Event Camera-based Visual Odometry for Dynamic Motion Tracking of a Legged Robot Using Adaptive Time Surface

Shifan Zhu[1], Zhipeng Tang[1], Michael Yang[1], Erik Learned-Miller[1], and Donghyun Kim[1]

***Abstract*— Our paper proposes a direct sparse visual odometry method that combines event and RGB-D data to estimate the pose of agile-legged robots during dynamic locomotion and acrobatic behaviors. Event cameras offer high temporal resolution and dynamic range, which can eliminate the issue of blurred RGB images during fast movements. This unique strength holds a potential for accurate pose estimation of agile-legged robots, which has been a challenging problem to tackle. Our framework leverages the benefits of both RGB-D and event cameras to achieve robust and accurate pose estimation, even during dynamic maneuvers such as jumping and landing a quadruped robot, the Mini-Cheetah. Our major contributions are threefold: Firstly, we introduce an adaptive time surface (ATS) method that addresses the whiteout and blackout issue in common time surfaces by formulating pixel-wise decay rates based on scene complexity and motion speed. Secondly, we develop an effective pixel selection method that directly samples from event data and applies sample filtering through ATS, enabling us to pick pixels on distinct features. Lastly, we propose a nonlinear pose optimization formula that simultaneously performs 3D-2D alignment on both RGB-based and event-based maps and images, allowing the algorithm to fully exploit the benefits of both data streams. We extensively evaluate the performance of our framework on both public datasets and our own quadruped robot dataset, demonstrating its effectiveness in accurately estimating the pose of agile robots during dynamic movements.**

I. INTRODUCTION

Legged robots are developed to tackle a range of demanding tasks, such as disaster response [1], search-and-rescue operations [2] [3], patrolling and exploring challenging environments such as forests, mountains, underwater, and even space [4]–[9]. In order to navigate through such rough terrain, one essential function is to accurately estimate a robot's position and orientation with respect to the ground. However, the accuracy of traditional RGB-based visual odometry (VO) or integration of VO and inertia measurement unit (VIO) [10]–[12] can significantly drop in dark or highly dynamic environments, where images can be blurred or under-exposed. To address this issue, researchers began to utilize a new type of camera, called event cameras, which offer microsecond-scale temporal resolution and a high dynamic range of up to 140 dB, compared to the standard RGB camera's dynamic range of 60 dB.

Event cameras differ from RGB cameras in that they detect brightness changes in the scene asynchronously and independently for every pixel, which allows for almost continuous sensing without image blur even under dynamic movement or low-light conditions [13]. Unlike RGB cameras which generate images with 3-channel values, the event data stream includes the position, timestamp, and polarity of emitted events. A common approach to using event data is constructing an event image by accumulating events in a single frame at a constant rate, then applying techniques developed for RGB images. [14]–[16] proposed feature-based algorithms, which detect and track features (e.g. corners, lines, etc) from event images. Although the prior works showed impressive tracking performance during dynamic movements, the feature detection and tracking are not robust to random movements because the event image depends on not only texture but also the motion of the camera. This motion-dependency problem is particularly significant in legged robots, which involve sudden motion direction changes and impact disturbances from touch-downs and jumps [17]. In addition to these feature detection and tracking challenges, the process of feature extraction and matching is time-consuming and may require sacrificing the low-latency features of an event camera.

Unlike feature-based (indirect) methods, direct methods [19] use pixel brightness to estimate the pose, rather than identifying and matching features. Therefore, direct methods are more robust to changes in the scene as long as the brightness remains consistent and the changes are gradual, which are necessary conditions to solve photometric error minimization. However, event images are constructed by binary event data, resulting in discrete changes in gradients that make the optimization difficult to converge smoothly to

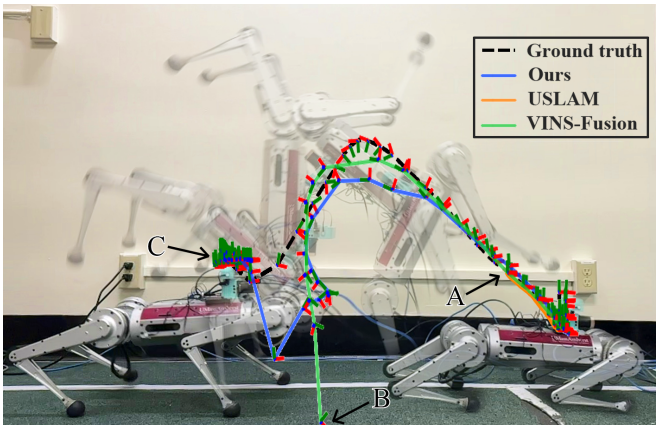


Fig. 1. **Pose estimation during the backflip of Mini-Cheetah.** The trajectory of the robot is shown in transparent figures, while the trajectories from different algorithms are shown in different colors. USLAM [15] diverges at point A. VINS-Fusion [18] diverges at point B. Our method converges to the accurate position C.

[1]University of Massachusetts Amherst, 140 Governors Dr, U.S. donghyunkim@cs.umass.edu

Manuscript 2788 submitted to 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 2, 2023.

a correct pose. A popular approach to remedy the issue is to use a time surface [20], which is a 2D map constructed by decaying grayscale values based on the timestamp of the last spiked event (Fig. 4). The resulting smooth gradient allows the optimizer to converge to the correct pose.

While time surfaces can create smooth gradients for pose estimation, choosing the correct decay rate is often challenging in real-world scenarios because the time surfaces may overlap or lose event data depending on camera motion speed and the complexity of the environment's textures if the decay rate is incorrectly set. [21] proposed a speed-invariant time surface, but this approach only considers motion speed and does not account for texture complexity, which can be an issue in environments with rich textures. To address this limitation, we propose an adaptive time surface (ATS) that adjusts the decay rate based on both camera motion and environment textures. Our ATS computes the pixel-wise decay rate by analyzing the temporal event density of neighboring pixels. This allows the ATS to decay faster in regions experiencing high-texture environments or fast motion, generating a better-represented time surface map. Conversely, in parts of the ATS with low-texture environments or experiencing slow motion, the ATS keeps event data longer, resulting in clear and distinct selected pixels.

