Using Qualitative and Model-Based Reasoning for Sensor Validation of Autonomous Mobile Robots

Gerald Steinbauer [*] **Alexander Kleiner** [**] **Franz Wotawa** [*]

[*] *Institute for Software Technology, Graz University of Technology, Inffeldgasse 16b/2, A-8010Graz, Austria, {steinbauer,wotawa}@ist.tugraz.at*
[**] *Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Georges-Köhler-Allee, D-79110 Freiburg, Germany, kleiner@informatik.uni-freiburg.de*

Abstract:
Mobile robots performing tasks in dynamic environments have to rely on both reliable sensor data and data processing. Since actions performed by robots in the real world are based on this information, uncertain or even wrong data can be fatal. For instance, a compass may be significantly disturbed by metal structures in the environment. Truly autonomous robots must be able to cope with such situations.
In order to enhance robotic systems with this capability we propose a sensor validation approach. The key idea is to qualitatively compare trends of sensors which are related by some physical constrain. These observations together with an abstract model of these relations are utilized by a model-based reasoning approach for finding the root cause of data unreliability. The result of the reasoning process can be used online on the robot platform to adapt its sensor data processing in order to mitigate the effect.

1. INTRODUCTION

Mobile robots performing tasks in dynamic environments have to rely on both reliable sensor data and data processing. Actions performed by mobile robots in the real world are typically based on high-level features generated from sensor data or from the sensor data directly. Such features can be, for example, the current location of the robot in the environment, positions of relevant objects, and any other important fact of the world, such as whether a door is open or closed. The number of different sensors for mobile systems and the number of algorithms to process their data constantly grows. Usually all these algorithms perform sensor fusion. Sensor fusion combines information from different sensors with different modalities in order to determine the most likely state of the world. Actually, most of these algorithms are based on probabilistic models in order to cope with sensor uncertainty and ambiguity. For an overview see citethrunbook.

For an autonomous mobile robot it is crucial being able to cope with uncertain or ambiguous sensor readings. For example, data from GPS sensors can be erroneous due to weather conditions or even drop out because of signal path propagation reflected by buildings or trees. Probabilistic sensor fusion methods can cope with such situations to some extend and for a limited amount of time. However, if erroneous readings drastically increase, for example, due to a significant change in the environment, fusion algorithms tend to wrong interpretations. Within dangerous situations, this might have fatal consequences.

Sensor validation is a common approach to minimize the effect of uncertain or faulty sensors The idea is to use a validation mechanism for determining the reliability of a single sensor or the output of a sensor fusion algorithm by performing a plausibility check on the data. The result of the validation allows a meta mechanism executed on the robot to actively deal with the problem.

The reaction of the meta mechanism can be twofold. The mechanism may simply ignore the output of the sensor data processing. The mechanism can also react in a more intelligent way by using the information gained in the validation process and reasoning about the true root cause of the problem. Reasoning techniques, like model-based diagnosis, allow the localization of malfunctioning parts in the system. Once erroneous parts of the system are identified, counter measures can be derived in order to stabilize sensor processing. If the root cause is detected, involved sensors can be ignored for some time period or their uncertainty measure increased. The approach works effective if an outages can reliably be detected, such as faulty GPS data cased by occlusion, or distortion of compass measurements caused by metal structures. Moreover, such interferences are usually limited to a certain period of time and can usually be compensated by other sensor or sensor modalities, as for example, replacing GPS readings with readings from the wheel odometry.

In order to equip an autonomous mobile robot with the capabilities to actively cope with such situations, we propose an approach based on qualitative reasoning and model-based diagnosis. The approach is an extension to the diagnosis system we developed for the control software of autonomous mobile robots, based on monitoring communication pattern of software modules Kleiner et al. [2008]. The basic idea of the extension is to observe trends in the data from sensors or sensor fusion algorithms. Usually there can be some relations found in the output of different sensors or algorithms. For example, two outputs can be proportional, or one output is the integration or differentiation of another. These relations are fixed by physics or by some algorithms in the control software of the robot. For instance, the differentiation of the yaw measured by a compass is direct proportional to the angular velocity measured by a gyro of an inertial measurement unit (IMU).

