Proc. 7th Int. Conf. on Control, Automation, Robotics and Vision,
pp. 246-251, Singapore, 2002.

Parallel Scheduling of Planning and Action for Realizing an Efficient and Reactive Robotic System

Jun Miura and Yoshiaki Shirai

Department of Computer-Controlled Mechanical Systems, Osaka University
Suita, Osaka 565-0871, Japan
Email: jun@mech.eng.osaka-u.ac.jp

Abstract

This paper describes a method of parallel scheduling of planning and action to realize an efficient and reactive robotic system in dynamic environments. The method uses a partial planning result, which comes from an iterative refinement planning, to determine feasible actions to be executed in parallel with the planning process. The method is applied to a multiple-camera multiple-person tracking problem. Simulation results show the effectiveness of the method.

1 Introduction

Intelligent systems are usually designed as a set of multiple functional units, each of which is responsible for a part of necessary tasks. An example is the conventional sense-plan-act model, in which a system is composed of the sensing, the planning, and the execution part. Usually each part is activated sequentially in this order being given the result of its preceding part. If resource and time constraints are not severe, such a sequential activation may be sufficient. Otherwise, we need some strategy to increase the efficiency and the reactivity of the system.

Physical agents like robots often have multiple processors which can run in parallel with each other. For example, a vision-guided mobile robot may have three processors for environment recognition, path planning, and motion control. The first step to increase the efficiency of such a multi-processor system is to distribute the functional units over processors and to make them run in parallel. This paper thus focuses on how to coordinate such multiple units under a central control scheme.

Real-time search [8] or interleaving [11] are suitable bases for a simple parallelization of planning and action; i.e., executing the current step while planning the next step (or subsequent steps). Goodwin [5] also dealt with a similar parallelization but he explicitly compared two options, pure planning and planning with execution of the current best action. This simple parallelization, however, may have two drawbacks when the planning cost is high owing to, for example, the necessity of considering many contingencies. One drawback is that once the execution of a planned step has been completed, the execution part has to idle about waiting for the next planning result. The other drawback is that since the cycle of execution is determined by the planning time, the reactivity to environment changes may not be enough in a highly dynamic environment.

Many layered architectures have been proposed for controlling autonomous agents (e.g., [2]). These works mainly discuss how to integrate deliberative and reactive activities in dynamic and uncertain environments. The parallelization realized by them is, however, similar to those of the above; i.e., executing the current step while planning the next step.

To seek more effective parallel execution, let us consider a simple example. Suppose you are driving to the east part of a town from its center and your colleague is still searching for the path to the destination on the map. In this case, you can start moving east immediately after you find that the destination is on the east of the town, because this motion impose little extra cost on you; you will go east anyway. What this example suggests is that if enough information is obtained to determine an appropriate action from now to some future time point, the action can start *before* the completion of the effort of seeking more information.

Such a parallelization is one of the keys to solve the problem of reactivity vs. efficiency. Increasing the reactivity by simplifying the reasoning may result in an inefficient behavior due to local reasoning; on the other hand, increasing the efficiency by adopting a more elaborated reasoning may decrease the reactivity. By executing multiple activities in parallel, an agent (or a system composed of multiple agents) may be able to behave efficiently without losing much reactivity. The aim of this paper is to propose a parallel scheduling method to realize such a behavior. We use a multiple-camera multiple-person tracking problem as an example problem in dynamic environments.

We have proposed a method of parallelizing planning and action for a mobile robot [10]. The method uses a criterion to determine a set feasible actions which can be executed in parallel with the current planning process. We have also shown that the method improved the overall performance. However, the planning problem treated was the one in a static environment. In addition, we have not examined how selection of the criterion affected the performance. We therefore apply the method to a planning in a dynamic environment, the coordination of multiple cameras for tracking multiple persons.

Plan execution management has recently been an im-

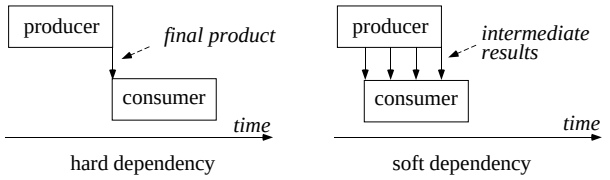

Figure 1: Difference between hard and soft dependencies.

portant research theme (e.g., [1][12]). The main issue in these works is how to manage the planning process adaptively to situation changes. This paper is, in contrast, focused on realizing efficiency and reactivity simultaneously for a pair of specific planning and action processes.

