arXiv:2303.08778v1 [cs.RO] 15 Mar 2023

面向自主无人机飞行的全神经形态视觉与控制

F. PAREDES-VALLÉS† , J. J. HAGENAARS†∗ , J. D. DUPEYROUX† , S. STROOBANTS, Y. XU, 和 G. C. H. E. DE CROON

代尔夫特理工大学航空航天工程学院微型飞行器实验室，荷兰代尔夫特*通信作者：*j.j.hagenaars@tudelft.nl*。

†贡献相同。

生物感知与处理具有异步、稀疏的特点，因此能够实现低延迟、节能的感知与行动。在机器人学领域，用于事件相机视觉的神经形态硬件和脉冲神经网络有望展现类似特性。然而，由于当前嵌入式神经形态处理器的网络规模有限，且脉冲神经网络难以训练，机器人实现仍局限于采用低维感觉输入和运动动作的基本任务。在此，我们展示首个用于控制自由飞行无人机的全神经形态视觉到控制流水线。具体而言，我们训练了一个脉冲神经网络，该网络接收高维原始事件相机数据，并输出用于执行自主视觉飞行的底层控制动作。网络的视觉部分由五层和 28.8k 个神经元组成，将输入的原始事件映射为自运动估计，并利用真实事件数据进行自监督学习。控制部分由单个解码层构成，并通过无人机模拟器中的进化算法进行学习。机器人实验表明，这一完全训练所得的神经形态处理流水线成功实现了从仿真到现实的迁移。无人机能够准确跟踪不同的自运动设定值，从而实现悬停、着陆和侧向机动，即使同时进行偏航也不例外。该神经形态处理流水线在机载英特尔 Loihi 神经形态处理器上运行，执行频率为 200 Hz，每次推理仅消耗 27 µJ。这些结果展现了神经形态感知与处理在实现更小、更智能的机器人方面的潜力。

1. 引言

过去十年来，深度人工神经网络（人工神经网络）彻底改变了人工智能领域。其中一项重要成就是视觉处理能力的显著提升，如今计算机视觉在特定任务上的表现甚至已经超越人类 [1]。机器人学也从这一发展中获益匪浅，深度人工神经网络在立体视觉[2, 3], 光流估计 [4–6], 分割 [7, 8], 目标检测 [9–11], 和单目深度估计 [12–14]等任务中达到了最先进的性能。然而，这种高性能通常依赖规模庞大的神经网络，因此需要相当繁重且耗电的处理硬件。这限制了大型机器人（如自动驾驶汽车）能够执行的任务数量，甚至使其无法部署在资源限制极为严格的小型机器人上，例如小型飞行无人机。

神经形态硬件或许能为这一问题提供解决方案，因为它模拟了生物大脑中感知和处理的稀疏、异步特性 [15]。例如，神经形态事件相机中的像素只会传输亮度变化的信息 [16]。由于通常只有一小部分像素的亮度会发生显著变化，因此视觉输入是稀疏的，后续事件之间的间隔可达微秒量级。与传统的基于帧的计算机视觉相比，事件相机视觉输入的异步、稀疏特性代表着一种范式转变。理想情况下，处理过程应利用这些特性，实现更快速、能效更高的处理。然而，目前基于事件的视觉处理的主要方法，是在较长时间内累积事件，创建一个表示较长时间信息的“事件窗口”。随后，采用类似于传统图像帧的方式，利用人工神经网络处理该窗口 [17–20]。要充分发挥神经形态视觉的潜力，一个重要途径是借助专为实现脉冲神经网络（SNN）而设计的神经形态处理器，异步处理到来的事件 [21, 22]。这类网络的时间动态特性更接近生物神经元。具体而言，神经元具有膜电压，可对传入输入进行积分，并在电压超过阈值时产生脉冲。脉冲的二值特性使其能够实现比传统人工神经网络中的浮点运算

更加节能的处理。尽可能减少脉冲活动也能进一步提升能效，这同样是生物大脑的一项主要特性 [23]。将神经形态视觉与神经形态处理相结合，有望实现低能耗、低延迟的视觉感知与行动，敏捷动物（如飞行昆虫）便展示了这种能力 [24]。

本文介绍了首个用于控制自由飞行无人机的全神经形态视觉到控制处理流水线，展示了神经形态硬件的潜力。为此，我们克服了当前神经形态感知与处理面临的若干挑战。例如，与人工神经网络相比，脉冲神经网络的训练目前仍要困难得多 [25, 26], ，这主要是由于其稀疏、二元和异步特性。人工神经网络可以直接将连续值作为输入或输出，而脉冲神经网络则必须对其进行编码或解码。尽管已有多种编码选项可用 [27–29],，但鉴于问题的特性，哪种选择最佳仍远未明朗——生物大脑如何编码信息也仍然未知 [30–33]。脉冲神经网络学习中最为人熟知的难点，是脉冲激活函数不可微，因此无法直接使用朴素的反向传播。目前，借助替代梯度，这一问题已得到相当成功的解决 [34, 35]。此外，虽然更丰富的神经动力学有望表示更复杂的时序函数，但也更难以调节；训练过程中神经活动还可能饱和或衰减，从而阻碍进一步学习。由于可能起作用的参数众多，其原因很难分析。视模型而定，相关参数可能包括神经泄漏值和阈值、循环权重，以及突触迹的时间常数。解决方案或许在于学习这些参数 [36, 37], ，但这会进一步增加学习问题的维度。最后，若面向机器人应用，现有嵌入式神经形态处理平台的限制也使脉冲神经网络的训练和部署更加复杂；这些平台在神经元和突触数量方面通常仍相当有限。例如，ROLLS芯片 [38] 可容纳256个脉冲神经元；Intel Kapoho Bay（采用USB闪存盘外形，搭载两枚Loihi 芯片 [39] ）可容纳262.1k个神经元 [40], ；而 [41] 768中的SpiNNaker版本可容纳k个神经元。尽管这些芯片除了神经元数量外还有许多其他差异，但这个小样本已足以表明，当前最先进的脉冲神经网络难以轻易部署到机器人上。近期在光流确定等视觉复杂任务上训练的脉冲神经网络 [42, 43], ，其网络规模对于在当前用于嵌入式系统的神经形态处理硬件上实现而言仍然过大。这些研究中规模最小的脉冲神经网络是用于光流估计的LIF-FireFlowNet [43],，但它仍拥有370万个神经元（输入分辨率为128x128）。

因此，该领域的开创性研究在复杂度方面一直受到限制。很早期的一项研究通过进化脉冲神经网络的连接结构，将一台轮式 Kephera机器人的 16 个视觉亮度输入映射到两个电机输出 [44]。该进化得到的脉冲神经网络在软件中仿真，使机器人能够在黑白条纹环境中避开墙壁。探索脉冲神经网络在机器人学中的大多数研究都聚焦于仿真。例如，在 [45], 中，模拟事件相机产生的事件（分辨率为 128x128 像素）被累积成机架，再经过时间压缩，转换为 8x4 个泊松输入神经元。这些输入能够捕捉道路边界清晰的白线，随后直接映射到两个输出神经元，并借助奖励调制的脉冲时序依赖可塑性（R-STDP）学习，使机器人保持在道路中央。机器人示例

在硬件上进行神经形态处理的实例则较为少见。一个早期实例见于 [41], ：该研究将一台分辨率为 128x128 像素的事件相机连接到 SpiNNaker神经形态处理器，使一台行驶机器人能够通过一个由 128 个神经元组成的赢家通吃网络，区分以不同频率闪烁的两盏灯。在 [46] 中，研究人员设计了一个脉冲神经网络，使机器人能够跟踪视场上半部分的光目标，同时避开视场下半部分事件密集的区域。该网络成功部署在 ROLLS神经形态芯片 [38] 上，并在办公室环境中进行了测试。近年来，研究者越来越关注飞行机器人，即无人机，因为它们需要快速响应，同时在尺寸、重量和功耗（SWaP）方面受到严格限制。在 [40],中，研究人员将一个脉冲神经网络部署在固定于试验台的双旋翼装置上，使其横滚角与相机前方的黑白圆盘对齐。该脉冲神经网络结合了用于寻找线条的视觉霍夫变换[47] ，以及用于生成螺旋桨指令的比例—微分（PD）控制器。最后，在 [48],中，一个脉冲神经网络先在仿真中进化，再部署到 Loihi 上，用于自由飞行无人机的视觉着陆。该控制网络仅由 35 个神经元组成，因为视觉处理仍由传统的基于机架的计算机视觉方法完成。此外，值得注意的是，该脉冲神经网络只控制无人机的垂直运动；无人机的横向位置则由传统控制算法和外部运动捕捉系统控制。

一种完全神经形态的基于视觉的导航解决方案

本文提出的视觉到控制流水线由一个脉冲神经网络组成，该网络经过训练，可接收高维原始事件相机数据，并以约 200 Hz 的频率输出底层控制动作，从而执行自主的基于视觉的自运动估计与控制。此外，我们提出了一种学习方案，以规避在控制策略学习中对事件相机视觉输入进行缓慢且不准确仿真的问题。具体而言，该方案将视觉与控制分开，使网络的视觉部分能够通过自监督学习进行训练，而控制策略则可以在无需仿真事件的无人机模拟器中学习。最终的处理流水线如图1C 所示，已在 Loihi 神经形态处理器上实现 [39] ，并搭载于一台小型飞行机器人（见图1B）上，用于基于视觉的导航。图1A 展示了所采用硬件配置的示意图。该系统能够完全自主地跟踪自运动设定点，即无需任何外部辅助，例如定位系统。图1D 展示了我们的神经形态处理流水线在无人机控制回路中进行着陆实验的一个示例。图中显示高度平稳下降（蓝线），以及光流发散——即缩放速度向量 $\nu^{\mathcal{B}} = v^{\mathcal{B}}/p^{\mathcal{WB}_z}$ 的垂直分量，其中$p^{\mathcal{WB}_z}$ 是无人机距地面的高度。该发散曲线是光流发散着陆的典型表现：先逐渐接近设定点 $\nu^{\mathcal{B}_z,sp} = -0.5$ 1/s，随后在接近地面时变得更加振荡 [49]。

如前所述，在嵌入式神经形态硬件上部署此类处理流水线的主要挑战在于，这项技术尚处于初步发展阶段，因此可用计算资源受到严格限制。在本项目中，我们做出了一些设计决策，以适应这些限制。首先，视觉处理流水线假设无人机上的事件相机 DAVIS240C [50],朝下，拍摄静态平坦表面。预先了解视觉场景的结构，有助于简化对相机自我运动（

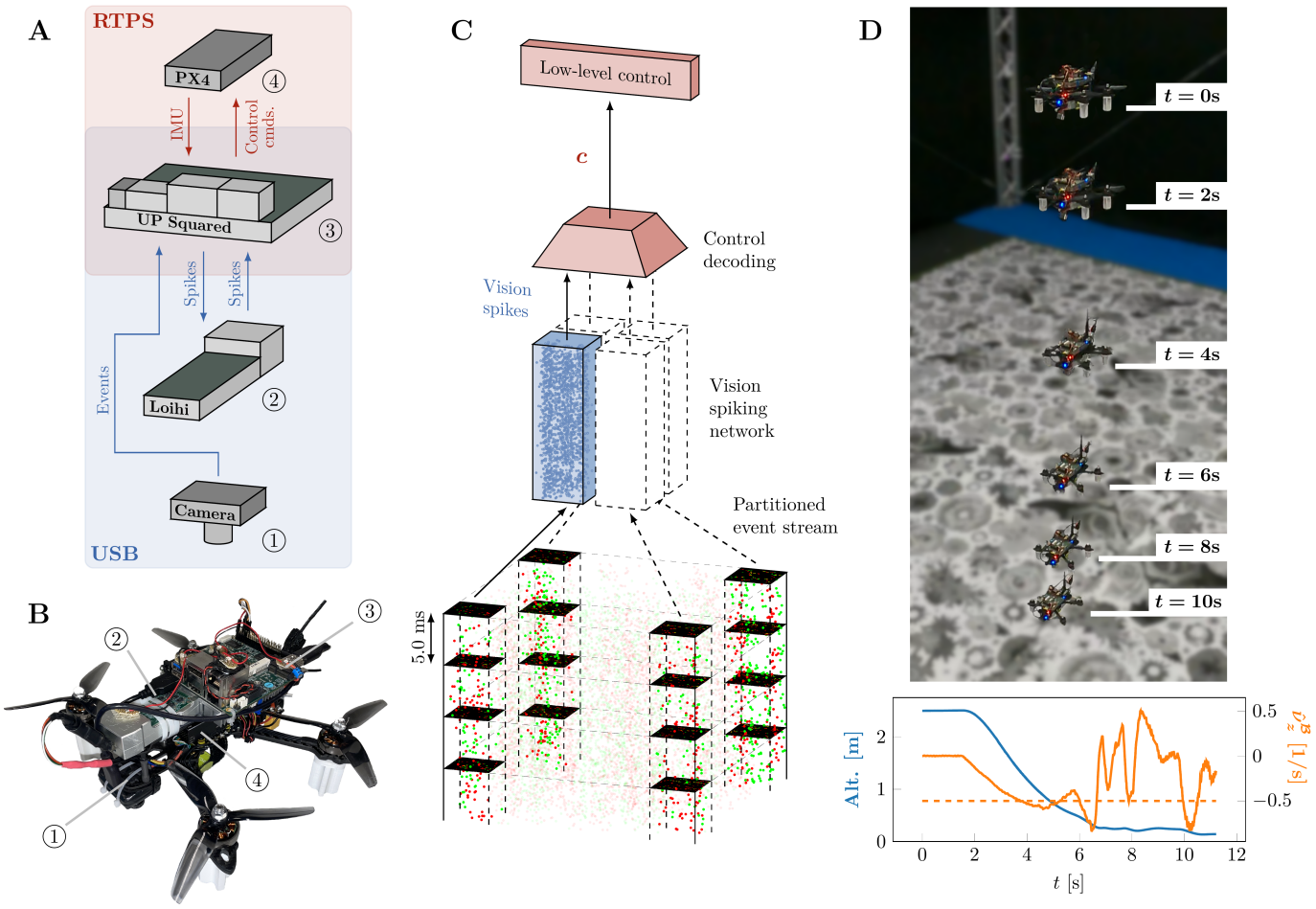

图 1. 所提出系统的概览。(A)本研究中使用的四旋翼飞行器（总重量 1.0 kg，轴尖间距直径 35 cm）。**(B)**硬件概览，展示了事件相机、神经形态处理器、单板计算机和飞行控制器之间的通信。**(C)**处理流水线概览，展示以事件为输入，经视觉网络处理并解码为控制指令的过程。**(D)**光流发散着陆系统演示。