A direct absolute quantitative comparison of the trends in different data streams will fail because of the noise in the data. The idea is to use techniques of qualitative reasoning (QR) and compare only a qualitative statement on the trends. A faulty behavior is observed if the relation of two data stream is in contradiction with the desired relation. Such observations together with a logic model of the relations of the different data streams are used to reason about the root cause of the undesired behavior. This reasoning can be done using model-based diagnosis (MBD). Once the root cause, e.g., a faulty or disturbed sensor, is located, the meta mechanism decides to ignore data from the particular sensor, or to adapt the sensor fusion algorithm.

The paper continues with a running example, which explains how such a sensor validation can be implemented on a real autonomous robot. In Section 3 we depict the trend observation using QR. The next section depicts how the relations can be modeled and how diagnosis can be done with such models. In Section 5 preliminary results of the approach on a real robot system are shown. The last section draws some conclusions and outlines future work.

2. SENSOR VALIDATION ON A RESCUE ROBOT

As a running example for our sensor validation approach we use the Telemax rescue robot of the University of Freiburg. It is a very robust robot equipped with tracks, flippers and a manipulation arm and is originally used for bomb disposal by security forces. The robot is depict in Figure 1.

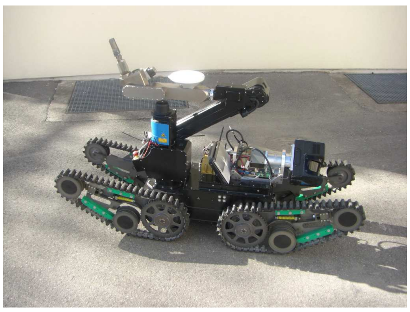

Fig. 1. The Telemax rescue robot.

For research in simultaneous localization and mapping (SLAM) in the area of urban search and rescue (Kleiner and Dornhege [2007]) the robot is equipped with several sensors: a GPS receiver, an inertial measurement unit (IMU), odometry, a 2D laser scanner and a 3D laser scanner. The task of the robot is to automatically explore a disaster area and to generate a map of that area.

The robot uses the data from the gyroscope of the IMU, the compass of the IMU, the odometry of the robot and the 2d laser scanner in order to generate a map of the environment and to localize itself in that map. Moreover, the robot uses the data of the 2d laser scanner and scan matching techniques in order to provide an improved odometry (Lu and Milios [1997]).

The different sensors and sensor modalities provide different measurements of the robot motion in its environment. The compass of the IMU provides the absolute heading against north. The gyroscope of the IMU provides an estimation of the angular velocity around the vertical axis. The odometry provide information about the distance traveled and the angle turned by the robot. The sensors and its modalities are summarized in Table 1

Sensor	**Modality**	**Unit**	**Symbol**
GPS Receiver	Global Position	m	$< x_{GPS}, y_{GPS} >$
IMU Compass	Yaw	rad	Θ_{IMU}
IMU Gyroscope	Angular Velocity	rad/s	Ω
Odometry	Distance Traveled	m	d_{ODO}
Scan Matching	Distance Traveled	m	d_{SM}
Scan Matching	Yaw	rad	Θ_{SM}

Table 1. The sensors of the rescue robot and its modalities.

Based on simple physical considerations we can obtain the following relations between different sensor streams. These relations have to hold if the system and all its sensors show the correct behavior.

The derivation of the measurements of yaw of the compass is proportional the velocity measurements of the gyroscope ($\frac{d}{dt}\Theta_{IMU} \sim \Omega$). The physical anchor of that relation is the turning velocity of the robot. The derivation of yaw of the scan matching is proportional the gyroscope ($\frac{d}{dt}\Theta_{SM} \sim \Omega$). The derivations of the measurements of yaw of the compass and the scan matching are proportional ($\frac{d}{dt}\Theta_{IMU} \sim \frac{d}{dt}\Theta_{SM}$).

In order to supervise these relations we use observers based on qualitative reasoning techniques. For details see Section 3. The idea is to compare the trends (derivation) of the data streams in a qualitative manner. Simply spoken if data stream A is related to data stream B both streams have to show a similar qualitative behavior. For instance if the values in data stream A increase also the values in data stream B have to increase. The observer does not operate on absolute values but on abstracted trends. The observer for different relations raises an alarm if the behaviors of the two data stream significantly differ.

