COBRA - A Generic Architecture for Robust Treatment of Uncertain Information

Werner Brockmann, Andreas Buschermöhle, Jan Hendrik Schoenke

University of Osnabrück
Smart Embedded Systems Group
Albrechtstrasse 28
49069 Osnabrück
Werner.Brockmann@Uni-Osnabrueck.de
Andreas.Buschermoehe@Uni-Osnabrueck.de
jschoenk@Uni-Osnabrueck.de

Abstract: This paper introduces the COBRA-architecture as an approach to handle uncertainties in autonomous robots and embedded systems in order to increase their robustness. In the COBRA-architecture, the trustworthiness of information is thereafter modeled explicitly by meta-signals, called trust signals. At a coarse-grained level, the architecture also extends normal functional modules to a triplet of modules, one containing the normal functionality with handling of deficient information, one containing a fall-back strategy for the case of too insufficient information, and one mitigating between them depending on the respective trustworthiness. The COBRA-architecture incorporates the trust signals also at a fine-grained level into the basic signal processing. As an example the extension of a typical class of continuous functions is presented to achieve more trustworthy results and to generate also a trust signal for the output. Both basic aspects of the COBRA-architecture are illustrated by the control architecture of the walking robot WALTER.

1 Introduction

Autonomous mobile robots as well as other types of embedded systems get increasingly complex and work in an unknown, time-variant yet safety-critical environment. Additionally, different kinds of dynamically varying disturbances and anomalies are very likely to occur at runtime. They influence sensor readings as well as the interaction with the environment. This affects at the same time the validity of internal information about the system’s state and about its environment and results in an ambiguity. Hence, the values of internal signals only give a more or less correct representation of reality, i.e. the information content of a value is more or less trustworthy or uncertain, respectively. Each module receiving such uncertain information has thus an output which is subject to uncertainties as well just in the sense of ”garbage in and garbage out”. As a robotic architecture normally consists of many modules processing and propagating information, the uncertainty is hence propagated likewise throughout the whole system.
It is important to note that these dynamically varying informational uncertainties are
not faults in the classical sense of fault tolerance. Nevertheless, comparably catastrophic
consequences may be caused due to the difference of the true value of a (physical) parameter
to the one being processed. Depending on its severity, the possible consequences of an
uncertainty are either only critical or can even be equated to a severe fault in the classical
sense. Accordingly, the operating mode can be distinguished between normal, critical, and
catastrophic.

Previous work in the DFG priority research programme Organic Computing [MSSU11]
showed that the boundary between normal and critical mode has to be handled differently
than the one between critical and catastrophic. The behavior in catastrophic mode must
be completely different from the other modes as information with a too high degree of
uncertainty has to be completely discarded. Thus in this mode processing is based on
different information, similar to classical fault tolerance which copes with normal and
catastrophic operation modes and discretely switches between the two.

The approach presented in this paper hence is complementarily focused on normal and
critical operation modes aiming at a smooth transition between the two. It consists of an
architectural part as well as an algorithmic part which both explicitly represent the degree
of uncertainty and consider this additional information to increase safety and robustness.
The goal is to provide a robust system operation which achieves the highest possible
performance for given uncertainties.

Such treatment of uncertainties in embedded systems places specific requirements on the
architecture and its algorithms. In normal operation mode the extension of uncertainty
treatment at the algorithmic level must not decrease performance and safety. In critical
operation mode several criteria must be optimized at the same time. In an integral approach,
the uncertainties of the outcomes of each module have to be reflected at architectural
level to allow successive modules to react to these uncertainties. With gradually varying
degrees of uncertainty the safety must be guaranteed at runtime while maintaining optimal
performance as far as possible at the same time. Due to guarantee system safety, the goal
is to design the system to be safe by construction at engineering level. Hence measures
based on probabilities, i.e. which are correct only some percentage of time, are improper
and avoided completely.

