

スパイキングニューラルネットワークを用いた ドローン飛行制御に関する研究調査

A Survey on Spiking Neural Network for Drone Flight Control

指導教員：福田盛介 教授

修士課程 1 年 37-256564 平田涼馬

Abstract

AI-based flight control for drones has gained significant attention for applications in unknown and dynamic environments. Particularly, in environments where communication delays are significant, such as Mars, there is a need for AI models that can operate onboard, and the application of Spiking Neural Networks (SNNs) is anticipated. This survey analyzes recent research related to SNN-based drone flight control, focusing on (1) methods using reinforcement learning, (2) methods using imitation learning, and (3) methods combined with event-based vision sensors. Finally, we summarize and compare the key results of each approach and discuss their future prospects to the flight control of Mars exploration drones.

1 序論

ドローンは地形や障害物の影響を受けにくい高い移動能力や導入・運用コストの低さなどの特徴から農業や建設、物流、惑星探査など幅広い用途への応用が進んでいる。これらの応用において、高い自律性を持つドローンの開発が求められている。ドローンの飛行制御には、従来の制御理論に基づく PID 制御やモデル予測制御などが用いられることが一般的であったが、これらの手法には未知の環境や動的な環境への適応性、複雑なタスクの遂行能力に限界がある。これらの課題に対処するため、近年ではニューラルネットワークを用いた制御手法に注目が集まっている [1]。ニューラルネットワークを用いたドローン制御ネットワークの推論は、通信を用いて高性能なコンピューターにデータを送信して推論を行う方法と、ドローンに搭載したエッジデバイスで行う方法に大別される。通信を用いる場合、高い計算能力を利用して高度な行動計画を立てたり、高い精度での物体認識などを活用した制御が可能である一方で、通信遅延の影響からリアルタイム性に欠けるという課題がある。一方、エッジデバイスを用いる場合、リアルタイムでの制御が可能であるが、電力の使用が飛行時間の短縮につながってしまう点や、計算能力の不足からモデルの軽量化を行った結果、推論精度が低下するという課題がある [2]。これらの課題に対処するため、省電力かつ高効率な計算が可能なスパイキングニューラルネットワーク (SNN) を用いたドローンの飛行制御手法が注目されている。

1.1 スパイキングニューラルネットワーク

スパイキングニューラルネットワーク (SNN) とは、生物の脳が 0,1 のスパイクを用いて情報伝達することを模倣した AI モデルである。このスパイクを処理するニューロンのモデルとして使用されているのが LIF モデルである。中でも Current based-LIF モデル (Cuba-LIF) は、シナプス電流と膜電位の 2 つの状態の時間変化を考慮したモデルであり、以下の式で表される。

$$\begin{aligned} v_i(t+1) &= \tau_i^{mem} v_i(t) + i_i(t), \\ i_i(t+1) &= \tau_i^{syn} i_i(t) + \sum_j W_{ij} s_j(t) \end{aligned} \quad (1)$$

ここで、 $v(t)$ は膜電位、 $i_i(t)$ は入力電流、 τ_{mem} は膜時定数、 τ_{syn} はシナプス時定数を表す。スパイクが入力されると、膜電位が上昇し、入力が無い場合は静止膜電位に向かって減衰する。連続でスパイクが入力されることで、膜電位が閾値を超えるとニューロンが発火し、出力スパイクを生成する。この Cuba-LIF モデルは、Intel のニューロモルフフィックチップである Loihi のデフォルトの LIF モデルであり、広く使われている [3]。SNN の特徴として、入出力情報が 0,1 のスパイクで表現されることや、スパイクの入出力がある場合のみニューロンが活動するため、専用のニューロモルフフィックチップ上に実装した場合、計算量が少なく、省電力、リアルタイム処理が可能であることが挙げられる。一方で、SNN は入出力で扱うスパイクが微分不可能である

ため、ANN で一般的に用いられる誤差逆伝搬法を直接適用できないため、学習が困難であるという課題がある。現状、SNN の学習手法としては、ANN で学習したパラメータを SNN に変換する手法や、代理勾配を用いて誤差逆伝搬法を適用する手法などが用いられているが、ANN と比較して学習効率が低いという課題がある [4]。また、ニューロモルフィックチップの性能も発展途上であり、ANN と比較してチップの計算能力が劣る場合が多い。

1.2 火星探査への展望

火星探査において、ドローンは地形の把握や科学観測、通信中継など多様な役割を担うことが期待されている。NASA の火星ドローン「Ingenuity」は、火星大気中での飛行実証を成功させ、その後も複数の飛行ミッションを遂行している [5]。将来的な火星探査ミッションでは、ドローンがより高度な自律飛行能力を持つことが求められており、未知の地形や動的な環境に適応できる制御手法の開発が必要である。この制御手法として、SNN を用いたドローン制御は、省電力かつリアルタイム処理が可能であるため、火星探査ドローンへの応用が期待されている [6]。



図 1: Ingenuity Mars Helicopter [5]

