Beschreibung

2D Flugzeugmodell in Längenbewegung
mit Low-Level Eingaben

CreamyFoam

6. August 2021

Inhaltsverzeichnis

Zusammenfassung

CreamyFoam hat selber ein Flugzeug-Simulationsmodell in 2D in Vertikal-Ebene erstellt und versucht, mit Reinforcement Learning (RL) zu tranieren. Die Eingaben von Modell sind Low-Level, m. a. W. die Stellgrößen, die direkt das Flugzeug steuern, wie Höhenruder oder Querruder. Hier habe ich als Eingabe Höhenruder-Ausschlag(HR), Klappen-Ausschalg(Hochauftriebshilfe, KP), und Schubhebelstellung(SH) ausgewählt. Für Agent haben wir DDPG Agent benutzet. Ich habe versucht, den Agent tranieren zu lassen. Weil die Eingaben sehr Low-Level sind, sodass das Modell am Anfang lernen muss, wie es fliegt, hat das Training leider nicht geklappt. Aber es ist ein Interessanter Vergleich mit dem Maja-Modell, bei dem die Eingaben die Bahnrichtung (χ, γ) sind.

1 Erklärung des Flugzeugmodell

1.1 Bewegungsgleichung

Die für das Modell angewandte Gleichungen sind aus Manuskript von Flugmechanik in Seite 49. Die entsprechende Bedeutung von Variablen werden im Skript gefunden. Hier werden die wichtigste Gleichungen aufgestellt.

$$
\begin{aligned}
\dot{V_A} &= -\frac{1}{m}W + \frac{1}{m}\cos(\alpha + i_F)F_{res} - g \cdot \sin\gamma \\
\dot{\gamma} \cdot \dot{V_A} &= -\frac{1}{m}A + \frac{1}{m}\sin(\alpha + i_F)F_{res} - g \cdot \cos\gamma \\
\dot{\alpha} &= q - \dot{\gamma} \\
\dot{q} &= -\frac{1}{I_y}M
\end{aligned} \tag{1}
$$

Beim Aufbau von Modell, ist i_F vernachlässigt.
Um Auftrieb(A), Widerstand(W), Moment(M) und Schub(F_{res})zu rechnen, sind die Parameters, wie Geometrie, benötigt. Ich habe Cessna 172 R als das Referenz-Flugzeug ausgewählt. Die verwendete Gleichungen sind im Code "ClassFZ.m" zu finden.

1.2 Simulation

Die benötigte Parameters für Simulation werden die Position (X (horizontale Distanz), Y (Höhe, nach oben[1])) und Bahnneigungswinkel γ mit Differenzverfahren gerechnet.

$$
\frac{\partial}{\partial t}\begin{pmatrix} X \\ Y \\ \gamma \end{pmatrix} = \begin{pmatrix} \cos\gamma \\ \sin\gamma \\ 0 \end{pmatrix} \cdot V_A + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot \dot{\gamma} \tag{2}
$$

Diese oben geschriebene Gleichung werden mit einem Time-Step(Ts) wie folgende umgeschrieben.

$$
\begin{pmatrix} X(n) \\ Y(n) \\ \gamma(n) \end{pmatrix} = \begin{pmatrix} X(n-1) \\ Y(n-1) \\ \gamma(n-1) \end{pmatrix} \begin{pmatrix} \cos\gamma(n) \\ \sin\gamma(n) \\ 0 \end{pmatrix} \cdot V_A(n) \cdot Ts + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot \dot{\gamma}(n) \cdot Ts \tag{3}
$$

n bezeichnet den n-ten Zeitschritt. Da $\gamma(n)$ in Cosinus und Sinus Funktion bei n-ten Zeitschritt noch nicht verfügbar ist, wird es mit $\gamma(n-1)$ ersetzt und folgt daraus Quasi-Diskretisierung.

$$
\begin{pmatrix} X(n) \\ Y(n) \\ \gamma(n) \end{pmatrix} = \begin{pmatrix} X(n-1) \\ Y(n-1) \\ \gamma(n-1) \end{pmatrix} \begin{pmatrix} \cos\gamma(n-1) \\ \sin\gamma(n-1) \\ 0 \end{pmatrix} \cdot V_A(n) \cdot Ts + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot \dot{\gamma}(n) \cdot Ts \tag{4}
$$

Somit werden die alle Parameters gerechnet.
In der Simulation werden auch ein Zielpunkt und ein Hindernis dargestellt. Diese werden für die Aufgabe des Trainings verwendet.