也就是无人机的运动）的估计，方法是利用光流信息，如[48, 49, 51–53]所示。光流，即场景点在图像空间中的表观运动，可以通过多种方法从事件相机的输出中估计得出，方法范围从稀疏特征跟踪算法 [54] 到稠密（即逐像素）的机器学习模型 [17, 19, 43]。为了构建高效且高带宽的视觉处理流水线，第二项设计决策是降低基于事件的视觉数据的空间分辨率，仅处理图像角点的信息，而非整个图像空间。具体而言，如图1C 和2A 所示，我们提出在每个图像角点分别应用一个小型脉冲神经网络；经过最近邻下采样后，每个角点的尺寸为 16x16 像素。每个网络由分布在五个脉冲层中的 7.2k 个神经元和 506.4k 个突触构成，即一个输入层、三个自循环编码器和一个池化层。四个角点的网络参数（即权重、阈值和泄漏值）完全相同，并估计相应角点的光流，单位为像素每毫秒。由于采用静态平面场景假设，四个图像角点处场景点的表观运动编码了相机速度的非度量信息（即按距离缩放）

表面沿光轴的距离）及其旋转速率之间的线性关系 [55]。我们将这一关系与在仿真中训练的线性控制层结合起来，把编码光流的脉冲直接转换为推力和姿态控制指令（给定一个设定点）。这使我们能够以完全神经形态的方式，处理自由飞行无人机的视觉控制问题。

我们将视觉到控制流水线的训练拆分为两个独立的框架。一方面，负责将输入事件映射为光流的流水线视觉部分，采用对比度最大化框架以自监督方式进行训练 [56, 57]。这种方法的核心思想是，通过补偿运动边缘触发的事件之间的时空错位（即事件去模糊），可以获得准确的光流信息。在本研究中，我们采用 [43] 提出的表述方式，如图2所示。我们的脉冲网络按顺序处理非重叠的 5 毫秒时间窗口内的角点事件，并在每个时间步长提供光流估计。仅在训练期间，我们利用四个角点的运动信息来参数化单应性变换；在假设平面静止的前提下，该变换使我们能够像 [55, 58–60]中那样获得稠密光流。遵循 [43], 我们累

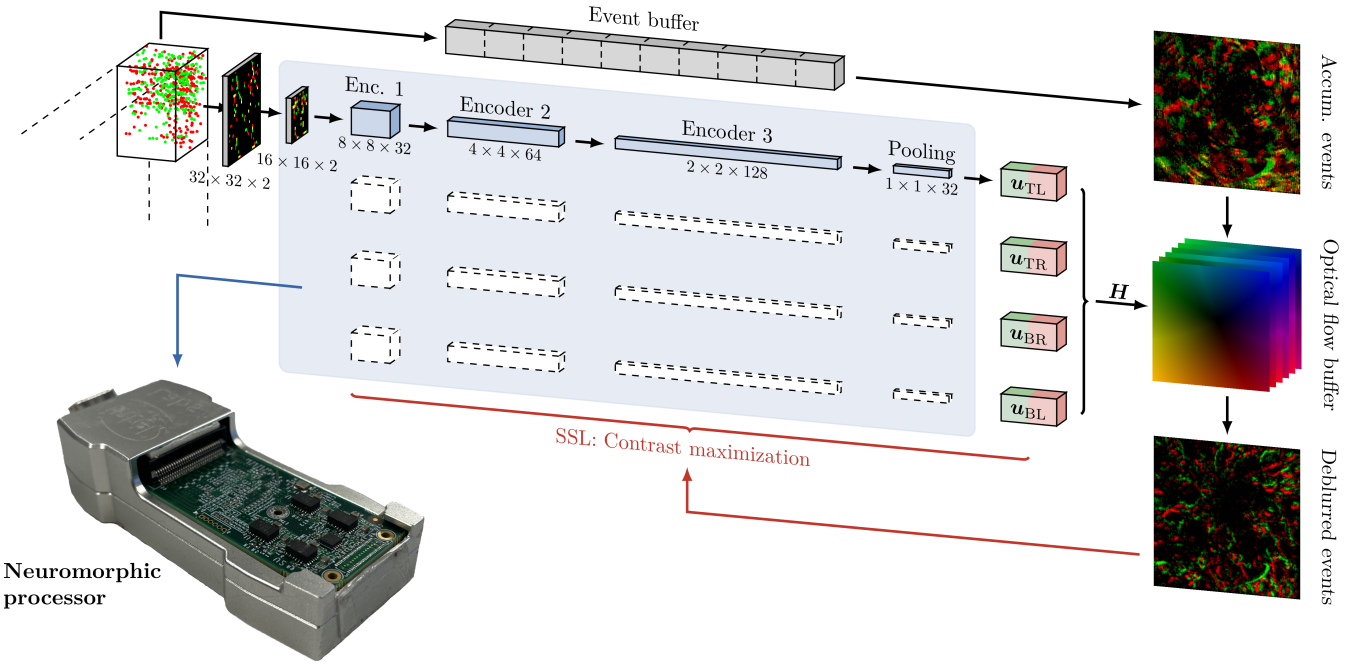

图 2. 脉冲视觉网络概览。 网络以约 200 Hz 的频率运行，累积事件（每个角最多 90 个事件），然后将其输入由三个编码器（卷积核大小为 3x3，步幅为 2）和一个脉冲池化层组成的视觉网络。脉冲被解码为两个浮点数，表示该角的光流。该网络复制到另外三个角，从而得到四个角的光流向量。在训练期间，这些向量用于单应性变换，以推导稠密光流，随后将其用于自监督损失。在真实世界飞行测试期间，整个网络都运行在神经形态处理器上。

积多个时间步的事件和光流元组，通过对比度最大化获得稳健的自监督信号；只有在处理完 25 毫秒的事件数据后，才计算去模糊损失函数，并通过网络进行一次反向传播（使用时间反向传播）。为了解决神经元脉冲函数不可微的问题，我们使用替代梯度 [34]。

另一方面，网络的控制部分由从四个角点的运动到推力和姿态控制指令的线性映射构成，并使用遗传算法在无人机模拟器中进行训练。图 3 对此进行了概述。为避免在仿真中加入基于事件的视觉流水线，我们使用模拟无人机的真实状态，通过连续单应变换生成预期的角点光流 [61]，，并据此构建缩放速度和偏航角速度估计值，作为相机自运动的视觉可观测量 [51, 62]。线性控制映射的输入包括这些视觉可观测量、绝对横滚角和俯仰角（来自无人机的惯性测量单元），以及视觉可观测量的期望设定点。随后，控制器的输出（即期望的总推力、俯仰角和横滚角，以及偏航角速度）会施加到无人机模型上，以控制无人机。在进化过程中，根据评估期间累积的视觉可观测量误差来确定控制器的适应度。我们在一组（重复的）设定点上评估种群中的每个个体，这些设定点代表水平飞行和垂直飞行；通过随机变异生成后代，并选出表现最佳的个体进入下一世代。训练完成的控制器无需重新训练，即可直接移植到真实机器人上。

结果

由于处理流水线的视觉部分与控制部分彼此分离，我们可以分别评估它们的性能。将视觉部分估计的角点光流与动作捕捉系统获取的真实值进行比较，同时在仿真中评估控制部分。将视觉与控制连接起来后，我们通过真实世界飞行测试展示完整神经形态视觉到控制流水线的性能。为进一步说明基于视觉的状态估计的鲁棒性，我们在设定点不断变化的情况下进行真实世界测试，并在各种光照条件下开展测试。最后，我们将能耗与可能采用的机载图形处理器解决方案进行比较。

鲁棒的基于视觉的状态估计

为了避免模拟事件相机时出现现实差距问题，我们使用同一平台（即无人机和朝下安装的事件相机）在同一室内环境（即静态、平面且照明恒定）中录制的真实事件序列，来训练和评估处理流水线中的视觉部分。该数据集包含约 40 分钟的事件数据，我们将其划分为 25 分钟用于训练、15 分钟用于评估，其运动统计数据如图 4A 所示。除视觉数据外，数据集还以 180 Hz 的频率提供无人机随时间变化的真实位姿（即位置和姿态），且这些数据仅用于评估。图 4A 展示了真实位姿示例；利用相机标定可将其转换为稠密光流，图中还同时展示了室内环境的地面纹理。请注意，本数据集随本文的补充材料一并提供。

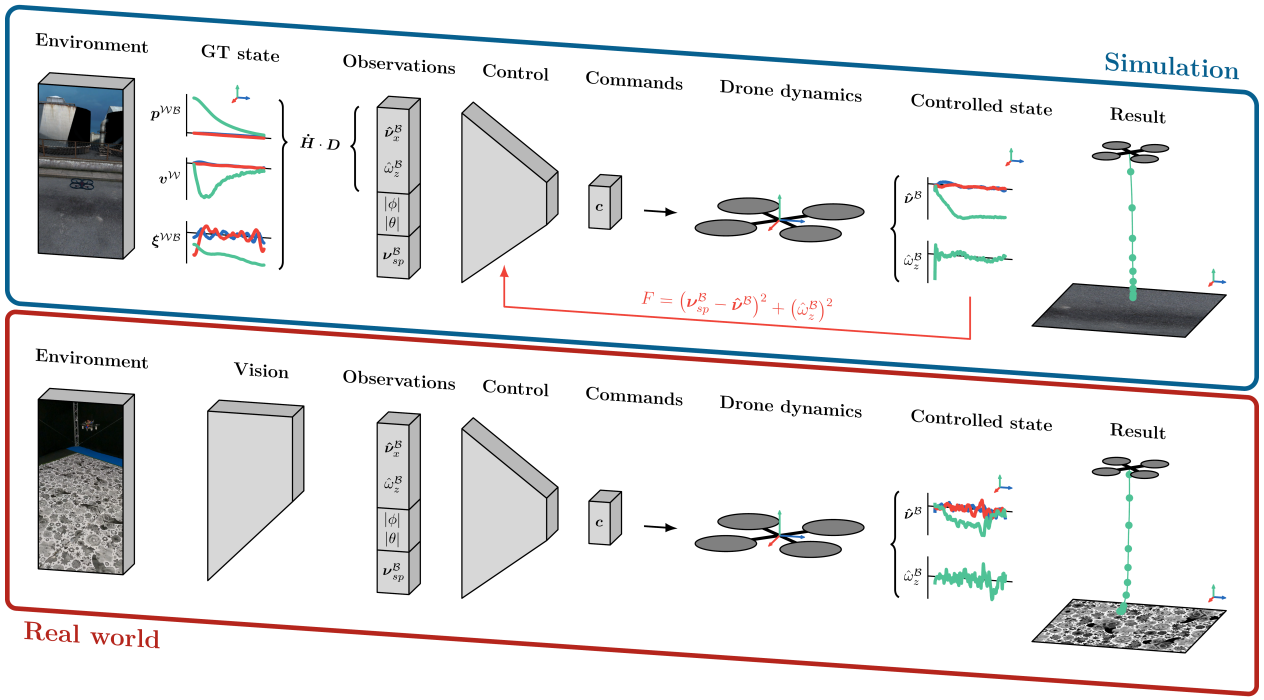

图 3. 仿真和真实世界测试的控制流程概览。 在仿真训练期间，我们通过连续单应性变换从真实值构建视觉可观测量。控制解码将这些可观测量与横滚、俯仰和设定点结合起来，输出用于控制无人机动力学的指令。我们使用基于进化算法的方式训练控制器，适应度信号用于量化控制器跟踪水平飞行和垂直飞行设定点的能力。在真实环境中，我们接收来自视觉网络的角点光流，通过一次矩阵乘法将其转换为视觉可观测量和控制指令，从而向自动驾驶仪发送推力和姿态等低级控制指令。

我们使用 [43] 中的自监督对比度最大化框架，以及一种量化感知训练流程来训练视觉脉冲神经网络；该流程会模拟目标神经形态硬件中的神经元和突触模型。完成训练后，我们使用运动程度各异的序列，评估脉冲网络在平面事件光流估计任务上的性能。定性结果见图 4B，其中将估计得到的视觉可观测量（由图像角点处估计的光流向量构建）与对应的真实值进行比较。这些结果证实了我们方法的有效性。尽管所提出的方案存在架构限制（例如基于脉冲的处理、有限的视场、仅支持自循环，以及权重和状态量化），且训练期间无法获取真实值信息，但它仍能生成光流估计，准确捕捉输入事件流中编码的运动，也就是相机的自运动。对于图中所示的快速运动序列，这一表现尤其突出：在序列末尾，相机以约 200 度/秒的速度旋转。

在图 4C 中，我们展示了视觉脉冲神经网络处理快速序列（见图 4B）时，在图像空间左上角的内部脉冲活动，以及解码得到的光流矢量。这些定性结果揭示了所提出架构所执行的处理类型：它基于脉冲，因此具有稀疏性和异步性。值得注意的是，尽管输入序列中的运动很快，脉冲神经网络的所有层的激活水平均保持在可用神经元的 50% 以下。请注意，该网络并未经过明确训练以促进稀疏激活。此外，我们可以区分出活动水平与输入活动高度相关的层（即编码器 1 和池化），以及依赖显式循环连接来维持活动水平、使其相对独立于输入统计特性的层（即编码器 2 和编码器 3）。

