Industrial Verification Challenge 2015

Rafik HENIA, Laurent RIOUX
Thales Research & Technology
1 Avenue Augustin Fresnel, 91767, Palaiseau Cedex, France
{Rafik.Henia, Laurent.Rioux}@Thalesgroup.com

I. PRESENTATION OF THE VERIFICATION CHALLENGE

The use-case provided by Thales consists of an aerial video system to detect and track moving objects, e.g. vehicles on a roadway. Aerial video tracking systems are mission critical real-time systems since they embed intelligence, surveillance, reconnaissance, tactical and security applications characterized by strict constraints on timing. The main system tasks consist in:

- displaying high quality video images to the user
- following the tracked object even when it is temporarily hidden from view (e.g. the vehicle proceeds in and out of a tree obstructed area) through motion prediction
- detecting patches of the image that may be moving differently from the background by combining image registration and motion prediction

Figure 1: Subsystems of the aerial video tracking system

For simplicity sake, the use-case is limited to two subsystems of the aerial video tracking system, as represented in Figure 1:

1. Video frame processing
2. Tracking and camera control

As suggested by its name, the first subsystem processes the video frames sent by the camera. This includes embedding tracking data into the video, converting the frames to the required format and displaying a high quality video running at 25 frames per second on the monitor. The second subsystem performs motion prediction for the tracked object. Based on this prediction and the aircraft sensors data (position, direction, speed, etc.) it calculates new camera angles and sends instructions to control the camera.

In the following, we propose a timing verification challenge related to each subsystem of the aerial video tracking use-case.

A. Challenge 1: Video Frame Processing

The functional view of the video frame processing subsystem is illustrated in Figure 2. It consists of a sequence of 4 functions processing the video frames from the camera to the display. The Pre-processing function removes reflections from the frames and normalizes the intensity of the individual particles images. The Processing function embeds tracking information into the pre-processed frames and executes zoom in & zoom out instructions. The Filtering function resizes the processed frames and removes the noise. Finally, the D/A converting function converts the frames from digital to analog and sends them to the monitor.

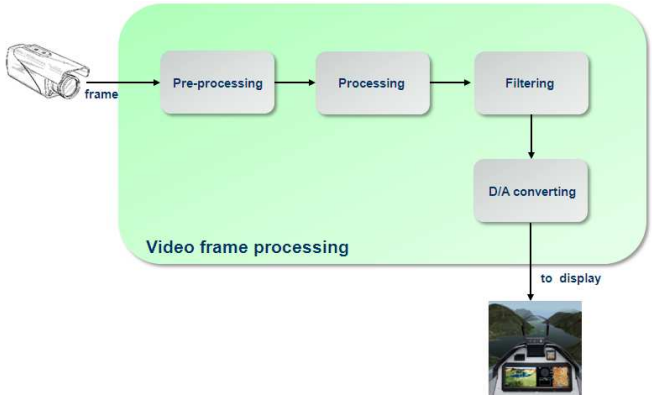

Figure 2: Functional view of the video frame processing subsystem

For simplicity, we assume that each of function is executed by a single task, as illustrated in Figure 3. Each task is assumed to be mapped to a distinct processor. Table 1 shows the execution time for the tasks T_1, T_3 and T_4 and the response time for task T_2.

Task	***Execution time***
T_1	[28ms,28ms]
T_3	[8ms,8ms]
T_4	1ms or 10ms
Task	***Response Time***
T_2	[17ms,19ms]

Table 1: Task execution time

Each frame sent by the camera activates the task T_1. The frames are sent strictly periodically, i.e. the time distance between two consecutive frames sent by the camera is **constant**. The exact value of the period is however unknown since it may slightly vary from camera to camera. However, we know that it ranges between 40ms + 0,01% and 40ms - 0,01%.

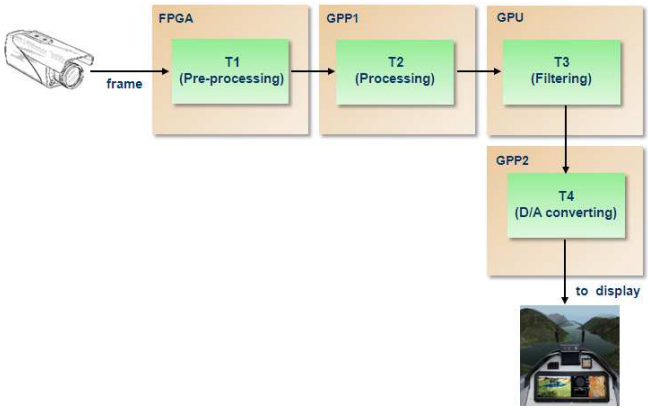

Figure 3: Architectural view of the video frame processing subsystem

After each execution, T_1 sends a frame through its output that activates the task T_2. The response time of T_2 is given in Table 1. A register is used for the communication between T_2 and T_3. At the end of each execution, T_2 sends a frame through its output that overwrites the register content.

