Proceeding of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space: i-SAIRAS 2001, Canadian Space Agency, St-Hubert, Quebec, Canada, June 18-22, 2001.

Model Based Autonomy - How Model without Human Expertise Can Facilitate Fault Diagnosis and Other Spacecraft Tasks ? -

Shinichi Nakasuka and Shiro Ogasawara,
Department of Aeronautics and Astronautics, University of Tokyo
Hongo 7, Bunkyo-ku, Tokyo 113-8656, JAPAN
nakasuka@space.t.u-tokyo.ac.jp
Masayuki Takata, and Taizo Yamamoto
Information Processing Center, The University of Electro-Communications
Chofu-gaoka 1-5-1, Chofu-city, 182-8585, JAPAN

Keywords Autonomy, Fault Diagnosis, Model Based System, Attribute Generation, Machine Learning

Abstract

A model based fault diagnosis system for satellites is proposed which autonomously infers the cause of failure from telemetry-type time history data using functional models of satellite subsystems. The key research issue is how and to what extent a model without human expertise can solve the fault diagnosis problem. The evaluation results of the prototype system indicate that though non-deterministic search process guided by some weak logic can solve the problem within practical time to some extent, it still has severe limitations coming from the lack of capability to measure the similarity of two telemetry data. Extraction and utilization of attributes useful for diagnosis task is a promising approach, and one method is proposed for autonomously defining such important attributes from telemetry data.

1. Introduction

Autonomy is one of key technologies for future spacecraft ([1]-[3]). It not only reduces the project cost and personnel by reducing the ground operation workload, but also improves the satellite survivability against unpredicted failures or environmental hazards. Especially, for small, micro or nano scale satellites build by universities or deep space exploration spacecraft for which we cannot provide enough ground support, autonomy will be essential. Besides, recent tremendous advancement of high performance micro computer and large memory is beginning to make it possible to perform quite sophisticated onboard information processing including Artificial Intelligence-type reasoning even on micro or nano satellites.

One way of building autonomous onboard system is to incorporate expert systems using knowledge base of human expertise. Though large number of such systems have been proposed and some of them are actually used, the weak point of such systems is that the performance of the system is strictly limited by the quality and quantity of the implemented knowledge. Especially, recent spacecraft is becoming so complicated that even the human experts cannot grasp the embedded causal-effect relationships of the whole system, which makes it difficult, for example, for the knowledge based system to infer the real cause for a certain anomalous behavior of the telemetry data.

Model based system, which utilizes "model of causal-effect relationships of spacecraft" as the basis of inference, have been studied for years to compensate for the drawbacks of this expert system as well as actually demonstrated in space such as on DS-1 of New Millenium Project. The strong points of this method are that (1) it can deal with any fault diagnosis or task planning problem, even too complicated for human experts to solve in a short time, so far as the modeled causal-effect relationships can apply, and that (2) the derived solution is already proved by the model. However, in order to apply it to the real world problems, we must solve the difficult problems of knowledge capture (because the system requires lots of fundamental knowledge), combinatorial explosion of search space, and knowledge compilation (transforming the basic knowledge into ones suitable for model based inference), model coverage (how various situation including component failure can be modeled) and so on.

This paper describes a prototype software of a model based fault diagnosis system developed in our laboratory. Our aim in this research is to find out to what extent the model based system without human expertise can solve such complicated problems. The strong points as well as limitations of such model based system will be discussed and a new approach to compensating for the weak points will be given later. Finally the possibility to utilize a common model for various tasks such as fault diagnosis, task planning, reconfiguration, etc, will be discussed.

2. Model Based Fault Diagnosis System

2.1 Overview of the System

A prototype of a model based fault diagnosis system for satellite has been developed. As the first attempt, fault diagnosis of attitude control loop has been tried. The model includes the attitude control related subsystems (25 subsystems) including sensors (gyros, Earth sensor), actuators (reaction wheels, thrusters), an onboard computer, and support infrastructure such as a fuel tank and valves, a battery and other electric subsystems as in Fig.1. A detailed simulator has been developed for these subsystems including 90 possible failure modes such as in Table 1. A certain number of telemetry (such as gyro readout, command to thrusters, voltage of a certain line) are assumed which provide the information for fault diagnosis, as encircled by ellipses in Fig.1.

