Multi-Agent System to Design Next Generation of Airborne Platform

Ludovic Grivault, Amal El Fallah-Seghrouchni, Raphaël Girard-Claudon

Laboratoire d’Informatique de Paris 6 - LIP6
&
Thales Airborne Systems - TSA

IDC 2017
12th October 2017
The Studied Airborne Platform

- Remotely Piloted Aircraft System (RPAS)
- Employed during critical military operations
  > Convoy protection, surveillance, rescue
- Used in complex environments
  > Highly dynamic
  > Up to 500 field objects
  > Pop-up & high-velocity objects on the field
- Complex to be controlled and managed through the Mission Manager (MM)
- A large set of instruments (including sensors) is required to achieve missions in real-time

1 Constituting the tactical situation or the “TacSit”
Sensors

- Are complex instruments
- Perceive the platform’s environment
- Can be in conflict (e.g., 2 antennas for 1 frequency)
- Have limited processing capacity
- Provides multiple functions
- Work in different physical domains
  - RF, Optics, IR, Laser, etc.

The embedded sensors
Abstract View of our Framework

The Operator

The Mission Manager (MM)

The Multi-Sensors System (MSS)

- Propose essential functions for operations to the operator through the mission manager
  - Collect field data
  - Communicate
  - Navigate
  - Ensure the platform safety

- Enhance the sensors management
  - Improve the sensors' products
  - Coordinate the mission tasks
  - Balance the loads
Abstract View of our Framework

The Operator

The Mission Manager (MM)

The Multi-Sensors System (MSS)

- Propose essential functions for operations to the operator through the mission manager
  > Collect field data
  > Communicate
  > Navigate
  > Ensure the platform safety
- Enhance the sensors management
  > Improve the sensors’ products
  > Coordinate the mission tasks
  > Balance the loads
MMS in the Next Generation Airborne Platform

Short term changes

- Increasing complexity
- Growth of functions number
- Constraints reinforcement
  - Hardware, threats, prices, size, etc.
- Expectations changes
  - Modularity, autonomy

Today’s MSS limits

- Dedicated communications links
- Lack of sensors’ cooperation and coordination
- Missing an unified sensors’ architecture
Problem Settings

- Build efficient framework for the NGAP
- Overcome MSS limitations & short term evolutions
- Define new architecture with autonomy and modularity
- Enhance cooperation and coordination between sensors

Proposition

Define an agent-based architecture for Multi-Sensors Systems endowed with efficient sensors coordination
Proposition

Agent-Based Architecture for MSS

In the proposed architecture:

Agent $\rightarrow$ Field Object’s Digital Twin

Each agent symbolizes one field object: “Tactical Agent”.

In the proposed architecture
Agents: low-level & high-level roles

( Mission Manager $\leftrightarrow$ Hardware Sensors)

Sensors Coordination

Fast & efficient scheduling algorithm

Working under high constraints & complex precedence scheme
Tactical Agent

Main Goal
- Collect data of the embodied field object it represents

Three Roles
- Provide data to the MM
- Comply to the MM’s orders
- Generate sensors plans

Required data by agents
- Platform’s data
- Orders & Policies (from MM)
- Sensors and Field Objects Data
- Operational and Sensors knowledge

Tactical Agents are generic when created
They are instantiated according to new sensors data
From Agents to Sensors

The MSS agent-based architecture
Agent-Based Architecture For MSS

The MSS agent-based architecture
Agent-Based Architecture For MSS

MAS bootstrap
Creation of the RPAS Agent

Tactical Agents (TA)
Interface Mission Manager ↔ Sensors

Track Merger Artifact
- Merges the sensors data
- Creates TA w.r.t. the observed objects in the theater
- Forward new data to the TA

Agents-Scheduler-Sensors-Merger
- 4-Step sensors’ control loop

A simpler view of the architecture
Coordination Based on Scheduling

$K$ Plans of $N$ tasks on $R$ indivisible resources

- Complex General Precedences Relations
  > Start-Start, End-Start, Min/Max Start-Start Delay & Start-Start + exact $\Delta$

- Plans precedences (e.g. $P_i$ before $P_j$)

- Priority: $\Pi(\text{Agent}) = \Pi(\text{Plans}) = \Pi(\text{Tasks})$

- Strict plans release & due dates, no delay allowed

Scheduling Criteria

- Minimize resources’ free time

- Maximize scheduled plans priority

A strongly constrained plan of tasks
Scheduling Algorithm

Determines for each task its starting time \( (t_s) \)
Greedy fast algorithm, sweeps tasks and plans to find matching \( t_s \)

Sort the \( K \) plans by priority

\[ \text{For each plan } j \]

\[ \text{For each task } t \]

Search compatible task’s \( t_s \)

If found, go to next task

Else, shift \( t_s \)

Else, shift the plan \( t_r \)

If one task not scheduled

Abandon \( j \)

Input: \( P \) the ordered set of the whole plans of tasks
Output: \( t_s \): the start time of each task \( T_n \) for each plan \( P_k \)
Data: \( T_H \): Temporal horizon

\[ P \leftarrow \text{sortPlansByPriority}(P); \]

\[ \text{foreach Plan } P_k \text{ of } P \text{ do} \]

\[ \text{while } \exists T_n \text{unscheduled } \in T_k \land P_k \cdot t_r < P_k \cdot t_d \text{ do} \]

\[ \text{foreach Task } T_n \text{ of } T_k \text{ do} \]

\[ T_n \cdot t_s \leftarrow \text{precedence}(C_n); \]

\[ \text{while } \neg \text{isCompatible}(T_n) \land T_n \cdot t_s < T_H \text{ do} \]

\[ \text{if matchingConstraints}(T_n) \text{ then} \]

\[ \text{nextTaskOfThePlan}(P_k); \]

\[ \text{else} \]

\[ \text{shift}(T_n \cdot t_s); \quad /* \text{increment } t_s */ \]

\[ \text{if } \neg \text{isCompatible}(T_n) \text{ then} \]

\[ \text{shift}(P_k \cdot t_r); \quad /* \text{increment } t_r */ \]

\[ \text{if } \exists T_n \text{unscheduled } \in T_k \text{ then} \]

\[ \text{abandon}(P_k); \]

\[ \text{adviseAgent(“unplanned”,ownersOf(P_k))}; \]

\[ \text{else} \]

\[ \text{addToGlobalSchedule}(P_k); \]
Simulation

- Special Test Scenario
  - Operational Scenario
  - 10 usual tasks (e.g. localization, image capture by camera or radar, etc.)
- Developed in Java 1.7
- Simulation revealed
  - Improved MSS autonomy
    (orders are launched autonomously by agents)
  - Strong Modularity
  - Efficient Resources sharing
  - High field data granularity

Simulation's Main Frame → Tactical situation, agents, knowledge base
### Scheduling Simulation

- Fast scheduling: 88ms for 100 plans on a i7 processor
- Highest priority → first scheduled
- Satisfying integration in the MSS architecture

Example of output schedule
Conclusion

MAS for MSS
First breakthrough

First simulations
Requirements satisfaction

Upcoming researches
Multi-platform applications
Track merging solutions
Thank you for your attention