Time variant environments and system properties pose a special challenge. The latter can
be distinguished into time variance of the robot, e.g. changing friction or sensor drifts,
and time variance of the internal system behavior through adaptive or online-learning
system modules [BHB12]. Both categories cover system properties which are not modeled
or rather cannot be modeled at design time. Consequently, time variance dynamically
influences the uncertainties and hence reliability of knowledge about external properties
and internal parameters which are estimated or learned at runtime as well. So to guarantee
the system safety, a demand for controllability of the design requires the system behavior to
be predictable beforehand and to be interpretable afterwards for further manual optimization.
In practice, scalability is also essential, restricting the extension of uncertainty treatment to
critical functionalities. And finally, processing complexity is of concern within embedded
systems.
In the following, a reflection of related work on principles of uncertainty representation, architectures dealing with uncertainties, and especially the algorithmic treatment of uncertain inputs is given. Then the approach at the architectural level is presented and exemplarily illustrated on the algorithmic level for a typical class of algorithms. The paper closes with a discussion of the approach, a conclusion and future directions.

2 Related Work

2.1 Formal Representation of Uncertainty

Several ways of representing uncertainty have been developed. A general survey trying to summarize all representations within the common framework of generalized information theory is presented in [Kli05]. The simplest way of representing the uncertainty of an information is to form a set of all possible values or, as a special case, to require the value to lie within an interval. The use of this technique has been formalized in info gap decision theory [BH06]. It allows to consider all possible values of an information parameter and their respective results, but it has several drawbacks. Depending on the restrictions on the used sets, the complexity of calculations increases and additional information is necessary to estimate the set of possible values, e.g. from a single measurement. Furthermore, in the end of processing, e.g. when actuators of a robot are addressed, the set of possible values has to be projected in a meaningful way to a single scalar value as the output.

Further representations elaborate on the set of possible values by giving each possible value a weight. The most well-known approach is probability theory, where each value of the set is given a probability similar to the frequency of occurrence [TBF05]. This approach gives a well-defined theory of calculation, but again generally the calculations get much more complex and additional knowledge is necessary to map a measurement to a certain probability distribution. Usually a simplifying assumption is used, namely that of a normally distributed random variable. In this case the additional assumption only needs to give an estimate of the mean and its standard deviation, hence the calculations remain in their original complexity. Yet, with a normal distribution only bounded random variables like noisy measurements can be represented properly, but not all sources of uncertainties can be considered adequately this way [BBH10].

Similar to probability theory, fuzzy set theory gives weights to the elements of a set but with a different semantical meaning [Zad65, GS07], which can be viewed as a possibilistic approach, representing the vagueness of an information. The fuzzy membership function is hence not restricted to a total of one over all elements of the set. Again the characteristic function, which represents the degree of membership across the set has to be derived from additional information, e.g. by the designer, and the complexity of calculations increases.

In recent years, another layer of uncertainty is added for increased expressiveness and the weights of the set’s elements are viewed as a weighted set as well. In the case of probability theory this results in imprecise probabilities, e.g. Dempster-Shafer theory [DS76, Wal00, Wei00] where each probability gets an upper bound called plausibility and a
lower bound called belief. Similarly fuzzy sets can be extended to type-2 fuzzy sets [CM08]. But obviously, the extensions increase the complexity even more, both for calculation and for additional assumptions to derive and to engineer the distributions. And the designer is hampered in controllability and interpretability of the system behavior.

In contrast to these approaches representing the uncertainty by addressing all possible values, another way is to attribute the information with meta-signals. For example, in the ORCA-architecture (Organic Robot Control Architecture) [BMM05, MLESA07, BMG11] health signals are used to attribute the informational and operational health status of the system modules. They represent all kinds of uncertainties in a uniform way normalized to the range of \([0, 1]\). This approach is generalized to the Trust Management approach in [BBH10, BBHR11] which potentially attributes every information by a trust signal \(\vartheta\) representing its trustworthiness again normalized to the interval \([0, 1]\) and utilizes these trust signals in a more general way. A certain information is thus fully trustworthy and hence reliable \((\vartheta = 1)\), whereas uncertainty, e.g. in case of complete randomness or ambiguity, corresponds to no trustworthiness \((\vartheta = 0)\) and may cause a catastrophic event. Representing the trustworthiness and vice-versa the uncertainty by an attribute is a way to incorporate uncertainties which still only needs to process the sole value of the information. The trust level attribute can then be used to rate and control the trustworthiness of the system’s behavior. Thus the engineering and processing complexity increases only minimally.

2.2 Architectures with Uncertainty Treatment

Architectures for autonomous robots and embedded systems in general have also been extended to cope with uncertainties in recent years. Two categories can be distinguished here, depending on the way uncertainty treatment influences the system operation. On the one hand a detailed assessment and treatment of the uncertainties takes place at the level of concrete parameters, henceforth called fine-grained.

