The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Instant Prediction for Reactive Motions with Planning

Hisashi Sugiura, Herbert Janßen, and Christian Goerick

Abstract— **Reactive control and planning are complementary methods in robot motion control. The advantage of planning is the ability to find difficult solutions, optimize trajectories globally and not getting stuck in local minima but at higher computational cost. On the other hand, reactive control can handle dynamic or uncertain environments at low computational cost, but may get stuck in local minima.**

In this paper, we propose a new approach to integrate both reactive control and planning using a short time prediction. The system is mainly composed of a predictor, a planner and a reactive controller. The system uses the planner to modify the target for the reactive controller. The predictor simulates the robot future states and evaluates the reactive motion to trigger the planner in advance.

The latency due to the high computational costs for planning is compensated since the motion is already simulated, therefore, the resulting motion is smoother than without the predictor.

When the system's environment becomes more uncertain and dynamic, the system works reactively and iterates faster so that the system adapts to the environment automatically.

We tested the scheme in a simulator and realized it on our humanoid robot ASIMO.

I. INTRODUCTION

Traditionally, robots have been working in static and well-known environments. Planning methods which optimize trajectories globally are appropriate for these environments.

Nowadays, some kinds of robots, for example, humanoid robots are expected to work in human environments which are uncertain and dynamic. Using planning methods in these environments is not always suitable because computational costs and thus latencies are much higher compared to the sampling time of the robots' control loop.

In order to reduce the computational costs in high dimensional space, many planners have been proposed. One of the most popular methods for planning on humanoid robots is the Rapidly exploring Random Trees (RRT) method [1]. Based on this method, some researchers tried to reduce the cost for planning further. Yoshida proposed multi-level degrees of freedom (DOFs) exploitation [2] which reduces the dimensionality of exploration depending on the environmental constraints. Vahrenkamp et.al. proposed a method to reduce joint space dimensionality [3] and they also proposed a fast collision checking with a coarse collision model [4].

These approaches work quite well in static and semi-dynamic environments, however, in highly dynamic environments which change every sampling time, they may not always be able to react in time to changes in the environment and dimensionality cannot always be reduced.

The authors are with the Honda Research Institute Europe GmbH, Carl-Legien Strasse 30, D-63073 Offenbach/Main, Germany. hisashi.sugiura@honda-ri.de

To integrate reactive control and planning is another approach to handle dynamic environments. Layered or hierarchical architectures are commonly used for this [5], [6]. Gat proposed a three-layer architecture which comprises a controller, a sequencer and a deliberator [7]. There are some controllers in the control layer which are selected by the sequencer at a given time.

Ranganathan proposed a system which can switch between reactive control and planning in a three layer architecture [8]. When the robot takes longer than a given time to track the given trajectory with the reactive controller, the system switches to the planner. If the angle deviation between motion directions drops below a certain threshold, the system switches back to the reactive controller.

These layered approaches work nicely for dynamic environments. They were tested on mobile robots which have multiple DOFs and the criterion for switching between reactive and planning is computed based on the resulting trajectories the robot has already executed without prediction.

There are some search techniques in discrete space which are applied to real time planning in dynamic environments. For example, Koenig proposed the Real Time A* (RTA*) [9] and Stentz proposed the Dynamic A* (D*) [10] which are an extension of the A* [11] search algorithm for real time search in dynamic environments.

These work fine in a low dimensional space but in a higher dimensional space, the computational cost to discretize the space into many grids is much higher.

Our target is to realize a system that can be used in complex, uncertain and dynamic environments with smooth trajectories. Our proposal integrates reactive control and planning, and it works on robots with many DOFs at low computational cost.

We propose the *instant prediction* scheme which is mainly composed of a predictor, a planner and a reactive controller. If the predictor anticipates a difficult situation, the robot can escape the situation in advance. Since the planner is able to start planning from the state the predictor has already predicted, the latency due to the higher computation cost for planning is compensated. The system automatically changes the influence of planning for the overall resulting motion depending on its environment. If the environment is more uncertain or highly dynamic, the planning time is reduced and the system will work reactively. If the environment is relatively static, then the system allows the planner to use sufficient time for planning and the resulting motion is more optimal.

