Gaining diagnostic information for fault isolation

Conference Paper · August 2013

DOI: 10.1109/ICInfA.2013.6720379

6 authors, including:

Zeyang Xia
Chinese Academy of Sciences
59 PUBLICATIONS **174** CITATIONS
SEE PROFILE

Ying Hu
Chinese Academy of Sciences
58 PUBLICATIONS **245** CITATIONS
SEE PROFILE

Proceeding of the IEEE
International Conference on Information and Automation
Yinchuan, China, August 2013

Gaining Diagnostic Information for Fault Isolation

Jun Tang[1,2,3] Jun Zhang[1,2,3] Xiaojun Wang[1,2,3] Zeyang Xia[1,2,3*] Ying Hu[1,2,3*] Jianwei Zhang[4]
1.Center for Cognitive Technologies, Shenzhen Institutes of Advanced Technology,
Chinese Academy of Sciences, Shenzhen, China
2.Guangdong Provincial Key Laboratory of Robotics and Intelligent System, Shenzhen Institutes of Advanced Technology,
Chinese Academy of Sciences, Shenzhen, China
3.The Chinese University of Hong Kong, Hong Kong, China
4.TAMS, Department of Informatics, University of Hamburg, Hamburg, Germany
*Correspondence should be addressed to Drs. Xia and Hu at [zy.xia;ying.hu]@siat.ac.cn/+86-755-8639-2181/2182.

***Abstract*—Robots in dynamic and uncertain environments are vulnerable to mission failures due to external perturbation or internal malfunctions. Diagnosis is the process to detect, locate or even assess the fault. Since robots rely on their function modules to sense the external environment, it is difficult to locate the fault under uncertainties of robot components. The situation can be worse when there is also uncertainty about the environment. To resolve this issue, this paper proposes a new method to actively gain diagnostic information to locate the failure-cause more accurately under uncertainties. An integrated strategy of self-function-checking and diagnostic-plan is described. Validation using JSHOP2 planner showed that robots using this strategy was able to locate failure-cause with high autonomy.**

***Keywords*—active diagnosis; mobile robot; realtime knowledge**

I. INTRODUCTION

Mobile robots in dynamic and uncertain environments often encounter unexpected mission failures. Failures are always from two resources, (i) malfunctions or accidental failures of sensors/actuators and (ii) inappropriate plans. Three factors usually make it intractable to form a perfect mission-plan: (i) robots usually do not have full knowledge of the environment; (ii) other agents (human or non-human) may make adverse changes in the environment unexpectedly; (iii) actions may not work out as expected.

As failures are often inevitable, it is commonly accepted that the ability to detect and fix problems is crucial [1]. Thus, strategies of diagnosis and monitoring are widely used to determine what fault caused the mission failure. Both diagnosis and monitoring have been extensively studied, and there are many strategies. The ones most relevant to this paper are expectation-based monitoring and history-based diagnosis. The former strategy describes some expectations of actions [2, 3]. The expectations will be checked by some sensor to verify the success of the monitored action [2]. Prior approaches mainly have two shortages: (i) they are not applicable for actions that are hard to verify; (ii) monitoring actions are usually pre-coded and static, thus not able to tackle some dynamic environments. History-based diagnosis approaches, on the other hand, mainly focus on mining knowledge out of the passive observation of mission history. Their main purpose is to form reasonable explanations for mission-failures and sift out the most probable one [4]. The main drawback of history-based diagnosis is their lack of instructive information, as they work like doctors who only listen to patients but ask no professional questions. On the other hand, believing in a wrong diagnosis may waste the robot much energy and time to disprove it through trial. In the worst case, it may even lead to evitable mission failures. Thus we feel it reasonable to spend some time and energy to gain more diagnostic information for fault-isolation, especially when mission-success is more important than energy/time saving.