1.3 本調査の目的

本調査では、ドローンの飛行制御において、SNN のネットワークを姿勢安定化や移動などの低レベル制御に用いた 2 つの研究と、動的な環境での経路計画を含めたナビゲーションに SNN を用いた 1 つの研究について紹介する。1 本目の研究では、IMU センサーを用いて姿勢推定と制御を行うことを提案した Neuromorphic attitude estimation and control [7] を紹介する。2 本目の研究では、入力にイベントカメラを使用し、自己教師あり学習と進化アルゴリズムを組み合わせた複合学習により視覚ベースの自己運動制御を行うことを提案した Fully neuromorphic vision and control for autonomous drone flight [8] を紹介する。3 本目の研究では、強化学習アルゴリズムである PPO により SNN を学習させてドローンの高機動ナビゲーションを行ふことを提案した Bio-Inspired Drone Control A Reinforcement Learning-Trained Spiking Neural Networks for Agile Navigation in Dynamic Environment [9] を紹介する。最後に、各手法の分析から将来の火星探査ミッション

におけるドローン飛行制御への応用可能性について議論を行う。

2 SNN によるドローンの低レベル制御

2.1 IMU と模倣学習による姿勢推定・制御

2.1.1 概要

この研究では、ドローンの自律制御を単一のニューロモルフィックチップのみで実現することを目的として、IMU の入力からモーターコマンドを出力する SNN ベースの制御システムを提案している。本手法では、学習時にネットワークを姿勢推定と制御の 2 つのサブネットワークに分割し、それぞれを教師あり学習、模倣学習で訓練すること、SNN を用いることにより生じるセンサーバイアスやフィードフォワード遅延に対象するための工夫を行っている。また、提案手法は Crazyflie 上に実装され、評価が行われた。

2.1.2 ネットワーク構成

本手法では、ネットワークへの入力として IMU を用いている。IMU は加速度センサーとジャイロスコープから構成され、ドローンの 3 自由度の加速度と角速度を提供する。ネットワーク全体の構成は図 2 に示す通りである。2 つに分割されたサブネットワークの前段では、IMU の 6 つの入力から、ドローンの姿勢の推定値が算出された。次に、ネットワークの後段では姿勢の推定値と目標の姿勢が入力され、モーターのトルクコマンドが出力された。この

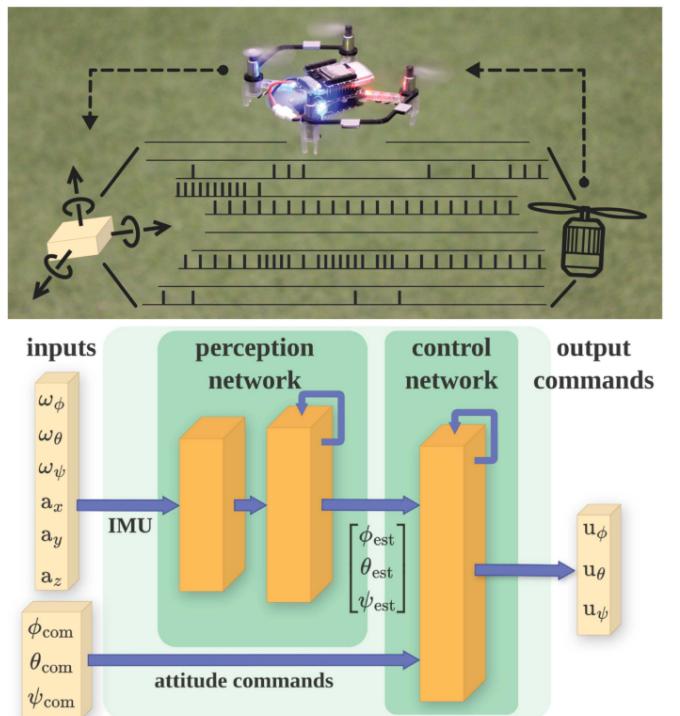


図 2: SNN network architecture [7]

ネットワークに用いられたニューロンモデルはシナプス電流の時間変化を考慮した CUBA-LIF モデルであり、学習には代理勾配を用いた Back Propagation Through Time(BPTT) が用いられた。ここで扱われる入出力はスパイクであるが、ネットワークの入出力は連続値であるため、線形層を用いた変換が行われている。これにより、スパイクのエンコードとデコードを学習に組み込むことで、最適化された変換を可能にしている。また、学習時には分割されている、姿勢推定ネットワークの出力の線形層の重み W_o と、制御ネットワークの入力の線形層の重み W_i は、推論時には以下の式で表されるように統合され、一つのネットワークとして動作する。

$$\begin{bmatrix} \phi_{\text{est}} \\ \theta_{\text{est}} \\ \psi_{\text{est}} \end{bmatrix} = W_i W_o s(t) \quad (2)$$

