**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 a *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 *state*˙ = *f*(*time,state,input*) with initial state *state*0 to generate *reference trajectory state*ˆ (*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.

2

**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.

**Input Vector** (eq. 6.2) is defined as the linear velocity of UAS *v* and angular speed of

rigid body *ωroll,ωpitch,ωyaw*.

Velocity vector function (eq. 6.3) is defined through the standard rotation matrix and linear velocity *v*, oriented velocity [*vx*, *vy*, *vz*] given by (eq. 6.4).

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

|  |  |
| --- | --- |
| **Discretization** for *fixed step k* we start with discretization of the model:  The *linear velocity* in text step is given: |  |
| The *roll, pitch, yaw* for next step are given |
|  | (6.6) |

−

*yaw*(*k* + 1) = *yaw*(*k*) + *δyaw*(*k*)

The *δv*(*k*) is *velocity change*, *δroll*(*k*), *δpitch*(*k*), *δyaw*(*k*), are *orientation changes* for current discrete step *k*. If the duration of *transition* is 0*s* (as. 1) then 3D trajectory evolution in discrete time is given as:

The *δx*(*k*), *δy*(*k*), *δz*(*k*) are positional differences depending on *input vector* for given discrete time *k*:

(6.8)

The *state vector* for discrete time is given:

(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 1*s*.

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

0*s.*

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 the *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).

**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◦.

4

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

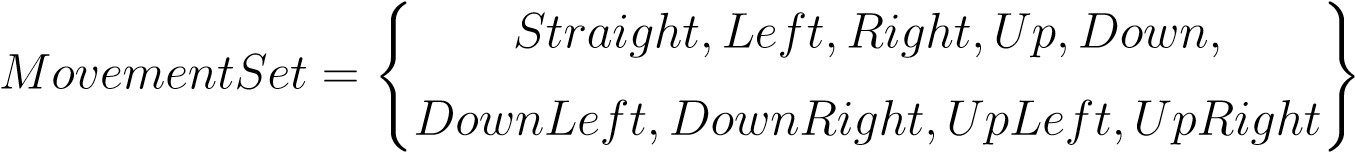
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Parameter |  | Movement | | |  |
| Straight | Down | Up | Left | Right |
| *δx*(*k*)[*m*] | 1.00 | 0.98 | 0.98 | 0.98 | 0.98 |
| *δy*(*k*)[*m*] | 0 | 0 | 0 | 0.13 | -0.13 |
| *δz*(*k*)[*m*] | 0 | -0.13 | 0.13 | 0 | 0 |
| *δroll*(*k*)[◦] | 0 | 0 | 0 | 0 | 0 |
| *δpitch*(*k*)[◦] | 0 | 15◦ | -15◦ | 0 | 0 |
| *δyaw*(*k*)[◦] | 0 | 0 | 0 | 15◦ | -15◦ |

Table 6.1: Input values for main axes movements.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter |  | Movement | |  |
| Down-Left | Down-Right | Up-Left | Up-Right |
| *δx*(*k*)[*m*] | 0.76 | 0.76 | 0.76 | 0.76 |
| *δy*(*k*)[*m*] | -0.13 | 0.13 | 0.13 | -0.13 |
| *δz*(*k*)[*m*] | -0.13 | -0.13 | 0.13 | 0.13 |
| *δroll*(*k*)[◦] | 0 | 0 | 0 | 0 |
| *δpitch*(*k*)[◦] | -15◦ | -15◦ | 15◦ | 15◦ |
| *δyaw*(*k*)[◦] | 15◦ | -15◦ | 15◦ | -15◦ |

Table 6.2: Input values for diagonal axes movements.

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

 (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*}.