Fig. 2 shows an example of telemetry for the normal case, and Fig.3 shows the case where the x-axis gyro has large bias error in the same situation. The fault diagnosis problem can be formalized as "given a set of anomalous telemetry (called "actual telemetry" hereafter), the failed subsystem and its failure mode should be identified." When designing the prototype system, the following philosophy has been adopted.

(1) Human expertise should be excluded as much as possible from the inference mechanism
(2) The causal-effect relationships between subsystems and a simulator of the target system ("model") should be utilized as much as possible

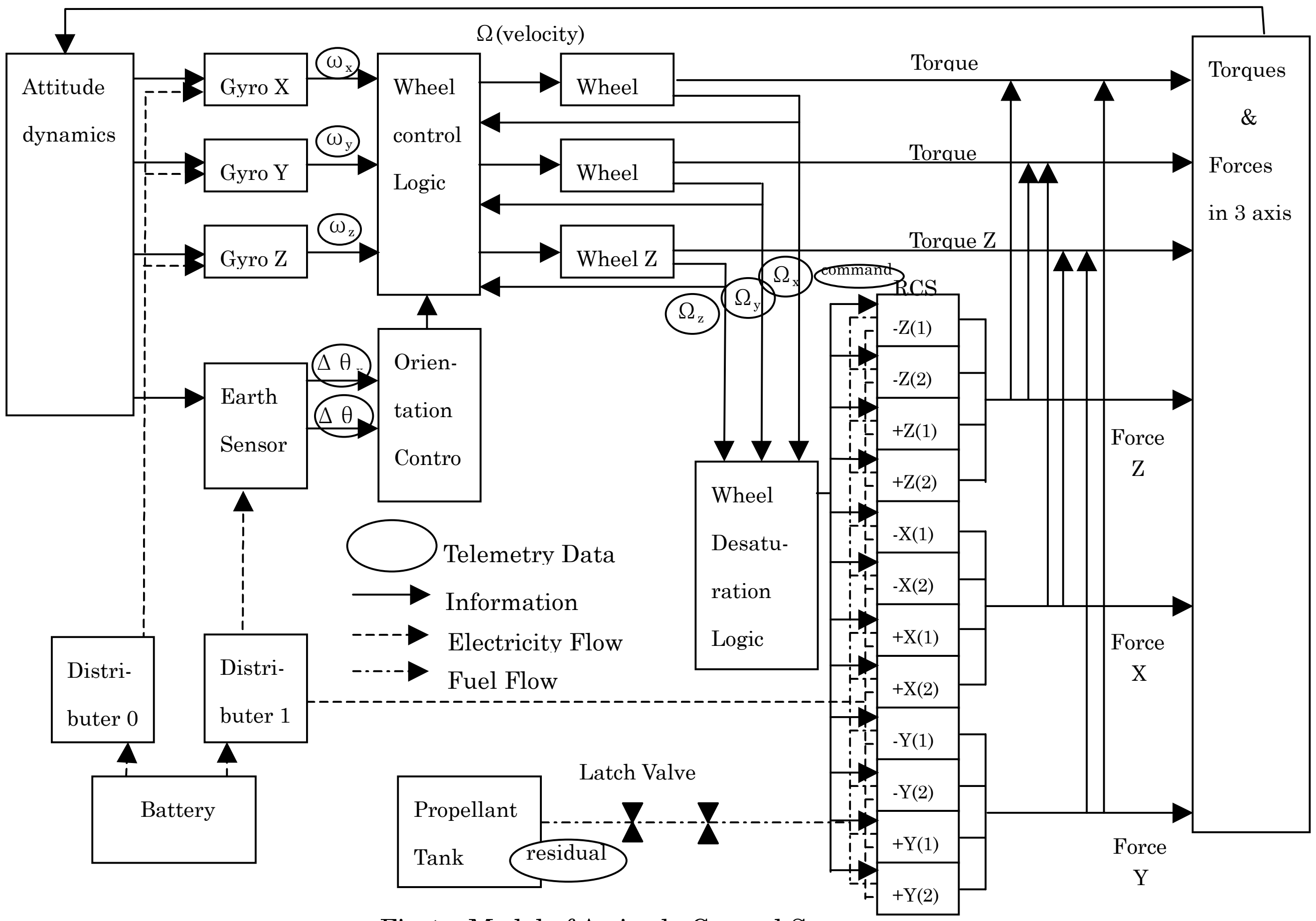


Fig.1 Model of Attitude Control System

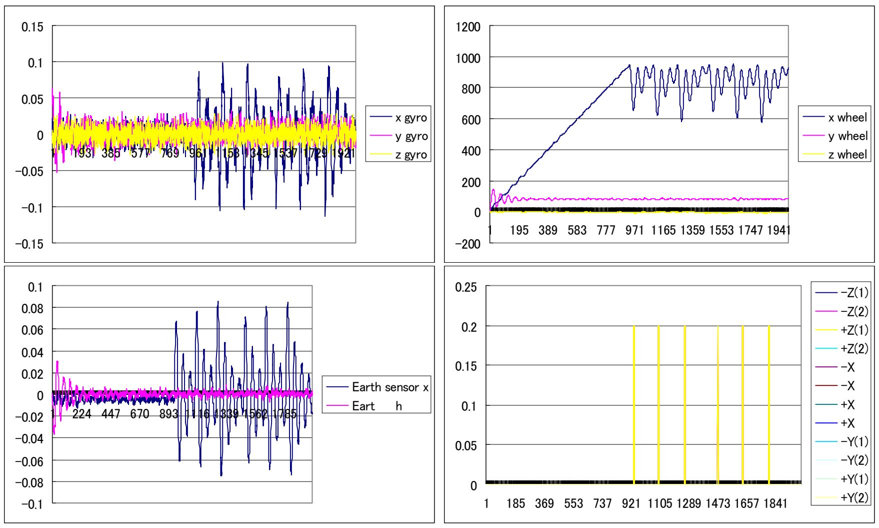


Fig.2 Telemetry for Normal (without failure) Case

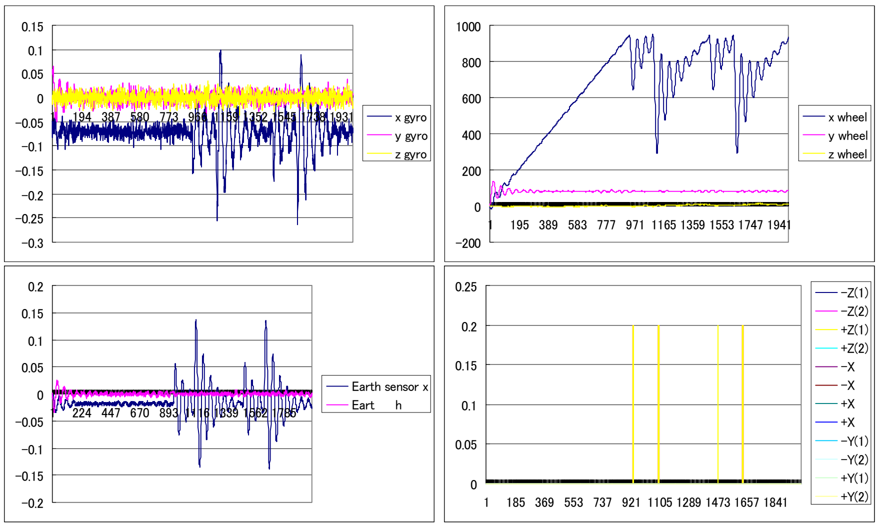


Fig.3 Telemetry for "X Gyro Large Bias Error" Case

