Chapter 3

Background Theory

Motivation: Cooperative and Non-Cooperative *Sense and Avoid* (SAA) systems are key enablers for the *Unmanned Aerial Systems* (UAS) to routinely access non-segregated airspace [1]. Both cooperative and non-cooperative SAA systems are being developed to address this integration requirement.

The *DAA capability* is defined as the automatic detection of possible conflicts by the UAS platform under consideration and performing avoidance maneuvers to prevent the identified collisions. An analysis of the available DAA candidate technologies and the associated sensors for both cooperative and non-cooperative SAA systems is presented in [2].

Non-cooperative *Collision Detection and Resolution* (CD&R) for UAS is considered as one of the major challenges that need to be addressed [3] for the insertion of UAVs in non-segregated air space. As a result, many non-cooperative sensors for the SAA system have been adopted. Light Detection and Ranging (LIDAR)is used for detecting, warning and avoiding obstacles for low-level flying [4].

An approach to the definition of encounter models and their applications to SAA strategies is presented in [5] for both cooperative and non-cooperative scenarios.

Since 2014, there is a visible strong political support for developing rules on drones, but regulations are harmonizing slowly. The European Aviation Safety Agency (EASA) has been tasked to develop a regulatory framework for drone operations and proposals for the regulation of "low-risk" UAS operations. In achieving this, EASA is working closely with the Joint Authorities for Regulation of Unmanned Systems (JARUS) [6].

Background Areas: Following Areas are introduced in this chapter:

1. *UAS System Model* (sec. 3.1) - continuous and discrete mathematical models.

2. *Reach Sets* (sec. 3.2) - introduction to representation and calculation methods.

3. *Hybrid Automaton* (sec. 3.3) - intuitive definition of the *hybrid automaton.*

4. *LiDAR* (sec. 3.4) - a summary of *LiDAR* technology and terminology introduction.

3.1 UAS System Model

This section strongly follows [7].

3.1.1 Continuous-time Systems

Consider a class of systems given by functions:

$$\begin{aligned} StateEvolution &: input(time) \rightarrow state(state_0, time) \\ input(time) &: [0, FinalTime] \rightarrow \mathbb{R}^p \\ input(time) &\in \mathbb{R}^p, state(time) \in \mathbb{R}^n \end{aligned} \tag{3.1}$$

Where $input(time)$ and $state(state_0, time)$ are a sets of continuous-time signals. These are often called continuous-time systems because they operate on continuous-time signals.

Frequently, such systems can be defined by differential equations that relate the input signal to the output signal.

A prototypical description of a controlled (there is a control input signal) continuous-time system is:

$$\begin{aligned} &\text{d/dt state}(time) = \\ &\qquad f(time, state(time), input(time)), input(time) \in Inputs(time) \end{aligned} \tag{3.2}$$

Where $f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^p \rightarrow \mathbb{R}^n$ satisfies the conditions for existence and uniqueness of the ordinary differential equation and u is our control [8].

3.1.2 Discrete-time Systems

Consider another class of systems given by functions

$$\begin{aligned} StateEvolution \,&: input(k) \rightarrow state(k), \\ k \in&\{0, t_s, 2.t_s, 3.t_s, \ldots i.t_s\}, i \in \mathbb{N}^+ \\ input(k) \in&\mathbb{R}^p, state(k) \in \mathbb{R}^n \end{aligned} \tag{3.3}$$

Where $input(k)$, $state(k)$ is a set of discrete-time signals. They can be represented by a function f like $f : \{0, t_s, 2.t_s, 3.t_s, \ldots i.t_s\} \rightarrow \mathbb{R}^n, i \in \mathbb{N}^+$ where t_s is sampling time and i is discrete step [9].