2 Erklärung des Trainings

Für den Agent ist DDPG Agent ausgewählt, da die Eingabe und Ausgabe kontinuierlich sind. Die Einstellungen von Agent und Neural Network sind "default"von Matlab.
Die Aufgabe des Trainings ist, den Zielpunkt zu erreichen, ohne dass das Flugzeug stürzt oder ins Hindernis zu stoßen.
Als Reward werden folgende Gleichung benutzt.

$$
Reward = CloserBonus + TimeBonus + XNavigation + InputPenalty + TerminationPenalty \tag{5}
$$

[1] um die Visualisierung auf Matlab zu erleichtern, ist die Höhe mit Y bezeichnet und die Richtung nach oben festgesetzt.

CloserBonus vergibt Belohung abhängig von der Distanz zum Zielpunkt. Um das Flugzeug beim Lernen von Flug zu fördern, ist TimeBonus eingesetzt, das mehr Belohnung bei zeitlich längerem Flug gibt. XNavigation führt dazu, dass Flugzeug mehr in X Richtung fliegt. Weil je weniger man das Flugzeug steuert, desto besser es ist, gibt es inputPenalty, die aus der Unterschied zwischen derzeitigen und letzten Eingaben gerechnet werden. Wenn das Flugzeug stürzt, oder in das Hindernis stoßt, werden der Versuch abgebrochen und TerminationPenalty wird aufgerufen.

