Bridging the mission-control gap: A flight command layer for mediating flight behaviours and continuous control.

Piotr Rudol[1] and Patrick Doherty[1]

Abstract— **The use of UAVs, in particular, micro VTOL UAVs, is becoming prevalent in emergency rescue and security applications, among others. In these applications, the platforms are tightly coupled to the human users and these applications require great flexibility in the interaction between the platforms and such users. During operation, one continually switches between manual, semi-autonomous and autonomous operation, often re-parameterising, breaking in, pausing, and resuming missions. One is in continual need of modifying existing elementary actions and behaviours such as FlyTo and TrackObject, and seamlessly switching between such operations. This paper proposes a flight command and setpoint abstraction layer that serves as an interface between continuous control and higher level elementary flight actions and behaviours. Introduction of such a layer into an architecture offers a versatile and flexible means of defining flight behaviours and dynamically parameterising them in the field, in particular where human users are involved. The system proposed is implemented in prototype and the paper provides experimental validation of the use and need for such abstractions in system architectures.**

I. Introduction

Technological advances with Unmanned Aerial Vehicles are now at a point where these systems are seeing increasing use in many application areas, including that of emergency rescue. With this increase in the use for complex applications comes a need to be able to operate these systems in a flexible and robust manner. Common modes of operation combine manual control, using RC-control and joysticks, together with semi-autonomous and autonomous modes. During a particular mission all of these modes may in fact be used in an interleaved manner, placing new constraints on how control architectures should be embedded and interfaced into larger system architectures to allow as much flexibility and robustness as possible in the human operation of such systems. Finding the optimal way to combine mission abstraction levels with elementary command and conventional control levels is very much an open issue.

At a mission abstraction level, an operator might have the high-level goal of scanning a region in search of salient objects such as injured people. In order to scan a region, an appropriate region and scan pattern in that region should be delineated. Additionally, many other parameters associated with the mission such as velocity or altitude must be determined. Many of these constraints on missions in the field are highly context dependent. Rescuers need to adapt a basic mission to contingencies at hand on-the-fly and in real-time.

In current state-of-the-art, one assumes a basic set of well-defined elementary actions, or commands that can be sequentialised either manually, through operator interfaces, or less commonly as the output of task and motion planners or their combination. For VTOLs (e.g. see Figure 1), examples of such elementary actions are TakeOff, Hover, FlyTo, TrackTarget, Yaw, and Land. These elementary actions are often hard-wired into the architecture and their parameterisation occurs internally relative to the constraints associated with the control system in a platform.

Fig. 1: LinkQuad 4.2 quadrotor platform used for experimental validation.

Such hard-wiring of elementary actions runs counter to the operational needs of a rescuer in the field and makes it difficult for high-level task planners to generate plans that permit a wide variety of operational and safety constraints to be specified for specific mission tasks. Ideally, one would want to be able to contextualise not only mission tasks, but the elementary actions themselves both for pre-flight planning, but also during operation.

Some examples of what we have in mind would include an operator being able to modify altitude relative to terrain, either choosing a constant altitude or allow for flying relative to terrain to maintain the same image resolution for a sensor and maximise the chances of finding a salient object. During the mission, an operator may want to suspend the execution of a planned mission and take direct control of the platform to fly interactively and investigate more closely a salient point identified in a video feed before resuming the scanning mission. After resuming, an operator may want to modify heading behaviour to allow the platform to always point towards a specific identified human on the ground. An operator might want to continually intercede and change the velocity of the platform relative to the complexity of the search environment in addition to easily taking full velocity control and command the platform using a joystick. Finally, one would want to flexibly be able to apply safety constraints at the mission and elementary action level. For instance, no matter what elementary action is executed, the platform can not fly outside a specified region, higher then a specified altitude or closer to a human object than a specified safety

This work is partially supported by the Swedish Research Council (VR) Linnaeus Center CADICS, the ELLIIT network organisation for Information and Communication Technology, and the Swedish Foundation for Strategic Research (CUAS Project and SymbiKcloud Project).

[1]Piotr Rudol and Patrick Doherty are with the Department of Computer and Information Science, Linköping University, SE-58183 Linköping, Sweden piotr.rudol@liu.se, patrick.doherty@liu.se

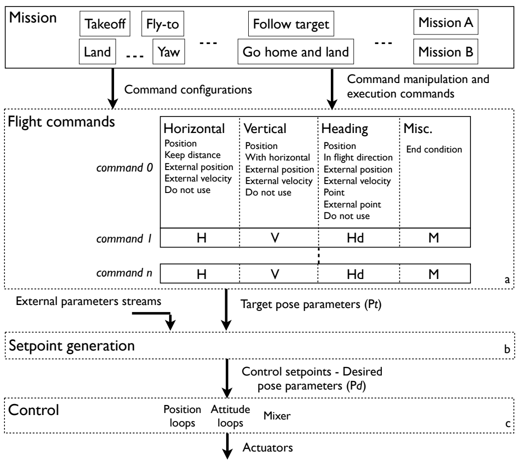


Fig. 2: System overview with the main elements denoted by the dashed lines: flight commands (a), setpoint generation (b) and a control system scheme (c).

distance.