Table 1 Failure Modes for Each Subsystem

――――――――――――――――――

<0> Gyro x <1> Gyro y <2> Gyro z
1) Scale Factor Error
2) Bias Error
3) Alignment Error
4) Zero Output
5) Stack to a Value
<3> Wheel x <4> Wheel y <5> Wheel z
1) Scale Factor Error
2) Large Friction
3) Large Time Constant
4) Constant Velocity
5) Sudden Stop
6) Lower Speed Limit
<6> Thruster -z(1) <7> Thruster -z(2) < 8> Thruster +z(1)
<9> Thruster +z(2)<10> Thruster -x(1) <11> Thruster -x(2)
<12>Thruster +x(1)<13> Thruster +x(2)<14> Thruster -y(1)
<15> Thruster -y(2)<16> Thruster +y(1)<17> Thruster +y(2)
1) Thrust Change
2) Large Minimum Impulse
3) No Thrust
4) Cannot Stop Thrust (valve stack open)
<18> Earth Sensor
1) No Output
2) Stack to a Value
3) Bias Error
<19> Battery
1) No Output
<20> Electricity Distributing System 0 (5v)
1) No Output
<21> Electricity Distributing System 1 (12v)
1) No Output
<22> Propellant Tank
1) No Fuel Supply
<23> Latch Valve <24> Valve
1) Stack closed