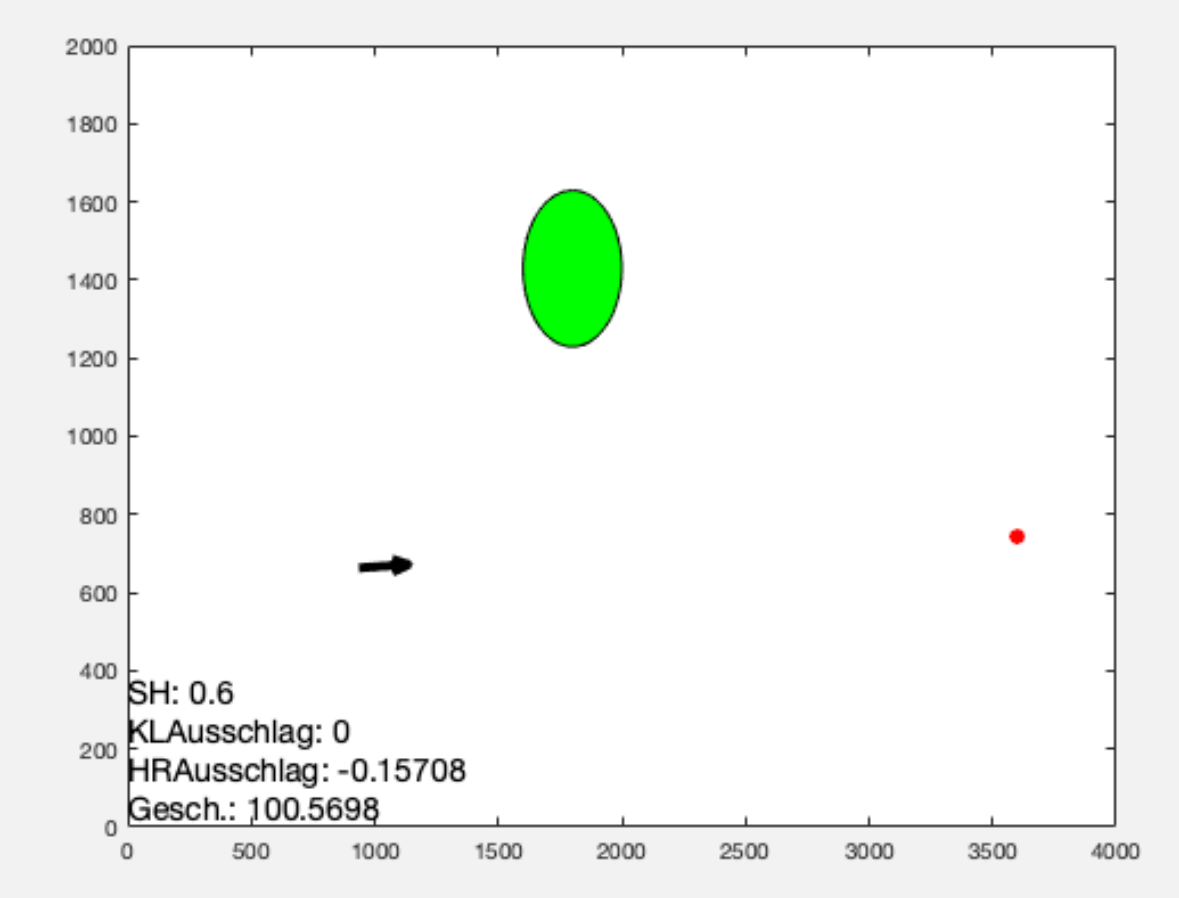

Abbildung 1: Simulation. Der grüne Kreis zeigt das Hindernis und der rote den Zielpunkt, deren Positionen sich bei jeder Episode beliebig ändern. Der Pfeil ist das Flugzeug, dessen Länge seine Geschwindigkeit und dessen Richtung seine Ausrichtung jeweils zeigt. Startposition des Flugzeug wird auch bei jeder Episode randomiert.

3 Ergebnis

Beim Training wurden 1000 Episoden durchgeführt.

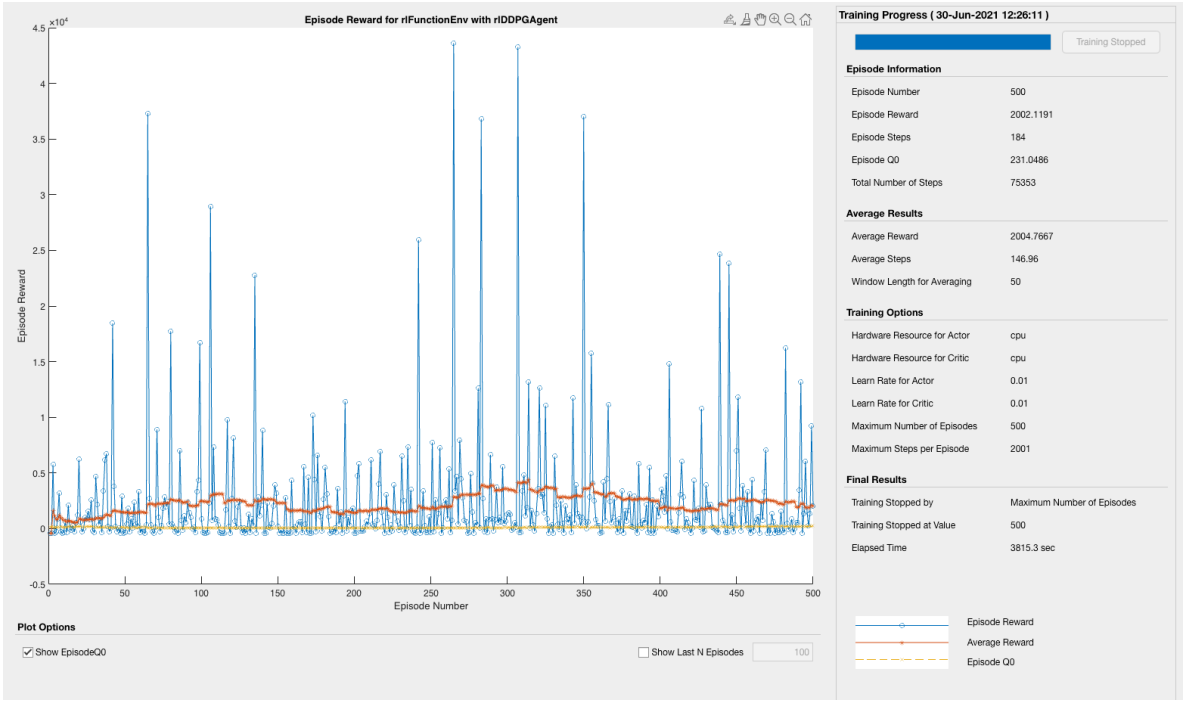

Abbildung 2: Verlauf des Trainings

Manchmal fliegt das Flugzeug lang und hat die X-Koordinaten von Zielpunkt erreicht. Aber oft ist das Flugzeug sehr schnell gestürzt, indem es akut nach oben oder nach unten fliegt.
Was man daraus sagen kann ist, dass es sehr lange Zeit dauern würde, wenn das Flugzeug mit Low-Level Eingabe das Fliegen lernt. Um diese Zeit zu verkürzen, sollen die Eingabe High-Level sein, wie Richtung, was wir mit Maja gemacht haben.

4 Abschluss

Durch dieses Training ist es klar geworden, dass Low-Level Eingabe sich nicht für RL geeignet ist, da Zeitaufwand sehr groß ist, das Fliegen am Anfang zu lernen.