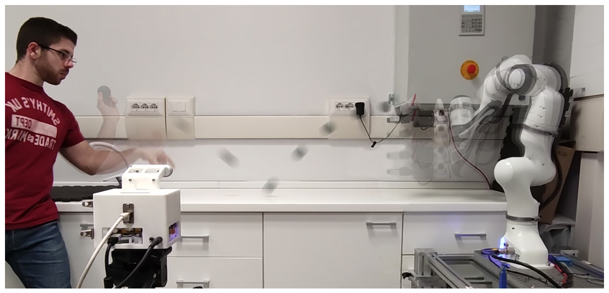
Fast Trajectory End-Point Prediction with Event Cameras for Reactive Robot Control

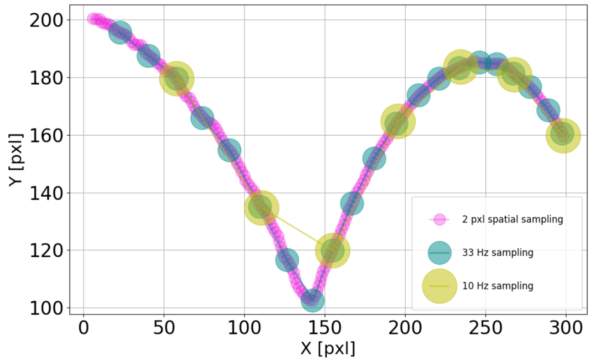
Marco Monforte, Luna Gava, Massimiliano Iacono, Arren Glover, Chiara Bartolozzi
Event-Driven Perception for Robotics
Istituto Italiano di Tecnologia, Italy
{marco.monforte, luna.gava, massimiliano.iacono, arren.glover, chiara.bartolozzi}@iit.it

Abstract

Prediction can be crucial for tasks with tight time constraints if a robot has limited speed and power. A low-latency, high-frequency perception system can reduce the time needed to converge on the expectation of the future state of the world, giving the robot additional time to act - or to choose a safer action. In this paper, we exploit event cameras for asynchronous motion-driven sampling, inherent data compression, and sub-millisecond latency to reduce the convergence time of a data-driven trajectory prediction algorithm. As a use-case, we use a Panda robotic arm to intercept a ball bouncing on a table. To predict the interception point as early as possible, and cope with the intrinsic variability of trajectory length - that cannot be defined a-priori for event cameras - we adopt a Stateful Long Short-Term Memory network, that asynchronously updates the prediction for each incoming point of the trajectory and does not require a predefined, fixed length input. We adopt a sim-to-real methodology in which the network is first trained on simulated data and then fine-tuned on real trajectories. Experimental results demonstrate that the dense spatial sampling performed by event cameras significantly increases the number of intercepted trajectories compared to a fixed temporal sampling typical of traditional "frame-based" cameras. Results motivate further exploration of the use of event cameras for prediction in higher-complexity robotic tasks.



(a)



(b)

Figure 1. The robotic arm intercepting a bouncing ball showing (a) the experimental set-up, and (b) various sampling strategies depending on sensor type and parameters. We compare 1. asynchronous event-driven spatial sampling (the target position is sampled every time it moves 2 pixels) with 2. synchronous fixed-rate 10 and 33 Hz.

1. Introduction

Human interaction with the environment strongly benefits from predictive capabilities [1–3]. The robustness, reliability, and precision of robots can be improved by implementing the ability to estimate the future state of the environment and moving agents within it. When the task has highly varying dynamics or short temporal duration, the robot has a narrow time window to act. Low-latency prediction leads to earlier action planning, improving the capabilities of affordable robots that do not have costly high-precision motors and finely engineered actuators.

Perception plays an important role in early prediction. We propose to exploit the high temporal resolution and low latency featured by *event cameras*. Differently from traditional cameras, they only transmit information in response to pixel-level changes - e.g. from the movement of an object. Data is produced proportional to the stimulus dynamics, rather than on a fixed independent clock. The data is inherently sparse and asynchronous, promoting energy saving, avoiding data redundancy, and yielding sub-millisecond latency [4]. The information encoding makes event cameras highly suitable for high-speed dynamic applications, like tracking and prediction [5].

We investigate the advantages of event-driven, asynchronous (*spatial*) sampling with respect to a standard frame-based (*temporal*) sampling, for predicting a bouncing ball's spatio-temporal location (a similar task as [6]). The proposed system predicts the final position of the ball in space and time within a robot's arm field of motion, which is used for interception, as shown in Fig. 1a. Accurate visual data early in the trajectory (e.g. within the first milliseconds) can lead to early estimation of the interception point, and hence the sensor choice is important for such tasks.

Differently from the model-based approaches typically used in the case of parabolic trajectories, where the physical model of the object trajectory is learnt [7–9] and its parameters are estimated [6], we use a model-free, data-driven approach, based on Long Short-Term Memory (LSTM) [10] state of the art recurrent neural networks, that were already proven to outperform other regression methods [11], with the aim of developing a framework that can be applied to tasks that have complex environmental interactions.

