A Hierarchical System Integration Approach with Application to Visual Scene Exploration for Driver Assistance

Benjamin Dittes, Martin Heracles, Thomas Michalke, Robert Kastner, Alexander Gepperth, Jannik Fritsch, Christian Goerick

2009

Preprint:

This is an accepted article published in Proceedings of the 7th International Conference on Computer Vision Systems, Liége, Belgium, Octobre 13-15, 2009. The final authenticated version is available online at: https://doi.org/[DOI not available]

A Hierarchical System Integration Approach with Application to Visual Scene Exploration for Driver Assistance

Benjamin Dittes[1], Martin Heracles[2 1], Thomas Michalke[1], Robert Kastner[3], Alexander Gepperth[1], Jannik Fritsch[1], and Christian Goerick[1]

[1] Honda Research Institute Europe GmbH, Carl-Legien-Str. 30, 67073 Offenbach,
`<firstname>.<lastname>@honda-ri.de`
[2] CoR-Lab Bielefeld, Bielefeld University, Universitätsstraße 25, 33615 Bielefeld
`heracles@cor-lab.uni-bielefeld.de`
[3] Darmstadt University of Technology, Landgraf-Georg-Straße 4, 64283 Darmstadt
`robert.kastner@rtr.tu-darmstadt.de`

Abstract. A scene exploration which is quick and complete according to current task is the foundation for most higher scene processing. Many specialized approaches exist in the driver assistance domain (e.g. car recognition or lane marking detection), but we aim at an integrated system, combining several such techniques to achieve sufficient performance. In this work we present a novel approach to this integration problem. Algorithms are contained in hierarchically arranged layers with the main principle that the ordering is induced by the requirement that each layer depends only on the layers below. Thus, higher layers can be added to a running system (incremental composition) and shutdown or failure of higher layers leaves the system in an operational state, albeit with reduced functionality (graceful degradation). Assumptions, challenges and benefits when applying this approach to practical systems are discussed. We demonstrate our approach on an integrated system performing visual scene exploration on real-world data from a prototype vehicle. System performance is evaluated on two scene exploration completeness measures and shown to gracefully degrade as several layers are removed and to fully recover as these layers are restarted while the system is running.

Keywords: Visual scene exploration, System integration, Hierarchical architecture, Driver assistance

1 Introduction

In real-world scenarios, artificial intelligence systems face an overwhelming abundance of sensory data and high information density. One example is the domain of driver assistance, where camera images and range finder measurements are taken while driving on an inner-city road. Additionally, the amount of processing power available is limited. To handle this abundant data with limited resources, the system must select important positions in the input data space before launching a – usually computationally expensive – detailed analysis. Since our main focus is on driver assistance systems analyzing camera images we call this process visual scene exploration and the important positions 'visual targets'.

Several approaches for general visual scene exploration exist, presenting both algorithms (e.g. [1, 2]) and integrated systems (e.g. [3, 4]). However, the analysis of the integration process performed to arrive at such systems is very rarely done. It is much more common that specific systems are presented without a discussion about the benefits or drawbacks of the chosen integration approach. On the other hand, many existing vision systems are oriented strongly along the underlying software architecture, most notably blackboard[5], agent-based[6] or data-flow modeling[7].

Lömker et. al.[8] also see this problem of increasing system complexity and react with the introduction of a technical software environment on top a modern xml-based blackboard architecture[9]. Leibe et. al.[10] present a 3D scene analysis system with a strong decomposition and explicit top-down communication, but without formalizing it as a system integration approach.

In this work we will present a hierarchical system integration approach largely independent of the used software platform (necessary assumptions are discussed). It is based on the Systematica approach introduced in [11], a synthesis of the two major contributions to cognitive system architecture so far, the subsumption[12] and 3-tier[13] approaches. We go beyond this work by placing a strong restriction on the dependence of layers as the main decomposition principle. Furthermore, we go towards practical use with a generic mechanism to utilize top-down information while keeping independence at design- and run-time and a discussion of main design challenges when building practical systems.