Operating UAVs requires low-level control systems which provide all the required building blocks for controlling quantities such as accelerations, rates, velocities, and positions. Defining missions at that level is impractical. Instead, it is natural to define them in terms of behaviours or elementary actions. The above examples show, however, that the ability to richly parametrise such behaviours and elementary actions as well as the freedom to seamlessly switch between them is needed. This of course has to be achieved without compromising safety and therefore the ability to impose constraints on these behaviours and elementary actions both statically and dynamically is essential. Defining flight behaviours at an appropriate level of abstraction is necessary not only to take advantage of automated planning techniques but also to provide an added degree of flexibility in the design and specification of Human-Machine Interfaces. These requirements would facilitate more effective use and operation of UAV platforms in emergency assistance scenarios.

The main idea of this paper is to add an additional level of command abstraction between elementary actions and conventional continuous control kernels. At this abstraction level, elementary actions can be parametrised more richly and defined in terms of sequences of *flight commands* which in turn interface to the control kernel in a safe and robust manner.

Our contribution in this regard is threefold. *First*, we propose the idea of configurable *flight commands* which govern *horizontal*, *vertical* and *heading* control channels (Figure 2a). A list of such parametrised commands, along with optional external target value streams and house-keeping commands, allows for specifying a wide range of flight behaviours, from a simple *takeoff* or *flyto* to a complete mission from takeoff to landing (see Section II). *Second*, we propose the use of a setpoint generation step which, on the input side, is configured using the *flight commands* and, on the output side, allows for specifying constraints on accelerations, speeds and positions being generated (Figure 2b). The purpose of this module is to assure continuity and bounds of the setpoints for the underlying control system. This is in contrast to the target and externally provided parameters, which can change in an arbitrary way. Additionally, it increases safety of operation by, for example, making it impossible to command a UAV to leave a designated operational area or to exceed a safe speed in the context of a mission being executed. It also alleviates the problems associated with traditional mode switching as it assures continuity of setpoint signals for all possible flight commands. *Third*, we present a lightweight control system scheme based on cascaded PID controllers suitable for executing flights from takeoff to landing (Figure 2c). A description of the setpoint generation module and the control system structure proposed in this work is presented in Section III.

A. Related work

Flight capabilities and control schemes for UAVs have been heavily researched allowing for flying a wide range of manoeuvres – from waypoint following [1], through takeoffs [2] and landings [3] to, for example, employing machine learning to improve the flight performance over time [4]. This has contributed to the success of UAVs in emergency assistance scenarios such as wilderness search and rescue [5] in addition to urban operations [6].

Software architectures for mobile robots have gravitated toward the use of three-layered (3T) structures, where each layer has specific temporal properties associated with latencies in decision cycles. The first *reactive layer*, with the shortest cycles (milliseconds), consist of a set of reactive skills which map sensors directly onto actuators with minimal internal state. The second *sequencing layer*, with a larger decision cycle (seconds), is responsible for sequencing of activities. The third layer, with the longest decision cycles measured in seconds or even minutes, is the *deliberative layer*, which is responsible for reasoning about mission goals through the use of search-based techniques [7].

The traditional 3T structure has been extended in a number of ways. For example, a four layer variant has been presented in [8], and an architecture with a more *hybrid* flavour and slightly different naming of the layers, *deliberative, reactive and control* (HDRC3), has been proposed in [9]. A mission management system based on a 3T architecture for a VTOL UAV has been proposed in [10]. The paper presents the overall structure of an architecture and proposes a *supervisory system* that interacts with a *sequence control system*. It, in turn, contains *movement primitives*. The model of the transition between these primitives includes a *slow down* state which brings the system into a *stand by* mode. Similar facilities exist in the HDRC3 architecture, where a *braking* mode brings a system into a *hover* state. This results in *fly-brake-hover-fly* behaviour when switching between different flight behaviours.

The work proposed in this paper, is situated as part of the control and reactive layers of an architecture and has a clear interface with the deliberative layer. The proposed structure is fully compatible with existing high-level mission planning systems such as the one presented in [11], but also with most conventional control kernels. Not only can the flight command abstraction be easily employed to define *elementary actions* such as *hover-at* and *fly-to*, in addition to more complex flight behaviours such as *scan-region*, but it also provides an additional level of flexibility by making it straightforward to mix fully autonomous operation with semi-autonomous operation where operator interaction is heavily involved.

Control of multi-rotor UAVs is challenging because of system nonlinearities, fast dynamics, and cross couplings due to the gyroscopic moments and underactuation. There has been a substantial amount of research dealing with these control issues but often in separation from the overall integration into a full architecture such as the 3T architectures described above, where this body of work would basically implement the *movement primitives*.

A comprehensive survey of methods used for rotorcraft control can be found in [12]. In summary, theses approaches can be categorised into three major groups: learning-based (e.g. reinforcement learning in [13]), model-based nonlinear (e.g. Model Predictive Control (MPC) [14]) and linear. There also exist approaches where different controllers are used and mode switching between them is employed. For example, these techniques have been applied to trajectory generation and control in the context of precise aggressive manoeuvres [15].

Despite that fact that linear controllers suffer from performance degradation when operated outside their nominal conditions, they are the most prevalent for deployed flight controllers. In this category, PID controllers are used most often. They are also arguably the least computationally expensive making them suitable for microcontroller implementations. Additionally, from the perspective of the type of mission scenarios we are interested in here, they are adequate, as no aggressive manoeuvring is required.

II. Flight command structure

Flight commands are a central component of the proposed method for interfacing with an underlying control system. A *flight command* consists of three main components managing the control channels, namely *horizontal*, *vertical*, and *heading*, as well as a *miscellaneous* component dealing with aspects such as an end condition of a command.