To fully investigate the potential advantages of event cameras, the LSTM must also be suited to effectively use the form and type of data generated by event cameras. To this aim, unlike in [11] where an Encoder-Decoder LSTM architecture was used, we resort to a Stateful LSTM to learn trajectory interception points. The advantage of the stateful architecture is that it does not have a fixed, *a-priori* defined input length, but instead keeps memory of past information indefinitely and continually improves its prediction from each input. This suits the event-driven paradigm, for which the number of data points cannot be defined in advance, as they depend on the speed and trajectory (e.g. the number of bounces) of the target. We pre-train on simulated trajectories and fine-tune the models with real-world data. We compare event-driven asynchronous spatial sampling, with fixed temporal sampling of the event steam, and also with a traditional RGB camera pipeline, as shown in Fig. 1b.

Finally, we consider two possible control strategies for the robotic arm: the first is to move as soon as the prediction converges, to minimize the velocity required to perform the movement; the second is to wait as long as possible, accumulating information to reduce the prediction error and move as fast as the robot can.

Our contributions are as follows:

- quantitatively compare event camera performance against frame-based performance for the prediction task, further validating event camera potential for computer vision and robotic systems;
- perform real-time and online experiments with an event camera and robot actuator in the loop to demonstrate the advantage of the low-latency vision for robots outside of drone experiments;
- propose the Stateful LSTM neural network for asynchronous event-based prediction, highly suited to event camera data;
- simulated trajectory datasets and code available to the community[1] for which, to the author's knowledge there is currently none.

We limit the scope of the experiments to a singular, specific, task, putting the focus on the comparison of event cameras against traditional cameras in a real robot control task. The task makes the following assumptions: (i) the ball is the only object moving in the scene and the camera is stationary; (ii) the trajectory of the ball is planar and parallel to the camera plane; and (iii) the robot has a single open-loop attempt to hit the ball. For (i), additional moving objects would increase the complexity of the tracker design, that is nevertheless not the focus of this paper. For (ii), 3D trajectories could be considered using a measurement of depth (e.g. stereo) and augmenting the datasets with such trajectories. Such an enhancement should be handled identically in the event camera domain, as for traditional camera implementation.

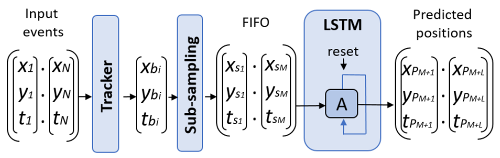
2. Related Work

Visual prediction with event cameras for robot control has been studied for a variety of problems with different proposed solutions. One of the classic scenarios showing the potential of combining event-driven vision with robot control is the robotic goalie of [12]. The goalie clearly shows advantages of event-cameras for the simple tracking scenario and 1-DoF robotic actuator. More recently, the catching of a ball in flight with a robotic arm has been tackled using a stereo method to track its 3D position and an Extended Kalman Filter (EKF) for predicting future states through numerical integration for a finite horizon [7, 8]. A combination of Support Vector Regression and EKF tracks markers on an object and predicts up to 1 s ahead at 200 Hz [9], enabling grasping of objects in flight. A drawback of EKF methods, however, is that all the predicted steps from the current instant must be recomputed every time a new measure arrives. In place of EKF or other

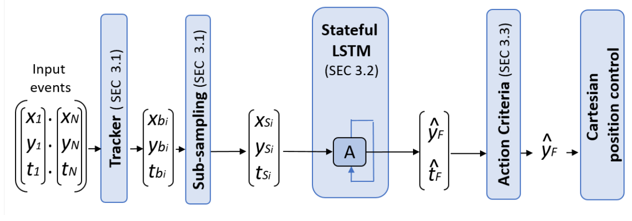
[1] https://github.com/event-driven-robotics/end-point-prediction

model-based methods, data-driven methods like neural networks have been used in [6, 13]. Zhao et al. [13] address the human-robot handover task, highlighting the importance of a well-timed robot movement for a person to perceive it as human-like. An LSTM network predicts the robot's next joints' configuration in response to human motion. In [14] a Variational Recurrent Neural Network (VRNN), combined with a dynamic model of the task, is capable of predicting the trajectory end-point of a ball bouncing on a table in the 2D frontal plane of a robot, which has to intercept it on the vertical axis at the end of its visual space [6]. The whole system is trained end-to-end in simulation and deployed in the real world. The fast convergence of the online estimated parameters allows probabilistic predictions even in case of blind spots along the trajectory. A single experiment lasts for $2-3$ s, with the system running at 20 Hz and refining the estimation frame after frame. All these works make use of powerful, external, and energy-consuming devices - like GPUs in [13], the VICON motion capture system in [8], or the IDP Express RF2000F system and the ad-hoc finger cameras in [7]. Problems arise with autonomous systems having limited batteries, or in outdoor tasks where motion capture cameras cannot be placed.