We illustrate this approach on the problem of scene exploration in the driver assistance domain. As an integration problem, this is interesting because it does not simply require a sequence of processing steps producing a set of visual targets given the input data. Rather, there is a dualism between scene exploration (i.e. detecting important visual targets) and subsequent scene analysis (i.e. recognizing, tracking or storing these targets): based on a set of visual targets, the analysis of these targets may yield the necessity to tune parameters in the scene exploration sub-system. This may include setting parameters in other algorithms or directing the focus of the scene exploration to areas where important objects are expected to appear. This reflux of data from scene analysis to scene exploration is what we call top-down modulation and what makes system integration significantly more powerful, but also much more complex than sequential composition.

In Sec. 2 we will present our hierarchical system integration approach. We will then proceed to illustrate this integration process on the example of a concrete system instance performing visual scene exploration on a prototype vehicle in Sec. 3. An evaluation in Sec. 4 will show the behavior and positive qualities of this implemented system and discuss that this is a result of the integration approach. Discussion and outlook follow in Sec. 5.

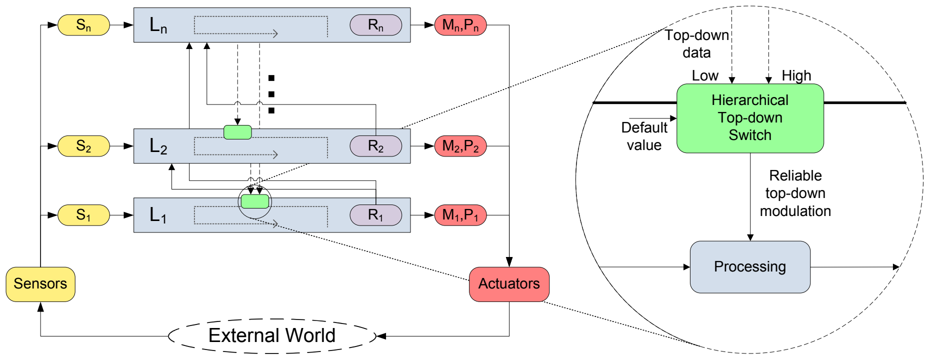

Fig. 1. The proposed hierarchical integration approach – **Left**: each layer L_i with internal processing runs in parallel and receives a sub-space S_i of the sensor data. It generates a representation R_i to be read by higher layers (solid lines), a motor command/priority pair M_i, P_i sent to actuators and top-down information sent to lower layers (dashed lines). Finally, it receives and integrates top-down information coming from higher layers (encircled box). **Right**: For each kind of top-down information received, the layer has to guarantee that it responds correctly *but does not depend* on the data. This is achieved by the 'top-down switch': a hierarchical switching module which chooses the valid top-down data from the highest layer and can fall back to a default if no top-down data is received.

2 The incremental architecture approach

For the domain of large-scale artificial intelligence systems we see several requirements an integrated system has to fulfill:

- The high processing load requires sub-systems to be distributed to multiple processing units, thus they must be able to run asynchronously in parallel
- To allow collaboration of many researchers building the system, sub-systems must have clear interfaces and should continue to work without modifications when new sub-systems are added
- Operating in a real-world domain requires a high level of robustness to unexpected input and errors within the system, which is why sub-systems must be able to continue to work even if some of the other sub-systems are removed to save resources or fail

We propose an approach decomposing the integration problem into hierarchically arranged layers $L_i, i = 1..n$. All layers run in parallel in an asynchronous manner and each layer can receive any subset S_i of available sensory data and produce motor commands M_i together with motor command priorities P_i to allow selecting among competing commands. Communication between layers is done through explicit channels: every layer L_i can access data and events produced in any lower layer $L_{j,j<i}$ through bottom-up channels and transmit data or events to any lower layer $L_{k,k<i}$ through top-down channels. Fig. 1(left) illustrates this schematically.

