Vision Technologies for Intelligent Vehicles

M. Bertozzi, A. Broggi, L. Bombini, C. Caraffi, S. Cattani, P. Cerri, A. Fascioli, M. Felisa, R. I. Fedriga, S. Ghidoni, P. Grisleri, P. Medici, M. Paterlini, P. P. Porta, M. Posterli, and and P. Zani

VisLab — www.vislab.it – Università di Parma, ITALY

Abstract. This paper surveys the different technologies involved in the development of computer vision applications for Intelligent Vehicles derived from the 15 year experience of VisLab. Some illustrative examples are also discussed.

Key words: machine vision, intelligent vehicles, intelligent transportation systems

Introduction

The use of vision for Intelligent Transportation Systems developed in the last 20 years and, after the first experiments, rapidly evolved and is now widely considered as one of the most convenient perception technologies. The first experiments were performed in the second half of the eighties but it took some years before different prototypes of vision-based autonomous vehicles were tested in real situations. In the last few years, initiatives like the Grand Challenge and Urban Challenge caused a novel flourishing of research activities in the autonomous vehicles field [2].

In order to effectively develop vision based systems for intelligent vehicles both hardware and software issues have to be considered.

The different technologies involved in the development of these systems at VisLab, University of Parma, are described in the following sections: hardware and software.

The hardware side affects both the sensing technologies, namely the cameras, and the computing engine. Different camera technologies can be exploited: daylight, far infrared (thermal), near infrared, or even range cameras; the choice mainly depends on the specific application and on a cost vs benefit analysis. Also in the case of the computing engine, the potential choices highly depend on the application and on computational power constraints. In fact, in the intelligent vehicles field, real time performance is generally mandatory. Few years ago, dedicated hardware was designed to obtain enough computational power to fulfill this constraint. Nowadays, standard off-the-shelf components, namely standard PCs, can be used since they deliver enough power. Anyway, the technology transfer of these systems towards a mass market production often requires to minimize size and power supply requirements of the computing engine, therefore also embedded systems like DSPs or FPGAs have to be considered.

On the software side, two different issues have to be examined: the development phase and the final system software. In order to enable the programmers to focus on the specific vision problem without having to care about other details, a rapid application development tool have to be used. In the last 10 years, VisLab developed such a tool (called GOLD) which provides programmers with a rich set of functionalities and APIs for commonly required tasks like I/O operations (acquisition from cameras and other sensors as well, sensors synchronization, local net and file management, data graphical display. . .) or low-level image processing functions. A convenient graphical user interface is also used to support programming activities. Anyway, for some applications, such a software environment is not always suited to be used in the final system especially in the case of small or embedded architectures. Therefore, particular care has been used in designing the software internal structure: in fact, GOLD is structured in different logical layers, each one devoted to a specific task. Thanks to a great degree of independence between the different layers, the software can be easily shrunk removing functionalities and layers not necessary to the specific application.

In order to show how these technologies dwell in the real world, this paper also includes the description of some real applications characterized by different requirements. In the case of the Grand and Urban Challenges the huge number of functionalities and sensors require to use a powerful computing system made of multiple PCs. Conversely, in the case of the Start Inhibit functionality, after an initial development phase in which a standard PC was used, embedding constraints require to port the system towards a DSP architecture. A third example (Boat Speed Monitor) completes the picture by showing the capabilities of the GOLD software: GOLD has been adapted removing unnecessaries functionalities and the porting towards an ARM-based architecture embedded into an intelligent camera is under development.

This paper is organized as follows: the next section illustrates the different camera technologies and computing engines that can be used for artificial vision while section 2 details the functionalities and structure of the software development framework used at VisLab. Section 3 ends the paper discussing examples of three different systems developed by VisLab researchers.

1 Hardware for vision-based Intelligent vehicles

In the following, camera technologies and computing architectures used on board of vision-based intelligent vehicles are discussed.

Camera technologies. Images acquired in different spectral domains have different characteristics and bear different costs.

In the far infrared (FIR) domain the image of an object relates to its temperature, namely on its emitted heat and barely depends on illumination. In addition, FIR images lacks of textures and colors. Therefore, FIR cameras are suitable for the detection of objects warmer (or colder) than the background

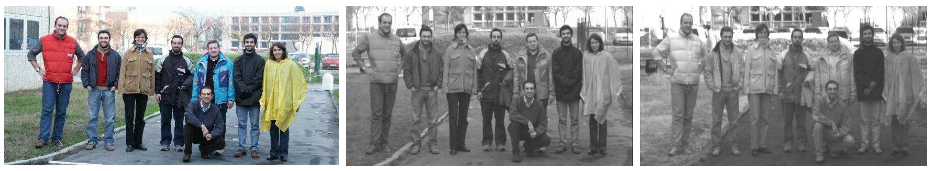

Fig. 1. Comparison amongst color, grey level, and NIR images.

and ease the initial steps of a detection process thanks to the absence of small details.