As faults are difficult to locate, we focused this paper on gaining diagnostic information to help locating the actual fault among multiple plausible explanations-false knowledge, component malfunction or previously unnoticed failure. We do not tackle fault-detection and diagnostic information processing is only briefly discussed. Diagnostic information can be gained through the following approaches this paper presents: self-function-checking (SFC) and diagnostic plans. SFCs are pre-coded actions intended to check the general status of function modules and gain certainty about their action-results indirectly. This approach can be valuable when action results are hard to verify directly. In addition, we give the robot some general guide about what kind of diagnostic actions can be conducted to check a specific type of failures. When a failure happens, robots can determine which diagnostic method is applicable in the present situation. The results of diagnostic plans can render valuable information to help locate the cause of the mission-failure.

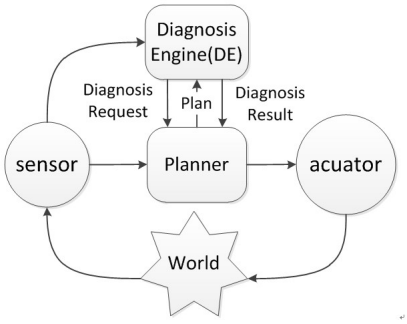

Fig. 1. DE commands diagnosis and process diagnostic information

The remaining parts of this paper are arranged as follows. Section II introduces related work. Section III discusses what diagnostic information can be gained through active diagnosis. Sections IV and V explore two strategies to gain diagnostic information. Section VI discusses the techniques for arranging multiple diagnostic actions and processing the diagnostic information; Section VII provides some test scenarios to illustrate

how the presented strategies can help locate the failure-cause. Discussions and conclusions are in Sections VIII and IX respectively.

II. RELATED WORKS

A. Classic Approaches

Two categories of approaches to tackle unexpected mission failures have been reported, expectation-based monitoring and passive-diagnosis.

Approaches of expectation-based monitoring pre-describe explicit expectations (or deduce implicit ones) about the outcome of actions, the expectations will be checked by some sensor to check actions' results [2, 3]. Expectation-based monitoring is limited when robot's functions have no outcome that can be easily verified. For instance, if the camera module of a robot declares some visual target to be absent, no other module can easily check the absence of the target.

Approaches of passive-diagnosis are usually based on the passive observation of mission-history. They try to reason about what happened or what didn't happen unexpectedly or what initial believes were wrong, so as to form reasonable explanations for the mission failure [5]. In order to isolate the most attractive explanation out of a possibly large set of hypotheses, probabilities and the amount of variation from the original "history" are usually used to rank the alternatives [6]. Passive-diagnosis approaches usually focus on reasoning techniques to find the most probable failure-cause. However, they usually have to rely on the meager knowledge gained in the passive observation of the mission "history". As a result, the limitation of knowledge often makes passive-diagnosis injudicious when there are multiple potential failure-causes.

B. Gainning Information

Researchers of industrial fields and distributed systems have explored the idea of active diagnosis [7, 8]. For robotic systems, there are also some researches involving active diagnostic-information gaining to some extent. However, previous approaches usually rely on the redundancy of the system-they usually use a redundant sensor/actuator to check the suspicious one. For example, [9] presents a method that can accomplish original tasks while accumulating diagnostic information at the same time. In its case, there are multiple plans to accomplish the same task. When a failure happens-some function-modules become suspicious-they form diagnostic plans. In these plans, they eschew some suspicious modules while retaining some others. Then, different modules can be decoupled and their status can be accessed separately and more accurately.

Unfortunately, because of energy and/or economy concerns, mobile robots usually don't have redundant modules. Consequently there are many cases prohibiting robots from using a redundant sensor/actuator to check the suspicious ones. Trying to get some redundant modules, [10] tries to communicate with other robots and use them as a judge or referee. However, not every robot has access external referees. In the present paper, we extend robots' ability to gain diagnostic information into non-redundancy and no-referee situations.

