6.2 UAS Model and Control

The key feature of *Movement Automaton* is to interface *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) - simple plane model used in this work as *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 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* have 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 have decoupled roll, yaw and pitch angles. The focus is on *reach set approximation methods*, therefore *UAS model* is simplified.

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

$$state = [x, y, z, roll, pitch, yaw]^T \tag{6.1}$$

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

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

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

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

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

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}time} &= v\cos(pitch)\cos(yaw); & \frac{\mathrm{d}roll}{\mathrm{d}time} &= \omega_{roll}; \\ \frac{\mathrm{d}y}{\mathrm{d}time} &= v\cos(pitch)\sin(yaw); & \frac{\mathrm{d}pitch}{\mathrm{d}time} &= \omega_{pitch}; \\ \frac{\mathrm{d}z}{\mathrm{d}time} &= -v\sin(pitch); & \frac{\mathrm{d}yaw}{\mathrm{d}time} &= \omega_{yaw}; \end{aligned} \tag{6.4}$$

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. **??**). To define primitives the *minimal time* is $1s$. The *maximal duration* is also $1s$.

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

Under the assumption (as. 1) the *movement transitions* (def. **??**) have 0 duration.

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

Movements (def. **??**) for *fixed step k* we start with discretization of the input variables. The *linear velocity* in text step is given:

$$v(k+1) = v(k) + \delta v(k) \tag{6.5}$$

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

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

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:

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

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

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

The *state vector* for discrete time is given:

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

The nonlinear model (eq. 6.4) is then reduced to *linear discrete model* (eq. 6.10) given by *apply movements* function (eq. 6.5, 6.6, 6.7).

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

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

$input(movement)$	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
$\delta roll(k)[°]$	0	0	0	0	0
$\delta pitch(k)[°]$	0	15°	-15°	0	0
$\delta yaw(k)[°]$	0	0	0	15°	-15°

Table 6.1: Input values for main axes movements.

$input(movement)$	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
$\delta roll(k)[°]$	0	0	0	0
$\delta pitch(k)[°]$	-15°	-15°	15°	15°
$\delta yaw(k)[°]$	15°	-15°	15°	-15°

Table 6.2: Input values for diagonal axes movements.

Note. Movement set in shorten form is given as

$$MovementSet = \left\{ \begin{matrix} Straight, Left, Right, Up, Down, \\ DownLeft, DownRight, UpLeft, UpRight \end{matrix} \right\} \tag{6.11}$$

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

$$Buffer \in MovementSet^*(eq.6.11), \quad |Buffer| \in \mathbb{N} \tag{6.12}$$

Trajectory (eq. 6.13) is then given as the time-series of discrete states:

$$Trajectory(state(0), Buffer) = \left\{ \begin{matrix} state(0) + \sum_{j=0}^{i-1} input(movement(j)) : \\ i \in \{1 \ldots |Buffer| + 1\}, \\ movement(\cdot) \in Buffer \end{matrix} \right\} \tag{6.13}$$

Trajectory (eq. 6.13) is ordered set of states bounded to discrete time $0 \ldots n$, where n is member count of *Buffer*. Trajectory set has $n + 1$ members:

$$Trajectory(state(0), Buffer) = \left\{ \begin{aligned} state(0) &= state(0) + \{\} \\ state(1) &= state(0) + input(movement(1)) \\ state(2) &= state(0) + input(movement(1)) + input(movement(2)) \\ \vdots &= \vdots \\ state(n) &= state(0) + input(movement(1)) + \cdots + input(movement(n)) \end{aligned} \right\} \tag{6.14}$$

State Projection (eq. 6.15) for the *Trajectory* (eq. 6.13) is given as follow:

$$StateProjection(Trajectory, time) = Trajectory.getMemberByIndex(time + 1) \tag{6.15}$$

Note. Movement Automaton for system (eq. 6.4) with given (as. 1) is established with all related properties (sec. **??**).

Bibliography

[1] Ondřej Šantin and Vladimir 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] 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, 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.