与输入活动（即编码器 1 和池化）相关，而其他层则依靠其显式循环连接来维持活动水平，使其相对独立于输入统计特性（即编码器 2 和编码器 3）。

在表1中，我们基于平均端点误差（EPE）（即预测光流向量与真实值光流向量之间的欧几里得距离），对我们的解决方案与其他类似的循环架构进行了定量比较。这项评估不仅展示了我们脉冲网络的性能，也评估了处理流水线中各项机制的影响，这些机制旨在实现可在 Loihi 上以 200 Hz 目标频率部署的解决方案。结果可得出以下几点结论。首先，人工神经网络的表现大幅优于其脉冲变体，而在所测试的显式循环方式中，自循环效果最弱。其次，与一次处理整个图像空间相比，为每个图像角点分别部署一个架构有利于我们的架构，而对基线模型只有轻微的不利影响。将一次可处理的事件数限制为每个角点 90 个，也有助于所评估的脉冲神经网络，因为这有助于降低内部活动水平。最后，采用 Loihi 特定的权重和状态量化会导致我们的架构误差增加。

通过视觉观测量进行控制：从仿真到真实环境

与视觉部分分开，我们训练并评估处理流水线中的控制部分。这是一个线性映射，将视觉观测量（即缩放速度估计值$\hat{\nu}^{\mathcal{B}}$和偏航角速度估计值 $\hat{\omega}^{\mathcal{B}_z}$）、绝对横滚角和俯仰角，以及视觉观测量设定值映射为推力和姿态指令。这些映射构成的种群——

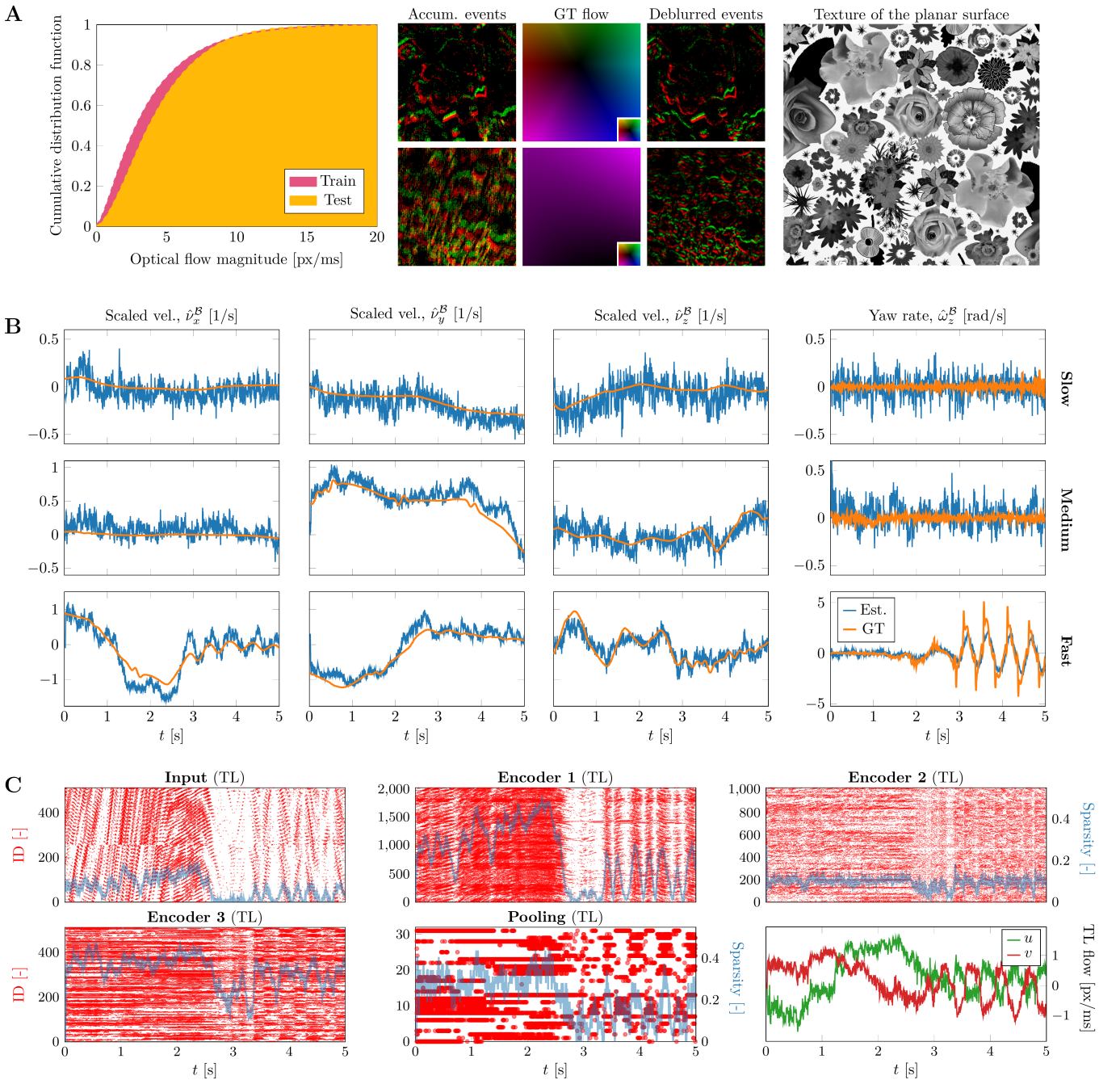

图4. 基于视觉的状态估计结果概览。（A）平面光流训练数据集的定性分析。利用真实光流可对累积且模糊的事件进行去模糊处理。事件由地面上重复的纹理产生。训练数据集和测试数据集中的光流幅值分布均匀。**（B）**对比运动速度不同（慢、中、快）的序列中估计得到的视觉可观测量与真实视觉可观测量。**（C）**快速运动序列中由左上角事件产生的网络活动。

在仿真中，对包含 16 个视觉观测量设定值的集合进行演化。每个缩放速度设定点 $\boldsymbol{\nu}^{\mathcal{B}}{}_{sp}$ 最多只有一个非零元素 $\in \{\pm 0.2, \pm 0.5, \pm 1.0\}$ 1/s。换言之，它们代表悬停、以三种速度着陆的垂直飞行（不包括上升飞行），以及以四个方向、三种速度进行的水平飞行。除非另有说明，偏航角速度的设定点 $\omega^{\mathcal{B}}{}_{z,=}\ 0$。图 5 的第一列展示了演化得到的线性网络控制器在仿真中的表现——就估计的缩放速度 $\hat{\boldsymbol{\nu}}^{\mathcal{B}}$（图 5A）以及所有设定点对应的世界位置 $\boldsymbol{p}^{\mathcal{WB}}$ 随时间的变化（图 5B ）而言。控制器在所有情况下都能达到设定点，并能够使非飞行方向的缩放速度保持接近零。特别是对于 $\nu^{\mathcal{B}}_{*},_{sp} = \pm 1.0$ 1/s，会出现超调，但鉴于这是不含任何微分控制的线性映射，这种情况是可以预料的。

我们通过将该控制器部署到现实世界中，并用视觉网络估计的可观测量替换真实值视觉5 ，得到了图

表 1. 不同架构之间的定量比较。右下角的数值以粗体标出，表示最终采用的架构。按行排列的架构选择和按列排列的设计决策会影响平均端点误差（端点误差）的测试性能。

	完整图像	+2x 下采样。	+角落裁剪	+限制事件	+Loihi量化。
卷积门控循环单元人工神经网络	+2.45%	Best, 0.056 端点误差	+0.94%	+2.94%	-
卷积循环神经网络脉冲神经网络	+42.48%	+41.96%	+42.25%	+25.48%	+25.32%
自循环神经网络脉冲神经网络（本文方法）	+94.16%	+76.45%	+48.00%	+38.74%	**+50.04%**

可观测量。观察不同设定点对应的缩放后的速度图，我们发现，设定点越高、飞行速度越快，数据中的噪声就越少；三维位置图也显示了这一点。这是因为基于视觉的状态估计的信噪比会随着运动幅度增大而提高（运动很小时，大多数事件都源于噪声，如图4所示）。此外，无人机的惯性在较高速度下也能提供一定的稳定性。总体而言，结果表明，全神经形态视觉到控制流水线已成功部署。然而，除若干设定点（例如着陆、$\nu^{\mathcal{B}}_{\{x,\ y\},\ sp} = \pm 0.2$ 1/s）外，控制器无法达到期望的设定点：稳态误差似乎与设定点的幅值成正比。这可能是因为，尽管控制器采用线性映射，姿态角与由此产生的前向/侧向速度之间的关系却因阻力而呈非线性。在训练期间向网络提供绝对姿态输入并模拟阻力（如 [63]中所示），仍不足以弥补这一差异。此外，控制器的训练所依据的模拟无人机动力学（机体特性、电机动力学）可能与进行飞行测试的真实无人机存在差异，尽管我们已将控制输出抽象为姿态指令。最后，阻力模型不够精确也可能造成误差（在本例中，现实环境中的阻力似乎比仿真中的更大）。

图 5的第三列展示了将手动调参的比例积分（PI）控制器与基于视觉的状态估计相连后获得的结果。我们将其与线性网络控制器进行比较。观察所有方向和设定点可以发现，PI 控制器比网络控制器更快到达设定点。在水平飞行中，网络控制器完全无法到达设定点 $\nu^{\mathcal{B}}_{\{x,\ y\},\ sp} = \pm 1.0$ 1/s；在 ± 0.5 1/s 的情况下也只是勉强做到，这大概是线性控制的局限性所致。PI 控制器则没有这个问题，因为它可以逐步增大控制指令，以消除稳态误差。在垂直飞行中，当 $\nu^{\mathcal{B}}_{z,\ sp} = -1.0$ 1/s 时，网络控制器和 PI 控制器都有相当明显的超调。不过，这种现象之所以如此明显，是因为无人机以这样的速度从这样的高度（即 2.5 m）飞行时，几乎刚到达设定点就触及地面，因此没有多少时间来补偿超调（观察 PI 控制器在 $\nu^{\mathcal{B}}_{z,\ sp} = -0.5$ 1/s 时的表现；那里的超调程度相近，但很快就得到了修正）。适当降低增益或加入微分项或许会有所帮助。

迪斯科、黑暗、方形和飞盘：多功能性与鲁棒性的示例

我们可以将基于视觉的状态估计与 PI 控制器结合起来，通过其他多项测试展示前者的多功能性和鲁棒性，同时无需将这些测试纳入线性网络控制器的训练中。图 6A 展示了这些测试。顶行显示用户交替设置 X 和 Y 方向上的不同缩放速度设定点（同时保持偏航不变），以使无人机沿正方形轨迹飞行。控制器能够相当迅速地达到期望的设定点，从而实现较为急促的转弯，但偏航方向存在明显漂移，导致第二个正方形相较于第一个略有旋转。

图 6A 的底行展示了一项实验：无人机必须沿直线飞行，同时像飞盘一样绕 Z 轴旋转。无人机接收到非零偏航角速度设定值 $\omega^{\mathcal{B}}_{z,sp} = 0.2$ rad/s。若同时采用 $\nu^{\mathcal{B}}_{y,sp} = 0.5$ 1/s 的设定值，无人机将沿圆形轨迹飞行。为避免这种情况并实现类似飞盘的旋转效果，我们根据偏航角旋转 $\nu^{\mathcal{B}}_{y,sp}$。第一幅和最后一幅图表明，这种方法行之有效：尽管 X 和 Z 方向存在一些漂移，设定值仍得到了良好跟踪，3D 位置轨迹也相当笔直。第二幅图显示，期望的偏航角速度得到了良好跟踪，偏航角则持续增大。

图 6B 展示了在各种光照条件下着陆的情况，发散量为 $\nu^{\mathcal{B}}_{z,sp} = -0.5$ 1/s（通过勒克斯测量量化）。左上角区域的事件显示在底行中。明亮与较暗的设置看起来相似，但闪烁的灯光会导致事件数量大幅增加，而最暗的设置几乎不会产生事件。从中间一行图表可以看出，尽管光照条件颇具挑战性，控制器仍能较好地跟踪设定点（黑色虚线），估算的缩放速度也接近其真实值。只有最暗的设置会给状态估计带来实际问题：在这种情况下，估算的缩放速度 $\hat{\boldsymbol{\nu}}^{\mathcal{B}}$ 与缩放速度的真实值 $\boldsymbol{\nu}^{\mathcal{B}}$ 相差过大，无法成功着陆。

在神经形态硬件上提升推理速度并降低能耗

表2 比较了 Loihi 神经形态处理器与 NVIDIA Jetson Nano 在运行视觉网络时的功率/能量和运行时间。测试所用序列包含不同程度的运动，因此输入事件密度也各不相同。SNN 在 Loihi 上以硬件方式运行，在 Jetson Nano 上则以软件（PyTorch）方式运行。Loihi 的测试在 Nahuku 板上进行，该板包含 32 个 Loihi 芯片。我们尽可能确认，在 Nahuku 上使用两颗芯片具有代表性，可模拟 Kapoho Bay（至少在执行时间方面）；Kapoho Bay 是无人机上使用的双芯片外形规格。尽管如此，这两项基准测试都无法完全代表现实环境中的测试：基准测试使用已加载到内存中的数据，因此只衡量网络的处理过程，不涉及输入/输出和预处理造成的任何瓶颈或影响；而飞行测试则涉及对流入并以在线方式处理的事件数据进行流式传输

