Design and Evaluation of a UAS combining Cognitive Automation and Optimal Control

Sebastian Clauß* , Stefan Brüggenwirth† , Pierre Aurich‡ and Axel Schulte§
Institute of Flight Systems, University of the Bundeswehr Munich, GERMANY

Vladimir Dobrokhodov¶ and Isaac Kaminer‖
Naval Postgraduate School, Monterey CA

The University of the Bundeswehr Munich (UBM) and the Naval Postgraduate School (NPS) conduct cooperative research in the field of single operator UAV guidance and ISR payload control. For the first time cognitive automation and Optimal Control elements are combined to join high-level mission guidance with path planning and real-time path following. In this scope, the CoCAMPUS (Cooperative Cognitive Automation through Mathematically optimized Path-Following of UAVs) project is conducted to explore how to properly support a single operator in mission-guidance and flight-control to enhance the overall system performance, based on a simplified Air-Attack mission including static threats. The Cognitive System Architecture COSA[2] is used to implement cognitive automation behavior on the basis of explicit knowledge models, dynamically deriving implicit cost-functions, serving the purpose of path optimization.

I. Introduction

The growth of UAV applications in nowadays military scenarios leads to an increasing system complexity that has to be dealt with for successful mission execution. Most current UAVs therefore already feature flight control systems that relieve the operator from high-frequent tasks and allow him to command the system on a more abstract layer such as waypoint navigation.[1] Especially unforseen events, which cannot be considered during mission planning or the system design phase, may lead to situations that so-called conventional automation lacks the capabilities to properly handle. Here, on-site human knowledge-based performance is required to manage and preprocess given problems. To allow a UAV system with a single operator to manage fast-paced inner-mission planning and decision making as well as unforseen difficulties it is imperative that the human is properly supported in both mission-guidance and flight-control.

I.A. Related Work and Contribution

Research conducted by[234] show that the consideration of human factors in system design plays an important role when dealing with complex, dynamic scenarios aiming at maximizing the overall system performance. Efforts for the implementation of symbolic cognitive frameworks supporting the operator in mission fulfillment include TacAirSoar[5] which was demonstrated as major part in the Synthetic Theater of War 1997, as well as the UK's surragate UAV programme[6].[7]

The Naval postgraduate School (NPS) has conducted research in the field of trajectory generation and real-time path-following of UAVs based on Optimal Control theory. Single or multiple UAVs can be commanded by formulating a set of goal states and constraints, relieving the operator of the demanding task

*Research scientist, Dept. of Aerospace Eng., 1st Lt (GAF); sebastian.clauss@unibw.de
†Research scientist, Dept. of Aerospace Eng.; stefan.brueggenwirth@unibw.de
‡Aerospace Eng., 1st Lt (GAF); pierre.aurich@online.de
§Professor, Flight Dynamics and Flight Guidance, Dept. of Aerospace Eng.; axel.schulte@unibw.de
¶Research Associate Professor, Dept. of Mech. and Aerospace Eng., AIAA Member; vldobr@nps.edu
‖Professor, Dept. of Mech. and Aerospace Eng., AIAA Member; kaminer@nps.edu

to formulate continous step by step instructions.[8] A 3D-trajectory is generated and optimized according to a given set of desired optimization criteria, such as time, fuel consumtion etc. Results show that spatial and temporal requirements can both be met by the three-dimensional path-following approach, allowing time-critical maneuvers more dynamically than discrete waypoint following.[9,10]

Conventional automation and Optimal Control elements share the flaw that the formulation of constraints and parameters is solely left to the human operator, which leads to high workload resulting in a raised error probability especially in case of unpredicted events.[11] To approach this issue, the Institute of Flight Systems proposes UAV-guidance on a task-based level,[12] where the operator issues tasks on an abstraction level comparable to inter-human interaction. Artificial Cognitive Units (ACUs) aboard the UAV are used to implement task-based guidance and mimic human rationality to some extent, forming so-called cognitive automation, processing information mostly on a symbolic level.[2] The ACU's behavior is completely defined by its goal, which is explicitely represented as knowledge in its implementation.[13] It allows knowledge-based performance such as decision making and planning regarding its own understanding of the current situation. It also offers semi-autonomous capabilities, in a sense that the ACU pursues a given task autonomously using the automation present inside the UAV's system boundaries.[14]

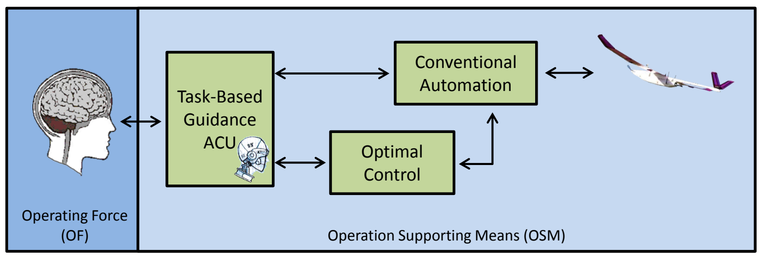

Figure 1: UAV worksystem, showing its Operating Force (OF) and its Operationg Supporting Means (OSM).

The desire to combine the strengths of both the Optimal Control trajectory generation and path-following as well as the cognitive automation given by a task-based guidance ACU as schematically shown in Figure 1, led to the initiation of the CoCAMPUS (Cooperative Cognitive Automation through Mathematically optimized Path-Following of UAVs) project. Its aim is to design, implement and evaluate a UAV system, which enables a single human operator to perform a complex Air-Attack mission with a single UAV, relieving him of high-frequent control tasks and additional control of sensor and system automation elements, by introducing a proper task-based guidance architecture.

The main contribution of this article consists of a description of the design of a UAV system in which a symbolic mission management unit with explicit, static knowledge is used to generate implicit goals and constraints for Optimal Control trajectory generation to fulfill given tasks in a task-based guidance paradigm. The paper describes the first UAV application of an ACU based on our new COSA^2 architecture for cognitive agents and presents a novel knowledge modeling methodology to generate its static knowledge according to mission and vehicle requirements. A systems engineering approach is chosen to implement the UAV system, which is tested during a first flight test evaluation.

This paper is organized as follows: Section II describes the concept of cognitive automation and its implementation with COSA^2. Section III shows the concept of Optimal Control trajectory generation for path following augmenting a conventional flight-control setup. Further, section IV depicts the UAV system architecture based on the CoCAMPUS mission, the systems engineering approach used to incorporate human factors into system design as well as our novel knowledge modeling methodology for COSA^2, while we present flight test results in section V.

II. Cognitive Automation

A worksystem as a general ergonomics concept is defined by its work objective, e.g. a mission order, as its main input to the process of work, which is influenced by environmental conditions, information and supply such as weather, threats and obstacles. At all times, the current work state (e.g. mission progress) can be observed through the work result as far as information is accessible.[2] The worksystem itself consists of two major elements, the operating force (OF) and the operation supporting means (OSM) (Cf. Figure 1). The OF contains at least one human operator, i.e. the UAV operator in a ground control station, and is in charge of achieving the work objective. The OF can select from and command the different OSMs available, according to a comprehensive understanding of the work objective and the current situation. Thus it's the high-end decision-making component of the worksystem, representing its highest authority. The OSM can perform certain, well-defined sub-tasks and incorporate the aircraft as well as all automation available onboard.

