A NEW METHODOLOGY FOR DEALING WITH UNCERTAINTY IN ROBOTIC TASKS

Şule Yıldırım and Turhan Tunalı
yildirim, tunali@ube.ege.edu.tr

International Computer Institute, Ege University
Bornova, İzmir, 35100, TURKEY.

ABSTRACT

A new method that evaluates the execution of robotic operations at the task level is proposed. Traditional robot controllers use variety of feedback loops at motion level to solve dynamical control and motion level planning problems asssuming that the original task plan is never modified. However, a drastical change in the operation environment of the robot might as well affect the sequence of tasks to be executed. In such a case, modifying motion plan may not be sufficient to adapt to the unexpected change in the environment. In this study, a planning environment that allows the original task plan to be modified or entirely changed according to the changing environmental conditions is proposed. A heuristic approach is taken to decide on whether the original plan should be modified or an entirely new plan should be generated. A simple case on pick-and-place sequencing of blocks-world is studied to demonstrate the idea.

Keywords: Robotics, task planning, feedback, artificial intelligence, replanning, learning.

1. INTRODUCTION

The ultimate goal in robotics research is to develop robotic systems that are capable of planning and executing tasks that are specified at highest possible level of abstraction. For this purpose, a multi-level approach to robotic software architectures is well recognized and used [1-4]. In this approach, the complicated structure is divided into various layers: At the top most layer, a task planner produces primitive task sequences that lead to the completion of the ultimate goal. In the middle, the trajectory or motion planner generates the necessary trajectories for the movement of the robot coordinates to perform a preplanned primitive task. At the lowest layer, the trajectory controller of the robot generates the necessary commands to make the robot track the precomputed trajectory in Cartesian coordinates. Various feedback information including the vision, position, velocity, force and torque sensors are used at the bottom two layers to let the respective controllers execute robust control policies [2]. Figure 1 shows a typical robot system.

Although the feedback controls of the bottom two layers produce dependable systems to a certain extent, the task planning level is usually developed without such feedback from the world model. The lack of such feedback might as well lead to the completion of the execution of the original task plan that might have already become obsolete after an unpredictable change in the world model. In such a case, it becomes inevitable to modify the original task plan to compensate for the unpredictable change in the world model. A simple example of such a situation could be given from pick and place sequences operating in blocks-world. What happens if the robot drops an object while

carrying it to complete a task? In such a case, new motion trajectory is only helpful if the motion planner receives a modified task plan from the upper layer.

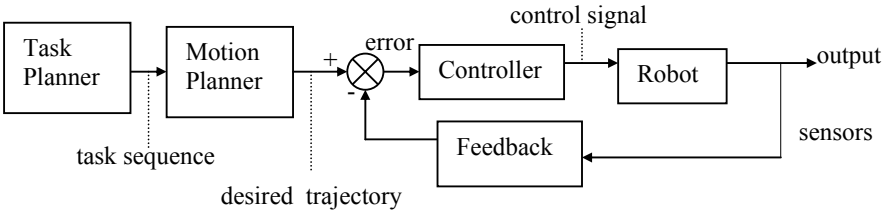

Figure 1: A typical robot system.

Developing task planners for robotic tasks is a widely studied area in the robotics literature. While the early planners lacked the robustness necessary to make them implementable on robots, the recently developed planners attempt to solve many of the practical problems. Some of the work such as [5,6] integrates motion planner with the controller so that it becomes an investigation/decision component based on the sensory measurement. Another group of work addresses the problem of planning under changing world conditions [7-9]. In [10-12], a completely operational system for mobile robot task planning is developed. The interleaving technique used in [12] can be considered as the modification of the original plan as new tasks are requested from the robot.

In this paper, a new architecture for a robotic system that allows replanning under changing world conditions is proposed. A case study from blocks-world is given to demonstrate the idea. The essential tool is the feedback information supplied to the task planner as shown in Figure 2. The planner continuously monitors the world model to see if there is an unpredicted change. If not, it continues executing the original plan. If there is a change, then the planner stops executing the original task sequence and starts a modification algorithm. This is the replanning phase that basically consists of making a decision on two alternatives:
Alternative 1: The first alternative is to generate an intermediary task sequence to compensate for the unpredicted change in the world model. That is, a task sequence is generated to put the blocks-world to the state right before the unpredicted change. The planner then proceeds with the execution of the original plan from the point it had stopped while executing.
Alternative 2: The second alternative is to accept the modified world model as a new initial state and generate an entirely new task plan that leads to the goal state.