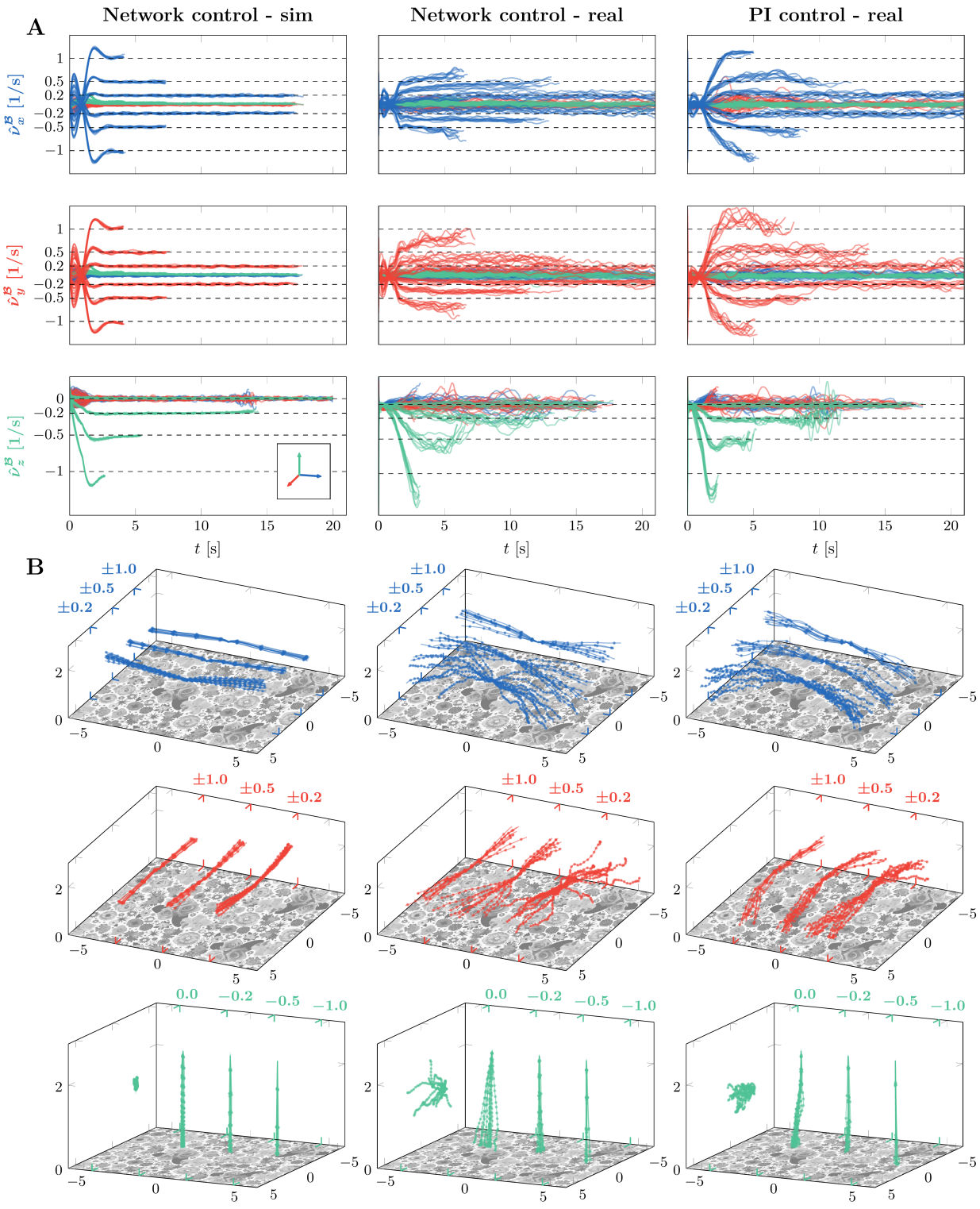

图5。 仿真和真实世界飞行测试结果的比较。 **（A）**线性网络控制器在仿真和真实环境中，以及手动调校的比例-积分（PI）控制器在真实环境中，针对三个场景下三条轴上的16个不同设定点所估算的缩放速度。 **（B）**相同飞行测试中的三维世界位置轨迹，为了在单张图中容纳更多曲线，对轨迹进行了偏移。 **（B）**中的每个立方体都与 **（A）**中对应位置的曲线相匹配。

。这反映在表 2 中的 inLoihi 执行频率上，这些频率远高于飞行测试期间达到的 200 次推理/秒。

由于 Jetson Nano 不提供静态和动态功耗分量，我们比较空闲功耗与运行功耗之间的差值，并据此计算每次推理能耗。

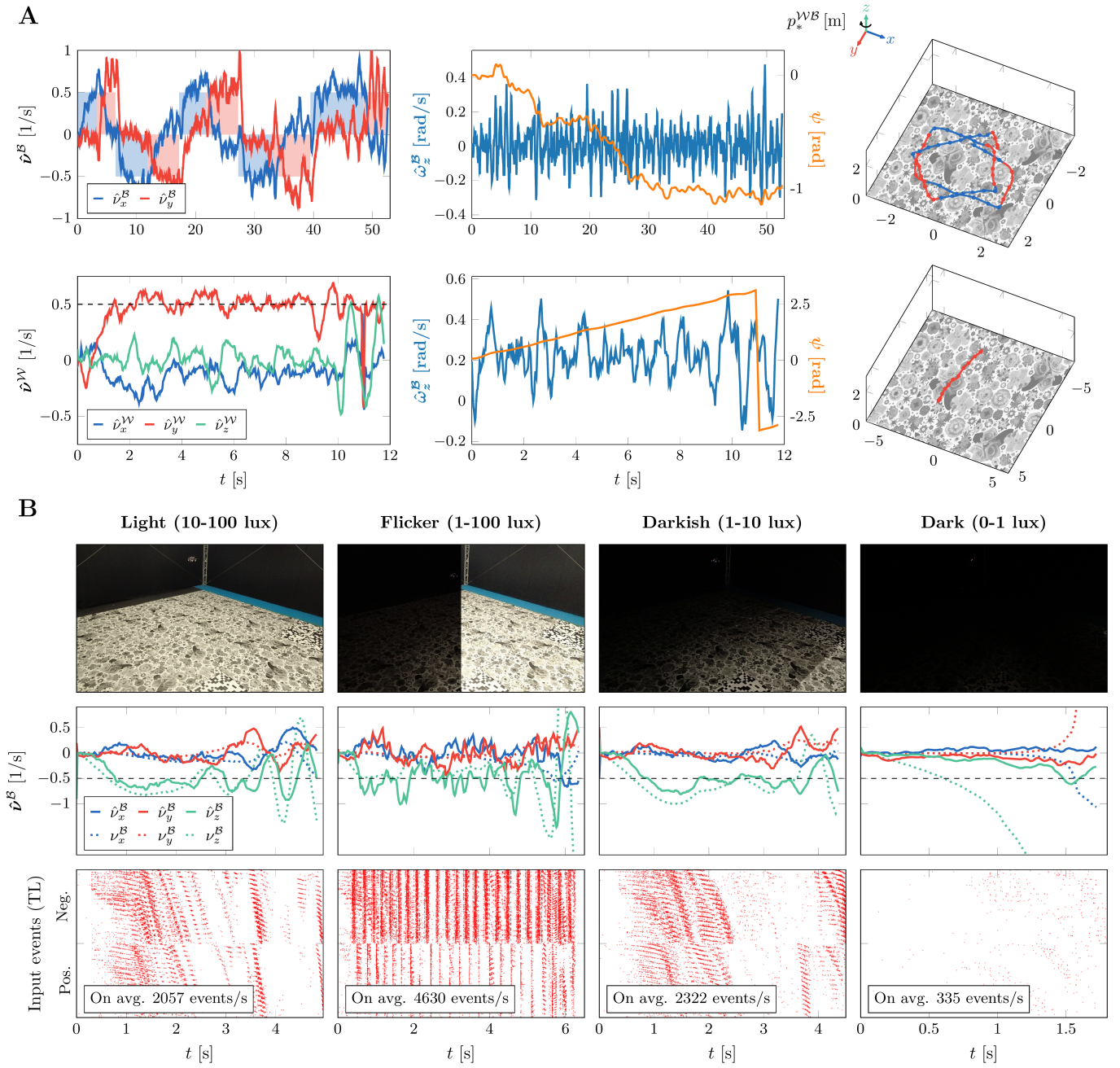

图 6. 采用视觉网络和比例-积分（PI）控制器的结果。(A)上排：交替设置 X 轴和 Y 轴设定点，以沿正方形轨迹飞行。下排：将缩放速度设定点按偏航角旋转，可使无人机在沿直线飞行的同时绕其 Z 轴旋转。**(B)** 不同光照条件下的着陆实验。虽然闪烁的灯光会产生更多事件，但只有在光线暗到几乎没有事件时，视觉可观测量估计（因而控制）才会发散。

根据序列的不同，Loihi 的表现比 Jetson Nano 高出三到四个数量级，执行频率也提升了一个到两个数量级。此外，随着事件稀疏度增加（从快速运动序列到慢速运动序列），Loihi 的执行频率随之提高，这体现了神经形态处理的优势。请注意，像 Jetson Nano 这样的图形处理器并未针对脉冲神经网络仿真进行优化；若将多个事件时间窗口作为输入，运行前馈人工神经网络或许能实现更高效的方案。

讨论与结论

我们提出了首个用于控制自由飞行无人机的全神经形态视觉到控制流水线。具体而言，我们训练了一个脉冲神经网络，该网络接收高维原始事件相机数据，并生成低级控制指令。真实环境中的实验表明，仿真到现实迁移成功：无人机能够准确跟踪各种自运动设定值，执行悬停、着陆和横向机动，即使在偏航角速度恒定的情况下也不例外。

我们的研究证实了完全神经形态

表2. 三种序列（慢速、中速和快速）下各类设备的近似能量和功率特性。 平均而言，慢速序列有 28.6 个事件/次推理，中速序列有 106.9 个事件/次推理，快速序列有 186.6 个事件/次推理。功率差是空闲功率与运行（总）功率之差，用于计算每次推理的能量。动态功耗是开关和短路所需的功率，而静态功耗源于漏电；两者之和也等于运行（总）功率。Nahuku 是一块搭载 32 个 Loihi 芯片的板卡（Kapoho Bay 搭载 2 个）。表中还包括一种 Nahuku 配置，称为“空配置”：不发送脉冲，只分配芯片和核心（不分配突触）。Jetson Nano 有低功耗（5W）和高功耗（10W）两种模式。

设备	Seq.	静态 [W]	动态 [W]	空闲 [W]	运行功率 [W]	功率增量 [W]	次推理/秒	微焦耳/次推理
Nahuku 32（空）	any	0.861	0.040	0.897	0.901	0.004	60496	0.071
Nahuku 32	slow	0.899	0.048	0.935	0.947	0.012	1637	7.165
	中等	0.900	0.044	0.936	0.945	0.008	411	20.602
	fast	0.902	0.043	0.938	0.946	0.007	274	27.207
Jetson Nano（5W）	slow	-	-	1.053	2.229	1.177	14	86111.152
	中等	-	-	1.027	2.245	1.218	14	85576.977
	fast	-	-	1.030	2.238	1.208	14	86187.999
Jetson Nano（10W）	slow	-	-	1.043	2.974	1.931	26	75246.618
	中等	-	-	1.058	2.981	1.923	26	75347.505
	fast	-	-	1.044	2.991	1.947	25	76522.389

视觉到控制处理流水线的潜力：该系统可在机载条件下以 200 Hz 的执行频率运行，每次网络推理仅消耗 27 μJ。不过，要充分发挥此类处理流水线的系统优势，仍面临重要障碍，例如将其嵌入极轻型（如 <30 g）无人机中。

要充分发挥其潜力，无人机的整个感知、处理和执行硬件都应采用神经形态设计，从加速度计传感器到处理器和电机皆如此。目前尚无此类硬件，因此我们将研究范围限定于视觉到控制流水线，终点为推力和姿态指令。对于神经形态处理器而言，最大的进展或许来自输入/输出带宽和接口选项的改进。目前的处理器无法通过 AER 直接连接事件相机；在我们先进的用例中，我们也达到了神经形态处理器在所需高执行频率下可发送和接收脉冲数量的上限。这也是我们将研究范围限定于线性网络控制器的原因：为编码设定点和姿态输入所需增加的输入脉冲，会大幅降低处理流水线的执行频率。最终，从数字神经形态处理器转向模拟硬件，或许能进一步提高效率，但这将带来更大的开发和部署挑战。

尽管存在上述局限，当前研究仍朝着面向无人机的神经形态感知与处理迈出了重要一步。研究结果令人鼓舞，因为它们表明，神经形态感知与处理有望让小型自主机器人也能运用深度神经网络。随着时间推移，这或许能使它们逐渐具备接近飞行昆虫等动物的敏捷性、多功能性和鲁棒性。

材料与方法

在此，我们介绍所提出的全神经形态视觉到控制处理流水线的主要组件，首先说明我们的 SNN 神经元模型，以及如何以自监督方式使用真实事件相机数据进行训练。接下来，我们介绍如何将基于视觉的状态估计用于导航，以及如何在此基础上训练控制器。最后，我们讨论真实世界测试、硬件以及所进行的能耗基准测试。

神经形态状态估计：脉冲发放、序列处理

在本研究中，我们采用了一种基于电流型泄漏积分发放（CUBA-LIF）神经元的脉冲神经元模型，其膜电位U 和突触输入电流 I 在时间步长 t 时可表示为：

$$U_i^t = \tau_U(1 - S_i^{t-1})U_i^{t-1} + I_i^t \quad \textbf{(1)}$$

$$I_i^t = \tau_I I_i^{t-1} + \sum_j w_{ij}^{\mathrm{ff}} S_j^t + w_{ii}^{\mathrm{rec}} S_i^{t-1} \quad \textbf{(2)}$$

其中，j 和i 分别表示层内的突触前（输入）神经元和突触后（输出）神经元，$S \in \{0,1\}$ 表示神经元脉冲，而w^{ff}和w^{rec} 分别表示前馈连接和自循环连接（如有）。该神经元模型的两个内部状态变量的衰减（或泄漏值）是通过学习得到的，分别记为 τ_U和 τ_I。如果膜电位超过阈值 θ，神经元就会发放输出脉冲；该阈值同样通过学习得到。脉冲发放会触发膜电位的硬重置。请注意，在本研究中，同一层内的所有神经元共享相同的衰减值和发放阈值。

Loihi 神经形态处理器上的神经元也遵循 CUBA-LIF 模型 [39]，不过，要准确仿真这些片上神经元，必须考虑若干因素。首先，这两个状态变量都在整数域中进行量化。因此，与这些变量相关的参数也采用相同方式量化：$w \in [-256 .. 256 - \Delta w]$，其中$\Delta w$ 为突触权重的量化步长，$\tau_{\{U、I\}} \in [0 .. 4096]$ 为衰减参数，$\theta \in [0 .. 131071]$ 为阈值。我们在 $\Delta w = 8$（6 位权重）的神经网络仿真和训练中采用这一量化方案。