Conversely, object detection in the visible domain is often more difficult due to the presence of details, shadows, and changes in the luminance or sharpness of images. Anyway, small details are of paramount importance for object classification. Moreover, the use of color can be important in some applications, such as traffic signs detection.

Another interesting technology is near infrared (NIR). Objects reflect incident NIR radiation very similarly to visible light and the appearance of objects depends on how the material reflects the NIR radiation. Clothings generally reflect well the NIR light, therefore exhibit less grey-level variability in the image, although some fabric absorb all NIR light. A comparison between the appearance of clothings in color visible images, grey level visible images, and NIR images is shown in figure 1.

Daylight and NIR cameras generally provide larger resolutions than FIR sensors and are cheaper and smaller.

Recently, alternative sensing technologies are being considered. For example range cameras can acquire depth information. This technology is quite new and, unfortunately, current sensors feature a very low resolution and a limited working range. Anyway, many research activities (like in the EU project PReVENT) are trying to overcome the current limits of this technology.

Computing engines. At the beginning of the 90s, dedicated and expensive hardware were still mandatory for artificial vision. Just a few years later COTS (Commercial Off-The-Shelf components, like standard PCs) started to be used as a computing engine for vision system, since they were able to deliver enough computational power.

As an example, in 1996 lane and obstacle detection took approximately 75 ms to be executed on dedicated system (the PAPRICA multi-processor architecture). Only three years after, in 1999, a 200 MHz Pentium Processor was capable to complete the same task using the available MMX instructions [1].

Nowadays, PCs are able to handle much more complex tasks. As an example, in the Grand and Urban Challenges a huge list of vision based functionalities are performed by PCs exploiting modern CPUs features; multithreading oriented algorithms are able to use multi core CPUs and small portion of codes are optimized using assembly multimedia instructions. Moreover, also the modern graphics processing units, specifically designed to perform image processing, can be exploited as a coprocessor for image processing tasks.

Unfortunately, PCs are not suited to fit into small embedded systems; in such a case devices like FPGAs and DSPs are used. They require the porting and adaptation of algorithms.

2 The GOLD software framework

GOLD is a framework aimed at providing a complete set of tools to allow fast development of computer vision applications. Moreover, GOLD is also used to run the on-board final system.

Functionalities. GOLD allows to build vision-based applications as independent plugins. Each application can take advantage from an increasingly rich set of tools and debugging facilities. These components are described in the following.

GOLD frees the developer from using the real hardware, offering an abstraction layer over real devices. Various classes of devices are currently supported, several cameras, laserscanners, radars, and inertial systems. Data can be acquired from multiple sensors with or without synchronization. This feature allows to easily develop data fusion oriented algorithms. Moreover, virtual devices are also implemented to use pre recorded data. GOLD also allows to preprocess acquired data performing common operations (i.e. image stabilization, distortion removal. . .) and sharing the results among all the active plugin applications.

Developers can also easily produce highly interactive and consistent user interfaces, through the use of a set of widgets specifically designed to seamlessly integrate with the processing code. A windows subsystem provides a set of simple and powerful drawing primitives. These can be used to display intermediate and final results during the debugging phases or as the final result. This system hides the complexity of the underlying graphics API, and thus ensures portability and ease of use. The window subsystem itself supports many target platforms (OpenGL, X11, SVG. . .). Hardware acceleration is used to reduce the CPU load and boost the performance of the visualization system.

Developers may use GOLD to record data sequences from multiple data streams in order to allow a later playback for in-lab development. Captured data can be stored in different formats, depending on their nature and on the application constraints. GOLD allows advanced playback capabilities (such as looping, stepping, jumping, moving forward or backwards at a given frame rate and setting bookmarks), which guarantee high flexibility for the development and testing of algorithms. During playback, acquired data are fed to the application through the same APIs used for acquiring from real devices. Recorded data contain an index file with a timestamps for each data. This additional file is used to cope with problems involving time, such as tracking and motion stereo, allowing a time based mode of replay. In this case, playback is performed considering the actual processing time and therefore emulating the system behavior in real working conditions.

GOLD architecture. The GOLD architecture has been conceived to ensure scalability and independence among its components. The whole system is organized into several subsystem layers, which provide the different functionalities to the developers. Thanks to a cleanly defined interface, subsystems can be easily added or removed depending on the specific needs of the target project. Also the vision-based applications are seen as plug-ins by the framework, and can be easily removed or ported to different platforms.

