IEEE RTSS 2022 Industry Challenge http://2022.rtss.org/industry-session

Frédéric Boniol
ONERA
Toulouse, France
frederic.boniol@onera.fr

Sibin Mohan
School of Electrical Engineering and Computer Science
Oregon State University
sibin.mohan@oregonstate.edu

I. Background

A. Context

High-performance processors have made considerable progress in recent years. They involve hybrid processing units such as general purpose cores, and specific accelerators (Graphical Processing Units, Neural Processing Units, Deep Learning Accelerators, etc.). This advance now allows for the practical use of AI software in embedded systems to manage complex missions in 'real-time' (such as pedestrian detection for ground vehicles or emergency landing of aerial vehicles). An example of this trend is given by the significant development of embedded machine learning-based functions, such as computer-vision systems, running on GPU architectures, that offer interesting capabilities for autonomous functions such as localizaton, driving, take-off and landing, etc.

The rapid development of this technology offers a great opportunity to make vehicles more reactive to their environment, safer (by avoiding human errors) and less energy-consuming (by optimizing the trajectory of the vehicle), even for safety-critical missions.

However, this trend faces some issues. One of them relates to the computing resources needed to run such complex software functions. Due to space and weight constraints, the number of on-board computing units is limited. They must be shared as much as possible by all the functions running on the platform according to their timing constraints and their safety requirements. The issue is then how to run, in parallel, several high-performance functions on a reduced number (ideally one, or two for redundancy) of CPUs and HW accelerators.

Another question that naturally arises is how to analyze the behavior of those types of systems, and provide a way to prove they meet real-time constraints. For instance, let us imagine an Unmanned Aerial Vehicle (UAV) that has to navigate in an urban environment while avoiding buildings and other vehicles. The response time of the computations required to detect obstacles, plan and adapt the trajectory of the UAV must be compliant with the dynamic behavior of the vehicle and with the frequency of events occurring in the environment.

B. An Illustrative Case-Study

To illustrate these issues, let us consider a vision-based navigation system [1]. This system is representative of embedded systems that can be found in ground robotics and aerial vehicles. The architecture of the system is sketched in Figure 1 and shares common features as architectures promoted in literature [2]. The aim of this system is to compute the position of the vehicle based on visual information. Such a system can act as redundancy localization when classical positioning sensors are no more available.

The system is composed of three software tasks (Visual Odometer, Scene Interpretation and Absolute Localization), a database (GIS) and (if available) positioning sensors (IMA and GPS). Visual information stems from two cameras (denoted as left and right camera) mounted together on a stereoscopic rig. The geographic information system (GIS) is a database that

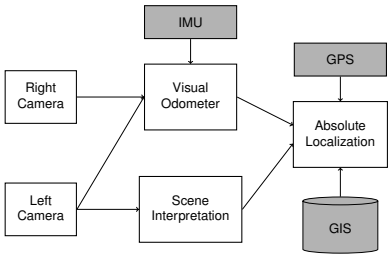

Fig. 1. A stereovision-based system

contains detailed maps including road signs, ground marks and buildings that can be found in the operational area of the vehicle. Example of GIS data used by an UAV operating around an airport is given Figure 2.

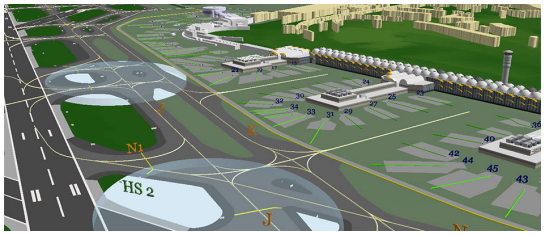

Fig. 2. Map (copyright Performance AirportMaps^{TM})

The Visual Odometer (VO) task uses sequences of stereoscopic images to estimate the trajectory (position and orientation) of the vehicle. The general principle of VO is to locate a camera with respect to a 3D map of the environment [3]. It uses a set of two cameras rigidly assembled on a

baseline in front of the vehicle. Remarkable features (i.e., groups of pixels characterized by a high level of contrast with respect to their neighbors) are tracked in the images from the left camera. Each remarkable feature that can be associated with a 3D position and that can be supposed as a fixed point in the environment is considered as a *landmark*. When the vehicle moves on, the cameras move on in the same way and the landmarks apparently also move in the images. If a sufficient number of landmarks are visible in the image, the pose (position and orientation) of the left camera is computed by comparing the 3D positions of landmarks and their current localization in the image plane. However, it could happen that some of the landmarks leave the camera's field of view, because of the motion of the vehicle. In that case, if the number of landmarks in the current image is no more sufficient to compute the pose with a high level of confidence, VO constructs a map by stereovision using the two cameras: some remarkable features are extracted from the left frame, matched in the right one and associated to a 3D position by triangulation. This computation is time-consuming and required compute-intensive architectures such as GPU. Note that VO is also periodically fused with inertial measurement units (IMU) if available.