Depending upon their role in the work system, ACUs may support or even replace a human operator with software concepts and algorithms used within the artificial intelligence and rational agents community. For this purpose, the Institute of Flight Systems has developed a dedicated software framework called Cognitive System Architecture with Centralized Ontology and Specific Algorithms (COSA^2) that enables ACUs to exhibit knowledge based behavior.[15] Applications developed for COSA^2 follow a conceptual processing scheme based on the Rasmussen scheme[16] of human cognitive performance. By using this analogy we believe that (1) the modeling and ontology creation by human domain experts as well as the interface to a human operator may benefit from the semantics used in the cognitive process and (2) intelligent agents using the COSA^2 framework have the ability to act in a flexible and comprehensible way.

II.A. COSA^2

The modified Rasmussen scheme[2] shown in Figure 2, laid the intellectual foundation in the attempt to establish an analogous computation paradigm for intelligent software agents. It breaks down human cognitive performance into simplified cognitive subfunctions, shown as gray boxes in the figure. Cognitive subfunctions are arranged along three layers of increasing abstraction. Skill based behaviour describes subconscious, basic action skills. Procedure based behaviour occurs consciously as a reaction in well known situations. The highest level of flexibility is achieved by subfunctions of concept based behaviour, which allow the human to derive novel solutions in unknown situations on the basis of relevant background knowledge.

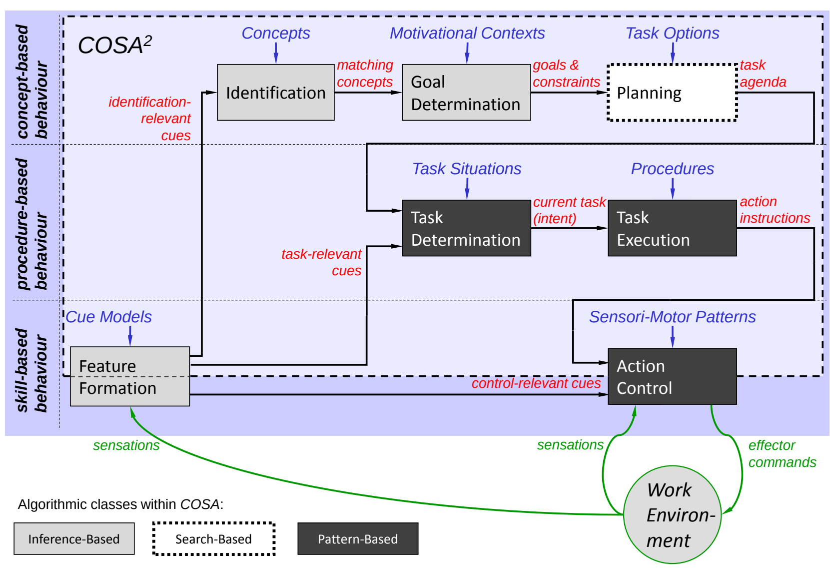

Figure 2: Modified Rasmussen Scheme (cf.[2]).

The modified Rasmussen scheme distinguishes static, a-priori knowledge specified during design time (shown in blue in Figure 2, e.g. CONCEPTS or TASK OPTIONS) and situational knowledge, generated during runtime and stored in the agents working memory (shown in red in Figure 2, e.g. identification relevant cues or matching concepts). A unique feature of COSA^2 is the claim for a central knowledge representation. This means that both situational and a-priori knowledge are specified exactly once and are accessible from all the different subfunctions. As a consequence, each cognitive subfunction may generally read all situational knowledge elements available in working memory. However, it writes specifically only into the situational knowledge indicated by the outgoing arrow. For example, the cognitive subfunction Identification is implemented as an inference algorithm which uses CONCEPTS as a-priori knowledge. It may read from anywhere in working memory (with its primary source being identification relevant cues) and exclusively modify the situational knowledge on matching concepts.

During runtime, COSA[2] periodically steps through the different cognitive subfunctions, which are implemented by three different classes of algorithms as shown in Figure 2. The first three subfunctions, Feature Formation, Identification and Goal Determination use an inference mechanism to update and interpret the new set of input data. Feature Formation is responsible for associating a symbolic representation from raw input data, which is then interrelated to derive an abstract, comprehensive description of the situation by the Identification subfunction. On the basis of these matched concepts, Goal Determination decides whether the currently perceived situation violates a set of modeled goals and constraints. Should a goal be violated, the Planning subfunction, which is implemented as a search algorithm, decides upon Task options to be executed by the ACU, in order to transition into a new environment state, in which no more goals are violated. These tasks are stored as task agenda that is then executed by the remaining subfunctions. Task Determination detects patterns (cues) that trigger the activation of the next tasks or monitor for execution failures. Task Execution then associates with each high level task a set of low-level action scripts which result in output commands sent by Action Control.

III. Optimal Control

This section introduces an approach that facilitates onboard trajectory planning and following for the task execution by a semi-autonomous UAV. The ultimate goal of this design is to enable near real-time planning or rapid modification should the operational conditions change and need arise. As soon as the mission objectives are defined for the high-level cognitive component, the set of specific tasks targeting an application of the UAV in an a priori given operational environment, needs to be designed. The task should specify the trajectory (both the path and the velocity profile) such that the mission objectives are achieved, the UAV and its payload do not exceed the flight dynamics constraints, and that the operational constraints such as radar detection, airspace deconfliction and collision avoidance conditions are met. Thus, this section outlines a path generation approach that is suitable for near real-time computation of feasible trajectories for a single UAV that accounts for the flight dynamics and operational constraints, and can be followed by resorting to the path following algorithm.[17]

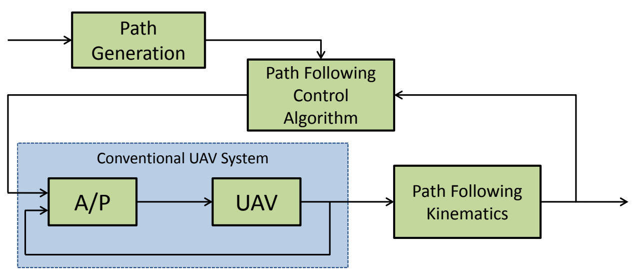

Figure 3: Optimal Control setup augmenting a conventional UAV System (cf.[9]).

Figure 3 shows the schematic setup of the Optimal Control approach. The conventional UAV system, consisting of the UAV flight dynamics and a commercial autopilot, is augmented by a Path Following control loop. The setup tracks generated trajectories, given by the Path Generation module upon operator command.

III.A. Feasible Path Generation

Consider a single UAV that is tasked to fly a typical mission while avoiding detection by an a priori known set of radars; the UAV performs its mission starting from its current state (initial conditions) and arriving to the final conditions specified by the mission planner. While the exact duration of the mission is not known a priori, it is quite typical that the duration needs to be minimized; this can be justified by the nature of the mission itself and/or the restrictions on energy expenditures imposed by the UAV platform. Furthermore, suppose the objective of the UAV task is not only to minimize the mission duration or the vehicle energy expenditures while meeting dynamical constraints (e.g. bounds on maximum accelerations), but also to minimize the probability of being detected by a number of radars.

The described problem belongs to the class of optimal control problems and it would be desirable to have an algorithm capable of solving it in near real-time. Unfortunately, classical indirect approaches of calculus of variations (Bellman's dynamic programming, Pontrjagin's maximum principle) can handle only very simple problems like a double integrator where even an analytical solution is possible. However, obtaining an optimal solution off-line for more realistic system is still quite difficult. That is why various simplifying approaches have been developed to provide a near-optimal solution in close to real-time rather than an optimal solution (collocation method, method of differential inclusions, etc.) off-line.

