Context-based Selection and Execution of Robot Perception Graphs

Nico Hochgeschwender*†, Miguel A. Olivares-Mendez†, Holger Voos†, and Gerhard K. Kraetzschmar*
*Bonn-Rhein-Sieg University, Sankt Augustin, Germany
{forename.lastname}@h-brs.de
†University of Luxembourg, Luxembourg
{forename.lastname}@uni.lu

***Abstract*—To perform a wide range of tasks service robots need to robustly extract knowledge about the world from the data perceived through the robot's sensors even in the presence of varying context-conditions. This makes the design and development of robot perception architectures a challenging exercise. In this paper we propose a robot perception architecture which enables to select and execute at runtime different perception graphs based on monitored context changes. To achieve this the architecture is structured as a feedback loop and contains a repository of different perception graph configurations suitable for various context conditions.**

I. Introduction

Service robots operating in the air, underwater and on the ground are expected to perform a *wide range* of challenging tasks in dynamic environments. To perform these tasks robustly robots need to extract task knowledge about the world from the data perceived through the robots sensors even in the presence of varying context-conditions. Examples of varying context-conditions are environmental changes (e.g., illumination variations), decreasing resources such as memory and energy, and context-changes effecting the ability to perform tasks at all such as sensor failures. Unmanaged context-changes usually degrade the performance of perceptual systems [1] and thus robot's performance. As Crowley *et al.* [2] and Borzenko *et al.* [3] described in order to cope with these context-changes intelligently a perception architecture should be able to adapt to the wide range of situations autonomously by adjusting component parameters [1], replacing perceptual components [4][5] or even dynamically replacing complete processing chains [6]. Within robotics where increased autonomy is desired the adaptation should be performed preferably without human intervention. However, the design and development of such adaptive (robot) perception architectures is a challenging exercise. First, representations are required to model the functionalities and components which can be adapted. Second, adaptation methods need to be developed which perform reasoning on the models with the aim to derive adaptation actions such as starting, stopping or modifying certain functionalities. Third, both aspects need to be integrated in a fully functioning robot perception architecture.

In this work-in-progress paper we propose an adaptive robot perception architecture which enables to *select* and *execute* robot perception graphs (see Sec. II) based on monitored context-changes faced during runtime. We exemplified and evaluated the overall approach for the task of detecting markers in the presence of continous and discrete illumination changes faced by robots in real-world scenarios. The motivation to focus in this work solely on illumination changes is based on the observation that robots deployed in long-term scenarios (from several hours to several days) need to cope with them in order to robustly provide their service as their perceptual functionalities are immediatly effected by them. In summary, we make the following contributions:

- We employ the Robot Perception Specification Language (RPSL) [6] to model perception graphs implementing different marker detection approaches.
- We introduce a simple, yet powerful reasoning algorithm which selects an appropriate perception graph.
- We introduce an adaptive robot perception architecture structured as a feedback loop which integrates the previous aspects and enables to *select* and *execute* robot perception graphs dynamically at runtime based on monitored context-changes.

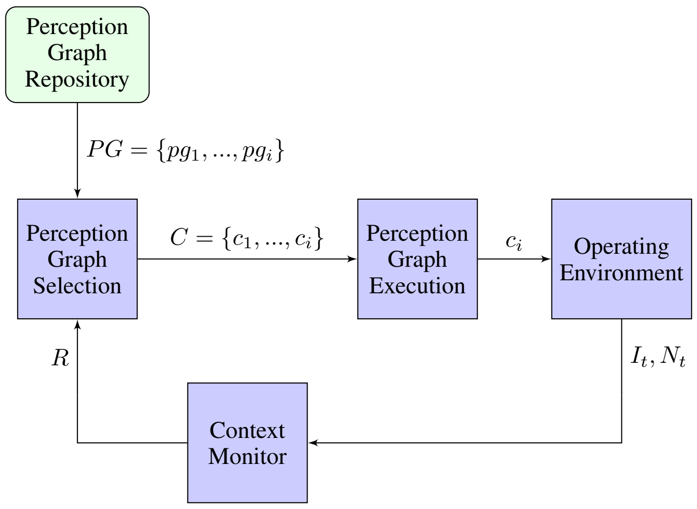
Perception Graph Repository
$PG = \{pg_1, ..., pg_i\}$
Perception Graph Selection
$C = \{c_1, ..., c_i\}$
Perception Graph Execution
c_i
Operating Environment
I_t, N_t
R
Context Monitor

Fig. 1: The adaptive perception architecture structured as a feedback control loop.