3.1.3 Adversarial Behavior in Continuous-time Systems

Consider a subclass of continuous time systems where are two sets of control signals $uas(time)$ and $adversary(time)$ which are accommodated in following system:

$$\begin{aligned} \text{d/dt state}(time) &= f(t, state(time), uas(time), adversary(time)), \\ uasControl(time) &\in UASInputs(time) \subset \mathbb{R}^u, \\ adversaryControl(time) &\in AdversaryInputs(time) \subset \mathbb{R}^v \end{aligned} \tag{3.4}$$

This system representation is often used in definition of problem of pursuit/evasion problem. Krasovskii developed a solution approach to this problem in [10]. A complex example of can be found in article [11].

3.2 Reach Sets

Informally, the *Reach Set* of a system described by a differential equation is the *set of all states that can be reached from an initial state within a given time interval.* Similar definitions applies to the systems with different representation and control, such as *hybrid automaton.*

3.2.1 Definitions

For following definitions consider *nonlinear UAS system* described in (sec. 3.1).

Definition 1 (Reach set starting at a given point)**.** *Suppose the initial position and time* $(state_0, time_0)$ *are given. The reach set* $ReachSet[\tau, time_0, state_0]$ *of* nonlinear system *at time* $\tau \geq time_0$*, starting at* $(state_0, time_0)$ *is given by:*

$$ReachSet[\tau, time_0, state_0] = \bigcup \{state(\tau) : input(s) \in Inputs(s), s \in (time_0, \tau]\} \tag{3.5}$$

Reach set starting at given set can be used to determine reach set in case of *hybrid system* input control switch and it is defined as follow:

Definition 2. *set starting at a given set] The reach set at time* $\tau > t_0$ *starting from set* $States_0$ *is defined as:*

$$ReachSet[\tau, time_0, States_0] = \bigcup \{ReachSet[\tau, time_0, state_0] : state_0 \in States_0\} \tag{3.6}$$

Reach set for adversarial behavior can be used to calculate possible escape routes from pursuer and it is defined as follow:

Definition 3 (Reach set under adversarial behavior)**.** *Consider now the case of adversarial behavior([10, 11]). where* $input(t)$ *is our control and* $adversary(t)$ *is adversary control which is independent of* $input(t)$*, let* $differentialControl(t) = input(t) - \sup_{state \in state(t)}$ $adversary(t)$*, which represents worst possible input change in given state and time, then* reach set for system *is represented as:*

$$ReachSet \begin{bmatrix} \tau, \\ time_0, \\ state_0 \end{bmatrix} = \bigcup \left\{ state(\tau) : \begin{matrix} differentialControl(s) \in \\ DifferentialControlSet(s) \end{matrix}, s \in (time_0, \tau] \right\} \tag{3.7}$$

Reach set under state constraints are usable to define state constrained systems in terms of dynamics and technical capabilities.

Definition 4 (Reach set under state constraints)**.** *Suppose the initial position and time* $(state_0, time_0)$ *and state constraints are given* $state(t) \in \mathbb{A} \subset \mathbb{R}^n, \dot{x}(t) \in \mathbb{B} \subset \mathbb{R}^n$*. The reach set* $ReachSet[\tau, time_0, \vec{state}_0]$ *of* nonlinear UAS system *at time* $\tau \geq time_0$*, starting at position and time* $(state_0, time_0)$ *is given by:*

$$ReachSet \begin{bmatrix} \tau, \\ time_0, \\ state_0 \end{bmatrix} = \bigcup \left\{ state(\tau) : \begin{matrix} \forall s \in (time_0, \tau], state(s) \in \mathbb{A}, \\ \dot{state}(s) \in \mathbb{B}, \\ \exists input(s) \in Inputs(s) \end{matrix} \right\} \tag{3.8}$$

3.2.2 Computation of Reach Sets

Several techniques for reachability analysis of systems have been proposed. They can be (roughly) classified into two kinds:

1. Purely symbolic methods based on:

   a. the existence of analytic solutions of the differential equations and

   b. the representation of the state space in a decidable theory of the real numbers.