Aiming at similar purposes, work of [3, 11] are quite instructive as they use semantic knowledge to deduce implicit expectation of actions to form diagnosis/monitoring actions-like checking for beds to confirm/disprove that a robot has got into a bedroom. However, the reliance on pre-coded knowledge about world state constrains this approaches' applicability in relatively static environments or short-run operations. We will show that pre-coded knowledge is neither indispensable nor the most efficient.

C. DIAGNOSTIC INFORMATION

Diagnostic information can be gained through forming and conducting diagnostic plans/actions. These plans/actions may be intended to (i) check whether the world state is as expected, or (ii) check whether the robot's components are functioning properly, or (iii) check whether preceding actions are really successful. These three checking operations correspond to the three kinds of faults listed below:

1) False knowledge: initial errors in the robot's belief about the world state, including errors caused by unnoticed external changes.

2) Function failure: the malfunction (accidental failure) of a function module.

3) Previous failure: previously unrecognized failure which caused the current recognized one.

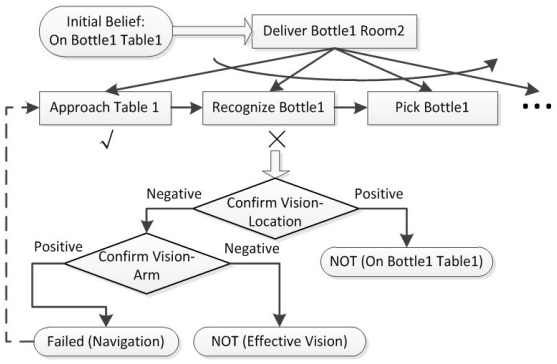

Fig. 2. Examplary diagnosis for recognition-failure after navigation

Generally, positive results in one of the three kinds of checking can extenuate the probability of its corresponding failure-cause; on the other hand, negative results in any of them will be a confirmation of the culpability of its corresponding failure-cause.

Among the three failure-causes, false-knowledge detection is not a focus of this paper. This is because when the robot has highly credible sensor-modules, it could simply take its observation as the new belief with confidence. Thus, we focus on gaining certainties about the status of robots' components.

In the remaining parts, we assume the world and the robot to "change slowly": no new external changes or malfunctions in robot-components will arise during the diagnosis process, nor will a defunct component recover automatically. This simplification doesn't deprive us of much generality, because even if new faults occur during diagnosis, we could recognize them with high possibility. In fact, it is more troublesome that we may find the new fault instead of the one caused

the original failure. However, through recursive conduction of active diagnosis, the robot is likely to locate all faults. Loosening this slow-change assumption just renders a worst case scenario-new changes arise constantly-in which no robot is likely to have good performance.

III. SELF FUNCTION CHECKING

To check the status of function modules, our approach employs self-function-checking (SFC) methods. By self-checking, we do not mean the ordinary internal signal-based checking like that computer systems do when they start. Instead, self-checking here are some pre-coded representative actions that reflect the general status of components.

The essence of SFC is to form a scenario in which the expected result of a module can be controlled by another one, thus the actual result can be compared with its expectation to reflect the status of function modules. In order to make different aspects of SFC controllable, usually only components of the robot should be involved. For instance, as in Fig 3, to check whether the camera module can recognize objects properly, the robot may position its grasper in the ken of the camera, and then check if the vision module can recognize the grasper (we assume recognition of different objects use the same camera and software/algorithm). Furthermore, if the vision module can locate targets in three-dimensional space (3-D), its report about the grasper's location can be compared with the report of encoders in the manipulation module. In this way, the robot can check whether its vision module and manipulation module is functioning properly.

```
(: method (confirm-vision-arm)            ; method name
( (available arm) (available vision) )    ;condition of this method
( (!set-arm-reference) (!recognize-grasper) (!reset-arm-reference) )
                                          ;actions to be conducted )
```

Fig. 3. Examplary SFC for checking vision/arm

In monitoring community, traditional idea of using a sensor/effector to verify the results of another sensor/effector is widely studied-a classic example is trying to use robot arms to catch a target to check whether the visual module has accurately located a target. However, in some situations, actions' results may not be easy to verify, especially for sensing actions. On the contrary, SFCs are devised so that its result can be accessed easily and credibly.