The choice of modes, parameters and modifiers has been designed to achieve the required level of flexibility with minimal complexity in implementation. The goal is to be able to cover most of the typical VTOL UAV mission types and behaviours used in emergency rescue. This takes into account not only typical flight manoeuvres (e.g. fly-to, takeoff, land), but also requirements and opportunities provided by use of specific types of sensors (e.g. speed modification based on altitude), and additionally, aspects of missions with several cooperating UAVs (e.g. execution synchronisation).

A. Horizontal channel

The horizontal channel governs movement of a UAV on a plane. Even though it could be further subdivided into lateral and longitudinal channels, this would not add much to the expressivity or range of manoeuvres which can be performed. Additionally, such subdivision would make it more difficult to assure that the channels are parametrised correctly (e.g. handling coordinates in a world or body coordinate systems).

The following modes with parameters and modifiers (see Figure 3a) are available for the horizontal control channel:

- **Position** (H_{Pos}): Fly to an *absolute*, *relative* or *body relative* 2D position with the specified *speed* and have the specified *end speed* upon arrival. Speed can be modified when used with appropriate vertical or heading commands (see below).
- **External position** (H_{ExtPos}): Similar to H_{pos} but the parameters are provided externally as a stream of values. *Offset* parameters allow to specify a constant offset from a target position (e.g. 2m to the north of the target).
- **Keep distance**: Fly at a *distance* away from an externally provided target position.
- **External velocity** (H_{ExtVel}): Fly with the velocities provided from an external source (e.g. a joystick).
- **RC velocity** (H_{RCVel}): Fly with the velocities provided by the RC transmitter of the backup pilot calculated on-board the platform.
- **Do not use** (H_{DNU}): Do not use the horizontal flight channel i.e. the setpoints governing this channel will remain constant.

Upon arriving at the desired position, the *end flag* (H_{end}) is set and can be used to indicate the end of the current command (see Subsection II-D.1). The end condition is evaluated based on the position setpoints rather than the actual values (e.g. GPS position). This results in the setpoints arriving exactly at the the desired values (e.g. a waypoint).

The speed modifier flags (i.e. *vertical* and *heading*) allow for influencing (or modulating) the speed of flight in relation to other control channels. For example, using the *heading* flag, it can be specified using one command that while the heading is changing towards a target, the UAV should not start flying to a specified waypoint (i.e. "yaw before flight" behaviour). Similarly, the *vertical* flag can be used when, for example, flying with a constant distance to the ground (measured with an appropriate AGL sensor). In case of an abrupt measured altitude change, the flight in the plane direction would be stopped until the new desired altitude is achieved. Examples of using these flags in flights are provided in Section IV.

B. Vertical channel

The following modes with parameters and modifiers (see Figure 3b) are available for the vertical control channel:

- **Position** (V_{Pos}): Equivalent of H_{Pos} for the vertical channel i.e. fly to a specified *absolute* or *relative* altitude with specified *speed* and have *end speed* upon arrival.
- **With horizontal** ($V_{WithHor}$): Extends the H_{Pos} to a 3D case, and results in flying a straight line path to a waypoint specified including the altitude.
- **External position** (V_{ExtPos}): Equivalent of H_{ExtPos} for altitude. *Offset* allows to specify a constant vertical offset from a target (e.g. 2 m above the target).
- **External velocity** (V_{ExtVel}): Fly with the vertical velocity provided from an external source (e.g. a joystick).
- **RC velocity** (V_{RCVel}): Fly with the vertical velocity provided by the RC transmitter of the backup pilot calculated on-board the platform.
- **Do not use** (V_{DNU}): Do not use this flight channel.

The channel has two additional flags: *takeoff* and *allow landing*. These are used to indicate that the commands are in fact takeoff or landing manoeuvres, respectively. The two are described in more detail in sections III-C.1 and III-C.2.

Mode name	Parameters	Modifiers	Output
Position (H_{Pos})	north east	absolute relative body relative	end flag (H_{end})
	speed	vertical, heading	
	end speed	-	
External position (H_{ExtPos})	offset north offset east	-	end flag (H_{end})
	speed	vertical, heading	
Keep distance ($H_{KeepDist}$)	distance	-	end flag (H_{end})
	speed	vertical, heading	
External velocity (H_{ExtVel})	-	-	end flag (H_{end})
RC velocity (H_{RCVel})	-	-	-
Do not use (H_{DNU})	-	-	-

(a) Horizontal

Mode name	Parameters	Modifiers	Output
Position (V_{Pos})	altitude	absolute relative	end flag (V_{end}) speed modifier (V_{sm})
	speed	heading	
	end speed	-	
With horizontal ($V_{WithHor}$)	altitude	-	end flag (V_{end})
External position (V_{ExtPos})	offset altitude	-	end flag (V_{end}) speed modifier (V_{sm})
	speed	heading	
External velocity (V_{ExtVel})	-	-	end flag (V_{end})
RC velocity (V_{RCVel})	-	-	-
Do not use (V_{DNU})	-	-	-

(b) Vertical

Mode name	Parameters	Modifiers	Output
Position (Hd_{Pos})	heading	absolute relative stop at heading	end flag (Hd_{end}) speed modifier (Hd_{sm})
	rate	heading	
In flight direction ($Hd_{FlightDir}$)	offset	-	speed modifier (Hd_{sm})
	rate	-	
External position (Hd_{ExtPos})	offset	-	end flag (Hd_{end}) speed modifier (Hd_{sm})
	speed	heading	
External velocity (Hd_{ExtVel})	-	-	end flag (Hd_{end})
RC velocity (Hd_{RCVel})	-	-	-
Point (Hd_{Point})	north east offset rate	-	end flag (Hd_{end})
External point ($Hd_{ExtPoint}$)	offset	-	end flag (Hd_{end})
Do not use (Hd_{DNU})	-	-	-

(c) Heading

Fig. 3: Control channel modes with parameters and modifiers.

