Situation recognition as a step to an intelligent situation-aware crew assistant system

Quint Mouthaan Patrick Ehlert Leon Rothkrantz

Delft University of Technology, Mekelweg 4, 2628 CD, Delft

Abstract

In this paper we present a system that can recognize situations during a flight in real-time based on data from simulated aircraft systems. The system uses Bayesian belief networks to calculate the probabilities of both the start and end of all possible situations and from this distillates the most probable situation. The situation recognizer system is part of our test environment to create better human-machine interfaces in the cockpit.

1. Introduction

Anyone who has seen the cockpit of an F-16 aircraft knows that it is stuffed with control buttons, meters, and displays, providing the pilot with a wealth of information. Since the F-16 is capable of speeds of over 2000 Km/h, pilots have very little time to process the large amount of available information and make decisions. To help a pilot deal with information processing and decision-making and avoid information overload, an intelligent pilot-vehicle interface or Crew Assistant System (CAS) or has been proposed [1,2]. A typical CAS is shown in Figure 1. The idea is that the system presents relevant information to the pilot at the right moment and in the appropriate format, depending on the situation, the status of the aircraft, and the workload of the pilot. It is even possible that the CAS takes over (simple) tasks. Not only military pilots can benefit from such a system, it is useful for commercial pilots as well.

The Intelligent Cockpit Environment (ICE) project is a project of the Knowledge Based Systems group of Delft University of Technology. The goal of the ICE project is to design, test, and evaluate computational techniques that can be used in the development of intelligent situation-aware CASs. Using methods from artificial intelligence, ICE focuses primarily on the data fusion and reasoning part of these systems. Special issues addressed in the ICE project are situation recognition, mission or flight plan monitoring, pilot workload monitoring, and attack management [3,4].

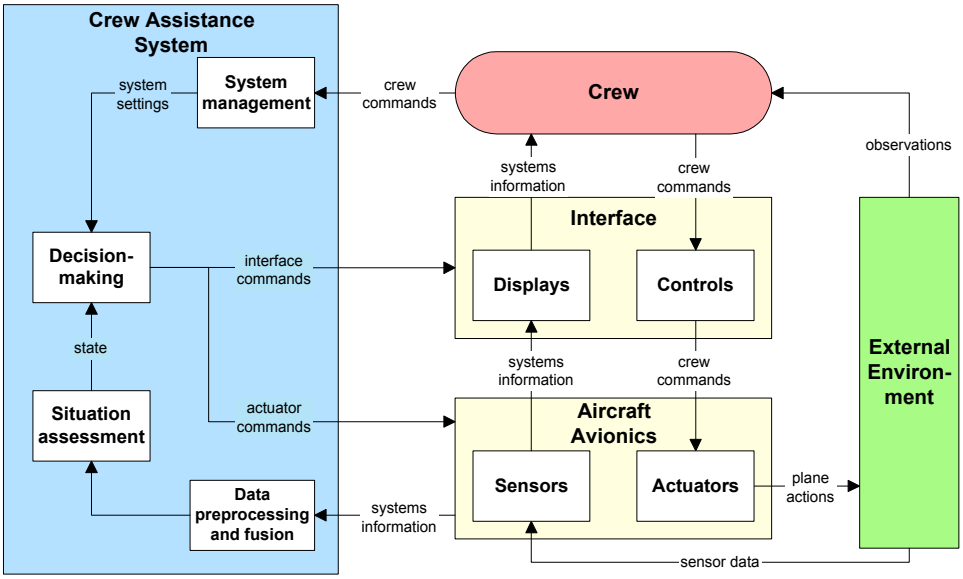

Figure 1: a generic crew assistance system architecture

In this paper we present a system for real-time situation recognition, which is an important subsystem for situation assessment in any CAS. First, we will describe the design of the recognition system. Then we will discuss an example scenario that was used to test our system. Finally, we will draw some conclusions about the performance of the system.

