Probabilistic Failure Isolation for Cognitive Robots

Dogan Altan and **Sanem Sariel-Talay**
Artificial Intelligence and Robotics Laboratory
Department of Computer Engineering
Istanbul Technical University, Istanbul, Turkey
{daltan,sariel}@itu.edu.tr

Abstract

Robots may encounter undesirable outcomes due to failures during the execution of their plans in the physical world. Failures should be detected, and the underlying reasons should be found by the robot in order to handle these failure situations efficiently. Sometimes, there may be more than one cause of a failure, and they are not necessarily related to the action in execution. In this paper, we propose a temporal and Hierarchical Hidden Markov Model (HHMM) based failure isolation method. These HHMMs run in parallel to determine causes of unexpected deviations. Experiments on our Pioneer 3-AT robot show that our method successfully isolates failures suggesting possible causes.

Introduction

An autonomous robot may face several failure situations during the execution of its actions to achieve a goal (Karapinar, Altan, and Sariel-Talay 2012) due to non-deterministic actions or different sources of uncertainty in physical dynamic environments. A monitoring system is pivotal in order to achieve goals robustly in the face of uncertainties. While detecting failures is one of the central problems for robust execution (Pettersson 2005; Usug, Altan, and Sariel-Talay 2012; Karapinar et al. 2013), the robot should also identify the reasons of the failure for efficient recovery. Isolation of a failure requires an inference process to find the underlying reason behind the failure. In this research, our focus is on a probabilistic failure isolation method. We address action execution failures that may arise due to hardware/sensor limitations, limited knowledge on some environmental features (Bouguerra, Karlsson, and Saffiotti 2008) or external events.

Monitoring and reasoning about failures requires interpreting data from one or more sensors (e.g., vision, force, touch, pressure). Therefore, in order to detect a failure, the robot needs to interpret the scene and apply certain reasoning tools to come up with correct conclusions. To achieve complete isolation of the failure, the robot needs to maintain a priori information on the models of failures that are likely to occur in the environment. In some cases, the reason of a failure may not be directly related to the action that is being executed. The reason of the failure may be depending on a previous action that is executed by the robot or an external event. For example, in the blocks world domain, when the base block structure is not properly formed, the execution of a stack action on the existing tower may fail, and the whole structure may be completely destroyed. In order to isolate these types of failures, a temporal reasoning model is needed. Furthermore, there may be more than one cause of a failure. To deal with such cases, a probabilistic temporal model is useful to identify the possible failure cases.

We propose a Hidden Markov Model (HMM) based isolation method to determine causes of the failures. Our method includes parallel Hierarchical HMMs (HHMMs) running for tracking modelled failure types. The HMM model provides temporal analysis of states and propagates temporal failure information over time after a deviation occurs. Multiple hypotheses are tracked at the same time to identify underlying causes in a probabilistic manner. Our main contribution lies in modelling each failure type as a distinct HHMM running in parallel considering action-failure type relations.

The rest of the paper is organized as follows. First, model-based fault isolation methods are reviewed. Then, our proposed method is explained in detail. Afterwards, experimental results on a number of case scenarios are presented. Then, the paper is ended with concluding remarks.

Related Work

Failure detection and isolation (diagnosis) is an intensively investigated issue for robot systems (Pettersson 2005; Fritz 2005) due to the need for safe plan execution. Monitoring can take place both in plan level and action level. A common approach to detect failures is using an observer-based approach (Pettersson 2005). In this approach, predefined models and inconsistencies between the expected and observed outcomes are used to detect failures (Nan, Khan, and Iqbal 2008; Steinbauer and Wotawa 2009). Uncertainties may arise over the reasons of failures. This is handled with semantic-knowledge based execution monitoring where the robot estimates a probability distribution according to its expectations (Bouguerra, Karlsson, and Saffiotti 2007).