The role of perception on agents' actions' timing highlighted how the choice of the right sensing device becomes crucial for the success of a task [15]. Event cameras can leverage very high temporal resolution (for fast-moving targets), very high dynamic range ($\sim$140 dB) [4], and yet with low energy requirements. He et al. [16] propose a low latency, high precision pipeline for dynamic object avoidance with a quadrotor by fusing IMU data, depth, and events. The object's 3D trajectory is predicted by combining events and depth information to obtain the object's location in the depth image, and then estimated with a second-order polynomial. After an offline evaluation of the accuracy, online experiments show the quadrotor avoiding the ball under different scenarios. In [17] an event camera with an Encoder-Decoder [18] LSTM network predicts a handover task for an iCub robot. The pipeline predicts both spatial and temporal future points, allowing the robot to know in advance where to move, compensating for internal delays in the perception-action loop. In [19] the authors propose a grasping framework for eye-in-hand robotic manipulators endowed with event cameras, using both model-based and model-free multi-object grasping in clutter. While the former ensures higher precision, the latter is more general and applicable to real-world scenarios. Wang et. al [20] implemented a system for catching balls thrown by a tennis ball launcher with a 1-degree-of-freedom linear actuator. The proposed Binary Event History Image (BEHI) accumulates information from events in images that are fed to an impact prediction pipeline that controls the actuator motion timing. The actuator catches balls up to a top speed of 13m/s with a success rate of 80%.



(a)



(b)

Figure 2. Pipeline of the (a) previous work [11] and (b) proposed implementation. The previous work uses a many-to-many LSTM which requires an input buffer and predicts a number of upcoming points of the trajectory. The LSTM is reset between each network inference. In the proposed work, the Stateful LSTM accepts a single position and outputs a single interception position. The LSTM memory is reset only after an entire trajectory. The proposed work also includes on-line operation, with robot control and action decision criteria.

To date, however, there is still minimal investigation on architectures suitable for event camera data for time-series prediction and, as such, we propose a Stateful LSTM. In addition, the literature lacks experiments that directly compare event cameras to traditional cameras in real experiments with robotic actuators in the control loop, that concretely demonstrate the advantages of event cameras, especially in dynamic robot control scenarios. To investigate these gaps and focus on the prediction task without other confounding factors, we have proposed a set-up with reduced complexity, assuming a two-dimensional scenario without clutter, occlusions, or aliasing. The main objective is to validate the prediction step and the role of sampling strategies.

3. Methodology

The architecture, shown in Fig. 2b, comprises an ATIS [4] event camera placed near the robot, an asynchronous event-driven tracker [11] that localizes the target and tracks its centre of mass, a Stateful LSTM network [10] queried for each new tracked position and predicting the trajectory interception point in position and time, and a Franka Emika Panda robot manipulator controlled for interception.

3.1. ATIS and Asynchronous Ball Tracking

The ATIS [4] used has a resolution of 304×240 pixels. When a pixel is triggered by a brightness change, it emits an event $e =< t_i, x_i, y_i >$, containing the timestamp t_i and the spatial coordinates $< x_i, y_i >$[2].

The event-driven tracker [11] continuously receives events generated by the moving object as input. It asynchronously outputs its centre of mass position $< x_{b_i}, y_{b_i}, t_{b_i} >$ every time this moves, as shown in Fig. 3. An initial check on the collected events' distribution locates the object and initialises the tracker. Then, a buffer of N_{track} events is filled by looking into a Region Of Interest (ROI) containing the object and used to compute its centre of mass. The tracker updates the object position only when the target moves, thereby maintaining the asynchronous paradigm of the event camera.

A sub-sampling method outputs $< x_{s_i}, y_{s_i}, t_{s_i} >$ if the current tracked position $< x_{b_i}, y_{b_i}, t_{b_i} >$ is a fixed distance from the previous emitted position $< x_{s_{i-1}}, y_{s_{i-1}}, t_{s_{i-1}} >$. A *spatial sampling* of $D = 2$ pixels is used to avoid quantization error at the pixel level, without undermining the resolution of the sampled trajectory nor the accuracy of the future prediction. The spatial sub-sampling method is compared to *temporal sub-sampling* in the experimental section.

3.2. Stateful LSTM

Long Short-Term Memory networks maintain a temporal correlation between consecutive queries using recurrent connections and therefore are suited to process temporal trajectories. The asynchronous nature of the tracker is handled by also including the temporal information with each input.

Our previous implementation of LSTM for trajectory prediction followed a standard many-to-many approach in which a FIFO containing a fixed number (e.g. $M = 20$) of trajectory points was input to the network, which produced a dense prediction of the immediately following points of the trajectory (e.g. $L = 45$). In this work, we proposed two changes to the architecture to facilitate improved robot behaviour:

1. **Prediction of the interception point of the trajectory:** The prediction of the immediately following part of the trajectory is not informative of the point in space that the robot should use to plan its trajectory to intercept the target. The information would be available too late. Prediction of the full trajectory from the initial, to the interception point was less accurate compared with the prediction of only the final point and it requires the knowledge of the number of points in the trajectory in advance, which is not fixed in event cameras.

[2]polarity is discarded

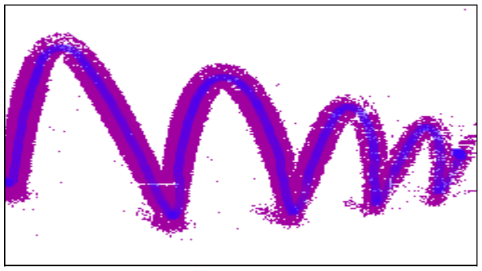


Figure 3. Superimposition of the events (in purple) generated by a ball bouncing in front of the camera and the tracker output (in blue) tracking the centre of mass.