Such observers allow us to supervise the system. If one or more observer raise an alarm it is an indicator that some parts of the system do not behave as expected. Using an abstract logical model of the relations together with these observations allows the robot to reason what is the root cause of an observed problem. We use a consistency-based diagnosis approach.

Once the robot localized the problem, e.g., a permanently or sporadically malfunctioning sensor, it has to automatically react on that circumstance. One solution is to forward this information to the high-level decision making system in order to allow the robot to derive an alternative plan. In our approach we propose that the robot use this information of the sensor validation to decide if the data of a sensor or algorithm are reliable enough to trust them. If this is not the case the robot has the possibility to suppress such unreliable sensors in order to improve the performance of the overall sensor data processing. If the disturbance of a sensor is only temporary, e.g., GPS blocked by a tree, this fact is also recognized and the robot is able to reuse the sensor once the problem has disappeared.

3. SENSOR MONITORING

In order to supervise a data stream from a sensor or sensor fusion and to determine mismatches in the relation of two data streams we use a set of observers. An observer takes two data streams, determines the qualitative trends in both streams and evaluates if there is a significant discrepancy in these trends.

In a first step the observer determines the qualitative trend of two data streams. The gradient of the data is mapped to three qualitative symbols $[-, 0, +]$. The symbol $-$ means that the values in the stream decrease. The symbol $+$ means that the values in the stream increase. The symbol 0 means that there is no significant change in the data. The Algorithm 1 depicts a method to determine the qualitative trend.

Algorithm 1 $abstractStream(S, b, w)$

```
r1 = deriveStream(S, w)
r2 = deriveStream(r1, w)
r3 = deriveStream(r2, w)
a[1] = 0
for i=2 to |S| do
    if r1[i] < −b then
        a[i] = −
    else
        if r1[i] > b then
            a[i] = +
        else
            if ¬((−b < r2[i] < b) ∧ (−b < r3 < b)) then
                a[i] = 0
            else
                a[i] = a[i − 1]
            end if
        end if
    end if
end for
return a
```

The algorithm takes a data stream S and a boundary b and returns a stream of qualitative symbols representing the trend in the stream S. b is a parameter, which determines the gradient, which is used as a classifier for the qualitative trends.

In a first step the algorithm calculates the first, second and third derivative of the data stream. The calculation of the derivative is done using Algorithm 2. It calculates a linear regression for a sliding window on the data stream. The parameter w determine the width of the window. The window is used to suppress noise and peaks in the data. Then each local gradient is classified by a comparison with the boundary b. If the gradient is below $-b$ or above $+b$ the gradient is classified as $-$ or $+$ respectively.

If the gradient lies with in the interval $[-b, b]$ the algorithm also evaluates the higher derivations in order to detect minima, maxima or saddle points in the stream. If no classification is applicable so far no statement about the trend can be given and the algorithm prolongs the trend of the last time stamp. This approach is similar to a approach proposed by Aichernig et al. [2009].

Algorithm 2 $deriveStream(S, w)$

```
for i=1 to |S| do
    start = min(⌈i + w/2⌉, |S|)
    end = max(1, ⌊i − w/2⌋)
    d[i] = linearRegression(S[start, end])
end for
return d
```

Figure 2 depicts the results of the abstraction process for the yaw measurements by the compass of the IMU during a mapping mission.

In order to monitor whether two related data stream differ significantly in their qualitative trends we use Algorithm 3. The algorithm compares the trends of two data streams S_1 and S_2. If there is a mismatch a counter is incremented otherwise the counter is decremented. The counter cannot fall below zero. If the counter exceeds a given threshold th the observer raises an alarm represented by the constant no_match. The counter and the threshold are used in order to guarantee that the observer only raises an alarm if a significant mismatch occurs e.g., a mismatch for a longer period of time.