We apply this scheme to our humanoid robot ASIMO and realize motions that do not get stuck in local minima.

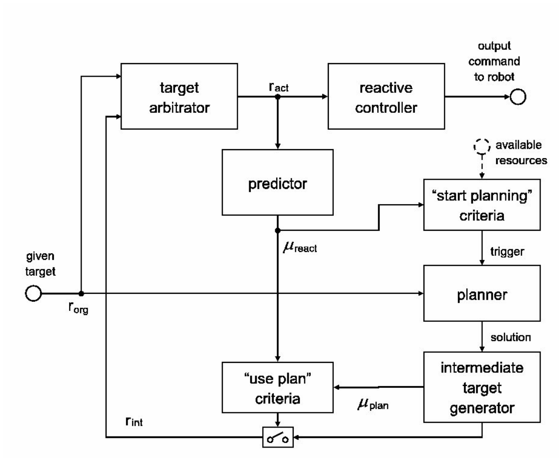


Fig. 1. The instant prediction scheme

This paper is organized as follows. In the next section we describe the instant prediction scheme. The predictor, the planner, the reactive controller and other functions are described here. Section III presents how the system works adaptively in its environment and Section IV shows how the system is implemented. In Section V we demonstrate the capabilities of the scheme in some scenarios on the simulator and on the robot. Finally we discuss and conclude in Section VI.

II. Instant prediction scheme

The overall scheme of the instant prediction is described in Fig. 1. The system uses a reactive controller to generate motion commands which are sent to the robot. The predictor simulates the future robot state using its internal model, the so-called *internal predictor model* [12]. If the fitness of the motions of the model μ_{react} is insufficient, the planner is triggered. If the fitness of another motion generated by the planner μ_{plan} is sufficient, the motion is given to the reactive controller as the actual target $\boldsymbol{r}_{act}$ through the target arbitrator.

The elements of the scheme are described below along Fig. 1. In the following sections, T_* represents the overall computation time for each internal model during N_* iteration cycles, ΔT_* represents the computation time for each internal model during one iteration and Δt_* represents the sampling time for each internal model, respectively. The suffix $*$ is *predict*, *react*, *plan* or *trig*.

A. Target Arbitrator

The target arbitrator is activated when a new target $\boldsymbol{r}_{org}$ or an intermediate target $\boldsymbol{r}_{int}$ is given. When it receives the intermediate target, it stores the original given target $\boldsymbol{r}_{org}$ and sends $\boldsymbol{r}_{int}$ to both the reactive controller and the predictor with time intervals of $\boldsymbol{r}_{act}$. When it receives a new given target $\boldsymbol{r}_{org}$, the stored data is discarded and it sends the new given target of $\boldsymbol{r}_{act}$.

B. Reactive Controller

The reactive controller receives $\boldsymbol{r}_{act}$ and computes the motion for the next sampling time based on the robot model, the so-called *internal reactive model*. The resulting motion commands are sent to the robot. Usually the reactive controller comprises some controllers such as a motion controller, a joint limit avoidance controller, a collision avoidance controller and so on [13].

C. Predictor

The predictor simulates the robot's future state by iterating the robot model, the so-called *internal predictor model*. The model can differ from the internal reactive model so that the predictor iterates the model faster and predicts further into the future. For example, using fewer DOFs than the real robot or using a longer sampling time interval $\Delta t_{predict}$ than the internal reactive model's sampling time Δt_{react}.

In contrast, the coarseness of the internal predictor model causes inaccuracy of the prediction. A finer model can be used in these cases:

- when μ_{react} becomes smaller.
- when the system predicts the near future.
- when the system predicts close to the given target $\boldsymbol{r}_{org}$ [3].

The predictor computes the fitness μ_{react} based on the model. This is a quality measure for the reactive motion which is simulated with the internal predictor model. The fitness μ_{react} represents the complexity of the robot's environment regarding the reactive motion. For example, when the internal predictor model moves closer to physical obstacles and the model reaches a dead-lock, then μ_{react} is reduced. The predictor stops predicting until the planning is finished in order not to re-trigger the planner during planning.

D. "Start planning" Criteria

The "start planning" criteria determines whether to start planning or not according to the fitness μ_{react} and the available resources such as computer resources if these are limited. For example, if μ_{react} drops below a certain threshold and the system has sufficient resources, then it starts planning.