2. The design

The goal of our system is to derive the current situation in real-time from available aircraft data. The system receives information from a simulator about the state of the airplane (e.g. airspeed, altitude, pitch), the actions of the pilot (e.g. lowering the landing gear, changing display settings), and the environment (e.g. other planes, or a missile that has been launched). Our system will use all this information to determine which situation is occurring. For every situation, the actions the pilot is expected to perform and typical situation-related events are defined. These events can either be changes in the state of the airplane or changes in the environment. During a flight the system will compare the received information with the stored situations data and it will try to determine which situation is occurring.

2.1. System architecture

The architecture of the situation recognition system is shown in Figure 2.

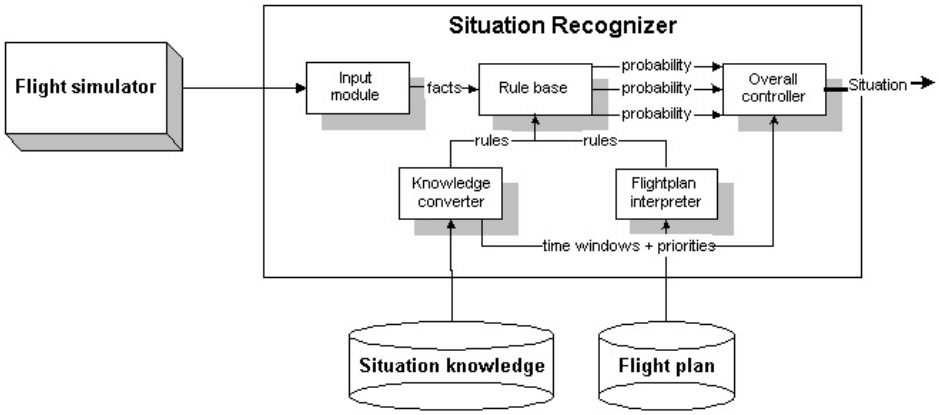

Figure 2: the architecture of the situation recognition system

The input module receives aircraft data from the flight simulator and converts this data to facts that are forwarded to the rule base.

The knowledge converter converts all the situations knowledge that is stored in an XML file to IF-THEN rules and puts them in the rule base.

The flight plan interpreter converts the information in the flight plan to a number of rules that are put in the rule base. These are rules that predict which situations will occur in the near future.

The rule base contains all the rules that have been generated by the earlier described modules. When data from the flight simulator is added to the rule base, some of the rules will fire and generate probabilities concerning the start or end of a situation that are passed to the overall controller.

The overall controller receives situation probabilities from the rule base as well as some extra information about the situations. The overall controller combines the probabilities and calculates for every situation the probability that it has started and the probability that it has ended. It then draws a conclusion about the situation that is most likely to be the current one. Calculating probabilities is done using Bayesian Belief Networks (BBNs), which will be discussed in the next section.

2.2. Probability inference using Bayesian belief networks

We want to calculate two probabilities for every situation: the probability that the situation has started and that it has ended. As a first order approach we have created two BBNs.

2.2.1. The start probability calculator

Figure 3 shows the BBN that is used to calculate the probability that a situation has started.