One of the most common approaches is to use the calculus of probability theory for an explicit representation of uncertainty [TBF05]. This modeling treats measurements as observations of random variables and state transitions depending on actions are modeled as conditional probabilities. Consequently, assumptions about the distribution of measurements and the conditional probabilities of actions have to be made and the results typically depend on the chosen frame of discernment. Alongside the complexity of additional information an engineer must give, the complexity of calculations increases significantly and often only approximate solutions, e.g. by particle filters [TBF05], are feasible.

On the other hand, uncertainties are treated at module level by steering the module’s activity, henceforth called coarse-grained. Here the functionality of the system is decomposed into several behaviors each designed for a specific sub-task. These modules interact with each other by the means of some meta-signals. This is similar to the activation-based behavior control architecture [ALBD03]. It works without explicit representation of uncertainty but uses meta-signals, called activity and target rating. Each behavior generates a target rating, expressing how much the behavior’s goal fits the current state. So modules
can be dynamically activated depending on the rating of a situation and it is possible to implicitly handle critical situations differently within this architecture without designing counteractions explicitly. The resulting actions of the robot are hence based on the normal signal flow from sensors to actuators as well as on these meta-signals. This activation mechanism hence allows to ensure the robot’s safety to a certain degree by activating only a defined set of behaviors, i.e. it deals implicitly with uncertainties at a coarse-grained level.

Another fine-grained robotic approach is the notion of a pain level which is introduced by [Fer94] to gradually model a sensor failure at runtime. The fault status of each sensor is estimated gradually based on the history of its measurements resulting in an additional signal attribute given to an injury agent declaring that the sensor is working or broken. In this architecture, each sensor with a pain level above some defined threshold is discarded from further processing and it is possible for a sensor to recover and being reintegrated into processing, thus making the approach comparable to coarse-grained fault tolerance.

As stated above, we and our partners further elaborated on the gradual nature of an uncertainty measure similar to pain levels, called a health signal regarding the health status not only of sensors but also of any processing module in the ORCA-architecture [MLESA+07, BMG+11, HSM13]. Here the normal signal flow is defined in basic control units (BCUs) providing the functionality of the system in a fault-free case. These are accompanied by organic control units (OCUs) which react to the health signals for supervising the BCUs and changing parameters within a BCU module as well as the interplay of the supervised BCUs in case of anomalies, thus allowing a coarse-grained handling of uncertainties. This influence is gradual and allows for a graceful degradation of the performance while maintaining the safety of the system.

In the ORCA-architecture, the system is fully functional without the OCUs. In order to get more flexibility and to improve expressiveness and performance, the biological motivation of health signals is abandoned with the introduction of the Trust Management framework [BBH10, BBHR11]. Its key features are to represent the trustworthiness of an information also by signal attributes, named trust signals here, but to integrate them directly into the signal processing and hence make the uncertainty treatment also an integral part of the BCUs. Due to a more general applicability of this underlying principle, Trust Management yields a generic uncertainty representation and treatment throughout the whole systems architecture. It thus no longer needs specific OCU-modules and the main advantage of this approach is a seamless migration from existing conventional and coarse-grained architectures to an integral fine-grained processing of uncertain information with a low additional complexity for uncertainty handling.

2.3 Uncertainty Treatment in Processing Modules

The fine-grained treatment of uncertain information or even missing inputs in single modules within an architecture is an unavoidable and well established part of real-world applications. A general overview of the most common methods within this field is given in [LEDL+05, GLSGFV10]. The entire field can be roughly divided into three areas.
First, with *imputation* a measured value of an uncertain input is replaced by a more certain one. The source to get the more certain value from depends on the imputation method. Several imputation methods are reviewed in [YXW12]. Basic imputation methods use filter, regression or nearest neighbor techniques to estimate an imputation value. Advanced imputation methods restrict the imputation values to comply to a given distribution. The module consuming the inputs is unaffected by this kind of uncertainty treatment and thus the complexity overhead is low. But the uncertainty about the inputs is not reflected in the output of this module which is thus treated as completely reliable. *Multiple imputation* methods generate a discrete set of imputation values resulting in a set of output values. Mean and variance of the set of output values define the most likely output value and its quality. But the complexity overhead is large due to multiple evaluations of the module.