SFC strategy is inspired by prior monitoring work to some extent, but it has important differences: (i) mission-independency-as SFCs offer general report about modules' status, one SFC can offer instructive information for diagnosis of diverse actions of a module; (ii) environment-independency-as no external resources are needed, SFCs have better applicability; in addition, as every aspect of SFCs can be arbitrarily controlled by the robot, uncertainties can be ruled out to the highest extent.

Furthermore, SFC inherited the advantages of traditional monitoring work: (i) it enables mutual checking between a pair of heterogeneous sensors/effectors-this characteristic is important to enhance applicability for robots that have no redundant sensor/effectors; (ii) it is relatively easy to devise, as only the function description of a module is needed; (iii) it can reflect the influence of environmental factors like illumination an temperature-it can detect "camera-malfunction" in dark rooms.

IV. DIAGNOSTIC PLANS

A. Comprehensive Checking

Although SFCs can be informative about a pair of components' status, they are not necessarily the most efficient. Luckily, it is possible to check a function module and verify the results of previous actions at the same time-when previous actions' success is the precondition of another module's action. To do this, robots need "helpful" real time knowledge about the world to deduce applicable diagnostic plans.

For example, when a delivery robot fails to recognize the target after navigating to approach it, whether this mission failure is caused by a navigation failure or visual malfunction may not be certain. As the robot need to check the status of its vision module, the SFC in Fig. 3 is optional. However, the robot may have some knowledge about the existence of another similar object at the same position. So, instead of activating the pre-coded SFC to recognize its own grasper, it can try to find the other object there. As in Fig. 5, the robot can first check whether there is a potential "reference-object" at the same position the target is expected to be. If there is one, the robot can try to recognize it. Success in finding the reference object could affirm both the effectiveness of the vision module and the success of the previous navigation (the effectiveness of the navigation module can also be verified).

```
(:- (have-reference ?object ?location)  ⇦
   (and (near robot ?location) (at ?object ?location) ) )
```

Fig. 4. Method for confirming vision and location

```
(: method (verify-vision-location)
 (  (have-reference ?object ?location)
    (effective arm)(effective vision)  )    ; preconditions
 (  (!confirm ?object ?table)(!grasp ?object ?table)
    (!put ?object ?table)  )  )             ;actions
```

Fig. 5. Axiom for determing whether there are references

B. For previous failures

Virtually, detection of previous-failures is a problem about observing the results of previous actions. Traditional approaches focus on observing the direct consequence; some also check the implicit expectations. Here, we move one step further, from "what should've been caused by this action" to "what should've been made possible by this action". Following this principle, we equip robots with result-verifying methods that

have the successful result of the being-checked action as a precondition. As the result of this method reflects the status of its preconditions, the result of the being-checked action can be checked indirectly. In addition, with the extension in optional diagnostic actions, we can extend our approach's applicability. Furthermore, as robots' real-time knowledge is used to deduce applicable diagnostic-plans, our approach is expected to have better performances in dynamic environments (and long-run operations) than traditional approaches that rely on pre-coded knowledge about the world.

To exploit opportunities offered by external factors, three kinds of knowledge are needed: (i) what can be confirmed by what actions, (ii) what preconditions are required for these actions (Fig. 4), (iii) what axioms can be used to help deduce whether these preconditions can be satisfied (Fig. 5). The first should be coded in the diagnosis engine, so that when a failure happens it can deduce what should be checked. The latter two can be coded in the planner (for JSHOP2 planner [12], in its domain description) so that the mission-planner can be also used as a diagnosis-planer.