2. Methods that combine

   a. numeric integration of the differential equations

   b. symbolic representations of approximations of state space typically using (unions of) polyhedra or ellipsoids.

These techniques provide the algorithmic foundations for the tools that are available for computer-aided verification of hybrid systems ([12], [13], [14]).

The set-valued Lebesgue integral provides a conceptual tool for the direct computation of the reach set. In what follows we describe techniques from dynamic optimization which are used to compute reach sets for dynamic systems.

The relation between dynamic optimization and reachability was first observed in [15]. A typical problem of optimal control can be formulated as follows:

$$\max \left(\begin{matrix} \int_{initialTime}^{finalTime} cost(time, state(time), contro(time))\mathrm{d}time + \dots \\ \dots + FinalCost(state(finalTime)) \end{matrix} \right) \tag{3.9}$$

For nonlinear system:

$$\dot{state}(t) = f(t, state(t), control(t)), control(t) \in ControlSet(t) \subset \mathbb{R}^p \tag{3.10}$$

Where *cost* is given as a cost function of time, state and input and $FinalCost$ represent cost functional.

There are two main techniques to solve this problem, the maximum principle, and dynamic programming.

The maximum principle gives necessary conditions of optimality. Dynamic programming may be used to derive sufficient conditions of optimality.

A good reference on the maximum principle is [16]. A less known reference with detailed geometric interpretations is [17]. A good reference on dynamic programming is given in [18].

3.3 Hybrid Automaton

First, the notion of *hybrid* automaton [12, 19, 20] needs to be introduced:

Definition 5. *Hybrid automaton (3.11) is given as structure:*

$$\begin{aligned} HybridAutomaton(&AutomatonStates, SystemState, VectorField,\\ &DiscreteTransition, ResetMap) \end{aligned} \tag{3.11}$$

The automaton States *is given as a set of discrete states, for every time time $\in$ Domain hybrid automaton stays in exactly one of the* states.

System State *is given in domain $x \in \mathbb{R}^n, n \in \mathbb{N}^+$, representing the trajectory evolution.*

emph Vector Field (3.12) is bounded to single AutomatonState and represents local System State evolution when given automaton State is Active.

$$VectorField : AutomatonStateState \times SystemState \rightarrow SystemState \tag{3.12}$$

DiscreteTransition *(eq. 3.13) indicates changes of states in the automaton; the changes are triggered by satisfying the specific condition given by Automaton State and System State.*

$$DiscreteTransition : AutomatonState \times SystemState \rightarrow AutomatonState \tag{3.13}$$

ResetMap *(eq. 3.14) defines changes of State to some default value, specific automaton State and System State triggers this change.*

$$ResetMap : State \times SystemState \rightarrow SystemState \tag{3.14}$$

Hybrid Automaton Example: An example of a *hybrid automaton* is given in (fig. 3.1). The automaton is used to control *UAS system* to perform simple level up (increase altitude) maneuver [21]. The automaton has three discrete states representing *hover*, *transition*, and, *level* portion of the maneuver.

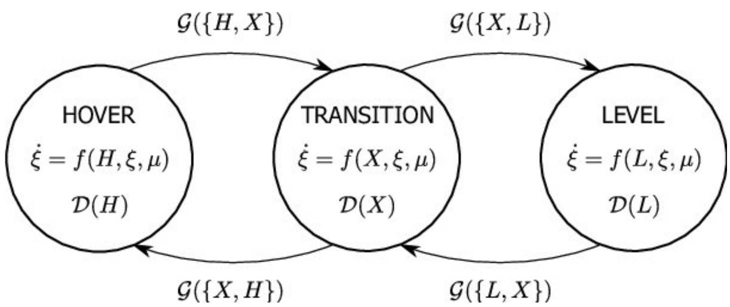


Figure 3.1: Example: The hybrid automaton example for UAS maneuver [21].