――――――――――――――――――

2.2 Fault Diagnosis Algorithm

Basically, the prototype system tries to find out the failed subsystem and its failure mode which would yield the actual anomalous telemetry readouts, using non-deterministic search procedure. This guarantees that any failure modeled in the system can be identified, except for the case when several failure modes yield the same telemetry outputs. Even in this case, however, the true failure mode will be among the several candidates which survive the screening process.

The problem of this method is, as is easily imagined, the combinatorial explosion of the search space. Therefore, the current system utilizes the following simple logic to exclude improbable failure modes.

<Logic 1> Let simulation be done, assuming that all the upstream subsystems of subsystem "A" behaves in the same way as "actual telemetry," and every subsystem is not faulty. Then, if the output of "A" coincides with "actual telemetry," "A" is not faulty.

<Logic 2> If in the above case the output of "A" does not coincide with "actual telemetry," "A" is faulty.

<Logic 3> Let simulation be performed, assuming that the subsystem "A" behaves in the same way as "actual telemetry," and every subsystem is not faulty. Then if the outputs of the downstream subsystems of "A" coincide with "actual telemetry," all the downstream subsystems of "A" are not faulty. (This logic assumes that the series of two or more faulty subsystems will never yield correct output.)

<Logic 4> The failures of an electric subsystem and fuel supply subsystem would have effect on all the downstream subsystems at the same time.

Please note that the above inference strategies are weak methods which do not depend on human expertise.

In order to do apply the logic, the simulator is equipped with the capability to simulate the system's behavior on condition that certain part(s) of the system (such as gyro output or wheel velocity) follows a prescribed time history. (This function can be called as "what-if analysis".)

The fault diagnosis system first searches for the faulty subsystem using Logic 2, and if no subsystem is found, then screens out not-faulty subsystems using Logic 1, 3, 4. If several candidates of faulty subsystem are identified, then number of simulations are performed assuming each of failure modes and parameter values one by one to find the combination of "failed subsystem/failure mode/parameter value" which yields simulated telemetry data the nearest to the "actual telemetry."

In the above procedure, it is required to measure the distance between the simulated telemetry and "actual telemetry" in order to decide "coincide" /"not coincide" or which is the nearest telemetry. For this objective, our system employs the following measure: the simulated telemetry is generated N times by using different random seeds for noise generation (for gyro measurement noise, disturbances, etc.), and the upper and lower bounds of the simulated telemetry are calculated. Then the distance between "actual telemetry" and these upper-lower bounds is calculated at each time instance which is accumulated over a certain time period to yield the distance measure. In this measure, too, no human expertise has been employed. Let us discuss the validity of this measure later.

3 Evaluation of the Prototype System

The prototype system is evaluated by randomly generated failure modes (single failure case). Table 2 shows the test results. The "true one in 1st-2nd means that the right answer was in the first and second candidates estimated by the prototype system. Performance for two types of requirements, the case that only failed part should be identified and the case that the failed part and failure mode should be identified, are evaluated. The computational time is about 3 to 20 minutes, using PC with Pentium 3 processor. Table 3 shows the diagnosis result for the telemetry of Fig.3, which shows that the true failure mode is estimated correctly.

Table 2 Estimation Accuracy of the Prototype System

Cases	Failed part only	Failed part + mode
True one in 1st candidate	65%	52%
True one in 1st-2nd	80%	74%
True one in 1st-3rd	85%	83%

Table 3 Fault Diagnosis Result for Fig.3 Telemetry

```
        Elapsed time(s) = 1078
        Candidates are:
    1)  Gyro x: Bias error    (param. -0.06)
    2)  Gyro x: Scale factor error (param. 0.80)
    3)  Gyro z: Scale factor error (param. 0.60)
    4)  Earth sensor bias error (param. 0.04)
    5)  Gyro y: Scale factor error (param. 0.40)

    Cf. True fault: Gyro x: Bias error (param. -0.07)
```

4. Discussion on Performance Limitation

Table 2 shows that the system without human expertise can estimate the cause of failure to some extent. The performance, however, falls short of the level that the system can be reliably utilized. Let us first try to find out the true cause of such limitations. The following factors seem relevant to the performance limitation.

