ANTS – Intelligent Vision In Urban Traffic

Steffen Görzig, Uwe Franke

Abstract— **The growing complexity of intelligent vision systems increasingly requires software architectures, which can deal with several perception modules running in parallel and different applications using these modules.**

In this paper a multi-agent system (MAS) is presented that has primarily designed as a software architecture for controlling multiple software modules. The described "Agent NeTwork System" (ANTS) offers components for several purposes: the integration and co-operation of various modules, distributed computing, the development of applications and test environments for system components.

The ANTS architecture consists mainly of two components: modules and administrators. Modules are the computational entities of the system. They can perform certain tasks, e.g. lane tracking. The modules are controlled by administrators. An administrator continuously determines the modules, which are currently necessary to achieve a specific application.

The first application we are working on is an autonomous Stop&Go driving with our UTA (urban traffic assistant) demonstrator. For this, ANTS administrates the computer vision tasks, vehicle control and driver interface modules.

Keywords— **intelligent vision, software architecture, autonomous vehicle guidance, multi-agent system**

I. Introduction

In the past great efforts have been made for driving autonomously on highways. For example, in 1986 Dickmanns [1] demonstrated autonomous driving on highways at speeds up to 100 km/h. Also at the final presentation of the European PROMETHEUS project our VITA II demonstrator showed fully autonomous driving including planning and performed overtaking manoeuvres [2]. The very promising results encouraged us to go a step ahead and enter the much more complex urban environment for computer vision applications. The first application we are working on is an autonomous Stop&Go driving with our UTA (urban traffic assistant) demonstrator[3]. Driver assistance systems in this scenario are challenging not only from the algorithmic but also from the system architecture point of view.

The architectures of most autonomous driving systems are usually tailored to a specific application, e.g. lane keeping on highways. The functionality is achieved by a few computer vision and vehicle control modules, which are connected to each other in a hard-wired fashion. Although this kind of architecture is suitable for a lot of applications, it also has some disadvantages:

- The architecture is not scaleable for a larger number of modules.
- There is no uniform concept for the co-operation of modules (e.g. for sensor fusion).
- New applications usually require extensive reimplementations.
- Reuse of old modules can be difficult due to missing interfaces.

The growing complexity of autonomous systems hence reinforces the development of architectures, which can deal with the following requirements:

- integration and co-operation of various computer vision algorithms
- different abstraction level of action/perception/control
- sensor fusion
- economical use of resources
- test environments for system components
- integration of new algorithms without a complete redesign of the system
- simple enhancement to new computer vision applications
- distributed computing.

To meet these requirements, several architectures were proposed, for example in [4] and [5]. The approach described in this paper is a multi-agent system for intelligent vision control, which allows UTA (urban traffic assistant) to perform Stop&Go driving. The "Agent NeTwork System" (ANTS) administrates computer vision, vehicle control and driver interface processes on three PowerPC 604e and a Pentium II PC to achieve this application. ANTS integrates several vision algorithms for various image understanding tasks, e.g. stereo based object detection, lane detection, pedestrian recognition, traffic sign, and traffic light recognition. It selects and controls these algorithms and focuses the resources on relevant tasks for specific situations. For example, there is no need to look for new traffic signs or continuously determine the lane position, while the car is slowing down in front of a red traffic light.

II. Multi-Agent Systems

AGENT software is a rapidly developing area of research. Since heterogeneous research is summarised under this term there is no consensus definition for the word "agent". A working definition of a multi-agent system (MAS) can be defined as "a loosely-coupled network of problem solvers that work together to solve problems that are beyond their individual capabilities"[6]. The smallest entity of a MAS is an agent. An agent could be described as a computational entity, which provides services and has certain degrees of autonomy, co-operation and communication.

With this definitions it is not difficult to see how a MAS can be applied for autonomous vehicle guidance. Each computer vision or vehicle control module contains some functionality which can be useful for autonomous driving. The capabilities of these modules have to be combined to allow more complex applications like autonomous Stop&Go driving in urban environments. The idea is to add the missing autonomy, co-operation and communication to the modules to create a MAS.

