to the previous task and reach the destination. The perceptive reference is never blocked.

The transition function and the output function are $q_{AP2} = \delta(q_{AP1}, T2(m_o, n_o))$ and $\eta(q_{AP2}, reset) = A2(m_{a2}, n_{a2}, r_{a2})$, respectively.

The motion planner receives the action $A2(.)$ from the action planner to generate the trajectory. This automaton has a state transition on the upper automaton due to the unexpected event and returns to the previous state after processing the unexpected event. The triggered procedure is the same as in the task execution process.

In teleoperation, The time delay of the communication between the operator and controller will significantly affect the control performance, it is crucial to synchronize the command from the operator to the arm controller. If the controller cannot receive the next position or velocity command before it finishes executing the current one. It has to stop or just keep going on the current velocity, the former reduces the smoothness of execution while the latter causes control errors. In the hybrid model, the delayed command is thought of as an unexpected event on action level, that can prevent the discrete reference from evolving.

The routing for the UE is to switch to another action which accelerates the motion to make the trajectory smooth until the next control command is received then switch back to the previous state and the path plan is modified. As the result, the hybrid perceptive reference can not be blocked.

### 6. CONCLUSION

This paper presents a hybrid model for the perceptive robotic systems. The model integrates the continuous perceptive reference and discrete references for planning and control using hybrid automata. Compare to continuous reference, the hybrid perceptive reference keeps evolving despite the occurrence of an unexpected event, which could block the continuous perceptive reference or the discrete perceptive reference. It can be shown that the system is stable during the switches.

**REFERENCES**


Song, M., T.J. Tarn and N. Xi (2000). Integration of task scheduling, sensing, planning and control in robotic manufacturing system. Proceedings of the IEEE.