(1) Different failure modes sometimes yield almost the same telemetry patterns.
(2) The distance measure employed in our system (stated in 2.2) sometimes cannot detect and utilize "pattern-type" important information in the telemetry data.

(1) is the limitation coming from the design phase; i.e., once the definition of telemetry has been made, the fault diagnosis system cannot overcome the limitation in this respect. Proper definition of telemetry in order to enhance fault diagnosis capability will be another important issue, but will not be addressed here.

(2) is an important issue which seems to limit the capability of many of current AI systems; i.e. a problem related to representation capability. For example, two telemetry data which behaves like sinusoidal curve with exactly the same frequency and amplitude but different phases are considered to have large difference in the current distance measure. But human can easily find the two telemetry has a similar symptoms. This is because human can extract "pattern-type" attributes from the original data and utilize them for inference. Important attributes will be such ones that two telemetry data having similar attributes tend to indicate the same failure

Therefore, one straightforward way to enhance the system capability is to include such attributes, which human experts think are important for fault diagnosis, in the calculation of distance. But this strategy still has the limitations coming from the limitation of human capability; i.e. even human experts cannot specify all the important attributes especially for diagnosing complicated systems. In conclusion, in order to pursue truly capable fault diagnosis system, the system itself should have the capability to define "the measure of distance" or "important attributes useful for the diagnosis task." The next chapter will propose one method for this objective.

5. Autonomous Attribute Generation

5.1 Overview

We have investigated a methodology to extract features (or attributes) useful for a certain objective (such as classification) autonomously. This has been achieved by combining unsupervised and supervised learning. Learning Vector Quantization (LVQ [4]-[6]) method is used as the unsupervised learning method which derives several feature candidates which seem important.

LVQ can be used to generate clusters inherent in the data and to give "typical" patterns of each cluster. If the generated cluster corresponds to classes, i.e., data of different classes tend to belong to different clusters, then the obtained typical pattern of each cluster will be possibly informative for classification. The original LVQ method, however, cannot take the class information into account for clustering, and so its modified version (explained later) is utilized so that the above-mentioned property is achieved. The second problem is that even the modified version of LVQ tends to generates quite many "typical" features not all of which have discriminative capability for classification. So, the system must have the second ability to select from the pool of generated features useful ones for classification.

For this objective, supervised learning using decision trees is performed following LVQ. In this method, sequences of decision making, using one attribute at a time, are generated in a form of a tree. Which attribute should be used in what way at each decision point is determined so that it has the largest discriminative power of different classes. Though this selection of an attribute can be said to be rather local, the generation process of a decision tree can certainly indicate which attributes are useful for classification. We use this method after LVQ generates many features in order to identify only such ones truly relevant for classification.

5.2 Overall Algorithm

An example classification task assumed in this paper is to discriminate x-y curves of mathematical functions of several families. A certain number of training data of such time-histories with class names attached are assumed to be given, which are used for feature extraction and decision tree generation. No *a priori* information is given as to important shape/pattern primitives

The system operates in the following manner. First,

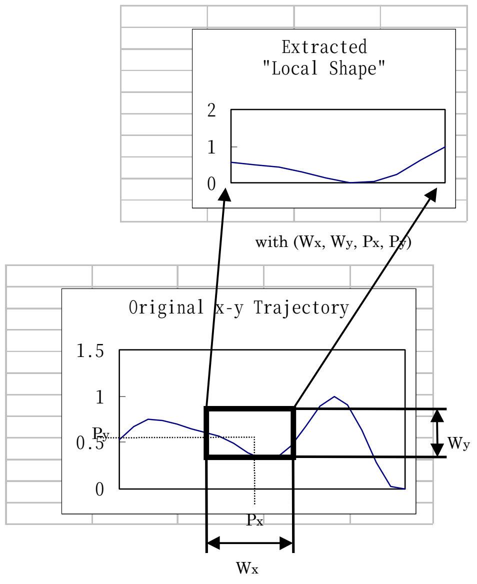


Fig.4 "Widowing" Operation: Original x-y Curve and One of Extracted Local Shape

from each of the given training data, several local shapes are extracted using "windowing" operation (see Fig.4).