S. Görzig and Uwe Franke are with the Daimler–Benz Resarch, Image Understanding (FT3/AB) , HPC T728, D–70546 Stuttgart, Germany. E-mail: {goerzig,franke}@dbag.stg.daimlerbenz.com

III. ANTS – Components

THE system is completely written in C++. The main components of ANTS are (see also figure: 1):

- Data base
- Administrators
- Communication interfaces
- Modules
- Servers

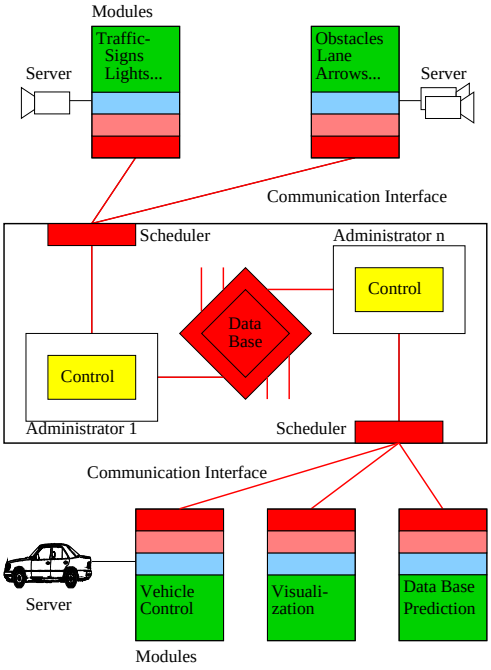

Fig. 1. ANTS – Components.

The modules are the computational entities of the system, whereas the administrators contain the autonomy and the co-operation of the software agents. Combined with the communication interface they build the MAS as described above. This distinction between a computation component and the "intelligent" component of an agent has several advantages:

- The components can be executed parallel
- Existing modules can be used
- The modification of a component does not necessarily require a modification of the other components

Although ANTS is a MAS for various applications, we will now focus on the Stop&Go driving with UTA (see figure 1) and explain the main components more detailed.

A. Data Base

The central component of ANTS is the data base. It contains all incoming and outgoing data of the modules. This data is typically on a symbolic level and can be used e.g. for driver information or for the co-operation of modules. For instance the results of a lane recognition module and an obstacle detection module are useful for the lateral and longitudinal vehicle control module.

The data base has several access methods for its data like exclusive write and concurrent read. For example, the stereo obstacle module can access its symbolically represented results in the exclusive write mode to track old and detect new obstacles, whereas the visualization module can access the results of all computer vision modules concurrently to submit important information to the driver.

The different computer vision modules provide 2D as well as 3D information about the environment. This information can be visualized in a driver interface module (see figure 2). With the vehicle sensor data the position and relative motion states of the 3D objects can be estimated by means of Kalman filtering.

This prediction is useful to set the region of interest for computer vision modules. It generates also a good basis to make decisions about following steps and allows to update data when other image processing tasks are more important at the moment. The prediction is done by a data base prediction module, which can be treated like any other module.

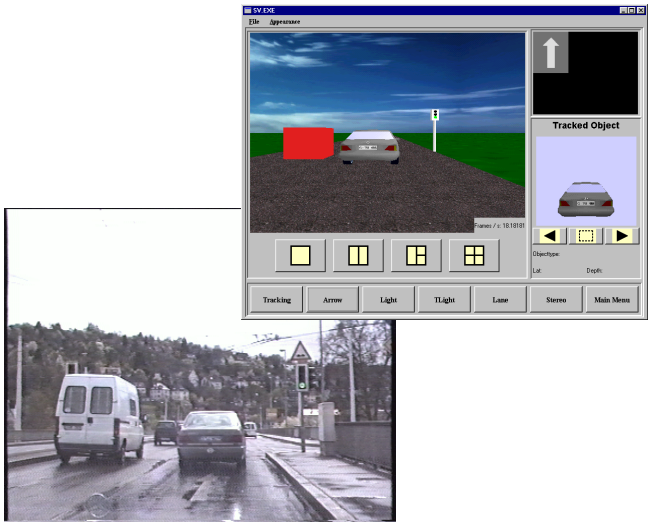

Fig. 2. Visualization of the computer vision results.