3.4 LiDAR

LiDAR(Light Detection And Ranging) is an active form of remote sensing: information is obtained from a signal which is sent from a transmitter and reflected by a target and detected by a receiver back at the source. Following types of information can be obtained:

1. *Range to target* - topographic LiDAR or laser altimeter.
2. *Chemical properties of target* - differential absorption LiDAR.
3. *The velocity of target* - Doppler LiDAR.

Chemical properties of the target are out of scope. The velocity of target seems as interesting property to investigate, but this type of LiDAR is usually used for meteorological measurements of wind currents [22]. Extended research in LiDAR as obstacle detection sensor has been executed by research group around Sabatini [4] and Ramasy [23].

A LiDAR output is represented as point cloud it is described by the definition:

Definition 6 (Scanned point and Point-cloud)**.** *Consider viewpoint as the origin of* $\mathbb{R}^3$ *space, Let* $point \in PolarCoordinates$ *be defined as:*

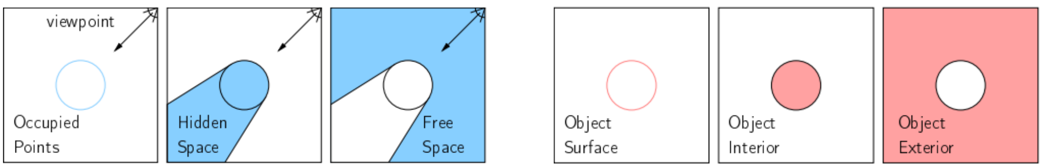
$$point = [distance, horizontal^\circ, vertical^\circ, time]^T \tag{3.15}$$

Where $horizontal^\circ$ *is the horizontal angle from the origin,* $vertical^\circ$ *is a vertical angle to the origin, and,* $time$ *is a time of retrieval.*

Point-cloud is set of points scanned in small enough time-frame, based on processing raw point data it can have following representations:

1. Local point-cloud - *position of the sensor is used as the origin of space and points can be represented in orthogonal or planar representation.*
2. Global point-cloud -*global position of the sensor is used as a reference to calculate the global position of points.*

Point-cloud is usually addressed as *raw point-cloud* in case if it is represented in Local planar coordinates. Other forms of point cloud require further processing, and they are not feasible for real-time obstacle detection and avoidance [24].



(a) Space type definitions (b) Object properties definitions

Figure 3.2: Six space classifications [25].

Because of real-time obstacle avoidance, it is necessary to introduce the following terminology:

1. *Occupied points* - points which have been detected by LiDAR (also addressed as visible points).

2. *Hidden space* - space which is hidden behind occupied points, from the viewpoint it is uncertain what is in that space.

3. *Free space* - space which is visible from viewpoint and it is not occupied by known objects.

4. *Object surface* - detected and undetected object surface

5. *Object interior* - occupied space by the object.

6. *Object exterior* - free space around known objects.

The existing method for space segregation [25] leads to the following definition:

Definition 7 (Accessible space)**.** *Consider known space as space explored by sensor (it can have different viewpoint along previous 3D trajectory). The intersection between object exterior (Exterior) and free space $Free$ gives us Accessible space (Accessible).*

$$Accessible = Exterior(object) \cap Free(object) \tag{3.16}$$

Accessible space S_A (def. 7) is our bordering limitation for reachable space of system $ReachSet[\tau, time_0, state_0]$ (def. 1.).

Bibliography

[1] Thomas P Spriesterbach, Kelly A Bruns, Lauren I Baron, and Jason E Sohlke. Unmanned aircraft system airspace integration in the national airspace using a ground-based sense and avoid system. *Johns Hopkins APL Technical Digest*, 32(3):572–583, 2013.

[2] Adrian Muraru. A critical analysis of sense and avoid technologies for modern uavs. In *Mechanical, Industrial, and Manufacturing Engineering–Proceedings of 2011 International Conference on Mechanical, Industrial, and Manufacturing Engineering (MIME 2011)*, 2011.