Algorithm 3 $observe(S_1, S_2, th)$

```
counter = 0
for i=1 to |S1| do
    if S1[i] ≠ S2[i] then
        count = counter + 1
    else
        counter = max(0, counter − 1)
    end if
    if counter > th then
        o[i] = no_match
    else
        o[i] = match
    end if
end for
return o
```

Figure 3 depicts the results of the observer process for the trends of the yaw of the scan matching algorithm and the gyroscope of the IMU.

4. MODELING AND MODEL-BASED DIAGNOSIS

Given the information obtained from the observers, which compare the outcome of the different sensors in a qualitative manner, we are interested in finding preferable unique causes for mismatches. Model-based and in particular consistency-based diagnosis provides a foundation

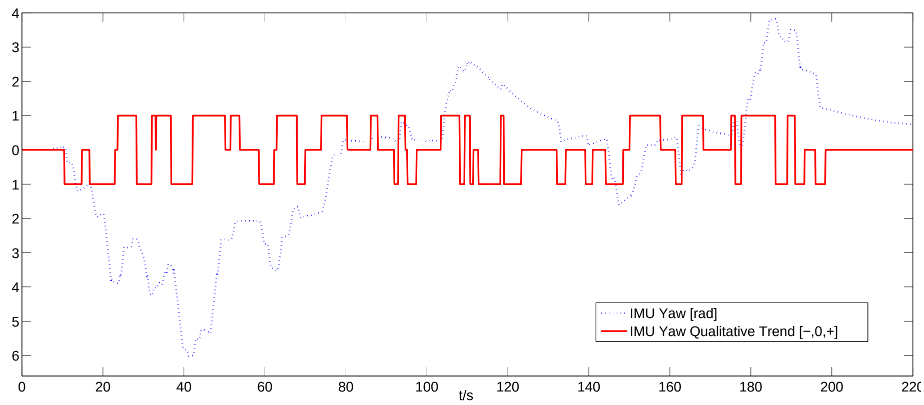

Fig. 2. Qualitative trends [-,0,+] of the yaw measurements by the compass of the IMU.

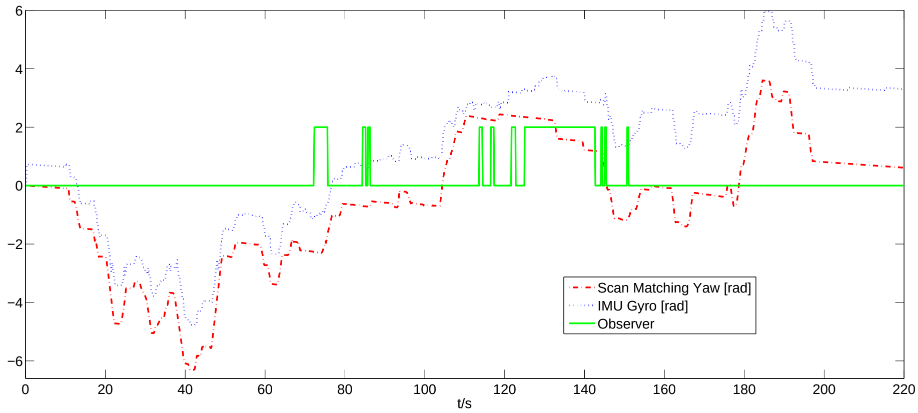

Fig. 3. Observation of mismatches of trends of the yaw of the scan matching algorithm and the gyroscope of the IMU.

for this purpose. In this paper we follow Reiter's approach (Reiter [1987]) and define the diagnosis problem as tuple $(SD, COMP, OBS)$ where SD is a logical description of the model, $COMP$ is a set of components that might cause undesired effects, and OBS is a set of observations. In our particular application SD describes the relationship between sensor information and the comparison results obtained from the observer, $COMP$ is the set of sensors, and OBS is a set of predicates that are provided by the observer, which hold in a certain point of time. According to Reiter [1987] a diagnosis is a set $\Delta \subseteq COMP$ where the logical sentence $SD \cup OBS \cup \{\neg AB(c) | c \in COMP \setminus \Delta\} \cup \{AB(c) | c \in \Delta\}$ is satisfiable. The predicate AB stands for abnormal and is used in SD for explicitly stating correctness or incorrectness of a component.