The window's horizontal size is varied among several values in order to extract both quite local shapes and rather global shapes. The extracted shapes are normalized both in horizontal and vertical dimensions. By this normalization, the information of the width and height of the window and the absolute values (i.e., where the window is) will be deleted, and so such information is also attached to each shape data as other attributes. The shape data obtained in this way are called "local shapes" hereafter.

Each local shape has the information of its x-y trajectory (normalized, in a vector form), its horizontal and vertical width (Wx and Wy), and the shape's averaged position in the original trajectory data (Px and Py) and its class (which is the same as the class of its original x-y trajectory). These data are utilized for both of LVQ and generation of a decision tree.

Then, LVQ is used to generate attribute candidates. Only x-y trajectories of local shapes are used in LVQ, because the other four attributes (Wx, Wy, Px, Py) are already in a form of "attributes." A modified version of LVQ is utilized, whose algorithm is given below:

(1) Perform usual LVQ without taking the class information of data into account. The original seeds of clusters are generated by selecting randomly from the pool of the local shapes. As a result, several clusters and their "typical" vectors are generated, which will be used as the seeds in the following steps.

(2) For each cluster, calculate how many data are coming from each class, and attach the name of the majority class to the cluster.

(3) If the cluster can be said to consist of several classes (i.e., the number of the second (or third, fourth, etc.) majority class data exceeds a certain threshold), then another seed is generated at the averaged position of the data belonging to this cluster and coming from this class. Attach the class name to this new seed, too.

(4) LVQ is performed again using the seeds generated in (2) and (3), but this time the following update rule is used:

For each local shape given as a training data (named $\underline{x}$),

(4-1) Find out the nearest cluster to $\underline{x}$ from among those clusters which have the same class as $\underline{x}$.
(4-2) Update the vector of the selected cluster using the normal LVQ algorithm.

The generated typical vector of each cluster is called "a Typical Shape Vector (TSV)" hereafter.

(5) Calculate one sigma of each of the vector elements within each cluster. This information will be used later for shape matching (concretely, if a shape is within one sigma from a certain TSV, then the shape is decided to match with this TSV).

The last process is the generation of a decision tree. Please note that we want to discriminate the original x-y trajectories, not the local shapes generated by windowing process from these trajectories. The attributes to be used are the local shape and its other four attributes (Wx, Wy, Px, Py). The decision formula at each node in the tree has the following schema:

IF the x-y trajectory has such a local shape which is matched with a certain TSV,

or

IF the x-y trajectory has such a local shape which is matched with a certain TSV and this local shape's Wx (or Wy, Px, Py) > (or <) a certain value

As the criterion for deciding the best decision formula at each node, information gain is utilized. (same as ID3 or C4.3[7]) A tip node is not extended any more if the ratio of the number of the majority class data belonging to the node exceeds a certain threshold (in the following experiments, 0.95 has been found to be the best number).

5.3 Experimental Results

As the first experiment, we try to discriminate the x-y curves of mathematic functions of three different classes:

(class 1) *Combinations of (up to three) linear functions*

(class 2) *Combinations of (up to three) quadratic functions*

(class 3) *Combinations of (up to two) sinusoidal functions*

Examples of x-y curves of class 2-3 is given in Fig.5.

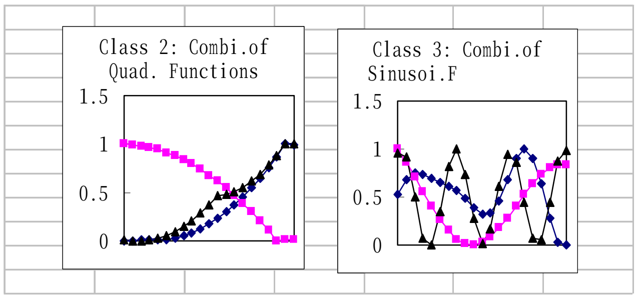


Fig.5 Examples of x-y Curves of Class 2 and 3

A hundred x-y curves are generated randomly for each class, from which about 3800 local shapes are extracted by windowing. The horizontal widths (Wx) of the local shapes are set at 1/4, 1/2, 1/1 of the original data. The vertical axis of the original x-y trajectories as well as local shapes are normalized between 0 and 1.