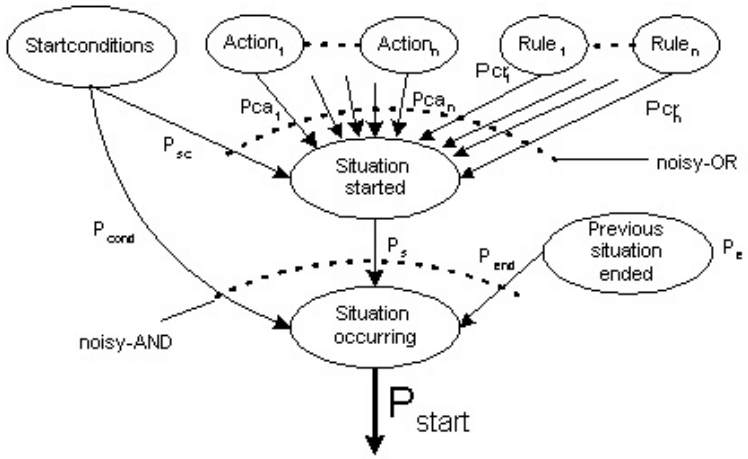

Figure 3: BBN that calculates the probability that a situation has started

The start conditions for a situation are conditions that must be satisfied before a situation can possibly have started. When the start conditions are satisfied, the probability of the start constraints that is specified in the situations knowledge will be the output of this node.

The action probabilities are passed to the BBN by the rules in the rule base when the pilot performs a particular situation-related action. These probabilities all contribute to the probability that the situation is occurring (has been started).

The additional rules are rules that fire when the state of the aircraft changes or when a specific event happens. When they fire they can generate a probability that a situation has started or ended.

The probability calculator (situation started) combines the probabilities of the nodes that have been described above using the noisy-OR model.

The previous situation influences the start probability of a situation because the probability that a situation is occurring should rise when the probability increases that one of the previous situations that can lead to this situation has ended.

Based on this BBN the probability that a situation has started and is occurring can be calculated with the following formula:

$$P_{start} = P_{cond} * P_{end} * P_s$$

$$P_{cond} = \begin{cases} 0 & if\ P_{sc} = 0 \\ 1 & if\ P_{sc} > 0 \end{cases}$$

$$P_s = 1 - ((1 - P_{sc}) * \prod_{i=1}^{n} (1 - Pca_i) * \prod_{j=1}^{n} (1 - Pcr_j))$$

In this formula, P_{sc} is the probability of the start conditions, P_{end} is the probability that one of the previous situations has ended, Pca_i is the probability of the i-th action that should be performed during the situation and Pcr_j is the probability of the j-th event or change in state that can occur during the situation.

2.2.2. The end probability calculator

In Figure 4 the BBN is shown that calculates the probability that the situation has ended. In this BBN we see a lot of the same nodes as in the belief network for the start of the situation. The nodes that are different are discussed below.

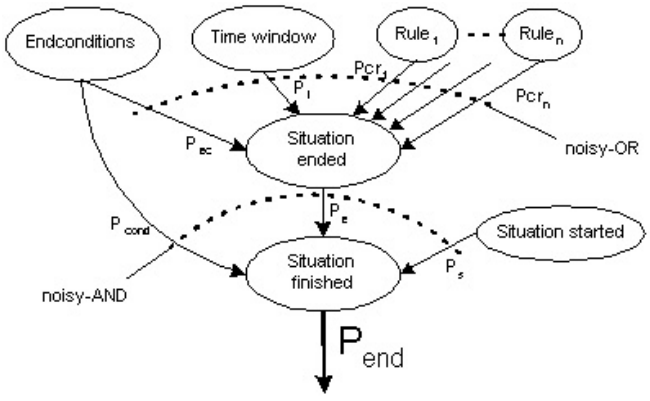

Figure 4: BBN that calculates the probability that a situation has ended

The time window for a situation is the maximum duration of that situation. If the start of a situation has been detected the probability that it has ended should grow after a certain time.

The situation started node produces a 1 if the situation has started and a 0 if the situation has not yet started. This node is necessary because we only want to calculate the probability that the situation has ended, after (a probable) start of that situation.

The probability that the situation is ended can be calculated with the following formula:

$$P_{end} = 1 - ((1 - P_{sc}) * (1 - P_t) * \prod_{i=1}^{n} (1 - Pcr_i))$$

More information about the design of our system can be found in [5].