In addition to the challenges associated with time surfaces, detecting and maintaining distinctive pixels remain significant issues in direct methods too. For instance, [22] selects pixels based on a constant threshold that can result in either too sparse or dense pixel selection, and the poorly distributed points can impede optimization [19]. Also, picking pixels from distinctive features in the scene is important to achieve consistent key point matching since direct methods do not explicitly detect or track features. To address this issue, we propose a novel approach for selecting pixels directly from event data and performing filtering based on ATS. We employ two filtering processes. Firstly, we eliminate all pixels falling on the black areas of a median blurred ATS. Next, we further select only the pixels with high grayscale value and gradient of ATS. Additionally, we enforce a minimum distance between all selected pixels to achieve better point distribution. This approach helps to avoid selecting pixels in noisy regions and obtain well-distributed points in distinctive areas, such as the edges or corners of the scene.

Selected key points that are correlated with depth information are used to construct a map keyframe (Fig. 2), or to compute a pose. During pose estimation, photometric errors in both RGB-based and event-based maps and images are simultaneously minimized to fully exploit both data streams. Through these three significant algorithmic improvements, namely ATS, pixel selection and filtering, and simultaneous optimization over RGB and event data, we achieved an accurate and robust pose estimator. Prior research on this topic has proposed an event-based direct method [23], but their algorithm may need to compromise the event camera's high temporal resolution because the event generation model (EGM) requires RGB images to generate a brightness increment image, which can be affected by the motion blur of RGB images.

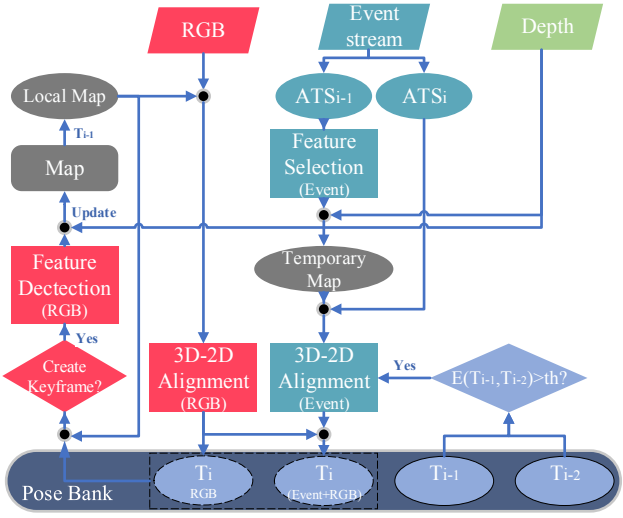


Fig. 2. **Overview of our pose estimation framework.** During slow motion, we utilize RGB image-based pose estimation alone (indicated by red) to calculate the current pose, $\mathbf{T}_i$. When a substantial movement is detected, we activate the event stream for pose estimation by creating a temporary map and performing 3D-2D alignment (indicated by cyan) in conjunction with the RGB-based map and image.

Significant progress has been made in event-based visual odometry in recent years; however, prior methods have mainly been tested in aerial systems (e.g. drones) or wheeled ground vehicles, which typically do not undergo sudden changes in motion direction. Moreover, the primary purpose of pose estimation in these systems has been obstacle avoidance or path following, which can tolerate relatively large errors in estimation accuracy. In contrast, legged robots traverse rough terrains by making contact with the ground, which demands greater robustness and accuracy of pose estimation. For example, the dynamic maneuvering of a legged robot involves jerky movements and impacts from ground touchdowns, which can significantly disturb vision sensors. Also, even a slight error in estimation can cause stumbling or falling, leading to balance failure. To the best of our knowledge, prior event camera-based estimation algorithms have not been tested on legged robots, and our experiments show that state-of-the-art event-based algorithms [15], [18] quickly diverge once a robot makes dynamic locomotion involving aerial phase, highlighting the need for more robust and accurate pose estimation methods for legged robots.

Our contributions can be summarized as follows: 1) development of a direct-method-based estimation framework that integrates RGB-D and event data to achieve accurate and robust pose estimation of legged robots, without requiring an IMU sensor, 2) extensive algorithmic improvements of pixel selection and tracking, including a novel pixel-wise adaptive decay rate of time surface, an effective pixel selection algorithm using event data and our ATS, and a simultaneous pose estimation using both RGB-based and event-based data, and 3) compelling 6-DoF motion evaluations on both a

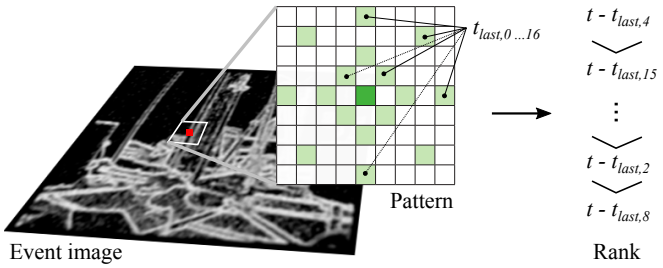


Fig. 3. **Adjacent pixel selection pattern.** When building an ATS, we first pick n pixels (dark green grid) and select the latest event data (6 selected light green grids) around each pixel using the given pattern.

