Online Evaluation of Manipulation Tasks for Mobile Robots in Industry 4.0 Scenarios

Sebastian Zug, Stefan Wilske, Christoph Steup, Arnd Lüder
Otto-von-Guericke Universität Magdeburg
Magdeburg, Germany
Email: {zug, wilske, steup, lueder}@ovgu.de,

***Abstract*—The concepts of "Industry 4.0" are founded on Cyber-Physical Systems (CPS) that interact flexibly with each other. The system is no longer an individual robot equipped with a predefined set of sensors, drives and manipulators. Furthermore, it represents a task-specific selection of all available CPS in a certain area. This adaptive composition provides flexible handling of varying environmental conditions and stabilizes the perception quality related to the current task requests. But the permanent adaptation includes a number of challenging tasks - the selection and adjustment of the involved CPS has to be done on run-time now.**

In this paper we propose a new approach for analyzing CPS configurations for manipulation tasks. If a mobile robot has to handle an object, the algorithm explores all available actuators and sensors. The proposed concept applies a graph-based model to define the individual failures of each component and the geometrical links in between. Based on this representation, all suitable combinations are determined, the expected failure level is calculated and compared in relation to the requested handling precision.

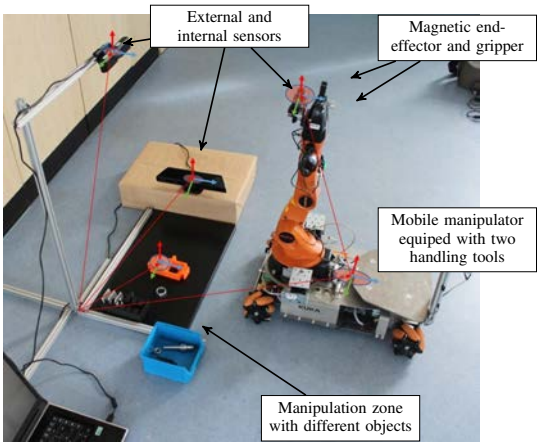


Figure 1. Exemplary scenario used for the evaluation. A mobile manipulator has to move objects from the black area, that is monitored by different internal and external sensors. The localization uncertainty of each entity is visualized by the red ellipses.

I. Introduction

The idea of "Industry 4.0" combines a number of significant changes in industrial production in order to provide versatile automated plants especially optimized for small charges [1]. The consequential need for permanent adaptation cannot be applied with traditional approaches and systems that have to be programmed, configured and adjusted by human workers and engineers. Instead, we require a kind of self-organization, applying a flexible interaction between intelligent devices (sensors, actuators, controllers). If we transfer this idea on the autonomous transportation and manipulation platform depicted in Fig. 1, it should offer its main services (move from A to B, grasp an object) and its environmental perceptions on raw-data level (camera images, laser scans) as well as on feature level (humans, objects) for all other entities. In the other direction, the robot receives data from other sensing devices and considers them for its operations. Accordingly, the planning and control algorithm of the platform and the manipulator will be confronted with a large bandwidth of situations related to

- environmental conditions related to light, temperature, etc.
- available sensors and corresponding data sets, due to the presences of other robots or an intelligent infrastructure
- requested precision and accuracy of the manipulation related to position and force and
- the properties of objects that have to be handled.

In various domains, these parameters are summarized by the term "context" of a task [2], [3]. In a conventional application the context is known to the developer at design time. Based on this knowledge he starts an iterative process combining the

1) aggregation of the context parameters,
2) comparison between requested and provided capabilities
3) decision about the applicability of a certain pickup and setting

In case of a negative result the engineer probably integrates additional sensors, refines the control or reconfigures the manipulator. Hence, he partially manipulates the context and optimizes the task execution with regard to costs, duration [4], power consumption or safety risks [5].