Each software layer has been developed to achieve the maximum reusability and performance, often using machine-code level optimizations; multithreaded code has also proven very effective in boosting speed (up to 40%) of CPU-intensive tasks on multi-core machines and coping with I/O latencies.

The hardware abstraction layer masks the complexity of input devices to the programmer. Sensors are organized in a tree structure: the first level contains the set of different device classes (cameras, laserscanners or inertial measurement units) while the second level manages the connected devices of each set. Applications can traverse the tree to the available devices. The tree can also be dynamically reconfigured to take in account device hotplug or malfunctions.

A second software layer for image processing enables programmers to use primitives that cover many fields of computer vision: low-level image processing and stabilization, features tracking, perspective mappings, stereovision...

A graphical user interface layer has been developed to provide graphic widgets and controls for interacting with applications allowing programmers to forget about available graphic backends. A rich set of APIs allow programmers to output final or partial results, producing graphics with very low resource consumption, thanks to its multithreaded client-server architecture.

Porting to embedded architectures. DSPs and FPGAs feature different constraints and capabilities than a modern PC and the code must be rewritten and/or best-managed to run on such architectures. As an example of major steps to be performed, floating point must be converted to fixed-point and conditional loops need to be converted to operations that can better exploit DSP pipeline. External RAM accesses are slow and algorithms have to efficiently manage the small internal RAM and L2 cache.

Proprietary compiler for embedded systems often provide an efficient way to develop software, regardless of compatibility to language standard. Anyway, even such compilers are not able to reach the same code optimization of a skilled assembly programmer. Therefore, critical portions of the code have to be manually implemented.

3 Examples of vision-based systems

Vision technologies for intelligent vehicles can be employed in a number of different applications that present different constraints. In the following, two different examples are given: the Grand and Urban Challenges sensing architecture and a Start Inhibit system. In the first case, several sensing tasks had to be performed on a prototype vehicle; therefore, a very powerful hardware architecture and a complete version of the GOLD platform were used. Conversely, the Start Inhibit system was conceived to be installed on commercial vehicles; in this case, embedding constraints required to port the system on small architectures.

In addition, also the application of the same technologies in a different field is presented in order to show the versatility of such an approach: a surveillance system for boats speed detection.

Grand & Urban Challenges. The DARPA Grand Challenge was a robotic vehicle competition intended to energize the engineering community to tackle the major issues in autonomous vehicle development. Vehicles had to operate with full autonomy as they maneuvered around obstacles on a 132 mile route in an off-road environment.

Oshkosh Truck Corporation, Rockwell Collins, and the University of Parma partnered together to form Team TerraMax$^{\text{TM}}$in 2005. The developed vehicle (Fig. 2.a) is an MTVR truck equipped with electronic actuators for steering, brake, throttle, and transmission control. Team TerraMax$^{\text{TM}}$is one of the only five participants that completed the course, and the only one that used vision as its primary sensor [3].

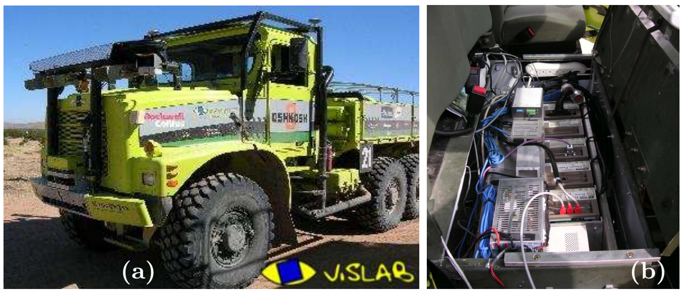

Fig. 2. The TerraMax vehicle (*a*) and its on-board PCs (*b*) fitted under the passengers seat.

VisLab is now developing the artificial vision systems to sense the environment for the 2007 *Urban Challenge*: a new competition in which fully autonomous vehicles have to complete a 60-mile long race in an urban environment in less than 6 hours, obeying to road rules and negotiating traffic.

The vision software is now spread over 4 high performance dual core PCs, used to sense the environment thanks to 11 cameras, and to provide a wide range of information: drivable path, lane position, far and close frontal obstacle, close rear obstacles and overtaking vehicles, and traffic at junctions.

In such complex scenarios, the design efforts are totally focused on reaching the needed levels of accuracy, robustness, speed, and completeness. Issues regarding power consumption, hardware size, etc. are not taken into account. The PCs (fig. 2.b) were chosen among the most powerful ones. Each one is hosting a full-featured copy of GOLD for both acquisition and processing.