When activated, the task T_3 reads the current frame stored in the register. For simplicity, we assume that there are no conflicts between the read and write accesses to the register. The activation of T_3 is strictly periodic, i.e. the period value is **constant**. Due to the clock drift, the exact period value is unknown, however it ranges between 40/3ms + 0,05% and 40/3ms - 0,05%. Note that since the task T_3 is activated more frequently than the task T_2, it will process the same register content more than once.

At the end of each execution, the task T_3 produces a frame. Produced frames originating from the same register content are identical copies and are therefore assigned identical indices. The produced frames are sent to a buffer at the input of T_4. The buffer has a limited size n. For each frame, the following conditions must be met, to be stored in the buffer:

1. The buffer is not full.
2. No other frame having the same index (i.e. identical copy) was already stored in the buffer.

If one of the above mentioned conditions is not met, the frame is discarded. The time required to discard a frame or to store it in the buffer can be ignored.

The task T_4 is activated strictly periodically, i.e. the period value is **constant**. Due to the hardware clock drift, the exact period value is unknown, however it ranges between 40ms + 0,01% and 40ms - 0,01%. Each activation of T_4 leads to one execution. In case the buffer is empty, the execution of T_4 takes 1ms. In case the buffer is not empty, T_4 consumes a single frame from the buffer. In this case, its execution takes 10ms. At the end of its execution, if a frame has been processed, the task T_4 sends a frame to be displayed on the monitor.

The communication between all processors, the access to the register between T_2 and T_3 and the access to the buffer between T_3 and T_4 are considered to be timeless.

In the following, we present the video frame processing timing verification challenge:

1. Compute the maximum latency for a frame sent by the camera and reaching the display, for a buffer size n = 1.
2. Compute the maximum latency for a frame sent by the camera and reaching the display, for a buffer size n = 3.

Due to the different clock drifts and the limited buffer size, all frames with identical indices (i.e. all copies originating from the same register content between T_2 and T_3) may be discarded at the entrance of the buffer. I.e. no copy of the corresponding frame produced by the camera will ever reach the display.

3. Compute the minimum time distance between two frames produced by the camera that will be discarded at the buffer entrance, assuming a buffer size n = 1.
4. Compute the minimum time distance between two frames produced by the camera that will be discarded at the buffer entrance, assuming a buffer size n = 3.

B. Challenge 2: Tracking and Camera Control

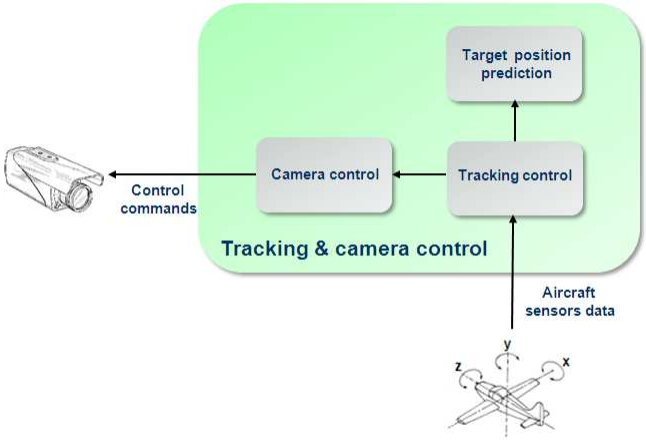

Figure 4: Functional view of the tracking and camera control subsystem

The functional view of the tracking and camera control subsystem is illustrated in Figure 4. It consists of 3 functions. The Tracking control function processes the aircraft sensors data (position, direction, speed, etc.), controls the whole tracking process and generates alerts and various tracking data. The Target position prediction function receives data about the

aircraft speed, position and direction from the Tracking control function and performs motion prediction for the tracked object. The Camera control function receives data about the position of the tracked object from the Tracking control function and calculates a new angle for the camera based on the aircraft position, speed and direction and the tracked object motion prediction.