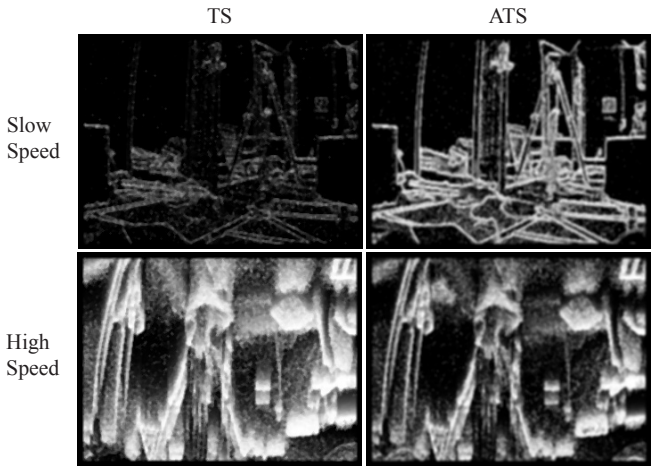


Fig. 4. **Time surface (TS) and adaptive time surface (ATS).** Time surface and our adaptive time surface in walking motion (low speed) and backflip motion (high speed). ATS provides a clearer representation in both situations.

public dataset and our own quadruped robot dataset. Our results demonstrate pose estimation less than 7 cm position error during dynamic locomotion such as trotting, pronking, and bounding. In addition, for the first time, our method successfully captures the acrobatic backflip motion of a quadruped robot (Fig. 1) without divergence.

II. Pose Estimation Framework

The core of the proposed method follows the PTAM [24] model that separates the SLAM system into tracking and mapping threads. As shown in Fig. 2, the architecture takes synchronized and aligned RGB, depth, and event data as input. The left part is RGB and depth-based mapping (gray blocks) and tracking (red blocks) algorithm, which utilizes RGB-D data from depth images to construct a map based on an initial pose and estimated pose by a direct method using 3D-2D alignment. The right part shows a fusion strategy utilizing both RGB and event data (cyan blocks) to estimate a pose when a large motion is detected. During normal operation, if the relative pose between consecutive frames is below a threshold, the tracking module employs only the RGB-based local map and the current RGB image in 3D-2D alignment to estimate the current pose T_i^{RGB}. However, when the relative motion surpasses the threshold, the tracking module fuses information from the RGB-based local map, RGB image, event-based temporary map, and ATS to achieve superior tracking performance to estimate $T_i^{\mathrm{RGB+Event}}$. In the following sections, we will first explain the process of adaptive time surface map (**ATS**) construction and pixel selection, and then introduce the mapping and tracking modules.

A. Adaptive Time Surface

A time surface map is a 2D image that visualizes the history of moving brightness patterns at each pixel and emphasizes the most recent event data with a higher grayscale value. Specifically, the grayscale value at each pixel location $\mathbf{x}$ is calculated based on the following equation:

$$\mathcal{T}(\mathbf{x}, t) = 255 \times \exp\left(-\frac{t - t_{\text{last}}\,(\mathbf{x})}{\tau(\mathbf{x})}\right), \qquad (1)$$

where τ is typically set to a constant value, which makes all the pixels decay at the same ratio. However, depending on the camera motion and environment texture, the constant decay rate can cause an image with either too little or too much event data, neither of which is desirable. In this paper, we propose a novel adaptive time surface that calculates pixel-wise decay rate, $\tau(\mathbf{x})$, based on the surrounding pixels' timestamp. The decay rate of ATS is calculated by

$$\tau(\mathbf{x}) = \max\left(\tau_u - \frac{1}{n}\sum_{i=0}^{n}(t - t_{last,i}), \tau_l\right), \qquad (2)$$

where τ_u and τ_l are the upper and lower bounds of the decay rate, respectively. $t_{last,i}$ is the timestamp of the n latest pixels around x that are selected by the patterns depicted in Fig 3. Note that the pixels with $t_{last} = 0$, meaning that the pixels have not been activated, are not included in the ranking computation. Once we pick the n latest pixels around x, then the upper bound is subtracted by the average of the time gap between the timestamp and the current time. The subtracted number sets the decay rate of the pixel x unless it is smaller than the lower bound. Then blur and median blur filters are applied to produce a smoother result.

Our ATS utilizes this adaptive decay rate based on the complexity of the environment being processed. This approach ensures that the time surface decays faster in high-texture environments or during a high-speed motion to prevent overlapping pixels or white-out issues and construct a time surface with distinct pixels. In contrast, the ATS decays slower in low-texture environments or during low-speed motion, which ensures that the time surface captures sufficient information over a longer period, leading to improved pixel selection. Fig. 4 compares the ATS algorithm with a traditional time surface that uses a constant decay rate, τ. The figure clearly demonstrates that the ATS algorithm produces a more distinct and clear surface under both slow-speed and high-speed motion.

B. Pixel selection and filtering

We utilize different pixel selection strategies for RGB images and ATS. For RGB images, we followed the idea

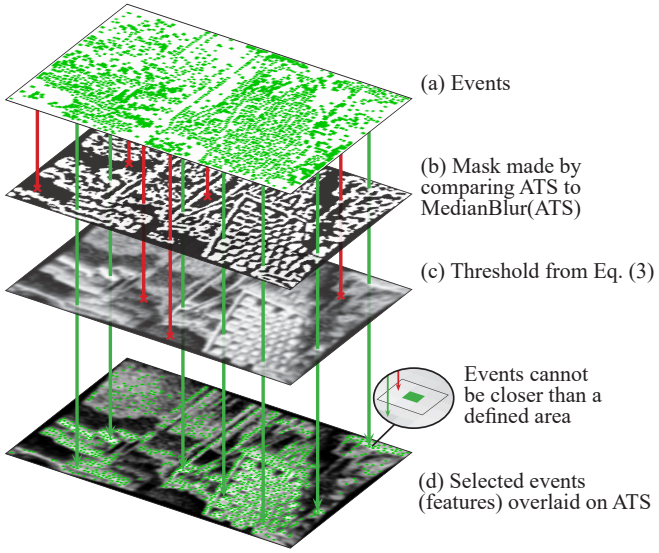


Fig. 5. **Pixel detection and filtering on ATS.** Events (a) are filtered through bright regions in the ATS through the use of a mask (b), then through a grayscale threshold (c). Each event must be a certain amount of pixels apart to prevent crowding. The result is a set of selected pixels (d). The rejected events are marked as red crosses at the end of red lines.

explained in [19] to ensure well-distributed key pixels. In our algorithm, we first divide an image into $d \times d$ blocks, then select pixels with gradients exceeding a certain threshold, which is adjusted based on the number of key pixels selected in each block. Therefore, each block has a different threshold depending on the underlying texture, and the adjusted thresholds help every block to contain an adequate number of key pixels.