C. Heading channel

The following modes with parameters and modifiers (see Figure 3c) are available for the heading control channel:

- **Position** (Hd_{Pos}): Set the heading to a specified *absolute* or *relative* value with the specified *rate*. If *stop at heading* is false, the yawing will be carried out continually with the specified *rate*. The *use sign* flag allows to force the rotation direction. If set to false, the rate sign will be ignored and the closer rotation direction will be used.
- **In flight direction** ($Hd_{FlightDir}$): Set the heading to the *offset* value relative to the flight direction (e.g. heading along the path when *offset* is 0, or fly "backwards" for *offset* of 180 degrees).
- **External position** (Hd_{ExtPos}): Set the heading to a value provided from an external source.
- **External rate** (Hd_{ExtVel}): Yaw with the rate provided from an external source (e.g. a joystick).
- **RC rate** (Hd_{RCVel}): Yaw with the rate provided by the RC transmitter of the backup pilot calculated on board.
- **Point** (Hd_{Point}): Set the heading to point towards a specified location plus an *offset*.
- **External Point** ($Hd_{ExtPoint}$): Similar to Hd_{Point} but target location point is provided externally.
- **Do not use** (Hd_{DNU}): Do not use this flight channel.

D. Miscellaneous

Beside the above described channel components of a flight command, it also consists of the elements described below.

1) Command end condition: This allows for specifying when the current command should be considered finished. The end condition consists of a number of elements with parameters summarised in the table in Figure 4. The three elements are evaluated as a conjunction if more than one type is used at a time.

The first type of end condition is related to the three command channels. Depending on their configuration, H_{end}, V_{end}, Hd_{end} flags are set to true when appropriate. This happens, for example, when a waypoint position is reached, or the desired altitude or heading are achieved. Then, these flags are evaluated if E_H, E_V, E_{Hd} flags (respectively) of the end condition are specified (or if any of the channel flags are true for E_{Any}).

Name	Parameters	
Channel	horizontal (E_H)	any (E_{Any})
	vertical (E_V)	
	heading (E_{Hd})	
User confirmation	- (E_{User})	
Wait	time (E_{Wait})	

Fig. 4: Flight command end condition structure with parameters.

The second type of the end condition can be specified to allow finishing a command only if an explicit user confirmation is provided through an external signal (E_{User}). The confirmation can be performed at any point of execution of the current command. This user confirmation functionality allows to specify a command (when H_{DNU}, V_{DNU}, Hd_{DNU} modes are used) with an infinite wait requiring an input. This facility can be used before a landing command of a mission, so that an operator can explicitly confirm that the landing location is safe and the manoeuvre can commence. Similarly, it can be used to synchronise mission execution if several UAV platforms take part in a mission.

The third type allows for specifying a timeout which triggers ending of a command (E_{Wait}). If a flight command consists only of a *wait* end condition it effectively becomes a pause.

2) External data: A number of flight command modes allow the system to be configured to accept external streams of data influencing the flight behaviour. The purpose of these modes is to allow more flexibility when it comes to changing flight behaviour in a timely manner, for example, in a closed loop control fashion. A typical use case is a joystick commanding velocities to the platform. Only one flight command is used in such a case along with a stream of the desired velocities provided continuously and controlling one or more channels (see examples in Section IV).

Beside the parameters required for the particular modes, which are the same as for their non-external counterparts (c.f. tables in Figure 3), other kinds of data are also sent with the external data streams. For velocity commands, a validity time of the data is included. After this time, if no new values are received, the system sets the desired values to 0. This prevents a platform from continuing to fly if communication is lost with the entity providing the data.

3) Composing flight commands: The previous sections introduced the concept of flight commands which can be used to specify a particular flight behaviour. The commands are put on a list or a queue. The list manipulation operators include uploading command configuration at a specific index, starting and stopping execution. It is up to a user to manipulate the command execution list as desired to achieve the behaviours with the required level of sophistication.

From the perspective of a higher level interface, a behaviour can be composed of one or more flight commands. For example, a takeoff, a landing or flying to a waypoint can be implemented using a single flight command, while a behaviour "go home and land" can be composed of a number of sequential commands. Furthermore, a sequence of commands can implement a complete mission from takeoff to landing including complex behaviours in-between.

More advanced flight command list manipulation possibilities will be further explored in future work, but the current set of operators permits specifying and executing of adequately complex autonomous missions.

III. Setpoint generation and control

The interplay of the setpoint generation step with the underlying control system plays a major role in the proposed system. First, it allows for arbitrary and seamless change or adjustment of commands being executed. For example, during flying to a waypoint: it can be switched to another waypoint; the speed of flight can be adjusted; an operator can take full or partial control (e.g. adjust altitude, heading), or it can directly transition into a landing etc. Second, it assures that the generated setpoints are compatible with the underling low-level control system and platform's properties as well as the requirements of a mission being executed. Additionally, limits on the generated values are enforced at all times contributing to the safety of execution.

In this section we describe how the flight commands introduced in the previous section are used in the process of calculating the control signals. Figure 5 presents an overview of that process.