The intended flexibility of Industry 4.0-scenarios does not allow such off-line analysis and requires an evaluation and optimization process at run-time. The main challenge, a task-oriented analysis of the context is depicted in Fig. 1. The paper presents a new approach that compares the required precision and accuracy of the manipulation task with the capabilities of available sensor/actuator combinations. The corresponding

978-1-4673-7929-8/15/$31.00

pose deviation (position and orientation) between the manipulation objects and the end effector is determined by robot and/or sensor displacements as well as measurement errors. The errors are represented by Mahalanobis ellipses in Fig. 1. In order to evaluate these parameters, we model them by an undirected graph and integrate specific error assumptions. Based on this general representation, the effects of a platform and sensor displacement in conjunction with disturbed measurements and manipulator uncertainties can be mapped on the pose of the end effector. The function $p_f(c, \mathbf{p})$ describes the probability of a pose $\mathbf{p}$ based on the current context c. On the other hand, the error tolerance of the end effector is defined by a success function $p_s(c, \mathbf{p})$. Our approach matches both functions and calculates the probability of a successful task completion. The contribution of the paper is defined by the description of the generic graph model representing errors in sensor/actuator systems and its evaluation in relation to tasks requests. The on-line analysis guarantees high flexibility and an optimal use of all available resources.

This paper has the following structure. The next section describes the basic concepts, assumptions and algorithms of the approach followed by a first evaluation related to the exemplary scenario. Sec. IV reflects the current state of the art in the field of context aware manipulation tasks. The last section summarizes and defines future work.

II. Concept

In order to compute the feasibility of a given manipulation task, we have to define abstract representations of the robot, its components, external and internal sensors as well as the environment. It should be noted, that we have not considered dynamic aspects so far and that we are evaluating an isolated snapshot of the system. Possible improvements of the positioning precision and accuracy of the manipulator due to integrated control applications are not considered at this stage of the project. Hence, the analysis represents a worst case study and can provide fundamental assessments of the configuration.

A. System description

For the intended evaluation we need to design a mathematical description of

- the geometric relations between the components,
- the included possible deviations in position and orientation and
- the fault-tolerance of the manipulation tool.

In the first part of the section we discuss these points and explain the interaction in a second part.

1) Location related information: We assume each sensor and actuator as a smart entity providing machine readable information about its configuration and parameterization beside the interfaces for sensor data and actuation commands. One aspect in these descriptions are a pose information $\mathbf{p}$ (6DOF) related to parent coordinate systems. Beside, each CPS publishes its id, time stamp and the local frame name periodically embedded in a heart beat signal. The self-describing approach guarantees high flexibility and avoids a reconfiguration in case of appearing or disappearing components [6].

These information are gathered by a collector component in order to generate a model of the system context. Fig. 2 illustrates its output. Obviously, we have an actuator chain containing 2 joints, 2 manipulators (magnet, gripper). Additionally the robot is equipped with one sensor (sensor 0) Due to the possibility of the existence of multiple information sources for one object an equivalence table is needed which holds the information of this association. For example, one object (*object*) is tracked by different sensors (*sensor 0*, *sensor 1* and *sensor 2*) that may have different positions, orientations, error models and observation areas. These should not be represented by a single node in order to avoid a loss of details. Hence, we add separate entries for each sensor and its associations based on the evaluation of its pose and measurement area.

In a second phase the pose table is transformed to a directed graph. The relations between the different frames and poses are condensed in a transformation tree (see figure 2).

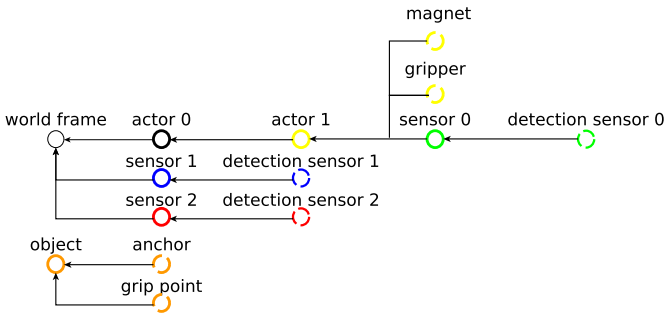


Figure 2. Representation of the relations between different CPS as a directed graph (transformation tree)

2) Error-related information: The second aspect, that is mapped on the graph representation concerns the characterization of the individual positioning errors and their effects on the end effector. Hence, we have to consider two aspects in order to predict the position quality of the manipulation tool - the individual error characteristic and their transformation through the graph.