This work adopts the key idea of the approach proposed in[18] . The method is based on explicit separation of the path and the velocity associated with it. Both the path and the velocity profile are represented by using some reference functions expressed via a limited number of variable parameters such as, for example, length of the path. The remaining states and controls can be determined using inverse dynamics of original non-linear equations driving the system. As soon as the values of variable parameters are fixed, the path and the associated velocity profile are given, and the required controls can be quickly evaluated using the inverse dynamics. When done, a cost function representing the objective of the optimization task can be calculated. Implementing this procedure iteratively enables calculating the best solution (path and velocity profile) by varying a limited number of variable parameters. A brief explanation outlines the key elements of this approach related to the tactical components of the cognitive mission planner in the Appendix. For detailed presentation of the approach and its historical evolution, an interested reader is referred to the publication[18]and the references herein.

A particular example of tactical UAV mission solved by utilizing the path generation algorithm will be provide later in section V.

IV. System Implementation

The UAV system architecture incorporating both elements of cognitive automation and Optimal Control, orginates from requirements derived from the CoCAMPUS mission. While designs of systems using conventional flight-guidance automation have been variously documented, there is no standardized systems engineering approach regarding the integration of symbolic, high-level mission management and subsymbolic automation components.

For the design and development of our UAV system, we adapted the Harmony system- and software development methodology by Telelogic,[19] which is a model based systems engineering methodology using a request-service driven approach. We supplemented its "Design Synthesis" step with a worksystem analysis (Cf. Figure 4) . The worksystem analysis uses the worksystem concept to identify the different OF and OSM elements necessary to accomplish a specified work objective. It describes the interaction between the different elements according to specific environmental and system information and shows what work result to expect with a certain setup. Most importantly the worksystem analysis allows the integration of a human operator into the design process of the UAV system and thus incoporates aspects of human-machine interaction allowing to identify an operator overload as a result of a certain work objective in specific situations. The introduction of an ACU as an element of cognitive automation, used to relieve the operator of certain tasks, can thus be designed and analysed according to given mission requirements and operator needs. The worksystem analysis allows for isolating such tasks which require a knowledge-based decision-making and planning capability during mission execution and are conventionally conducted by humans. Additionally, it helps to identify such tasks in which an ACU can support the human operator and situations in which it's imperative for an ACU to independently take over control to satisfy safety measures and/or mission requirements.[20] This adaptation of the SE-methodology gives an adequate derivation of workshare between the operator and an ACU as the decision-making engine of the semi-autonomous UAV system, guided on a task-based level. It also allows for an identification of the full scope of interactions between the ACU and the human operator as well as the other onboard automation elements, leading to respective interface definitions.

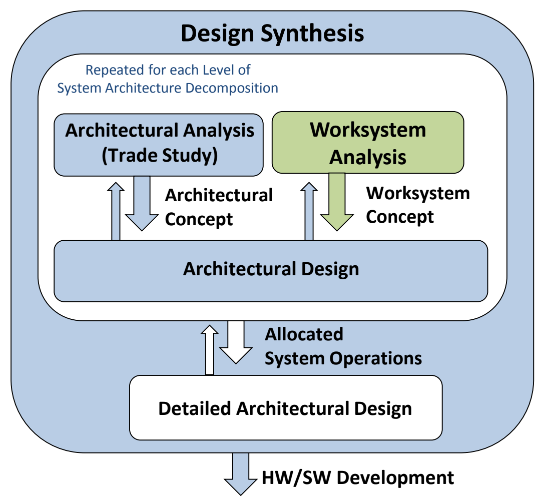

Figure 4: Adapted Harmony - Systems Engineering methodology (cf.[21]).

IV.A. The CoCAMPUS Mission

In the course of CoCAMPUS a generic Air-Attack mission is used as its basis for system requirements and flight test evaluation. Its goal is for a single human operator to perform a Combat Air Patrol (CAP) and, upon detection of a ground vehicle, to approach and attack the target. No-fly zones as well as static threats in form of SAM-sites are to be avoided and thus restrict the available aircraft airspace. The ground vehicle is located on a straight road inside the mission area and its coordinates are given by an external reconnaissance system cooperating with the operator. The approach on the target shall be performed, optimizing the angle of attack for the UAV by finding a proper basepoint on the road and approaching the vehicle in an angle suitable for weapons release. Mission requirements were derived from necessities claimed in the current conceptional publication about the deployment of UAV by the German Ministry of Defense (BMVg).[22]

IV.B. Knowledge Modeling Methodology for COSA^2

Comparably to expert systems, COSA^2 provides a generic shell that operates on a domain specific knowledge base. This static a-priori knowledge has to be iteratively acquired by the knowledge engineer, e.g. by interviews of domain experts. As a guideline, we have derived a systematic modeling approach, that is based on object oriented analysis techniques.

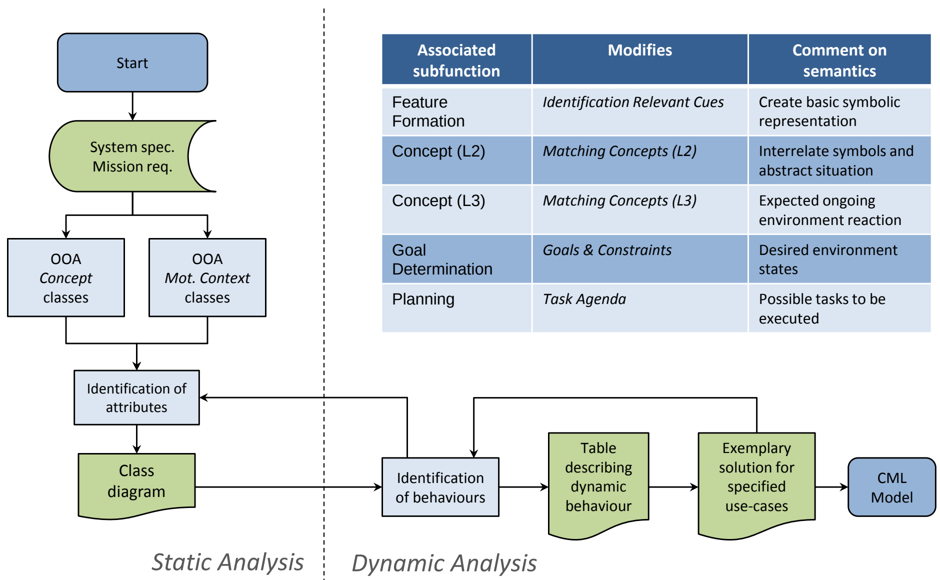

Figure 5: Knowledge modeling process for COSA^2.

As shown in the flowchart in Figure 5, the process starts with a set of system specifications and mission requirements obtained from a system level analysis. Classes are then identified from the given mission scenario. Generic mission relevant elements are enlisted under CONCEPTS, goals under MOTIVATIONAL CONTEXTS. Following traditional object oriented methodology, object properties are further described by attributes to yield a static class diagram. The second phase of the process, called the dynamic analysis, is used to model the behaviour and interactions of objects. The outcome of this phase is to associate cognitive subfunctions with the modeled real-world dynamics. As shown in the table, one indicator is the situational knowledge element the subfunctions modify, another assertion can be made using its real world semantics. This iterative process usually requires modifications to the static class diagram, and should be driven along a set of typical use cases. To verify the design, exemplary task agenda solutions to these use cases can be derived as shown for experimental results in Figure 9.