In addition to this structural definition, the main design principle defining our approach is the requirement that every layer L_i depends *only* on the presence of lower layers $L_{j,j<i}$ to function. As a result, a layer L_i may perform any or all of the following functions:

1. Receive a subspace S_i of all sensory data S
2. Access bottom-up information provided by lower layers
3. Synchronize to events emitted by lower layers
4. Provide a set of internal states, called a 'representation' R_i, to higher layers
5. Produce motor commands and priorities M_i, P_i and send to system actuators
6. Produce top-down information and send to lower layers
7. Allow modulation of the processing for items 4, 5 and 6 by top-down information received from higher layers

As a result, a layer can depend on and synchronize to lower layers, as well as use all their representations for it's own processing. Every layer is added to the system as an *extension*: relying on all existing lower layers, but making no assumptions on anything that might be added later, except providing 'ports' where top-down modulation may be received. Since the arrival of this top-down information cannot be relied on by any layer, a layer *must not* depend on the presence or arrival of information from higher layers or wait for events from higher layers. In the following we will discuss the main challenges in composing systems in such a way:

Influence of Layer Ordering. The key to success or failure of a system integrated with this approach is the hierarchical composition of layers. The most obvious constraint imposed by the approach is the unidirectional dependence of layers, but sometimes this is not enough to arrive at a unique ordering. Communication between two layers can occasionally be formulated either as top-down modulation or as access to bottom-up representations, making the ordering seemingly arbitrary. However, there are additional guides: Firstly, the ordering defines the final behavior of each of the possible running sub-systems $L_{1..i,i=2..n}$ consisting of only a sub-set of layers. Thus, if one layer L_j is necessary for the first intended sub-system to function while another layer L_k would add additional functionality to the next-higher sub-system, L_j must be sorted under L_k. Secondly, it is often prudent to operate lower layers at higher frequencies than higher layers in order to avoid too fast top-down modulation leading to oscillation. Finally, the failure of one layer may cause the failure of all higher layers, therefore more reliable layers should be sorted lower.

On System Design Methodology In addition to finding a compromise between the constraints on the layer ordering every system design is the results of a more of less suited problem decomposition. For most problems, including the example chosen in Sec. 3, decomposition is a problem in itself as there is no "natural" way to achieve it. We believe no system integration approach can, by itself, solve this problem in an automated methodology and problem decomposition will remain an art that requires as much skill as theoretical background. Therefore, this contribution focuses on the formalization and implementation of integrated systems given a decomposition and cannot provide a complete methodology. However, experience has shown that a formalization helps both in the process of designing by suggesting crucial questions to be answered as well as in the communication about the design by providing a shared language.

Handling of Top-down Information. To provide the demanded independence of layers, the most important requirement is that layers must be able to react to top-down information *without* depending on it. More precisely, there are three cases to be considered: first, a layer must be able to function if the layer(s) sending the top-down information are not present yet; second, it must use the received top-down information immediately after higher layers appear; third, it must return to default operation if higher layers are removed or fail. This is non-trivial since processing is asynchronous and the number of layers sending top-down information may not even be known while designing the current layer. We propose to solve all three by the help of a 'top-down switch' (Fig. 1(right)), a component which receives all top-down data of one type, analyzes their temporal behavior and returns the valid top-down data from the highest layer, or a default if none is valid. Validity is computed by comparing the input's time t_{data} to the typical temporal behavior $\mu_{\Delta t}, \sigma_{\Delta t}$ of data coming from the sending layer. Given a scaling factor ν, data is considered valid if $t_{\mathrm{now}} - t_{\mathrm{data}} \leq \mu_{\Delta t} + \nu\sigma_{\Delta t}$. This is similar to the 'suppression' mechanism introduced in the subsumption architecture[12], but it i) allows a clear definition of the top-down interface, ii) accepts an arbitrary number of top-down inputs and iii) adds a temporal adaptation mechanism.