Model-based failure diagnosis has been investigated by many researchers previously (Frank, Ding, and Marcu 2000). Structural abstraction is used for model-based diagnosis in earlier works (Friedrich, Stumptner, and Wotawa 1999; Chittaro and Ranon 2004). In some model-based fault detection and isolation systems, HMMs are used to monitor processes (Hovland and McCarragher 1996) or to diagnose failures in different domains (Kwon and Kim 1999; Ying et al. 2000; Ocak and Loparo 2001; Ge, Du, and Xu 2004; Lee et al. 2004; Li et al. 2005). Dynamic Bayesian Networks and Particle Filters are also used for failure diagnosis (Flores-Quintanilla et al. 2005; Verma et al. 2004). In another work, a hierarchical representation is used for failure diagnosis (Verma, Simmons, and Fernandez 2002) to handle a single fault. HMMs are also used in a mobile robot system to estimate the most appropriate mode for the robot in execution (Peynot et al. 2011). Logic programming with situation calculus is studied in order to explain the unexpected deviations in task execution using hypotheses and their costs (Gspandl et al. 2012). Extended action models are also proposed in a control loop that integrates monitoring, diagnosis and recovery (Micalizio 2013).

Our HMM-based failure isolation method differs from earlier work because of its ability in recognizing failures hierarchically in parallel and its usage of relations between actions and failure types. Our system can determine that a failure, which can not be observed directly from sensory information, may be caused by multiple fault sources, and it may provide explanations for the causes of a failure if there is no clear indication.

Probabilistic Failure Isolation

Given an initial and goal state, the robot takes consecutive actions in order to reach the goal. In the symbolic level, the robot maintains the models of its operators corresponding to the $actions$ that it can execute in the real world. Each action is represented by a set of facts to be satisfied before executing it, namely, preconditions, and effects that occur in the world after the action is executed by the robot. In an object manipulation scenario, the robot's goal is to transport objects to a destination. Initially, all objects are in their initial positions. The robot is capable of executing some actions namely *move-to-loc*, *pick-up*, *put-down* and *move-to-obj*. Action *move-to-loc* is executed in order to move the robot to a location, whereas action *move-to-obj* moves the robot to a desired object location. Action *pick-up* is to pick up an object by the robot's gripper. Similarly, action *put-down* is executed for releasing an object from its gripper on the ground. Several types of failures may be faced during the execution of the actions. For example, the robot may fail in executing action *pick-up* on an object because of a wrong grasp position. Another failure may occur if the robot's vision system fails. Yet in another scenario, the object may be manipulated by another agent instantly. In such cases, the isolation model should give relevant explanations for the causes of these situations that may lead to failures.

HHMM-based Failure Isolation

Hidden Markov Models (HMMs) (Baum and Petrie 1966) are probabilistic temporal structures to model Markov processes. Since failures that occur during the execution of a plan generally propagate over time, the problem of identifying failures should be analysed temporally. Furthermore, uncertainties in sensing and non-determinism make HMM-based models suitable for the failure isolation problem.

An HMM consists of five components namely, states, observations, transition model, observation model and the initial state distribution. There are N hidden states in an HMM. Each hidden state is denoted with $s_i \in \mathcal{S}$ where $\mathcal{S}$ is the set of hidden states. Transition model, $\mathcal{A} = a_{ij}$, defines the probability of transferring from state s_i to state s_j where s_i, s_j $\in \mathcal{S}$. Observations, denoted with the y_t, represent the observations of the model gathered at time t. Observation model, denoted by $\mathcal{B} = b_{s_i}(y)$, defines the probability of gathering the observation y at state s_i. Initial state distribution probabilities are represented with $\pi = \pi_i$.

An HHMM (Fine, Singer, and Tishby 1998) is a derived structure of an HMM in which each state in the model is itself an HMM. Once a state is activated in the HHMM, its sub-HMM becomes active. Transitions can only occur inside that sub-HMM until a final state is reached in it. After then, a transition occurs to the upper parent state. Similar transitions take place to the other parent states sequentially.

In our approach (Altan and Sariel-Talay 2014), the type of a failure that may occur during the execution of an action is modelled as a distinct HHMM which is similar to the HHMM representation explained above. Parent states represent actions, and lower level HMMs corresponds to their execution states. This representation forms a two-level HHMM. Each model is denoted by M_i where i is the index of the corresponding failure type. In this representation, there are two hidden states, namely *success* and *failure*. During the execution of an action in the plan, a lower level HMM is activated under the corresponding node, and its model is updated. Once the action comes to an end, this lower level HMM is used to refine a result to the upper level of the corresponding HHMM. The problem is to find models that include latent variables labelled with *failure* and have failure probabilities over a given threshold.

