

## 6.2 UAS Model and Control

The key feature of *Movement Automaton* is to interface the *UAS system* as the *discrete command chain*. Following topics are introduced in this section:

1. *Movement Automaton Applications* (sec. 6.2.1) - the listing of related work and similar approaches to ours.
2. *UAS Model* (sec. 6.2.2) - a simple plane model used in this work as the *controlled plant*.
3. *UAS Movement Automaton* (sec. 6.2.3) - movement automaton for *UAS Nonlinear Model* constructed from scratch.

### 6.2.1 Movement Automaton Applications

*Movement Automaton* is a basic interface approach for discretization of *trajectory evolution* or *control input* for any *continuous or discrete system model*.

Main function of *Movement Automaton* is for system given by equation  $\dot{state} = f(time, state, input)$  with initial state  $state_0$  to generate *reference trajectory*  $\hat{state}(t)$  or *control signal*  $input(t)$ .

Using *Movement Automaton* as *Control Proxy* will provide us with *discrete command chain* interface. This will reduce the *non-deterministic* element from *Evasive trajectory* generation, by reducing infinite maneuver set to finite *movement set*.

*Non-determinism* of *Avoidance Maneuver* has been discussed as an issue in following works:

1. Newton gradient method for evasive car maneuvers [1].
2. Non-holistic methods for trajectory generation [2].
3. Stochastic approach to elliptic trajectories generation [3].

*Examples of Movement Automaton Implementation as Control Element* can be mentioned as follows:

1. Control of traffic flow [4].
2. Complex air traffic collision situation resolution system [5, 6].
3. SAA/DAA capable avoidance system [7].

### 6.2.2 UAS Model

**Motivation:** Simplified rigid body kinematic model will be used. This model has decoupled roll, yaw and pitch angles. The focus is on *reach set approximation methods*; therefore the *UAS model* is simplified.

**State Vector** (eq. 6.1) defined as a positional state in euclidean position in right-hand euclidean space, where  $x, y, z$  can be abstracted as latitude, longitude, altitude.

$$state = [x, y, z, roll, pitch, yaw]^T \quad (6.1)$$

**Input Vector** (eq. 6.2) is defined as the linear velocity of UAS  $v$  and angular speed of rigid body  $\omega_{roll}, \omega_{pitch}, \omega_{yaw}$ .

$$input = [v, \omega_{roll}, \omega_{pitch}, \omega_{yaw}]^T \quad (6.2)$$

Velocity vector function (eq. 6.3) is defined through the standard rotation matrix and linear velocity  $v$ , oriented velocity  $[v_x, v_y, v_z]$  given by (eq. 6.4).

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} v \cos(pitch) \cos(yaw) \\ v \cos(pitch) \sin(yaw) \\ -v \sin(pitch) \end{bmatrix} \quad (6.3)$$

**UAS Nonlinear Model** (eq. 6.4) is given by *first order equations*:

$$\begin{aligned} \frac{dx}{dt} &= v \cos(pitch) \cos(yaw); & \frac{droll}{dt} &= \omega_{roll}; \\ \frac{dy}{dt} &= v \cos(pitch) \sin(yaw); & \frac{dpitch}{dt} &= \omega_{pitch}; \\ \frac{dz}{dt} &= -v \sin(pitch); & \frac{dyaw}{dt} &= \omega_{yaw}; \end{aligned} \quad (6.4)$$

**Discretization** for *fixed step*  $k$  we start with discretization of the model:

The *linear velocity* in text step is given:

$$v(k+1) = v(k) + \delta v(k) \quad (6.5)$$

The *roll, pitch, yaw* for next step are given

$$\begin{aligned} roll(k+1) &= roll(k) + \delta roll(k) \\ pitch(k+1) &= pitch(k) + \delta pitch(k) \\ yaw(k+1) &= yaw(k) + \delta yaw(k) \end{aligned} \quad (6.6)$$

The  $\delta v(k)$  is *velocity change*,  $\delta roll(k)$ ,  $\delta pitch(k)$ ,  $\delta yaw(k)$ , are *orientation changes* for current discrete step  $k$ . If the duration of *transition* is 0s (as. 1) then 3D trajectory evolution in discrete time is given as:

$$\begin{aligned}
x(k+1) &= x(k) + v(k+1) \cos(\text{pitch}(k+1)) \cos(\text{yaw}(k+1)) &= \delta x(k) \\
y(k+1) &= y(k) + v(k+1) \cos(\text{pitch}(k+1)) \sin(\text{yaw}(k+1)) &= \delta y(k) \\
z(k+1) &= z(k) - v(k+1) \sin(\text{pitch}(k+1)) &= \delta z(k) \\
\text{time}(k+1) &= \text{time}(k) + 1 &= \delta \text{time}(k)
\end{aligned} \tag{6.7}$$