II. Context-based Selection and Execution of Robot Perception Graphs

An overview of the proposed adaptive perception architecture is shown in Fig. 1. The architecture is structured as a feedback control loop and contains the following constituents:

- A *perception graph repository* where domain models of different marker detection perception graphs $PG = \{pg_1, ..., pg_i\}$ are stored.
- A *perception graph selection* module which selects an appropriate perception graph from the repository. The selection is triggered by an adaptation request R from the context monitor.
- A *perception graph execution* module which stops the current perception graph and eventually starts a new perception graph.
- An *operating environment* where the perception graph is deployed and executed.
- A *context monitor* which monitors context conditions. In the context of this paper the context monitor observes the number of currently detected markers N_t and the current illumination condition I_t. In the case of a performance violation (minimum number of markers not detected) the *context monitor* triggers an adaptation request R.

A. Perception Graph Repository

The Robot Perception Specification Language (RPSL) proposed in [6] is used to represent the knowledge required for the context-based selection of an appropriate marker detection perception graph. The RPSL is a Domain-Specific Language (DSL) [7] which enables to model two crucial elements of perception systems in a declarative and formal manner, namely perception graphs and the data types. In RPSL a *perception graph (PG)* is a composition of components in the form of a Directed Acyclic Graph (DAG). Here, components are distinguished between a) *sensor components* representing sensors such as cameras, range-finders, etc., and b) *processing components* implementing perceptual algorithms such as filters, feature descriptors and marker detection methods. In the context of this work a PG encodes one particular configuration of the Aruco [8] marker detection algorithm. The Aruco framework allows to configure two aspects of the core Aruco detection algorithm, namely parameters for the adaptive tresholding and the type of method to refine detected corners (e.g., Harris, Subpix etc.). For the adaptive tresholding the first parameter defines the block size of the pixel neighborhood that is used to calculate a treshold value for the pixel and the second parameter defines a constant subtracted from the mean. Both parameters are real-valued. We observed that modifying these configuration options have a significant impact on the performance of the marker detectors in the presence of illumination changes. Here, performance is defined as whether or not to detect markers and how robust the pose estimation of detected markers is. As PGs consume and produce data on spanning multiple levels of abstractions ranging from raw sensor data and subsymbolic representations to symbolic information a suitable knowledge representation mechanism is required. In RPSL the vector-based Conceptual Space (CS) [9] knowledge representation framework is employed. A CS contains the following constituent parts:

- A **Conceptual Space** is a metric space where **Concepts** are defined as convex regions in a set of domains (e.g., the concept *Color*).
- A **Domain** includes a set of **Domain Dimensions** that form a unit and are measurable (e.g., the color space *RGB* with *red*, *green*, and *blue* dimensions).
- A **Prototype** is an instance which encodes typical values for a Concept (e.g., the color *Red* with the following values for red, green and blue: 255, 0, and 0)

In the context of this work the CS representation is not only used to model the data in- and output of perception graphs such as images and configuration options of PGs as described above, but also to encode *performance profiles* in the form of prototypes. A performance profile is a histogram of a concrete illumination condition where a specific configuration of a PGs performed good. We obtained these performance profiles during a training phase (see Fig. 2) where we applied different PG configurations to different lightning scenarios. In the case the PG performed good (at least 5 out of 10 existent markers where detected) we stored the histogram of the gray-value image of the current scene with a bin size of 10. Hence, each PG in the repository is attached with a set of performance profile prototypes which are later used to compute the most suitable PG. In Table I the compilation of the perception graph repository is shown.

Configuration	Corner method	T1	T2
A	Harris	30	30
B	Harris	60	60
C	Harris	40	40
D	Subpix	50	50
E	Harris	15	2

TABLE I: Compilation of the perception graph repository.