To overcome the need for imputation, *ensemble* methods have been developed as a second means [MSP06]. A single module performing one task is replaced by an ensemble of modules all performing the same task, but each using a different subset of the complete set of inputs. To avoid propagating values of uncertain inputs, only the performing unit which does not use any of the uncertain inputs but uses all of the certain inputs is evaluated. For this kind of approach the number of modules increases exponentially with the number of uncertain inputs, and hence the engineering effort.

The third way to treat inputs uncertainty is to model the uncertainty directly. Therefor each input is extended according to one of the methods of chapter 2.1 and the module has to be able to handle the extended inputs. The main advantage of modeling uncertainties directly is the low complexity compared to ensemble methods in the cases of multiple uncertain inputs. The main disadvantage is the need to correctly model the uncertainties to get reliable results. For several uncertainty representations special solutions have been developed. The interval representation of uncertainties is used in [WA92, WWY92], the fuzzy representation in [Hla07] and (imprecise) probabilities in [WYXC07, NAD10]. These approaches are only fine-grained extensions to a conventional architecture, which do not distinct between critical and catastrophic case and inherit the calculation and design issues from their respective uncertainty representation.

### 3 Architectural Approach to Uncertainty Treatment

In the following the integration of coarse-grained and fine-grained uncertainty treatment is proposed to increase the robustness in such a way that highest possible performance and safety are gradually ensured as long as possible. Therefore the architecture is also extended for integration of the well founded concepts of fault tolerance. The goal is to get a fine-grained handling of uncertainties with low additional computing and engineering complexity. The concept is based on the gradual distinction of normal and critical operation modes, and a sharp distinction to the catastrophic one depending on the trustworthiness of the information at hand. Each module can work normally, with fine-grained or even coarse-grained gradual incorporation of the trustworthiness. If the trustworthiness of an information is so low that a safe operation cannot be guaranteed, a fault tolerance scheme is applied in a coarse-grained manner to get a safe fall-back strategy.
This principle is used for every critical module in the architecture and imposes two demands upon the design. First, the trustworthiness of an information has to be estimated at the source of the respective information which needs further knowledge about the information at hand. This effort is usually less than for other uncertainty representations. Second, every module has to be complemented by a safe fall-back strategy in the presence of (information) failure.

3.1 COBRA Architecture

The COBRA-architecture (Confidence Optimization-Braced Real-time Architecture) builds upon the Trust Management framework and hence accompanies potentially every information by a trust level, reflecting its uncertainty. Fig. 1 shows the principle organization of a COBRA-extended processing module. It adapts the concept of the Trust Management framework of fully integrating the OCUs functionality into the BCU by gradual blending from normal to critical operation mode and influencing internal parameters. Internally the trust level of the input data and possibly the trustworthiness of internal parameters and information is considered within the processing of the BCU. Together with the output of the BCU an accompanying trust level is given.

Each BCU is complemented explicitly with an additional specific fall-back control unit (FCU) to handle the catastrophic case. FCUs implement a fall-back strategy as a bridge from uncertainty treatment to classical fault tolerance methods. If the trust level of a BCU falls below a threshold, the FCUs output is used, i.e. in case of a lack of information due to uncertainty or to handle a complete failure. The FCUs therefore operate in a different way and/or on different information. This switching is handled by a meta-control unit (MCU) that fuses the output of the BCU and its corresponding FCU. This way the blending
between normal and critical operation is seamlessly handled in the BCU in a fine-grained manner and the boundary between critical and catastrophic mode is separately handled within the MCU in a coarse-grained manner. It hence acts as a guard of the respective functional module. This generic principle can also be applied hierarchically to encapsulate submodules the same way.

One key point of the COBRA-architecture is that the integration of critical informational states requires each functional module, i.e. each BCU, to be extended to incorporate the input’s trust level into its elementary processing and to give a trust level for its output. This extension again is based on the principles of Trust Management [BBH10]. If the trust level of an input signal is equal to one, it must influence the output in such a way as if the module has been designed without uncertainty handling. Contrariwise, if the trust level of an input signal is zero it must not have any influence on the BCUs output. In between the trust level should have a gradual influence. Such an extension of a BCU is exemplarily shown in the next section.