2.1.3 SNN の学習方法

SNN の学習には模倣学習が用いられた。学習データは、姿勢推定に相補フィルタ、制御にカスケード PID 制御を使用し手動で 20 分間飛行させ、500Hz で収集された。学習には、教師あり BPTT が用いられ、損失関数は以下の式で表される平均二乗誤差 (MSE) とピアソンの相関損失を組み合わせたものとして定義された。

$$J(p) = \text{MSE}(x, \hat{x}) + \frac{1}{2}(1 - \rho(x, \hat{x})) \quad (3)$$

また、非連続なスパイク関数の微分を近似するために、代理勾配が用いられた。本手法では、スケール付きアークタンジェントの導関数であり、以下の式で表される。

$$\frac{d}{dx} \left(\frac{1}{s} \arctan(sx) \right) = \frac{1}{1 + (sx)^2} \quad (4)$$

SNN は、電流と電圧のリーコンを考慮するため、出力が出されるまでにニューロンに一定以上の入力が蓄積される必要があり、出力に遅延が生じる。この遅延を補償するために、学習データのラベルを 6 ステップ先の値にシフトさせて学習が行われた。さらに、模倣学習を用いた事による現実とのギャップを低減するために、学習データの拡張が行われた。具体的には、(i) 初期の学習データで訓練された SNN を用いて飛行させ、その時に学習データ用のコントローラーで得られるはずだったデータ、(ii) 通常の PID のコントローラーにランダムな外乱を加えたデータが収集され、学習データに追加された。これらのデータ拡張により、より広範な状態空間での学習が可能となり、現実世界での適応性を向上させている。また、制御ネットワークの学習において、定常的な誤差を低減するための I 制御の学習は、(i) ニューロンの漏れにより長期間で情報を蓄積できない

こと、(ii) PD 制御の成分の学習が優先されることから困難である。これに対して、制御ネットワークの一部のニューロンのパラメーターを固定し、リーコンをなくすことで膜電位が積分として機能するようにし、そのニューロンの出力の正解ラベルを I 制御の出力のみに設定して学習を行った。

2.1.4 制御精度の評価

位置移動の能力を評価するために、提案手法を用いて 1m 前進し原点に戻るタスクを実施し、通常の PID 制御との比較が行われた。タスクは 10 回施行され、提案手法は全ての実行で PID 制御と同等の性能を示し、安定した目標追従が可能なことが示された。次に、各学習方法による SNN 制御と PID 制御による姿勢応答の比較を行うため、ロールを 0° , $+10^\circ$, -10° , 0° と変えて行ったときの応答の比較が行われた（図 3）。提案手法において、ロール角の目標値と制御値の RMSE は 3.03° であり、姿勢推定及び制御が正確に行われてことが示された。また、PID の RMSE は 2.67° であり、提案手法より高い精度であったが、これは外部から姿勢情報を直接与えられており、制御のみが誤差の要因であるためだと考えられる。提案手法では、姿勢推定と制御の両方が誤差の要因となるため、このことからも提案手法は十分な精度を達成していることが示された。

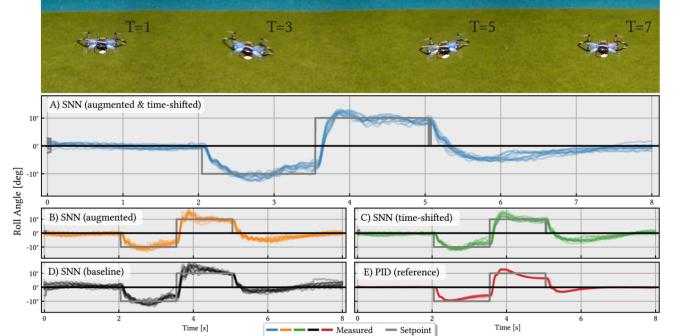


図 3: Attitude step responses of A) the fully-trained SNN system, B) the SNN trained with augmentation, C) the SNN trained with time-shifted data and D) the regular PID flight stack. [7]

2.1.5 消費電力の分析

最後に、本手法における消費電力の分析が行われた。提案手法は使用した小型クアッドローターに対して、現在利用可能なニューロモルフィックチップが大きすぎるため、従来のマルチプロセッサにて実装が行われた。一方で、SNN を用いることによる省電力性は単一のニューロモルフィックチップで動作させた場合に得られる。そのため、潜在的な消費電力の削減効果を評価するために、ネットワークが行う演算数に基づいて、消費電力の推定が行われた。その結

果、PID ベースの制御は 3000 回の加算、提案手法は 7500 回の加算が必要と等価と見積もられ、SNN 制御器は PID 制御と同じ桁のエネルギー効率を達成することが推定された。このエネルギー効率は、将来的にシステムが画像処理タスクと統合された場合に、スパイクの疎性により乗算を削減できるため、向上すると考えられている。また、イベントベース制御への拡張により、ホバリングなどの制御を行わない場合に、更に消費電力を削減できる可能性があると示唆された。

2.1.6 考察と今後の課題

本手法で示された、SNN の学習時のデータ拡張や、積分動作のニューロンによる実装などは、手法を拡張させる際にも有効であると考えられる。一方で、ラベルをシフトさせる手法は、ドローンがより複雑な軌道で飛行する場合や、移動速度が早くなる場合には適切に補償できなくなる可能性がある。そのため、より一般的に適用可能な遅延補償手法の検討が必要であると考えられる。また、示された電力効率は PID 制御と同等の桁であることが主張されたが、2 倍以上の計算量であり、特にバッテリー量が限られる火星ドローンにおいては無視できない差である。そのため、画像を用いた制御と統合した場合の消費電力と精度のトレードオフの分析が必要であると考えられる。