Hierarchical HMMs are used to represent failure models instead of classical HMMs because a hierarchy between the plan and its actions is needed in order to isolate persistent failures on a specific object or event. For instance, the robot's vision system may fail to recognize a specific object in the environment all the time. Causes of failures that are addressed in this paper are: *vision failures for all objects*, *localization failures*, *hardware limitations (gripper)* and *external events*. *Vision(X)* failure model for a specific $object(X)$ represents a faulty situation in the vision algorithm for that object (i.e., error on detection the object itself or its relations). Model *HardwareLimitation* indicates the situations that are beyond the capabilities of the robot. *ExternalEvent* stands for external events that change the world outside the control of the robot. *Localization* failures model the faulty situations where the robot cannot localize itself. Depending on the action that is being executed at a given world state and its parameters, the related failure models are activated. The relations that define which action is related to which failure type are given in Table 1 for a specific $object(X)$.

At the very beginning of the execution of the plan, the first state for each failure type is activated considering observations in the initial state. During the execution of each action, if that action and a failure type are related, the corresponding failure type's HHMM is activated for that time step, and this model is considered as an *active model*. Perceptions are gathered from the sensors of the robot (e.g., pressure sensor, RGB-D sensor, etc.) and these observations are mapped into a probability distribution which depends on the statistical analysis of the sensory input. Depending on the observations made at the current state, the new state's contents are defined considering the previous state.

Table 1: Action-Failure relations used in the model.

Action	*Failure Type*
move-to-obj(X)	Vision(X), ExternalEvent Localization(Robot)
move-to-loc(destination)	Localization(Robot)
pick-up(X)	ExternalEvent, Gripper(Robot) Vision(X), Localization(Robot)
put-down(X)	Gripper(Robot)

If a failure model is not related to the action in execution, it is considered as a *passive model*, and the last state of the corresponding HHMM's time interval is augmented in such a way that corresponding state also includes that time step without generating a new state. This procedure goes on until the robot reaches the goal state or a failure is detected. We assume that failure detection is done by applying an observer-based approach. In case of a failure, the Viterbi algorithm (Viterbi 1967) is invoked to label latent variables with either *success* or *failure* in each HHMM.

Algorithm 1 FailureIsolation(P,M)

```
Input: Plan $P$, failure models $M$
Output: $list$, the list of the candidate causes of a failure
while $P$ != $\emptyset$ and $status$ == $success$ do
    $action = POP(P)$
    while $(status = execute(action)) == inExecution$ do
        $updateLowerLevelModels(M, action)$
    end while
    $labelLowerLevelLatents(M, t)$
    $transitToUpperLevel(M, t)$
    $updateModels(M, action)$
end while
for $all\ M_i$ do
    $labelUpperLevelLatents(M_i)$
end for
$list = isolateModels(M)$
$return\ list$
```

In order to explain the proposed solution, Algorithm 1-2 are presented. Algorithm 1 accepts a plan and failure models as parameters. Plan P includes a sequence of actions to be executed by the robot. An action can be in one of the following states: *inExecution*, *success* and *failure*. *inExecution* corresponds to the state in which the robot executes that action. *success* stands for the state that the robot reached the expected outcomes of the corresponding action. *failure* corresponds to the situations where the expected outcomes of an action are not met. During the execution of each action, all HHMM-structured models' lower levels related to the corresponding failure types are considered as *active models* and updated by the *updateLowerLevelModels(M,action)* function where M is the set of HHMM failure models. After execution of the corresponding action, a transition to the upper level occurs with the *transitToUpperLevel(M,t)* function by using the maximum failure probability in the related model's lower level for the corresponding action's time interval. *updateModels(M,action)* updates all HHMMs' upper levels considering the action-failure type relations. The content of this routine is explained in Algorithm 2. *labelUpperLevelLatents(M_i)* procedure assigns the hidden values of each state in each model in the upper levels while *labelLowerLevelLatents(M, t)* assigns the values of the lower level at time t using the Viterbi algorithm (Viterbi 1967). *isolateModels(M)* procedure presents a list of possible causes among the failure models considering calculated probabilities in case of a failure.

Algorithm 2 updateModels($M, action$)

```
Input: Failure models $M$, current $action$
Output: $M_i$ is updated according to observations
for $all\ M_i$ do
    if $isRelated(M_i, action)$ then
        $addNewState(M_i)$
    else
        $extendPreviousState(M_i)$
    end if
end for
```

Algorithm 2 updates all related models according to the action in execution and its outcomes. In order to infer this relation, the predefined action-failure type relations and the parameters of the executed action are used. In an *active model* case, a new state is generated, and the contents of the newly generated state are updated. Otherwise, the previous state's time interval is extended.