2 Classification of Dependencies

2.1 Producer/Consumer view of activities

We view a pair of activities with a dependency relation as a producer and a consumer. The producer generates a product and the consumer uses it. For example, a sensing activity generates an environment map and a planning activity uses the map for path generation; the generated path is then used by a motion control activity. This paper deals with the case where a planning process is the producer and a reactive action-selection process is the consumer.

2.2 Hard and soft dependencies

If a producer is a *one-shot* type activity that outputs a final product after a certain period of processing, a consumer gets no information until the final product is generated. We call this relation between the producer and the consumer a *hard dependency*. If two activities are hard-dependent, they cannot basically be parallelized[1] , as in the case of ordering constraints in scheduling [16].

If the producer can output some intermediate results, such as a set of product candidates, in the middle of processing, and if the consumer can utilize such results to determine its action, the consumer does not have to wait for the completion of the producer's processing. We call this relation a *soft dependency*. Soft dependency is a key concept of the proposed parallel scheduling method. Fig. 1 compares a hard and a soft dependency.

3 Parallel Scheduling for Soft Dependency

3.1 Illustrative example

This paper deals with the following multiple-camera multiple-person (MCMP) tracking problem (see Fig. 2):

MCMP problem[2] *:* There are M persons arbitrarily walking in a room. Each person may sometimes go out of the room through one of doors and come back later. There are N cameras fixed on the ceiling of the room. Each

[1] The ith execution of the consumer and the i+1th execution of the producer are, of course, parallelized as in the case of parallelization using real-time search or interleaving explained above.

[2] Note that this is *not* a multi-agent planning problem, in which distributed coordination of multiple cameras is an issue (e.g., [9]). We are interested in a central control of multi-processor systems.

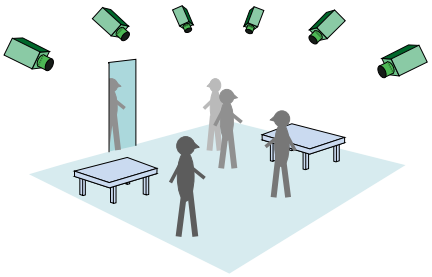

Figure 2: A multiple-camera multiple-person tracking problem.

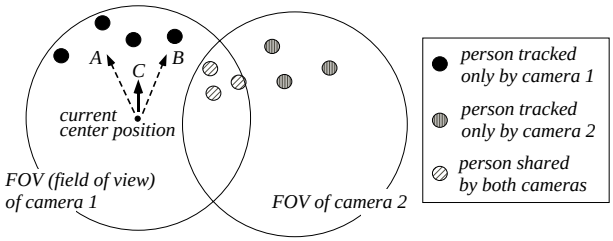

Figure 3: Possible movements of camera 1.

camera can change the viewing direction within a predetermined range. A single planning process assigns currently tracked persons to cameras, and each camera tracks the assigned persons independently of the other cameras. The goal of the whole system is to track as many persons as possible during a certain period of time. Each camera is assumed to be able to recognize any person as long as the person is inside the field of view of the camera.

Fig. 3 shows an example case where a camera can start moving before the completion of assignment determination, using a soft dependency relation. In the figure, *camera 1* can track seven persons, three of which *camera 2* can also track. Only *camera 1* can track the other four persons. In this situation, the planning process will determine the assignment of the three shared persons. Although *camera 1* cannot know the exact assignment at this time, it knows that the number of assigned persons will range from four to seven. If only the left four persons are assigned, movement *A* seems best; if all seven are assigned, movement *B* seems best. So a good movement will be between them (e.g., movement *C*). The meaning of "good" here is that *camera 1* will have little loss for any possible assignments; *camera 1* will shift the focus of attention upward anyway. If we can select such a movement and execute it in parallel with the current planning process, the reactivity in tracking is expected to be kept high without losing much efficiency.