A composite cost function is used to evaluate two alternatives. Depending on the result, a decision is made. The system is not expected to lead to the right decision in the beginning however, because of the learning nature of the algorithm, as the unexpected change cases build-up, the system makes use of the accumulated knowledge and makes a decision that optimizes the case according to the selected cost function.

The paper is organized as follows: In Section 2, the new architecture is introduced and the algorithmic details are given to demonstrate the idea in a blocks-world pick-and-place application. In Section 3, the details of the case are introduced, simulation results on the blocks-world pick-and-place case are given and performance issues are discussed. Finally, in Section 4, concluding remarks are made.

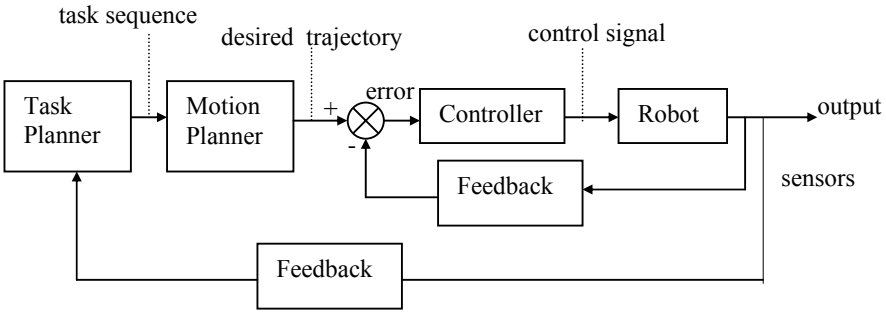

Figure 2: A robot system where the feedback information is supplied to the task planner.

2. THE PROPOSED ARCHITECTURE

Since feedback controls of the motion planning and control layers are insufficient for producing dependable systems, a task planning level supplied with feedback is proposed. A task planning level receiving feedback becomes the decision component in the robotics software architecture and in case of an unexpected event, it obtains the necessary information to decide whether a task plan will lead to completion or not. If task planning level decides that current task plan will not lead to completion, then it can either modify or completely change the current task plan.

For the purpose of demonstrating such a system, a task planner for a pick-and-place case in blocks-world has been written. The task planner receives simulated world model states as feedback instead of real world model states from the vision system since studies on vision system are still being carried out (obtaining the images of the world model by the vision system and interpreting the images).

The task planner begins execution by generating a task plan by a task plan generation module (Figure 3). Later task planner monitors the world model continuously by a monitoring module to detect any possible unexpected changes in the world model while the task plan is in execution. If the task planner detects an unexpected change in the world model such as an unexpected change in location of objects at any time during the execution of the task plan, it stops the execution of the current task plan and calls the replanning module to cope up with the unexpected situation. The replanning module has two alternatives mentioned in Section 1 to choose from.

At the beginning, the replanner might not give the right decisions since the threshold has been assigned a value randomly. As time passes, more and more unexpected events occur and the replanning module tries to obtain a threshold value which leads to better decisions. The replanner adjusts the value of the threshold by making use of the knowledge of the previous unexpected events. This knowledge is kept in a cost table in form of costs (Figure 4). A cost table is a two dimensional array with dimensions P x Q. P is the maximum value of a distance metric and Q is the maximum number of unexpected events allowed for a distance metric. The cost table is organized as columns which contain costs for different distance metric values. Threshold might be assigned a value such that $0 < threshold\_value < P$.

When an unexpected event occurs, a distance metric is calculated for the unexpected event. This distance metric is compared with the currently assigned value of threshold. If the calculated distance metric is smaller than the value of the threshold, then

alternative 1 is executed, else alternative 2 is executed. In either case, the cost table is updated.

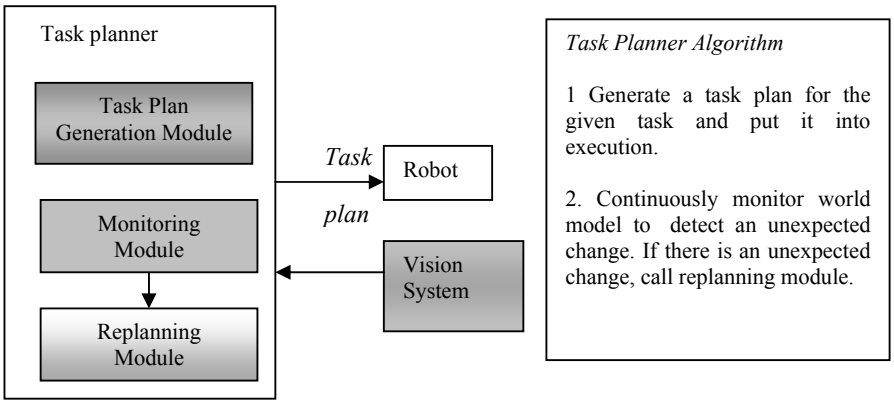

Figure 3: General framework for the task planner with vision feedback.