[3] John Lai, Jason J Ford, Luis Mejias, Peter O'Shea, and Rod Walker. See and avoid using onboard computer vision. *Sense and Avoid in UAS Research and Applications, Plamen Angelov (ed.), John Wiley and Sons, West Sussex, UK*, 2012.

[4] Roberto Sabatini, Alessandro Gardi, and Mark A Richardson. Lidar obstacle warning and avoidance system for unmanned aircraft. *International Journal of Mechanical, Aerospace, Industrial and Mechatronics Engineering*, 8(4):702–713, 2014.

[5] Mykel J Kochenderfer, Leo P Espindle, J Daniel Griffith, and James K Kuchar. Encounter modeling for sense and avoid development. In *2008 Integrated Communications, Navigation and Surveillance Conference*, pages 1–10. IEEE, 2008.

[6] JARUS regulations. `http://jarus-rpas.org/regulations`. Accessed: 2018-10-28.

[7] Edward A Lee. *Structure and interpretation of signals and systems*. Lee & Seshia, 2011.

[8] John Charles Butcher. *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*. Wiley-Interscience, 1987.

[9] Lawrence F Shampine and Mark W Reichelt. The matlab ode suite. *SIAM journal on scientific computing*, 18(1):1–22, 1997.

[10] Nikolai Nikolaevich Krasovskij, Andrei Izmailovich Subbotin, and Samuel Kotz. *Game-theoretical control problems*. Springer-Verlag New York, Inc., 1987.

[11] NN Krasovskii and AI Subbotin. Game-theoretical control problems. translated from the russian by samuel kotz, 1988.

[12] Conrado Daws, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. The tool kronos. In *Hybrid Systems III*, pages 208–219. Springer, 1996.

[13] Thomas A Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. *Information and computation*, 111(2):193–244, 1994.

[14] Thomas A Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. Hytech: the next generation. In *Real-Time Systems Symposium, 1995. Proceedings., 16th IEEE*, pages 56–65. IEEE, 1995.

[15] G Leitmann. Optimality and reachability via feedback controls. *Dynamic systems and mycrophysics*, pages 119–141, 1982.

[16] LS Pontryagin, VG Boltyanskii, and RV Gamkrelidze. Ef mischenko the mathematical theory of optimal processes (english translation by k. n. trirogoff). *Interscience, New York*, 1962.

[17] Igor Vladimirovich Girsanov. *Lectures on mathematical theory of extremum problems*, volume 67. Springer Science & Business Media, 2012.

[18] Martino Bardi and Italo Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Springer Science & Business Media, 2008.

[19] Mircea Lazar. Model predictive control of hybrid systems: Stability and robustness, 2006.

[20] Francesco Borrelli, Alberto Bemporad, Michael Fodor, and Davor Hrovat. An mpc/hybrid system approach to traction control. *IEEE Transactions on Control Systems Technology*, 14(3):541–552, 2006.

[21] Pedro Casau, David Cabecinhas, and Carlos Silvestre. Autonomous transition flight for a vertical take-off and landing aircraft. pages 3974–3979, 12 2011.

[22] S Martin, J Bange, and F Beyrich. Meteorological profiling of the lower troposphere using the research uav" m 2 av carolo". *Atmospheric Measurement Techniques*, 4(4):705–716, 2011.

[23] Subramanian Ramasamy, Roberto Sabatini, Alessandro Gardi, and Jing Liu. Lidar obstacle warning and avoidance system for unmanned aerial vehicle sense-and-avoid. *Aerospace Science and Technology*, 55:344–358, 2016.

[24] Qi Chen. Airborne lidar data processing and information extraction. *Photogrammetric engineering and remote sensing*, 73(2):109, 2007.

[25] Theodore C Yapo, Charles V Stewart, and Richard J Radke. A probabilistic representation of lidar range data for efficient 3d object detection. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–8. IEEE, 2008.