B. Perception Graph Selection

The main objective of the perception graph selection module is to select perception graphs dynamically based on requests. In the context of this work a request R is triggered by the context monitor in the case of a performance violation (minimum number of markers not detected). The minimum number of markers to detect is configurable, but in the context of this work defined as 5. For each monitored performance violation a request R is triggered. The request contains the current illumination condition represented in the same format as performance profiles described in Sec. II-A. As shown in Algorithm 1 the perception graph selection module then iterates over each output o_i of a perception graph pg_i and checks whether the concept defined there matches with the concept defined in the request. More precisely, we check whether a performance profile for a perception graph is given or not. To select a perception graph which matches the current context-condition (illumination condition) most closely we compute the Kullback-Leibler divergence between the illumination condition defined in the request and the one stored in the repository and associated with a perception graph. The Kullback-Leibler divergence defined as follows $KL(P,Q) = \sum_{x \in X} P(x) \cdot \log \frac{P(x)}{Q(x)}$ is a suitable similarity measure for this task as we model the illumination condition as a distribution of pixels over intervals (histogram). It is important to note that dependent on the application domain the selection Algorithm 1 can be equipped with different similarity measures such as euclidian distance or jaccard distance.

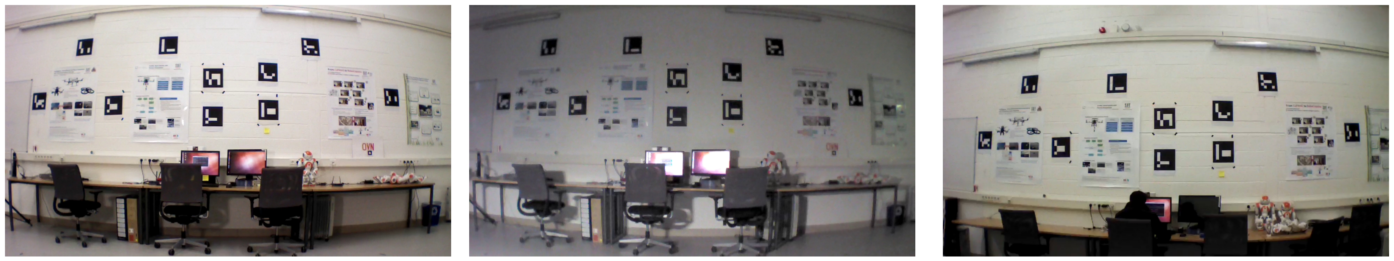

Fig. 2: Different illumination scenarios employed during the training phase.

Algorithm 1 Perception graph selection.

```
Input:
Request $R$
Set of perception graphs $PG = \{pg_1, ..., pg_i\}$
Output:
Set of candidates $C = \{c_1, ..., c_i\}$
where $C \subset PG$
  for all Output $o_i$ in $PG$ do
    if Concept $C_R$ of $R$ matches the concept $C_i$ of $o_i$ then
      if $o_i$ includes Prototype $p_i$ then
        $d_i \leftarrow KullbackLeibler(C_R, p_i)$
        $C \leftarrow C \cup pg_i$
```

C. Perception Graph Execution

Based on the list of candidates $C = \{c_1, ..., c_i\}$ provided by the perception graph selection module the perception graph execution module executes one perception graph candidate c_i. In the context of this work the candidate with the smallest distance d_i is executed if it is not already being executed.

III. Experimental Evaluation

We evaluated the presented approach in the context of two challenging scenarios with changing illumination conditions. In both scenarios we manually controlled the illumination condition in our laboratory through turning several lights on/off and/or activating/deactivating rolling shutters. This lead to the definition of two scenarios. First, a scenario with continous decreasing and increasing illumination conditions in the following called *continous scenario*. Second, a scenario with rapid illumination changes in the following called *discrete scenario*. In both scenarios a camera was placed in front off a wall labeled with Aruco markers (similar setup as for training see Fig. 2). For both scenarios we recorded an image stream of around 120 seconds which we later used for evaluation. It is important to note that the evaluation scenarios are different in terms of the concrete lightning situation and it's particular duration to the one used for training.