其次，为了模拟处理器在更新神经元状态时执行的算术左移（位移）操作，我们在应用衰减后进行向零舍入。考虑到这些因素，仿真得到的脉冲神经元与片上脉冲神经元的匹配度达到100%。我们采用量化感知训练（量化前向传播、浮点反向传播），以尽量减少脉冲神经网络部署到 Loihi 时的性能损失。

作为脉冲函数的替代梯度σ，我们选用反正切函数的导数$\sigma'(x) = \mathrm{aTan}' = 1/(1+\gamma x^2)$ [37]，其中 $\gamma = 10$ 为替代宽度，$x = u - \theta$。

神经形态状态估计：平面单应性

假设$\boldsymbol{x} = [x、y、1]^T$ 以及$\boldsymbol{x}' = [x'、y'、1]^T$ 是来自平面场景的两个未经畸变的对应点，以齐次坐标表示，并由针孔相机在不同时间点拍摄。平面单应变换是一种线性射影变换，它将$\boldsymbol{x} \leftrightarrow \boldsymbol{x}'$ 映射为：

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \boldsymbol{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}; \quad \text{with } \boldsymbol{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \tag{3}$$

其中$\boldsymbol{H}$ 是一个3×3非奇异矩阵，以下称为单应矩阵，其具有八个自由度，并且定义到一个尺度因子 λ。

根据公式 3，我们可以构造 $\boldsymbol{A}_k\boldsymbol{h} = \boldsymbol{b}_k$，这是针对第 k 个点对应的欠定线性方程组，其中：

$$\boldsymbol{A}_k = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y \end{bmatrix} \tag{4}$$

$$\boldsymbol{h} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{21} & h_{22} & h_{23} & h_{31} & h_{32} \end{bmatrix}^T \tag{5}$$

$$\boldsymbol{b}_k = \begin{bmatrix} x' & y' \end{bmatrix}^T \tag{6}$$

如图 2A 所示，我们的视觉网络会预测特定时间窗口内角点像素的位移。利用这些信息，我们可以通过 $\boldsymbol{h} = \boldsymbol{A}^{-1}\boldsymbol{b}$ 求解单应矩阵的各个分量，其中 $\boldsymbol{A}$ 和 $\boldsymbol{b}$ 分别由每个点对应 $\boldsymbol{A}_k$ 和 $\boldsymbol{b}_k$ 拼接而成 $\forall k \in \{\mathrm{TL,TR,BR,BL}\}$。这种方法称为单应性变换的四点参数化 [55]，，并已证明在机器人应用的事件相机文献中行之有效 [60, 64]。然而，据我们所知，这种方法从未用于经自监督学习训练的脉冲神经网络。

估计出单应矩阵后，我们可以按如下方式估计稠密（即逐像素）的光流图：

$$\boldsymbol{u}(\boldsymbol{x},\boldsymbol{H}) = \begin{bmatrix} u(\boldsymbol{x},\boldsymbol{H}) \\ v(\boldsymbol{x},\boldsymbol{H}) \end{bmatrix} = \boldsymbol{H} \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \tag{7}$$

它编码了时间窗口$\boldsymbol{H}$内像素$\boldsymbol{x}$ 的位移。

神经形态状态估计：自监督学习

为了以自监督方式训练我们的脉冲架构来估计四个角点像素的位移，我们采用对比度最大化框架进行运动补偿[56, 57]。假设光照恒定，运动边缘触发的事件之间的时空错位（即模糊）中编码了精确的光流信息。为提取这些信息，需要学会补偿这种运动（即对事件分区进行去模糊），方法是让事件在时空中移动。获得逐像素的光流估计$\boldsymbol{u}(\boldsymbol{x},\boldsymbol{H})$（根据公式 7）后，我们便可通过以下线性运动模型将事件传播至参考时间 t_{ref}：

$$\boldsymbol{x}_i' = \boldsymbol{x}_i + (t_{\mathrm{ref}} - t_i)\boldsymbol{u}(\boldsymbol{x}_i, \boldsymbol{H}) \tag{8}$$

聚合传播事件的结果称为时刻 t_{ref} 的扭曲事件图像（IWE）。

作为损失函数，我们采用 [43]基于扭曲事件图像 [18, 65]中逐像素、逐极性的平均时间戳所定义的聚焦目标函数的改写形式。该指标越低，事件去模糊效果越好，因此估计出的光流也越准确。我们通过双线性插值，为每种极性生成一幅逐像素平均时间戳图像 p'：

$$\begin{aligned} T_{p'}(\boldsymbol{x};\boldsymbol{u}|t_{\mathrm{ref}}) &= \frac{\sum_j \kappa(x - x_j')\kappa(y - y_j')t_j}{\sum_j \kappa(x - x_j')\kappa(y - y_j') + \epsilon} \\ \kappa(a) &= \max(0, 1 - |a|) \\ j = \{i \mid p_i = p'\}, &\quad p' \in \{+,-\}, \quad \epsilon \approx 0 \end{aligned} \tag{9}$$

按照 [43]，，我们首先用至少包含一个扭曲事件的像素数量，对扭曲过程产生的时间图像平方和进行缩放：

$$\mathcal{L}_{\mathrm{contrast}}(t_{\mathrm{ref}}) = \frac{\sum_{\boldsymbol{x}} T_+(\boldsymbol{x};\boldsymbol{u}|t_{\mathrm{ref}})^2 + T_-(\boldsymbol{x};\boldsymbol{u}|t_{\mathrm{ref}})^2}{\sum_{\boldsymbol{x}} [n(\boldsymbol{x}') > 0] + \epsilon} \tag{10}$$

其中，$n(\boldsymbol{x}')$ 表示扭曲事件图像中每个像素的事件计数。

与 [18, 20, 43]，中一样，我们分别以正向（$t_{\mathrm{ref}}^{\mathrm{fw}}$）和反向（$t_{\mathrm{ref}}^{\mathrm{bw}}$）执行扭曲过程，以避免反向传播期间出现时间缩放问题。用于训练基于事件的光流网络的总损失为：

$$\mathcal{L}_{\mathrm{contrast}} = \mathcal{L}_{\mathrm{contrast}}(t_{\mathrm{ref}}^{\mathrm{fw}}) + \mathcal{L}_{\mathrm{contrast}}(t_{\mathrm{ref}}^{\mathrm{bw}}) \tag{11}$$

$$\mathcal{L}_{\mathrm{flow}} = \mathcal{L}_{\mathrm{contrast}} + \lambda_{\mathcal{L}}\mathcal{L}_{\mathrm{smooth}} \tag{12}$$

其中，$\mathcal{L}_{\mathrm{smooth}}$ 是应用于后续逐角点光流估计的时间域 Charbonnier 平滑先验 [66]，而 $\lambda_{\mathcal{L}}$ 是用于平衡两项损失影响的标量。我们通过经验将该权重设为 $\lambda_{\mathcal{L}} = 0.1$。

如前文所述，[43]，输入事件分区中必须有足够的线性模糊，才能使该损失函数成为可靠的监督信号 [57, 67]。由于我们按顺序处理事件流，且每次前向传播只考虑少量事件，因此定义了所谓的训练分区$\boldsymbol{\varepsilon}_{k\to k+K}^{\mathrm{train}} = \{(\boldsymbol{\varepsilon}^{\mathrm{inp}}{}_i, \hat{\boldsymbol{u}}_i)\}_{i=k}^{K}$。这是一个缓冲区，每次前向传播都会将输入事件及其对应的光流估计存入其中。如图 2A 所示。在训练时，当缓冲区中累积了 5 个连续的事件-光流元组（即 25 毫秒的事件数据）后，我们使用缓冲区中的内容，通过时间反向传播进行一次反向传播；随后更新模型参数，将其状态从计算图中分离，并清空缓冲区。我们使用批次大小为 16，并采用 Adam 优化器训练至收敛 [68]，学习率为 1e-4。

从基于视觉的状态估计到控制

基于视觉的状态估计所得到的角点光流 $[\boldsymbol{u}_{\mathrm{TL}}, \boldsymbol{u}_{\mathrm{TR}}, \boldsymbol{u}_{\mathrm{BR}}, \boldsymbol{u}_{\mathrm{BL}}]^T \in \mathbb{R}^{8\times 1}$ 可用于控制无人机。具体而言，我们可以将角点光流转换为视觉可观测量估计 $\boldsymbol{\nu}^{\mathcal{C}} \in \mathbb{R}^{3\times 1}$，$\omega^{\mathcal{C}}$ [51]，$\mathcal{C}$ [62]，其中包括缩放速度 $\hat{}$ 和偏航角速度 $\hat{}_z$（均在相机坐标系中），如下所示：

$$\boldsymbol{u}_k = -\begin{bmatrix} \nu_x^{\mathcal{C}} \\ \nu_y^{\mathcal{C}} \end{bmatrix} + \nu_z^{\mathcal{C}} \boldsymbol{x}_k + \begin{bmatrix} \omega_z^{\mathcal{C}} \\ -\omega_z^{\mathcal{C}} \end{bmatrix} \circ \boldsymbol{x}_k \tag{13}$$

其中，$\circ$ 表示哈达玛积（逐元素乘积），而 $\boldsymbol{x}_k = [x, y]^T$ 表示视场角点处的世界点在像素阵列上的投影。此外，假设：1）场景静止且为平面；2）俯仰角和横滚角较小；3）光流已针对俯仰和横滚进行去旋转。对所有四个角点求逆该关系，即可通过最小二乘估计得到缩放速度 $\hat{\boldsymbol{\nu}}^{\mathcal{C}}$ 和偏航角速度 $\hat{}\omega^{\mathcal{C}}{}_z$，随后可将其转换到机体坐标系 $\mathcal{B}$。为进行控制，可让用户选择设定点 $\boldsymbol{\nu}^{\mathcal{B}}{}_{sp}$ 和 $\omega^{\mathcal{B}}{}_{z,sp}$，并使用经过训练或手动调节的控制器，尽量减小估计得到的视觉可观测量与其设定点之间的差异。

由于式 13是线性变换，因此可以与其他线性变换“合并”。这也适用于视觉脉冲神经网络中从脉冲到角点光流的解码，这意味着使用线性控制器时，我们可以通过一个线性变换直接将脉冲映射为控制指令。同样，我们也可以利用这一思路连接分别训练的脉冲神经网络，合并它们的线性解码与编码。如果两者均在神经形态硬件上实现，就无需进行片外传输。图 7 展示了这些概念。

在仿真中训练控制

我们通过线性变换视觉观测估计值 $\hat{\boldsymbol{\nu}}^{\mathcal{B}} \in \mathbb{R}^{3\times 1}$ 和 $\hat{}\omega^{\mathcal{B}}{}_z$、无人机的绝对横滚角 $|\phi|$ 和俯仰角 $|\theta|$ 以及缩放速度设定点 $\boldsymbol{\nu}^{\mathcal{B}}{}_{sp} \in \mathbb{R}^{3\times 1}$，得到控制指令 $\boldsymbol{c} \in \mathbb{R}^{4\times 1}$，其中包括机体坐标系 $\mathcal{B}$下向上的推力（相对于悬停推力的偏移量，并按质量 $\bar{}$归一化）$f_{0,c}$、横滚角 ϕ_c 和俯仰角 θ_c，以及偏航角速度 $\omega^{\mathcal{B}}{}_{z,c}$，以达到特定的缩放速度设定点 $\boldsymbol{\nu}^{\mathcal{B}}{}_{sp}$ 和偏航角速度设定点 $\omega^{\mathcal{B}}{}_{z,sp}$（始终为 0）。

由于准确模拟基于事件的相机输入代价高昂（这需要帧间亚像素位移，因此快速运动时需要较高的帧率），控制部分与视觉部分分开训练。仿真采用经过修改的无人机模拟器 Flightmare [69]。为模拟基于视觉的状态估计网络的输出，我们首先根据无人机的状态计算真实值连续单应性 [61, 70]：

$$\dot{\boldsymbol{H}} = \boldsymbol{K}\left([\boldsymbol{\omega}^{\mathcal{C}}]_\times + \frac{1}{p_z^{\mathcal{WC}}} \boldsymbol{v}^{\mathcal{C}} (\boldsymbol{e}_{-z}^{\mathcal{W}})^T\right) \boldsymbol{K}^{-1} \tag{14}$$

其中 $\dot{}\boldsymbol{H}$ 是连续单应性，$\boldsymbol{K}$ 是相机内参矩阵，$[\boldsymbol{\omega}^{\mathcal{C}}]_\times \in \mathbb{R}^{3\times 3}$ 是表示无穷小旋转的反对称矩阵，$p^{\mathcal{WC}}{}_z$ 是位置向量从世界坐标系 $\mathcal{W}$ 到相机坐标系 $\mathcal{C}$ 的 Z 分量（表示地平面到相机的垂直距离），$\boldsymbol{v}^{\mathcal{C}}$ 是相机的速度，而 $\boldsymbol{e}^{\mathcal{W}}{}_z$ 是世界坐标系中指向负 Z 方向的单位向量。为了获得相机坐标系中的角速度和速度，我们使用

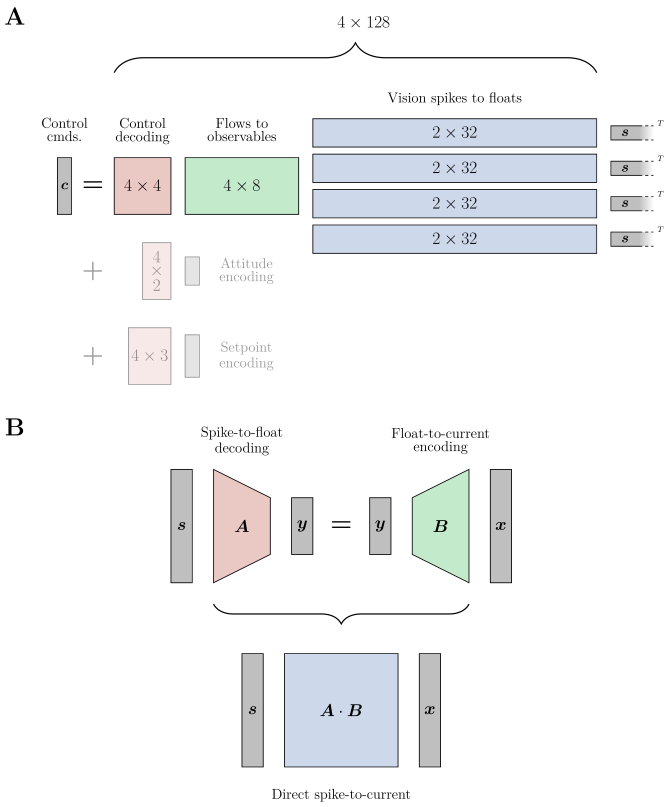

图 7. 合并线性变换。（A）我们通过将相关线性变换矩阵相乘，直接通过一次线性解码，将视觉网络的输出脉冲 $\boldsymbol{s}$ 转换为控制指令 $\boldsymbol{c}$。**(B)** 同一原理也可用于以脉冲发放方式连接两个分别训练的脉冲网络，将脉冲 $\boldsymbol{s}$ 转换为电流 $\boldsymbol{c}$，适用于神经形态硬件。