We consider erroneous position assumptions for the distributed components (CPSs and manipulation objects) caused by:

- sensor measurements and
- uncertain precision and accuracy of the actuators
- missing or inaccurate calibration of manual mounted sensors

Errors can be defined in multiple ways. In this paper errors refer to multivariate displacements between assumed and real poses. It is assumed, that each node can inject errors related to an error distribution $p_f : \Re^6 \rightarrow \Re, \mathbf{p} \mapsto p$. The individual error models can be separately specified in two ways:

a) Parametric distributions: In many processes errors are described by Gaussian distributions. The decisive advantage is caused by the low computational effort for a combination of

two distributions and linear transformation operations. Another aspect relates to the number of parameters that have to be stored. A six degree of freedom pose is defined by a 6 dimensional vector of the mean μ and a 6x6 covariance matrix Σ. The probability of a specific pose error can be calculated by a multi-variant normal distribution:

$$f(\mathbf{p}) = \frac{1}{(2\pi)^3 \left|\Sigma\right|^{\frac{1}{2}}} exp\left(-\frac{1}{2}(\mathbf{p}-\mu)^T \Sigma^{-1}(\mathbf{p}-\mu)\right)$$

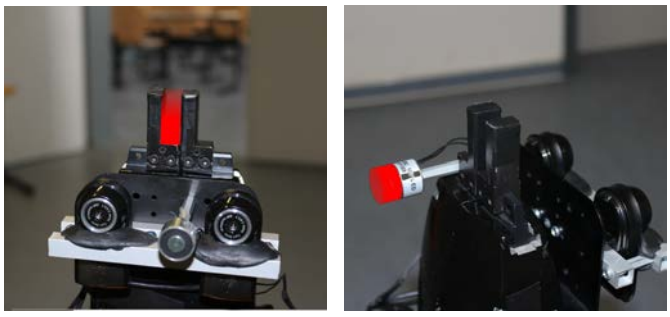
Related to parametric error models we consider only Gausian distributions in a our first implementation. The covariance matrix Σ has to be registered in the graph structure beside the estimated pose μ.

b) Non-parametric distributions: Parametric distributions lead to a fast computation but may not be the first choice for representing pose errors with multiple maxima, unbalance or uncommon shapes. For example, this applies to a fence and its effect on the error of a pose. Here we have to cut the parametric distribution at a certain axis and to describe the resulting distribution in a non-parametric way. Standard metrics like $mean$, std or $quartile$ are not suitable. Consequently, we decided to include a sample point based representation in our framework. The frequency of a pose is determined by the frequency of the corresponding sample point. This can lead to a high computational effort but gives the user the freedom to define:

- complex distribution shapes
- with a specific (local, global) resolution.

A low resolution may lead to low quality of the result, this is a trade off of computation time and quality. The sample points could be directly gained from the reference measuring for sensor and actor calibration.

Similar to the pose and frame data, all CPSs must be aware of the specific pose errors. Electronic data sheet concepts for this purpose are described in [6].



(a) Original Kuka-Youbot end-effector (pair of gripper jaws) (b) Magnetic end-effector

Figure 3. Parts inside the red area will be manipulated correctly. The transparency level of the red block illustrates different probabilities.

3) End-effector related information: The evaluation operation compares the possible displacement of the effector and its error tolerance. For this purpose we need to define a success function. It is task-specific and maps the pose error on a probability of a correct tasks execution. Furthermore, if an error pose were result in a damage or hazard the success function will provide a small probability.

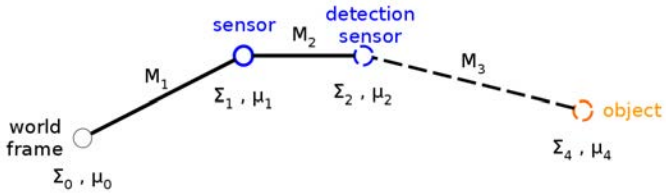
Fig. 3 illustrates two examples of the success function. The red zones in Fig. 3a and Fig. 3b show the area of a successful manipulation operation based on a series of grasping operations with objects at different positions. The various levels of transparency represent the probabilistic aspect. Manipulation tasks are potentially highly successful, if the object is located in the solid red zone. In the periphery the success probability decreases very fast, so that there is nearly no transparency level visible. The individual structure depends on the physical principles of the end effector.