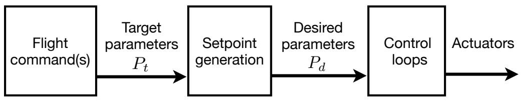


Fig. 5: An overview of the process of calculating control signals based on flight commands.

The configuration of a flight command being executed produces a set of target parameters $P_t = [\vec{p}_t, \psi_t, speed, speed_{end}]$. Depending on the used modes of the three channels, the set of target parameters varies. Similarly, the *speed* can be given for 2D and altitude channels separately ($\mathrm{H_{Pos}}$, $\mathrm{V_{Pos}}$), or as one parameter for 3D case ($\mathrm{H_{Pos}}$, $\mathrm{V_{WithHor}}$). Just as the specific set of parameters, their values can have arbitrary, non-continuos, values. The role of the setpoint generation module is to produce trajectories of desired parameters for positions, velocities, and accelerations: $P_d = [\vec{p}_d, \vec{v}_d, \vec{a}_d]$ based on P_t. This is achieved using the constant acceleration model and taking into account the constraints in form of allowed maximum values $P_{env} = [\vec{p}_{env}, \vec{v}_{env}, \vec{a}_{env}]$.

A. Setpoint generation

Given the current desired position setpoints as well as the a new target, the direction vector is calculated $\vec{dir} = \vec{p}_t - \vec{p}_d$. Next, the desired velocity vector is calculated based on the wanted *speed*:

$$\vec{v}_d = \hat{dir} \cdot speed = \frac{\vec{dir}}{\parallel \vec{dir} \parallel} \cdot speed \tag{1}$$

If needed, it is then scaled to not exceed maximum allowed speeds given by $\vec{v}_{env}$.

At this point $\vec{v}_d$ contains velocities for the three axes compatible with the velocity envelope. The values might have to be further modified to take into account the distance to the desired position ($dist_d$) as well the limit imposed by the position envelope ($dist_{env}$):

$$dist_d = \parallel p_d - p_t \parallel \tag{2}$$

$$dist_{env} = \parallel p_{env} - p_t \parallel \tag{3}$$

Given distances to limits, desired velocity v_d and the end velocity v_{end} (calculated using Equation 1 for $speed_{end}$) the following equations (constant acceleration model) are used to limit the velocity to guarantee p_d to stay within bounds:

$$v_{max} = \sqrt{v_{end}^2 + 2 \cdot dist_t \cdot a} \tag{4}$$

$$v_{env} = \sqrt{2 \cdot dist_{env} \cdot a} \tag{5}$$

$$v_d = min(v_d, v_{max}, v_{env}) \tag{6}$$

and the final value of the position is updated according to:

$$p_d = p_d + v_d \cdot t \tag{7}$$

The limits in position i.e. the allowed flight volume, is handled in a way such that going from inside to outside the volume is forbidden, but the opposite is allowed. Only flight within the volume or *towards* it is allowed.

B. Low level control

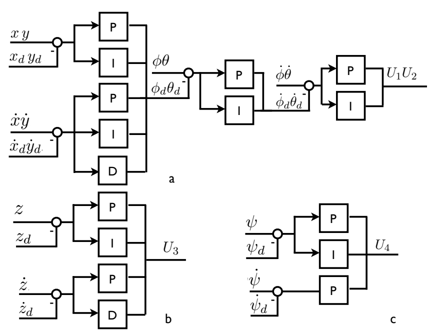
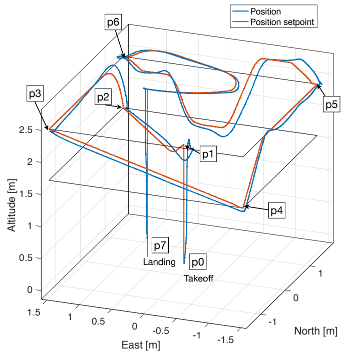


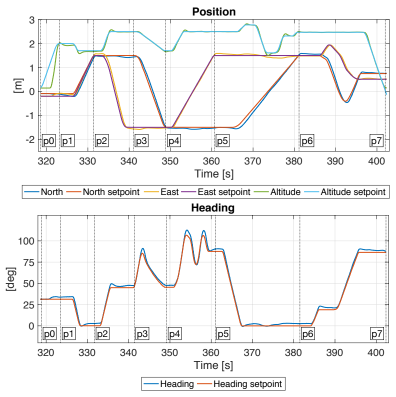
Fig. 6: PID controllers' structure: lateral and longitudinal (a), altitude (b) and heading (c).

This section describes the proposed control system structure. It is based on classical proportional-integral-derivative (PID) controllers organised in cascades. The desired pose, obtained using the process described in the previous sections, consists in the position with its derivatives $[\vec{p}_d, \vec{v}_d, \vec{a}_d]$. These constitute setpoints for the PID controllers with the configuration presented in Figure 6. The position controller uses PI loops for x, y as well as PID loops for velocities $\dot{x}, \dot{y}$. The attitude pitch and roll angles (θ, ϕ) are then controlled using PI loops on angles followed by PI loops on rates $(\dot{\theta}, \dot{\phi})$. The altitude control is achieved using PI loops on position and PD on vertical velocity. Heading uses PI configuration on the value of ψ and P on the rate $\dot{\psi}$.

Signals $U_{1...4}$ control longitudinal, lateral, vertical and heading control channels, respectively. They are mixed to produce the platform's motor speed controller inputs.



(a) Position and its setpoints. Grey lines at alt. 1.7m and 2.5m depict a flight pattern for sequence p2-p6.



(b) Position, heading and their setpoints.