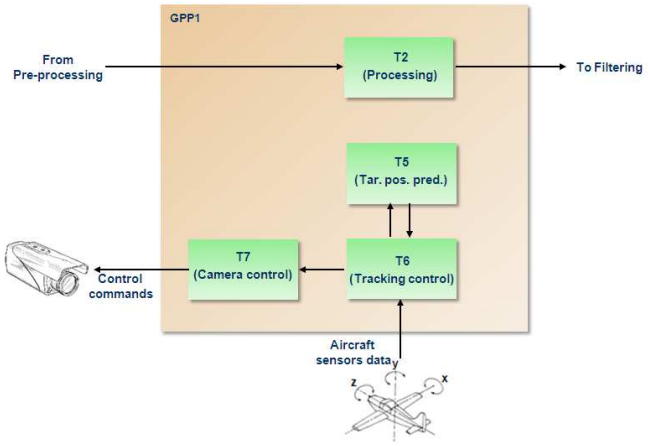

Figure 5: Architectural view of the tracking and camera control subsystem

For simplicity, we assume that each of function is executed by a single task, as illustrated in Figure 5. All tasks are mapped to a same processor GPP_1 to which the task T_2 belonging to the video frame processing subsystem is also mapped. All tasks are triggered by the arrival of data at their inputs. We assume fixed priority preemptive scheduling on the GPP_1 with the following priority order:

$$T_2 > T_6 > T_5 > T_7$$

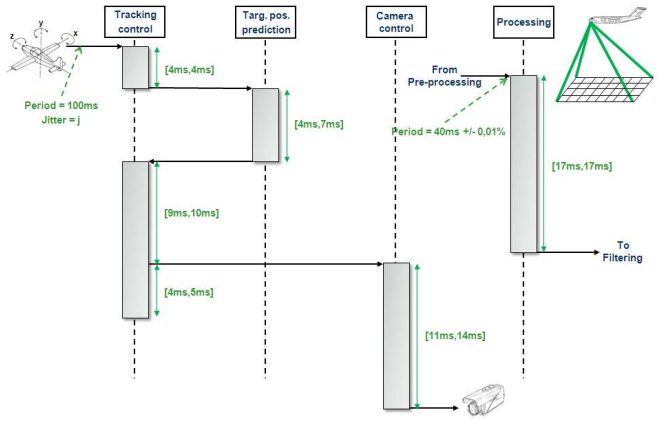

Figure 6: Sequence diagram of the functions on GPP_1

Figure 6 represents the sequence diagram of the functions on GPP_1. The Tracking control function is activated periodically every 100ms. Its periodic activation can however deviate by a jitter value = j. As in Challenge 1: Video Frame Processing, the Process function is activated strictly periodically, i.e. the time distance between two consecutive frames is **constant**. The exact value of the period is however unknown since it may slightly vary from camera to camera. However, we know that it ranges between 40ms + 0,01% and 40ms - 0,01%.

A first segment of the Tracking control function is executed by the task T_6. Then the Tracking control function performs a synchronous call to the Target position prediction function and is suspended waiting for the answer. At the end of the Target position prediction function, the task T_6 resumes executing a second segment of the Tracking control function. An asynchronous call is then performed to the Camera control function executed by T_7 while the last segment of the Tracking control function is executed by the T_6. All execution times by the tasks of the individual functions and function segments are given in Table 2.

Function		***Corresponding Task Execution time***
Tracking control	Segment 1	[4ms,4ms]
	Segment 2	[9ms,10ms]
	Segment 3	[4ms,5ms]
Target position prediction		[4ms,7ms]
Camera control		[11ms,14ms]
Processing		[17ms,17ms]

Table 2: Tasks execution times

In the following, we present the tracking and camera control timing verification challenge:

1. Compute the best-case and worst-case end-to-end latencies from the activation of T_6 to the termination of T_7 for a jitter value j = 0ms.
2. Compute the best-case and worst-case end-to-end latencies from the activation of T_6 to the termination of T_7 for a jitter value j = 20ms.

Let us now assume that T_2 and T5 have access to a shared resource (because the prediction requires information from image). The resource is mutually exclusive and is protected by a priority ceiling protocol. The access to the shared resource takes 2ms for both tasks.

1. Compute the best-case and worst-case end-to-end latencies from the activation of T_6 to the termination of T_7 for a jitter value j = 0ms.
2. Compute the best-case and worst-case end-to-end latencies from the activation of T_6 to the termination of T_7 for a jitter value j = 20ms.
3. Compute the optimum priority assignment minimizing the worst-case latency for a jitter value j = 0ms and j = 20ms.