E. Planner

The planner generates solutions (usually trajectories) which are locally or globally optimized. Any kinds of planners can be used here, however, faster planners or planners which give a minimal solution even if they are interrupted during the computations are especially suitable since the time for the planning is restricted.

F. Intermediate Target Generator

The intermediate target generator converts the solution from the planner to the intermediate targets $\boldsymbol{r}_{int}$ so that the fitnesses for both the reactive motion and the planned motion can be compared and the target arbitrator can deal with it. For example, in case the planner uses configuration space and the given target $\boldsymbol{r}_{org}$ uses task space, the intermediate target generator converts the configuration space to task space. Based on $\boldsymbol{r}_{int}$ it computes the fitness μ_{plan} and sends it to the "use plan" criteria.

G. *"Use plan" Criteria*

The "Use plan" criteria compares the fitnesses μ_{plan} and μ_{react} which represent quality measures of the planned motion and the reactive motion respectively. The motion which has the larger fitness is chosen and its intermediate targets are used for $\boldsymbol{r}_{act}$.

III. Adaptive Planning

The predictor is able to predict up to the robot future state at the time $t_{predict}$ which is the prediction depth.

Let the computation time for the reactive controller be T_{react}, for the predictor be $T_{predict}$ and for the planner be T_{plan}, respectively. For the latency compensation, the following condition is necessary.

$$t_{predict} \geq T_{plan} + T_{predict}. \tag{1}$$

Let the prediction iteration cycle for $t_{predict}$ be $N_{predict}$ and the computation time for one cycle be $\Delta T_{predict}$. We obtain,

$$t_{predict} = N_{predict} \Delta t_{predict}, \tag{2}$$

$$T_{predict} = N_{predict} \Delta T_{predict}. \tag{3}$$

Under these conditions, Eq. (1) becomes

$$T_{plan} \leq N_{predict}(\Delta t_{predict} - \Delta T_{predict}). \tag{4}$$

If $\Delta t_{predict} - \Delta T_{predict}$ is constant, $N_{predict}$ restricts the computation time for the planning T_{plan}.

In static or certain environments, $N_{predict}$ can be sufficiently large for planning. However, if the environments are dynamic or uncertain, the predictor does not always predict the robot's future state up to $t_{predict}$ because the planner can be triggered before $t_{predict}$. The time t_{trig} between the start of the predictor and the triggering of the planner is

$$t_{trig} \leq t_{predict}, \tag{5}$$

$$t_{trig} = N_{trig} \Delta t_{predict}. \tag{6}$$

where N_{trig} is the prediction interaction cycle for t_{trig}.

Eq. (4) can be formulated as

$$T_{plan} \leq N_{trig}(\Delta t_{predict} - \Delta T_{predict}). \tag{7}$$

The planner should generate trajectories in the more restricted time t_{trig}. At the same time, the overall system loop time is reduced.

In other words, the system works more reactive when the environment becomes more dynamic or uncertain. The time chart is visualized in Fig. 2.

If $N_{predict} = 0$, the system is equivalent to the reactive control. If the planner does not satisfy Eq. (7), the system works equivalent to a planner without the prediction.

For the dynamic environment, $t_{predict}$ should be short so that the predictor updates μ_{react}, which represents the environment, frequently.

IV. Implementation

We applied the instant prediction scheme to our humanoid robot. The elements which are explained in the previous section are implemented as follows.

A. *Target Arbitrator*

The target arbitrator receives intermediate targets $\boldsymbol{r}_{int}$ with their timing and then they are sent to the reactive controller and the predictor. We use attractor points [14] as $\boldsymbol{r}_{int}$.

B. *Reactive Controller*

The reactive controller comprises the whole body motion controller [15] and the collision avoidance controller [13]. The two controllers are dynamically blended depending on a danger measure. The computation time of the reactive controller changes depending on the environment.

1) Whole Body Motion Controller: The whole body motion controller is based on a redundant control with null space criteria. It includes the joint limit avoidance and the singularity robustness in its task space or null space.