The number of original seeds for LVQ is 30. There may be some sensitivity of the performance to this number, but not studied here. At the step (3) of the above algorithm, the number of second (or third, fourth, etc.) majority class data exceeds:

(total number of local shapes) / (number of clusters× 2.0)

then a new seed is generated for the specific class.

The generated decision tree is finally evaluated in terms of how correctly it infers the right class, using the x-y trajectories different from those used in the above learning process.

In LVQ process, total 45 clusters are formed. Fig.6 shows the features used in the decision tree, which are the final output of "attributes useful for classification task". The size of the tree is 16 nodes, which seems reasonable number to discriminate 300 data. The local shapes shown in Fig.6 are used in the decision tree, and so can be said important features for classification. Observing these shapes and how these shapes are used to discriminate the classes, we can say that the system autonomously generates and selects such primitive features as can be said important intuitively.

The classification capability of the generated decision tree is evaluated using the x-y trajectories of three classes also generated using random variable. The training data and these test data do not overlap. Table 4 shows how correctly the tree can infer the class of given x-y trajectories. The lower part shows the relationships between the true classes and the inferred classes. For example, the tree can infer the class 1 with 99% correctness when the true class is class 1, and 1% of class 1 data is miss-identified as class 3. It is observed that the class 1 can almost perfectly be detected, and there are the most miss-classification between class 2 and class 3, which coincides with our intuition.

We plan to apply this method for generation of attributes useful for fault diagnosis task.

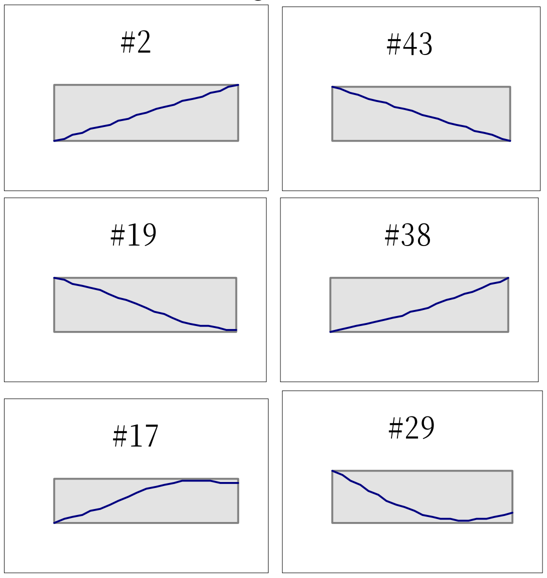


Fig.6 Generated Attributes Useful for Classification

Table 4 Classification Results Using the Generated TSV and Decision Tree

Number of Test data		3000
Classification Accuracy		92.65 %
Error (in %)	1 → 2	0.0
	1 → 3	1.0
	2 → 1	2.0
	2 → 3	9.0
	3 → 1	3.5
	3 → 2	6.5

6. Conclusions and Future Plan

A model based diagnosis system has been described and its performance is evaluated. The proposed autonomous attribute generation method is now being incorporated into the current fault diagnosis system to enhance its capability to measure the similarity of telemetry data.

The ultimate target of our autonomy research project is to construct an integrated system in which various autonomous functions, such as fault diagnosis, reconfiguration, task planning and scheduling are performed by model based inference using a common model of the target satellite in different ways (Fig. 7). In this architecture, too, we would like to avoid inclusion of human expertise as much as possible, and strategic knowledge useful for problem solving (such knowledge for reducing the search space, etc) should be compiled from the common satellite model and domain knowledge. By using a common satellite model, we can manage the change of the satellite design, subsystem design or operational procedure in a unified way.

As the first step, we have begun to study about the content and representation scheme for the common model, by trying to divert the model dedicated for the fault diagnosis task to planning task to find out what contents are missing and how the representation should be modified.