3.2 Iterative refinement processing

To realize a soft dependency, the producer should be able to output intermediate results repeatedly. This entails an iterative processing scheme which gradually reduces the set of candidates for the final product. There are a variety of iterative refinement processing, such as planning as refinement search [7], sensor fusion for re-

ducing uncertainty [3] or ambiguity [6], and many coarse-to-fine processing methods. Anytime algorithms [4] are also suitable for iterative refinement. However, the iterative processing required here does not have to necessarily be *anytime*; a producer only have to give its consumer some useful information for limiting the set of possible consumer's actions.

An iterative processing terminates when the final product is generated. Considering the limitation of computational resources, especially in dynamic environments, some additional termination conditions may be necessary such as:

- A predetermined time for one planning cycle has elapsed. This condition is to keep the minimum level of reactivity.
- Further processing has turned out to be almost useless, for example, by referring to the estimate of future improvement [13].

3.3 Consistency criterion

A consumer uses intermediate results from its producer to determine the next action. For this purpose, we introduce the notion of *consistent* action. An action of a consumer is considered to be consistent with the current process of a producer if the consumer's action is effective for *any* of possible products. For each soft dependency, we define a *consistency criterion* which checks if an action is consistent. The consumer selects and execute an action among consistent ones. Usually a consistency criterion is problem-specific and should be defined problem to problem.

What consistency criterion to use will affects the performance of the whole system. A loose consistency criterion, which allows many actions of the consumer to be consistent, enables a consumer to start early but may degrade the overall performance owing to selection of not-so-good actions. A tight consistency criterion, on the other hand, may force the consumer to start late but the overall performance may be good owing to well-screened actions. This tradeoff will be examined experimentally later.

3.4 Summary of planning and parallel scheduling

The pseudo code in Fig. 4 summarizes the planning and scheduling algorithms of the producer and the consumer in soft dependency relations. The algorithms for one cycle (i.e., generation of one product) are shown. Once the producer sends a product to the consumer, it may wait for the completion of the execution of the consumer.

4 Implementation and Experimental Evaluation

This section describes an implementation and the experimental evaluation of the proposed method using the multiple-camera multiple-person (MCMP) problem.

```
Producer(){
  candidates = generate_initial_candidates();
  while (true){
    send_info_to_consumer(candidates);
    candidates = refine(candidates);
    if (termination_condition(candidates)) break;
  }
  product = select_best_product(candidates);
  send_info_to_consumer(product);
}
Consumer(){
  while (true){
    candidates = receive_info_from_producer();
    if (candidates == final_action) {
      execute_action(final_action); break; }
    candidate_actions = select_consistent_actions(candidates);
    best_action = select_best_action(candidate_actions);
    execute_action(best_action);
}}
```

Figure 4: C-like code of parallel planning and scheduling algorithms for producer and consumer in soft dependency relations.

4.1 Detailed problem settings

We made a simulator for the MCMP problem, as shown in Fig. 5. In addition to the general problem description mentioned above, we use the following detailed settings. The room is a $40[m] \times 40[m]$ square and 16 cameras are placed in a 4x4 array on the ceiling of $10[m]$ high, as shown in Fig. 5. The field of view of each camera is a circle of $5[m]$ radius. Each camera can move the focus of attention (the center of the field of view) within the circle of $5[m]$ radius centered at the home position right below the camera. The maximum speed of the focus of attention is $0.5[m/s]$.

In each simulation, we generate 50 walking persons, who come into the room from one of the four doors at the initial time. The initial, maximum, and minimum velocities are $1.0[m/s]$, $3.0[m/s]$, and $0.1[m/s]$, respectively. At each time step, each person changes its velocity and moving direction randomly according to some predetermined distributions of acceleration and change of moving direction. If a person comes in a predetermined distance from a door, the person will go out with a certain probability and come back later from the same door.