The Scene Interpretation (SI) task uses frames from the left camera to build a semantic description of the scene in front of the vehicle. SI is composed of a set of sub-task such as object detection and classification, semantic segmentation, or object tracking that take an image frame and produce a symbolic representation of its visual content, usually associating geometry (the *where* part) and semantics (the *what* part) and often qualified by a score (see Fig. 3 and prior work [4] for more details). The role of SI in the system is twofold: first, it characterizes remarkable objects that appear in the camera's field and that can be found in the GIS database (e.g. a traffic sign and its meaning, a remarkable building, etc.); second, it detects obstacle an provide a category for each of them (e.g. moving/static object, person/car, etc.).

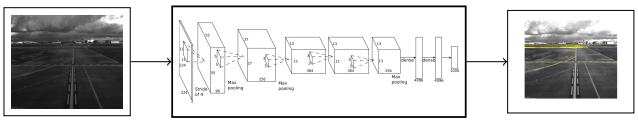

Fig. 3. SI overview (copyright Airbus)

The Absolute Localization (AL) task combines information from all other components (including the GPS source if avalaible) to estimate the position/orientation of the mobile in an absolute world coordinate frame such as WGS 84 (World Geodetic System 1984). The basic operation here is to match objects extracted by SI to landmarks in the GIS so as to change and/or refine the estimated trajectory. Matching landmarks is helped by characteristics provided by SI and by the approximate 3D localization deduced from VO information.

Even when the dynamic of the vehicle is low (e.g. low speed and small steering angle), SI, VO and AL must be done at a high frequency to improve their confidence (typically between 20 and 40Hz). The idea is to compute quite identical situation many times, in order to provide a temporal redundancy.

To acheive this timing requirements, the system must be implemented on a on-board high-performance platform composed of hybrid processors.

C. Background on high-performance architectures

Previously dominated by single-core and then multi-core CPU processors, the world of embedded computing has seen a recent evolution toward hybrid architectures — composed of multi-core CPUs and HW accelerators. Among them, GPUs are becoming a central component for embedding software tasks such as VO and SI. Initially designed for video games and grapical applications, GPUs are increasingly being used for time-consuming embedded systems programmed in CUDA and Open-CL. Successful example of GPUs usage in embedded systems can be found in the automative domain (see for example the Kepler NVIDIA-Telsa architecture[1]) and now in the avionics domain (see the ATTOL experiment by AIRBUS[2]).

Modern hybrid processors typically combine a multi-core CPU with one or two GPUs associated with other accelerators. All these components generally use a DDR memory that is tightly shared between the CPUs, the GPUs and the other accelerators. Some processors also include field-programmable gate arrays (FPGAs) for specific and fast tasks. An example of hybrid processor is presented in Figure 4. This processor is composed of 6 CPU cores, one GPU, two accelerators for deep learning, accelerators for audio and video, etc.

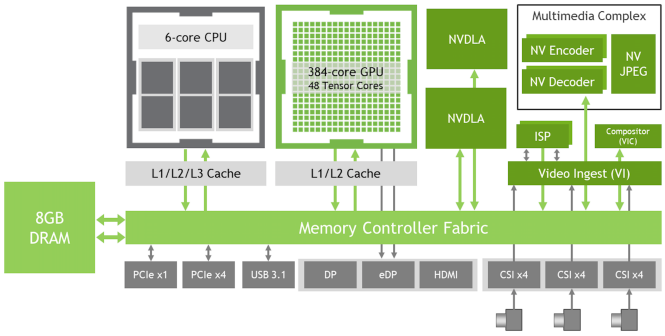

Fig. 4. Example of hybrid processor: Jetson NX Xavier (copyright NVIDIA)

This evolution makes hybrid processors and GPUs the *de facto* choice for many embedded systems, especially those that involve data-/compute-intensive tasks. The Jetson familly from NVIDIA (Figure 4), the iMX8 family from NXP, and the Jacinto family from Texas Instruments are well-known examples of such processors.