To evaluate whether the proposed adaptive approach is beneficial in terms of detecting more or fewer markers than idividual configurations we replayed both scenarios to the individual configurations of the Aruco marker detector (see Table I) and our adaptive approach. As seen in Table III the adaptive approach performs best (looking at the average value of detected markers) for the continous scenario and third for the discrete scenario. However, assessing solely the average performance is not beneficial as we expect that an adaptive approach is in particular beneficial in situations where a configuration needs to be selected in order to outperform all the other available configurations. Such a situation is visualized in Fig. 3 which shows an excerpt of the continous scenario. The first row depicts the illumination condition as an average grey-level pixel value over time. The second row shows the performance of the adaptive approach (number of detected markers over time) and the remaining rows show the performance of the individual configurations. In this particular situation only configuration E is able to detect markers which is also selected by the adaptive approach. Afterwards and with some delay the adaptive approach switches from configuration E to B. We also investigated the average adaptation time that is the time required to detect a performance violation, selecting a perception graph, and executing it. As seen in Table II for both scenarios the adaptation time is quite similar. For the continous scenario we switched 7 times the configuration and for the discrete scenario we switched 45 times the configuration.

Scenario	μ	σ	# switches
	seconds	seconds	
Continuous	≈ 0.57	≈ 0.23	7
Discrete	≈ 0.64	≈ 0.15	45

TABLE II: Timing behavior.

Scenario	Configuration	μ	σ
		# markers	# markers
Continuous	**Adaptive**	≈ 5.88	≈ 3.50
	A	≈ 1.32	≈ 1.38
	B	≈ 5.57	≈ 3.82
	C	≈ 3.45	≈ 3.14
	D	≈ 4.63	≈ 3.71
	E	≈ 0.39	≈ 1.33
Discrete	**Adaptive**	≈ 6.37	≈ 3.34
	A	≈ 0.73	≈ 1.04
	B	≈ 7.70	≈ 2.20
	C	≈ 3.56	≈ 2.86
	D	≈ 6.45	≈ 2.80
	E	≈ 0.00	≈ 0.03

TABLE III: Performance of the adaptive approach with respect to the individual configurations.