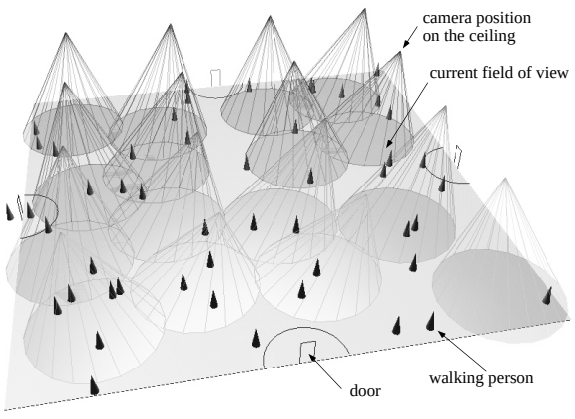

Figure 5: MCMP simulator.

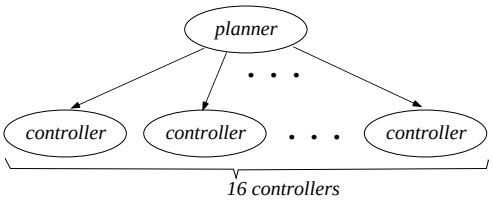

Figure 6: Dependencies between *planner* and *controllers*.

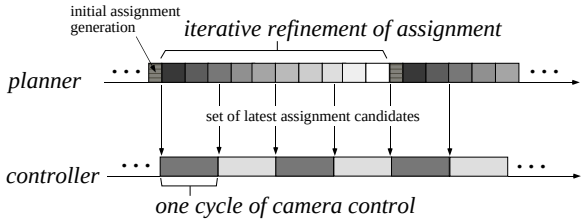

Figure 7: Time chart of *planner* and *controller*.

4.2 Planner activity and controller activity

In this planning problem, the producer is the planning process (called *planner*), which determines the assignment of tracked persons to the cameras, and the consumers are the camera controllers (called *controllers*) (see Fig. 6). *Planner* iteratively refines assignment candidates and sends the latest set of candidates to *controllers* upon request of *controllers*. All *controllers* operate in the same cycle time (set to $1[s]$); every second, each controller determines and executes an action (a movement of the corresponding camera) according to the candidate set. The time chart of *planner* and one *controller* would look like the one shown in Fig. 7.

4.3 Iterative refinement formulation

The person-to-camera assignment problem is formulated as follows. For persons $p_i\,(i = 1 \ldots P)$ and cameras $c_j\,(j = 1 \ldots C)$, let $\mathcal{A}$ be an assignment defined as:

$$\mathcal{A} : p_i \;\rightarrow\; c_j \quad (j = 0 \ldots C),$$

where c_0 means that a person is not being tracked and is not considered in the current assignment calculation. The objective of *planner* is to determine the best assignment.

4.3.1 Initial assignment

Before starting planning, the position of tracked persons in a future (currently, $1[s]$ ahead) is predicted using a linear extrapolation from the current position and velocity estimate. Using the predicted positions, *planner* first generates the initial assignment, in which, to each camera, all *trackable* persons are assigned; a trackable person is the person whom the camera can track in the next control cycle considering the predicted position and the constraints on the field of view and the motion of the camera. Starting from this initial assignment, *planner* iteratively searches for better assignments.

4.3.2 Evaluation of assignment

Assignments are evaluated by considering the following two factors. One is the number of tracked persons. The other is how widely the focuses of attention of the cameras are distributed in the room. The second factor is considered because it is not a good situation where many cameras are focused on a specific area of the room. We measure the degree of distribution, dod, by the variance of the distance from each camera to its nearest neighbor. dod is defined by:

$$\begin{aligned} dod &= E\left[(mindist_i - \overline{mindist})^2\right], \\ mindist_i &= \min_{j \neq i} dist(\boldsymbol{p}_i, \boldsymbol{p}_j) \end{aligned}$$

where $E[\cdot]$ is the expectation, $\boldsymbol{p}_i$ is the focus of attention of the ith camera to be determined for this assignment [3], $dist(\cdot,\cdot)$ is the distance between two positions, $\overline{mindist}$ is the mean of the minimum distances. The smaller dod indicates the better (i.e., wider) distribution. At the initial state, each camera is at the home position (i.e., looking right down) and dod is zero.