The  $\delta x(k)$ ,  $\delta y(k)$ ,  $\delta z(k)$  are positional differences depending on *input vector* for given discrete time  $k$ :

$$\text{input}(k) = \begin{bmatrix} \delta x(k), \delta y(k), \delta z(k), \delta v(k), \\ \delta \text{roll}(k), \delta \text{pitch}(k), \delta \text{yaw}(k), \delta \text{time}(k) \end{bmatrix}^T \tag{6.8}$$

The *state vector* for discrete time is given:

$$\text{state}(k) = \begin{bmatrix} x(k), y(k), z(k), v(k), \\ \text{roll}(k), \text{pitch}(k), \text{yaw}(k), \text{time}(k) \end{bmatrix}^T \tag{6.9}$$

### 6.2.3 UAS Movement Automaton

**Motivation:** An *UAS Nonlinear Model* (eq. 6.4) can be modeled by *Movement Automaton* (def. ??).

**Movement Primitives** by (def. ??) are given as (eq. ??). Each movement primitive will last for fixed duration 1s.

**Assumption 1.** Let assume that transition time of roll, pitch, yaw, and the linear velocity is 0s.

Under the assumption (as. 1) the *movement transitions* (def. ??) have zero duration. Therefore movement primitives can be considered as movements.

*Note.* The assumption (as. 1) can be relaxed under the condition that *path tracking controller exists*.

**Movements** satisfying (def. ??), for the nonlinear model (eq. 6.4) reduced to *discrete model* (eq. 6.10), are given by *apply movements* function (eq. 6.5, 6.6, 6.7).

$$\text{state}(k+1) = \text{applyMovement}(\text{state}(k), \text{input}(k)) \tag{6.10}$$

**Movement Set** for the discrete model (eq. 6.10) is defined as a set of unitary movements on main axes (tab. 6.1) and diagonal axes (tab. 6.2).

The maneuvering capability of several commercial small fixed-wing UAS was abstracted together. The turning rate on horizontal/vertical is defined as  $15^\circ$ .

The deltas are posed in *UAS body-fixed coordinate frame* (ap. ??) for discrete time  $k$ .

| Parameter                 | Movement |       |      |      |       |
|---------------------------|----------|-------|------|------|-------|
|                           | Straight | Down  | Up   | Left | Right |
| $\delta x(k)[m]$          | 1.00     | 0.98  | 0.98 | 0.98 | 0.98  |
| $\delta y(k)[m]$          | 0        | 0     | 0    | 0.13 | -0.13 |
| $\delta z(k)[m]$          | 0        | -0.13 | 0.13 | 0    | 0     |
| $\delta roll(k)[^\circ]$  | 0        | 0     | 0    | 0    | 0     |
| $\delta pitch(k)[^\circ]$ | 0        | 15°   | -15° | 0    | 0     |
| $\delta yaw(k)[^\circ]$   | 0        | 0     | 0    | 15°  | -15°  |

Table 6.1: Input values for main axes movements.

| Parameter                 | Movement  |            |         |          |
|---------------------------|-----------|------------|---------|----------|
|                           | Down-Left | Down-Right | Up-Left | Up-Right |
| $\delta x(k)[m]$          | 0.76      | 0.76       | 0.76    | 0.76     |
| $\delta y(k)[m]$          | -0.13     | 0.13       | 0.13    | -0.13    |
| $\delta z(k)[m]$          | -0.13     | -0.13      | 0.13    | 0.13     |
| $\delta roll(k)[^\circ]$  | 0         | 0          | 0       | 0        |
| $\delta pitch(k)[^\circ]$ | -15°      | -15°       | 15°     | 15°      |
| $\delta yaw(k)[^\circ]$   | 15°       | -15°       | 15°     | -15°     |

Table 6.2: Input values for diagonal axes movements.

*Note.* The *movement set* in shortened form is given as:

$$MovementSet = \left\{ \begin{array}{l} Straight, Left, Right, Up, Down, \\ DownLeft, DownRight, UpLeft, UpRight \end{array} \right\} \quad (6.11)$$

The *implemented movement set example* (fig. 6.1) shows the movement used as basic building blocs of the trajectory for fixed-wing UAS:

1. *Initial position* (red plane) - the initial position, before any movement execution.
2. *Straight movement application* (blue plane) - the *neutral movement application* brings plane forward.
3. *Main axes movements* (cyan planes) - the application of movements from (tab. 6.1)  $\{Up, Down, Left, Right\}$ .
4. *Diagonal axes movements* (magenta planes) - the application of movements from (tab. 6.2)  $\{DownLeft, DownRight, UpLeft, UpRight\}$ .