By performing the static analysis, a class diagram (cf. Figure 6) has been developed. With respect to the generic Air-Attack mission, approachable objects like cars, CAPs or UCAS home bases have been identified. All those items are possible destinations to achieve suitable mission tasks. Threats need to be avoided, thus tactical elements (e.g. SAM-sites with different radar coverage) are taken into account to initiate feasible path generations while avoiding radar detection. All elements are characterized by their positions and their characteristic traits. Those attributes are used to describe the interdependence between all different objects.

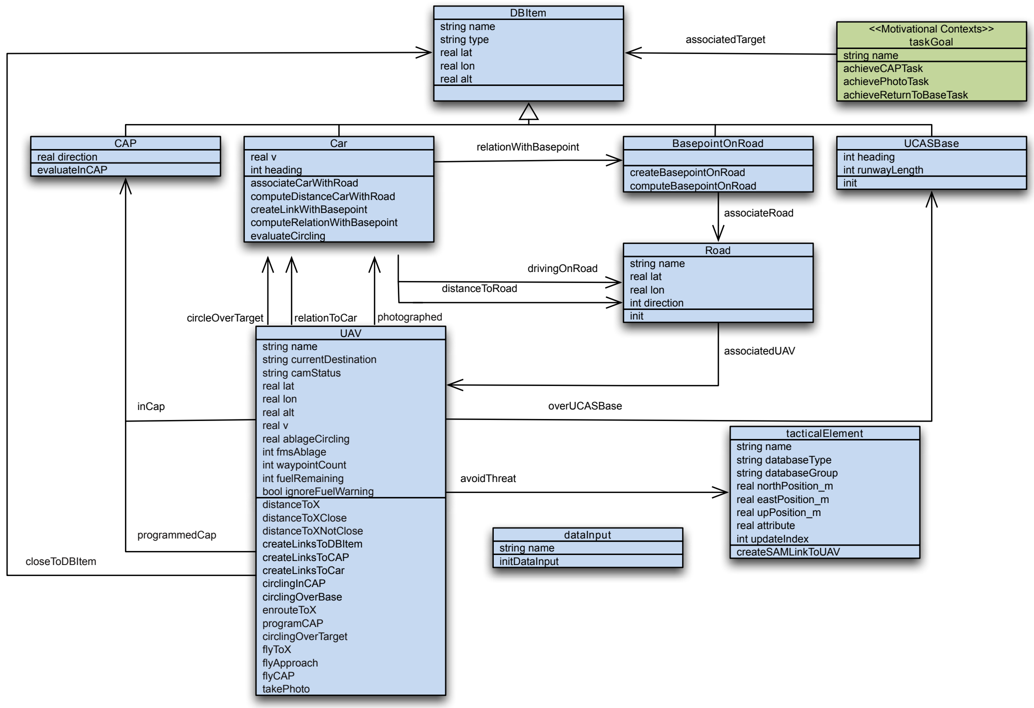

Figure 6: CoCAMPUS class diagram.

Let's consider a car, which could be a possible target. A UAV can be positioned near or far in relation to the car; furthermore the UAV can circle over it. Beside those position related statements, more use-oriented descriptions like to be photographed are possible. Similar characterizations are used for the other identified objects.

IV.C. System Architecture

Based on the results of the systems engineering methodology, a UAV system demonstrator and its experimental environment setup were developed. The resulting system architecture comprises the UAV demonstrator as well as a mobile ground control station (GCS) as the operator workstation, forming the airborne and its respective ground segment. Both segments hold various hardware elements, such as computers to host the software devices plus additional system relevant hardware. Figure 7 shows the different hardware and software devices as well as relevant hardware interfaces.

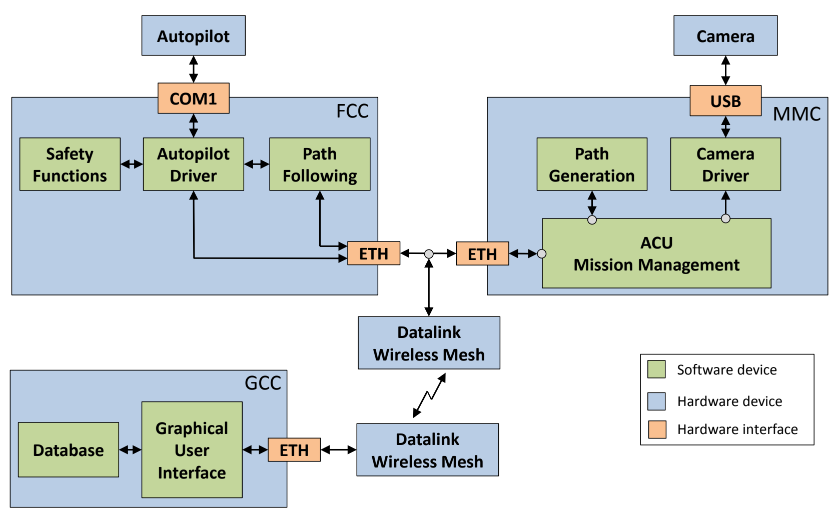

Figure 7: UAV system architecture and ground section elements.

The system's GCS is based on a Dodge Sprinter truck which is used for transporting the UAV to the flight test range and which acts as the operator working environment during flight operations. It is equipped with multiple PC workstations, LAN hardware and GPS equipment and additionally hosts the datalink components used for wireless communication between the UAV and the GCS. In a system perspective, the ground segment can be reduced to the ground control computer (GCC) with its respective software components and the datalink device. The GCC hosts the graphical user interface (GUI) as the single display and control device for the human operator. The GUI allows different levels of UAV commands, such as waypoint navigation, trajectory based Optimal Control commands as well as task based guidance using the ACU aboard the UAV. A database holds the current tactical situation including threats, obstacles and tactical elements, detected inside mission boundaries. It allows comprehensive decision making by providing congruent information for both the human operator and the onboard ACU and is updated during mission execution upon external information input and UAV object detection.

The airborne segment is based on a SIG Rascal model airplane, with a custom built payload segment inside its fuselage. It hosts two dual-core Intel Atom Pico-ITX computer boards, interconnected by a LAN hub aboard the UAV. The flight control computer (FCC) hosts all software devices for flight control purposes, such as the Optimal Control path following element, the autopilot driver and experiment-specific safety functions. The autopilot driver commands a Cloudcap Piccolo autopilot,[23] which is used as a commercial autopilot solution, connected with a RS232 serial link to the FCC board. The Piccolo device allows waypoint navigation as well as rate command modes and provides current telemetry as well as system status data. Commands to the autopilot are issued by either the operator, the ACU or by the Path Following module, tracking a generated trajectory. Flight-safety functions include airspace protection, datalink surveillance and counteracting system failures, i.e. engine kill. The mission management computer (MMC) hosts the ACU as the mission management unit, the Optimal Control path generation module and a camera driver. For experimental purposes a camera serves as the effector in the generic CoCAMPUS Air-Attack mission, giving recordable, repeatable results for post-flight analysis. The camera is connected to the MMC using its USB interface, allowing direct access for commands and imagery data. The ACU is connected to all mission-relevant devices throughout the UAV system, which offers a broad spectrum of automation elements to either task or gain access to information with.