相机外参，包括旋转 $\boldsymbol{R}^{\mathcal{CB}}$ 和平移 $\boldsymbol{T}^{\mathcal{CB}}$：

$$\boldsymbol{\omega}^{\mathcal{C}} = \boldsymbol{R}^{\mathcal{CB}} \boldsymbol{\omega}^{\mathcal{B}} \tag{15}$$

$$\boldsymbol{v}^{\mathcal{C}} = \boldsymbol{R}^{\mathcal{CB}} \left(\boldsymbol{v}^{\mathcal{B}} + [\boldsymbol{\omega}^{\mathcal{B}}]_\times \boldsymbol{T}^{\mathcal{CB}}\right) \tag{16}$$

接下来，我们使用连续单应性来获得四个角点的光流 [61, 70]：

$$\boldsymbol{u}_k = -\left(\mathbf{1} - \boldsymbol{x}_k (\boldsymbol{e}_{-z}^{\mathcal{W}})^T\right) \dot{\boldsymbol{H}} \boldsymbol{x}_k \tag{17}$$

其中 $\mathbf{1}$ 是单位矩阵，而 $\boldsymbol{x_k} = [x, y, 1]^T$ 表示视场角点处世界点在像素阵列上的齐次坐标投影（因此，$\boldsymbol{x}_{BL} = [0, 180, 1]^T$ 和 $\boldsymbol{x}_{TR} = [180, 0, 1]^T$，注意这与公式 13的区别）。我们向光流 $\mathcal{N}(0, 0.025)$ 添加噪声 $\boldsymbol{u}_k$（基于对视觉脉冲神经网络的特性分析）。随后使用公式 13 将角点光流转换为相机坐标系中的视觉可观测量，再将其变换回机体坐标系以进行控制。

我们采用仅含突变的遗传算法，种群规模为 100，以进化线性控制器矩阵的权重 $\in \mathbb{R}^{4\times 9}$；其初始值为 $\mathcal{U}(-0.1, 0.1)$。具体而言，我们通过对每个亲代的所有参数添加从 $\mathcal{N}(0, 0.001)$ 中抽取的突变来生成后代，然后评估亲代和后代的适应度。下一代由表现最好的 100 个个体组成，我们重复这一过程，直到

收敛（约 25k 个世代）。我们使用 Flightmare，在各种视觉可观测量设定点下评估适应度：每个个体都在一组包含 16 个设定点的集合上进行评估，每个缩放速度设定点 $\boldsymbol{v}^{\mathcal{B}}{}_{sp}$ 至多有一个非零分量 $\in \{\pm 0.2, \pm 0.5, \pm 1.0\}$ 1/s；跳过 Z 方向的正向设定点，并包含悬停。所有情况下，偏航角速度设定值均设为 $\omega^{\mathcal{B}}{}_{z,sp} = 0$。每个设定点重复十次，因此每个个体总计进行 160 次评估。适应度 F 计算如下：

$$F = \frac{1}{N_{\text{eval}}} \sum_{i \in N_{\text{eval}}} \sum_{j \in N_{\text{steps}}} \boldsymbol{w} \cdot \left(\boldsymbol{v}^{\mathcal{B}}_{sp,i} - \begin{bmatrix} \hat{v}^{\mathcal{B}}_x \\ \hat{v}^{\mathcal{B}}_y \\ v^{\mathcal{W}}_z \end{bmatrix}_j \right)^2 + \left(\hat{\omega}^{\mathcal{B}}_z \right)^2 \quad \textbf{(18)}$$

其中，$N_{\text{eval}} = 160$ 是评估次数，$N_{\text{steps}} = 1000$ 是每个回合的步数，$\boldsymbol{w} = [1, 1, w_z]^T$ 是一个用于对不同轴的适应度进行加权的向量，对于满足 $v^{\mathcal{B}}{}_{z,sp} = 0$ 的设定点，我们设定$w_z = 10$。请注意，对于 Z 方向，我们使用世界坐标系中的真实缩放速度 $v^{\mathcal{W}}{}_z$，而不是机体坐标系中的速度，因为当无人机以与其姿态相同的倾角上升或下降时，后者为零，因此不会受到惩罚，导致额外的垂直漂移。此外，如果智能体在回合结束前越界或坠毁，它将被重置，且不会受到任何额外的适应度惩罚。

我们采用域随机化 [71] 来获得更稳健的控制器并缩小现实差距：在十次重复中的每一次，我们都会在控制层接收到的绝对俯仰和横滚角上添加一个随机恒定偏置 $\mathcal{U}(-0.001, 0.001)$ 弧度。为保证公平，该偏置在整个种群中保持一致。此外，对于每个个体的 160 次评估，我们会随机改变初始位置 $\boldsymbol{p}^{\mathcal{WB}} = [0, 0, 2]^T + \mathcal{U}(-1, 1)$ 米（水平设定点除外；由于此处控制器的线性特性，如后文所述，我们将$p^{\mathcal{WB}}{}_z$ 固定为 1.5 米）、初始速度 $\boldsymbol{v}^{\mathcal{WB}} \in \mathcal{U}(-0.02, 0.02)$ 米/秒、初始姿态四元数 $\boldsymbol{q}^{\mathcal{WB}} \in \mathcal{U}(-0.02, 0.02)$（归一化后），以及初始角速度 $\boldsymbol{\omega}^{\mathcal{B}} \in \mathcal{U}(-0.02, 0.02)$ 弧度/秒。

我们修改了 Flightmare，使其纳入由平移运动产生的阻力 $\boldsymbol{f}^{\text{drag}}$，并假设该阻力作用于所谓的“平机体”坐标系 $\mathcal{B}'$ 中。该坐标系是机体坐标系绕无人机的横滚和俯仰角旋转后得到的，因此 Z 轴与世界坐标系的 Z 轴对齐。按照 [63]，我们采用一种阻力模型，该模型中 X 和 Y 方向的阻力与速度成线性关系，但阻力系数取为 $k_{v,x} = k_{v,y} = 0.5$。由此得到：

$$\boldsymbol{f}^{\mathcal{B}}_{\text{drag}} = -\boldsymbol{R}^{\mathcal{BB}'} \left(\begin{bmatrix} k_{v,x} \\ k_{v,y} \\ 0 \end{bmatrix} \circ \boldsymbol{R}^{\mathcal{B}'\mathcal{B}} \boldsymbol{v}^{\mathcal{B}} \right) \quad \textbf{(19)}$$

线性控制器 $\boldsymbol{c} \in \mathbb{R}^{4\times 1}$ 的输出被限制在 $[-1, 1]$ 范围内，并输入级联低层控制器（推力、姿态和角速度控制器）的不同部分。为弥补线性控制器的一些不足，我们根据无人机的姿态对推力进行补偿。

所有动力学方程均采用 4th 阶 Runge-Kutta 方法进行积分，时间步长为 2.5 ms。仿真的频率为 50 Hz。其余细节请参阅补充材料。

从仿真到现实世界

我们通过几种策略成功实现了从仿真到现实的迁移。一种是域随机化 [71]，我们通过添加噪声到观测光流中、对姿态估计值施加随机偏差，以及改变仿真四旋翼飞行器的初始条件来实现。另一种是抽象 [72]，我们利用低级控制器将推力和姿态转换为旋翼转速，并在每次飞行前校准姿态和悬停推力偏差，确保它们很小或为零（与仿真中的情况一样）。最后，我们对计算得出的视觉可观测量进行平滑处理（低通滤波）和缩放，并调整对现实世界中的控制层输出进行缩放的增益。

使用公式 13 将视觉网络产生的角点光流转换为视觉可观测量（缩放速度$\hat{\boldsymbol{v}}^{\mathcal{B}}$ 和偏航角速度 $\hat{}\omega^{\mathcal{B}}{}_z$）；绝对横滚角 $|\phi|$ 和俯仰角 $|\theta|$ 取自无人机的加速度计，缩放速度设定点 $\boldsymbol{v}^{\mathcal{B}}{}_{sp}$ 由用户提供。偏航角速度设定值 $\omega^{\mathcal{B}}{}_{z,sp} = 0$ 固定不变。

我们还通过将视觉网络连接到手动调参的比例积分（PI）控制器进行了额外实验。其余细节请参阅补充材料。

硬件配置

现实环境实验使用了一架定制四旋翼飞行器，搭载事件相机（DAVIS240C）、单板计算机（UP Squared）和神经形态处理器（Intel Kapoho Bay，配备两颗 Loihi神经形态研究芯片）。图 1 展示了总体概览，所有组件则列于表 3 中。我们使用 PX4[1] 作为自动驾驶固件，并使用 ROS[2] 进行通信。具体而言，事件相机产生的事件通过 USB 和 ROS1 传送至 UP Squared。这些事件在 UP Squared 上经过处理（降采样、裁剪），再通过 USB 以脉冲形式发送至运行于 Kapoho Bay 上的视觉网络。处理完成后，输出脉冲通过 USB 发回 UP Squared，并在此解码为角点光流。随后，ROS1 节点发布角点光流，并通过 ROS1-ROS2桥接发送至 ROS2。以 ROS2 节点运行的线性控制器（或 PI 控制器）及其周边处理模块，会接收角点光流、来自 PX4 的姿态估计以及用户提供的设定点，并输出控制指令。该指令随后通过 ROS2 发送至 PX4，并由其中的低层控制器处理。ROS2 使用 RTPS 进行通信，可实现 UP 与 PX4 之间的高频、高带宽消息传输，因此整个处理流水线能够以 200 Hz 运行。在测试运行之间进行位置控制以及作为故障保护时，我们使用 OptiTrack动作捕捉系统。

能耗基准测试

Loihi 的能耗基准测试是在 Nahuku 板上进行的[3]，该板包含 32 个 Loihi 芯片，因为 Kapoho Bay 不支持能耗测量。这些（软件）探针会报告多种功耗测量结果以及执行时间。根据文档，静态功耗源自晶体管漏电，动态功耗源自开关；测量空闲功耗时，Loihi 上的嵌入式 CPU 核心仍在运行时钟，但神经形态部分处于非活动状态；总功耗／运行功耗则是所有组件共同产生的功耗。Kapoho Bay 支持测量执行时间，我们借此确认 Nahuku 和 Kapoho Bay 的执行时间几乎相同。此外，文档指出，Nahuku 会关闭未使用的

[1] https://px4 .io/ [2] https://ros.org/ [3] 主机：Intel Xeon Platinum 8280 CPU，主频 2.7 GHz，内存 126 GB，Ubuntu 20.04，NxSDK 1.0.0

表3。 真实世界测试飞行所用硬件组件清单。

组件	产品
机架	GEPRC Mark 4 225 毫米
电机	Emax 2306 Eco II 系列
螺旋桨	Ethix S5 5 英寸
电池	Tattu FunFly 1800mAh 4S
飞行控制器	Pixhawk 4 Mini
ESC	SpeedyBee 45A BL32 4合1
单板计算机	UP Squared ATOM 四核 08/64
事件相机	DAVIS240C
神经形态处理器	Intel Loihi，Kapoho Bay 外形规格

芯片，这意味着它能够以极低的开销模拟 Kapoho Bay（配备两枚芯片）的能耗。不过，这些基准测试并不能很好地代表无人机的实际使用情况，因为实际使用涉及以在线方式接收和处理流式数据；而这里的数据会预先全部加载到内存中，然后尽可能快地进行处理。因此，这些基准测试反映的并非整个处理流水线的能耗和执行速度，而只是网络本身的情况，不受输入/输出和预处理造成的任何瓶颈或影响。这也解释了为什么 Loihi 的执行频率远高于现实世界测试中的频率；在现实世界测试中，执行频率始终约为 200 次推理/秒。

在 Jetson Nano 上，我们使用 PyTorch 仿真脉冲神经网络[4]。Jetson Nano 有两种功耗模式：低功耗（5W）模式和最高功耗（10W）模式，区别在于启用的 CPU核心 数量不同（两个或四个）。功耗使用 tegrastats 工具测量，运行时间则通过 Python 代码测量。请注意，由于无法获取相关测量值，因此这里无法区分静态功耗和动态功耗。基准测试运行后，在一段时间内测量空闲功耗。

致谢

本研究获得了荷兰科学研究组织（资助编号 NWA.1292.19.298 和 TOP项目资助 612.001.701）、空军科学研究办公室（资助编号 FA8655-20-1-7044）以及全球海军研究办公室（资助编号 N629092112014）的资助。此外，我们感谢英特尔神经形态计算实验室和英特尔神经形态研究社区对 Loihi 提供的支持。