In the case of pixel selection in ATS, we tried to avoid pure gradient-based methods to pick pixels from distinctive features in the scene. One issue of gradient-based key pixel selection in time surfaces (TS) is that TS usually employ filtering techniques that smooth out the gradients of TS, which can make it challenging to extract key pixels since high gradients are commonly used to identify them. To address this issue, we propose a novel pixel selection and filtering algorithm that selects pixels around the brightest regions in the ATS. Our approach starts by making a new image by applying a median blur to the original ATS, retaining only the pixels whose grayscale value on ATS image exceeds the median value of the blurred image, which is shown in Fig. 5 (b). Next, we project events accumulated during a quarter duration of the time used for ATS building onto the mask, and only the points that fall on the white region of the mask are considered for further filtering. In the subsequent filtering round, we project the remaining events back to ATS, selecting only those with high grayscale value and gradient as final candidates. We further reduce the number of key pixels based on the distance between the points. The final filtering round is summarized in Eq. (3), and the resulting pixel selection and filtering output on the ATS is shown in Fig. 5(d).

$$S_{\text{pixel}} = \{(u,v) | I(u,v) + \alpha \nabla_I(u,v) > h, \\ |u_i - u_j| > d, |v_i - v_j| > d\} \tag{3}$$

where $I(u,v)$ is the grayscale value and $\nabla_I(u,v)$ is the gradient for that pixel and α is a scale factor. New pixels should be away from the existing pixels for d pixels both in X and Y coordinates.

C. Mapping

Fig. 2 illustrates the RGB and depth-based mapping module, which follows a conventional SLAM architecture. The mapping operation is only executed when inserting keyframes. We insert a keyframe based on the number of tracking pixels and their distribution. Specifically, the image is divided into nine regions, and each region is considered healthy if the number of tracking pixels exceeds a designated threshold. The total number of healthy regions and tracking pixels determines whether a keyframe should be inserted, new pixels selected, and new map points constructed. Additionally, each map point consists of multiple grayscale value arrays, with each array storing grayscale values of the new pixel and adjacent pixels on each pyramid layer.

To improve the tracking accuracy in dynamic environments, we construct a temporary map that is built when the relative motion between the previous two frames exceeds a predefined threshold. The temporary map utilizes detected pixels from the ATS and depth data as input, with each map point containing the same information as the primary map point, but with grayscale values sourced from ATS. We calculate the relative motion factor based on the angular velocity and linear velocity of the camera between the T_{i-2} and T_{i-1} frames. This factor can also be replaced by an IMU sensor or image blur detection module. The proposed approach significantly enhances the tracking robustness of our system in challenging, rapidly changing scenarios.

D. Tracking

We have two tracking modules, one solely based on RGB images and another using event data along with the RGB images. The second module is activated when we detect large movement change, $\mathbf{T}_{i-1} - \mathbf{T}_{i-2} > threshold$. Here, we explain the second module, which integrates both RGB and event data. The primary goal of the tracking module is to find T_i from the following equation,

WRONG!

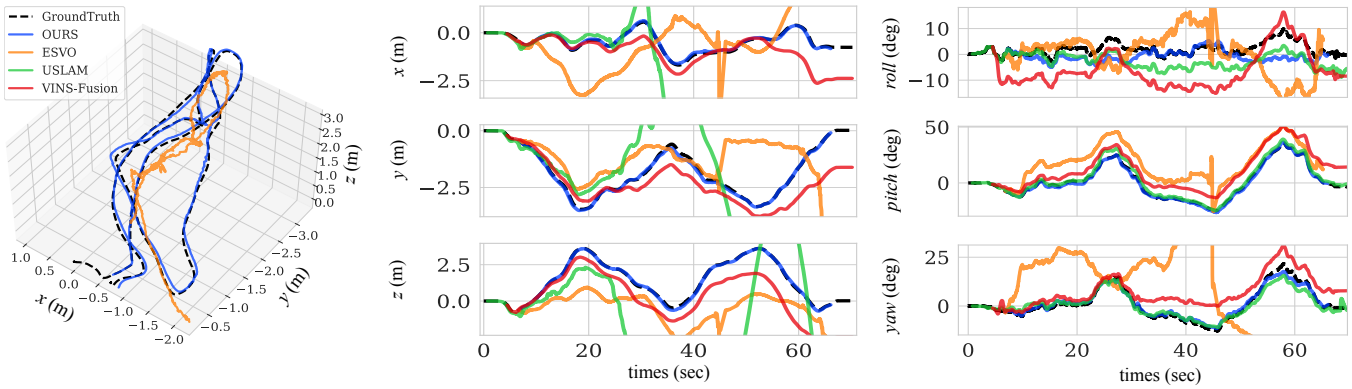
$$\min_{\mathbf{T}_i} \quad \omega_1 \sum_{j \in M_{T_{i-1}}} {e^j}^\top W_p^j e^j + \omega_2 \sum_{k \in N_{T_{i-1}}} {e^k}^\top W_q^k e^k$$

s.t.

$$e^j = \sum_{p=0}^{8} \left(z_p^j - \pi_{\text{RGB}} \left(\mathbf{T}_i^{\text{RGB}} M_p^j \right) \right) \tag{4}$$

$$e^k = \sum_{q=0}^{13} \left(z_q^k - \pi_{\text{event}} \left(\mathbf{T}_{\text{RGB}}^{\text{event}} \mathbf{T}_i^{\text{RGB}} N_q^k \right) \right),$$

where ω_1 and ω_2 respectively are the weight factors for RGB-based tracking and ATS-based tracking, and $W_{p,q}$ are



(a) 3D path (b) Position (c) Orientation

Fig. 6. **Qualitative comparison of trajectory, position, and orientation of MVSEC indoor flying sequence** (a): the 3D paths of ESVO, GroundTruth, and our method. (b): Position error in XYZ axis. (c): Orientation in RPY order. (a) depicts the 3D path of the best accurate trajectory for clear representation. (b) and (c) are zoomed in to provide a closer view of the trajectory that is in proximity to the GroundTruth.