The communication betwen the GCS and the UAV is realized via a Wireless Mesh datalink solution, serving as a bridge between the respective LANs. Our adaptation of the spread toolkit, SpreadCom[24] , serves as network based interprocess communication solution. It uses internal network ports for the exchange of data and one or multiple server located inside the LAN for its distribution. The data may include predefined structs and classes, which are serialized using the boost library[25] and is distributed in a multicast manner, with each software process registering for the data streams necessary for its operation.

V. System demonstration and experimental results

In order to test and evaluate the developed UAV setup and its implementation, a hardware in the loop (HIL) simulation has been set up. Simulated tests have shown very promising results, which led to performing flight tests in a real-world environment. These flight trials have been conducted on the test range of Camp Roberts, California Army National Guard in summer 2011. Tests included the general UAV system setup, with focus on onboard interprocess communication as well as the datalink between the ground and the airborne segment. The flight trials were conducted by a single human operator, focussing on the supporting ACU commanding the onboard automation on a task based level while performing the CoCAMPUS test mission.

During all phases of flight operation a safety pilot was present, to conduct take-off and landing procedures and relieve command of the UAV to the onboard automation in a state when operational flight-safety had been established. The safety command link was established using the radio link of the proprietary groundstation of the Piccolo autopilot bundle, to avoid interference with the wireless mesh datalink of the test setup and to ensure a high range of communication.

V.A. Flight Test Setup

The environmental conditions on the test-range of Camp Roberts led to a specific realization of the CoCAMPUS mission for the flight tests. Figure 8 shows a plot of the actual flight path of the UAV, amended by tactical elements and static threats.

The tests were conducted in restricted airspace, bounded by no-fly zones to the north and the east of

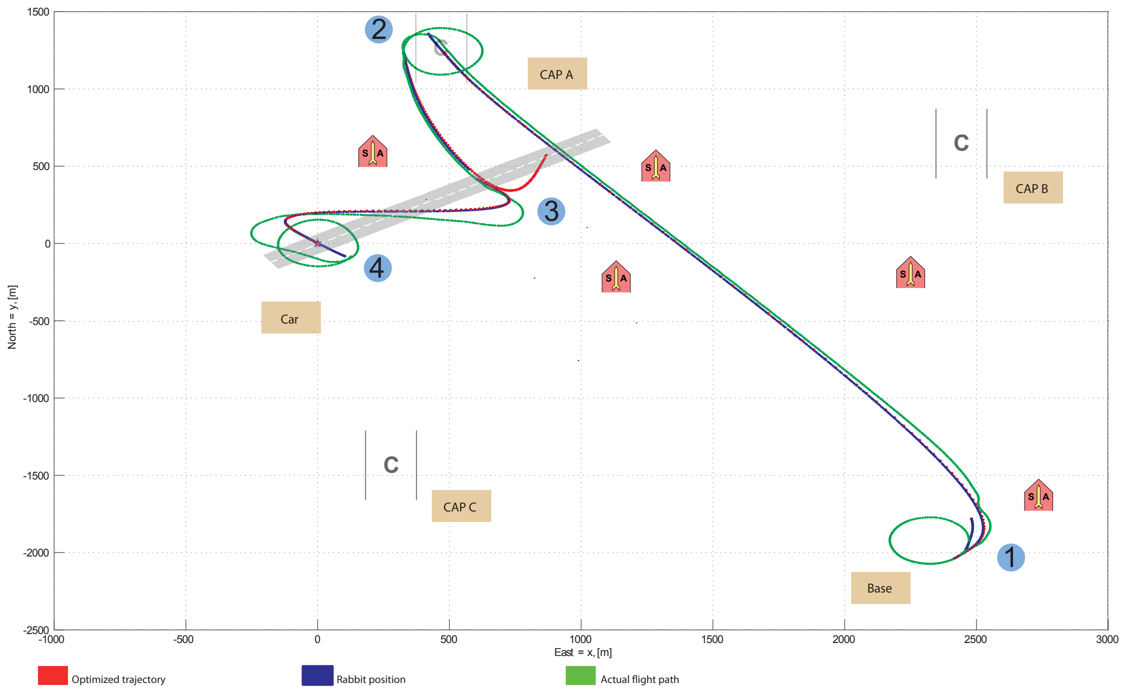

Figure 8: Flight test results, amended by tactical elements static threats.

the airstrip. The GCS was located next to the runway, which served as both the start/landing site as well as the generic street during mission execution. The target vehicle, which was to be attacked during the mission, was a specified position on the road, while the base for the experiment in its south-east direction. (Cf. Figure 8) According to section I this led to the runway being used as the generic street during mission execution. The positions of the different CAPs as well as the SAM sites were dynamically inserted into the system's database during mission execution and then loaded into the ACU's working memory. The figure shows the available CAPs as a "C" with two vertical lines to its left and right hand side, while SAM sites are shown as red polygons.

V.B. Flight Test Results

According to section II a goal violation leads to the generation of a task-agenda during the ACU's planning phase, consisting of previously specified task-options. The task-agenda is a sequence of steps taken to reclaim a desired state, combining ACU actions and projections of future environment states. The aim is to combine available automation alternatives to revert all goal violations, and thus to fulfil the operator given task. Figure 9 shows an exemplarily combined view of task-agendas derived for the goals "achieveCAPTask", "achievePhotoTask" and "achieveReturnToBaseTask", which are triggered upon the according operator given tasks. Actions are selected and linked in order to satisfy certain preconditions, given by the goals and specified during development in the a-priori knowledge of the ACU.

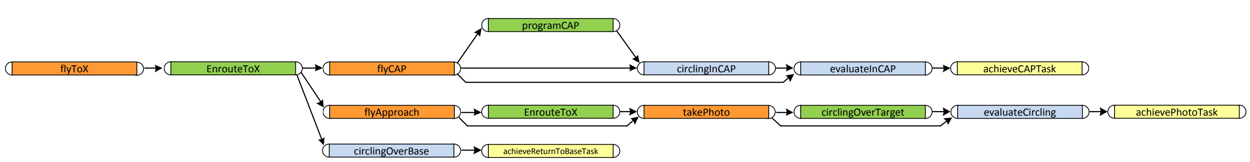

Figure 9: Combined view of task-agendas planned by the ACU.

Figure 9 illustrates the ACU's solution for the task "GoToCAP" during the execution of the CoCAMPUS mission. The incoming task triggers the according goal in the ACU, which is defined as satisfied when the UAV flight-state is successfully evaluated as circling in the specified CAP. This requires the UAV to first fly to the CAP position and then, upon arrival, start loitering. Commands to the underlying automation elements are color-coded as orange, while internal evaluations are blue and the desired goal-state is yellow. Actions that require time and thus cannot be achieved instantaniously, such as the CAP approach and the CAP entry, are environment projections of the ACU and color-coded as green. "TakePhoto" is handled and color-coded in the same manner as the previously described task. The ACU considers the goal to be achieved when a photo of the target is taken and the ACU is circling the target's position. Accordingly, the UAV first approaches a basepoint on the road that the target vehicle is on, then starts an approach to take a photo and finally circle its position.