Command/ segment # (endpoints)	**Horizontal** north, east, speed	**Vertical** altitude, speed	**Heading** heading, rate	**End condition**	**Flags**
0 (p0-p1)	**DNU**	**Pos** 2m, 0.5m/s	**DNU**	E_V	Takeoff
1 (p1-p2)	**Pos** 1.5m, 1.5m, 0.5m/s	**Pos** 1.7m, 0.5m/s	**Pos** 0deg, 20deg/s	E_H, E_V, E_{Hd}	-
2 (p2-p3)	**ExtPos** 1.5m, -1.5m, 0.5m/s	**Pos** 2.5m, 0.5m/s	**Pos** 45deg, 20deg/s	E_H, E_V, E_{Hd}	H_{VSM}
3 (p3-p4)	**Pos** -1.5m, -1.5m, 0.5m/s	**WithHor** 1.7	**ToPoint** 1.5m, 1.5m	E_H, E_V	H_{HdSM}
4 (p4-p5)	**Pos** -1.5m, 1.5m, 0.3m/s	**Pos** 2.5m, 0.3m/s	**ExtVel**	E_H, E_V	-
5 (p5-p6)	**Pos** 1.5m, 1.5m, 0.5m/s	**ExtVel**	**FlightDir** 0deg	E_H	-
6 (p6-p7)	**ExtVel**	**ExtVel**	**ExtVel**	E_V	Allow landing

(c) Configuration of flight commands.

Fig. 7: Experiment 1: flight through points p0 to p7.

It is important to note that the underlying control system can have a different configuration or use a completely different control approach. For example, a more complex structure could be required depending on a specific platform configuration. But as long as the control system is capable of producing the control signals, based on the desired pose derived as described in the previous section, it can be used with the proposed flight command concept.

C. *Takeoff and landing*

1) Takeoff: The takeoff manoeuvre is configured using a flight command but has a number of additional properties. If a vertical command has the *takeoff* flag set, first a check is performed whether the system is ready for a takeoff. It includes a check whether the platform is in the air.[1]If that is the case, the throttle is increased with a constant rate up to a point when the UAV almost lifts off of the ground. At that point a flight to a specified altitude with the specified speed is performed.

2) Landing: The landing manoeuvre flight command is configured using the *allow landing* vertical channel flag. This flag overrides the lower altitude limit of the flight volume, and enables the landing detection check which is used to finish the flight command. Typically the manoeuvre includes a vertical flight command to a negative altitude, but it can also be performed using ExtVel, RcVel, ExtPos commands. Touch-down is detected when the throttle command achieves a very low predefined level (e.g. 15% of the throttle stick) for a specified period of time (typically for 2 seconds).

IV. Examples and experimental validation

The VTOL platform used in the experiments presented in this section is a LinkQuad 4.2 (Figure 1) system developed in-house. Due to its compact design (80 cm tip-to-tip), the platform is suitable for both indoor and outdoor use. The LinkQuad is equipped with an in-house designed flight control board configured to use two ARM-Cortex microcontrollers running at 72MHz that implement the core flight functionalities. The inner and outer control loops are calculated at 500Hz and 50Hz, respectively. The other functionalities described in this paper are scheduled at a 10Hz rate.

The experimental flights presented in this paper have been performed in a motion capture equipped arena using ten T10 and six T40-S cameras from Vicon. The flight volume is approximately $10 \times 10 \times 5$ meters. The state provided by this system (position, velocities and heading angle) has been used instead of a state estimator based on the onboard GPS, IMU, compass and a pressure altimeter used for outdoor experimentation. The state is sent to the onboard system using a wireless RS-232 serial connection at 10-25 Hz to closely replicate outdoor operation. Other than replacing the state-estimation, the rest of the system operates in the same manner for both indoor and outdoor environments.

The remainder of this section provides examples of defining flight commands as well as results of experimental flights showing the use of the proposed system. First, an example mission from takeoff to landing, configured using 7 flight commands executed in sequence, is presented. This showcases the flexibility in achieving different flight behaviours using the proposed method. Second, results of example flights, where an active command is seamlessly switched to another one, are presented. This example showcases the freedom to switch flight commands at any time to achieve mission break-ins. The experiments are structured to showcase some of the flight manoeuvres outlined in the emergency rescue mission scenario described in Section I.

The following parameters of the setpoint generation module were used: $\pm$3.5m position (minimum -1.5m position in the east direction in Experiment 2), $\pm$1m velocity, $\pm$0.5m/s^2 acceleration for the horizontal channel; 1.0–3.5m altitude, -0.5m/s descent and 1.0m/s ascent speeds, $\pm$0.5m/s^2 acceleration for the vertical channel; $\pm$45deg/s yaw speed, $\pm$45deg/s^2 rate change for the heading channel.

1) Experiment 1: Command sequence: Figures 7a and 7b present position and heading setpoints, as well as actual position and heading, during a flight through 8 points. Details of the command configuration for all segments of the flight

[1]If the commanded throttle value is above an experimentally determined threshold value. It is obtained once and repeated if a considerable change takeoff weight occurs.