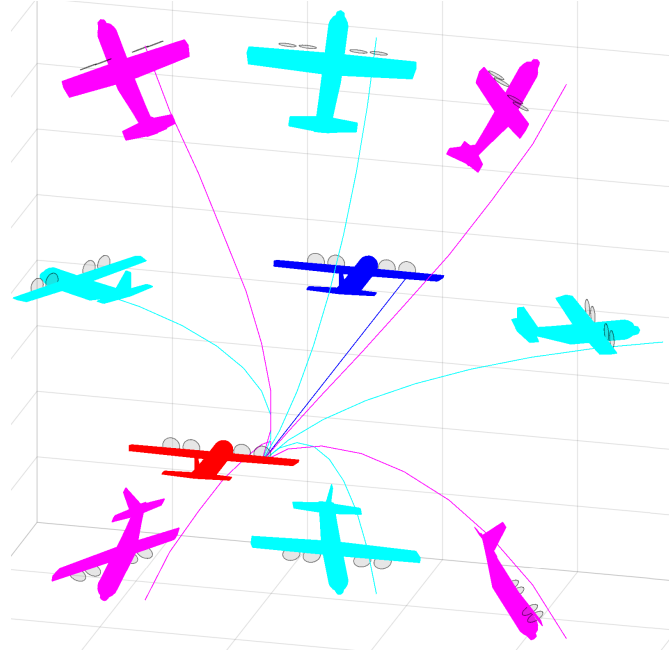


Figure 6.1: Implemented movement set example.

**Trajectory** by (def. ??) for initial time  $time = 0$ , initial state  $state(0)$  and *Movement Buffer* (from def. ??):

$$Buffer = \left\{ movement(j) : \begin{array}{l} movement(j) \in MovementSet(eq.6.11), \\ j \in 1 \dots n, n \in N^+ \end{array} \right\} \quad (6.12)$$

**Assumption 2.** *The buffer is always non-empty, ordered, finite list of movements.*

*Note.* The buffer has finite count  $n$  of movements stored. The buffer is the planning instrument used by higher level navigation/avoidance algorithm to control UAS (Control/Command interface) (fig. ??).

The discrete trajectory (eq. 6.13) is ordered set of states bounded to discrete time  $0 \dots n$ , where  $n$  is movement count of *Buffer*. Trajectory set has  $n + 1$  members defined like the following:

$$Trajectory(state(0), Buffer) = \left\{ \begin{array}{l} state(0) = state(0), \\ state(1) = applyMovement(state(0), movement(1)), \\ state(2) = applyMovement(state(1), movement(2)), \\ \vdots = \vdots \\ state(n-1) = applyMovement(state(n-2), movement(n-1)), \\ state(n) = applyMovement(state(n-1), movement(n)) \end{array} \right\} \quad (6.13)$$

The  $movement(k)$  vector is selected from movement tables (tab. 6.1, 6.2).

*Note.* Parameter movement( $\cdot$ ) (eq. 6.13) is a movement order index in buffer (eq. 6.12).

# Bibliography

- [1] Ondřej Šantin and Vladimír Havlena. Combined partial conjugate gradient and gradient projection solver for mpc. In *Control Applications (CCA), 2011 IEEE International Conference on*, pages 1270–1275. IEEE, 2011.
- [2] François G Pin and Hubert A Vasseur. Autonomous trajectory generation for mobile robots with non-holonomic and steering angle constraints. Technical report, Oak Ridge National Lab., 1990.
- [3] Ralph G Andrzejak, G Widman, K Lehnertz, C Rieke, P David, and CE Elger. The epileptic process as nonlinear deterministic dynamics in a stochastic environment: an evaluation on mesial temporal lobe epilepsy. *Epilepsy research*, 44(2-3):129–140, 2001.
- [4] Yoshiaki Kuwata, Justin Teo, Gaston Fiore, Sertac Karaman, Emilio Frazzoli, and Jonathan P How. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, 2009.
- [5] Emilio Frazzoli. *Robust hybrid control for autonomous vehicle motion planning*. PhD thesis, Massachusetts Institute of Technology, 2001.
- [6] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Trajectory tracking control design for autonomous helicopters using a backstepping algorithm. In *American Control Conference, 2000. Proceedings of the 2000*, volume 6, pages 4102–4107. IEEE, 2000.
- [7] Alojz Gomola, João Borges de Sousa, Fernando Lobo Pereira, and Pavel Klang. Obstacle avoidance framework based on reach sets. In *Iberian Robotics conference*, pages 768–779. Springer, 2017.