We see three beneficial system-properties resulting from this approach:

- *Incremental Composition.* Higher layers can be added at run-time, thus providing top-down modulation or additional motor commands and increasing system performance without restarting the whole system. The new top-down channels usually lead to the formation of new internal control loops; it is the responsibility of the added layers to prevent any negative effects, e.g. providing top-down modulation at a lower rate to avoid oscillation.
- *Graceful Degradation.* The removal of a layer only affects the layers on top of it, all lower layers will continue to function and still provide an acceptable, albeit lower level of performance.
- *Reduction of Design Space.* Extending an existing integrated system by adding a new layer does not require modification of existing layers: new layers passively consume existing sensor inputs and representations of lower layers and they provide top-down modulation to (existing) ports of the existing top-down switching modules.

3 The visual processing hierarchy

To approach the problem of scene exploration we applied the above principles to an integrated system which employs several state-of-the-art methods. The system is designed with four layers (see Fig. 2): it combines saliency computation, drive path computation, visual target selection including inhibition of return and a parameterization with task-relevant modulatory data.

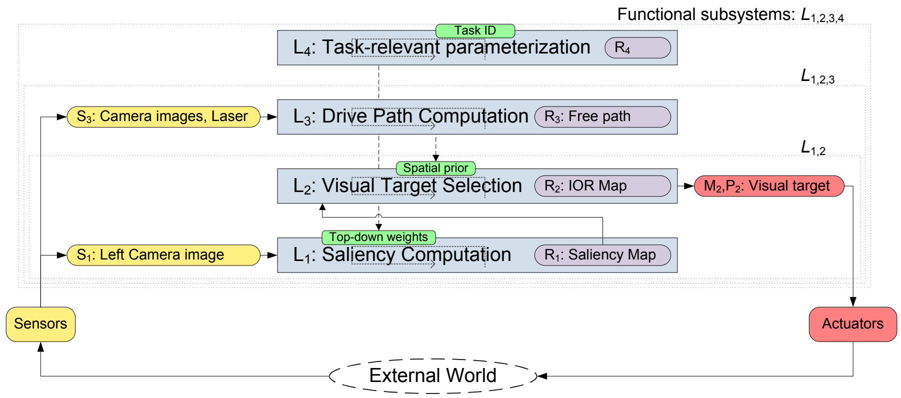

Fig. 2. Overview of the implemented hierarchical system – The figure shows the layers composing the implemented scene exploration system. It is composed of four layers performing saliency computation, visual target selection, drive path computation and task execution which communicate through the depicted bottom-up (solid) and top-down (dashed) channels. Top-down switching is done in three places: once for switching between top-down and default weights for the saliency, once for switching between top-down and default spatial prior for visual target selection and a third time for selecting the task id, although this switch is (currently) not receiving any top-down input. Since layer L_2 produces motor output, the first functional sub-system is $L_{1,2}$, followed by $L_{1,2,3}$ and $L_{1,2,3,4}$ (dotted boxes).

Saliency Computation. For the lowest layer L_1 we use an algorithm[14] which evaluates a multitude of image features (DoG, Gabor, RGBY, motion, . . .). It is tuned to real-world scenarios and allows setting weights to specify selectivity. The result is an image-sized, float-valued saliency map using either default weights, or the ones received by top-down modulation – therefore this is the first point in the system where top-down switching is needed.

Visual Target Selection. The core task of the scene exploration system is the selection of a visual target, performed in L_2. Here, we use the saliency map produced by L_1, multiply it with a top-down spatial prior and select the maximum as the current visual target. A top-down switching is needed to reliably integrate this top-down spatial prior under the constraint of loose coupling to the higher layers producing it. To get multiple visual targets per image, we apply an inhibition of return (IOR)[15] by subtracting a Gaussian at the last selected position from an IOR map. This map is provided to higher layers in case they need to focus processing on regions that have not been covered yet.