Figure 8 shows the mission progress during the CoCAMPUS flight trials. The flight path can be roughly separated into two consecutive parts. First, the operator chose and commanded the desired CAP "CAP_A" to the ACU in point 1. Upon circling in the CAP (point 2) and reevaluating the situation, the operator tasked the ACU to approach the vehicle on the road and attack. The UAV started task execution by dynamically calculating a suitable basepoint on the target road (point 3) before using it to approach and attack the vehicle in point 4. Upon taking a picture, the ACU commanded to circle the target's position. At all times during the mission, the ACU observed the UAV's system and flight states and matched them with its own projection of future environment states, to ensure the fulfilment of the task-agenda and possibly counteract aberrations. The path generation module took into account any given threats and dynamic constraints to ensure a safe, feasable trajectory and to allow a defined approach of the target. The Path Following setup met its desired performance limits. Trajectories were approached and tracked, showing position errors of less than 50m, as aspected for path following without L1-adaptation.[9]

The following tables show a protocol of relevant communication during mission execution between the software elements of the test setup. During the mission, the human operator interacted with the UAV system using a graphical user interface (GUI), for the formulation of tasks for the ACU and to gain information about the mission progress and system states.

Time in s	Event		
0	Task: GoToCAP.		
Time in s	Sender	Receiver	Message Content
0	GUI	ACU	Task: GoToCAP, CAP_A
5,9	ACU	PG	Generate trajectory to position of CAP_A.
6,0	PG	PF	Polynominal function of trajectory.
6-102	PF	APD	Speed commands to track trajectory.
102,9	ACU	APD	Activate loitering at position of CAP_A.

Upon receiving the "GoToCAP" task by the operator through the GUI, the ACU derived a suitable task-agenda as shown in Figure 9. On completion, it commanded the Path Generation module (PG) to generate a trajectory to the CAP position at specified speed and attitude. The PG module planned the trajectory and transmitted it to the Path Following module (PF) hosted by the FCC in form of a 6th degree polynominal function. This concurrently activated path following, commanding the autopilot driver (APD) handling the Piccolo autopilot hardware device. Matching the UAV and the CAP position respectively, allowed the ACU to detect its arrival in the CAP and to command the APD to loiter, which led to task completion.

The "Take Photo" task extended the ACU's workshare of the prior task by deriving a proper basepoint position to use for the target approach, based on a perpendicular connection formulation, as part of the planning process. Upon reaching the basepoint position, the ACU commanded the generation of an additional trajectory to the target, where it used the camera driver (CamD) to take a picture of the target and loiter its position.

The tables show that the generation of path trajectories in any given situation is realized in almost real-time. The PF module could commence trajectory tracking about 0.1 seconds after the ACU commanded the PG module. Switching between speed mode navigation and waypoint tracking / loitering happened instantaneously. Processing an incoming task and deriving a task-agenda to revert resulting goal violations took a considerable amount of time. In the first case about 5.9 seconds and in the second case 7.3 seconds were necessary for the process. These durations also include reading the task and writing commands using the IO-interface of the COSA[2] architecture according to the Rasmussen layers of human behavior (Cf. section II) . Planning for the "TakePhoto" required more time, because the task is of higher complexity, involving more automation elements and thus more task options for completion. As can be seen, the ACU could almost instantaneously command the target approach after reaching the basepoint, as the prior plan already

Time in s	Event		
165,9	Task: TakePhoto.		
Time in s	Sender	Receiver	Message Content
165,9	GUI	ACU	Task: TakePhoto, CAR.
173,2	ACU	PG	Generate trajectory to position of basepoint on the road.
173,4	PG	PF	Polynominal function of trajectory.
173-225	PF	APD	Speed commands to track trajectory.
225,0	ACU	PG	Generate trajectory to position of CAR with feasible attitude.
225,2	PG	PF	Polynominal function of trajectory.
225-271	PF	ADP	Speed commands to track trajectory.
271,4	ACU	CamD	Activate camera.
271,5	ACU	APD	Activated loitering at position of CAR.

had included this step.

In a human-like manner, the ACU chooses numeric information from the symbolic elements in its working memory to formulate commands to the underlying automation. During flight trials, position data, numeric constraints for the path generation and loitering attributes are examples for such numeric information. On the other hand subsymbolic information is interpreted accordingly to gain symbolic information. Matching two positions - the target's and the current UAV's position - enables the UAV to set flags such as "close to target".

A specific flight test showcased the extraordinary performance capability and robustnes of the goal driven semi-autonomous UAV system. As a result of a datalink loss, one of the safety modules was activated during the execution of the "TakePhoto" task, which led to the execution of the homecoming-function of the UAV. Upon regaining datalink connection and thus reastablishing safe flight conditions, the ACU autonomously recognized, that the mission could still be accomplished with the current setup. It commanded the generation of a new trajectory and an approach to the target vehicle, which eventually led to a successful mission execution. The explicitly formulated goals and task options in the ACU could be used to dynamically resolve an unforseen event without an operator intervention, unlike in UAV systems using a precedural knowledge in mission guidance.

VI. Conclusion & Further Work

This paper presented an approach to integrate highly automated mission management capabilities with Optimal Control trajectory generation and following, using an adapted systems engineering approach including a worksystem analysis. A cognitive, knowledge-based agent, called an Artificial Cognitive Unit (ACU), enables a single human operator to execute a generic Air-Attack mission using task-based guidance. The ACU uses an explicit representation of existing system components to counteract goal violations by deriving and executing an appropriate task-agenda. Its implementation is based on the Cognitive System Architecture (COSA^2), where its a-priori knowledge is implemented using a novel knowledge-modeling methodology. The Optimal Control architecture accounts for static threats and dynamic constraints, while planning an optimized trajectory with operator defined intial and terminal conditions in real-time.

Flight tests have shown very promising results. The real-world flight trials have proven the feasibility and the benefits of the presented approach. The goal-driven ACU enhances the system robustness in mission execution and supports the human operator, resulting in a reduced workload.

Future work will focus on cognitive resource management, to enable the ACU to choose the most suitable out of multiple feasible task options, i.e. different route planning algorithms. Datalink management will be regarded, to allow predictable, safe UAV operation and data distribution in areas of low or jammed datalink connectivity.

Appendix

The approach to path generation exploits decoupling of spatial and temporal specifications. Let $p_c(\tau) = [x(\tau), y(\tau), z(\tau)]^\top$ denote a desired path to be followed by a single UAV in 3D space, parameterized by $\tau \in [0, \tau_f]$. For computational efficiency, assume each coordinate $x(\tau), y(\tau), z(\tau)$ is represented by an algebraic polynomial of degree N of the form

$$x_i(\tau) \quad = \quad \sum_{k=0}^{N} a_{ik}\tau^k, \qquad i = 1, 2, 3, \tag{1}$$

where we set $x_1 = x, x_2 = y, x_3 = z$ for notational convenience. The degree N of polynomials $x_i(\tau)$ is determined by the number of boundary conditions that must be satisfied. Notice that these conditions (that involve spatial derivatives) are computed with respect to the parameter τ; this parameter will be later related to actual temporal derivatives. Let d_0 and d_f be the highest-order of the spatial derivatives of $x_i(\tau)$ that must meet specified boundary constraints at the initial and final points of the path, respectively. Then, the minimum degree N^* of each polynomial in equation (1) is $N^* = d_0 + d_f + 1$. For example, if the desired path includes constraints on initial and final positions, velocities, and accelerations (second-order derivatives), then the degree of each polynomial is $N^* = 2 + 2 + 1 = 5$. Explicit formulae for computing boundary conditions $p_c'(0), p_c''(0)$ and $p_c'(\tau_f), p_c''(\tau_f)$ are given later in this section. Additional degrees of freedom may be included by making $N > N^*$. As an illustrative example, Table 1 shows how to compute the polynomial coefficients in equation (1) for polynomial trajectories of 6^{th} degree. For these trajectories, an additional constraint on the fictitious initial jerk (third-order derivatives) is included, which increases the order of the resulting polynomial and affords extra (design) parameters $x_i'''(0); i = 1, 2, 3$.