2. **Using a Stateful LSTM:** Instead of using multiple inputs and resetting the network after each inference, each tracker position is input individually (and asynchronously), the memory is retained for the full trajectory, and the network is only reset after a complete trajectory. The Stateful LSTM is a more efficient implementation that better handles variability in the input length which comes with asynchronous data streams.

The architecture is comprised of 3 input neurons, $< x, y, \Delta t >$, corresponding to the position and the temporal difference from the previous tracker output. Inputs are normalised between -1 and +1. There are 2 output neurons that represent $< \hat{y}_F, \hat{t}_F >$: the predicted height and time of the interception point. The interception point $\hat{x}_F$ is fixed for our task. The network is defined by a single layer of N_H recurrent hidden neurons with a *tanh* activation function. Δt is used instead of an absolute t as the final prediction time, $\hat{t}_F$, is actually a Δt since the first tracker position is input to the network.

3.2.1 Training Data

An Unreal Engine environment (UE) was used to generate a total of 5400 synthetic trajectories (4000 for training, 1000 for validation, and 400 for testing). Each trajectory was comprised of a sequence of RGB frames generated by UE at 500 Hz and then fed to E-SIM [21] to generate a stream of events. Finally, the event tracker was run offline to generate $< x, y, t >$ spatial coordinates and timestamp of the ball's centre of mass. Synthetic trajectories were generated with a variation in initial position, and initial velocity, as well as ball bounce properties and air drag, as shown in Fig. 4

A second set of 310 real trajectories were gathered with the setup shown in Fig. 1a, using the ATIS camera and running the tracker offline to generate the ball trajectories. Initial training was performed on the simulated data, and fine-tuning was performed on the real data. The comparative statistics between the real and simulated trajectories are shown in Fig. 4, the simulated datasets covered a much

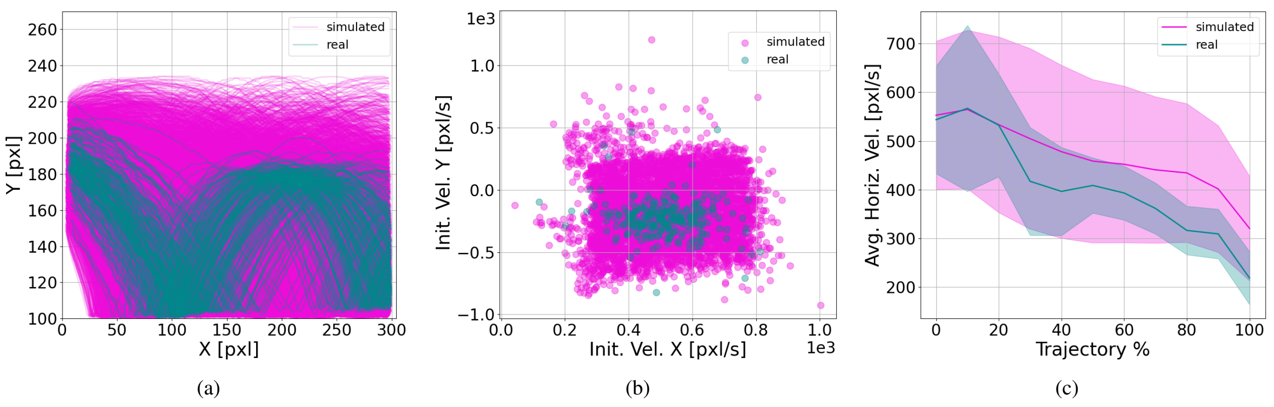


Figure 4. Validation of simulated trajectories, the simulated (magenta) and real (cyan) datasets are: (a) superimposed over the image plane; (b) showing the initial X and Y velocity components; and (c) showing the average velocity along the horizontal axis, representing the energy loss due to bounces and air drag in the simulation environment.

greater variation in parameters than the real datasets, which results in greater generalisation capability of the model.

3.3. Prediction convergence criteria and parameters

The Stateful LSTM provides a predicted output, $<\hat{y}_F, \hat{t}_F>$ for each asynchronous data input, $<x, y, \Delta t>$. With only a few input samples the uncertainty on the predicted point is high. We define a parameter γ to determine at run-time when the prediction has converged and, therefore, the first time at which action could be taken. This parameter is based on the rate of change of the previous N estimates of the final vertical position of the target y_F:

$$\gamma(i) = \frac{1}{N} \sum_{j=i-N+1}^{i} \frac{|\hat{y}_F(j) - \hat{y}_F(j-1)|}{t(j) - t(j-1)} \tag{1}$$

where t_i is the time at the i-th point, with $i = N, \ldots, M$, N depending on the sampling strategy and M is the total number of samples for the current trajectory. In such a way, we can define the *convergence instant* t_{conv} as the first moment such that $\gamma(i)$ is below a user-defined threshold γ^*:

$$t_{conv} = t_1 \in T = \{t(i) | \gamma(i) < \gamma^*\} \tag{2}$$

with t_1 being the first element of the set of time instants, T, where the condition on γ is satisfied. Moreover, knowing the average robot velocity v^{robot} and its starting position y^{robot}_{start}, we can compute the time needed to reach any point in the task space. This value, along with the final time $\hat{t}_F$, is used to compute the last moment the robot can wait before moving $\hat{t}_{dec}$, to reach $\hat{y}_F$:

$$\hat{t}_{dec}(i) = \hat{t}_F(i) - \frac{y^{robot}_{start} - \hat{y}_F(i)}{v^{robot}} \tag{3}$$

Fig. 5 shows an example trajectory, the temporal window in which the robot can move (between t_{conv} and $\hat{t}_{dec}$), and γ.