Consider the given scenario in Figure 1. All models are initialized in the beginning. Note that in Figure 1, only vision failure model for object A, hardware limitation (particularly for the gripper) for the robot and localization failure models are shown. After executing *move-to-obj(A)* action, the localization failure model and vision failure model for object A are updated since they are related to the action. The next action to be executed in the plan is action *pick-up(A)*. Since this action is related to the vision, hardware limitation and localization models, they are updated during the execution. However, vision failure model for object B is not updated since the executed action is irrelevant for the object B. At the end of each action, the corresponding lower level HMM is finalized, a transition is taken, and the upper level HMM is updated.

Experimental Results

The proposed method is tested on our Pioneer 3-AT robot equipped with a gripper to manipulate objects. The robot completes its sense-plan-act cycle in a completely autonomous way. The world state of the robot is maintained by using sensory and motor information. The on-board RGB-D camera input is processed by a modified version of

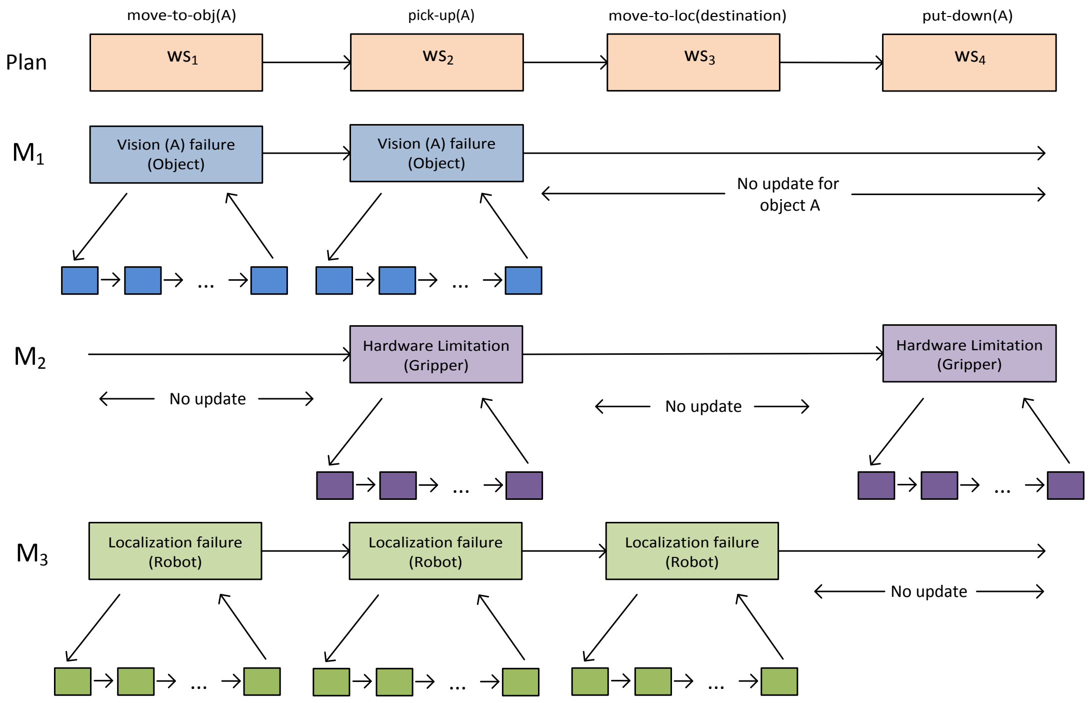

move-to-obj(A)
pick-up(A)
move-to-loc(destination)
put-down(A)
Plan
WS$_1$
WS$_2$
WS$_3$
WS$_4$
M$_1$
Vision (A) failure (Object)
Vision (A) failure (Object)
No update for object A
M$_2$
No update
Hardware Limitation (Gripper)
No update
Hardware Limitation (Gripper)
M$_3$
Localization failure (Robot)
Localization failure (Robot)
Localization failure (Robot)
No update

Figure 1: An example model for a manipulation scenario.

the LINE-MOD algorithm (Ersen, Sariel-Talay, and Yalcin 2013). Furthermore, a segmentation algorithm is executed to find clusters in the point cloud of the scene. A scene interpretation module maintains relevant predicates in the knowledge base, and updates them according to new observations (Ozturk et al. 2014). Experiments take place in our laboratory environment that contains objects placed on the ground. The experiments are performed on three objects: a blue plastic bowling pin, a beach volleyball ball, and a small plastic purple ball. To analyse some failure cases, failures are manually injected to the settings except from vision failures.