The control architecture of one leg of the walking robot WALTER (WALking TEst Robot) (see Fig. 2 and Fig. 3) shall illustrate this architecture by an example. WALTER is a simple demonstrational platform with four legs, each consisting of two servo-driven joints in a row. On top, a balancing weight allows to shift the center of gravity towards the supporting polygon while one leg is swinging. The control architecture for one leg is shown in Fig. 4 as an example for the COBRA-architecture. Here joint angle and motor current are measured for both the upper and lower joint and attributed with a trust signal. To get a trust signal for these measurements, background knowledge is used, e.g. noise levels, unexpectedly large changes in a measurement or measurements at the range limits are reflected by a reduction of the respective trust level. Such trust modeling of sensors is described in detail in [BBH10]. The augmented sensor information is given to a preprocessing module transforming the motor current measurements, i.e. leg load, and the foot-point acceleration to an estimate about the contact strength between leg and floor. A contact strength of zero means no contact and increasing values a harder contact. As this module gets inputs with trust levels,
internally it is build according to Fig. 1. This way, the contact estimation directly makes use of the trust level within the BCU and produces a trust signal (see Sec. 4 for details). The accompanying FCU keeps the last trusted contact strength estimate alongside as a persistence fall-back. As long as the BCU provides an output with a high trust level, the MCU provides this output and the respective trust. Otherwise, if the trust level of the BCU falls below a threshold, the MCU chooses the fall-back value as the module’s output and continuously decreases the output trust level in every step to model the aging of the fall-back value. The result is processed by the leg motion control behavior to generate trajectories while the robot motion control is mainly a discrete finite state machine selecting between the swing and stance phase for each leg.

In this way the leg and robot motion control get trust-optimized information. To keep the architecture as simple as possible, the uncertainty on the control action side is not modeled. The generated trajectories are targeted the way down to the lower and upper joint control which also get the trust attributed inputs from the joint angle measurement. So here again the scheme of Fig. 1 is applied to generate a safer control value. This means, if the current measurement is not trustworthy, a safe fall-back strategy, namely stopping the motion, is taken to prevent the system from reacting pathologically and the robot motion control (level) takes over.

This simple exemplary architecture shows the two main types of functionalities which cover most parts of any embedded and robotic system architecture. On the one hand, there are continuous functions as in the module determining the contact strength or the joint control. On the other hand, there are finite state machines as in the robot motion control module. Both types of algorithms need to be extended for a fine-grained integration of the input trust signals into their basic operation and to generate an output trust signal on their own. In this paper this extension is presented only for the first case of continuous functions in the next chapter which is illustrated in detail by an example thereafter.

### 3.2 Fine-grained Trust Extension of Continuous Functions

In low-level architectures, continuous functions are mostly implemented by simple functions $f(x)$ that are linear in the parameters, e.g. like PID-controllers, polynomials, Takagi-Sugeno fuzzy systems or radial basis functions, and thus easy to calculate. The general structure of this class of algorithms is given by equation (1), where $m$ is the number of inputs $x_k$, $n$ is the number of base functions and $\alpha_i$ is the respective linear parameter of the $i$-th base function $\phi_i(x)$.

$$f : \mathbb{R}^m \rightarrow \mathbb{R}; \quad f(x) = \sum_{i=1}^{n} \alpha_i \phi_i(x); \quad x = (x_1, \ldots, x_m)^T$$  \hspace{1cm} (1)

The main problem with this type of functions in presence of uncertain inputs is that the exact actual position $x$ within the input space $\mathbb{R}^m$ where the function has to be evaluated is unknown and therefore in general the output of the function is unknown too. Only if the output of the function is independent of the uncertain inputs given the certain inputs,
i.e. there is no ambiguity, a certain output can be generated. This dependency between the uncertain inputs and the output ambiguity also may vary depending on the certain inputs. But it is undesirable for an uncertain input (garbage in) to produce a random output (garbage out), because the function $f$ would not be safe by construction. So the question whether and how far an uncertain input is relevant to calculate the output can only be answered dynamically by the function $f$ itself. Thus the output calculation for uncertain inputs has to prevent the output from being disturbed by the uncertainty of inputs and has to determine the relevance of the uncertain inputs concerning the output to determine the output trust level $\vartheta_{\text{out}}$.