The success function illustrated for the gripper in Fig. 3a can be described mathematically

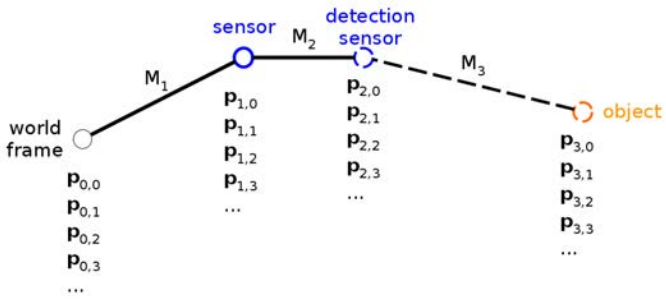
$$p_s(\mathbf{p}) = \begin{cases} 1.0 & if\ -0.010m < x < 0.010m \\ & \&\ -0.005m < y < 0.005m \\ & \&\ -0.005m < z < 0.005m \\ 1.0 - \frac{(x-0.010m)}{0.005m} & if\ 0.010m < x < 0.015m \\ & \&\ -0.005m < y < 0.005m \\ & \&\ -0.005m < z < 0.005m \\ 0.0 & else \end{cases} \quad (1)$$

The non-parametric success function is described by a set of sample points containing position and orientation information.

At this stage we are able to combine all three aspects and to generate the undirected graph enhanced by error and transformation data. Fig. 4 illustrates the result based on an extracted graph representing the scenario of Fig. 1. For each node a specific error model is assigned, each edge is described by a transformation matrix in homogenous coordinates.



(a) normal distributed error



(b) Non parametric distributed error

Figure 4. Exemplary graph model with additional context data (error assumptions for each node and transformation information for each edge)

The transformation of the errors from frame i to frame $i+1$ is applied by the jth transformation matrix M_j. This is shown

in equation 2 for the mean values μ_i or kth sample points $\mathbf{p}_{i,k}$ respectively. The tilde symbol indicates the transformation in the actual frame.

$$\tilde{\mathbf{p}}_{i+1,k} = M_j \mathbf{p}_{i,k} \tag{2}$$

The error propagation has to be calculated in the same way. Equation 3 shows the transformation of the covariance matrix Σ_i into the next frame $\tilde{\Sigma}_{i+1}$ with R_j as $3x3$ rotation matrix of the jth edge.

$$\tilde{\Sigma}_{i+1} = \begin{bmatrix} R_j & 0 \\ 0 & R_j \end{bmatrix} \Sigma_i \begin{bmatrix} R_j & 0 \\ 0 & R_j \end{bmatrix}^T \tag{3}$$

The transformed errors have to be combined with the errors occuring in the next frame. The error propagation is defined by equation 4 for parametric error models and in 5 for non-parametric models.

$$\Sigma_{i+1,total} = \tilde{\Sigma}_{i,j} + \Sigma_{i+1} \tag{4}$$

$$\mathbf{p}_{i+1,m} = \tilde{\mathbf{p}}_{i+1,k} + \mathbf{p}_{i+1,l} \tag{5}$$

As visible in the last equation, by merging the previous samples of $\mathbf{p}_{i+1}$ with $\tilde{\mathbf{p}}_{i+1}$, the number of sample points increases. The evaluation algorithm normalizes the distribution and calculate a new sample set with a constant size. The corresponding configurations are globally defined.

For executing of the task, two elements must have the same pose. The gripper must be located at the grip-point or the magnet must meet the armature core assembled on the object. The system needs the IDs of the two elements to compute the relative error.