To comply with constraints on robot joint acceleration and velocity, and to estimate the usefulness of timely and accurate prediction of the end-point of the trajectory, the control is implemented in open loop, using only a single value of $<\hat{y}_F>$. The robot can then either move as soon as the prediction converges, at t_{conv}, which gives the more time to reach the target, and a slower movement can be performed. Alternatively, motion can be performed at, $\hat{t}_{dec}$, with a more accurate estimate of the interception point, $\hat{y}_F$, but must move at maximum speed.

4. Experiments and Results

Experiments are divided into offline (with the simulated datasets) and online (with the tracker receiving live data from the camera and the robot receiving live input from the prediction network refined on the real dataset). The former is used for a quantitative assessment of the system, using the available ground truth; the latter to quantitatively assess the overall system performance in the task.

In each experiment, the tracker receives asynchronous raw events from the event camera. The target position estimate is updated asynchronously and the output of the centre of mass of the tracked target can be sent to the input of the Stateful LSTM with different sampling strategies (with different LSTM models trained for each):

- *Spatial sampling* (dubbed *events*): output asynchronously after the target centre of mass moves by a fixed pixel distance. [11] informs a choice of 2 pixels for these experiments.

- *Temporal sampling* (dubbed *events10Hz* and *events33Hz*): the tracker output is sent at fixed time intervals, similarly to traditional camera sampling strategies.

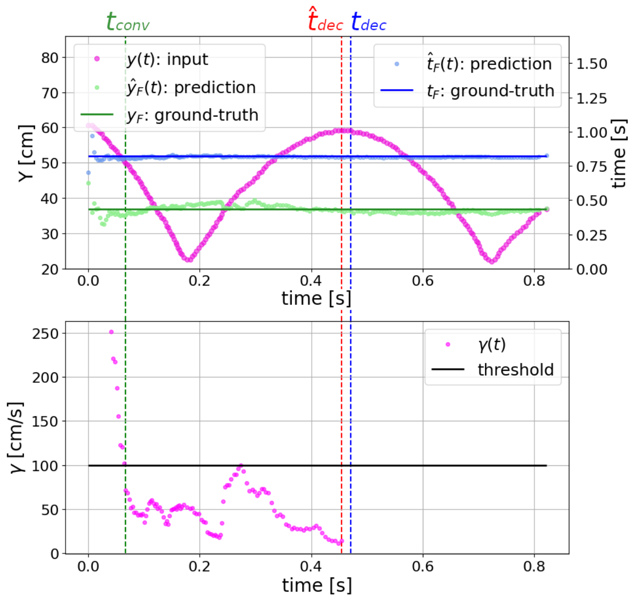


Figure 5. Single trial ground truth and prediction of the final y coordinate of the ball over time: γ is the average rate of change of the previous N predictions. The first time γ goes under a threshold defines t_{conv}, and the last moment the robot can wait to move defines $\hat{t}_{dec}$ (and is determined by the maximum robot's acceleration and distance to the estimated target end-point). The robot can act only in the time window between these two instants and if γ is below the threshold.

In addition, we use a RealSense D435i camera set to 33Hz and 60Hz as a frame-based comparison. A colour segmentation algorithm from OpenCV [22] was used to perform robust tracking of the dark ball against the light background. The experiment was designed to enable a simple tracking method in order to minimise latency and give as much advantage to the traditional vision system for comparison.

Models were trained with hyper-parameter optimization, using a Mean Squared Error (MSE) loss function. The Adam optimization algorithm was used with standard parameters, except for the initial learning rate, which was set to $\alpha = 0.01$ and enabling the learning rate decay policy. A single hidden layer with $N_H = 350$ neurons was found to be the appropriate network size for all three sampling strategies.

The 7-DoF Panda robot performs the interception. Intermediate points between the initial and final position are generated with a quintic polynomial, to limit both the initial velocity, final velocity, and acceleration. Under such limits, the vertical range assumed for the task is 60 cm, viable in 1.1 s. The calibration procedure was a quadratic regression to map pixel coordinates to the robot Cartesian position uses 10 iterations of 8 pairs acquisitions of $(< x_p, y_p >, < y_r >)$ where p indicates pixel space and r indicates robot reference frame. A successful interception is considered when the ball (of radius 2.5 cm) makes contact with the robot gripper, which has a size of 2 cm. An error threshold of $\hat{y}_F$ 3.5 cm was used based on these sizes.

Inference was performed on a Intel i7-9750H CPU.

4.1. Offline Experiments

Each of the three models trained on events (*events*, *events10Hz*, and *events33Hz*) was evaluated on 400 offline trajectories. For each testing execution, the model produced the trajectory over time and the error in predicted position $\hat{y}_F$ and time $\hat{t}_F$ for a fixed final x_F were recorded, as shown in Fig. 6.