If an augment in complexity and time/energy consumption is tolerable, diagnostic plans can be applicable in more adverse environment and render more credible diagnostic information. In some situations, there may be no instant applicable diagnostic plans. In order to make one possible, "preparatory actions" may be needed to change the robot's location or the environment to some extent. For instance, when method confirm-vision-location (Fig. 3) is needed, there may not be a "reference-object" at the very location the robot is at. Chances are that there may be one on a nearby table. To check its position and vision, the robot may move a short distance to the nearby table and try to recognize the reference there (and move back if necessary). Although helpful in some situations, preparatory actions should be considered only when no direct diagnostic plans are possible, because (i) preparatory actions will take extra time and energy, (ii) and as the robot is very likely to have malfunctions under the presence of failure, further actions increase risks both in changing the world in an unwanted way and leading to more action failures, (iii)finally, as robots' performances are usually influenced by external environment, diagnostic plans carried out in a changed environment may not render reliable information for the original situation. To achieve a higher credibility, other monitoring approaches may be used in a diagnostic plan to ensure an accurate report. Although just like preparatory actions, this may consume extra time and energy, in some cases, such sacrifice can be reasonable. For example, the method in Fig. 3 activates a visual-recognition trial to check visual module. For some robots, their visual-module may recognize "ghost objects" and report false successes. To rule out such possibilities, the robot may uses its arm to try to catch the object [3].

C. For Function Failures

The general strategy to detect function failures is to try this function in a task. In fact checking previous actions and checking function modules is not necessarily isolated. Methods similar to those for previous failures can also be used to check function modules. The only difference lies in their purpose and the situations in which they can be applicable; the similarity is

```
(: method (verify-vision-location)
 (  (have-reference  ?object  ?location)
    (effective arm)(effective vision)  )    ; preconditions
 (  (!confirm  ?object  ?table)(!grasp  ?object ?table)
    (!put ?object ?table)  )  )                ;actions
```

Fig. 6. Method for verifying visual report by grasping

that, both of them involves external resources and use robots' real-time knowledge to deduce what is applicable.

For checking a specific module, compared with SFCs, diagnostic plans can better reflect modules' performance in ordinary missions and are easier to devise. However, as diagnostic pans rely on external resources, they may not be applicable in some "barren" environments and induces more uncertainty from external environment.

V. FAULT LOCATION

In this section, we briefly discuss how to arrange the order to conduct different diagnoses and how to process the information gained to locate failure-cause-to deduce the posterior probabilities of plausible failure-causes and find the most probable one.

A. Arranging Diagnostic Plans

For mission failures there are usually multiple suspicious failure-causes, consequently there can be multiple optional diagnostic plans. Determining the order in which they should be conducted is also an interesting topic [13, 14]. An intuitive idea is to weigh the costs and gains. However, that is not easy because many actions' costs are hard to estimate beforehand. For example, navigation to a place nearby is expected to be relatively "cheap". But if the direct way to such a place is blocked, it may take even more time than navigating to a faraway place.

As our ability to weigh the costs and gains is limited, determining the optimal (even sub-optimal) sequence of diagnostic plans requires very sophisticated algorithms [13], thus in this paper we just use an intuitive method. We take the failed action as the root: check the module conducting this action first and then check its preconditions in random order.

B. Processing Information

There have been relatively sufficient articles on reasoning about the failure-cause which take the history of missions as input [4, 11]. Our diagnosis plans can be an informative supplement of the original mission history. In the following section, we used the Bayes function to deduce the posterior probability of failure causes after conducting the diagnosis plans.

During diagnosis, if the probability of some fault passes a guilty-threshold (for example 0.8), this fault will be viewed as the actual fault and no further diagnosis action is needed. After all diagnosis plans are conducted, if every potential cause's probability is under the innocent-threshold (for example 0.3),

the failure will be considered accidental, and the robot may just try once again. If faults' probabilities are between the guilty-threshold and innocent-threshold, the most probable one will be believed to be the actual fault.