Start Inhibit for Heavy Good Vehicles. Many accidents involving heavy goods vehicles are related to the limited field of view of the driver (see fig. 3.a). Although, some of these blind areas can be partly covered by additional mirrors, an electronic system provides a better and safer solution.

VisLab developed a system that monitors the driver's frontal blind area; it warns the driver and prevents the vehicle from taking off when any obstacle or pedestrian is present, namely a Start-Inhibit functionality.

The system has been tested on several prototypes: a Volvo truck, an Iveco truck, and it is also used as close obstacle detector on the TerraMax vehicle. Anyway, for commercial vehicle installation, the use of an embedded architecture is mandatory. Initially, the system has been implemented on a barebone PC (see fig. 3.b) with the output directly sent on the CAN-bus. Thanks to the low computational power needed by this algorithm, it was also successfully ported to a DSP-based architecture. In particular, a Texas Instruments DM642 DSP running at 720 MHz has been chosen: on this platform the system runs up to a 30 Hz rate. This allows the system to be easily integrated on-board real vehicles and become a viable solution for mass-production.

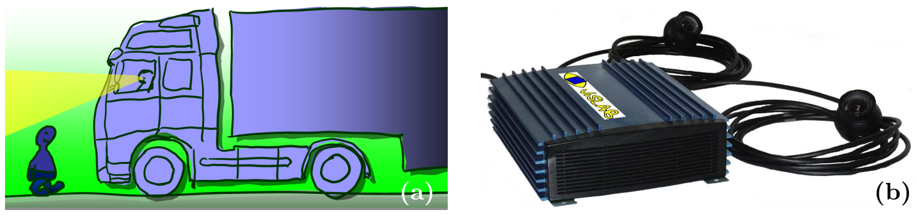

Fig. 3. (*a*) the driver's field of view does not allow to detect pedestrians and (*b*) the barebone PC on which the whole system has been installed and tested.

Boat Speed Monitor. The vision technologies specifically developed for intelligent vehicles can also be successfully exploited in other research fields. As an example, in the following a surveillance system for boats speed estimation is presented. Despite the different issues and scenario involved in developing this system, the usual GOLD platform was used to develop and run this system.

In the city of Venice, a lot of watercrafts pass too fast through water channels, generating waves that erode seacoasts. The preservation of this coast is important to save the unique ecosystem of Venice.

To tackle such a problem, a system that can detect and display the boat speed with the intent to deter excessive speeds has been developed.

A high resolution camera (Tattile TAG 1600) is mounted on the top of a 10 m pole and used to detect boats and evaluate their speed. The camera is connected via Ethernet to an off-the-shelf PC (Pentium 4, 3 GHz, 1 GB RAM) used for the whole processing. The PC is connected by a RS232 link to two large displays that show the speed to the boat's driver.

For this application only a reduced portion of GOLD is needed: only low level vision filters, hardware abstraction layer, and real time management are used. The final commitment is to port the whole system on a smart camera (Tattile TAG PLUS) with an ARM processor and a Linux embedded operating system. Therefore, the GOLD platform will be pruned, keeping its low-level structure and the processing plug-in only.

4 Conclusions

In this paper the most important technologies for the development of on-board vision-based systems for Intelligent Vehicles have been presented, examining both software and hardware issues.

On the hardware side, the different camera technologies and the potential choices for the processing engine have been discussed.

For the software issues, the GOLD platform for the rapid development of vision-based systems has been presented, and its structure and functionalities detailed. GOLD can be used both as a development system and as the engine of the final system, enabling a fast software deployment towards the final system.

Two different case studies were presented: the perception system for the Grand and Urban Challenges and a Start Inhibit system. Moreover, to demonstrate the adaptability of these vision technologies in other situations, also a boats speed monitor system has been presented.

References

1. M. Bertozzi, A. Broggi, G. Conte, and A. Fascioli. The Experience of the ARGO Autonomous Vehicle. In *Procs. SPIE - Enhanced and Synthetic Vision 1998*, volume 3364, pages 218–229, Orlando, FL, Apr. 1998.
2. M. Bertozzi, A. Broggi, and A. Fascioli. VisLab and the Evolution of Vision-Based UGVs. *IEEE Computer*, 39(12):31–38, Dec. 2006. ISSN: 0018-9162.
3. D. Braid, A. Broggi, and G. Schmiedel. The TerraMax Autonomous Vehicle concludes the 2005 DARPA Grand Challenge. In *Procs. IEEE Intelligent Vehicles Symposium 2006*, pages 534–539, Tokyo, Japan, June 2006.