2.2 イベントベース視覚の複合学習による自己運動制御

2.2.1 概要

この研究は、イベントカメラを用いて取得したイベントベースの視覚情報を入力とし、低レベルの制御アクションを出力する SNN を提案している。ネットワークは視覚処理部と制御部にモジュール分割されており、視覚処理部は自己教師あり学習、制御部はシミュレーター上で進化アルゴリズムを用いた学習が行われた。また、提案手法は Intel Loihi プロセッサを用いて実装され、評価が行われた。

2.2.2 イベントカメラの特性

本手法では、視覚処理を行うためのセンサーとしてイベントカメラを用いている。イベントカメラは、従来のフレームベースのカメラとは異なり、ピクセル毎に独立して輝度変化を検出し、変化があった場合のみイベントとして出力するセンサーである。このカメラの特徴として、(i)高い時間分解能、(ii)低消費電力、(iii)高いダイナミックレンジが挙げられる [10]。

2.2.3 イベントベースのオプティカルフロー

オプティカルフローとは、連続する画像間での物体やテクスチャがどう動いたかをベクトルで表したものである。イベントカメラからの入力に対して SNN を用いてオプティカルフローを推定する手法も提案されており、エンコーダー・デコーダーモデルを用いた自己教師あり学習が用いられている [11]。ここで、自己教師あり学習が用いられるのは、オプティカルフローの正解ラベルをピクセル単位で大量に集めることができ難であるためである。SNN を用いた手法は、ANN と同等の水準を達成したことが示されているが、現状入手可能なニューロモルフィックチップ上に実装するにはネットワークのニューロン数が多すぎるという課題があり、実際の視覚処理に組み込むうと考える場合は、精度を維持しながらネットワークのサイズを削減する工夫が必要となる。

2.2.4 提案システムの概要

この研究では、図 4 に示すシステムが提案された。ドローンには、イベントカメラ、ニューロモルフィックプロセッサ、シングルボードコンピュータおよびフライトコントローラーが搭載され、イベントカメラからの情報を制御コマンドに変換する SNN がニューロモルフィックプロセッサ上で実行される。SNN の視覚処理部は、Loihi チップ上でイベントベースのオプティカルフローを実装する上で課題となる計算リソースの制約を克服するために、イベントカメラは静的でテクスチャが豊富な平坦な表面を見下ろしていることが前提となっている。この条件で、イベントカメラの画像平面の内、コーナーの閾値領域からの情報のみが用いられ、更に ROIあたりのイベント数を 90 に制限することで、200Hz の動作周波数が実現されている。

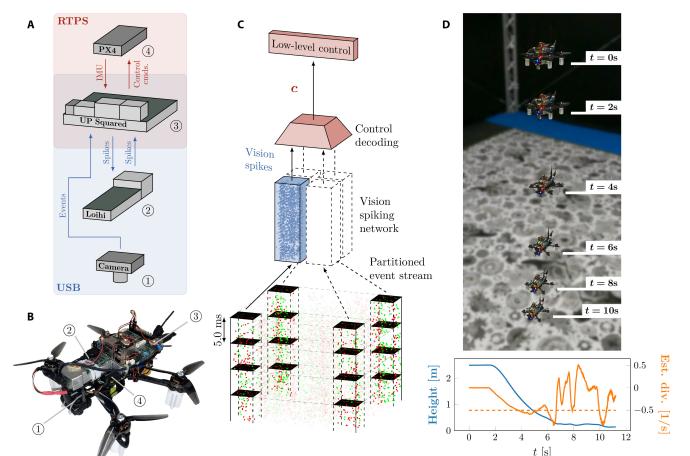


図 4: Overview of the proposed system ([8])

2.2.5 SNN の学習手法

本手法のSNNに用いられたニューロンモデルはCUBA-LIFモデルであり、学習には代理勾配が用いられた。視覚処理を行うネットワークは3つのエンコーダーとブーリング層で構成され、4つのROIからオプティカルフローが推定される。このフローから密なオプティカルフローを得るために以下の式が用いられた。

$$\mathbf{u}(\mathbf{x}, \mathbf{H}) = \begin{bmatrix} u(\mathbf{x}, \mathbf{H}) \\ v(\mathbf{x}, \mathbf{H}) \end{bmatrix} = \left(\mathbf{H}, \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \right)_{\text{Eucl}} - \begin{bmatrix} x \\ y \end{bmatrix} \quad (5)$$

ここで、 \mathbf{H} はホモグラフィ行列、 $\mathbf{x} = (x, y)$ は画像平面上のピクセル座標を表す。また、このホモグラフィ行列は、ある時間間隔におけるROI内のイベントの変位から計算される。この密なオプティカルフローを元に、動き補償のためのコントラスト最大化が行われた。以下にはネットワークの学習に用いられた損失関数を示した。

$$L_{\text{flow}} = L_{\text{contrast}} + \lambda L_{\text{smooth}} \quad (6)$$

この式(6)の内、 L_{contrast} は予測した動きを元にブレを補正した後のフロー画像がどれだけ正確であったかを評価するための損失である。また、 L_{smooth} はフローのスムーズさを評価するための正則化項であり、 λ はその重みを表す。