B. Administrator

An administrator controls a set of modules (see figure 3). The administrator has to choose the modules, which have to be executed in the next step, and give them to the scheduler.

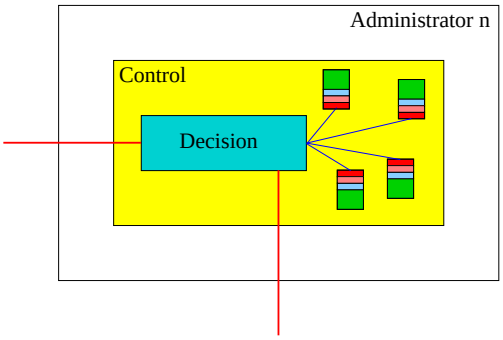

Fig. 3. Administrator.

The actual module selection is done by a decision com-

ponent within the administrator control. This control component is the core "intelligence" of the modules. The result of the decision depends on the current situation, i.e. the current data base entries. The traffic light recognition and the arrow recognition modules, for instance, can be switched off, when the vehicle is driving on a highway. Active modules can be ignored, as well as other modules, which have to wait for results of other modules or which are less important than others at the moment. It is also possible to have more than one module for a certain task: for example, a fast but unprecise obstacle tracking and a slower but more precise one. When the position of a tracked object is well known, the fast variant is used; otherwise, the slower one is more likely to be called.

Within the module control component, various methods for choosing modules can be implemented. The methods can be compared to each other by replacing the control component (see figure 4). While the currently implemented method uses a simple priority mechanism, there is ongoing work on more advanced AI methods like distributed planning, rule based reasoning, or neuronal networks.

The chosen modules are submitted to the scheduler. The scheduler communicates with the modules and executes the module tasks on the specified computational nodes. Due to the real-time constraint, all processes have already started waiting for messages from the administrator.

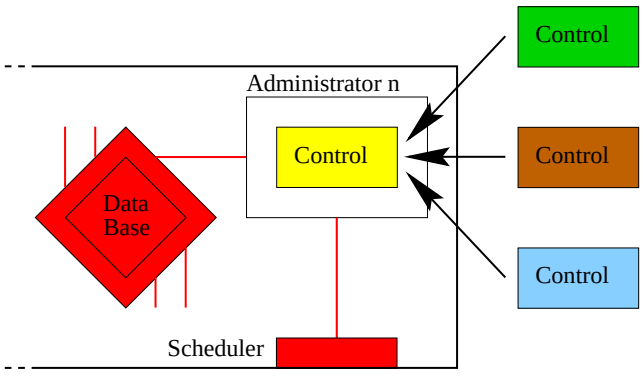

Fig. 4. Different controls can be used in a administrator.

ANTS can handle several administrators. Modules, which do not belong together, are assigned to different administrators. This is useful to avoid complex decision components. In UTA, for example, one administrator controls the computer vision modules, another one the vehicle control, and a third one the driver interface modules.

An administrator can also handle exceptions in modules. A critical error within a module is submitted to the administrator. The administrator can now decide to stop the entire system, or just disable the affected module, in the case that it is not needed for safe vehicle guidance or multiple modules are available for the same task.

C. Communication

The communication within ANTS is done with a C++ class library for message passing. The library is able to send and receive C++ objects and is based on the parallel virtual machine system PVM [7]. PVM is a portable, easy-to-use message passing tool written in C. Since the submitted data in ANTS is usually on symbolic level, PVM is fast enough for real-time applications. With PVM a set of heterogeneous computers can be combined into just one large virtual machine, which allows us to use three AIX/PowerPCs together with a Windows/Pentium II computer.

D. Modules and Servers

Each module in the system (like the vehicle control module) represents a computational entity. They can be distributed on several computational nodes and are handled uniformly by the system via the communication interface. The modules consist of a functional part and the system part (see figure 5).