TABLE I

COMPARISON ON MVSEC DATASETS. [$\mathbf{R}_{\text{RPE}}$: °/d, $\mathbf{t}_{\text{RPE}}$: cm/d, $\mathbf{t}_{\text{ATE}}$: cm]

	ESVO			VINS-Fusion			USLAM			Ours		
	$\mathbf{R}_{\text{rpe}}$	$\mathbf{t}_{\text{rpe}}$	$\mathbf{t}_{\text{ate}}$	$\mathbf{R}_{\text{rpe}}$	$\mathbf{t}_{\text{rpe}}$	$\mathbf{t}_{\text{ate}}$	$\mathbf{R}_{\text{rpe}}$	$\mathbf{t}_{\text{rpe}}$	$\mathbf{t}_{\text{ate}}$	$\mathbf{R}_{\text{rpe}}$	$\mathbf{t}_{\text{rpe}}$	$\mathbf{t}_{\text{ate}}$
flying1	0.69/1.02	2.90/4.98	14.40/47.65	0.45	2.38	167.38	0.42/0.45	5.00/–	58.56/–	**0.36**	**1.26**	**6.89**
flying2	0.96/–	5.51/–	338.28/–	0.82/–	7.07/–	55.75/–	0.59/–	13.87/–	159.22/–	**0.49**	**1.56**	**8.92**
flying3	0.59/0.64	2.71/2.73	11.10/16.03	0.42	2.77	205.02	0.39/–	4.24/–	31.27/–	**0.36**	**1.25**	**5.26**
flying4	–/–	–/–	–/–	0.69	3.79	46.13	**0.55**	4.45	38.58	0.63	**1.75**	**9.16**

The symbol / separates evaluation on partial trajectory and full trajectory, where the partial trajectory is manually cut before the algorithm diverges significantly. The symbol – indicates that the algorithm fails at an early stage of the experiment.

information matrices. j is the index of visible map points given the previous pose, $\mathbf{T}_{i-1}$, in the RGB-based map M. k denotes the index of the points in the event-based temporary map, N. z_p^j are the saved grayscale values when we construct the map point j (RGB), which are the same for z_q^k and the map point k (event). p and q are the indices of adjacent pixels around the selected pixel in the RGB-based map and event-based temporary map, respectively. The function π represents the camera-to-image projection. In summary, Eq. (4) minimizes the errors between the saved grayscale values and the grayscale values of the image at the projected points from the maps to the image through the pose, $\mathbf{T}_i$. If we use an RGB-only estimation process, ω_2 in Eq. (4) is set by zero.

The proposed strategy aims to mitigate the effects of motion blur while leveraging the constraints provided by RGB images and fusing event data together to provide better constraints. This complementary approach is advantageous because motion blur typically affects the parts of an RGB image that are perpendicular to the motion, and incurs pixel selection and tracking failures. In contrast, these regions often trigger the most events, which provide valuable constraints to the optimizer. By fusing event data with RGB images, the proposed approach can better leverage the strengths of each modality and provide more robust tracking results. In addition, constant motion and zero motion expectations are used to select the initial guess for RGB-based tracking. On the other hand, only zero motion prediction is applied to event-based tracking since the actual motion of the system is often random when event-based tracking is activated.

III. EVALUATION AND EXPERIMENT RESULTS

We evaluate the performance of our estimation framework on a public dataset, called the Multi-Vehicle-Stereo-Event-Camera Dataset (MVSEC) [25], and our self-collected dataset using a Mini-Cheetah robot. To ensure a fair comparison, several strategies were employed. Firstly, an $SE(3)$ alignment strategy is applied to the saved trajectory by taking the beginning frames into consideration. This is because each algorithm's local frame is defined when the algorithm is successfully initialized. Additionally, an $SO(3)$ alignment is applied to make the orientation of the first frame the same as the ground truth orientation. The alignment is done by EVO [26]. All the results are obtained by running the algorithms ourselves, except for DEVO [22], where we directly adopt the accuracy results from their original paper as the source code is not available. For relative pose error, degrees per frame are compared with DEVO. And we choose degrees per degree as the evaluation metric to compare with other algorithms in our dataset because the dataset includes static motion.

Estimation results with an absolute trajectory error (ATE) greater than 5 m are considered as diverged, while relative pose errors (RPE) above 1 °/d (degree per degree) and 0.2 m/d (meter per degree) are also considered as diverged. If an algorithm diverged in the middle of running, we compute the errors of the partial trajectory by cutting the

TABLE II
COMPARISON ON MVSEC DATASETS.
$[\mathbf{R}_{\text{RPE}} : {}^\circ/\text{f}, \mathbf{t}_{\text{RPE}} : \text{cm/f}, \mathbf{t}_{\text{ATE}} : \text{cm}]$

	DEVO			Ours		
	$\mathbf{R}_{\text{rpe}}$	$\mathbf{t}_{\text{rpe}}$	$\mathbf{t}_{\text{ate}}$	$\mathbf{R}_{\text{rpe}}$	$\mathbf{t}_{\text{rpe}}$	$\mathbf{t}_{\text{ate}}$
flying1	0.30	0.88	20.58	**0.15**	**0.57**	**6.89**
flying2	0.36	1.12	11.33	**0.20**	**0.70**	**8.92**
flying3	0.53	1.21	10.60	**0.15**	**0.60**	**5.26**
flying4	0.53	1.44	13.16	**0.26**	**0.81**	**9.16**

trajectory before its estimation diverges. Note that no loop closure was performed to maintain consistency across all the algorithms.