次に、制御は視覚処理での推定値を線形変換することを行われた。この推定値とは、3軸の速度と角速度、ピッチとロール、目標速度の9次元のベクトルであり、推力と目標とするロール、ピッチ、ヨーの4次元の制御コマンドを出力するための 4×9 の行列の重みが学習された。この学習には突然変異のみの進化アルゴリズムが用いられた。具体的には、100個のランダムな 4×9 行列の集団を用意し、 $\mathcal{N}(0, 0.001)$ のノイズを加えて適応度の評価を行い、適応度が上位100個の行列を次世代として、25,000世代まで学習が行われた。この際の適応度は以下の式で定義された。

$$F = \frac{1}{N_{\text{eval}}} \sum_{i \in N_{\text{eval}}} \sum_{j \in N_{\text{steps}}} \mathbf{w} \cdot \left(v_{\text{sp}, i}^{\mathcal{B}} - \begin{bmatrix} \hat{V}_x^{\mathcal{B}} \\ \hat{V}_y^{\mathcal{B}} \\ \hat{V}_z^{\mathcal{B}} \end{bmatrix}_j \right)^2 + (\hat{\omega}_z^{\mathcal{B}})^2 \quad (7)$$

ここで、 N_{eval} は評価回数、 N_{steps} は各評価におけるステップ数、 \mathbf{w} は重みベクトルであり、z垂直方向の誤差を10倍重視するように設定されている。

2.2.6 シミュレーションと実飛行での制御精度の比較

提案手法の評価は、視覚部と制御部で分けて行われた。まず、視覚部の評価では屋内環境でドローンの下部に搭載されたイベントカメラを用いて撮影されたイベントシーケン

スを用いて学習を行い、同時に計測したドローンの正確な位置と姿勢情報との比較が行われた。

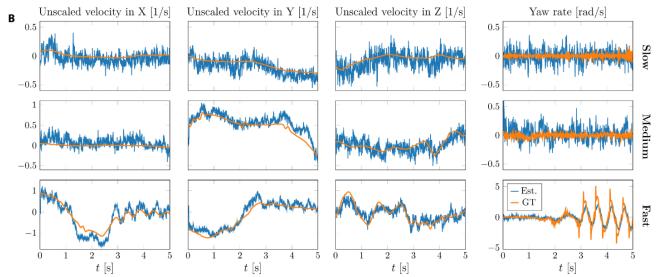


図5: Comparison of estimation and ground-truth visual observables for sequences with different motion speeds [8]

図5は、低速、中速、高速の3つの速度でのSNNによる速度の推定結果と正解ラベルの比較を示しており、どの速度でもネットワークは動きを正確に捉えることが出来ていることが示されている。特に、回転速度の推定においては、毎秒約4ラジアンの高速回転も推定可能であることが示されている。また、推定値が含む誤差に関しては、1つのカメラから運動の推定を行うことによるアーチャ問題が主な原因となっていると考えられている。また、推定したフローの平均エンドポイント誤差を元にした分析も行われた。まず、ANNの手法(Conv-GRU)とSNNを比較した場合、ANNが5.88、SNNが10.79となり、ANNの方が精度が高いことが示された。これは、floatを用いるANNのほうが高精度な計算が可能であるためだと考えられる。次に、画像全体を用いた場合と、ROIのみを用いた場合の比較において、SNNではROIのみの場合が精度が向上したことが示された。イベント数の制限でも同様に精度の向上が見られた。これは、ネットワーク内に膜電位として蓄積される情報が増えすぎる事による処理の不安定化が解消されたためだと考えられる。

次に、制御部の評価はシミュレーションと実飛行の両方で行われた(図6)。シミュレーションでは、16パターンの飛行目標が与えられ、学習が行われた。その結果、16パターン全てにおいて目標への到達が可能なことが示された。次に、実飛行での評価では、飛行には成功したもの多くのパターンで目標の姿勢、速度への到達を達成出来なかった。これは、単純な線形コントローラーを用いたことによる表現力の不足や、リアリティギャップ、積分制御の補正不足が原因であると考えられている。また、高速で飛ぶ場合の方が飛行軌跡が安定することも示された。これは、イベントカメラが生成するイベントが高速な場合ほど多くなるため、ノイズの影響が低減されたことが理由だと考えられている。また、PI制御器との比較においても上記の課題から、PI制御器の方が優れた性能を示した。