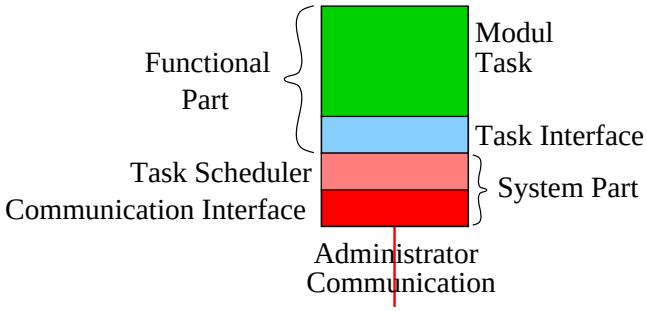

Fig. 5. Parts of a module.

The system part communicates with the administrator and has a small scheduler to call the functional part if requested by the administrator. This part also catches occurred exceptions in the functional parts and sends them to the administrator. The task interface defines all incoming and outgoing data for a module task and all communication between the parts is done via this interface. This is useful in many ways:

- existing software can be reused
- the developer of a module (e.g. a lane tracking) doesn't have to care about the ANTS components like message passing
- the developer of an administrator doesn't have to know the innermost details of the modules
- this separation allows test environments as described below

Modules can also obtain data from other modules via the data base and its task interface. This allows to implement sensor fusion modules, for example, to fuse the results of a stereo image obstacle detector and a radar obstacle detector.

Servers can also be accessed via the task interface. Servers are components, that can be used by more than one module task. There is, for example, a time server which provides a unique time on each computational node. For the computer vision modules, a stereo camera system and a colour camera can be accessed. There is also a vehicle interface to read the sensor data and to set the actuators.

IV. Test Environments

COMPLEX parallel applications inherently have many sources of errors. The range of errors covers unexpected system behaviour up to critical defects and can be caused by each component or the co-operation of components. Thus special attention should be paid to the test environment of the system.

The separation of the modules into the functional part and the system part with the task interface allows a replacement of one part without a modification of the other one. This can be used to test modules as well as to test the behaviour of the administrators.

A. Testing Modules

A standalone version of a module can be created by replacing the system part of a module with a test task (see figure 6). This is useful not only for module debugging, but also for the integration of new modules into ANTS.

The first step to add, for example, a traffic sign recognition software, is the definition of the task interface. After that, a standalone module version is created. The developer can now test and debug the traffic sign software as usually, without taking the rest of the ANTS components into account. As practical experiences show, little efforts have to be made to create a module without having multiple source codes of a module task. The traffic sign recognition, for example, can still be compiled without the task interface and the task scheduler. This facilitates the maintenance of the software. Once a standalone module has reached a stable version, it can be added to the MAS by replacing the test task scheduler by the system part.

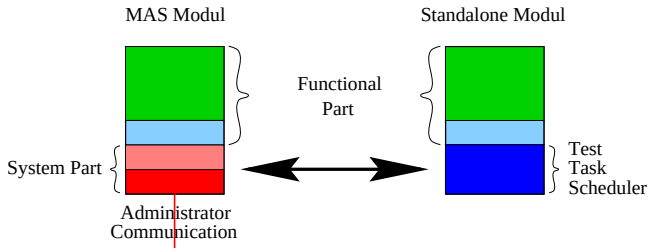

Fig. 6. Test environment for modules.

There are also several possibilities to test a module when it is already connected to the system. Each module of ANTS can be started within a debugger. This allows a detection of errors caused by the co-operation of components. As described above, critical exceptions of a module are caught and submitted to the administrator. The administrator is than able to decide how to handle the error. For developers of modules, it is often necessary, to inspect internal information during runtime. It is not practical to allow all modules to print this information at the same time. Thus each module has a debug level, which can be interactively incremented, for example, to get additional information about a computer vision algorithm.

B. Testing Administrators

The task interface of a module also allows a replacement of the module task without the modification of the entire system (see figure 7). This is used to test the system core, i.e. the behaviour of the administrators.

