A Neuromorphic Approach to Obstacle Avoidance in Robot Manipulation

Ahmed Faisal Abdelrahman, Prof. Paul G. Plöger (Bonn-Rhein-Sieg University of Applied Sciences) | Prof. Maren Bennewitz (University of Bonn) | Prof. Matias Valdenegro Toro (University of Groningen)

Introduction

- **Neuromorphic computing/engineering** implements computational principles of the brain and nervous system.
- **Event cameras** (ECs) mimic the functions of the retina. **Spiking neural networks** (SNNs) mimic the dynamics of biological neurons.
- These concepts are interesting for their i) biological fidelity and ii) practical advantages.
- However, the event-driven and spike-based paradigms remain scarcely explored in robotics.
- **Main contribution:** An implementation and experimental evaluation of a neuromorphic approach to a common robotics problem.
- **Application:** Online obstacle avoidance on a manipulator with an on-board camera.
- **Objective:** Investigating the viability and utility of neurologically-inspired sensors and algorithms through systematic experimentation.

A Neuromorphic Pipeline: Approach and Implementation

We present a pipeline of components that enable end-to-end processing of visual data into motion trajectory adaptations, utilizing ECs and SNNs (neuromorphic sensor and neural circuitry).

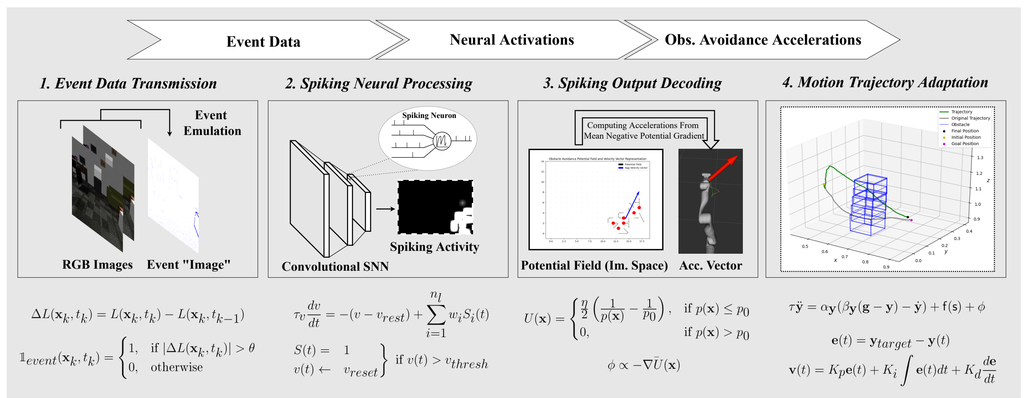


Figure 1: The main components and processing stages of our neuromorphic pipeline.

1. **EC/EC Emulator:** Produces event data. Either an event camera, or an emulator which transforms RGB data.
2. **Convolutional SNN:** Processes event data in a network of spiking neurons arranged in convolutional layers.
3. **Obstacle Avoidance Component:** Decodes SNN outputs (spike trains) into obstacle avoidance velocities using a potential fields (PF) method.
4. **Adaptive Motion Planner:** Generates end-effector positions of a pre-planned trajectory while adjusting the plan according to the obstacle avoidance component's output; a dynamic motion primitive (DMP). Positions set by the DMP are followed using a PID controller.

Evaluation Methodology

- Experiments: simple reaching tasks with a Kinova Gen3 arm involving obstacles.
- A distribution of 416 task scenarios provide a variety of testing conditions.
- Performance is compared to non-adaptive, baseline executions.

Simulation Experiments

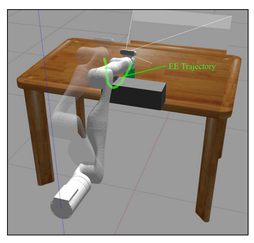
- We use disjoint sets of scenarios in a tuning → validation → testing procedure:
  1. **Tuning**: manually tune parameters to find candidate sets of parameter values.
  2. **Validation**: evaluate tuned parameter sets to identify best-performing set.
  3. **Testing**: perform final experiments with best-performing set.

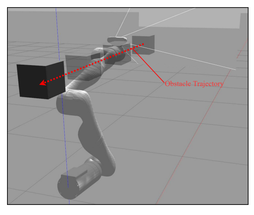
Real Robot Experiments

- We transfer the final parameter set to a real robot and run the same experiments.

Table 1: Variables that define task scenarios. Example scenario: {Task 2, Bookstore, Brick, Box, Medium}.

Variable	Values
Task	Task 1, 2, 3, or 4 [1]
Background	Empty, Office, Bookstore, Kitchen
Obstacle Type	Box, Buckyball, S. Sphere, Rock
Obstacle Color	White, Red, Y-B, Brick
Obstacle Speed	Low, Medium, High