In consistency-based diagnosis usually only the correct behavior of a component is described in SD with the consequence that every superset of a diagnosis is itself a diagnosis. Note that in Reiter's original paper (Reiter [1987]) diagnoses have to be always minimal. In our case we do not restrict the diagnosis definition but of course assume that the implementation delivers minimal diagnoses where minimality is defined with respect to subsets. I.e. a diagnosis is said to be minimal if no real subset of the diagnosis is itself a diagnosis accordingly to the given definition. We further restrict our model to only represent the correct behavior, which is sufficient for our purpose. Situation like the one described by Struss and Dressler [1989] hardly occur in our domain. Moreover, in case of further restrictions to be applied in the model we borrow the ideas of Friedrich et al. [1990] and represent impossibilities in the model directly.

In our application the observers described in the previous sections communicate with the diagnosis engine via specifying observations. This communication is done on a regular basis every pre-defined time step. The diagnosis model itself has no state and communicates back the results also once in every time step with a latency necessary for computing the diagnoses after receiving the observations. Since our model has no internal state the correctness or incorrectness of sensors is only done statically. This is possible because there is currently no need for a temporal model where time has to be treated explicitly. For communicating knowledge between the observers and the diagnosis engine we use the predicate *match* in cases where two sensors deliver the same results accordingly to the observer, and *no_match* otherwise. We assume that each observer either delivers *match* or *no_match* for a pair of sensors but never both.

In this section we use the following example for explaining the used model. We assume three sensors, i.e., a gyroscope IMU_GYRO, a compass IMU_YAW, and a sensor for angle measurements SM_YAW. All of them are compared pairwise using three observers, which deliver either $match(X, Y)$ or $no\_match(X, Y)$ for $X, Y \in \{IMU\_GYRO, IMU\_YAW, SM\_YAW\}$. For the diagnosis problem to be solved we assume all three sensors to be part of the set of components, i.e., $COMP = \{IMU\_GYRO, IMU\_YAW, SM\_YAW\}$.

In the following we formalize a model of the sensor system. For this purpose we use a predicate *relation* for stating that 2 sensors are "related", which means that there is an observer comparing the sensor values. Moreover, we introduce a predicate *sensor* stating that a certain constant represents a sensor. Hence, with *relation* and *sensor* we formalize the system's structure. Moreover, since *relation* is symmetric, we add the following rule to SD:

$$\forall X, Y : relation(X, Y) \leftrightarrow relation(Y, X)$$

For our example we add the following facts to SD:

$$\begin{gathered} sensor(IMU\_GYRO) \wedge \\ sensor(IMU\_YAW) \wedge \\ sensor(SM\_YAW) \wedge \\ relation(IMU\_GYRO, IMU\_YAW) \wedge \\ relation(IMU\_GYRO, SM\_YAW) \wedge \\ relation(IMU\_YAW, SM\_YAW) \end{gathered}$$

In the next step we define the behavior of a sensor. We rely on a very simple model stating that a correct sensor delivers a correct signal. Note that the opposite is not true in general. Thus an incorrect sensor might deliver correct or incorrect signals. We represent the sensor value using the predicate *ok* where the constant representing a sensor is used as parameter.

$$\forall S_i \in COMP : sensor(S_i) \rightarrow (\neg AB(S_i) \rightarrow ok(S_i))$$

If two sensors are related and thus compared with each other using an observer, a correct signal coming from one sensor should match a correct signal coming from the other connected sensor. Formally, we state this behavior using the following rule:

$$\begin{aligned} &\forall S_i, S_j \in COMP, S_i \neq S_j : \\ &\quad related(S_i, S_j) \rightarrow (ok(S_i) \wedge ok(S_j) \rightarrow match(S_i, S_j)) \end{aligned}$$

All of the described rules are part of the used model and are therefore element of SD. What is missing is a rule describing inconsistency between the behavior and the given observations. We do this by stating that it is impossible to derive *match* and *no_match* at the same time.

$$\begin{aligned} &\forall S_i, S_j \in COMP, S_i \neq S_j : \\ &\quad match(S_i, S_j) \wedge no\_match(S_i, S_j) \rightarrow \perp \end{aligned}$$

Using this model SD we are able to compute diagnoses for given observations. Let us first assume that the values of two comparisons do not match. E.g., we assume OBS to be

$\{match(IMU\_GYRO, SM\_YAW),$
$no\_match(IMU\_GYRO, IMU\_YAW),$
$no\_match(IMU\_YAW, SM\_YAW)\}$.