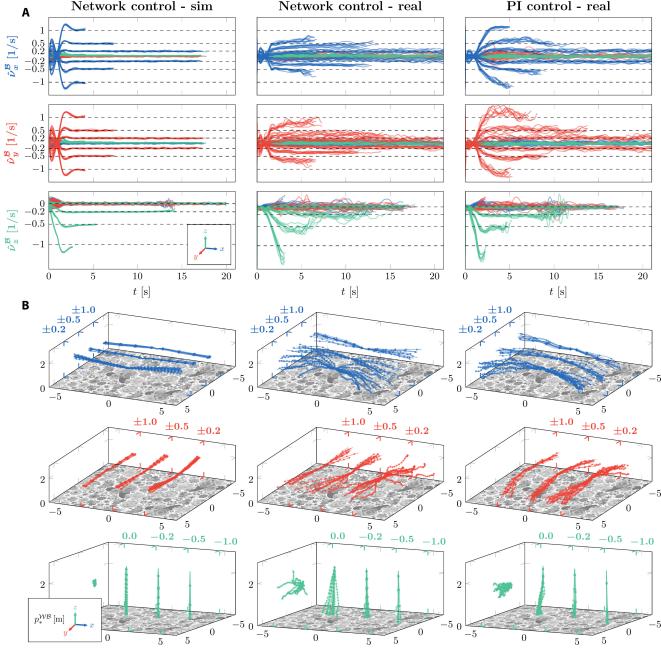


図 6: Comparison of results obtained in simulation and during real-world flight tests [8]

2.2.7 推論時間と消費電力

推論時の速度と消費電力の評価を行うために、提案手法を Loihi チップと NVIDIA Jetson Nano 上で動作させた場合の比較が行われた。Loihi チップでの SNN 実行時の消費電力は 0.95W であるのに対し、Jetson Nano の 10W モードで動作させた場合の消費電力は 2.98W であった。さらに、Loihi での消費電力の内、0.9W はアイドル時の電力であり、これを削減することでより低消費電力化が可能であると示唆された。また、推論速度も 1 秒間の推論回数において、ドローンが低速な場合、Loihi での推論は 1637inf/s であるのに対し、Jetson Nano では ANN を動作させた場合においても、60inf/s であり、Loihi の方が大幅に高速であることが示された。

2.2.8 考察と今後の課題

本手法では、ドローンの視覚制御をオンチップで 200Hz で実現することが出来ており、SNN を用いた視覚制御の有効性が示された。特に、イベントデータからのフロー推定について、軽量化のために行った、ROI による情報の制限が精度の向上に繋がったことは有用な結果であると考えられる。一方で、本手法で想定された安定した環境下での飛行は、実応用においては重大なアリティギャップを生じる可能性があり、ドローンの回転方向の動きが複雑化した場合や、床面のテクスチャが乏しい場合などに現状の視覚処理モデルでどの程度の性能が維持できるかは不明である。そのため、より多様な環境下での評価や、視覚処理モデルの

改良が必要であると考えられる。

3 SNN による高機動ナビゲーション

3.1 概要

本研究では、深層強化学習アルゴリズムである近位ポリシー最適化 (PPO) を使用して SNN 飛行ポリシーの学習が行われた。この飛行ポリシーでは、システムの状態をドローンの低レベルの制御コマンドに変換することを提案している。本手法はシミュレーション上でリングを回避しながら進むタスクにて、SNN の低計算量で時間情報を処理することが可能な特性を活かし、ANN と比べて高い成功率で、完了時間も短縮可能なことが示された。

3.2 PPO アルゴリズムの概要

近接ポリシー最適化 (PPO) は、強化学習アルゴリズムの一つであり、信頼性と効率性を両立させながら実装がシンプルであることが特徴である [12]。PPO では、目的関数として以下の式で表されるクリップ付き代理目的関数が用いられる。

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right] \quad (8)$$

この式では、ポリシーの更新において、古いポリシーと新しいポリシーの比率 $r_t(\theta)$ が 1 から大きく外れないように制約を設けることで、安定した学習を実現しており、様々なタスクで高い性能を発揮している。

3.3 PPO アルゴリズムによる SNN エージェントの学習

提案されたフレームワークでは、物理シミュレーション環境、PPO アルゴリズムで学習された SNN ベースのエージェント、低レベルコントローラーが組み合わされている（図 7）。高速で移動するゲートを通過するタスクにおいて、SNN エージェントはシミュレーションステップ毎に、システムの状態 $s \in \mathbb{R}^{19} := (d_{tgt}, q, v, \omega, d_{gate}, v_{gate})$ を受け取る。ここで、 d_{tgt} は目標位置までの距離、 q, v, ω はドローンの姿勢、速度、角速度、 d_{gate} はドローンと次のゲートの中心との距離、 v_{gate} はゲートの中心の速度を表す。このシステムの状態を元に、SNN エージェントは低レベルなフライトコントローラーのコマンドとなる共通推力とボディーレートを出力する。このとき、報酬関数によって定義される報酬信号 r が与えられる。SNN は、1024 個のニューロンを持つ 2 つの隠れ層から構成されており、シグモイド代理勾配関数を用いて学習が行われた。次に、報酬関数には以下

の式が用いられた。

$$r_t = r_{gate,t} + r_{pos,t} + r_{pos,t} \cdot (r_{up,t} + r_{spin,t}) - \beta \cdot r_{acc,t} \quad (9)$$

ここで, $r_{gate,t}$ はゲート通過に対する報酬, $r_{pos,t}$ はドローンの位置の報酬であり, 以下の式で定義される。

$$r_{pos,t} = \frac{0.05}{1 + d_{tgt}^2} + \frac{2.5}{1 + d_{gate}^2} - \frac{2.0}{1 + d_{ring}^2} \quad (10)$$