A. Experiment on MVSEC Dataset

Four indoor sequences in MVSEC are used for the evaluation because they include synchronized event data, grayscale images, depth data obtained by a LiDAR, and ground-truth trajectories captured by a LiDAR-based algorithm, which are necessary to run various algorithms including ours. We have compared our method with four state-of-the-art algorithms:

1) ESVO [27]: A stereo visual odometry algorithm that utilizes two event cameras (input: stereo-event streams),
2) DEVO: Latest event and depth data-based pose estimator (input: depth and event data), which has a similar sensor setup as our method,
3) UltimateSLAM (USLAM) [15]: state-of-the art event-based VIO that demonstrated great performance under aggressive motion (input: RGB images, event data, and IMU data),
4) VINS-Fusion [18], [28]: a leading RGB and IMU-based VIO (input: RGB images from a mono camera and IMU data)

All algorithms have been evaluated qualitatively and quantitatively on MVSEC.

Fig. 6 shows the pose estimation results in terms of 3D trajectory, position, and orientation on indoor flying sequence data. In Fig. 6(a), only the ground truth, our proposed method, and partial ESVO trajectories are shown, as the position errors of the other two algorithms are clearly worse than ours, which can be found in the position plots (Fig. 6(b)). Table I presents the quantitative results for all four indoor sequences, and both ATE and RPE are presented. Our method achieves less than 9.16 cm absolute trajectory error and outperforms all other algorithms in terms of both position and orientation. USLAM shows the best relative rotation error in flying4, which is a short and fast flying sequence where the IMU can provide accurate orientation information. Furthermore, algorithms that incorporate IMU sensor data are able to achieve decent orientation performance even when their ATE is large.

One interesting finding of USLAM is that it can be initialized even when most of the camera's view is toward the ground, which does not contain many features. However, USLAM diverges in the middle of flying2 and flying3 datasets, as indicated by the $-$ symbol. This may be due to improperly performed feature detection because, when the feature distribution is poor, the condensed features in a small image region do not provide sufficient constraints to obtain a 6-DoF pose. VINS-Fusion shows good tracking performance, but the gradual drifting of position estimation leads to high absolute trajectory error.

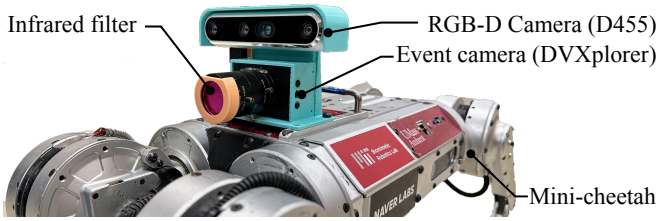


Fig. 7. **Experimental setups.** Event camera and RGB-D camera are attached to the top of the Mini-cheetah robot. We use an infrared filter to filter out the infrared array spread by a depth camera.

We consider ESVO to be the strongest competitor, as it achieves decent results on flying1 and flying3 datasets although it diverges in the middle of flying2 and flying4. As shown in Fig. 6(a), the trajectory exhibits significant vibration, which indicates unstable tracking. This can be attributed to the inability of the time surface to retain information over long periods, rendering tracking and triangulation vulnerable, particularly in slow motion.

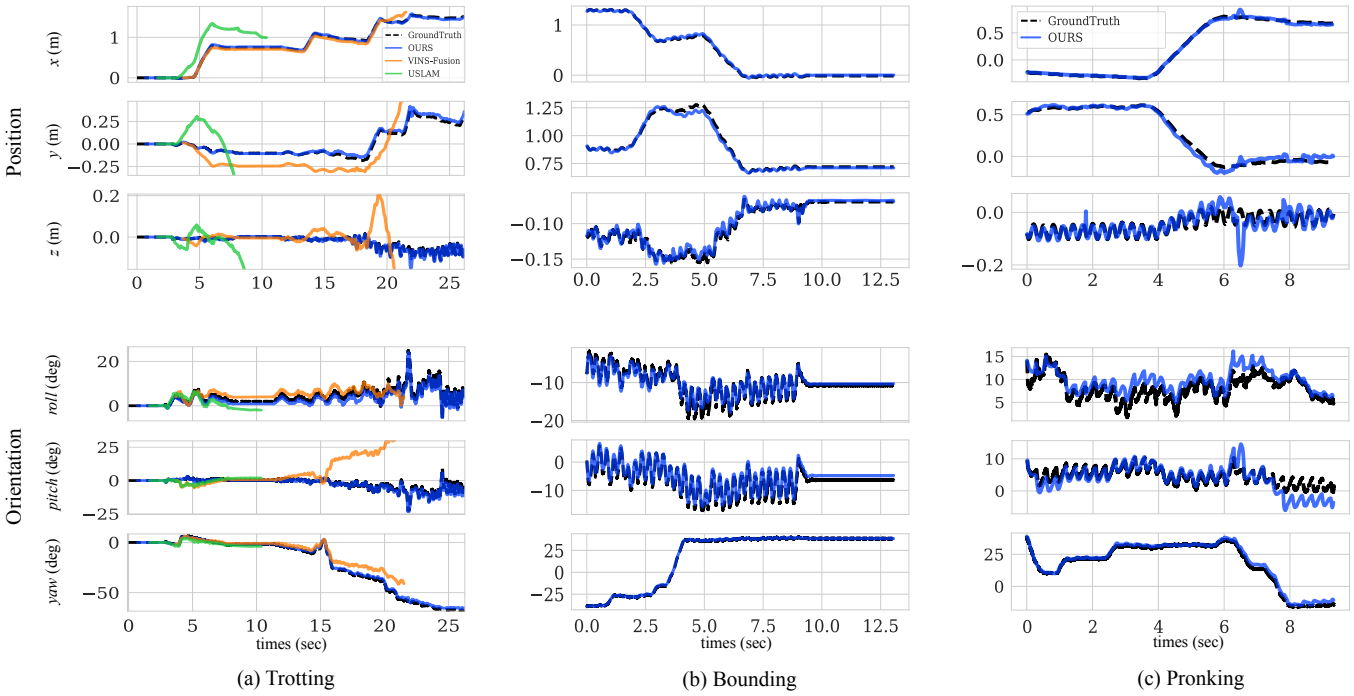
B. Experiment on Our Quadruped Dataset

We collect data from an event camera with 320 * 240 resolution (DVXplorer Lite) [29] and a Realsense D455 camera mounted on top of the Mini-Cheetah robot as shown in Fig. 7. The data was collected while the robot perform a range of dynamic motions, including trotting, pronking, bounding, and backflips. Notably, the backflip, pronking, and bounding motions involve significant aerial phase, with angular velocities up to $510\ {}^\circ/\text{s}$ in backflips and $260\ {}^\circ/\text{s}$ in bounding.