In this case there is only one single fault diagnosis $\{IMU\_YAW\}$ indicating the compass to be not working correctly.

If we assume another set of observations

$OBS' = \{match(IMU\_GYRO, SM\_YAW),$
$match(IMU\_GYRO, IMU\_YAW),$
$no\_match(IMU\_YAW, SM\_YAW)\}$,

we receive two single fault diagnoses $\{IMU\_YAW\}$ and $\{SM\_YAW\}$. Using the available information we cannot distinguish between these diagnoses.

Although the described model for sensor validation is rather simple, it can be extended for fitting to certain situations. In the following we describe how failure masking or information regarding fault propagation can be represented in our model. The latter situation deals with dependencies between sensors. We might know that two sensors react in the same way on environmental conditions. If one sensor delivers wrong values due to its environment, the other will do so as well and vice versa. For example, consider a situation where we have two sensors that do only work outside of buildings. In order to extend our model, we only need to add the following rule for our two sensors S_1, S_2 to our model SD:

$$(ok(S_1) \rightarrow ok(S_2)) \wedge (ok(S_2) \rightarrow ok(S_1))$$

This two rules ensure that whenever S_1 has to be assumed faulty, S_2 has to be assumed faulty as well in order to be able to resolve the contradiction.

Another issue is the handling of failure masking. We might have knowledge that if two sensors do match both must deliver a correct signal. We represent this knowledge by stating that whenever $match(S_1, S_2)$ is true for two sensors S_1 and S_2 their outputs must also be true.

$$match(S_1, S_2) \rightarrow (ok(S_1) \wedge ok(S_2))$$

Note that this kind of knowledge is very strong sometimes preventing the diagnosis engine to deliver a diagnosis. For example assume that failure masking is not allowed in our example and we obtain OBS' as the set of observations. In OBS' there is only one *no_match* predicate. Hence, from the other *match* predicates and the new rule, we immediately derive $ok(IMU\_GYRO)$, $ok(IMU\_YAW)$, and $ok(SM\_YAW)$ from which follows that $match(IMU\_YAW, SM\_YAW)$ has to be true regardless of any fault assumption. Hence, we are able to derive a contradiction from SD and OBS' but are not able to resolve it. A solution would be to make the assumption that failures are not masked itself explicit and relax the rule by saying that failure masking is only correct if the delivered observation itself is correct, i.e.:

$$match(S_1, S_2) \wedge CORR(match(S_1, S_2)) \rightarrow (ok(S_1) \wedge ok(S_2))$$

The diagnosis engine can use the new assumption $CORR$ to explain situations where no diagnoses can be derived due to the given observations.

Note that the introduced model is adapted to our application, i.e., the online validation of sensor input, with the objective to improve the sensor fusion via dynamic retracting and asserting sensors from the computation. The given models can be used in similar applications where more than 3 sensors are involved. Moreover, the system architecture behind, i.e., the coupling of observers with logic-based models, can also be used in other applications but requiring to adapt the underlying model.

The output of this diagnosis process is the basis for the decision of the robot which sensor data are reliable.

5. EXPERIMENTAL RESULTS

The proposed validation approach has been evaluated on a Telemax robot introduced in Section 2. The robot performed SLAM while driving from inside the robotics lab to an outdoor area of the Computer Science Department of the University of Freiburg.

The robot, which was manually controlled, started in right corner of the lab, went trough a stair case followed by a door, performed a large u-turn in the outdoor area, reentered the building and staircase, and moved within another u-shape trough the lab. It took 220 seconds to complete the path. The robot leaved the building after 70 seconds and reentered the building after 150 seconds.