Our approach to making the output immune to uncertain inputs by using trust levels $\vartheta^x$ is to linearly limit the influence of a particular input $x_k$ on the output measured by the partial derivative according to its current trust level $\vartheta^x_k$. Formalizing this requirement, the
Assuming a factorial separation of the base functions

\[ \phi_i(x) = \phi_i(x_1, \ldots, x_m) = \prod_{k=1}^{m} \phi_{i,k}(x_k) \]  

(3)

into one dimensional base factors \( \phi_{i,k}(x_k) \), a solution of the partial differential equations (PDE) defined in equation (2) is

\[ f^\theta(x) = \sum_{i=1}^{n} \alpha_i \prod_{k=1}^{m} (\vartheta_k^2 \phi_{i,k}(x_k) + (1 - \vartheta_k^2)c_{i,k}) . \]  

(4)

The incorporation of trust levels \( \vartheta^x \) leads thus to a linear combination between the raw evaluation of the one dimensional base factors \( \phi_{i,k}(x_k) \) and free design parameters \( c_{i,k} \) determining the behavior of the function \( f^\theta \) for the extreme case in critical mode. This blending based on the trust level \( \vartheta_k^\theta \) does not conclude directly from the solution of the PDE defined in (2), but follows the principle of Trust Management to explicitly engineer the function’s behavior in every case.

To quantify the trust level \( \vartheta^\text{out} \) of the output, the dependency between an uncertain input \( x_k \) and the output, i.e. the variance \( \sigma_k^2 \) along the uncertain input dimension \( ([x_k, \bar{x}_k]) \), must be mapped to the trust level \( \vartheta^\text{out} \). If the variance \( \sigma_k^2 \) along the uncertain input \( x_k \) is zero, the output is independent of the uncertain input \( x_k \). The trust level \( \vartheta^\text{out} \) of the output in this case may hence not be decreased by the uncertainty of input \( x_k \). With an increasing variance \( \sigma_k^2 \) along an uncertain input \( x_k \), the output dependency increases too and the trust level \( \vartheta^\text{out} \) of the output must decrease accordingly. This basic principle for one uncertain input \( x_k \) can be applied to multiple uncertain inputs \( x \) by weighting the different variances along different inputs \( x_k \) with their trust levels \( \vartheta_k^\theta \). The calculation of such a trust level \( \vartheta^x \) weighted variance \( \sigma^2 \) of the function \( f \) is defined in equation (5).

\[ \sigma^2 = \text{Var}(f(x), \vartheta^x) = \text{Var}(f(x_1, \ldots, x_m), (\vartheta_1^x, \ldots, \vartheta_m^x)) = \left( \sum_{i,j=1}^{n} \alpha_i \alpha_j \prod_{k=1}^{m} \left( \vartheta_k^2 \phi_{i,k}(x_k) \phi_{j,k}(x_k) + (1 - \vartheta_k^2) \int_{\bar{x}_k - \bar{x}_k}^{\bar{x}_k} \phi_{i,k}(x) \phi_{j,k}(x) dx \right) \right) - \left( \prod_{k=1}^{m} \left( \vartheta_k^2 \phi_{i,k}(x_k) \phi_{j,k}(x_k) + (1 - \vartheta_k^2) \int_{\bar{x}_k - \bar{x}_k}^{\bar{x}_k} \phi_{i,k}(x) dx \int_{\bar{x}_k - \bar{x}_k}^{\bar{x}_k} \phi_{j,k}(x) dx \right) \right) \]  

(5)

The complexity of the variance calculation (5) is in \( \mathcal{O}(n^2) \), which is a comparatively small overhead for an exact solution. An easy way to finally map the variance \( \sigma^2 \) to an output trust level is \( \vartheta^\text{out} = \max(0, 1 - c \cdot \sqrt{\sigma^2}) \), where \( c > 0 \) defines the sensitivity of the output trust level \( \vartheta^\text{out} \) to the variance measure \( \sigma^2 \).
All in all, the enhanced function $f^{\vartheta}$ gets robust to input uncertainty. The designer explicitly models the behavior of the function $f^{\vartheta}$ for the extreme case in critical mode, i.e. with trust levels $\vartheta^{x}$ of zero, through the imputation values $c_{i,k}$, whereas the input uncertainty is modeled by trust levels $\vartheta^{x}$. The imputation values $c_{i,k}$ can also be chosen purely mathematically to comply to some input distribution in order to force the output to contract to the expected value of the distribution. Or any other expert knowledge can be used to make the output reach robust values in case of uncertain inputs predefined by design. The variance $\sigma^{2}$ is a well founded mathematical measure for the relevance of an uncertain input $x$ and its calculation complexity is acceptable. It yields a reasonable trust level $\vartheta^{\text{out}}$ for the output reflecting not only the input trust levels $\vartheta^{x}$, but also the actual impact of every uncertain input $x_{k}$ with respect to the resulting ambiguity of the continuous function $f$ at runtime.