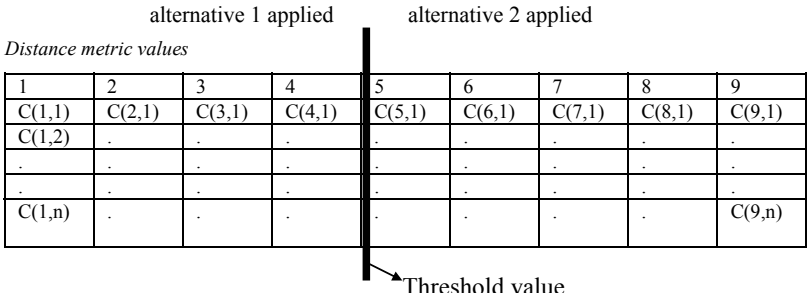

Figure 4: An example cost table

The replanner doesn't adjust the value of the threshold until the cost table is entirely filled in. When it is entirely filled in, the averages of each column in the cost table are obtained. An abnormal increase or a decrease between the averages of the threshold column and the column preceding it means that the threshold value needs an improvement. An improvement is made by increasing or decreasing the value of the threshold by an amount. A heuristic is used to decide on an increase or a decrease. The heuristic is that the threshold will be increased (shift right in the cost table) if there is an abnormal increase and will be decreased (shift left in the cost table) if there is an abnormal decrease (Figure 5 and Figure 6).

There is a value range where the threshold is stabilized and adjust operations are continued until the threshold value is in that range. If the initial value of the threshold is small, as learning proceeds, the threshold value increases to this stability region and stays there. Detecting a decrease means threshold value should be changed back to the previous value where there is an increase. The increases get smaller for increasing values of threshold so the increase detected before a decrease will probably be the smallest increase of all. The cost table is emptied after each adjust operation so that it can be used for accumulating costs of forthcoming unexpected happenings of the following adjust attemps.

Replanner Algorithm

1. Calculate a distance metric for the unexpected happening.
2. If the distance metric is smaller than the threshold value then execute alternative 1 else execute alternative 2.
3. Update the cost table.
4. If cost table is full, adjust the value of threshold and clear the table.
5. Go to step 1.

Figure 5: Replanner Algorithm

Adjust_Threshold Algorithm

1. Calculate the average cost for each column of the cost table and keep the averages in an average-cost array.
2. Check if an abnormal increase or decrease between the averages of threshold column and the column preceding it is detected.
3. If yes
   If this is an increase, shift threshold value to the right by *stepsize*.
   If this is a decrease, shift threshold value to the left by *stepsize*.
   Empty the cost table to accumulate new cost values with the new threshold value.
4. If no
   If an increase was detected previously shift threshold value to the right by *stepsize*.
   If a decrease was detected previously do nothing. This is the stabilized threshold value.

Figure 6: Adjust_Threshold Algorithm

3. A CASE STUDY

A case study to the unexpected changes in the world model is the mixed pieces problem. The problem is stated as follows: There is a N X N board on which there are $N^2 / 2$ pieces placed on different cells of the board. There are m different pieces and there are n_j items of each different piece where j: $1 \leq j \leq m$ and $\sum n_j = N^2 / 2$. The pieces are transferred from their positions in initial state to their final positions in final state by a robot arm. While this transfer occurs, the pieces on the board are mixed up (both the pieces in initial state and the pieces which have been carried to final positions from initial state might be mixed up). Our problem now is to find some optimal path to bring the mixed state of the board to the final state.

A composite cost function for a path is defined as follows:

$C_p = c_1 T_p + c_2 E_p$ where
c_1, c_2 constants,
T_P : The run-time to generate a new task plan for that path,
E_p : The energy to be spent by the robot arm to execute the task plans on that path.

Then, the costs of Alternative 1 and Alternative 2 are:
$C_1 = c_1 T_1 + c_2 E_1$
$C_2 = c_1 T_2 + c_2 E_2$ where
T_1, T_2 are the run times of generating a new task plan for Alternative 1 and Alternative 2 respectively.
E_1, E_2 are the energes to be spent by the robot arm while executing the task plans generated for Alternative 1 and Alternative 2 respectively.