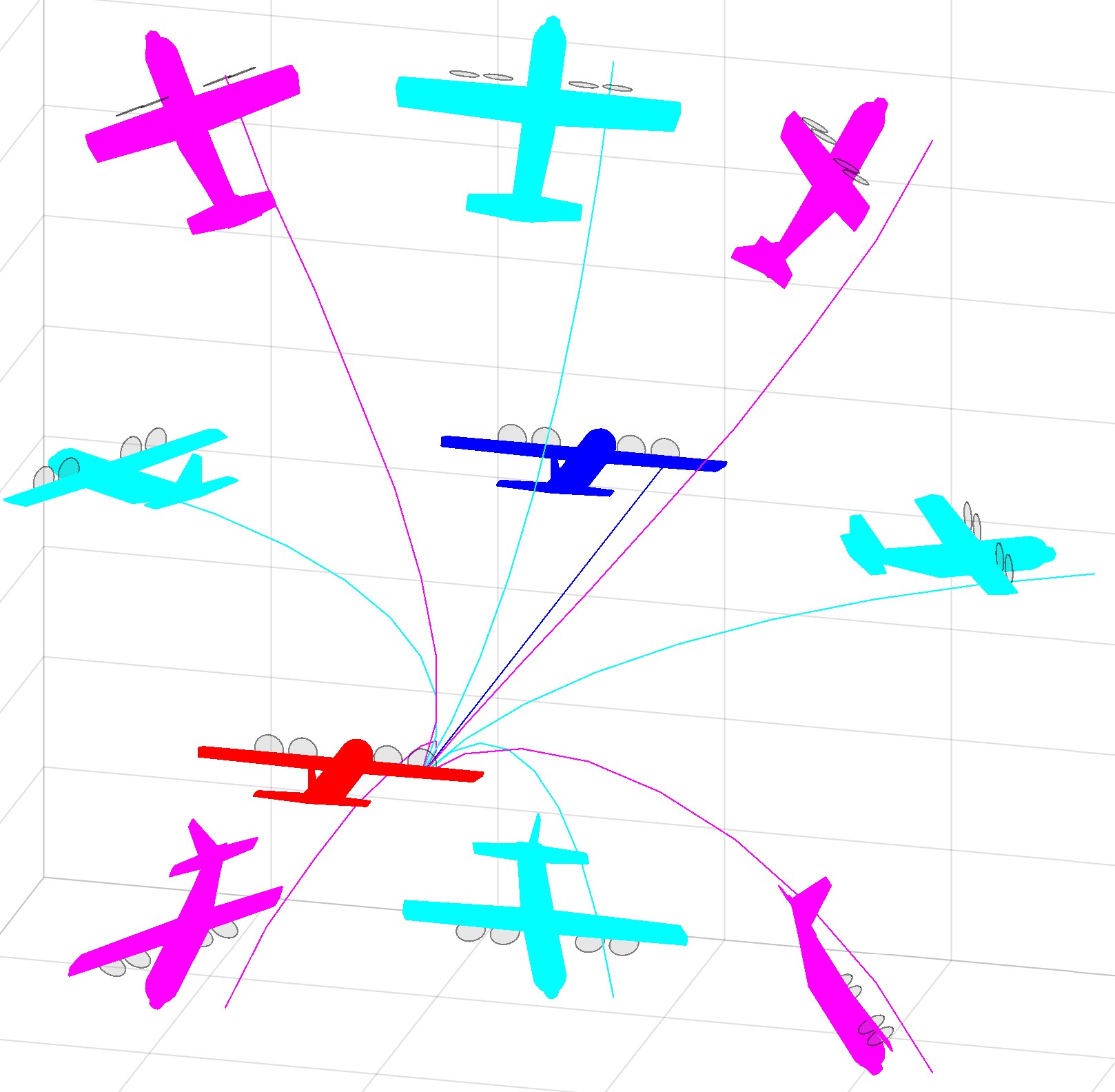
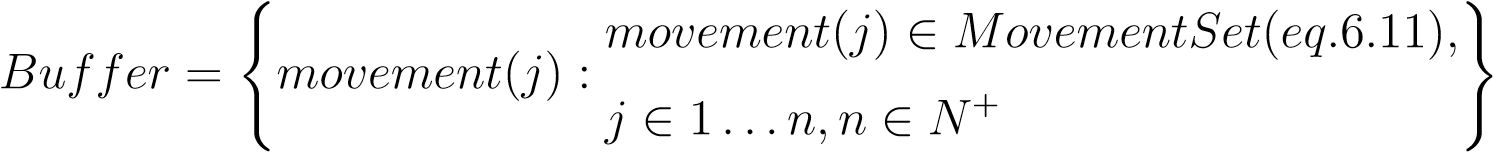


Figure 6.1: Implemented movement set example.

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

 (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*...n*, where *n* is movement count of *Buffer*. Trajectory set has *n* + 1 members defined like the following:

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

6

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

# Bibliography

1. Ondˇrej Santin and Vladimir Havlena. Combined partial conjugate gradient and gra-ˇ dient projection solver for mpc. In *Control Applications (CCA), 2011 IEEE International Conference on*, pages 1270–1275. IEEE, 2011.
2. Frangois 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, Jo˜ao 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.

7