4 Exemplary Application

To show the resulting behavior of such a fine-grained extension to continuous functions, the contact strength module of Fig. 4 is depicted for a concrete illustration. Here the information about foot-point acceleration and leg load is used to determine the contact strength between leg and floor. This relationship is expressed as a $7 \times 10$ grid-based control surface with linear interpolation as an instance of a linear in the parameter algorithm. Here the height at the grid points are the parameters $\alpha_{i}$. Consequently such a system can be extended with the mechanism described above. In this case the imputation values $c_{i,k}$ are chosen mathematically to conform with the expected value of an equally distributed input, which means that no further knowledge is introduced.

Figure 5: Surface plot of the TS fuzzy system estimating the contact strength (upper row) and the corresponding trust level of the output (lower row). From left to right the trust level of the acceleration input decreases (1.0, 0.66, 0.33, 0.0), i.e. the uncertainty increases.

The main idea of this module is that without contact the acceleration and motor load should be proportional and a low load value is expected. The more the load exceeds the expected load, the higher the contact strength is. Fig. 5 shows the functional behavior of the trust
extended module and its respective trust level for different degrees of uncertainty of the acceleration input for otherwise fixed parameters. The upper row shows that the output depends less on the acceleration input the lower its input trust level is. In the extreme case of no trust in the acceleration input, the output is completely independent of it (Fig. 5 right). So even for garbage on the input, the output is well defined.

Furthermore, depending on the ambiguity of the output (measured in the variance) with respect to the second input considering the uncertain first input, the result is more or less certain. This is reflected in the lower row of Fig. 5. In normal operation, the output is always certain (Fig. 5 left). If the acceleration input is totally uncertain (Fig. 5 right), the algorithm can still give a trustworthy output closer to the limits of the load input, because here the output is independent of the uncertain input. Whereas the outcome is highly dependent on the acceleration at a medium level of the load input and hence the trust level gets low.

5 Conclusion

This example of the walking robot WALTER shows that an existing architecture can be extended easily to integrate trust signals, and hence treatment of uncertainties in a fine-grained as well as coarse-grained manner. Only local changes are needed as the principle structure of an architecture is kept and only critical modules are extended and only necessary signals are attributed with a trust signal.

On the fine-grained level a more general rework of existing algorithms is needed to incorporate the input trust signals as an integral part into processing. Therefor a fundamental analysis and extension of the algorithm at hand is needed with three key issues. Firstly, the extension has to comply with the requirement of Trust Management that the influence of an input must be proportional to its trust level. Secondly, the algorithmic complexity should not increase with the extension to make its computation and engineering feasible. Lastly, the resulting output value again is attributed with an according trust level in a transparent way. The solution is then generally applicable, as the example for continuous functions presented here is applicable for a very widely used part of algorithms being linear in its parameters. But further classes of algorithms need to be extended.

As a result, uncertainties of potentially every information can be made explicit in the COBRA-architecture where needed. On a coarse-grained level the resulting output trust level is then transparently handled and offers a bridge to seamlessly integrate with fault tolerance as a fall-back for the case of information failure. The coarse-grained principle of uncertainty treatment can also be applied recursively to nested modules. Normal and critical operation in presence of uncertainties is then dealt with on the fine-grained level of the extended algorithms. And potentially catastrophic operation can be handled coarse-grained by the extended module. The resulting uncertainty is reflected in the output trust level and thus transparently propagated throughout the architecture, hiding internal uncertainty treatment as far as possible. This way the system is able to operate as long as possible in normal or critical mode as it dynamically adapts gradually to changing degrees of uncertainties and the catastrophic mode is mostly prevented. As these are only local
extensions, the scalability as well as the controllability and interpretability of the system behavior are still high.

Future work will have to address also finite state machines as another important class of algorithms. They must incorporate uncertainties by trust signals into the algorithm, not only at the inputs and at the output, but also at the internal state. And of course, future work will have to tackle the further integration of and to classical fault tolerance techniques. The COBRA-architecture is then an approach for robust treatment of uncertain information in autonomous robots and embedded systems in a general and generic way.

References