Observation probabilities are defined regarding the statistical analysis on the outputs of the sensors. Conditional Probability Table (CPT) for the vision failures is determined by using the performance of recognition and segmentation algorithms. The CPT for the localization model is created considering the offset difference between the odometer-based and feature-based localization of the robot. External event failures' CPT is defined regarding the outputs of the scene interpretation module. For hardware (gripper) failures, the pressure sensor outputs of the gripper are used for constructing its CPT. Observation sampling and failure model update frequency is at 0.33 Hz.

Scenario I In this scenario, our robot is tasked to move the pin (A) and the beach ball (B) to their desired destinations. Although the robot can correctly localize the pin, it fails in correctly recognizing the ball. Since the vision algorithm searches the objects by considering their surface normals, it falsely detects a small ball on the left boundary of the ball (Figure 2) due to the similarity of the objects' surface normals. This affects the parameters of the action *pick-up*. Therefore, the robot fails in picking up the ball with its gripper. After the failure, the robot is intentionally allowed to retry the action two more times. Since the pin is successfully recognized, action *pick-up* is taken without any failure.

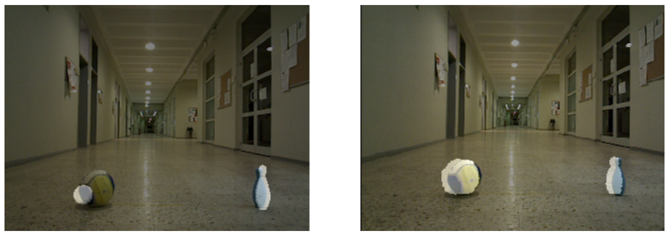
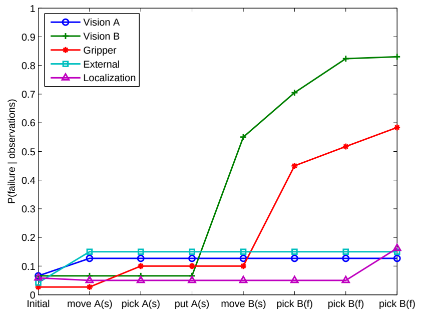


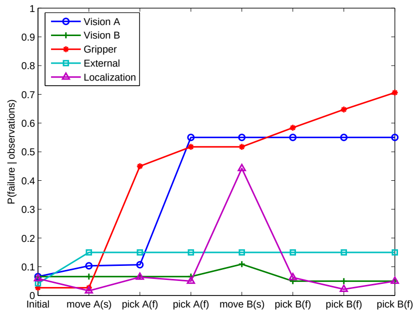
Figure 2: (left) The vision algorithm detects a small ball on the boundary of the big ball. (right) The segmentation algorithm correctly clusters the point cloud of the big ball.

Figure 3(a) illustrates how the posteriors on failure types change during the execution of each action. Five failure types are considered. As clearly seen from the figure, during the execution of action *move-to-obj(A)* and *pick-up(A)*, vision failure is not observed, therefore, the probabilities are stable during these time steps. The robot directly tries to put down the object after it picks up. Note that in the plots, letters s and f next to the actions are for stating that the action is succeeded or failed, respectively.

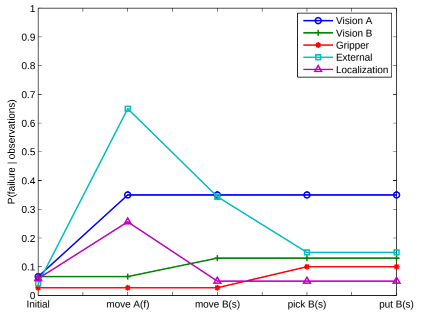
The vision failure probability for B starts to increase after the execution of action *move-to-obj(B)* due to the conflicts



(a) A consistent vision failure for a particular object results in an increase in the corresponding probability.



(b) A hand-made gripper failure is injected into the system. Probabilities change accordingly.



(c) The resulting probabilities when an external event exists in the scene.

Figure 3: Failure probabilities are continually updated in each plot during the execution of actions on the objects.

in the results of the vision algorithm and the detected sizes of the object by segmentation. Note that even when there is this type of vision failure, it is possible to successfully pick up the object if the center of recognition intersects with the center of the original object, but this is not the case in this scenario. It is also seen from the figure that while the robot attempts to pick up B, probability of having a gripper failure increases due to the propagation of the failure over time.