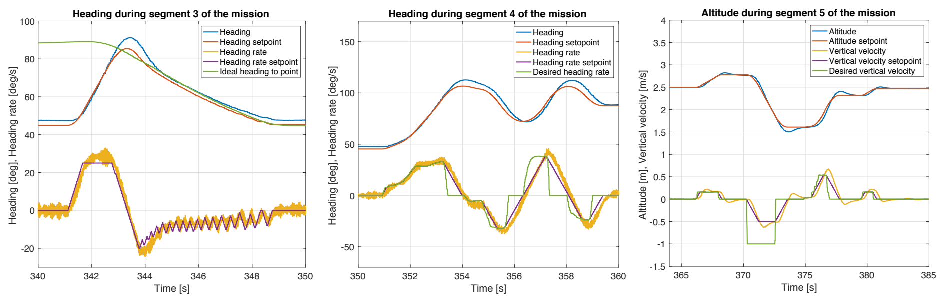


Fig. 8: Experiment 1: heading control during segments 3 and 4 (left and middle), altitude control during segment 5 (right).

are presented in Figure 7c.

The mission starts at an arbitrary position p0 with a takeoff manoeuvre configured using *Command 0*. Only the vertical channel is used with a flight to 2m altitude at 0.5m/s vertical velocity. The *takeoff* flag is set to true as described in Section III-C.1. The command ends when the vertical finish channel flag V_{end} becomes true.

Command 1 uses the horizontal channel to move the platform to position p2 (1.5m, 1.5m) at 0.5m/s speed. The vertical channel independently brings the UAV to an altitude of 1.7m at 0.5m/s vertical speed. The heading is set to a constant 0 degrees. The command finishes when all three control channel flags, H_{end}, V_{end}, Hd_{end}, become true.

Command 2 uses an externally provided position mode H_{ExtPos} (1.5m, -1.5m) and absolute altitude of 2.5m. Here, the vertical speed modifier V_{sm} is used so that the horizontal movement starts only when the desired altitude is close to be achieved. The heading is set to a constant 45deg throughout this segment of flight.

Command 3 is configured to achieve a straight line flight to position p4 (-1.5m, -1.5m, 1.7m) at 0.5m velocity. This is achieved through combining H_{Pos} with $V_{WithHor}$ modes. The heading is set throughout the flight to be pointing towards position p2. Figure 8a shows a detailed plot of the heading during this segment of the flight. The ideal heading to the point is computed based on the current and the target points. The jagged heading rate setpoint is a result of numerical inaccuracies, delays in the data being provided to the system, and different execution rates of the control functions. The heading rate in the plot is taken from a gyroscope. The command finishes when H_{end} and V_{end} flags become true.

Command 4 used between points p4 and p5 is similar to *Command 2*, but the heading is controlled externally by providing desired heading rates by the operator using a joystick. Figure 8b presents a detailed plot of the heading and rate during this segment of flight. It can be observed that the desired heading rate is limited by the maximum rate of change allowed in this flight.

Command 5 uses the horizontal channel to bring the platform to position p6. The vertical channel is commanded by the operator providing arbitrary desired vertical velocity. Figure 8c shows a detailed plots for this segment of flight. It can be observed that the desired vertical velocity is limited by the maximum allowed value during this flight. The heading channel uses $Hd_{FlightDir}$ mode which makes the heading towards point p6 during the flight.

Finally, in *Command 6*, the operator is given full velocity control over the platform. The UAV is commanded using a joystick towards an arbitrary landing position. The *allow landing* flag is used and the operator performs a landing manoeuvre by commanding negative vertical velocity.

Figure 7b presents position, altitude and heading setpoints, as well as the actual values logged onboard the UAV during the mission. The maximum absolute control error measured for the horizontal channel was: 28.5cm with root-mean-square (RMS) of 12.5cm. For the vertical channel (excluding takeoff) the values used were: 12.5cm and 3.6cm, respectively. Similarly, for the heading the values used were: 8.8deg and 2.8deg. Even though the PID loops of the control system were tuned manually, the performance is satisfactory. The maximum horizontal control error during the mission was approx. 35% of the platform's tip-to-tip diameter. Similarly, for the vertical channel the value is approx. 50% of the UAV's height.

The experiments are intended to demonstrate the versatility and flexibility of the flight command concept. By appropriately choosing and configuring the modes, it is possible to achieve richly configurable flight behaviours required for emergency assistance missions. *Takeoff* behaviour maps directly to flight Command 0. Flying to a waypoint can be performed in a number of ways: *Fly to a 3D waypoint, with a specific heading* (Command 1); *Fly to a 3D waypoint in a straight line, set heading towards a moving position* (Command 3); *Fly, and set heading, to a 2D waypoint, altitude controlled externally*, for example, relative to the ground (Command 5). Scanning an area (flying a *sequence of waypoints*) translates into several sequential commands (e.g. Command 3 repeated with the appropriate position parameters). Command 6 is an example of *direct operator interaction* (including a landing, in this case).

2) Experiment 2: Command switching: This experiment demonstrates the performance of the system when a command is executed while another one is in progress. This seamless switching of flight behaviours, at any time during execution, is essential in emergency assistance missions which are very dynamic in nature. This system property provides additional flexibility as the user does not have to deal with control mode switching or wait for the platform to be brought to a specific control state. Figure 9 presents three example flights.