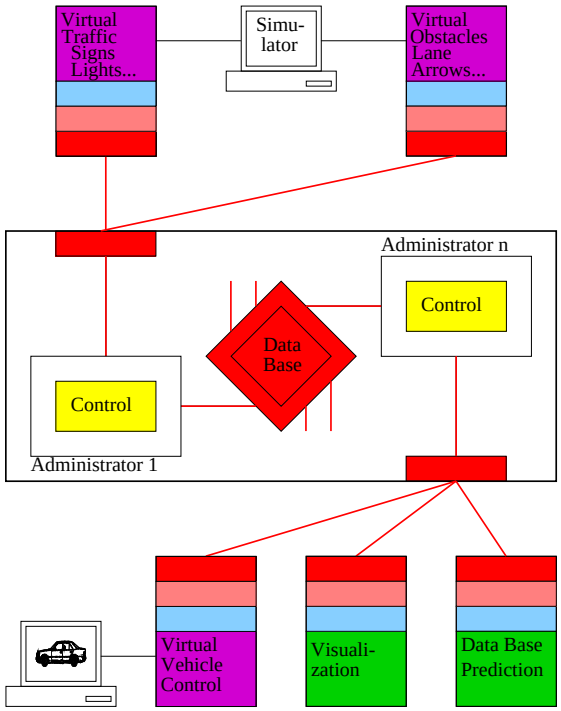

Fig. 7. ANTS Simulator environment.

As described in III-B, various methods for the module control can be implemented in an administrator. To obtain a fair comparison between these methods, it is necessary to create equal environment conditions. This is very difficult for complex computer vision applications. The solution is to replace the computer vision and vehicle control modules by a urban traffic simulator (see figure 7).

The urban traffic simulator generates typical traffic scenes, including cars, pedestrians, bikes, traffic lights and signs etc. (see figure 8). A car with the steering behaviour of an S-Class Mercedes interacts with the environment by virtual symbolic sensors and actuators. These sensors allow to perform virtual perception tasks. The virtual tasks replace the real computer vision module tasks by simulating their abilities without image processing.

The simulator allows to test the behaviour of the system in various scenes with comparable results for different control methods. Moreover the simulator can help to evaluate questions like this: How does the system react with a smaller camera lens angle? Can the system benefit from a new, in real life non-existent, perception module? What applications are possible by adding a new sensor, e.g. a radar?

V. Results

AUTONOMOUS Stop&Go driving with our UTA demonstrator is the first application we are working on in inner city environments. In the Stop&Go application,

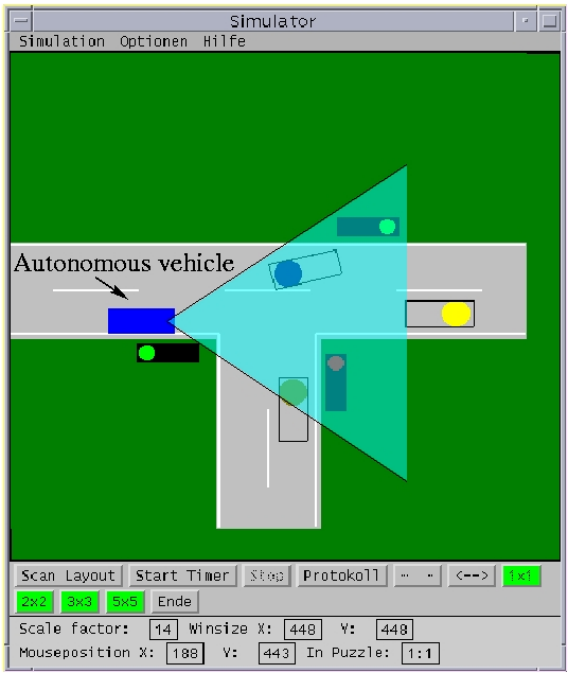

Fig. 8. Urban traffic simulator.

the driver can accept a vehicle ahead, which has been recognized as the 'leading vehicle' by the vision system. UTA follows this vehicle autonomously. There are several conditions which will end the autonomous driving. For example, if the leading vehicle leaves its lane, a red traffic light is detected, or if the driver switches the application off.