On average *events* converged to the correct estimate of the target final position earlier than *events10Hz* and *events33Hz* - after approximately 8%, 12%, and 30% of the trajectory was observed, respectively. However, there were large variances for all sampling strategies, indicating trajectory type affected the convergence time.

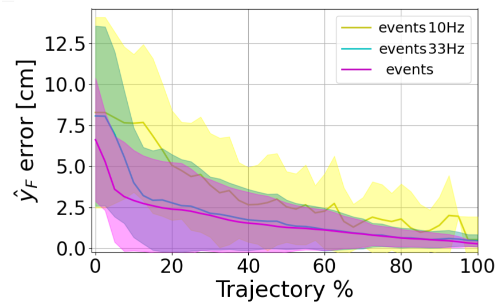
All sampling strategies predicted convergence time similarly as there is a direct relationship between initial ball speed and arrival time (while bounces cause non-linear relationships for $\hat{y}_F$). *events33Hz* had a significantly higher error over the full trajectory compared to the other two methods.

Considering the interception strategy defined in Section 3.3 trajectory prediction can be analysed in terms of interception success, comparing convergence time, and the time taken for the robot to reach the interception position, as shown in Fig. 7. Given the average velocity of the Panda robot at t_{conv} and setting the parameter N of equation 1 to 15 for *events*, 3 for *events33Hz*, and 1 for *events10Hz*, the target can be intercepted by the robot only if $t_{conv} < t_{dec}$. Fast convergence (i.e. a low t_{conv}) places the prediction high above the black line in Fig. 7. Late convergence (failure) is indicated by samples below the black line: 10 for *events*, 19 for *events33Hz*, and 46 *events10Hz*.

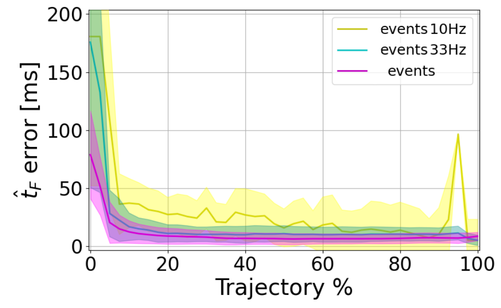
4.2. Online experiments

Trained networks were refined on the 310 real trajectories with up to 500 additional epochs (managed by the learning rate decay policy), to remove sim-to-real inconsistencies. The ball is launched by a human operator toward the robot interceptor and the tracker detects when the ball enters the field of view. Simultaneously the prediction model begins to produce interception positions. The robot is moved at t_{conv} or $\hat{t}_{dec}$, as defined in the criteria in Section 3.3. 50 trials are performed using each of the 3 event-based models and the 2 frame-based models, for each of the motion criteria resulting in a total of 500 trials.

Convergence of *events* is compared to *frame60Hz* in Fig. 8a, indicating also if convergence did not occur.



(a)



(b)

Figure 6. Run-time prediction average error (thick line) and standard deviation (band) of the target ending point: (a) Final height $\hat{y}_F$ and (b) final time instant $\hat{t}_F$ prediction error over trajectory execution for the three sampling strategies. Trajectories are normalised (trajectory %) for comparison as each trajectory has a different number of samples.

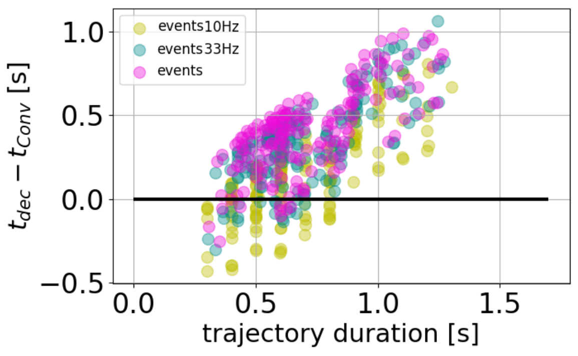
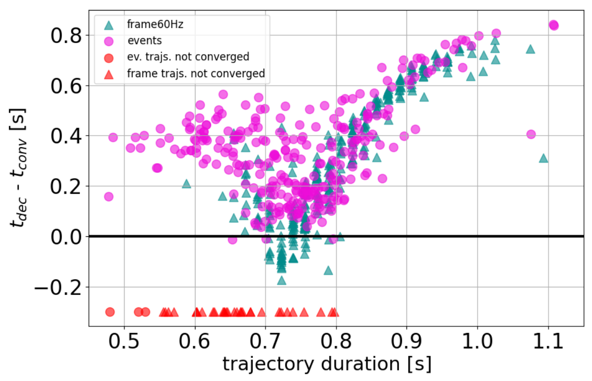
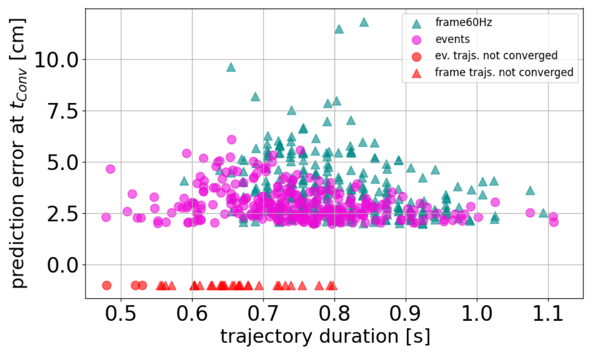


Figure 7. The success of the task is closely linked to the promptness and accuracy of the prediction, through $\hat{t}_{conv}$, but also to the robot velocity, through $\hat{t}_{dec}$. For a trial to be successful, $\hat{t}_{conv}$ must be earlier than $\hat{t}_{dec}$. This condition becomes harder to meet the faster the trajectory and the lower the sampling rate.