From an architectural point of view, GPUs are composed of many streaming multiprocessors, each of them involving

[1]https://en.wikipedia.org/wiki/Kepler_(microarchitecture)
[2]https://www.airbus.com/en/newsroom/press-releases/2020-06-airbus-concludes-attol-with-fully-autonomous-flight-tests

many hardware threads. From a programming point of view, data-/compute-intensive tasks that are to be executed on hybrid processors are composed of (at least) two parts: a *host* part that runs on one (or more) CPU(s) and a second part composed of *kernels* (divided into blocks and threads) launched on the GPUs (or other HW accelerators). Typically, the host part is dedicated to computations that do not require high-performance (for instance the landmark tracking sub-task in VO). It has also to set up the parameters and data for the kernels and plan their schedule. The kernel part is devoted to time-consuming computations such as SI and the 3D-map reconstruction of VO.

Obviously, concurrent tasks that run at the same time on different CPUs and that try to access the same memories and the HW accelerators may directly interfere with each other. They can interfere when using storage resources such as shared caches. But they also can interfere when trying to launch kernels on the same accelerators. On each accelerator, a hardware scheduler locally decides the mapping between the blocks of the kernels and the streaming multiprocessors. However, if there are too many blocks or too many kernels to launch, some of them might be queued until the resources become available. Moreover, the scheduling policy of each accelerator is usually not publicly known.

Consequently, mastering the release date and the response time of safety-critical data-/compute-intensive tasks (such as VO and SI) running on a shared hybrid architecture is a difficult challenge.

II. Challenges and call for contribution

As illustrative by the aforementioned case study, embedding compute-/data-intensive computations in safety-critical systems faces many challenges. The recent trend towards high-performance/heterogenous/GPU-style architectures can be a blessing and a curse, the latter due to the complexity in use and analysis of such systems.

Hence, the RTSS'2022 Industry Challenge invites contributions to tackle some of the important issues, viz.,

1) Due to power consumption and heat dissipation issues, the *number of processing units should be minimized* and their use optimized. To acheive this goal, several questions arise:
   - How to optimize resource allocation in case of software tasks (such as VO) that can use both CPUs and accelerators?
   - How to schedule concurrent tasks, that have to meet real-time constraints, on both, CPUs and HW accelerators?
   - Another important issue also deals with safety: how to detect and recover errors, failures, or timing faults on HW accelerators?
2) A further important concern relates to the certification argumentation: *how to demostrate that the safety-critical tasks meet their timing constraints?*
3) At design time, another challenge arises: hardware architecture dimensioning. The question is: *what kind of public benchmarks and datasets must be developed/used for characterizing and dimensioning hybrid architectures (CPUs and GPUs) for machine learning-based real-time systems?*

The RTSS'2022 Industry Challenge will focus on these issues. We invite colleagues in industry and academia to present their solutions to tackle these issues. The aim of this challenge is to foster better interactions and collaborations between industry and academic communities.

Contributions are welcomed for,

1) algorithms/designs/implementations to optimize resource allocations for (real-time) machine learning frameworks on heterogenous architectures involving GPUs.
2) timing analysis methods for machine learning workloads on such systems
3) novel benchmarks and datasets that can be used by the community in the future to evaluate both of the above.

Contributions should be in the following format:

> *A short paper (maximum **4 pages**, not including bibliography) that details the proposed solution, evaluation, results and a brief demo plan.*

The RTSS '22 Industry Track Program Committee will review these contributions, provide feedback and then select the top ones. Selected papers will be *posted on the RTSS '22 website*. Authors of the selected papers will be *required to attend RTSS '22* (a special session for this track) where they will:

1) present their paper [15 minutes] and
2) present a demo of their solution – either using simulations or on real hardware [15 minutes].

Important Dates

- **Jun. 10, 2022**: Industry Challenge Released to public
- **Sept. 20, 2022**: Submissions due
- **Oct. 25, 2022**: Selection announcement

III. Contacts

Queries regarding the challenge may be addressed to the RTSS '22 Industry Track Challenge Co-Chairs:

- **Frédéric Boniol** [frederic.boniol@onera.fr]
- **Sibin Mohan** [sibin.mohan@oregonstate.edu]

References

[1] F. Boniol, A. CHAN-HON-TONG, A. Eudes, S. Herbin, G. Le Besnerais, C. Pagetti, and M. Sanfourche, "Challenges in the Certification of Computer Vision-Based Systems," *Aerospace Lab*, Sept. 2020.

[2] S. Behere and M. Törngren, "A functional architecture for autonomous driving," in *Proceedings of the First International Workshop on Automotive Software Architecture*, WASA '15, pp. 3–10, 2015.

[3] M. Sanfourche, V. Vittori, and G. Le Besnerais, "evo: A realtime embedded stereo odometry for mav applications," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 2107–2114, IEEE, 2013.

[4] T. Rateke, K. A. Justen, V. F. Chiarella, A. C. Sobieranski, E. Comunello, and A. V. Wangenheim, "Passive vision region-based road detection: A literature review," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, p. 31, 2019.