Drive Path Computation. In the car domain, one very helpful spatial prior can be extracted from the drive path in front of the car. The drive path itself is computed in L_3 by using segmentation on a number of sensors based on training regions in the camera images, combined with a temporal integration[4]. The result is a binary road map used to produce a spatial prior with a broad range of excitation around the edges of the drive path and very little on the path

a) b) c) d)

Fig. 3. Ground-truth information and internal processing results – The images show input and internal states of the system when processing one image with all layers running. a) Input left camera image with added ground truth information about target objects (turquoise boxes); b) saliency map computed in L_1 with top-down weights from L_4 *(for b,c,d: darker = higher activation)*; c) drive path spatial prior computed in L_3, with artificially highlighted drive path; d) integrated attention map with added resulting visual targets (red), computed in L_2, and ground-truth (green: hit, gray: not hit).

or in the sky. This spatial prior allows to focus on obstacles and the side of the road and is provided to L_2 as top-down modulation.

Task-relevant Parameterization. In addition to improving the spatial selectivity, a suitable set of top-down weights can greatly increase the performance of the saliency computation. Layer L_4 therefore receives a task identifier as a top-down input and gives a matching weight set (e.g. for cars) to L_1.

The described system has three points of top-down switching: one for receiving the top-down weights in L_1, one for receiving the spatial prior in L_2, one for settings the current task in L_4. All switches use the mechanism described in Sec. 2, without any specialization needed. All layers provide a default in case that no top-down information is received. In this manner, each layer is designed, implemented and started without caring about the subsequent higher layers.

4 Evaluation

The system described in Sec. 3 was implemented on a software infrastructure for data-flow oriented real-world capable processing systems[16]. The platform provides the main prerequisites for utilizing the benefits of the presented approach: Firstly, it allows sub-systems provided by researchers to be 'wrapped' into opaque larger modules with clear interfaces. Secondly, it allows these modules to be started or stopped in separate system processes, potentially on different machines, while providing robust data communication between these processes.

On this basis, each of the layers seen in Fig. 2 has been implemented as a module running in a separate process, communicating with the others through the software infrastructure. The final system is capable of running on-board a prototype vehicle but experiments for this work where done on a data stream (recorded on that vehicle) with additional hand-labeled ground-truth information. We chose a $30s$ data stream, recorded at 10Hz during a typical drive on an inner-city road, with typical lighting conditions and occasionally missing lane markings, on a cloudy day, but without rain (see Fig. 3 for samples).

When on-board the vehicle, the system is distributed on three 2-GHz dual-core computers, one performing sensor acquisition and preprocessing (disparity, pitch correction etc.), one with layers L_1, L_2 and L_4 and one with layer L_3. The most time-consuming layers are L_1 and L_3, both are currently running at 3Hz, which is enough to achieve interactive processing when driving with slow velocity. Layer L_2 is running with 12Hz to extract four visual targets per processed image[4], layer L_4 sends top-down information with a rate of 1Hz.

To evaluate system performance, we annotated the stream with ground-truth information, which can be seen in Fig. 3: for each image $I(t)$, all task-relevant objects $o_i(t) \subset I(t)$ where marked. The labeling is consecutive, i.e. the identifier i is used in all previous and following images for the same object. Thus, an object's appearance, movement through the stream and disappearance can be tracked. The scene exploration system produces a set of visual targets $v(t) = \{v_j(t), j = 1..n, v_j(t) \in I(t)\}$, in our experiments $n = 4$. We can define several related values: the first time an object i was marked $t_0^i = \arg\min_t o_i(t) \neq \{\}$, the first time an object i was hit $t_1^i = \arg\min_t \exists_j v_j(t) \in o_i(t)$, the number of marked objects $N^{\mathrm{obj}}(t) = |\{i, o_i(t) \neq \{\}\}|$ and the number of hit objects $N^{\mathrm{hit}}(t) = |\{i, \exists_j v_j(t) \in o_i(t)\}|$.

We now define two measures: the 'time-to-hit' indicates for each object o_i how much time passed between the appearance of the object and the first time it was hit: $q_1(o, v) = \left\langle t_1^i - t_0^i \right\rangle_i$. The 'local hitrate' indicates for each image how many of all objects in this image where hit by visual targets: $q_2(o, v) = \left\langle N^{\mathrm{hit}}(t)/N^{\mathrm{obj}}(t) \right\rangle_t$. Both together give a good impression of how early and how reliably the scene exploration is focusing on the important objects in the scene.