VI. EXPREIMENT

In this section, we demonstrate the application of SFC and diagnostic plans in a typical office-delivery task. Our strategy is implemented on a mobile robot designed by colleagues in our lab. The robot was based on Robot Operating System and JSHP2 planner [12, 15]. Apart from some auxiliary sensors/actuators for navigation, the robot is mainly equipped with a bumblebee binocular camera and a 7 degree-of-freedom manipulator. The robots' visual module used SURF algorithm to recognize targets and the embedded algorithm of the camera was used to locate the target in 3-D. The manipulator could be activated to grasp an object and convey it to the desired destination.

A. SFC Scenarios

We applied self-function-checking in the manipulation module and vision module (Fig. 3). We pasted a visual-mark on the grasper to alleviate the difficulty for visual recognition. The SFC involved setting the grasper/visual-mark to a specific location and use the visual module to recognize the grasper/visual-mark and locate it in 3-D. Through experience, we knew our manipulator seldom has position error bigger than 2cm; and in normal conditions, visual module's reports of are seldom 2cm away from targets' accurate positions, thus we set the failure-threshold of this SFC to 4cm of difference between the two measurements.

In the experiment, by conducting this SFC, the robot succeeded in three scenarios: (i) detecting the malfunction of visual module when we covered the camera with a cloth and when we turned off the lights at night to form a dark environment; (ii) confirming the effectiveness of visual module and manipulation module in normal conditions; (iii) detecting the fault of manipulation module when we set the grasper 5cm away from the expected position. In addition, the recognition of a juice bottle-the typical target in our delivery experiments-and the recognition of the grasper had the same result (success or failure) in 8 out of 10 tests when we changed the light condition, partly because the two objects had similar sizes, shapes and distances from the camera.

As experiments showed, this SFC is able to reflect the general status of vision module and manipulation module: when the vision module is effective it can help to estimate the approximate condition of the manipulator; the manipulator, on the other hand, is able to help detect whether the vision module can recognize and accurately locate targets.

B. Diagnosis Plan Scenario

We tested our active diagnosis-plan strategy in the following scenario: our robot is required to get a bottle B1 from room R1, and convey it to another room R2, which is connected with R1 by a corridor. B1 and another bottle B2 were expected to be on table T1 (in R1) with probability of 0.7 and 0.95 respectively. The other 0.3 probability for B1's location was assigned to "unknown". The robot might navigate to a false

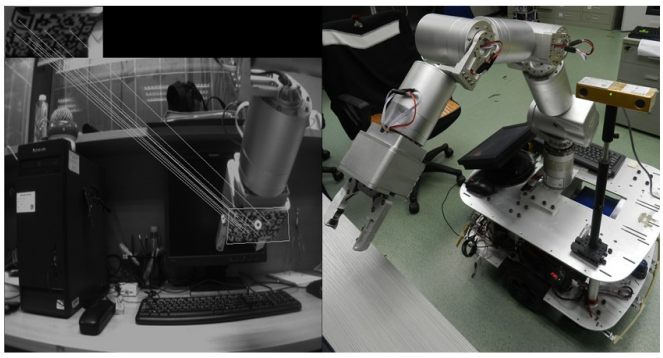

Fig. 7. Robot conducting confim-vision-arm(left is the vision module)

place with a probability of 0.15; the environment could be too dark with a probability of 0.2. The arm module is relatively reliable and had a probability 0.05 to be unable to reach desired positions. (All these prior probabilities were gained through experience and set manually before conducting the mission.) We set the guilty-threshold to probability 0.9 and the incessant-threshold to 0.3. We assumed the following simplifications: the vision module would not fail when light condition was favorable; the world "changed slowly"; all random events were probabilistically independent. In this experiment, we mainly used the diagnostic methods in Fig. 3, 4 and 5.