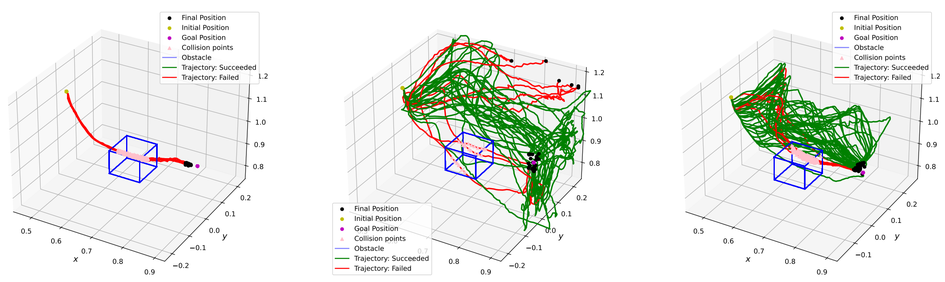




1 Task types: Reach position behind static obstacle (1); Reach position behind a dynamic obstacle (2); Reach for an object on a table behind a static obstacle (3); Avoid incoming dynamic obstacle and return to position (4).
2 Real experiment scenarios (task type, background, obstacle): Task 1, BG1, wooden block (R1); Task 2, BG2, hand (R2); Task 2, BG2, metal bar (R3); Task 2, BG2, wooden block (R4).

Experimental Findings

- Median success rates of 84% and 92% in simulated and real experiments.
- Obstacle-aware trajectories were quantitatively similar to nominal executions in terms of execution times, trajectory lengths, and velocity magnitudes.
- Trajectories were adequately reliable, predictable, safe, and smooth.



(a) Baseline (no adaptivity) *(b) Pre-tuning parameters* *(c) Post-tuning parameters*

Figure 2: Trajectories executed in a testing scenario (40 trials each).

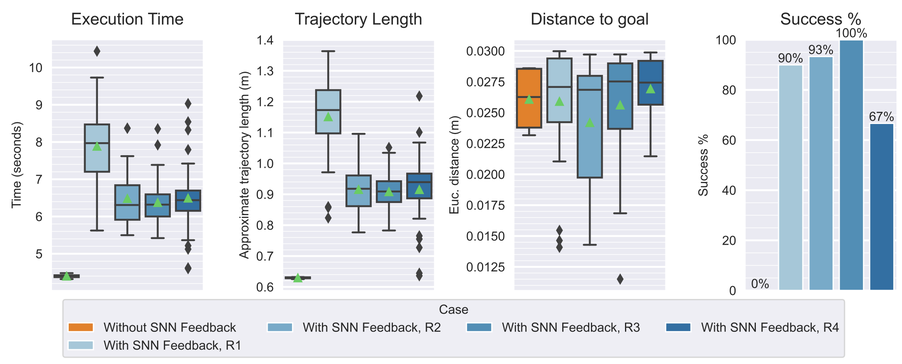
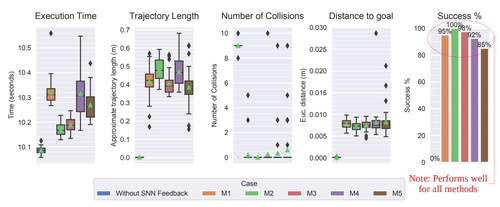


Figure 3: Quantitative metrics from multiple trials i) without and ii) with our pipeline (four scenarios[2]).

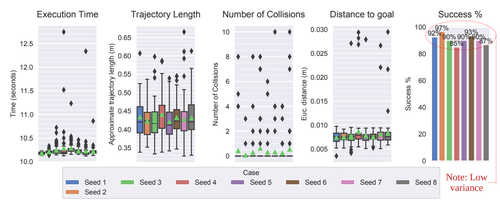
Further Analyses

Comparing Event Emulation Methods



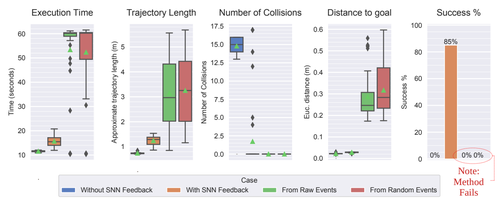
The SNN is robust to variations in event data, inherently handling variance and/or noise.

SNN Weight Initializations



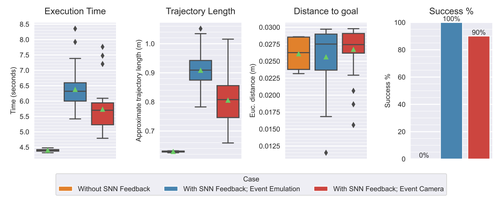
SNN weights have a non-negligible, but not excessive, effect on performance.

Decoding Velocities from Raw Event Data



This strategy fails, highlighting the importance of the SNN dynamics.

Real Event Camera Tests



Similar task performance with a real EC further validates our conclusions.

Summary

- We designed, implemented, and evaluated a neuromorphic approach to obstacle avoidance on a robot manipulator with a single on-board camera.
- Our pipeline transforms visual inputs into corrective obstacle avoidance maneuvers, combining high-level trajectory planning and low-level reactive adjustments using DMPs.
- Event-based vision and spiking networks provided neuromorphic sensing and processing.
- Experiments showed that our approach is successful in achieving **real-time, online obstacle avoidance** across **various task scenarios**, proving its utility over a non-adaptive baseline.
- Adaptive trajectories minimally deviated from baseline trajectories and were often **predictable**, **safe**, and reasonably **smooth**.
- Further analyses validated the SNN's useful properties and compatibility with a real camera, in addition to motivating future applications of learning-based optimization.

Contact

Ahmed Abdelrahman
Autonomous Systems Group
Hochschule Bonn-Rhein-Sieg

Email: ahmed.abdelrahman@inf.h-brs.de
Grantham-Allee 25
53757 Sankt Augustin, Germany