Let n_i, dod_i, n_j, and dod_j be the number of tracked persons and the distribution measure of assignments $\mathcal{A}_i$ and $\mathcal{A}_j$, respectively. $\mathcal{A}_i$ is considered to be better than $\mathcal{A}_j$ if:

1. $n_i > n_j$, or
2. $n_i = n_j$ and $dod_i < dod_j$.

4.3.3 Determination of focus of attention

For an assignment, the focus of attention (FOA) of each camera is determined as follows. The determined FOA is used both for calculating dod for evaluation and for generating the final product (the complete assignment and FOAs).

For a camera which does not have trackable persons, the FOA is determined to go back to the home position at the maximum speed.

For a camera which does not share any assigned persons with other cameras, the FOA is determined to be the center position of the region where the maximum number of assigned persons will be tracked in the next time step.

For a group of cameras whose member shares at least one person with one of the other cameras in the group, a temporal assignment is determined first as follows. We list the cameras in the descending order of the number of tracked persons. Then, we fix all assignments to the first camera in the list, remove the assigned persons from the rest cameras' assignments, and remove the first camera from the list. We repeat this until all cameras are processed. In the case of Fig. 8, for example, the following assignment is determined: persons 1-4 to camera A, 5-6 to B, and 7 to C. After the assignment is calculated, the FOA of each camera is determined as in the case of non-sharing cameras.

[3] Calculation of the focuses of attention will be described later.

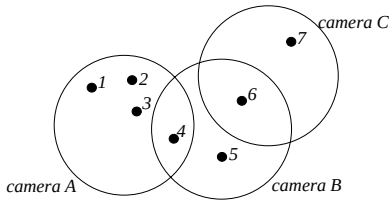

Figure 8: Example situation.

```
Refinement(group_of_cameras){
  shared_persons = get_shared_persons(group_of_cameras);
  for each person in shared_persons {
    sharing_cameras = get_sharing_cameras(person);
    for each camera in sharing_cameras {
      assign_person(person, camera);
      for each other_camera ≠ camera {
        remove_person(person, other_camera);
      }
      generate_new_candidate();
}}}
```

Figure 9: C-like code of assignment refinement step.

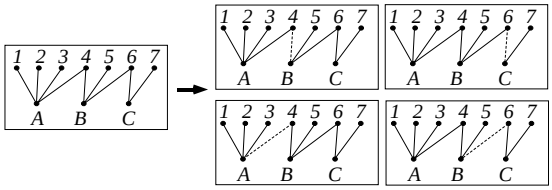

Figure 10: Example of refinement.

4.3.4 Refinement step

Assignment candidates are listed in the descending order of the evaluation. At each refinement step, the current best assignment is selected and refined. A refinement step consists of selecting a person who is shared by multiple cameras one by one, and then generating new assignment candidates for each possible assignment of the person to the cameras. Fig. 9 shows a pseudo code for a refinement step. Several new candidates are usually generated from a candidate.

As an example, let us consider the case where the assignment is represented by Fig. 8. The current assignment is the leftmost one in Fig. 10. In this case, the *shared* persons are 4 and 6. According to the person to select and the camera to be assigned, we will have four new candidates shown on the right in Fig. 10. Dotted lines in the figure indicate the removed assignments.

When a new candidate is generated, the determination of FOAs described above is also performed; by using the evaluation based on the FOAs, the new candidate is inserted into an appropriate place in the ordered list of candidates. To keep the size of the list within a certain level, we remove a candidate if its number of trackable persons is less than that of the best candidate (i.e., the first element of the list). This is a kind of greedy strategy.

4.3.5 Termination condition and final product

We currently use the following two termination conditions:

A sufficiently good assignment is obtained: for the currently best assignment $\mathcal{A}^*$ and the second best one $\mathcal{A}^{**}$, if the numbers of tracked person are the same for them and $dod_{\mathcal{A}^*} \,/\, dod_{\mathcal{A}^{**}}$ is larger than a threshold (currently, 0.99), the iteration terminates.

The iteration has spent a certain amount of time: if the number of generated candidates exceeds a threshold (currently, 60), the iteration terminates.

When the iteration terminates, the assignment (and its FOAs) is sent to *controllers* as the final product.