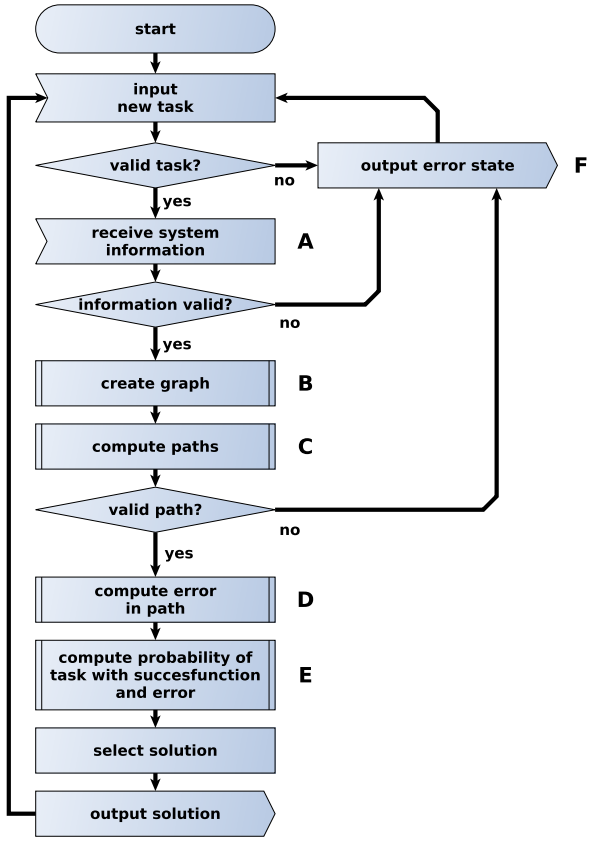


Figure 5. Sequence diagram for computing the execution probability of a task

B. Processing chain

The algorithm computes the probability in five main steps (see figure 5 **A-E**).

1) Receive system information: First, the actual system is captured by the *Collector*. It listens to the fault and geometric related information broadcast by the elements (*sensors* and *actors*) (**A**). If no information is available, it leads to an error state (**F**).

The *Collector* registers the elements, holds an actual set of data and watches for missing heart-beat signals in order to adapt the situation context. The monitoring process completes the actual data-sets and provides the access for one or more *Evaluation* components.

2) Create graph: The *Evaluation* starts computing after a request of the *Task manager*.

Therefore the task-related information (success function) is submitted by the request message. The *Evaluation* checks the message for consistency to avoid computational effort, e.g. if both elements of the task are the same.

Afterwards the actual set of available CPSs is requested by the *Collector*.

In the second phase, the geometric relations of the physical system are used to generate edges and nodes, which leads to the structuring of the system as an undirected graph (**B**) (see figure 6). The error information are stored in the corresponding nodes. With this approach the recombination and reuse of system elements can be easily handled and complies to the idea of CPS.

It should be noted again that all information is presented by nodes and edges. For example the knowledge of the existence of an anchor or grip point on a part must lead to an element broadcasting the error and geometric details so that this information can be represented in the graph.

The information of the equivalence table are depicted as dashed lines. The task related information are illustrated by a dotted line.

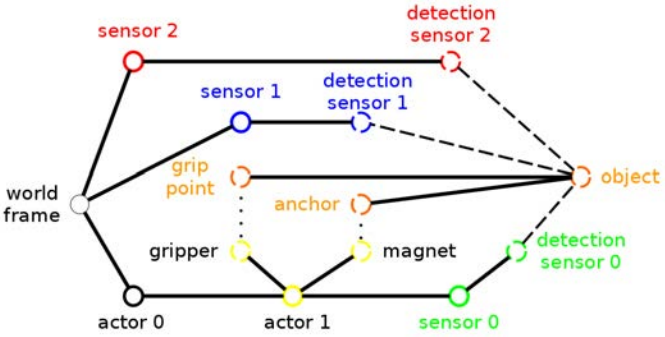


Figure 6. System and additional information represented as a graph

3) Compute paths: In a third step the graph is analyzed for the existence of a connection between the task relevant nodes (**C**, nodes representing the tool and the object). The algorithm regards the information of the equivalence table and creates different paths. This results in up to n valid paths or an error state (**F**), when the robot does not have enough information to perform the task. Each path represents a set of information sources with geometric relations and errors.

4) Compute error in path: By propagating and superimposing the error from the first node trough the path the overall fault relatively to the relevant node is gained in a fourth step (**D**). In this context, the result is influenced by every transformation and the shape of the fault. If all elements are free of error, the relative error will vanish.

5) Compute probability: In the last step, the error details of each set of sources in combination with the success function are used to calculate the n probability of a successful execution (**E**). If multiple (m) success functions (or actors) are located on the same pose, the probability can be calculated with the same error information and we obtain $k = m \cdot n$ probabilities. In combination with external information like cost functions, this can be used to make assumptions about global optimality and the use of resources.