参考文献

1. D. Cire¸san, U. Meier, J. Masci 和 J. Schmidhuber，“用于交通标志分类的神经网络委员会”，载于*2011*年国际神经网络联合会议，(2011)，第1918–1921页。2. X.Cheng, Y. Zhong, M.Harandi, Y.Dai, X. Chang, H.Li, T.Drummond 和 Z. Ge，“用于深度立体视觉的分层神经架构搜索匹配”，载于神经信息处理系统进展，第33卷（Curran Associates, Inc., 2020），第22158–22169页。3. X. Gu、Z. Fan、S. Zhu、Z. Dai、F. Tan和P. Tan，“用于高分辨率多视图立体与立体匹配的级联代价体，载于IEEE/CVF计算机视觉与模式识别会议论文集（2020），第2495–2504页。4. E. Ilg、N. Mayer、T. Saikia、M. Keuper、A. Dosovitskiy和T. Brox，“FlowNet 2.0：深度网络光流估计的进化”，载于*IEEE*计算机视觉与模式识别会议论文集，（2017），第2462–2470页。5. D. Sun、X. Yang、M.-Y. Liu和J. Kautz，“PWC-Net：利用金字塔、变形和代价体进行光流估计的卷积神经网络”，载于*IEEE*计算机视觉与模式识别会议论文集，（2018），第8934–8943页。6. Z. Teed和J. Deng，“RAFT：用于光流的循环全对场变换”，载于计算机视觉——*ECCV 2020*，（Springer International Publishing, Cham, 2020），《计算机科学讲义》，第402–419页。7. Y. Yuan、X. Chen、X. Chen和J. Wang，“分割变换器：用于语义分割的对象上下文表示”（2021）。8. Z. Liu、H. Hu、Y. Lin、Z. Yao、Z. Xie、Y. Wei、J. Ning、Y. Cao、Z. Zhang、L. Dong、F. Wei和B. Guo，“Swin Transformer V2：扩展容量与分辨率”，载于*IEEE/CVF*计算机视觉与模式识别会议论文集，（2022），第12009–12019页。9. R. Girshick，“Fast R-CNN”，载于*IEEE*国际计算机视觉会议论文集，（2015），第1440–1448页。10. J. Redmon和A. Farhadi，“YOLOv3：渐进式改进”（2018）。11. M. Xu、Z. Zhang、H. Hu、J. Wang、L. Wang、F. Wei、X. Bai和Z. Liu，“利用Soft Teacher进行端到端半监督目标检测”，载于*IEEE/CVF*国际计算机视觉会议论文集，（2021），第3060–3069页。12. R. Garg、V. K. B.G.、G. Carneiro和I. Reid，“用于单视图深度估计的无监督卷积神经网络：几何方法来助力”，载于计算机视觉——*ECCV 2016*，（Springer International Publishing, Cham, 2016），《计算机科学讲义》，第740–756页。13. C. Godard、O. Mac Aodha和G. J. Brostow，“利用左右一致性进行无监督单目深度估计”，载于*IEEE*计算机视觉与模式识别会议论文集，（2017），第270–279页。14. W. Yuan、X. Gu、Z. Dai、S. Zhu和P. Tan，“NeW CRFs：用于单目深度估计的神经窗口全连接条件随机场”（2022）。15. G. Indiveri和R. Douglas，“神经形态视觉传感器”，《科学》**288**，1189–1190（2000）。16. G. Gallego、T. Delbruck、G. Orchard、C. Bartolozzi、B. Taba、A. Censi、S. Leutenegger、A. Davison、J. Conradt、K. Daniilidis和D. Scaramuzza，“事件相机视觉综述”，《IEEE模式分析与机器智能汇刊》，第1–1页（2020）。17. A. Z. Zhu、L. Yuan、K. Chaney和K. Daniilidis，“EV-FlowNet：针对事件相机的自监督光流估计”，Robotics: Sci. Syst. XIV（2018）。18. A. Z. Zhu、L. Yuan、K. Chaney和K. Daniilidis，“基于事件的光流、深度和自运动无监督学习”，载于*IEEE/CVF*计算机视觉与模式识别会议论文集，（2019），第989–997页。19. M. Gehrig、M. Millhäusler、D. Gehrig和D. Scaramuzza，“E-RAFT：基于事件相机的稠密光流”，载于*2021*年*3D*视觉国际会议（*3DV*），（2021），第197–206页。20. F. Paredes-Valles和G. C. H. E. de Croon，“回归事件相机基础：通过光度恒常性对事件相机图像重建进行自监督学习”，载于*IEEE/CVF*计算机视觉与模式识别会议论文集，（2021），第3446–3455页。21. W. Maass，“脉冲神经元网络：神经网络模型的第三代”，《神经网络》**10**，1659–1671（1997）。22. A. Grüning和S. M. Bohte，“脉冲神经网络：原理与挑战”，载于欧洲人工神经网络研讨会，（比利时布鲁日，2014）。

[4]ARM Cortex-A57 CPU，1.43 GHz，4 GB RAM，Ubuntu 20.04，PyTorch 1.12.0

23. P. Sterling和S. Laughlin，《神经设计原理》（麻省理工学院出版社，2015）。24. F. T. Muijres、M. J. Elzinga、J. M. Melis和M. H. Dickinson，“苍蝇通过快速视觉引导的倾斜转弯躲避逼近目标”，《科学》**344**，172–177（2014）。25. M. Pfeiffer和T. Pfeil，“脉冲神经元深度学习：机遇与挑战”，《神经科学前沿》**12**（2018）。26. A. Tavanaei、M. Ghodrati、S. R. Kheradpisheh、T. Masquelier和A. Maida，“脉冲神经网络中的深度学习”，《神经网络》**111**，47–63（2019）。27. W. Bialek、F. Rieke、R. van Steveninck和D. Warland，“解读神经编码”，载于神经信息处理系统进展，第2卷（Morgan-Kaufmann，1989）。28. J. Dupeyroux、S. Stroobants和G. C. De Croon，“用于机器人学中神经形态感知的工具箱”，载于*2022*年第*8*届事件驱动控制、通信与信号处理国际会议（*EBCCSP*），（2022），第1–7页。29. C. Schuman、C. Rizzo、J. McDonald-Carmack、N. Skuda和J. Plank，“评估脉冲神经形态系统的编码与解码方法”，载于*2022*年神经形态系统国际会议论文集，（计算机协会，纽约，NY，美国，2022），ICONS ’22，第1–9页。30. K. J. Friston，“另一种神经编码？”，《神经影像》**5**，213–220（1997）。31. J. J. Eggermont，“存在神经编码吗？”，《神经科学与生物行为评论》**22**，355–370（1998）。32. M. Jazayeri和A. Afraz，“探寻神经编码：在神经空间中导航”，《神经元》**93**，1003–1014（2017）。33. O. Guest和B. C. Love，“脑成像的成功对神经编码意味着什么”，《eLife》**6**，e21397（2017）。34. E. O. Neftci、H. Mostafa和F. Zenke，“脉冲神经网络中的替代梯度学习：将基于梯度的优化能力引入脉冲神经网络”，《IEEE信号处理杂志》**36**，51–63（2019）。35. F. Zenke和T. P. Vogels，“替代梯度学习在脉冲神经网络中赋予复杂功能时展现出的非凡鲁棒性”，《神经计算》，第1–27页（2021）。36. S. S. Chowdhury、C. Lee和K. Roy，“探究泄漏对脉冲神经网络的影响”，《神经计算》**464**，83–94（2021）。37. W. Fang、Z. Yu、Y. Chen、T. Masquelier、T. Huang和Y. Tian，“引入可学习膜时间常数以增强脉冲神经网络的学习能力”，载于*IEEE/CVF*国际计算机视觉会议论文集，（2021），第2661–2671页。38. N. Qiao、H. Mostafa、F. Corradi、M. Osswald、F. Stefanini、D. Sumislawska和G. Indiveri，“一种包含256个神经元和128K个突触的可重构在线学习脉冲神经形态处理器”，《神经科学前沿》**9**（2015）。39. M. Davies、N. Srinivasa、T.-H. Lin、G. Chinya、Y. Cao、S. H. Choday、G. Dimou、P. Joshi、N. Imam、S. Jain、Y. Liao、C.-K. Lin、A. Lines、R. Liu、D. Mathaikutty、S. McCoy、A. Paul、J. Tse、G. Venkataramanan、Y.-H. Weng、A. Wild、Y. Yang和H. Wang，“Loihi：具有片上学习功能的神经形态众核处理器”，《IEEE微型计算机》**38**，82–99（2018）。40. A. Vitale、A. Renner、C. Nauer、D. Scaramuzza和Y. Sandamirskaya，“神经形态芯片上的无人机事件驱动视觉与控制”，载于*2021*年*IEEE*国际机器人与自动化会议（*ICRA*），（2021），第103–109页。41. F. Galluppi、C. Denk、M. C. Meiner、T. C. Stewart、L. A. Plana、C. Eliasmith、S. Furber和J. Conradt，“自主移动平台上的基于事件的神经计算”，载于*2014*年*IEEE*国际机器人与自动化会议（*ICRA*），（2014），第2862–2867页。42. F. Paredes-Vallés、K. Y. W. Scheper和G. C. H. E. de Croon，“用于光流估计的层次化脉冲神经网络无监督学习：从事件到全局运动感知”，《IEEE模式分析与机器智能汇刊》**42**，2051–2064（2020）。43. J. Hagenaars、F. P. Valles和G. D. Croon，“利用脉冲神经网络自监督学习基于事件的光流”，载于神经信息处理系统进展，（2021）。44. D. Floreano 和 C. Mattiussi，“面向自主视觉机器人的脉冲神经控制器的进化”，载于《进化机器人学：从智能机器人学到人工生命》（施普林格，柏林，海德堡，2001年），《计算机科学讲义》，第38–61页。45. Z. Bing、C. Meschede、K. Huang、G. Chen、F. Rohrbein、M. Akl 和 A. Knoll，“基于 R-STDP 的车道保持车辆脉冲神经网络端到端学习”，载于*2018*年*IEEE*国际机器人与自动化会议（*ICRA*），（2018年），第4725–4732页。46. M. B. Milde、H. Blum、A. Dietmüller、D. Sumislawska、J. Conradt、G. Indiveri 和 Y. Sandamirskaya，“使用混合信号模拟/数字神经形态处理系统实现机器人导航中的障碍物规避与目标获取”，《神经机器人学前沿》**11**（2017年）。47. D. H. Ballard，“将霍夫变换推广到任意形状检测”，《模式识别》**13**，111–122（1981年）。48. J. Dupeyroux、J. J. Hagenaars、F. Paredes-Vallés 和 G. C. H. E. de Croon，“使用Loihi处理器实现微型飞行器基于光流的着陆神经形态控制”，载于*2021*年*IEEE*国际机器人与自动化会议（*ICRA*），（2021年），第96–102页。49. G. C. H. E. de Croon，“通过光流机动和传出副本进行单目距离估计：一种基于稳定性的策略”，《生物启发与仿生学》**11**，016004（2016年）。50. C. Brandli、R.Berner、M. Yang、S.-C. Liu 和 T. Delbruck，“一款 240 × 180130 dB、3*M*s延迟的全局快门时空视觉传感器”，《IEEE固态电路期刊》**49**，2333–2341（2014年）。51. G. de Croon、H. Ho、C. De Wagter、E. van Kampen、B. Remes 和 Q. Chu，“自主着陆的基于光流的坡度估计”，《国际微型飞行器期刊》**5**，287–297（2013年）。52. B. J. Pijnacker Hordijk、K. Y. W. Scheper 和 G. C. H. E. de Croon，“微型飞行器使用基于事件的光流实现垂直着陆”，《野外机器人学期刊》**35**，69–90（2018年）。53. J. J. Hagenaars、F. Paredes-Vallés、S. M. Bohté 和 G. C. H. E. de Croon，“用于微型飞行器高速发散着陆的进化神经形态控制”，《IEEE机器人与自动化快报》**5**，6239–6246（2020年）。54. R. Benosman、S.-H. Ieng、C. Clercq、C. Bartolozzi 和 M. Srinivasan，“异步无帧事件光流”，《神经网络》**27**，32–37（2012年）。55. S. Baker、A. Datta 和 T. Kanade，“单应性的参数化”，技术报告 CMU-RI-TR-06-11，匹兹堡机器人研究所，宾夕法尼亚州（2006年）。56. G. Gallego、H. Rebecq 和 D. Scaramuzza，“事件相机对比度最大化统一框架及其在运动、深度和光流估计中的应用”，载于*IEEE*计算机视觉与模式识别会议论文集，（2018年），第3867–3876页。57. G. Gallego、M. Gehrig 和 D. Scaramuzza，“《焦点即是所需》：事件相机视觉的损失函数”，载于*IEEE/CVF*计算机视觉与模式识别会议论文集，（2019年），第12280–12289页。58. D. DeTone、T. Malisiewicz 和 A. Rabinovich，“深度图像单应性估计”，（2016年）。59. T. Nguyen、S. W. Chen、S. S. Shivakumar、C. J. Taylor 和 V. Kumar，“无监督深度单应性：一种快速且稳健的单应性估计模型”，《IEEE机器人与自动化快报》**3**，2346–2353（2018年）。60. N. J. Sanket、C. M. Parameshwara、C. D. Singh、A. V. Kuruttukulam、C. Fermüller、D. Scaramuzza 和 Y. Aloimonos，“EVDodgeNet：使用事件相机进行深度动态障碍规避”，载于*2020*年*IEEE*国际机器人与自动化会议（*ICRA*），（2020年），第10651–10657页。61. Y. Ma、S. Soatto、J. Košecká 和 S. S. Sastry，《三维视觉导论：从图像到几何模型》（施普林格，纽约，2004年）。62. H. C. Longuet-Higgins 和 K. Prazdny，“运动视网膜图像的解释”，《伦敦皇家学会会刊B辑：生物科学》（1980年）。63. C. De Wagter、F. Paredes-Vallé、N. Sheth 和 G. de Croon，“2019年人工智能机器人竞速竞赛获胜方案背后的感知、状态估计与控制”，《野外机器人学》**2**，1263–1290（2022）。