2) Collision Avoidance Controller: The collision avoidance controller effects the robot's two closest segments based on the distance computation. The distance computation uses a coarse model which is composed of sphere swept lines and spheres instead of finer models such as polygon models. This drastically reduces computation time. Based on the result of the distance computation, the collision avoidance generates virtual repulsive forces in order to separate two close segments. These forces are inversely proportional to the distance between segments if the distance becomes smaller than a certain threshold [13].

C. *Predictor*

The fitness in the internal predictor model uses the virtual forces computed in the collision avoidance controller. It represents the robot's physical environment. The fitness μ_{react} is computed as follows,

$$\mu_{react} = \boldsymbol{m}_{at} \cdot \alpha \boldsymbol{F}_{virtual}, \tag{8}$$

$$\boldsymbol{F}_{virtual} = \sum_{i=0}^{n-1} \boldsymbol{f}_i, \tag{9}$$

where $\boldsymbol{m}_{at}$ is a unit vector from the current robot state to the actual target state $\boldsymbol{r}_{act}$, α is a coefficient and $\boldsymbol{F}_{virtual}$ is the overall virtual force vector which the robot receives composed of virtual force vectors $\boldsymbol{f}_0, \boldsymbol{f}_1, ..., \boldsymbol{f}_{n-1}$. The each virtual force vector $\boldsymbol{f}_i$ is computed as follows,

$$\boldsymbol{f}_i = \begin{cases} k(d_a - d_i)\boldsymbol{e}_i & \text{if} \quad d_i < d_a, \\ \boldsymbol{0} & \text{otherwise}, \end{cases} \tag{10}$$

where k is a positive constant, d_i is a distance between closest points, d_a defines the threshold distance which generates the virtual force and $\boldsymbol{e}_i$ is the unit vector for the avoidance direction.

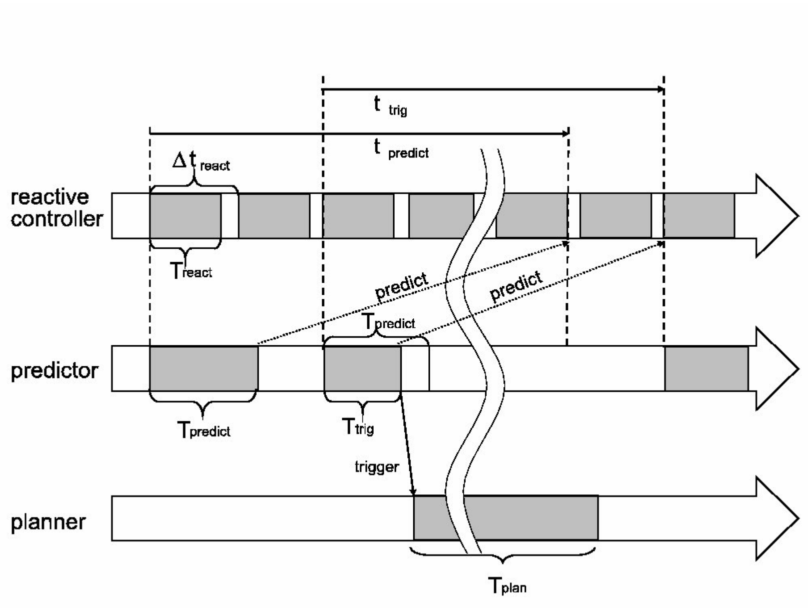


Fig. 2. The time chart of the system. T_{react} is the computation time for the reactive controller which is within Δt_{react}. $t_{predict}$ is the future prediction depth. The predictor predicts the robot's future state until the time $t_{predict}$, but it can predict until t_{trig} when the planner is triggered. T_{trig} is the necessary computation time for t_{trig}.

D. "Start planning" criteria

The predictor triggers the planner if μ_{react} drops below the threshold M_{react}. The available resources are not taken into account in our implementation.

E. Planner

The planner generates one attractor point for each trajectory randomly. Then, the planner simulates with the robot model whether the trajectory with the generated attractor point reaches the given target or not. If the planner finds the trajectory which reaches the target, the planner shifts the attractor point in order to minimize the overall path length.

Since the attractor point is already the intermediate target, the intermediate target generator is not used here.

F. "Use plan" Criteria

If the generated trajectory reaches the actual target, then it is always used.