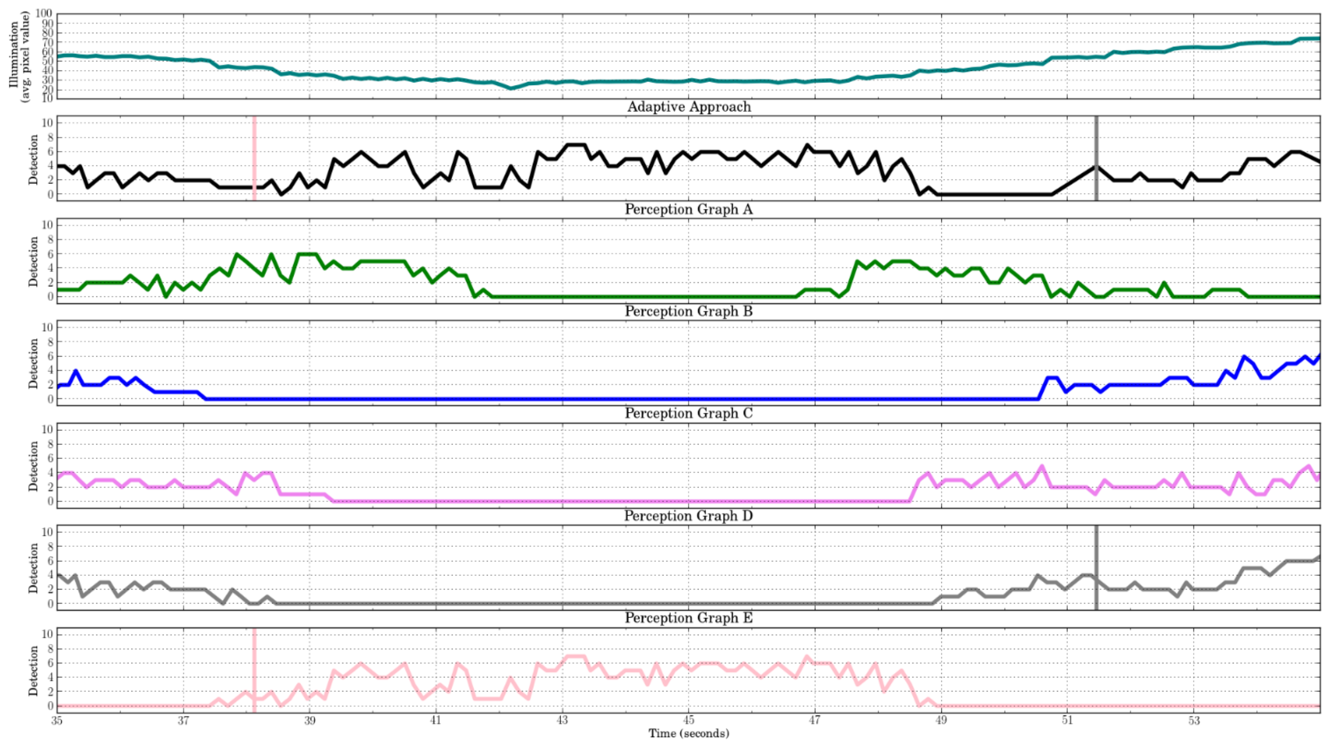

Fig. 3: Excerpt of the continous scenario showing the performance of the adaptive approach (second row) versus individual configurations. The colored vertical bars denote the point in time where a selected PG became active.

IV. Conclusion and Future Work

In this paper we presented our work-in-progress of an integrated adaptive robot perception architecture. We employed the RPSL to model perception graphs and we introduced a simple, yet powerful reasoning algorithm which selects an appropriate perception graph in the presence of context changes. The results are promising and supporting our future work to investigate more realistic robotic applications such as long-term navigation. To tackle these more challenging scenarios we will study how to detect and integrate more heterogeneous and combined context situations such as occlusions, sensor failures, and illumination in our approach. This also requires to investigate more advanced similarity measures incorporating several attributes representing heterogenous context conditions.

Acknowledgment

Nico Hochgeschwender is recipient of a PhD scholarship from the Graduate Institute of the Bonn-Rhein-Sieg University, which he gratefully acknowledges.

References

[1] D. Hall, "Automatic parameter regulation of perceptual systems," *Image and Vision Computing*, vol. 24, no. 8, pp. 870 – 881, 2006.

[2] J. L. Crowley, D. Hall, and R. Emonet, "Autonomic computer vision systems," in *International Conference on Computer Vision Systems (ICVS)*. Springer Verlag, 2007.

[3] O. Borzenko, Y. Lesperance, and M. Jenkin, "Invicon: A toolkit for knowledge-based control of vision systems," in *Computer and Robot Vision, 2007. CRV '07. Fourth Canadian Conference on*, May 2007, pp. 387–394.

[4] S. Moisan, J.-P. Rigault, M. Acher, P. Collet, and P. Lahire, "Run time adaptation of video-surveillance systems: A software modeling approach," in *Computer Vision Systems*, ser. Lecture Notes in Computer Science, J. Crowley, B. Draper, and M. Thonnat, Eds. Springer Berlin Heidelberg, 2011, vol. 6962, pp. 203–212.

[5] L. M. Rocha, S. Moisan, J.-P. Rigault, and S. Sagar, "Girgit: A Dynamically Adaptive Vision System for Scene Understanding," in *International Conference on Computer Vision Systems (ICVS)*, J. L. Crowley, B. A. Draper, and M. Thonnat, Eds., vol. 6962. Sophia Antipolis, France: Springer, Sept. 2011, pp. 193–202.

[6] N. Hochgeschwender, S. Schneider, H. Voos, and G. Kraetzschmar, "Declarative specification of robot perception architectures," in *Simulation, Modeling, and Programming for Autonomous Robots*, ser. Lecture Notes in Computer Science, D. Brugali, J. F. Broenink, T. Kroeger, and B. A. MacDonald, Eds. Springer International Publishing, 2014, vol. 8810, pp. 291–302.

[7] M. Fowler, *Domain Specific Languages*, 1st ed. Addison-Wesley Professional, 2010.

[8] S. Garrido-Jurado, R. Muñoz Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280 – 2292, 2014.

[9] P. Gärdenfors, *Conceptual Spaces: The Geometry of Thought*. Cambridge: MIT Press, 2000.