A board with dimensions 16 X 16 is used. There are two group of pieces (white and black) and there are 6 different pieces in each group. Table 1 shows the number of pieces for each type (Letters represent each item in each group).

The cost table is a 80 X 5 table. The distance metric used for this problem is the number of misplaced pieces on board This number indicates the amount of change between the state right before the change and right after the change. If this number is low enough then it is better to use Alternative 1. The distance metric is assumed to be in the range $0 < dm < 80$. Cost values of 5 unexpected happenings are accumulated in a distance metric column. 400 unexpected happenings should have occurred to fill a cost table completely.

White group		**Black group**	
Item name	*# of items*	*Item name*	*# of items*
V	4	V	4
S	4	S	4
A	8	A	8
K	16	K	16
F	16	F	16
P	16	P	16

Table 1: The number of pieces for each type.

As indicated in *Adjust_Threshold Algorithm,* when an unexpected event occurs algorithm will check if the table is entirely full. If it is, then it will look for an abnormal increase or decrease between the average cost value of threshold column and the column preceding it. An abnormal increase is assumed to have a value more than 1 and an abnormal decrease is assumed to have a value less than –1. For each increase, threshold will be incremented by 10 and for each decrease, threshold will be decremented by 10. The threshold is assigned value 5 at the beginning of the program. Table 2 shows the threshold values and respective changes at these thresholds.

Threshold	Change
5	+3.16
15	+7.68
25	+7.38
35	+5.37
45	-3.08

Table 2: Abrupt change of costs at 5 thresholds.

Figure 9-13 are the graphs of cost table column averages at different thresholds. The values below the threshold are Alternative 1 values while the values above threshold are Alternative 2 values. It is expected that the area under the graph of a cost table column averages will be less than the area of a graph which composes of only Alternative 2 values. This is based on the idea that Alternative 1 values need less effort at low distance metric values than Alternative 2 values. That is, it is sometimes better to modify a plan rather than throwing it away and forming a completely new plan. This expectation comes out to be true when the areas under graphs in Figure 13 and Figure 14 are calculated. Values of 1281 and 1336 are obtained for Figure 13 and Figure 14 respectively. Executing alternative 1 for low values of distance metric and alternative 2

for high values of distance metric gives a less area when compared with executing alternative 2 for all distance metrics. The difference between the two becomes higher at stabilized value of the threshold. The area under the graph of cost table column averages in Figure 12 at stabilized threshold value of 35 is 1522 while this value is 1602 if the graph is the graph of direct path for every distance metric. The difference between the two values is approximately 80 which is a considerable amount. It should be noted that the threshold value is stabilized after 5 adjustments.

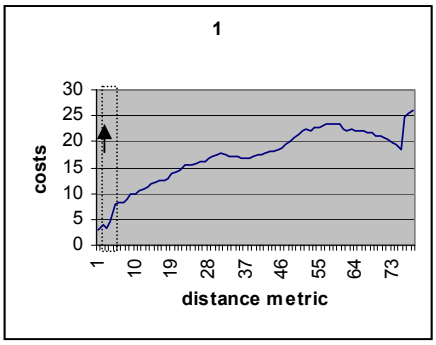

Figure 9: Cost table column averages for threshold 5.

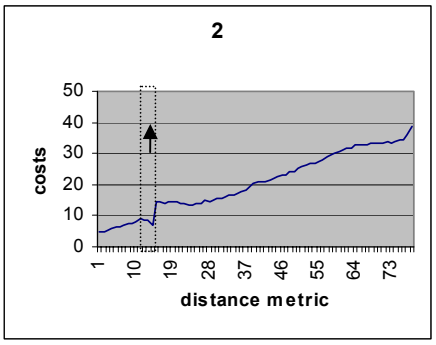

Figure 10: Cost table column averages for threshold 15.

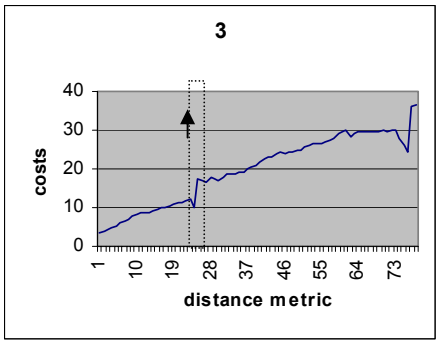

Figure 11: Cost table column averages for threshold 25.