V. Simulation and Experimental Results

Experiments have been carried out on the humanoid robot ASIMO. It has 26 DOFs of which we control 14 DOFs in our experiments: 4 DOFs per arm and 6 DOFs to describe the virtual link between heel and upper body. ASIMO and its coordinate systems are illustrated in Fig. 3. The height of the robot is about 120cm and its weight is about 50kg. The sampling time of the control loop Δt_{robot} is 5msec. The reactive controller works on ASIMO's embedded computer and the other computations are executed on a remote computer with Intel Xeon 3.06GHz and 4GB memory.

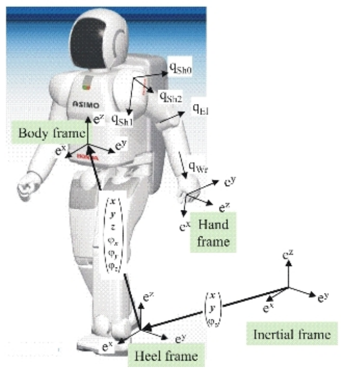


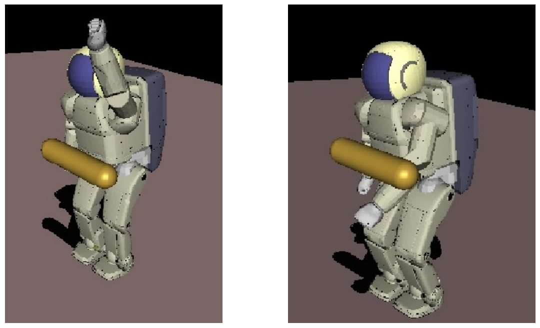
Fig. 3. ASIMO and its coordinate systems

A. Parameters

The sampling time for the internal reactive model Δt_{react} is 5msec, as same as Δt_{robot}. For the internal predictor model $\Delta t_{predict}$ is 20msec. The iteration cycle $N_{predict}$ is 20 times, i.e., the predictor predicts up to 400msec into the future. The coefficient α is $1/|\boldsymbol{F}_{virtual}|$ and the threshold M_{react} is -0.98. The threshold for the virtual forces d_a is $4cm$ and if $\boldsymbol{F}_{virtual}$ is $\mathbf{0}$, the planner is not triggered. The computation time for the prediction $T_{predict}$ dynamically changes depending on the environment but it is at most 30msec at $N_{predict} = 20$.

B. Simulation

In simulation the robot avoids an obstacle in front of it while standing. The robot moves from the initial state in Fig. 4a to the target state in Fig. 4b.



(a) Initial states (b) Target states

Fig. 4. Initial and target states of the simulations. The position of the obstacle varies depending on the simulation in order to highlight the effects of the system.

1) Latency compensation: We test the system both with the latency compensation and without the latency compensation. The result is described in Fig. 5. Without the latency compensation an attractor point is generated as an optimal 3D position when the planner starts. The planner calculates an attractor point that is optimal with respect to the given start position. If there is no latency compensation, at the time the planner is finished, this attractor point is not optimal anymore with respect to the current position. The latency compensation however allows to start the planner approximately with the starting position which will be current when the planner is finished.

2) Adaptive planning: We also tested the system with different prediction iteration cycles N_{trig}.

The result is depicted in Fig. 6. If N_{trig} is 20, the resulting trajectory is more optimal than N_{trig} is 10. If N_{trig} is 0, the system works as same as the reactive controller and the robot gets stuck in a local minimum. As N_{trig} becomes small, the resulting trajectory becomes less optimal.

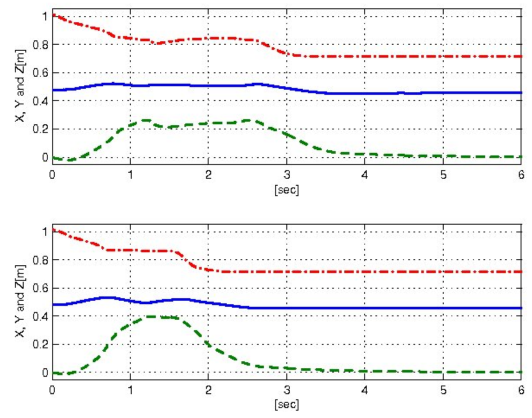
C. Real time Experiment on Robot

We tested the system on ASIMO with the typical situation that the robot is stuck in a dead lock that it can not escape from with the reactive controller alone. If the fitness becomes smaller, the planner generates attractor points for each arm. Then both arms finally reach the targets. The resulting pictures are shown in Fig. 7.