First, a nominal flight from p0 to p1 using the H_{Pos} mode, with coordinates (-1.5m, 0m) and (1.5m, 0m), respectively,

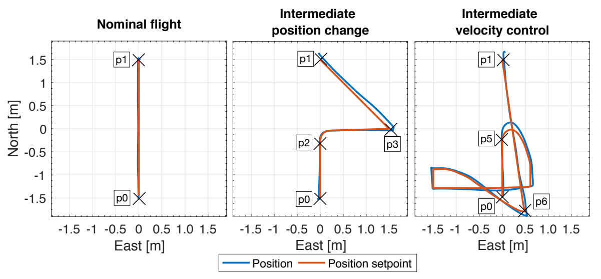


Fig. 9: Experiment 2: command switching. Position setpoints and the actual position during: nominal path (left), in-flight break-in with a flight to point p3 (middle), and in-flight break-in using external velocity command at point p5 (right).

is presented Figure 9a. Second flight (Fig. 9b) starts as the the first one, but at point p2, a new command is executed. It uses the H_{Pos} mode, to bring the UAV to point p3 with coordinates (0m, -1.5m) followed by a flight to the original destination p1. The third flight, presented in Figure 9c, again starts as the first one, but at position p3, a command which uses H_{ExtVel} mode is executed. At this point the operator directly commands desired velocities using a joystick. Note that the platform never goes beyond -1.5m position in the east direction, as this is the envelope limit set for this flight. When the operator decides, at point p6, the original flight to point p1 resumes.

This experiment demonstrates how a mission *break-in* is performed. This is possible due to the use of the setpoint generation module, described in Section III-A, between the flight commands and the underlying control system. It permits the execution of a command at arbitrary moments during a mission (e.g. commands at points p2 and p5 in the presented experiment), as well as resuming mission execution (as at points p3 or p6). This functionality allows for implementing *break-in* behaviours, *pausing* and *resuming* missions, or even completely changing a mission being executed, without finishing or even stopping the previous one. This kind of functionality is essential in emergency assistance type missions where quick adaptation to a dynamic change in conditions is a requirement. This capability, for example, allows the operator to suspend a scanning mission to closely investigate a salient point and seamlessly continue the flight afterwards.

V. Conclusion

This paper has presented a method for dynamic and contextual definition of flight behaviours for VTOL UAVs and their real-time re-parameterisation in the operational field. This is achieved by incorporating an additional layer of system abstraction between the elementary flight actions that flight behaviours are composed of and a continuous control kernel. The abstraction layer uses the concept of flight commands that parameterise three control channels. Through this abstraction layer, elementary actions and flight behaviours can be specified and parametrised flexibly in terms of sequences of flight commands controlling horizontal, vertical and heading channels. These flight commands in turn interface with the control kernel in a safe and robust manner through the use of a setpoint generation module. The proposed method has been fully implemented and tested in prototype. Examples have been presented as to how higher-level behaviours are translated into flight commands for a VTOL UAV to achieve a new level of flexibility that is required in the context of emergency assistance missions where humans are in the loop and dynamic interaction between human and platform is common. The system and ideas have also been experimentally validated. For future work, our interest lies in integrating task and path planners into the system where the output of these planners would be operators implemented as sequences of flight commands.

References

[1] S. hyun Lee, S. H. Kang, and Y. Kim, "Trajectory tracking control of quadrotor uav," in *Control, Automation and Systems (ICCAS), 2011 11th International Conference on*, Oct 2011, pp. 281–285.

[2] D. Cabecinhas, R. Naldi, L. Marconi, C. Silvestre, and R. Cunha, "Robust take-off for a quadrotor vehicle," *Robotics, IEEE Transactions on*, vol. 28, no. 3, pp. 734–742, June 2012.

[3] D. Cabecinhas, R. Cunha, and C. Silvestre, "A robust landing and sliding maneuver controller for a quadrotor vehicle on a sloped incline," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 523–528.

[4] M. Hehn and R. D'Andrea, "A frequency domain iterative learning algorithm for high-performance, periodic quadrocopter maneuvers," *Mechatronics*, vol. 24, no. 8, pp. 954–965, Dec. 2014.

[5] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. M. Humphrey, "Supporting wilderness search and rescue using a camera-equipped mini UAV," *Journal of Field Robotics*, vol. 25, no. 1-2, pp. 89–110, 2008.

[6] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 46–56, Sept 2012.

[7] E. Gat, R. P. Bonnasso, R. Murphy, and A. Press, "On three-layer architectures," in *Artificial Intelligence and Mobile Robots*. AAAI Press, 1997, pp. 195–210.

[8] J. D. Boskovic, R. Prasanth, and R. K. Mehra, "A multi-layer autonomous intelligent control architecture for unmanned aerial vehicles," *Journal of Aerospace Computing, Information, and Communication*, vol. 1, no. 12, pp. 605–628, 2004.

[9] P. Doherty, J. Kvarnström, M. Wzorek, P. Rudol, F. Heintz, and G. Conte, "HDRC3 - a distributed hybrid deliberative/reactive architecture for unmanned aircraft systems," in *Handbook of Unmanned Aerial Vehicles :*, 2014, pp. 849–952.

[10] F. Adolf and F. Andert, *Onboard mission management for a VTOL UAV using sequence and supervisory control*, 2010.

[11] P. Doherty, F. Heintz, and J. Kvarnstrm, "High-level mission specification and planning for collaborative unmanned aircraft systems using delegation," *Unmanned Systems*, vol. 01, no. 01, pp. 75–119, 2013. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S2301385013500052

[12] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics*, vol. 29, no. 2, pp. 315–378, 2012. [Online]. Available: http://dx.doi.org/10.1002/rob.20414

[13] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *Int. J. Rob. Res.*, vol. 29, no. 13, pp. 1608–1639, Nov. 2010. [Online]. Available: http://dx.doi.org/10.1177/0278364910371999

[14] O. Andersson, M. Wzorek, P. Rudol, and P. Doherty, "Model-predictive control with stochastic collision avoidance using bayesian policy optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[15] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, 2012.