We evaluate both scene exploration performance measures once for the three functional sub-systems (see Fig. 2) and, most importantly, a fourth time for a full system run with induced shutdowns in layers L_3 and L_4 and subsequent restarts. The results of our experiments can be seen in Fig. 4 and caption.

5 Discussion and Outlook

In this work we introduced a novel, generic, hierarchical system integration approach for practical application in large-scale artificial intelligence systems. The system instance we used to show the benefits of the approach represents a functional, extendable effort towards visual scene exploration, incorporating several complex, state-of-the-art algorithms running asynchronously and communicating through explicit bottom-up and top-down channels. At the same time, it exhibits the described system properties: each layer implements one specific algorithm, depending only on lower layers; layers can fail or be deactivated at run-time

[4] Although L_2 is running faster than L_1 it cannot introduce oscillations because there are no top-down channels from L_2 to L_1.

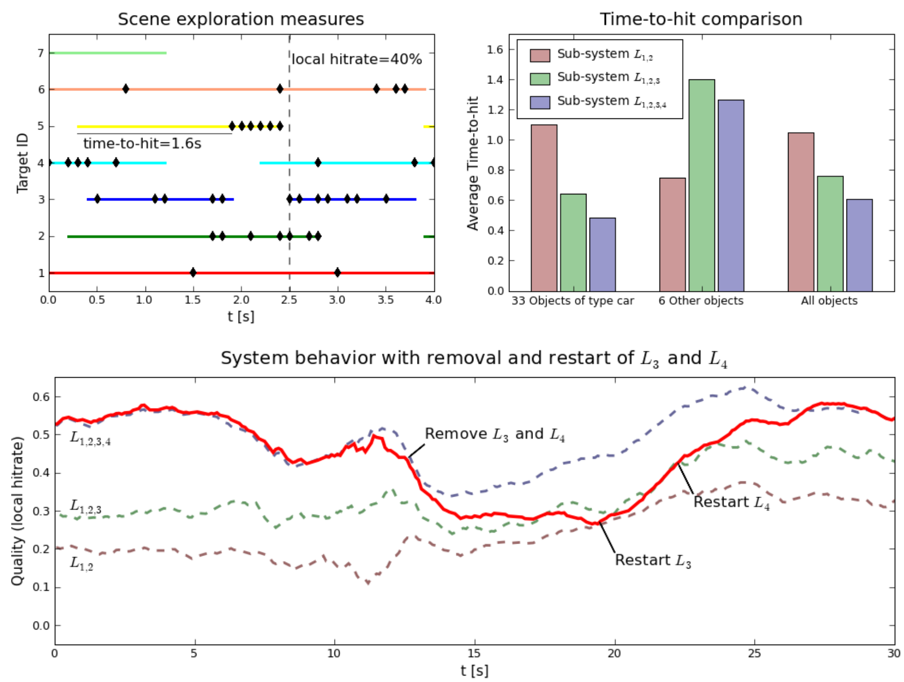

Fig. 4. Performance during several system runs with varying active layers – **Top Left:** Each of the target objects is plotted over time (colored bars). Black diamonds indicate when an object was hit by a visual target, in this case with subsystem $L_{1,2}$. The first measure (time-to-hit) can be seen in this visualization as the difference between the appearance of an object and the first hit, the second measure (local hitrate) evaluates how many of the objects at one specific point in time where hit. **Top Right:** The figure compares the time-to-hit performance over the entire data stream for the three possible functional subsystems. **Bottom:** The figure shows the local hitrate recorded over four separate runs of the system. The first three (dashed lines) where done with stable (sub-)sets of the full system: lowest $L_{1,2}$, middle $L_{1,2,3}$, highest $L_{1,2,3,4}$. Since the local hitrate shows considerable jitter in raw data, measurements have been averaged over $7s$. The fourth run (solid red line) demonstrates the system's ability for graceful degradation and recovery. The system is started with all layers, after $12.5s$ L_3 and L_4 are removed and restarted after $19s$ and $23s$, respectively. As can be seen, the system performance drops to the performance of the remaining functional sub-system $L_{1,2}$ after the removal (asymptotically, due to the floating average) and recovers as higher layers are restarted.

without taking down the full system; layers can be started and restarted at runtime leading to a full recovery of system performance (see Fig. 4 and caption). In order to utilize this robustness, we plan to add a 'quality of service' controller which autonomously shuts down layers to save resources and restart failed layers.