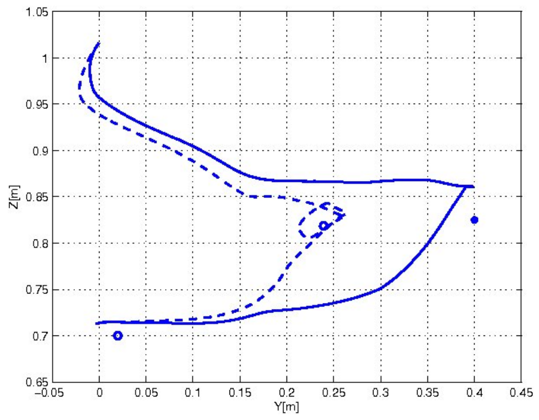
VI. Discussion and Conclusion

We proposed a new scheme for integrating reactive control and planning. The scheme compensates the latency resulting from the high computational costs for planning.

Our scheme is able to handle complex environments by integrating the reactive controller and the planner. If a difficult situation is detected by the predictor, the planner generates an appropriate trajectory beforehand. The robot can track the planned trajectory without reaching a difficult situation. The scheme avoids the shortcomings of pure reactive control and also handles dynamic environments which are difficult for planning, in particular on humanoid robots. Depending on how dynamic the environment is, the system will adjust the



(a) The resulting trajectory of the left hand without (top) and with (bottom) the latency compensation. The solid line, the dashed line and the dashed-dotted line are X, Y and Z with respect to the heel frame coordinate system, respectively. Without the latency compensation, it takes longer to reach the target and the movement is less smooth than with the latency compensation.



(b) The resulting trajectory of the left hand without (dashed line) and with (solid line) the latency compensation on the Y-Z plane with respect to the heel flame coordinate system which is the robot's coronal plane. The arm moves from the top to the bottom of the graph. Without the compensation, the planner generates an attractor point (open circle) i.e. closer to the target than the point with the compensation (filled circle). But it does not reach the target and then the planner generates a second attractor point which is closer to the target. With the compensation, the trajectory is smoother with only one attractor point. Both trajectories are not as smooth as the optimized trajectory generated by a planner only. Because the reactive controller is active even when moving to a planned attractor point, it avoids obstacles and other limitations reactively.

Fig. 5. Latency compensation

planning time so that the overall resulting motion is more reactive and less optimized globally.

Our scheme is different from other common layered architectures in which upper layers dominate lower layers. Most of these architectures always execute the planning (upper layer) first even if the reactive controller (lower layer) is able to handle the situation without planning, and then the reactive controller modifies or superposes the planned motions. On the other hand, our scheme uses planning only if the reactive motion simulated in the predictor becomes

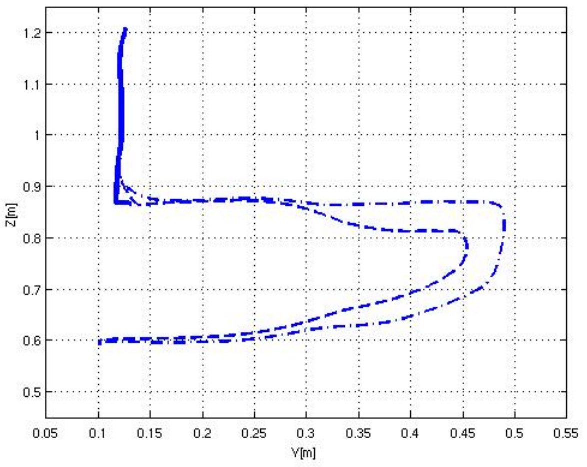
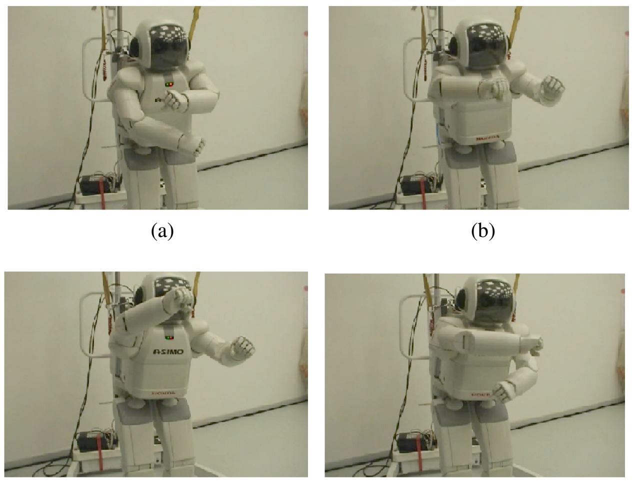