この式は, ドローンの位置の報酬は, 最終的な目標地点と通過するゲートの中心に近いほど高くなり, リングに近いほど低くなることを示している。また, d_{tgt} と d_{gate} の係数の比率から, 最終地点への到達が早いほど報酬が最大化されるように設計されている。さらに, $r_{up,t}$ と $r_{spin,t}$ はドローンの姿勢を水平に保つための報酬, $r_{acc,t}$ は急激な加速度を抑制するための罰則項であり, ドローンの軌道を滑らかにするために導入されている。シミュレーション環境では, 0.5~9.0m のランダムな距離で配置された 5 つのゲートを通過するタスクが設定された。また, 各ゲートは, Y-Z 平面内で振動するように設定された。

Training Flow

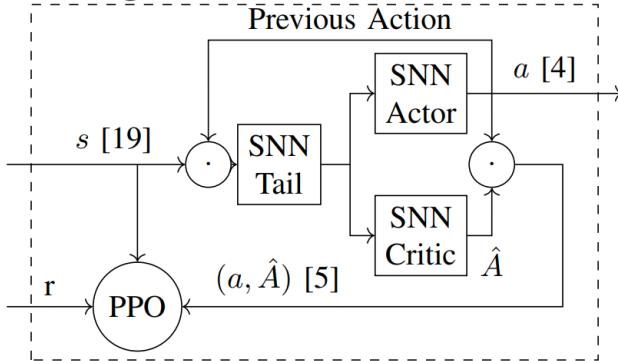


図 7: Schematic structure of the training flow [9]

3.4 SNN と ANN の比較

提案手法との比較を行うベースラインとして, ELU 活性化関数を備えた 4 層からなる ANN アーキテクチャが用いられた。まず, タスクの成功率の比較において, 提案手法はすべてのゲート間距離の設定において, ANN ベース手法よりも高い平均報酬を達成した。これは, (i) 高い成功率, (ii) 各エピソードの完了時間の短縮に起因するものである。まず, 成功率の比較について, SNN は SOTA である ANN の平均成功率である 83% を上回る 85% を達成した(図 8(b))。特に, ゲート間距離が 0.5m の場合と初期速度が高い場合において, ANN は成功率が大きく低下したが, SNN は安定した成功率を維持した(図 8(c))。これは,

SNN の方が急速な状態変化への適応性に優れていることを示している。この理由として, SNN のニューロンが膜電位として時間情報を保持できるため, 過去の状態に基づいて現在の行動を調整できることが挙げられる。次に, 完了時間の比較において, ゲート間隔が長くなるに連れて, SNN は ANN よりも短い完了時間であることが示された(図 8(a))。これについて, SNN は ANN よりも軌道全体で高い速度が維持されていたことが示された。これは, SNN が時間情報を適切に処理できることにより, ゲート通過時の加減速が効果的に行われたためであると考えられる。

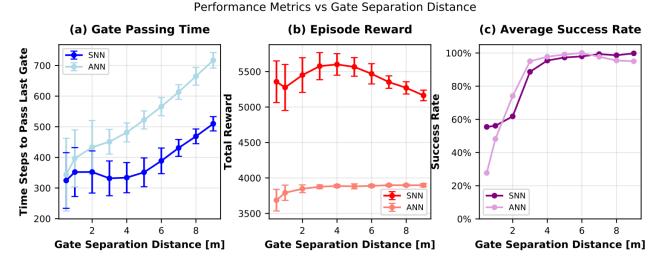


図 8: Comparison between SNN (our method) and ANN (SOTA) across three performance metrics [9]

3.5 エネルギーとメモリの分析

提案 SNN モデルは, 1024 個のニューロンを持つ隠れ層が 2 つで構成されており, ニューロンの総数は 2048 個であった。この内, 15~20% が活性化されており, ニューロンが 23 個の入力層と 4 個の出力層を含めた場合の実際の演算数は, 約 215,000 回であると見積もられる。また, モデルのメモリ使用量は, シナプス接続毎の重み, 層ごとの時定数, ニューロン毎のバイアスをモデルパラメータとし, float32 で表現した場合, 約 4.3MB であると見積もられた。また, これらのパラメーターを量子化し, int8 で表現した場合, 約 1.08MB に削減できると見積もられた。

3.6 考察と今後の課題

本手法の結果から, 提案された SNN モデルは SNN の時間情報を処理する能力から, 動的な環境におけるナビゲーションに優れた適応性を持つことが示された。一方で, 火星環境への応用を考えた場合に, 突風などの外乱を除けば, 静的な環境であるため, 動的な環境への対応よりも, 未知の環境への適応性が重要になると考えられる。そのため, 用いる報酬関数もそれに応じた設計へと変更することが必要である。次に, 本手法で行われた実験はシミュレーション上のものであり, 入力はシステムの状態という理想的なものであった。そのため, 実環境におけるドローンの飛行制御に適用しようとする場合, システムの状態を推定するた