3. Experiments and results

The system was tested using Microsoft's Flight Simulator 2002. Experiments were performed with an F-16 and with a Cessna airplane. The results of one of our experiments are presented in Table 1. The first column contains the names of the situations (19 different situation were defined) that occurred and/or were detected. The second column contains the times at which we considered the situations to be started. The third column contains the times at which the situations were detected by the system. The times are given in seconds from the moment the program was started. The particular mission of Table 1 consisted of an attack on a ground target in an F-16. During the flight to the target the pilot had to check his course twice (navigating). After the attack had been performed the pilot returned to the airbase and landed the airplane.

Table 1: results of an experiment flight

Situation	Time started (s)	Time detected (s)
Startup	0	0
Taxiing to runway	10	12
Taking off	14	15
Normal flight	43	34
Navigating	83	83
Normal flight	91	91
Navigating	123	123
Normal flight	128	128
Visual attack	186	193
Normal flight	221	221

Landing	361	366
Aborting a landing	-	410
Taxiing from runway	409	410
Shutdown	427	427
Error rate = 0.06 (26 seconds)		

From the table it is clear that the program is able to recognize most situations in a matter of seconds. The error rate was calculated by calculating the amount of time that the recogniser was incorrect. The program has some difficulty in detecting the situation "normal flight" after "taking off" (this problem occurred in other experiments as well and has to be looked into). The landing was a normal landing, but as is shown in the table the program thought for a moment that the landing was being aborted. This happened when the program knew that the landing had been finished and looked for the situation with the highest start probability. This turned out to be the situation *Aborting landing*. This is because at some point the pilot had moved the throttle to the maximum. This action was still in the memory of the program when the landing ended. The fact that the landing gear was raised at the start of the landing was also still in the memory of the program. Because of this the probability that the landing was being aborted was high and the situation *Aborting landing* became the current one once the landing had finished. However as soon as that happened the program realized that the airplane was actually taxiing and it corrected the mistake immediately.

Our other experiments showed similar results. On average the error rate over the performed experiments (4 flights) was 0.08, with 0.05 being the lowest recorded error rate and 0.11 being the highest.

4. Conclusions and future work

We have created a system that can recognize the current situation during a flight with an F16 and with a Cessna. The system is based on a probabilistic model. A rule base was created that compares data from a flight simulator with the situations knowledge defined in an XML file. The rules in the rule base generate a number of probabilities that are combined using BBNs to calculate the probabilities that the situations are occurring. Based on these probabilities a conclusion is drawn about the situation that is most likely to be occurring.

We did not prove the correctness of the system, but investigating a number of test scenarios, the system seems to works fairly well. It makes

few mistakes and is able to correct them. Furthermore it is able to come to a conclusion about the current situation in real-time.

Future work will consist of solving the "Taking-off/Normal flight"-transition problem, adjusting the timeframe a particular action or event is stored, and adding causality to make the system more reliable. In addition we would like to improve the system to include more and synchronous situations and compare its results to other approaches such as Dynamic network models or production systems. We also plan to use the system in conjunction with a workload assessment module that is under development to construct a more complete situation awareness module. This situation awareness module can be used in an intelligent cockpit system that monitors and supports the pilot during a flight.

References

[1] Banks, S.B and Lizza, C.S. (1991) "Pilot's Associate: a cooperative, knowledge-based system application", in *IEEE Intelligent Systems, Vol. 6*, No. 3, pp. 18-29, June 1991.

[2] Onken, R. (1997) "*The cockpit assistant system CASSY as an on-board player in the ATM environment*", paper presented at 1st USA/Europe Air Traffic Management R&D Seminar, Saclay, France, June 1997.

[3] Ehlert, P.A.M. and Rothkrantz, L.J.M. "*The Intelligent Cockpit Environment Project*", Research Report DKS03-04/ICE 04, Knowledge Based Systems, Delft University of Technology, The Netherlands.

[4] http://www.kbs.twi.tudelft.nl/Research/Projects/ICE/

[5] Mouthaan, Q.M. (2003) "*Towards an intelligent cockpit environment: a probabilistic approach to situation recognition in an F-16*", MSc. thesis, Knowledge Based Systems, Delft University of Technology, The Netherlands.