The modules to be controlled by ANTS are:

- stereo-based object detection and tracking (see [8])
- lane detection and tracking (see [9])
- traffic sign and traffic light recognition (see [10])
- recognition of road markings (direction arrows, see [3])
- crosswalk recognition (see [9])
- vehicle classification
- overtaking vehicles detection
- pedestrian recognition (see [11])
- vehicle control (lateral/longitudinal)
- driver interface (visualization)
- prediction of data base entries

So far, not all of the modules have been completed and integrated into the system. Currently, UTA is able to recognize traffic lights and direction arrows and to follow autonomously a leading vehicle. The next steps are to include the missing modules, and to adapt the system to multiple environments, i.e. to use the system for inner city traffic as well as for autonomous driving on highways. The system should be able to dynamically switch the autonomous driving applications and the modules to achieve these tasks.

Besides autonomous Stop&Go driving, the ANTS architecture allows to add other standard applications in the future:

- speed limit assistant: the driver is warned when driving faster than allowed on the current road.
- lane departure warning: leaving the lane on highway causes a warning message.
- enhanced cruise control: the vehicle slows down, when a car in front falls below the desired speed. It raises the speed again, if there is no obstacle ahead.

VI. Conclusion

INTELLIGENT vision systems will be applied to increasingly complex tasks in the future. The growing complexity of these system will require software architectures, which can cope with several perception modules and applications.

The described MAS architecture offers components for this kind of distributed programs. The "Agent NeTwork System" allows a continuous enhancement of the system by adding modules and implementing new applications. With the task interface of a module, it is also possible to reuse old software. The architecture also simplifies sensor fusion and enables the perception modules to contribute to various intelligent vision applications.

Due to the complexity of large distributed applications, special efforts have to be made for test environments. The ANTS modules can be tested and debugged as a standalone version as well as in combination with the entire system. To test the behaviour of the system, the module tasks can be replaced by virtual tasks. This allows to compare different module control strategies of the administrators.

The ability of the "Agent NeTwork System" to control cooperating modules can be applied for autonomous driving in complex scenarios. The first application is autonomous Stop&Go driving in an urban environment.

References

[1] E. D. Dickmanns and A. Zapp, "A curvature-based scheme for improving road vehicle guidance by computer vision," in *SPIE Conference on Mobile Robots*, 1986, vol. 727, pp. 161–167.

[2] B. Ulmer, "VITA II - Active collision avoidance in real traffic," in *Intelligent Vehicles '94, Paris*, Oct. 1994, pp. 1–6.

[3] U. Franke, S. Görzig, F. Lindner, D. Mehren, and F. Paetzold, "STEPS TOWARDS AN INTELLIGENT VISION SYSTEM FOR DRIVER ASSISTANCE IN URBAN TRAFFIC," in *IEEE Conference on Intelligent Transportation Systems*, Nov. 1997, pp. 601–606.

[4] Dirk Reichardt, *Kontinuierliche Verhaltenssteuerung eines autonomen Fahrzeugs in dynamischer Umgebung*, Ph.D. thesis, Universität Kaiserslautern, Jan. 1996, Forschung F1M/IA Daimler–Benz.

[5] M. Maurer and E. D. Dickmanns, "A SYSTEM ARCHITECTURE FOR AUTONOMOUS VISUAL ROAD VEHICLE GUIDANCE," in *IEEE Conference on Intelligent Transportation Systems*, Nov. 1997, pp. 578–583.

[6] G. O'Hare and N. Jennings, Eds., *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, 1996.

[7] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidyalingam S. Sunderam, *PVM: Parallel Virtual Machine A Users' Guide and Tutorial for Network Parallel Computing*, MIT Press, 1994.

[8] U. Franke and I. Kutzbach, "Fast Stereo based Object Detection for Stop&Go Traffic," in *IEEE Conference on Intelligent Transportation Systems, Tokyo*, Oct. 1996, pp. 339–344.

[9] F. Paetzold and U. Franke, "Road Recognition in Urban Environment," in *IEEE Conference on Intelligent Transportation Systems*, to be published Oct. 1998.

[10] W. Ritter, F. Stein, and R. Janssen, "Traffic Sign Recognition Using Colour Information," in *Math. Comput. Modelling, No. 4-7*, 1995, vol. 22, pp. 149–161.

[11] C. Wöhler, J. K. Anlauf, T. Pörtner, and U. Franke, "A Time Delay Neural Network Algorithm for Real-Time Pedestrian Recognition," in *IEEE Conference on Intelligent Transportation Systems*, to be published Oct. 1998.