4.4 Consistency criterion and action selection

A consistent action (movement of FOA) of *controller* is the one which does not entail much loss for any possible final assignments from *planner*, as mentioned before (see Fig. 3). Therefore, we first calculate the union of assigned persons for each camera from all remaining candidates. The set of such unions is regarded as an assignment, which is to be analyzed to select consistent actions.

If a camera does not share any assigned person in the assignment with other cameras, its motion can be determined independently, because the number of assigned persons never increases as the refinement proceeds. For such a camera, consistent actions are the set of movements of FOA to the region where all persons assigned to the camera are trackable, and the one to the center of the region is selected and executed.

If a camera shares persons with other cameras, the consistent actions are determined as follows (see Fig.11). We first calculate two positions. $\boldsymbol{g}_{ex}$ is the center of the convex hull formed by the predicted position of the persons who are exclusively assigned to the camera. $\boldsymbol{g}_{all}$ is the center of the convex hull formed by all persons. We use a heuristic that consistent actions lie in the triangle formed by these two points and the current FOA, $\boldsymbol{c}$, of the camera.

To select the best consistent action, we use two parameters. α controls the weights to the exclusively assigned persons and the other persons. We first calculate a division point $\boldsymbol{g} = \alpha\,\boldsymbol{g}_{ex} + (1-\alpha)\,\boldsymbol{g}_{all}$, and set the next FOA on the line connecting $\boldsymbol{c}$ and $\boldsymbol{g}$. Parameter β controls the speed of FOA movement; the maximum speed is set to that of the camera multiplied by β. Since a larger β makes the FOA movement larger, β represents the "looseness" of the consistency criterion. In the simulation, α is fixed to 0.8. β is changed to examine the effect of the "looseness" of the consistency criterion to the overall performance of tracking.

4.5 Experimental Results

We performed simulation for five data set. One data set is composed of 1000 steps (i.e., $1000[s]$) of movements of 50 person. We use as the measure of the overall performance the averaged ratio of the number of tracked persons to 50.

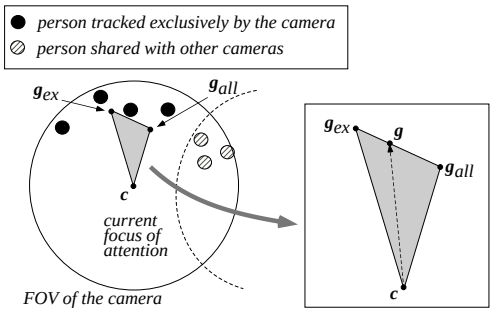

Figure 11: Consistency criterion and action selection.

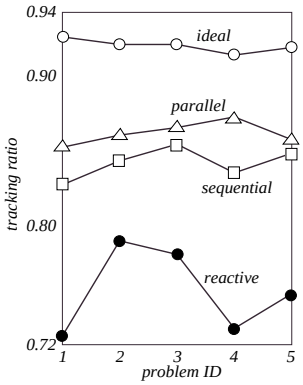

Figure 12: Comparison result.

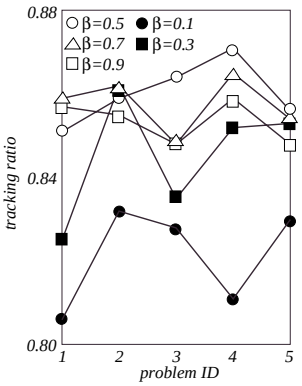

Figure 13: Effect of β.

First, we compared our method (called *parallel*) with the following three methods. *ideal* is a planner whose computational cost is negligible; that is, the maximum number of assignment candidates can always be examined within one cycle time. Its performance is used as a kind of upper bound. *reactive* is a planner which controls each camera independently without any coordination among cameras; using this planner, multiple cameras sometimes tracks the same person. Its performance is used as a kind of lower bound. *sequential* is a planner which does not generate intermediate results and, therefore, *planner* and *controller* are sequentially activated. We set the cost of examining one candidate to $1/20[s]$; the delay when the maximum (60) candidates are examined is $3[s]$. The comparison result in Fig. 12 shows the effectiveness of the proposed method.