Scenario II To test the performance of the system to detect a hardware malfunction, a hand-made gripper failure is injected to the robot. This is simulated by preventing the gripper from closing completely and by closing the sensory feedback to detect the state of the gripper.

The robot tries to pick up the plastic pin (A) and the small ball (B) but fails all the time for both objects due to the simulated failure in its gripper. According to the Figure 3(b), the robot's belief on gripper fault increases for each trial. Note that the robot's belief on vision failure for A increases since the vision algorithm locates two objects simultaneously within the same location with different similarity rates. During the execution of action *move-to-obj(B)*, the robot's belief on a localization failure is also increased due to the offset difference between the odometer-based and feature-based localization of the robot, but it is fixed in future steps. During the execution of action *move-to-obj(B)*, the robot's belief on a vision failure on B also increases since the vision algorithm detects the object with a lower similarity rate regarding its template during that period.

Scenario III In this scenario, failures due to external events are analysed. For this purpose, the object is manipulated by a human agent. Two objects (the pin and the small ball) are used and the pin (A) is taken out of the environment by the human while the robot moves to that object.

According to Figure 3(c), the robot increases its belief on an external event when the object disappears from the robot's view. The segmentation output also suggests this hypothesis. However, the vision failure for A is also in one of the likely hypotheses. One should also note that the posterior probability of external event model decreases after the robot starts to execute actions on the other object. Since their models for this failure type are not separated and the other object is always on the sight of the robot during the execution, the posterior starts to drop off.

Analysis of Memory Usage In all the investigated scenarios, we analyze the size of the state space. We show that use of action-failure type relations reduces the number of states generated in our method (Table 2). In the table, the number of states with and without using relations (Table 1) and the percentage of gain in terms of memory utilization are presented for each scenario. The overall memory reduction is 37.7% for these scenarios.

Table 2: Analysis of State Generation on HHMMs

Scenario	# of States w/o relations	# of States w/ relations	Gain
Case I	242	146	39.66%
Case II	258	173	32.94%
Case III	153	91	40.52%

Conclusion

In this paper, we present a temporal model for failure isolation that employs HHMMs. HHMMs are for modelling the possible failure types. Moreover, HHMMs ensure isolation of multiple faults and propose explanations for possible failure situations. Hierarchical structure of the HMM also provides the opportunity to isolate persistent failures on a specific object or event. Modelling each failure type as a distinct model provides a simple yet effective representation. Updating models using the relations between the actions and the failure types reduces the computational cost of the method. As a future work, we are planning to use isolation results to modify future plans of the robot after a failure occurs.

Acknowledgements

This research is funded by a grant from the Scientific and Technological Research Council of Turkey (TUBITAK), Grant No. 111E-286. Authors would also like to thank Mustafa Ersen, Melis Kapotoglu, Cagatay Koc and Melodi

Deniz Ozturk for their contributions on the design of the robot software system.

References

Altan, D., and Sariel-Talay, S. 2014. Hierarchical HMM-based failure isolation for cognitive robots. In *Proceedings of the 6th International Conference on Agents and Artificial Intelligence (ICAART)*.

Baum, L. E., and Petrie, T. 1966. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics* 37(6):pp. 1554–1563.

Bouguerra, A.; Karlsson, L.; and Saffiotti, A. 2007. Handling uncertainty in semantic-knowledge based execution monitoring. In *IROS*, 437–443.

Bouguerra, A.; Karlsson, L.; and Saffiotti, A. 2008. Monitoring the execution of robot plans using semantic knowledge. *Robotics and Autonomous Systems* 56(11):942–954.

Chittaro, L., and Ranon, R. 2004. Hierarchical model-based diagnosis based on structural abstraction. *Artificial Intelligence* 155(12):147 – 182.

Ersen, M.; Sariel-Talay, S.; and Yalcin, H. 2013. Extracting spatial relations among objects for failure detection. *on Visual and Spatial Cognition* 13.

Fine, S.; Singer, Y.; and Tishby, N. 1998. The hierarchical hidden markov model: Analysis and applications. *Machine Learning* 32(1):41–62.

Flores-Quintanilla, J.; Morales-Menendez, R.; Ramirez-Mendoza, R.; Garza-Castanon, L.; and Cantu-Ortiz, F. 2005. Towards a new fault diagnosis system for electric machines based on dynamic probabilistic models. In *Proceedings of the 2005 American Control Conference*.