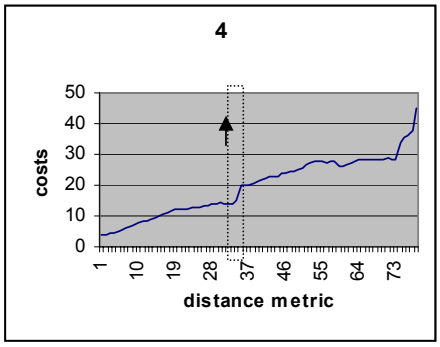

Figure 12: Cost table column averages for threshold 35.

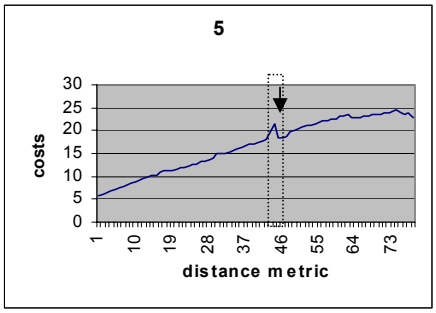

Figure 13: Cost table column averages for threshold 45.

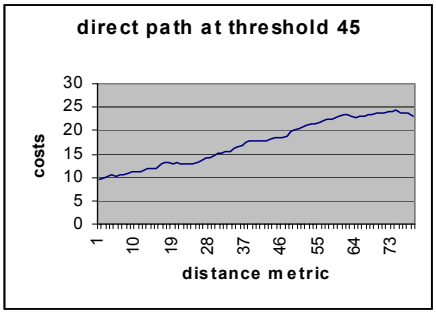

Figure 14: Alternative 2 column averages for all distance metrics in case of Figure 13.

4. CONCLUSION

In this study, a new method is proposed to deal with uncertainty in robotic environments. The method is based on the supply of feedback information about the world model to the task level of the robot software. The task planner's evaluation of any unexpected change leads to generation of new plans that yield optimal operation of the robot system.

Current work is progressing in two different directions. The first direction is the development of a general task planning environment that is suitable for the replanning paradigm. This involves variety of topics in world modelling and task representation in addition to the architecture introduced in this paper. The second direction is the integration of the vision system with the planner. Presently, the vision and robot system are both operational in our laboratory and their integration at task level is under study.

REFERENCES

[1] K. S. Fu, R. C. Gonzales, C. S. G. Lee, " Control, Sensing, Vision, and Intelligence", Mc-Graw-Hill, 1987.

[2] M. Brady, J. M. Hollerbach, T. L. Johnson, T. Lozano-Perez, M. T. Mason, "Robot Motion: Planning and Control ", MIT Press, 1986.

[3] T. Tunalı and G. S. Güler, "A Hierarchical Planner for Robot Tasks and Motion Control", Proc. IEEE workshop on Intelligent Motion Control, Istanbul, TURKEY, August 1990.

[4] G. S. Güler and T. Tunalı, "An Object-Oriented Approach to the Relational Representation of the Robotic Tasks and Workcells", Proc. IEEE workshop on Intelligent Motion Control, Istanbul, TURKEY, August 1990.

[5] N. Xi, T. Tarn, A. K. Bejczy, "An Intelligent Planning and Control for Multirobot Coordination: An Event-Based Approach", IEEE Transactions on Robotics and Automation, June 1996, vol 12, no. 3 .

[6] C. Tung and A. C. Kak, "Integrating Sensing, Task Planning, and Execution for Robotic Assembly", IEEE Transactions on Robotics and Automation, vol. 12, no. 2, pp. 187-201, April 1996.

[7] E. M. Atkins, E. H. Durfee and K. G. Shin, "Detecting and reacting to unplanned-for world states", AAAI Fall Symposium "Plan Execution: Problems and Issues", pp. 1-7, 1996.

[8] S. W. Bennet, G. F. DeJong, "Real World Robotics: Learning to plan for robust execution. Machine Learning", Machine Learning, vol 23, pp. 121-161, 1996.

[9] D. Borrajo, M. Veloso, "Incremental learning of control knowledge for improvement of planning efficiency and plan quality", AAAI Fall Symposium "Planning and Learning: On to Real Applications", pp. 5-9, 1994.

[10] Y. Gil, "Acquiring domain knowledge for planning by experimentation", Ph.D Thesis, School of Computer Science, Carneige Mellon University, 1992.

[11] R. Goodwin, "Meta-Level Control for Decision-Theoretic Planners", Ph.D Thesis, School of Computer Science, Carneige Mellon University, 1996.

[12] K. Z. Haigh, "Situation-Dependent Learning for Interleaved Planning and Robot Execution", Ph.D Thesis, School of Computer Science, Carneige Mellon University, 1998.