Fig. 6. The resulting trajectory of the left hand on the Y-Z plane with respect to the heel flame coordinate system. The solid line, the dotted-dashed line and the dashed line represent $N_{trig} = 0$, $N_{trig} = 10$ and $N_{trig} = 20$, respectively. If N_{trig} is 0, the system works without prediction, the robot cannot avoid the obstacle which is in front of it. If N_{trig} is 10, the robot is able to avoid the obstacle but the trajectory with $N_{trig} = 20$, is more optimized.



(a) (b) (c) (d)

Fig. 7. Fig. 7a to Fig. 7d show the sequence of the movement: In the initial state (Fig. 7a) the left arm is above the right arm. In the target state (Fig. 7d) the right arm is above the left arm.

insufficient because environments may change during planning. The planner and the reactive controller in the predictor are tightly coupled in the same loop.

Ranganathan's architecture [8] also integrates planning and reactive navigation in a similar manner to our scheme, however, our scheme includes the predictor which simulates faster and enables the robot to avoid difficult situations.

We tested our scheme on the simulator and realized it on our many-DOF humanoid robot.

We are now developing the system for more complex and dynamic environments with a longer prediction depth.

VII. Acknowledgment

The authors would like to thank the members of Honda's ASIMO and robot research teams in Japan for their kind support.

References

[1] J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Dynamically-stable motion planning for humanoid robots," in *Autonomous Robots*, vol. 12, no. 12, 2002, pp. 105–118.

[2] E. Yoshida, "Humanoid motion planning using multi-level dof exploitation based on randomized method," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems 2005*, 2005.

[3] N. Vahrenkamp, C. Scheuer, T. Asfour, J. Kuffner, and R. Dillmann, "Adaptive motion planning for humanoid robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems 2008*, 2008.

[4] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Efficient motion planning for humanoid robots using lazy collision checking and enlarge robot models," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems 2007*, 2007.

[5] R. C. Arkin, *Behavior-Based Robotics*. The MIT Press, 1998.

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

[7] E. Gat, "On three-layer architecture," in *Artificial intelligence and mobile roots: case studies of successful robot system*. AAAI Press, 1998, pp. 195–210.

[8] A. Ranganathan and S. Koenig, "A reactive robot architecture with planning on demand," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems 2003*, 2003.

[9] S. Koenig, "Real-time heuristic search," in *Artificial Intelligence*, vol. 42, 1990.

[10] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings IEEE International Conference on Robotics and Automation*, 1994.

[11] R. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," in *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, 1968.

[12] M. Gienger, B. Bolder, M. Dunn, H. Sugiura, H. Janssen, and C. Goerick, "Predictive behavior generation - a sensor-based walking and reaching architecture for humanoid robots," in *Autonome Mobile Systeme 2007*, K. Berns and T. Luksch, Eds. Springer Verlag, 2007.

[13] H. Sugiura, M. Gienger, H. Janssen, and C. Goerick, "Real-time collision avoidance with whole body motion control for humanoid roots," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2007.

[14] M. Toussaint, M. Gienger, and C. Goerick, "Optimization of sequential attractor-based movement for comapct behavior generation," in *Autonome Mobile Systeme 2007*, 2007.

[15] M. Gienger, H. Janssen, and C. Goerick, "Task-oriented whole body motion for humanoid robots," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2005.