Frank, P. M.; Ding, S. X.; and Marcu, T. 2000. Model-based fault diagnosis in technical processes. *Transactions of the Institute of Measurement and Control* 22(1):57–101.

Friedrich, G.; Stumptner, M.; and Wotawa, F. 1999. Model-based diagnosis of hardware designs. *Artificial Intelligence* 111(12):3 – 39.

Fritz, C. 2005. Execution monitoring – a survey. Technical report, University of Toronto.

Ge, M.; Du, R.; and Xu, Y. 2004. Hidden markov model based fault diagnosis for stamping processes. *Mechanical Systems and Signal Processing* 18(2):391 – 408.

Gspandl, S.; Podesser, S.; Reip, M.; Steinbauer, G.; and Wolfram, M. 2012. A dependable perception-decision-execution cycle for autonomous robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2992–2998.

Hovland, G. E., and McCarragher, B. J. 1996. Hidden markov models as a process monitor in robotic assembly.

Karapinar, S.; Altan, D.; and Sariel-Talay, S. 2012. A robust planning framework for cognitive robots. In *Proceedings of the AAAI-12 Workshop on Cognitive Robotics (CogRob)*.

Karapinar, S.; Sariel-Talay, S.; Yildiz, P.; and Ersen, M. 2013. Learning guided planning for robust task execution in cognitive robotics. In *The AAAI-13 Workshop on Intelligent Robotic Systems*.

Kwon, K.-C., and Kim, J.-H. 1999. Accident identification in nuclear power plants using hidden markov models. *Engineering Applications of AI* 12(4):491 – 501.

Lee, J. M.; Kim, S.-J.; Hwang, Y.; and Song, C.-S. 2004. Diagnosis of mechanical fault signals using continuous hidden markov model. *Journal of Sound and Vibration* 276(35):1065 – 1080.

Li, Z.; Wu, Z.; He, Y.; and Fulei, C. 2005. Hidden markov model-based fault diagnostics method in speed-up and speed-down process for rotating machinery. *Mechanical Systems and Signal Processing* 19(2):329 – 339.

Micalizio, R. 2013. Action failure recovery via model-based diagnosis and conformant planning. *Computational Intelligence* 29(2):233–280.

Nan, C.; Khan, F.; and Iqbal, M. T. 2008. Real-time fault diagnosis using knowledge-based expert system. *Process Safety and Environmental Protection* 86(1):55 – 71.

Ocak, H., and Loparo, K. 2001. A new bearing fault detection and diagnosis scheme based on hidden markov modeling of vibration signals. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)*, volume 5, 3141–3144.

Ozturk, M. D.; Ersen, M.; Kapotoglu, M.; Koc, C.; Sariel-Talay, S.; and Yalcin, H. 2014. Scene interpretation for self-aware cognitive robots. In *AAAI-14 Spring Symposium on Qualitative Representations for Robots*.

Pettersson, O. 2005. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems* 53:73–88.

Peynot, T.; Fitch, R.; McAllister, R.; and Alempijevic, A. 2011. Autonomous reconfiguration of a multi-modal mobile robot. In *Workshop on Automated Diagnosis, Repair and Re-Configuration of Robot Systems, IEEE International Conference on Robotics and Automation (ICRA)*.

Steinbauer, G., and Wotawa, F. 2009. Robust plan execution using model-based reasoning. *Advanced Robotics* 23(10):1315–1326.

Usug, U. C.; Altan, D.; and Sariel-Talay, S. 2012. Robots that create alternative plans against failures. In *10th IFAC Symposium on Robot Control*.

Verma, V.; Gordon, G.; Simmons, R.; and Thrun, S. 2004. Real-time fault diagnosis [robot fault diagnosis]. *Robotics Automation Magazine, IEEE* 11(2):56–66.

Verma, V.; Simmons, R.; and Fernandez, J. 2002. Probabilistic models for monitoring and fault diagnosis. In *The Second IARP and IEEE/RAS Joint Workshop on Technical Challenges for Dependable Robots in Human Environments*, 43–67.

Viterbi, A. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on* 13(2):260–269.

Ying, J.; Kirubarajan, T.; Pattipati, K.; and Patterson-Hine, A. 2000. A hidden markov model-based algorithm for fault diagnosis with partial and imperfect tests. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 30(4):463–473.