In the evaluation of pose estimation algorithms on our quadruped robot dataset, we compare the performance of our algorithm against VINS-Fusion and USLAM. In VINS-Fusion, we input RGB images and IMU data, while USLAM takes an event stream and IMU data as input since the event camera we are using does not have an RGB stream.

Note that our evaluation is conducted on a sequence of trotting, bounding, and pronking motions. Trotting is a relatively gentle walking gait, while bounding and pronking are more dynamic locomotion, as evidenced by the position and orientation change in Fig. 8. The experiment starts with trot gait and the gait is switched to bounding motion in 20 s. For a better understanding of the quadruped gaits, we refer to Fig. 5 in [30].

In the experiment, we found that the performance of USLAM, depicted by the green line in Fig. 8(a), significantly deteriorates as soon as the robot starts trotting. VINS-Fusion, on the other hand, maintains reasonable estimation accuracy during normal trotting but quickly diverges when the robot starts bounding. Our algorithm, however, demonstrates remarkable survivability and achieves an overall accuracy of



(a) Trotting (b) Bounding (c) Pronking

Fig. 8. Experiments on a sequence of trotting, bounding, and pronking motion. Top row: XYZ position in trotting, bounding, and pronking sequence. Bottom row: RPY in trotting, bounding, and pronking sequence. (a) Trotting motion: USLAM quickly diverges when the robot starts moving, while VINS-Fusion diverges during the aggressive motion at 20 seconds when the robot switches to bounding. (b) Bounding motion: The orientation changes quickly in the bounding motion. The proposed method accurately tracks the robot's motion. (c) Pronking motion: The position along the Z-axis changes rapidly. Despite this, the proposed method still accurately tracks the robot's motion.

TABLE III

COMPARISON ON OUR RECORDED DATASETS.

$[\mathbf{R}_{\text{RPE}} : {}^\circ/\text{d}, \mathbf{t}_{\text{RPE}} : \text{cm/d}, \mathbf{t}_{\text{ATE}} : \text{cm}]$

	VINS-Fusion			OURS		
	$\mathbf{R}_{\text{rpe}}$	$\mathbf{t}_{\text{rpe}}$	$\mathbf{t}_{\text{ate}}$	$\mathbf{R}_{\text{rpe}}$	$\mathbf{t}_{\text{rpe}}$	$\mathbf{t}_{\text{ate}}$
backflip1	2.46/2.87	4.17/–	16.75/–	**1.45**	**1.99**	**8.51**
backflip2	2.01/1.90	2.60/–	12.36/–	**1.18**	**1.75**	**5.31**
running1	0.82/–	1.42/–	22.08/–	**0.75**	**0.55**	**4.65**
running2	1.01/–	0.44/–	15.51/–	**0.71**	**0.56**	**4.74**
bounding	–	–	–	**0.66**	**0.42**	**2.26**
pronking	–	–	–	**0.84**	**0.92**	**6.90**

running1 and running2 are a combination of different gaits, including trotting, bounding, pronking, etc. The symbol / separates evaluation on partial trajectory and full trajectory, where the partial trajectory is manually cut before the algorithm diverges significantly. The symbol – indicates that the algorithm fails at an early stage of the experiment.

4.65 cm across all gaits. One potential reason for the failure of the other two algorithms is that the impact disturbance from touchdown is too large to maintain the stable estimation because SLAM algorithms including USLAM and VINS-Fusion have been developed for wheeled robots or drones, which experience little impact disturbance during maneuvering.

USLAM wraps events onto the image plane and utilizes the IMU to perform motion compensation to obtain a sharper event image, which can be challenging for feature detection and tracking. The motion of legged robots contacting with the ground can be quite random, resulting in an inconsistent event image between adjacent frames. Moreover, contacting with the ground generates noisy IMU data, which renders the strategy of using IMU for motion compensation ineffective, particularly without the RGB input.

VINS-Fusion succeeds in maintaining stable tracking during gentle trotting motion based on the RGB stream, but the more agile motion causes image blur, leading to feature tracking and pose estimation failure. In contrast to feature-based methods, the direct method that we use does not rely on feature detection and tracking. Instead, it utilizes all the available edge information in the image to provide constraints on the tracking module. This approach can be advantageous in situations where feature detection and tracking become challenging, such as in the case of aggressive motions where the images may become blurred. By using all the available edge information, the direct method can provide more robust pose estimation even in challenging scenarios. In addition, our algorithm takes advantage of both RGB-D data and event streams as input to constrain the tracking module. Specifically, the image blur caused by the aggressive motion lies on the texture of the image that is perpendicular to the motion, where many events are generated to provide constraints, making our algorithm effective during bounding and pronking.

The proposed system is tested with an Intel Core i7-5820K CPU on a desktop computer. The proposed algorithm sequentially processes the data queue, with the overall optimization completed within 10 ms and the RGBD-only tracking algorithm completed within 12 ms. In RGBD and event fusion mode, the tracking module takes approximately 80 ms. While the current frame rate is suboptimal, further improvements can be made to achieve real-time performance.