The introduced performance measures are specific to the visual scene exploration domain, resulting from the absence of any measures allowing to compare system integration strategies in general. For the same reason we cannot quantify the reduction of design or integration time following this approach, a comparison of identical teams using different approaches seems unrealistic.

It is also clear that no system integration approach will be able to give a generic solution to problem decomposition, which is why we discussed several design challenges in Sec. 2. However, we strongly believe that a discussion about the principles governing the construction of cognitive systems will enable learning from the success or failure of previous systems. It will thereby allow constructing more complex systems, or even allow a more straightforward comparison of existing systems as it is currently possible.

References

1. Frintrop, S., Backer, G., Rome, E.: Goal-directed search with a top-down modulated computational attention system. Lecture Notes in Computer Science **3663** (2005) 117
2. Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. IEEE Transactions on pattern analysis and machine intelligence **20**(11) (1998) 1254–1259
3. Marfil, R., Bandera, A., Rodriguez, J., Sandoval, F., Telecomunicacion, E.: A novel hierarchical framework for object-based visual attention. In: Intl. Workshop on Attention in Cognitive Systems, Springer London, Limited (2009) 27
4. Michalke, T., Kastner, R., Fritsch, J., Goerick, C.: A generic temporal integration approach for enhancing feature-based road-detection systems. In: 11th Intl. IEEE Conf. on Intelligent Transportation Systems. (2008) 657–663
5. Dodhiawala, R., Jagannathan, V., Baum, L.: Blackboard Architectures and Applications. Academic Press, Inc. (1989)
6. Shoham, Y., et al.: Agent-oriented programming. Artificial intelligence **60** (1993)
7. Abram, G., Treinish, L.: An extended data-flow architecture for data analysis and visualization. In: 6th Conf. on Visualization. (1995)
8. Lömker, F., Wrede, S., Hanheide, M., Fritsch, J.: Building modular vision systems with a graphical plugin environment. In: IEEE Intl. Conf. on Computer Vision Systems. (2006) 2–2
9. Wrede, S., Fritsch, J., Bauckhage, C., Sagerer, G.: An xml based framework for cognitive vision architectures. In: 17th Intl. Conf. on Pattern Recognition. (2004)
10. Leibe, B., Cornelis, N., Cornelis, K., Van Gool, L.: Dynamic 3d scene analysis from a moving vehicle. In: Conf. on Computer Vision and Pattern Recognition. (2007)
11. Goerick, C., Bolder, B., Janßen, H., Gienger, M., Sugiura, H., Dunn, M., Mikhailova, I., Rodemann, T., Wersing, H., Kirstein, S.: Towards incremental hierarchical behavior generation for humanoids. Intl. Conf. on Humanoids (2007)
12. Brooks, R.: A robust layered control system for a mobile robot. IEEE journal of robotics and automation **2**(1) (1986) 14–23
13. Gat, E., et al.: On three-layer architectures. Artificial Intelligence and Mobile Robots (1997)
14. Michalke, T., Fritsch, J., Goerick, C.: Enhancing robustness of a saliency-based attention system for driver assistance. Lecture Notes in Computer Science (2008)
15. Klein, R.: Inhibition of return. Trends in Cognitive Sciences **4**(4) (2000) 138–147
16. Ceravola, A., Stein, M., Goerick, C.: Researching and developing a real-time infrastructure for intelligent systems - evolution of an integrated approach. Robotics and Autonomous Systems **56**(1) (2007) 14–28