When calculating the probability every occurrence probability of a pose error is multiplied with the probability of the success function at this pose. This results in a common probability of the pose. By integrating over all poses the overall execution probability is computed.

$$P = \int_{-\infty}^{\infty} p_s(\mathbf{p}) \cdot p_f(\mathbf{p}) \, \mathrm{d}\mathbf{p} \tag{6}$$

III. Implementation and Evaluation

The system was implemented in C++ based on the robotic operating system (ROS). This framework represents the quasi-standard for mobile robotic application and provides a service or publish/subscribe communication middleware, a huge amount of robotic-specific applications for navigation and manipulation as well as a large number of hardware drivers. Due to the communication infrastructure, hardware components can be replaced transparently by simulated ones. In our case we used VRep for this purpose and implemented the scenario of Fig. 1 as visible in Fig. 8. VRep provides

- a large number of sensor models required for constructing the (virtual) intelligent environment,
- a physics engine in the background important for an examination of the grasping process and
- a powerful interface to control all components from ROS.

The graph implementation described in the previous chapter, integrates the ROS frame transformation (tf) system. Each ROS message type includes a frame identifier. Each frame represents a local coordinate system. In order to connect different frames their geometrical relations have to be described by special tf-messages. While creating the graph, the transformation of the links are read from the tf-tree and stored as part of the edge-information in a matrix. For this purpose, the eigen-lib is used and allows a simple and fast computation of sample-point, frame and covariance transformation and manipulation by matrix multiplication.

A. Evaluation example

The evaluation addresses the scenario depicted in Fig. 1. It contains four components: the mobile robot KUKA youbot with 5 DOF, a manipulator arm with a stereo camera and 2 RGB-D cameras in the surrounding environment. The robot is supposed to lift up objects from the ground. In the evaluation scenario we just consider the gripper jaw as end-effector. The robot and the sensors are positioned relative to a world frame. The pose details of the sensors as well as the detection results are disturbed by errors corresponding to the error assumptions. Our method is used to predict the probability of a success for the manipulation.

B. Realization

Fig. 7 illustrates the implementation of the scenario from a component perspective. The sensors and actuators on the left are application-specific and may appear or disappear at runtime. Each sensor/actuator provides its specific ID, frame ID, a heart beat signal and tf information (static or variable).

The components on the right represent the core elements of our approach: the *Collector*, the *Evaluation* and the user interface. A reconfiguration of the system is recognized by the *Collector* based on new component IDs (appearing) or missing heart beat signals (disappearing) or information updates (rearrangement). A *Task manager* query causes the *Evaluation* to use the actual system information from the *Collector* and to compute the execution probability. The interaction with the *Collector* and the *Evaluation* is controlled by sending special ROS messages. For that, some ROS-tools serve as user interface or GUI respectively. ROS implements many tools for introspecting and interaction with the system, e.g. topic-monitor, message-publisher and node-graph.

C. Simulation

For a first proof-of-concept testing we just consider the robot and one external RBB-D sensor. The framework provides a fault injection tool for each component, defining the error assumption of a component and generating the corresponding disturbances for poses. We consider three different error configurations with varying orientations and amplitudes as visible in Fig. 8 and Tab. I. The errors are Gaussian distributed. We used a non-parametric representation with 1 million samples.

After the evaluation script launched all components, the analysis *Evaluation* calculates the probability of a successful run (8 sec.). In a second phase the simulation was executed (17 sec.) on a quad-core Intel core i7 4th gen. with 4GHz and 32GB RAM. This process was repeated 100 times for each error model. The simulation and the algorithm results are shown in Tab. I.

Predicted and simulated results reflect similar trends. As expected, the success probability decreases with the increase of