- Slower trajectories (between 0.8 s and 1.1 s): a similar convergence occurs for both methods, with the gap between t_{conv} and $\hat{t}_{dec}$ decreasing linearly.
- Fast trajectories (0.65 s and 0.8 s): the majority of *events* converge successfully (non-failures and above the black line), while a significant number of *frame60Hz* trials (31 trials) converge late (below the black line) or not-at-all (8 trails).
- Fastest trajectories (less than 0.65 s): the majority of *events* converge successfully while the majority of *frame60Hz* do not converge.



(a)



(b)

Figure 8. (a) Temporal distance between t_{conv} and t_{dec} for the motion-driven sampling strategy of the event camera and the 60 Hz time-driven strategy of the RealSense. Values below 0 represent predictions that either converged too late for the robot to move, or did not converge at all (red marks). (b) Prediction error at t_{conv}.

In addition, the prediction error, shown in Fig. 8b, is on average lower for the *events* approach. The result indicates that for fast trajectories a temporal sampling strategy does not collect enough data early enough for quick reaction tasks, and the event-driven technologies instead have a benefit.

The difference between moving the robot at t_{conv} and $\hat{t}_{dec}$ is shown in Fig. 9, in which the interception is improved by approximately 3.5 cm and 6 ms by waiting till $\hat{t}_{dec}$. The same difference in terms of interception success is shown in Fig. 10, in which, on average, the $\hat{t}_{dec}$ (rightmost 5 bars) achieves more successes. However, moving at $\hat{t}_{dec}$ requires maximum speed but moving at t_{conv} can be performed at a slower and safer velocity.

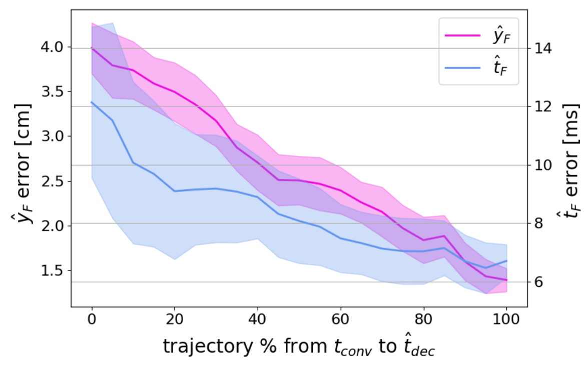


Figure 9. Average error and standard deviation from t_{conv} to $\hat{t}_{dec}$ for the event-driven model.

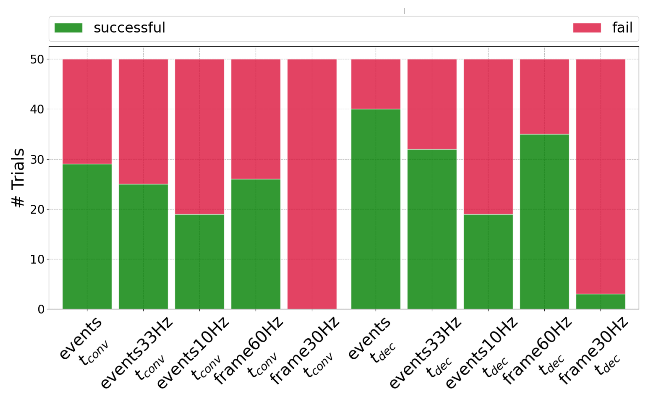


Figure 10. Tracking and prediction performance translate into different numbers of successful trials. Overall, the *events* sampling approach is the most accurate. Moving at t_{conv} leads to lower accuracy, resulting in a higher number of failures.

The *events* model achieves the most robot interception successes using either strategy, with both *events33Hz* and *frames60Hz* slightly less but comparable success rates, as shown in Fig. 10.

The temporal-sampling method of event-based data, *Events10Hz*, achieves significantly lower successes as, while the neural network predicts correctly, the low sampling rate can completely miss key features of the trajectories such as bounce inflection points. *events33Hz* achieves a comparable success to *events*, however, if the velocity increases further, also *events33Hz* should begin to fail more often; while *events* is instead speed invariant.

In regards to traditional camera methods, the *frames33Hz* approach achieved very few successes due to tracker failure from image motion blur, as the ball moved across multiple pixels during the aperture open period. As the trajectory velocity increases, or as illumination decreases, motion blur will also affect the *frames60Hz*; giving an added advantage to event-based cameras.

The latency of the event-driven tracker and prediction is on the order of < 5 ms an order of magnitude smaller than the timescale required for the robot to reach the goal position.