めの方法が必要となる。SNN の性質を最大限に活用するためには、イベントベースの視覚処理と組み合わせたニューロモルフィックな構成を作ることが望ましいが、視覚処理を SNN に含める場合、計算リソースの制約を満たすことができるのかについては検討が必要である。

4 まとめ

本調査では、ドローンの飛行制御において、適応性と堅牢性を省エネルギーで向上させるためのアプローチとして、SNN を用いた 3 つの論文を紹介した。まず、ドローンの姿勢推定について 2 本目の研究から、イベントカメラを用いた構成を用いる場合、低消費電力で視覚処理をリアルタイムに行えることが示されたが、入手可能なニューロモルフィックチップの計算能力の制約から、視覚情報の一部のみを用いるなどの工夫が必要であることが示された。この工夫による姿勢推定の精度の悪化については議論されていないが、飛行実験では制限された環境でのデータ収集が行われていたため、火星環境への応用を考える場合、姿勢推定に十分な情報が得られるかは不明である。そのため、一本目の研究で紹介された IMU と組み合わせたマルチモーダルなアプローチなどによる精度の向上が検討されるべきであると考えられる。次に、制御への SNN の応用に関しては、各学習方法における特徴が反映されると考えられる。模倣学習は、特にリアリティギャップの低減が可能であることが特徴であるが、学習データの多様性への依存性からデータの拡張が重要であると考えられる。一方、進化アルゴリズムは今回の適用例では単純な線形層の学習への適用に留まっており、モデルサイズをより小さくすることが求められる場合には、最適かもしれないが、複雑な環境への適応性を求める場合の制御コントローラーの学習には十分な精度が得られないという懸念がある。最後に強化学習は、未知の環境への適応性が高いことが特徴であるが、この特徴を最大限に活用するためには、適切な報酬関数の設計が重要である。今後の研究では、火星表面の風による外乱への堅牢性を獲得するために、外乱を含むシミュレーション環境での強化学習の実施や、イベントカメラによる視覚処理と強化学習を組み合わせた制御モデルを構築することを目指したいと考えている。

参考文献

- [1] P McEnroe, S Wang, and M Liyanage, "a survey on the convergence of edge computing and ai for uavs: Opportunities and challenges", *IEEE Internet of Things Journal*, Vol. 9, No. 17, pp. 15435–15459, 2022.
- [2] S Rezwan and W Choi, "artificial intelligence approaches for uav navigation: Recent advances and future challenges", *IEEE access*, Vol. 10, pp. 26320–26339, 2022.
- [3] M Davies, N Srinivasa, T-H Lin, G Chinya, Y Cao, S H Choday, G Dimou, P Joshi, N Imam, S Jain, et al., "loih: A neuromorphic manycore processor with on-chip learning", *Ieee Micro*, Vol. 38, No. 1, pp. 82–99, 2018.
- [4] K Yamazaki, V-K Vo-Ho, D Balsara, and N Le, "spiking neural networks and their applications: A review", *Brain sciences*, Vol. 12, No. 7, p. 863, 2022.
- [5] NASA. "ingenuity mars helicopter". <https://mars.nasa.gov/technology/helicopter/>, 1 2024. "(Accessed: 2025-11-05)".
- [6] D AR Harbour, K Cohen, S D Harbour, B Ratliff, A Henderson, H Pennel, S Schlager, T M Taha, C Yakopcic, V K Asari, et al., "martian flight: Enabling motion estimation of nasa's next-generation mars flying drone by implementing a neuromorphic event-camera and explainable fuzzy spiking neural network model", In *2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (DASC)*, pp. 1–10. IEEE, 2024.
- [7] S Stroobants, C De Wagter, and G CHE De Croon, "neuromorphic attitude estimation and control", *IEEE Robotics and Automation Letters*, 2025.
- [8] F Paredes-Vallés, J J Hagenaars, J Dupeyroux, S Stroobants, Y Xu, and G CHE de Croon, "fully neuromorphic vision and control for autonomous drone flight", *Science Robotics*, Vol. 9, No. 90, p. eadi0591, 2024.
- [9] Y-C Lee, S Mengozzi, L Zanatta, A Bartolini, A Acquaviva, and F Barchi, "bio-inspired drone control: A reinforcement learning-trained spiking neural networks for agile navigation in dynamic environment", In *2025 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, pp. 1–8. IEEE, 2025.
- [10] G Gallego, T Delbrück, G Orchard, C Bartolozzi, B Taba, A Censi, S Leutenegger, A J Davison, J Conradt, K Daniilidis, et al., "event-based vision: A survey", *IEEE transactions on pattern analysis and machine intelligence*, Vol. 44, No. 1, pp. 154–180, 2020.
- [11] J Hagenaars, F Paredes-Vallés, and G De Croon, "self-supervised learning of event-based optical flow with spiking neural networks", *Advances in Neural Information Processing Systems*, Vol. 34, pp. 7167–7179, 2021.
- [12] J Schulman, F Wolski, P Dhariwal, A Radford, and O Klimov, "proximal policy optimization algorithms", *arXiv preprint arXiv:1707.06347*, 2017.