In the first case, we took away B1. As expected, when the robot got to T1 it failed to recognize B1. Then the original delivery mission was suspended and diagnosis was activated. The visual module and result of previous navigation needed to be checked-the existence of B1 can't be checked directly. Retrieving in the diagnosis engine the robot found that method confirm-vision-location (Fig. 4) can check both the visual module and the robot's location. This method was activated as it was applicable in that situation because of the existence of a reference-B2. Thus the robot tried to recognize B2 on T1. This diagnosis plan succeeded, which confirmed both the robot's position and the proper function of visual module, reducing their malfunction probabilities to 0, leaving only one fault plausible-the false belief about the existence of B1. Thus the diagnosis engine took the belief that the failure was caused by the absence of B1, with this new belief, the planner found the mission not accomplishable and aborted it.

TABLE I. PROBABILITY OF FAILURE IN FIRST SCENARIO

	NavigationFailure	PoorLight	TargetAbsence
Prior Probability	0.2	0.2	0.3
After Failure	0.36	0.36	0.54
After Diagnosis	0	0	1

In this case, the robot retrieved an available "landmark" B2 in its real-time knowledge, and with one diagnostic action, it narrowed down the number of possible faults from 3 to 1.

In the second case, we set the robot's initial belief to it has reached T1 through previous navigation, while in fact we put it about two meters away. The robot tried to activate its camera to recognize B1 directly when it started the mission. As in the previous case, the first trial to recognize B1 failed and confirm-vision-location was conducted. This diagnosis action also failed, and no fault passed the guilty-threshold

but all of them passed the innocent-threshold. Then to determine which the actual fault was, verification of vision was conducted-trying to verify the function of navigation module will be much more "expensive". Thus, the robot activated the SFC confirm-vision-arm. The result is that, the vision module successfully recognized the grasper, alleviating the probability of the illumination-caused vision malfunction to zero, while the probability of the navigation failure passed the guilty-threshold. Then with the pre-coded recover strategy for false-navigation-"try again", the robot activated its navigation module again, and this time it successfully got to table, and the mission successfully carried out after the diagnosis.

TABLE II. PROBABILITY OF FAILURE IN SECOND SCENARIO

	NavigationFailure	PoorLight	Bottle1'sAbsence
Prior Probability	0.2	0.2	0.3
After Failure	0.36	0.36	0.54
After Comfirm-vision-location	0.54	0.54	0.81
After Comfirm-vision-arm	0.94	0	0.34

Without active diagnosis, the robot would find Bottle1's absence the most plausible explanation. If the robot took it as belief, it would have aborted the mission mistakenly. Luckily, the SFC involving vision module and robot-arm module dramatically enhanced the accuracy of the diagnosis and helped the robot to locate the actual fault-navigation fault.

VII. DISCUSSIONS

To cope with dynamic environments, we use robot's real-time knowledge to find available resources and deduce diagnostic plans. By using real time knowledge, we exploit the complexity of the world which is usually viewed as a burden for planning.

For SFC, apart from the examples in previous chapters, we believe there can be many other possible extensions of this self-checking idea. For example, a robot may wave its arm in front of its laser scanner to check the scanner's distance reports or the arm's movement. Furthermore, to achieve a smaller granularity, different components of a function module can be used to check each other. For instance, the robot can turn a cycle to check whether its encoders and gyro is working well. The counts of the encoder can be used to deduce the angle the robot has gyrated. This deduced angle can be compared with the report of the gyro, and an appreciable discrepancy will be a sign of malfunction in one of the two sensors.

Diagnosis plans/actions can enhance robots' robustness in multiple ways. As they can help locating the failure-cause, along with replanning or recovery actions[16], they can help robots to recover from unexpected anomalies. Even if the diagnosis asserts the failure to be unrecoverable-such as caused by a non-repairable sensor, as long as the diagnosis is accurate, it can save the robot some time and energy by preventing it from carrying out "mission-impossible".

Similar strategies can also be used for monitoring purpose without major changes-diagnostic plans and SFCs can also be conducted during the mission to check the status of sensors/actuators or recognize unnoticed action failures.