References

[1] D.Bernard and B.Pell, Designed for Autonomy: Remote Agent for the New Millenium Program, Proceedings of i-SAIRAS, pp.51-56, 1997

[2] P.Rossomando, The Achievement of S/C Autonomy through Thematic Application of Multiple Cooperating Intelligent Agents, NASA-CP-3141, 1992

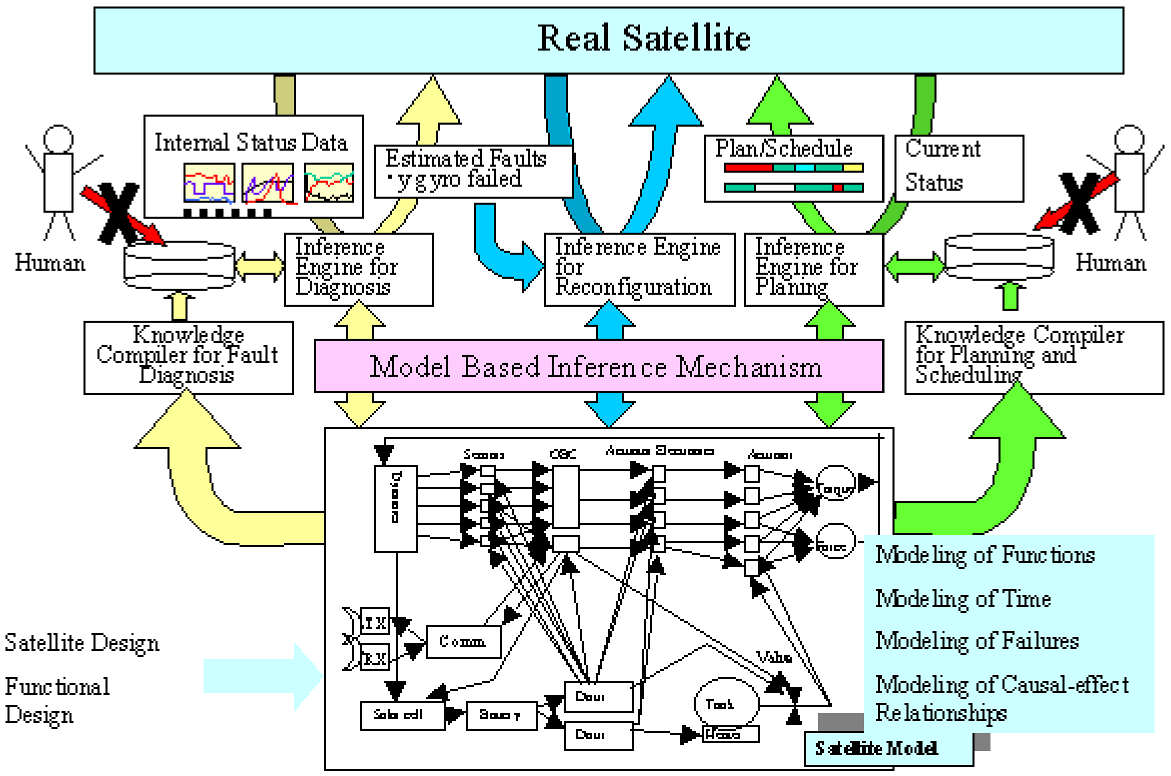
[3] N.Muscettola, C.Fry, K.Rajan, B.Smith, S.Chien, G.Rabideau, D.Yan, On Board Planning for Autonomous Spacecraft, Proceedings of i-SAIRAS, pp.229-234, 1997

[4] T.Kohonen, Self-Organization and Associative Memory, Springer-Verlag, 1987.

[5] H.Ritter,, T.Martinetz and K.Schulten, Neural Computation and Self-Organizing Maps, Addison Wesley, 1992.

[6] S.I.Gallant, Neural Network Learning and Expert Systems, The MIT Press, 1993.

[7] S.Nakasuka, and T.Koishi, Automated Extraction of Attribute Hierarchies for an Improved Decision-tree Classifier, Engineering Applications of Artificial Intelligence, Vol.8, No.4, pp.391-399, 1995.



Real Satellite
Internal Status Data
Estimated Faults
・y gyro failed
Plan/Schedule
Current Status
Human
Inference Engine for Diagnosis
Inference Engine for Reconfiguration
Inference Engine for Planing
Human
Knowledge Compiler for Fault Diagnosis
Model Based Inference Mechanism
Knowledge Compiler for Planning and Scheduling
Comm
Valve
Modeling of Functions
Modeling of Time
Modeling of Failures
Modeling of Causal-effect Relationships
Satellite Model
Satellite Design
Functional Design

Fig.7 Architecture for Integrated Autonomy Based on Common Model