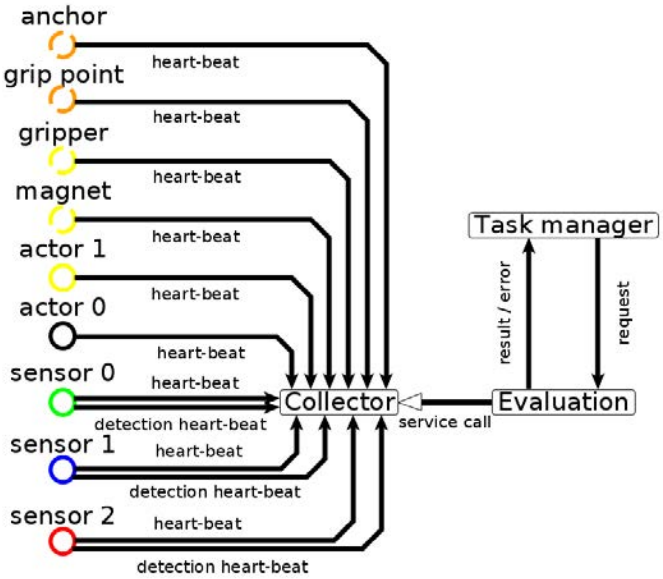


Figure 7. Components of the scenario depicted in Fig. 1. On the left side the sensors and actuators, on the right side the three core elements of the evaluation process.

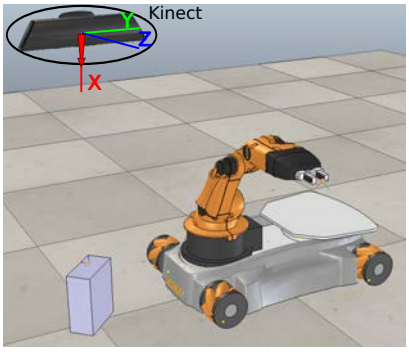


Figure 8. Simulation environment of the scenario. The arrows illustrate the orientation of the injected faults

the disturbance and with the number of disturbed coordinates. The relative deviation of simulation and prediction increases with higher disturbance as well. These deviations can be explained with multiple effects concerning the algorithm and the computation properties due to:

- random number generation for simulated disturbance
- number of samples for simulation
- discretization of normal distribution.

IV. State-of-the-Art

The idea of a context oriented self adaptation is a core feature of Industry 4.0 Cyber-Physical Systems (CPS) [7] and Internet of Things (IoT). Consequently, there is a large number of publications related to context specification situation assessment [3] or adaptive planning [8],

Nevertheless, the core features of the intended dynamic evaluation of manipulation tasks – an abstract representation of the position errors, methods for evaluating their propagation and effects on the system and a concept for describing application demands – are not covered as a whole. Hence, we have to explore the relevant work for the three mentioned topics separately. It should be noted that the following list includes only error processing approaches executed at runtime. Methods for design time evaluations like Failure Modes and Effects Analysis (FMEA) [9] are not considered at this point.

Table I
Comparison between predicted and simulated percentage of successful completed tasks

Configuration		Probability for success	
evaluation case	standard deviation	prediction result	simulation
(1) Disturbance on x	0.1m	0.6 %	10.0 %
	0.01m	40.7 %	55.0 %
	0.001m	100.0 %	100.0 %
(2) Disturbance on z	0.1m	4.3 %	10.0 %
	0.01m	48.5 %	52.0 %
	0.001m	100.0 %	100.0 %
(3) Disturbance on x and z	0.1m	0.2 %	1.0 %
	0.01m	18.9 %	29.0 %
	0.001m	100.0 %	100.0 %

A. Error description/characterization

Two different ways for error description are commonly used in robotic applications. First of all, we can use an interval analysis defining the maximum amplitude of an error or a probabilistic concept. The last one can address parametric or non-parametric distributions. Due to the benefits of the Gaussian distribution this model is often used in robotic applications. Correspondingly, this distribution is considered in many electronic data sheet concepts (for instance in [10]) or middleware implementations like ROS.

A number of approaches aim at an encapsulation of the error characteristic on a high abstraction level. For instance, Elmenreich, Pitzek, and Schlager propose a validity value between 0 and 1 for this purpose. In the same way, the authors of [12] proposed a validity value based on an adapted FMEA concept reaching from 1 to 1000. Due to the high level of abstraction the validity values can be used to identify an error level and to compare sensing values. However, the approach is focused on scalar values and does not cover position information yet.

B. Error propagation

The approaches mainly used for error propagation combine the already mentioned interval analysis [13] and probabilistic representations. Both approaches apply linear or non-linear First-Order Error Propagation models of the system to map the input error characteristic on an output error characteristic. For non-parametric distributions Monte-Carlo Simulations calculates the propagation of individual samples. A comprehensive overview is given in [14]. Another approach is given in [5]. The authors use an error representation based a on a vectorized structure containing amplitude and occurrence probability.

The propagation through the system is modeled by matrix operations.