Second, we examined the effect of the "looseness" parameter β of the consistency criterion to the overall performance. We set β to one of 0.1 (tightest), 0.3, 0.5, 0.7, and 0.9 (loosest). The comparison result shown in Fig. 13 indicates that too tight or too loose criterion is not appropriate. Although $\beta = 0.5$ seems best in this experiments, selection of best parameter in general is an open question.

5 Conclusions and Discussion

This paper has described a novel method of parallel scheduling of planning and action of a robotic system. The method is based on the notion of *soft dependency* between planning and action, and employs an iterative refinement planning and consistency criteria to select the consistent actions. The method is applied to a multiple-camera multiple-person (MCMP) tracking problem. The results have shown that both the efficiency and the reactivity are simultaneously realized. The effect of selecting consistency criteria to the overall performance was also analyzed. We are now planning to apply the proposed method to a real multiple camera system [15].

This paper is a step towards a general scheme of centralized coordination of multiple activities, in which not only planning and action but also other activities such as sensing are involved. A future work is to apply the method to other coordination problems. Comparison with distributed planning approaches [14] is another future work.

Acknowledgments

This research is supported in part by the Kurata Foundation, Tokyo, Japan.

References

[1] M. Beetz and D. McDermott. Local Planning of Ongoing Activities. In *Proc. the 3rd Int. Conf. on AI Planning Systems*, pp. 19–26, 1996.

[2] R.P. Bonasso, R.J. Firby, E. Gat, D. Kortenkamp, D.P. Miller, and M.G. Slack. Experiences with an Architecture for Intelligent, Reactive Agents. *J. of Experimental and Theoretical Artificial Intelligence*, Vol. 9, No. 2, 1997.

[3] A. Cameron and H. Durrant-Whyte. A Bayesian Approach to Optimal Sensor Placement. *Int. J. of Robotics Res.*, Vol. 9, pp. 70–88, 1990.

[4] T. Dean and M. Boddy. An Analysis of Time-Dependent Planning. In *Proceedings of AAAI-88*, pp. 49–54, 1988.

[5] R. Goodwin. Reasoning about When to Start Acting. In *Proceedings of the 2nd Int. Conf. on Artificial Intelligentce Planning Systems*, pp. 86–91, 1994.

[6] S.A. Hutchinson and A.C. Kak. Planning Sensing Strategies in a Robot Work Cell with Multi-Sensor Capabilities. *IEEE Trans. on Robotics and Automat.*, Vol. 5, No. 6, pp. 765–783, 1989.

[7] S. Kambhampati, C.A. Knoblock, and Q. Yang. Planning as Refinement Search: A Unified Framework for Evaluating Design Tradeoffs in Partial-Order Planning. *Artificial Intelligence*, Vol. 76, pp. 167–238, 1995.

[8] R.E. Korf. Real-Time Heuristic Search. *Artificial Intelligence*, Vol. 42, pp. 189–211, 1990.

[9] T. Matsuyama. Cooperative Distributed Vision – Dynamic Integration of Visual Perception, Action, and Communication. In *Proc. Int. Workshop on Cooperative Distributed Vision*, pp. 1–39, 1998.

[10] J. Miura and Y. Shirai. Parallelizing Planning and Action of a Mobile Robot Based on Planning-Action Consistency. In *Proceedings of the 2001 IEEE Int. Conf. on Robotics and Automation*, pp. 1750–1756, 2001.

[11] I. Nourbakhsh. *Interleaving Planning and Execution for Autonomous Robots*. Kluwer Academic Publishers, 1997.

[12] M. Pollack and J. Horty. There's More to Life than Making Plans: Plan Management in Dynamic, Multi-Agent Environments. *AI Magazine*, Vol. 20, No. 4, pp. 71–84, 1999.

[13] S. Russell and E. Wefald. *Do The Right Thing*. The MIT Press, 1991.

[14] K. Sycara. Multiagent Systems. *AI Magazine*, Vol. 19, No. 2, pp. 79–92, 1998.

[15] H. Tsutsui, J. Miura, and Y. Shirai. Optical Flow-Based Person Tracking using Multiple Cameras. In *Proceedings of the 2001 Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, pp. 91–96, 2001.

[16] M. Zweben and M. Fox, editors. *Intelligent Scheduling*. Morgan Kaufmann, 1994.