Although we present our strategy only in JSHOP2 planner, we believe the essential ideas can cope with other planners which have similar expressiveness as JSHOP2.

VIII. CONCLUSIONS

We presented a new perspective for gaining diagnostic information by combining two strategies: self-function-checking (SFC) and diagnostic plans. SFCs are standard actions, during which some variable will be measured by different modules. These measuring results can be compared to detect malfunction or verify effectiveness of function modules. In addition, robots are given some method patterns about what can be checked by what actions. With their real-time knowledge, robots can automatically deduce which of these diagnostic methods are applicable. By conducting SFCs and diagnostic plans, robots can gain certainty about the status of their function modules and the world state. Consequently, they can locate failure-causes more accurately.

ACKNOWLEDGMENT

This research was partially supported by the Key Fundamental Research Program of Shenzhen (JC201005270375A) and National Science Foundation of China (No. 61210013, 61105132).

REFERENCES

[1] K. Talamadupula, J. Benton,et al,*Planning for human-robot teaming in open worlds,* ACM Transactions on Intelligent Systems and Technology (TIST), vol.1:14, 2010

[2] R. J.Doyle, D. J.Atkinson,and R. S.Doshi, *Generating perception requests and expectations to verify the execution of plans,* AAAI,1986

[3] G. Steinbauer and F. Wotawa,*Robust plan execution using model-based reasoning,* Advanced Robotics, vol.23:1315-1326, 2009.

[4] M. Molineaux, U. Kuter, and M. Klenk, *DiscoverHistory: understanding the past in planning and execution,* Autonomous Agents and Multiagent Systems Vol2:989-996,2012.

[5] M. A.Sotelo,L. M.Bergasa,et al, *ADVOCATE II: ADVanced On-board diagnosis and Control of Autonomous sysTEms II,* Computer Aided Systems Theory-EUROCAST 2003, ed: Springer, 2003, pp. 302-313.

[6] A. Bouguerra,L. Karlsson,and A. Saffiotti, *Handling uncertainty in semantic-knowledge based execution monitoring,* Intelligent Robots and Systems,pp.437-443,2007.

[7] M. Sampath,S. Lafortune,and D. Teneketzis, *Active diagnosis of discrete-event systems,* IEEE Transactions on Automatic Control, vol.43:908-929, 1998.

[8] M. Brodie,I. Rish,S. Ma,A. Beygelzimer,et al*Strategies for problem determination using probing,* Techni-cal report, IBM TJ Watson Research Center, 2002.

[9] L. Kuhn,B. Price,J. De.Kleer,et al*Pervasive diagnosis: The integration of diagnostic goals into production plans,* Proc. AAAI-2008, 2008

[10] M. T.Long, R. R.Murphy,and L. E.Parker, *Distributed multi-agent diagnosis and recovery from sensor failures,* Intelligent Robots and Systems,pp.2506-2513,2003.

[11] A. Bouguerra,L. Karlsson,and A. Saffiotti, *Monitoring the execution of robot plans using semantic knowledge,* Robotics and autonomous systems, vol.56:942-954, 2008.

[12] O. Ilghami,*Documentation for JSHOP2,* Department of Computer Science, University of Maryland, Tech. Rep, 2006.

[13] A. Feldman,G. Provan,and A. Van Gemund, *FRACTAL: Efficient fault isolation using active testing,* International Joint Conference on Artificial Intelligence,pp.778-784,2009.

[14] A. X.Zheng,I. Rish,and A. Beygelzimer, *Efficient test selection in active diagnosis via entropy approximation,* arXiv:1207.1418, 2012.

[15] M. Quigley,K. Conley,et al., *ROS: an open-source Robot Operating System,* International Conference on Robotics and Automation, 2009.

[16] T. Eiter,E. Erdem,and W. Faber, *Plan reversals for recovery in execution monitoring,* NMR, Action and Causality Track,pp.147-154,2004,.