5. Conclusions

In this work, we investigated the advantages of using an event camera for predicting the end-point of a target's trajectory to be intercepted with a robotic arm. To do so we defined a singular task with a limited scope. However, we also chose a data-driven method that could be adapted to other tasks, as long as training data is available. To this end, we also demonstrate the use of simulated data for pre-training the network, which could also be generated for other tasks. We demonstrate the use of a Stateful LSTM for the prediction task, which is highly compatible with the asynchronous event-driven tracking output. Compared to other approaches [20], system inference runs smoothly and asynchronously without using a GPU. Data transfer to the GPU often imposes a minimum bound on the processing rate, leading to fixed-frequency processing.

Results indicated that the purely event-driven approach led to faster convergence, and lower end-point error, of the predicted interception positions, compared to fixed-rate sampling of event-driven data, and (inherent fixed-rate sampling when using) a traditional RGB camera. Faster convergence resulted in a greater number of successes of real-time, on-line, robot interception experiments. Motion blur caused the traditional camera to fail when using a shutter speed associated with 30 Hz acquisition rate.

Additional benefits of the event-driven camera over traditional cameras for such a task include a lower data transfer, as full frames are not acquired, but only changed pixel positions. As such a lower latency can be achieved leading to earlier real-time tracking and prediction. Further benefit should also be observed in low-light conditions in which a traditional camera requires a longer shutter time, and therefore more motion blur occurs. Additional experimentation is required to quantitatively analyse and understand the extents of these benefits.

To take such a system into a real application, target depth/position in 3D is probably required but should not invalidate the results comparing frame-based and event-based cameras that were investigated. Additionally, an improved closed-loop control strategy could be implemented to continually move the robot arm to the best predicted position.

References

[1] M. Bar, "The Proactive Brain: Memory for Prediction," *Phil. Trans. R. Soc. B*, vol. 364, pp. 1235–1243, 2009.

[2] N. W. Roach, P. V. McGraw, and A. Johnston, "Visual Motion Induces a Forward Prediction of Spatial Pattern," *Cur-*

rent Biology, vol. 21, no. 9, pp. 740–745, 2011.

[3] D. L. Schacter, D. R. Addis, and R. L. Buckner, "Episodic Simulation of Future Events," *Annals of the New York Academy of Sciences*, vol. 1124, no. 1, pp. 39–60, 2008.

[4] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, 2011.

[5] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, 2022.

[6] M. Asenov, M. Burke, D. Angelov, T. Davchev, K. Subr, and S. Ramamoorthy, "Vid2Param: Modeling of Dynamics Parameters from Video," *IEEE Robot. Autom. Lett.*, 2020.

[7] M. Sato, A. Takahashi, and A. Namiki, "High-speed catching by multi-vision robot hand," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9131–9136, 2020.

[8] K. Dong, K. Pereida, F. Shkurti, and A. P. Schoellig, "Catch the ball: Accurate high-speed motions for mobile manipulators via inverse dynamics learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6718–6725, 2020.

[9] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Trans. Robot.*, 2014.

[10] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] M. Monforte, A. Arriandiaga, A. Glover, and C. Bartolozzi, "Exploiting event-driven cameras for spatio-temporal prediction of fast-changing trajectories," *2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS2020)*, March 2020.

[12] T. Delbruck and M. Lang, "Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor," *Front. Neurosci.*, 2013.

[13] X. Zhao, S. Chumkamon, S. Duan, J. Rojas, and J. Pan, "Collaborative Human-Robot Motion Generation Using LSTM-RNN," in *IEEE-RAS Int. Conf. Humanoid Robot.*, 2019.

[14] J. Chung, K. Kastner, L. Dinh, K. Goel, A. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, (Cambridge, MA, USA), p. 2980–2988, MIT Press, 2015.

[15] D. Falanga, S. Kim, and D. Scaramuzza, "How Fast Is Too Fast? the Role of Perception Latency in High-Speed Sense and Avoid," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1884–1891, 2019.

[16] B. He, H. Li, S. Wu, D. Wang, Z. Zhang, Q. Dong, C. Xu, and F. Gao, "Fast-dynamic-vision: Detection and tracking dynamic objects with event and depth sensing," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3071–3078, 2021.

[17] M. Monforte, A. Arriandiaga, A. Glover, and C. Bartolozzi, "Where and when: Event-based spatiotemporal trajectory prediction from the icub's point-of-view," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9521–9527, 2020.

[18] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Adv. Neural Inf. Process. Syst.*, vol. 4, pp. 3104–3112, 2014.

[19] X. Huang, M. Halwani, R. Muthusamy, A. Ayyad, D. Swart, L. Seneviratne, D. Gan, and Y. Zweiri, "Real-time grasping strategies using event camera," *Journal of Intelligent Manufacturing*, vol. 33, pp. 593–615, 2022.

[20] Z. Wang, F. C. Ojeda, A. Bisulco, D. Lee, C. J. Taylor, K. Daniilidis, M. A. Hsieh, D. D. Lee, and V. Isler, "Ev-catcher: High-speed object catching using low-latency event-based neural networks," *IEEE Robotics and Automation Letters*, vol. 7, pp. 8737–8744, Oct 2022.

[21] H. Rebecq, D. Gehrig, and D. Scaramuzza, "ESIM: an open event camera simulator," *Conf. on Robotics Learning (CoRL)*, Oct. 2018.

[22] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.