C. Application related error evaluation

The concepts of fault tolerant control illustrate the lack of the current state of the art. Particularly, the model-based approaches are well suited to recognize erroneous states, but the configuration of the detection strategy and the evaluation rules (both represents the application needs) are defined at design-time. A description of the most common approaches is given in [15]

V. Conclusion and future work

The paper motivated the online analysis of manipulation tasks in CPS scenarios and presented concepts to manage the resulting problems. A simulated scenario was used to evaluate a first implementation. For this purpose, the results of a number of simulation runs were compared to the predicted success.

In the next steps the project will be enhanced by

- a new evaluation based on a simulation scenario including:
  - a variable set of multiple heterogeneous sensors,
  - a robot error model,
  - both actuator types,
- the integration of system dynamics and aspects of sensor actuator control loops and
- an application of the framework on a real world scenario.

Another important feature needed for an applicability of the approach is the automated generation of (sensor) error models based on the robot.

Acknowledgment

This work has been partially supported by the DFG, through project INST 272/221-1 “Mobile Cooperative Robotics” (MoCoRo).

References

[1] M. Brettel, N. Friederichsen, M. Keller, and M. Rosenberg, “How virtualization, decentralization and network building change the manufacturing landscape: an industry 4.0 perspective,” *International Journal of Mechanical, Industrial Science and Engineering*, vol. 8, no. 1, pp. 37–44, 2014.

[2] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, “Towards a better understanding of context and context-awareness,” in *Handheld and ubiquitous computing*, Springer, 1999, pp. 304–307.

[3] D. Zhang, X. H. Wang, and K. Hackbarth, “Osgi based service infrastructure for context aware automotive telematics,” in *Proceedings of the IEEE 59th vehicular technology conference, May*, vol. 5, 2004, pp. 2957–2961.

[4] S. Alatartsev and F. Ortmeier, “Improving the sequence of robotic tasks with freedom of execution,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*, 2014, pp. 4503–4510.

[5] S. Zug, T. Brade, J. Kaiser, and S. Potluri, “An approach supporting fault-propagation analysis for smart sensor systems,” in *Computer Safety, Reliability, and Security*, Springer Berlin Heidelberg, 2012, pp. 162–173.

[6] S. Zug, M. Schulze, A. Dietrich, and J. Kaiser, “Programming abstractions and middleware for building control systems as networks of smart sensors and actuators,” in *Proceedings of Emerging Technologies in Factory Automation (ETFA ’10)*, Bilbao, Spain, Sep. 2010.

[7] J. Shi, J. Wan, H. Yan, and H. Suo, “A Survey of Cyber-Physical Systems,” in *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*, IEEE, 2011, pp. 1–6.

[8] T. Heyer and A Graser, “Semi-autonomous initial monitoring for context-aware task planning,” in *Advanced Intelligent Mechatronics (AIM), 2011 IEEE/ASME International Conference on*, IEEE, 2011, pp. 667–672.

[9] D. H. Stamatis, *Failure Mode and Effect Analysis: FMEA from Theory to Execution*, 2nd ed. ASQ Quality Press, Apr. 2003.

[10] E. Song and K. Lee, “Understanding IEEE 1451-Networked smart transducer interface standard-What is a smart transducer?” *Instrumentation & Measurement Magazine, IEEE*, vol. 11, no. 2, pp. 11–17, 2008.

[11] W. Elmenreich, S. Pitzek, and M. Schlager, “Modeling Distributed Embedded Applications on an Interface File System,” in *Proceedings of the Seventh IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC’04)*, Vienna, Austria, 2004, pp. 175–182.

[12] S. Zug, A. Dietrich, and J. Kaiser, “Fault-handling in networked sensor systems,” in *Fault Diagnosis in Robotic and Industrial Systems*, G. Rigatos, Ed. St. Franklin, Australia: Concept Press Ltd., 2012.

[13] C. Carreras and I. D. Walker, “Interval methods for fault-tree analysis in robotics,” *Reliability, IEEE Transactions on*, vol. 50, no. 1, pp. 3–11, 2001.

[14] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[15] M. Blanke, M. Kinnaert, J. Schröder, J. Lunze, and M. Staroswiecki, *Diagnosis and fault-tolerant control*. Heidelberg: Springer, 2003.