During the run the robot collected data from the compass, the gyroscope, the laser range finder, and the odometry. This data was used for simultaneous localization and mapping. Figure 4 depicts the obtained grid-map for the first part of the path within the lab. During this section of the path all sensors delivered reliable data leading to an acceptable estimate of the map and robot location.

Fig. 4. Grid-map obtained by the robot within the lab. The robot starts in the right upper corner and moves until it reaches the stair case and the door in the lower right corner.

After the robot left the building, it faced a large open area with little structure, such as walls and corners. Consequently, the laser scanner frequently reported maximum range measurements. Due this fact the scan matching algorithm had not sufficient features for matching successive laser scans which is the basis of this algorithm. Therefore, estimates of the driven distance and angle suddenly became seriously inaccurate.

Algorithms not aware of such significant changes continue to integrate poor measurements leading to wrong position estimates and unusable maps. Figure 5 depicts this effect clearly. It shows an aerial image of the lab in the left lower corner. Moreover, it depicts the position estimate of the robot. It shows that during the beginning the estimates were correct. When the robot left the building (indicated by the white circle) the position estimates drifted away. The white circle indicates when the robot reentered the building. The errors in the position estimate can clearly be seen as the position estimation for that particular position is far behind the building at the left upper corner. This scenario clearly shows the need of an intelligent sensor validation approach.

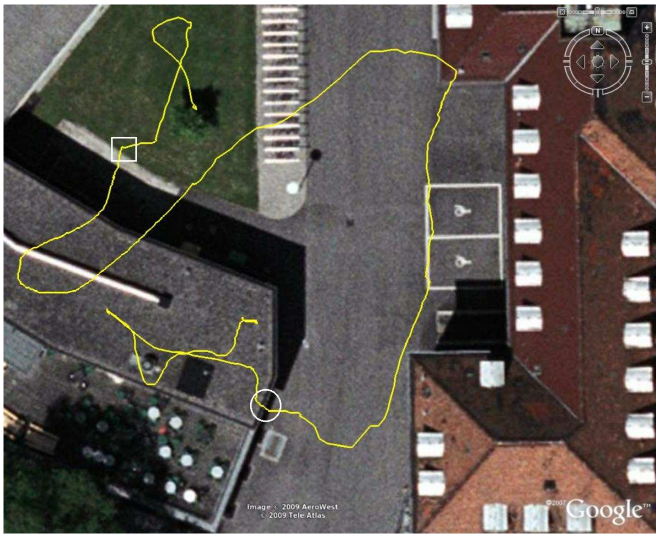

Fig. 5. Uncorrected path of the robot in the lab and on the parking lot. The white circle depicts the position on the path when the robot left the lab. It depicts also the true location of the door. The white square depicts the position on the path when the robot reenters the lab.

In a second experiment we computed from recorded data the same path while running the proposed sensor validation. Using the algorithms of Section 3 we observe the qualitative trends of the yaw measurements of the compass, the velocity measurements of the gyroscope and the yaw estimation of the scan matching. Figure 2 shows the qualitative abstraction of the yaw measurements of the compass. Moreover, we set up three observers which monitored the relation between the trends of the three sensors:
$relation(IMU\_GYRO, IMU\_YAW)$,
$relation(SM\_YAW, IMU\_YAW)$,
$relation(IMU\_GYRO, SM\_YAW)$

Figure 3 depicts the results of the observer for the trends of the yaw of the scan matching algorithm and the gyroscope of the IMU. Until 70 seconds the scan matching works correctly. From 70 to 150 seconds the alarms were triggered because of the mismatch of the trends. After the robot reentered the building, and thus the scan matching worked correctly again, the alarms disappeared.