It is important to clarify how temporal constraints may be included in the feasible path computation process. A trivial solution would be to make $\tau = t$. However, very little control exists over the resulting speeds even with fifth and sixth order polynomials, because once $x_1(t), x_2(t), x_3(t)$ have been computed to satisfy the boundary constraints imposed, speed v is inevitably given by

$$v(t) \quad = \quad \sqrt{\dot{x}_1^2(t) + \dot{x}_2^2(t) + \dot{x}_3^2(t)}. \tag{2}$$

We therefore consider a different procedure that will enable meeting strict boundary conditions and constraints without increasing the complexity of the path generation process. To this effect, let $v_{\min}, v_{\max}$ and $a_{\max}$ denote predefined bounds on the vehicle's speed and acceleration, respectively. Let $\eta(\tau) = d\tau/dt$, yet to be determined, dictate how parameter τ evolves in time. A path $p_c(\tau)$ (with an underlying assignment $\eta(\tau)$) is said to constitute a *feasible* path if the resulting trajectory can be tracked by an UAV without exceeding pre-specified bounds on its velocity and total acceleration along that trajectory. With an obvious use of notation, we will later refer to a spatial path only, without the associated $\eta(\tau)$, as a feasible path.

From equation (2), and for a given choice of $\eta(\tau)$, the temporal speed $v_p(\tau(t))$ and acceleration $a_p(\tau(t))$ of the vehicle along the path (abbv. $v_p(\tau)$ and $a_p(\tau)$, respectively) are given by

$$\begin{aligned} v_p(\tau) \;&=\; \eta(\tau)\sqrt{x_1'^2(\tau) + x_2'^2(\tau) + x_3'^2(\tau)} \;\;=\;\; \eta(\tau)||p_c'(\tau)||, \\ a_p(\tau) \;&=\; = ||\ddot{p}_c(t)|| = ||p_c''(\tau)\eta^2(\tau) + p_c'(\tau)\eta'(\tau)\eta(\tau)||. \end{aligned} \tag{3}$$

Table 1. Example of computation of the coefficients of 6^{th} order polynomial paths.

	6^{th} order
Boundary conditions	$x_i(0), x_i'(0), x_i''(0), x_i'''(0), x_i(\tau_f), x_i'(\tau_f), x_i''(\tau_f)$
d_0/d_f	$3/2$
N^*/N	$5/6$
Linear algebraic matrix equation to solve for the coefficients a_{ik}	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 & 0 \\ 1 & \tau_f & \tau_f^2 & \tau_f^3 & \tau_f^4 & \tau_f^5 & \tau_f^6 \\ 0 & 1 & 2\tau_f & 3\tau_f^2 & 4\tau_f^3 & 5\tau_f^4 & 6\tau_f^5 \\ 0 & 0 & 2 & 6\tau_f & 12\tau_f^2 & 20\tau_f^3 & 30\tau_f^4 \end{bmatrix} \begin{bmatrix} a_{i0} \\ a_{i1} \\ a_{i2} \\ a_{i3} \\ a_{i4} \\ a_{i5} \\ a_{i6} \end{bmatrix} = \begin{bmatrix} x_i(0) \\ x_i'(0) \\ x_i''(0) \\ x_i'''(0) \\ x_i(\tau_f) \\ x_i'(\tau_f) \\ x_i''(\tau_f) \end{bmatrix}$

In the scope of this work we are interested in small UAVs that operate essentially at constant speeds. Clearly, in this case speed constraints can be easily satisfied for any constant $v_p \in [v_{\min}, v_{\max}]$. This in turn defines

$$\eta(\tau) = \frac{v_p}{||p_c'(\tau)||} \tag{4}$$

$$\dot{p}_c(t) = v_p \frac{p_c'(\tau)}{||p_c'(\tau)||} \tag{5}$$

Now using equations (3) and (4) we obtain

$$\ddot{p}_c(t) = \frac{v_p^2}{||p_c'(\tau)||^2}(I - \frac{p_c'(\tau)(p_c'(\tau))^T}{||p_c'(\tau)||^2})p''(\tau). \tag{6}$$

Therefore, we can choose

$$p_c'(0) = \frac{\dot{p}_c(0)}{||\dot{p}_c(0)||} \tag{7}$$

$$p_c'(\tau_f) = \frac{\dot{p}_c(t_f)}{||\dot{p}_c(t_f)||}. \tag{8}$$

to satisfy boundary conditions on $\dot{p}_c(t)$. Similarly, setting

$$\begin{aligned} p_c''(0) &= \ddot{p}_c(0) \\ p_c''(\tau_f) &= \ddot{p}_c(t_f), \end{aligned}$$

satisfies equation (6) at the boundaries.

On the other hand, the total acceleration a_p of a vehicle flying along the path $p_c(\tau)$ at a constant speed is the product of the curvature of the path with its velocity along the path squared. The curvature of the path $p_c(\tau)$ is given by

$$\kappa(\tau) = \frac{1}{||p'(\tau)||}||\frac{d}{d\tau}\frac{p_c'(\tau)}{||p_c'(\tau)||}||.$$

Thus, using simple algebra it can be shown that

$$\begin{aligned} a_p(\tau) &= v_p^2 \kappa(\tau) \\ &= \frac{v_p^2}{||p_c'(\tau)||^2}||(I - \frac{p_c'(\tau)(p_c'(\tau))^T}{||p_c'(\tau)||^2})p''(\tau)||, \end{aligned}$$

which as expected is the norm of $\ddot{p}_c(t)$ (see equation (6). Therefore, for the case of constant velocities v_p a feasible path must satisfy the following set of constraints

$$v_{\min} \leq v_p \ \leq v_{\max}, \quad a_p(\tau) \leq a_{\max}, \ \forall \tau \in [0, \tau_f]. \tag{9}$$

for a pre-specified acceleration bound $a_{\max}$.

In this paper, we use this simple definition of a feasible path to address the problem of a mission planning of a tactical UAV whereby the UAV must avoid detection by a radar and accomplish the mission in a minimum time. The approach proposed here finds a feasible path that makes the minimum time mission planning problem easily solvable by a single UAV flying at constant speeds along the path. Next, we make these ideas more precise.

Let l_f denote the total path length and v_p denote its velocity along this path. Then

$$l_f = \int_0^{\tau_f} ||p_c'(\tau)|| \, d\tau.$$

It follows immediately that the time of flight t_f of UAV is given by

$$t_{f_{\min}} \quad = \quad \int_0^{\tau_f} \frac{||p_c'(\tau)||}{v_{\mathrm{p}}} \, d\tau.$$

Define a cost function $J = t_f$. Then, making J arbitrarily small over the set of feasible paths, feasible velocities and accelerations will result in the desired solution to the minimal time problem discussed above. Therefore, we propose to solve the following path generation problem

$$F : \begin{cases} \min\limits_{\tau_f, v_p} \{J\} \\ \textit{subject to boundary conditions and limitations } (9) \\ \ \textit{while} \min\limits_{j=1,\ldots,n} ||P_{det_i}(\tau)||^2 \leq E^2 \end{cases} \tag{10}$$