64. T. Ozawa、Y. Sekikawa 和 H. Saito，“利用鸟瞰视图变换提升事件相机运动估计的精度和速度”，《传感器》**22**, 773（2022）。65. A. Mitrokhin、C. Fermüller、C. Parameshwara 和 Y. Aloimonos，“基于事件的运动目标检测与跟踪”，载于*2018*年 *IEEE/RSJ*智能机器人与系统国际会议（*IROS*），（2018年），第 1–9 页。66. P. Charbonnier、L. Blanc-Feraud、G. Aubert 和 M. Barlaud，“用于计算成像的两种确定性半二次正则化算法”，载于第一届国际图像处理会议论文集，第 2 卷（1994 年），第 168–172 页，第 2 卷。67. T. Stoffregen 和 L. Kleeman，“事件相机、对比度最大化与奖励函数：分析”，载于*IEEE/CVF*计算机视觉与模式识别会议论文集，（2019 年），第 12300–12308 页。68. D. P. Kingma 和 J. Ba，“Adam：一种随机优化方法”（2017年）。69. Y. Song、S. Naji、E. Kaufmann、A. Loquercio 和 D. Scaramuzza，“Flightmare：一种灵活的四旋翼模拟器”，载于机器人学习会议，（2020 年）。70. S. Zhong 和 P. Chirarattananon，“基于迭代扩展卡尔曼滤波器的直接视觉惯性自运动估计”，《IEEE机器人与自动化快报》**5**, 1476–1483（2020 年）。71. J. Tobin、R. Fong、A. Ray、J. Schneider、W. Zaremba 和 P. Abbeel，“通过域随机化将深度神经网络从仿真迁移到真实世界”，载于*2017*年 *IEEE/RSJ*智能机器人与系统国际会议（*IROS*），（2017年），第 23–30 页。72. K. Y. W. Scheper 和 G. C. H. E. de Croon，“进化机器人学中的抽象、感觉—运动协调与现实差距”，《人工生命》**23**, 124–141（2017年）。73. E. Kaufmann、L. Bauersfeld 和 D. Scaramuzza，“敏捷四旋翼飞行的学习控制策略基准比较”（2022）。

补充材料

（续）材料与方法

仿真

Flightmare 的四旋翼动力学模型 [69]如下所示（符号说明参见 [73]）：

$$\dot{p}^{\mathcal{WB}} = v^{\mathcal{WB}} \tag{20}$$

$$\dot{q}^{\mathcal{WB}} = \frac{1}{2}\begin{bmatrix}0\\ \omega^{\mathcal{B}}\end{bmatrix}_{\times} q^{\mathcal{WB}} \tag{21}$$

$$\dot{v}^{\mathcal{WB}} = \frac{1}{m}\left(q^{\mathcal{WB}} \odot \left(f_{\text{prop}} + f_{\text{drag}}\right)\right) + g^{\mathcal{W}} \tag{22}$$

$$\dot{\omega}^{\mathcal{B}} = J^{-1}\left(\tau_{\text{prop}} - \omega^{\mathcal{B}} \times J\omega^{\mathcal{B}}\right) \tag{23}$$

$$\dot{\Omega} = \frac{1}{\tau_{\Omega}}\left(\Omega_c - \Omega\right) \tag{24}$$

此处，$\mathrm{p}^{\mathcal{WB}}$ 表示机体坐标系相对于世界坐标系的位置，$\mathrm{v}^{\mathcal{WB}}$ 表示机体坐标系相对于世界坐标系的速度，$\mathrm{q}^{\mathcal{WB}} = (\mathrm{q}_w, \mathrm{q}_x, \mathrm{q}_y, \mathrm{q}_z)$ 是一个单位四元数，表示机体坐标系相对于世界坐标系的姿态，$[0, \omega^{\mathcal{B}}]^T_{\times} \in \mathbb{R}^{4\times4}$ 是表示无穷小旋转的反对称矩阵，m 是无人机的质量，$\odot$ 是四元数向量积，f^{prop} 是所有旋翼沿机体 Z 轴产生的总推力，f^{drag} 是阻力，$\mathrm{g}^{\mathcal{W}}$ 是世界坐标系中的重力，$\omega^{\mathcal{B}}$ 是机体坐标系的角速度，$\mathrm{J} = \mathrm{diag}(\mathrm{J}_x, \mathrm{J}_y, \mathrm{J}_z)$ 是无人机的转动惯量，τ_{prop} 是旋翼产生的扭矩，$\Omega(\mathrm{c})$ 是（指令）电机转速，τ_{Ω} 是电机一阶模型的时间常数。

每个单独旋翼产生的推力和扭矩通过以下方式汇总为总推力和总扭矩：

$$f_{\text{prop}} = \sum_i f_i = m[0, 0, \bar{f}]^T \tag{25}$$

$$\tau_{\text{prop}} = \sum_i \tau_i + r_i \times f_i \tag{26}$$

其中 $\bar{f}$ 为质量归一化推力。

单个旋翼产生的推力采用二阶推力映射建模：$f_i = [0, 0, a\Omega^2 + b\Omega + c]^T$。不对旋翼阻力建模。结合式24中的电机动力学，可得到实际电机转速，进而得到单旋翼推力和扭矩。

线性控制器 $\boldsymbol{c} \in \mathbb{R}^{4\times1}$ 的输出随后被限制在 $[-1, 1]$ 范围内，并输入级联低级控制器的不同部分。首先，$\bar{f}_{0,c}$ 乘以比例增益 $6g$，并以悬停状态为中心设定：$\bar{f}_c = \bar{f}_{0,c} \cdot 6g + g$。随后，$\phi_c$ 和 θ_c 由比例控制器（增益为 $\frac{\pi}{2}$）处理，以设定相应的角速度 $\omega^{\mathcal{B}}{}_{x,c}$ 和 $\omega^{\mathcal{B}}{}_{y,c}$。接下来，三个机体角速度经过限幅，并由另一个比例控制器（相应增益为 16.6、16.6、5.0）处理，以确定目标机体扭矩 τ_{i_c}。指令中的质量归一化推力 f_c 经过限幅，并根据无人机的姿态进行补偿，以部分弥补线性控制器的不足（下一节将作更详细的说明）。

仿真中使用的所有未提及常数和系数均可在表 4中找到。

现实世界

视觉网络以像素/毫秒为单位输出角点光流，而用户提供以 1/s 为单位的缩放速度设定值。我们

表 4. 仿真和训练期间使用的系数。

描述	符号	值
无人机质量	m [千克]	1.535
无人机机臂	r [米]	0.255
最小电机转速	ω_{min} [转/分]	150
最大电机转速	ω_{max} [转/分钟]	1500
电机时间常数	τ_{ω} [秒]	0.025
电机推力映射	$a,\ b,\ c$	1.329825e-6, 0.003836, -1.768999
最大机体角速度	$\omega^{\mathcal{B}}{}_{\text{最大}}$ [弧度/秒]	$[6.0, 6.0, 6.0]^T$
相机内参矩阵	$\boldsymbol{K}$	$\begin{bmatrix}188.84, 0, 90.0\\ 0, 188.99, 90.0\\ 0, 0, 1\end{bmatrix}$
相机相对于机体的旋转	$\boldsymbol{R}^{\mathcal{CB}}$	$\begin{bmatrix}1, 0, 0\\ 0, -1, 0\\ 0, 0, -1\end{bmatrix}$
相机相对于机体的平移	$\boldsymbol{T}^{\mathcal{CB}}$ [米]	$[-0.005, 0.077, -0.033]^T$
最低高度界限	$p^{\mathcal{WB}}_{z,\text{最小}}$ [米]	0.2

通过手动缩放将二者关联起来，并对 $\hat{v}^{\mathcal{B}}$ 和控制输出进行手动缩放。此外，我们对计算出的视觉观测值进行平滑处理，使得 $\begin{bmatrix}\hat{v}^{\mathcal{B}}, \hat{\omega}^{\mathcal{B}}_z\end{bmatrix}^T = \begin{bmatrix}\hat{v}^{\mathcal{B}}, \hat{\omega}^{\mathcal{B}}_z\end{bmatrix}^T \circ \boldsymbol{\alpha}_{v,\omega} + (1 - \boldsymbol{\alpha}_{v,\omega}) \circ \boldsymbol{\beta}_{v,\omega} \circ \begin{bmatrix}\hat{v}^{\mathcal{B}}, \hat{\omega}^{\mathcal{B}}_z\end{bmatrix}^T$。

在悬停期间，我们估算悬停推力，并用它来抵消 $\bar{f}_{0,c}$。由于无人机的自动驾驶软件使用无量纲数值表示推力，我们还将其乘以比例增益 0.3，并提供姿态补偿（下文将讨论），以限制使用线性控制器带来的影响。其他控制指令也乘以比例增益 0.3。在悬停期间，我们还会校准横滚和俯仰方向的姿态偏置，并用其补偿控制器的输入 $|\phi|$ 和 $|\theta|$，以及控制指令 ϕ_c 和 θ_c。偏航方向的控制指令是角速度，但自动驾驶软件在姿态模式下只接受偏航角。为了将 $\omega^{\mathcal{B}}{}_{z,c}$ 转换为 ψ_c，我们使用外部运动捕捉系统提供的偏航角来初始化 ψ_c，然后通过 $\psi_c = \psi_c + 0.005\omega^{\mathcal{B}}{}_{z,c}$ 进行积分，其中 0.005 s 是循环的大致时间步长。

我们还进行了额外实验，将视觉网络连接到比例-积分（PI）控制器。同样，我们对（使用相同常数）进行平滑和缩放 $\hat{v}^{\mathcal{B}}$ 和 $\hat{\omega}^{\mathcal{B}}_z$，然后将其与设定点 $v^{\mathcal{B}}{}_{sp}$ 和 $\omega^{\mathcal{B}}{}_{z,sp}$ 进行比较。控制器的比例增益和积分增益经过手动调试，列于表5中，同时列出的还有真实世界测试中使用的所有常数。

如前所述，线性控制器存在若干缺点；若要成功地从仿真迁移到真实环境，就必须了解这些缺点。简而言之，它们包括：

- 要达到特定的缩放速度设定点，需要相应的姿态角。随着姿态变化，推力矢量中朝上的分量也会随之变化；这一效应是非线性的（小角度假设不成立），因此控制器无法将其考虑在内。向控制器输入绝对姿态角，可以让控制器进行一定的补偿，但仍需对

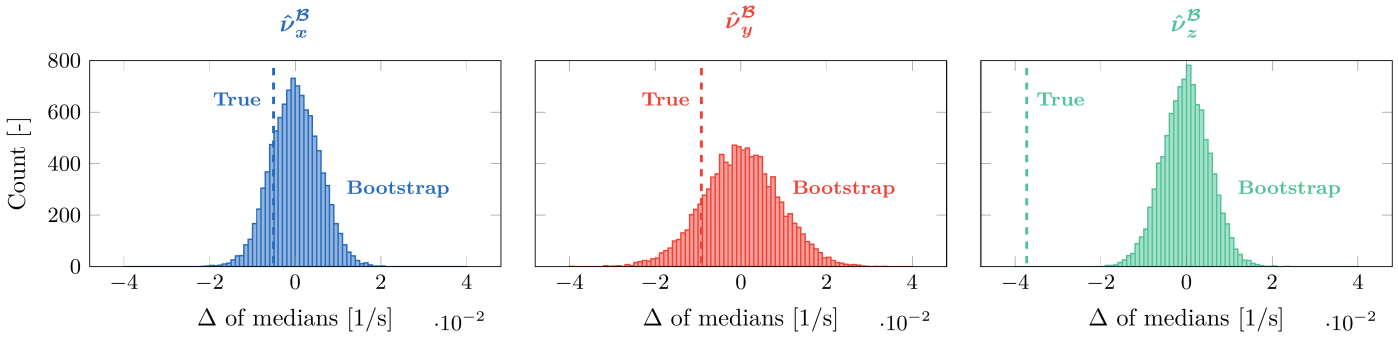

图 8. 比较有系留绳和无系留绳时的飞行测试。 真实世界飞行测试使用了系留绳。我们在有系留绳和无系留绳的情况下，以 $\boldsymbol{\nu}^{\mathcal{B}_{sp}} = [0, 0.5, 0]^T$ 1/s 的速度分别运行十次，并使用自助法量化系留绳的影响。具体而言，我们将初始瞬态之后的所有数据点合并为一个集合，抽取两个各含 454 个数据点的样本，并计算两者中位数之差。我们重复这一过程 10,000 次，并将结果与有系留绳和无系留绳测试中位数的真实差值进行比较。尽管系留绳对 $\hat{\nu}^{\mathcal{B}_z}$ 的影响显著，但其幅度也非常小。

推力进行临时的姿态补偿。

- 在不同高度下，相同的缩放速度值会产生不同的速度，因此需要不同的姿态角才能达到该速度。这种影响也是非线性的，因此我们决定所有水平飞行都从相同的起始高度开始。请注意，这并未考虑飞行过程中累积的虚拟漂移，因此这种影响仍会在一定程度上显现。

- 当无人机以与其姿态相同的倾角上升或下降时，$\nu^{\mathcal{B}_z} = 0$ 因此，需要额外的信息来抵消这种漂移。无人机的姿态可以提供这类信息，但这一效应同样是非线性的，因此总会残留一些漂移。

这些影响表明，不同设定值所产生的效果各不相同，仿真与现实世界之间也存在差异；这是由于系统具有非线性，以及仿真无人机与真实无人机的特性不同。

表 5。 真实世界测试期间使用的系数。

说明	符号	值
视觉观测量平滑	$\boldsymbol{\alpha}_{v,\omega}$	$[0.90,\ 0.90,\ 0.95,\ 0.90]^T$
视觉可观测量缩放	$\boldsymbol{\beta}_{v,\omega}$	$[0.9, 0.9, 1.0, 1.0]^T$
比例增益（推力、横滚、俯仰、偏航）	$\boldsymbol{P}$	$[0.10, 0.06, 0.06, 4.0]^T$
积分增益（推力、横滚、俯仰、偏航）	$\boldsymbol{I}$	$[0.0001, 0.0003, 0.0003, 0]^T$

系留绳影响量化

出于安全考虑，真实世界飞行测试中，无人机都系有一根轻质系留绳。图 8 通过自助法量化了这根系留绳的影响。我们在有系留绳和无系留绳的情况下分别进行了十次运行，$\boldsymbol{\nu}^{\mathcal{B}_{sp}} = [0, 0.5, 0]^T$ 1/s，并观察中位数的真实差值是否接近由自助法得到的中位数差值分布的中心。对于 $\hat{\nu}^{\mathcal{B}_x}$ 和 $\hat{\nu}^{\mathcal{B}_y}$，情况确实如此；但对于 $\hat{\nu}^{\mathcal{B}_z}$，则并非如此。不过，尽管系留绳对 Z 轴的影响显著，但影响也非常小。此外，Z 轴最容易受到外部因素的影响，例如悬停推力估计是否准确、电池退化等。