IV. Conclusions and Discussions

We present a novel event camera-based visual odometry approach that utilizes both RGB-D and event data to enhance pose estimation accuracy. Our method incorporates a pixel-wise adaptive time surface generation strategy and efficient pixel selection method to provide more robust key points for the tracking module, particularly during aggressive motion. Our results demonstrate significant enhancement in accuracy and robustness over the sudden movements of a robot compared to prior visual odometry algorithms. We expect a meaningful extension of the legged robot application because of the improved pose estimation of agile systems, which has been a less highlighted and unsolved problem in the traditional SLAM domain.

Due to limited access to codes and a compressed development timeline, we were unable to complete the evaluation of several recent algorithms [16] [23] [31] on our dataset. Continuous efforts will be made for further evaluation and comparison with other approaches in the future. Also, we plan to integrate an IMU sensor into the proposed method to exploit effectively its high-accuracy angular velocity measurement to the axes that are orthogonal to the gravity direction.

ACKNOWLEDGMENT

We express our gratitude to Naver Labs and MIT Biomimetic Robotics Lab for providing the Mini-cheetah robot as a research platform for conducting dynamic motion studies on legged robots.

References

[1] T. Yoshiike, M. Kuroda, R. Ujino, H. Kaneko, H. Higuchi, S. Iwasaki, Y. Kanemoto, M. Asatani, and T. Koshiishi, "Development of experimental legged robot for inspection and disaster response in plants," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4869–4876.

[2] B. Lindqvist, S. Karlsson, A. Koval, I. Tevetzidis, J. Haluška, C. Kanellakis, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Multimodality robotic systems: Integrated combined legged-aerial mobility for subterranean search-and-rescue," *Robotics and Autonomous Systems*, vol. 154, p. 104134, 2022.

[3] J. Delmerico, S. Mintchev, A. Giusti, B. Gromov, K. Melo, T. Horvat, C. Cadena, M. Hutter, A. Ijspeert, D. Floreano *et al.*, "The current state and future outlook of rescue robotics," *Journal of Field Robotics*, vol. 36, no. 7, pp. 1171–1191, 2019.

[4] S. Ha, "Quadrupedal robots trot into the wild," *Science Robotics*, vol. 5, no. 47, p. eabe5218, 2020.

[5] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[6] D. Kim, D. Carballo, J. Di Carlo, B. Katz, G. Bledt, B. Lim, and S. Kim, "Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2464–2470.

[7] C. Zhang, J. Zhang, J. Wu, and Q. Zhu, "Vision-assisted localization and terrain reconstruction with quadruped robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 571–13 577.

[8] P. Arm, R. Zenkl, P. Barton, L. Beglinger, A. Dietsche, L. Ferrazzini, E. Hampp, J. Hinder, C. Huber, D. Schaufelberger *et al.*, "Spacebok: A dynamic legged robot for space exploration," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6288–6294.

[9] G. Picardi, M. Chellapurath, S. Iacoponi, S. Stefanni, C. Laschi, and M. Calisti, "Bioinspired underwater legged robot for seabed exploration with low environmental disturbance," *Science Robotics*, vol. 5, no. 42, p. eaaz1012, 2020.

[10] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An overview to visual odometry and visual slam: Applications to mobile robotics," *Intelligent Industrial Systems*, vol. 1, no. 4, pp. 289–311, 2015.

[11] M. O. Aqel, M. H. Marhaban, M. I. Saripan, and N. B. Ismail, "Review of visual odometry: types, approaches, challenges, and applications," *SpringerPlus*, vol. 5, pp. 1–26, 2016.

[12] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[13] 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.

[14] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza, "Low-latency visual odometry using event-based feature tracks," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 16–23.

[15] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 994–1001, 2018.

[16] W. Guan, P. Chen, Y. Xie, and P. Lu, "Pl-evio: Robust monocular event-based visual inertial odometry with point and line features," *arXiv preprint arXiv:2209.12160*, 2022.

[17] D. Kim, D. Carballo, J. Di Carlo, B. Katz, G. Bledt, B. Lim, and S. Kim, "Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2464–2470.

[18] T. Qin, J. Pan, S. Cao, and S. Shen, "A general optimization-based framework for local odometry estimation with multiple sensors," 2019.

[19] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[20] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. B. Benosman, "Hots: A hierarchy of event-based time-surfaces for pattern recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 7, pp. 1346–1359, 2017.

[21] J. Manderscheid, A. Sironi, N. Bourdis, D. Migliore, and V. Lepetit, "Speed invariant time surface for learning to detect corner points with event-based cameras," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 245–10 254.

[22] Y.-F. Zuo, J. Yang, J. Chen, X. Wang, Y. Wang, and L. Kneip, "Devo: Depth-event camera visual odometry in challenging conditions," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2179–2185.

[23] J. Hidalgo-Carrió, G. Gallego, and D. Scaramuzza, "Event-aided direct sparse odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5781–5790.

[24] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 225–234.

[25] A. Zihao Zhu, D. Thakur, T. Ozaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multi vehicle stereo event camera dataset: An event camera dataset for 3d perception," *arXiv e-prints*, pp. arXiv–1801, 2018.

[26] M. Grupp, "evo: Python package for the evaluation of odometry and slam." https://github.com/MichaelGrupp/evo, 2017.

[27] Y. Zhou, G. Gallego, and S. Shen, "Event-based stereo visual odometry," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1433–1450, 2021.

[28] T. Qin, S. Cao, J. Pan, P. Li, and S. Shen, "Vins-fusion: An optimization-based multi-sensor state estimator." https://github.com/HKUST-Aerial-Robotics/VINS-Fusion, 2019.

[29] A. iniVation, "Understanding the performance of neuromorphic event-based vision sensors," Tech. Rep., 2020.

[30] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv preprint arXiv:1909.06586*, 2019.

[31] P. Chen, W. Guan, and P. Lu, "Esvio: Event-based stereo visual inertial odometry," *arXiv preprint arXiv:2212.13184*, 2022.