Solution to the optimization problem F includes an optimal paths and constant speed profile that together minimize J subject to boundary conditions and control limitations in equation (9) and an additional penalty function Δ that besides penalizing the UAV states $\eta(\tau)$, controls $\xi(\tau)$ and their derivatives $\pi(\tau)$ defines the UAV's detection probability P_{det_i} by a set of $i = 1, \ldots, n$ radars. The choice of penalty function is what makes the optimization task relevant to the tactical sense of the UAV mission. In application to the task at hand it can be represented as follows:

$$\begin{aligned} \Delta = \textstyle\sum_{j=1}^{k} w_j \cdot max(0; \eta - \eta_{bound})^2 + \sum_{j=1}^{l} w_{j+k} \cdot max(0; \xi - \xi_{bound})^2 \\ + \textstyle\sum_{j=1}^{m} w_{j+k+l} \cdot max(0; \pi - \pi_{bound})^2 + \sum_{i=1}^{n} w_{j+k+l+m+n} \cdot max(0; P_{det_i} - P_{bound})^2 \end{aligned} \tag{11}$$

where the values of η_{bound},ξ_{bound},π_{bound} are the design bounds defined a priory, and w_j are the weight coefficients ($\sum_{j=1}^{j+k+l+m+n} w_j = 1$) that are chosen heuristically to ensure specified accuracy of matching the constraints. As a result the optimization problem is reduced to a nonlinear programming problem with an objective of minimizing the scalar function of a limited number of variables.

The optimization problem F can be effectively solved in near real-time by constructing a penalty function G as discussed in works[26,18] and by using any zero-order optimization technique like the Hooke-Jeeves pattern direct-search algorithm[27] or Nelder-Mead downhill simplex algorithm.[28] In the task at hand the Hooke-Jeeves optimization algorithm was implemented online at 4 Hz thus producing a near optimal solution not faster than 4 times a second. Constraining the update rate of the optimization code allowed balancing the computational load of the onboard CPU.

References

[1]Meitinger, C., Jarasch, G., Schulte, A., *Development of Artificial Cognitive Units in UAV Guidance*, NATO SCI-202 Symposium on Intelligent Uninhabited Vehicle Guidance Systems, 2009.

[2]Onken, R., Schulte, A., *System-ergonomic design of cognitive automation: Dual-mode cognitive design of vehicle guidance and control work systems*, Heidelberg: Springer-Verlag, 2010.

[3]Cummings, M.L., Kirschbaum, A.T., Sulmistras, A., Platts, J.T., *STANAG 4586 Human Supervisory Control Implications*, Air and Weapon Systems Dept. Dstl Farnborough & the office of Naval Research, 2006.

[4]Kilgore, R.M., Harper, K.A., Nehme, C., Cummings, M.L., *Mission planning and monitoring for heterogenous unmanned vehicle teams: A humand-centered perspective*, AIAA Infotech@Aerospace Conferece, 2007.

[5]Jones, R.M., Laird, J.E., et. al., *Automated intelligent pilots for combat flight simulation*, AI magazine, Vol. 20, No.1, pp.27, 1999.

[6]Howitt, S., Richards, D., *Human Interaction with Teams of Autonomous UAVs*, 1st Moving autonomy forward conference, 2006.

[7]Baxter, J.W., Horn, G.S., *A multi-agent system for executing group tasks*, Palade, V., Howlett, R.J., Jain, L.C., KES 2003, pp. 697-703, 2003.

[8]Kaminer, I., Yakimenko, O., Pascoal, A., Ghabcheloo, R., *Path Generation, Path Following and Coordinated Control for Time-Critical Missions of Multiple UAVs*, American Control Conference, 2006.

[9]Kaminer, I., Yakimenko, O., Dobrokhodov, V., et. al., *Coordinated Path Following for Time-Critical Missions of Multiple UAVs via L1 Adaptive Output Feedback Controllers*, AIAA Guidance, Navigation and Control Conference and Exhibit, 2007.

[10]Dobrokhodov, V., Kaminer, I., et.al., *Flight Validation of a Metrics Driven L1 Adaptive Control*, AIAA Guidance, Navigation and Control Conference and Exhibit, 2008.

[11]Uhrmann, J., Strenzke, R., Rauschert, A., Schulte, A., *Manned-unmanned teaming: Artificial cognition applied to multiple UAV guidance*, NATO SCI-202 Symposium on Intelligent Uninhabited Vehical Guidance Systems, 2009.

[12]Uhrmann, J., Strenzke, R., Schulte, A., *Human Supervisory Control of Multiple UAVs by use of Task Based Guidance*, Conference on Humans Operating Unmanned Systems (HUMOUS'10), 2010.

[13]Jarasch, G., *Design Methodology for an Artificial Cognitive System applied to human-centred semi-autonomous UAV Guidance*, Conference on Humans Operating Unmanned Systems (HUMOUS'10), 2010.

[14]Kriegel, M., Schulte, A., Stütz, P., *Testbed Environment for future UAV Automation Technologies*, 2nd International Workshop on Aircraft System Technologies, Hamburg, 2009.

[15]S. Brüggenwirth, W. Pecher, A. Schulte, *Design Considerations for COSA2*, IEEE Symposium Series on Computational Intelligence, SSCI 2011, Paris, 2011

[16]Jens Rasmussen., *Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models*, IEEE Trans. Systems, Man, Cybernetics, SMC-13, pp. 257-266, 1983

[17], Kaminer I.I., Pascoal A.M., Xargay E.M., Hovakimyan N., Cao C.C., Dobrokhodov V.N., *Path Following for Unmanned Aerial Vehicles using $\mathcal{L}_1$ Adaptive Augmentation of Commercial Autopilots"*, AIAA Journal of Guidance, Control & Dynamics, 33(2):550-564, 2010.

[18]Yakimenko, O. A., *Direct method for rapid prototyping of near-optimal aircraft trajectories.*, AIAA Journal of Guidance, Control, & Dynamics, 23(5):865–875, 2000.

[19]Douglass, B.P., *The Harmony Process*, I-Logix, Inc., 2005.

[20]Kriegel, M., Brüggenwirth, S., Schulte, A., *Knowledge Configured Vehicle - A layered artificial cognition based approach to decoupling high-level UAV mission tasking from vehicle implementation properties*, AIAA GNC 2011, Portland, 2011.

[21]Hoffmann, H.P., *Harmony/SE: A SysML Based Systems Engineering Process*, Innovation 2008 Telelogic User Group Conference, 2008.

[22]BMVg, *Konzeptionelle Grundvorstellungen zum Einsatz unbemannter Luftfahrzeuge (Unmanned Aerial Vehicles) in der Bundeswehr (KGv UAVBw)*, 2008.

[23]Cloudcap Technology, *Piccolo II autopilot*, `http://www.cloudcaptech.com/piccolo_II.shtm`, accessed 04-18-2012.

[24]Böhm, F., *Scalable COTS based data processing and distribution architecture for UAV technolody demonstrators*, Accepted at European Telemetry and Test Conference etc2012, Munich, 2012.

[25]Boost C++ Libraries `http://www.boost.org`, accessed 04-20-2012.

[26]Dobrokhodov,V. N., and Yakimenko, O. A., *Synthesis of Trajectorial Control Algorithms at the Stage of Rendezvous of an Airplane with a ManeuveringObject*, Journal of Computer and Systems Sciences International, 38(2):262277, 1999.

[27]Hooke, R., and Jeeves, T. A., *Direct Search Solution of Numerical and Statistical Problems*, Journal of the Association for Computing Machinery, 8(2):212-229, 1961.

[28]Nelder, J. A., and Mead, R.,*A Simplex Method for Function Minimization*, The Computer Journal, 8(7):308-313, 1965.