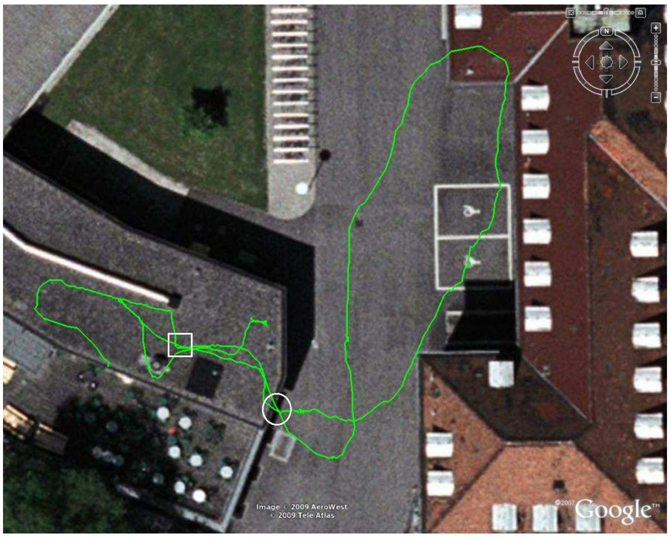

Fig. 6. Generated path of the robot in the lab and on the outer area using the proposed sensor validation algorithm. The white circle depicts the position on the path where the robot left the lab. It depicts also the true location of the door. The white square depicts the position on the path where the robot reentered the lab.

Using the logical model of the relations between the sensor, the observations made by the three observers, and model-based reasoning, we are able to reason back to the root cause that the scan matching algorithm did not work properly ($NAB(SM\_YAW)$).

The meta algorithm on the robot used this information to on-line adapt its sensor fusion. During periods where the scan matching is not reliable, the robot does not use the data for the localization and do a simply dead reckoning using the odometry. When there was no detection of erroneous data anymore, scan matching has been used again.

Figure 6 shows the improved path using the proposed sensor validation approach. There is still an error in the position estimation during the period the robot is outside the building but the error is significantly reduced. This is shown by the fact that the position estimation of leaving the building (white circle) and entering the building (white square) are much closer.

The results of the second run clearly shows that we are able to detect and localize problems in sensor data and that the feedback to the sensor fusion improves the overall performance of the system.

6. CONCLUSION AND FUTURE WORK

In this paper we proposed a sensor validation approach for autonomous mobile robots. The key idea is to compare qualitative trends in different sensor data streams in order to detect problems in one or more sensors. The abstraction of the data is based on qualitative reasoning techniques. It calculates the gradient in the data and maps it to some qualitative symbols. If two sensor streams are related by physical constrains, we are able to compare the trends of the two streams in order to detect unexpected behaviors from the data. Such unexpected behavior is usually a manifestation of a sensor failure.

We use observations of unexpected behaviors together with a logical model of the expected relation of different sensor streams to reason about the true root cause of a problem. We use consistency-based reasoning for this purpose. Once the root cause of a problem is located the robot is able to adapt its sensor processing on the fly in order to reduce the impact of the fault. For instance, the robot can decide to suppress a particular data source.

Preliminary experiments with a real robot in a search and rescue scenario have shown that the proposed approach is able to find and mitigate problems caused by disturbed sensors.

A major open question is how models of sensor relations can be obtained in general. For complex robot system it might be time consuming to generate such models manually. Moreover, the influence of parameters used in the proposed algorithms for the abstraction has to be further investigated. Finally, it is questionable if the proposed abstract models are expressive enough to represent all relations in complex systems.

REFERENCES

Bernhard Aichernig, Harald Brandl, and Franz Wotawa. Conformance test of hybrid systems with qualitative reasoning models. In *Fifth Workshop on Model-Based Testing*, York, UK, 2009.

Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl. Physical impossibility instead of fault models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 331–336, Boston, August 1990. Also appears in Readings in Model-Based Diagnosis (Morgan Kaufmann, 1992).

Alexander Kleiner and Christian Dornhege. Real-time Localization and Elevation Mapping within Urban Search and Rescue Scenarios. *Journal of Field Robotics*, 2007.

Alexander Kleiner, Gerald Steinbauer, and Franz Wotawa. Automated learning of diagnosis models for component-oriented robot control software. In *Proceedings of the Nineteens International Workshop on Principles of Diagnosis*, pages 257–282, Blue Mountains, Australia, 2008.

F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4: 333–349, 1997.

Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

Peter Struss and Oskar Dressler. Physical negation — Integrating fault models into the general diagnostic engine. In *Proceedings* 11^{th} *International Joint Conf. on Artificial Intelligence*, pages 1318–1323, Detroit, August 1989.