2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Where and When: Event-Based Spatiotemporal Trajectory Prediction from the iCub's Point-Of-View

Marco Monforte, Ander Arriandiaga, Arren Glover and Chiara Bartolozzi

Abstract— **Fast, non-linear trajectories have been shown to be more accurately visually measured, and hence predicted, when sampled spatially (that is when the target position changes) rather than temporally, i.e. at a fixed-rate as in traditional frame-based cameras. Event-cameras, with their asynchronous, low latency information stream, allow for spatial sampling with very high temporal resolution, improving the quality of the data and the accuracy of post-processing operations. This paper investigates the use of Long Short-Term Memory (LSTM) networks with event-cameras spatial sampling for trajectory prediction. We show the benefit of using an Encoder-Decoder architecture over parameterised models for regression on event-based human-to-robot handover trajectories. In particular, we exploit the temporal information associated to the events stream to predict not only the incoming spatial trajectory points, but also when these will occur in time. After having studied the proper LSTM input/output sequence length, the network performance are compared to other regression models. Then, prediction behavior and computational time are analysed for the proposed method. We carry out the experiment using an iCub robot equipped with event-cameras, addressing the problem from the robot perspective.**

I. Introduction

Humans rely on prediction for a wide range of tasks: to avoid approaching obstacles, shake hands, or catch a falling object. Such interactions would not be possible without an estimate of the future state of the target. Likewise, for autonomous systems operating in an unconstrained environment, being capable of predicting changes in the scene and plan actions in advance plays a fundamental role in the success of the task execution. In several fields, such as human-robot collaboration, virtual reality applications, gesture recognition and other types of real-time human-machine interaction, latency between detection to execution plays a crucial role. Systems are required to be highly responsive, with delays in the order of 100-200 ms [1], [2] to be satisfactory. The perception system should therefore be as fast as possible to give more time for action planning and execution.

Typically, prediction requires a tracking algorithm such that past and current state can be known. The perception pipeline must include both tracking and prediction components. Traditional cameras operate at a fixed temporal rate - usually 30 fps - and produce images regardless of whether there has been a change in the scene or not. This leads to two main drawbacks. The first is the high data redundancy in consecutive images, resulting in waste of computational and communication resources, that should be instead optimised in autonomous robots with on-board processing. The second is the strict dependency of the whole system on the frame rate. The constant interval needed to acquire a new image defines the maximum rate for the robot to operate, which can be a limiting factor in highly dynamic environments. A fixed-rate visual pipeline is therefore not ideal for a low latency prediction system.

Event-cameras, instead, output a sparse, asynchronous signal whenever a relative brightness change is perceived from a pixel at sensor level. The data representation avoids redundancy and has a high temporal resolution and dynamic range - respectively in the order of 15 μs and >120 dB [3]. The quantity of information available from the camera is directly proportional to the dynamics of the scene: fast stimuli produce a large amount of events, making applications like object tracking more robust; slow stimuli produce few events, avoiding data redundancy. The event-driven pipeline is fundamentally low-latency: prediction can begin after only a few events are transmitted, as opposed to several entire frames of pixels, and processing is not wasted on unchanged areas of the scene.

In previous work [4] we described a trajectory prediction system that combines event-cameras, a simple tracking algorithm, and a Long Short-Term Memory (LSTM) architecture for prediction, demonstrating that the asynchronous nature of the event-data enables a spatial sampling strategy that simultaneously reduced the computational load and prediction error. Differently from temporal sampling, where a sample is produced at every time step, in spatial sampling a sample is produced every time the target moves of a given amount in space. Key aspect of the spatial sampling framework is that time became an input variable of the LSTM network, providing additional information - with respect to frame-based systems - about the temporal dynamics of the trajectory. This information can be used to predict not only *where* the tracked object is going, but also *when* it will pass at the predicted location. While in [4] we examined the trajectory of a bouncing ball, in this work we address a more complex motion pattern: a person is handing an object to the iCub robot [5]. In this case, the robot must predict where the object is going, in order to intercept it smoothly with its own grasping strategy. With respect to the previous work, the tracked trajectory is more noisy due to the more complex visual scene. We therefore integrate into the pipeline a tracker robust to clutter [6]. This tracker is

M. Monforte is with Università degli Studi di Genova and EDPR Research Line, Istituto Italiano di Tecnologia, Italy; A. Arriandiaga, A. Glover and C. Bartolozzi are with EDPR Research Line, Istituto Italiano di Tecnologia, Italy. {marco.monforte, ander.arriandiaga, arren.glover, chiara.bartolozzi}@iit.it

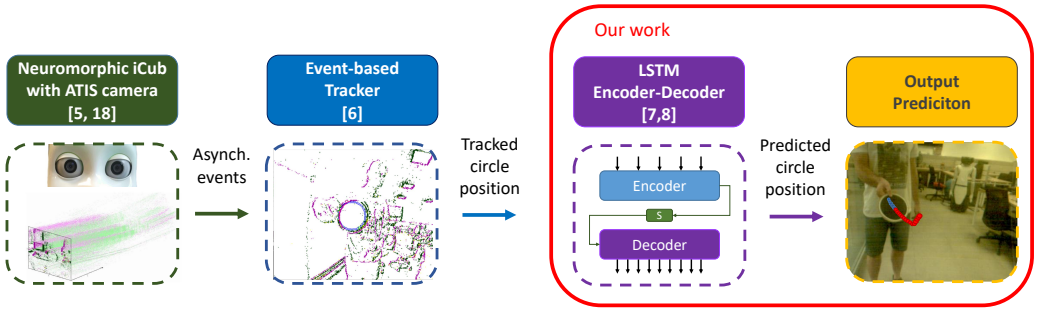


Fig. 1: Full pipeline: the output of event-cameras is fed into an event-based tracker. Our contribution consists of the LSTM encoder-decoder architecture that handles asynchronous event-based data and predicts the future trajectory of the target.

event-driven and is based on spatial sampling, in that the output of the tracker is triggered by a change in position of the target, allowing for the implementation of an asynchronous predictor as in [4]. With respect to the cited work, however, the handover trajectory is more "unpredictable", as the person does not always have a consistent movement pattern (e.g., the trajectory can end closer to or farther from the robot). Our hypothesis is that in this scenario learning is a crucial element to achieve good prediction results. We therefore compare a state-of-the-art sequence-based learning algorithm, the LSTM, against model-based regression prediction, assuming linear, quadratic and sinusoidal motion patterns to model and predict from the noisy trajectory data. The pipeline is also measured in terms of computational time, towards low-latency prediction systems for the iCub robot. Addressing the problem from the iCub point-of-view is challenging as we do not rely on external systems that can simplify the problem, but only on the on-board camera. This limits the field of view, and therefore the amount and quality of information obtainable, but is a problem that must be addressed to develop fully autonomous robots relying only on on-board sensors.

In the last decades, researchers have applied deep learning to a wide spectrum of problems, often outperforming pre-existing algorithms and determining new state-of-the-art levels. Long Short-Term Memory networks, originally proposed in 1997 in [7], captured the community attention thanks to their ability of taking into account long-term dependencies without suffering the vanishing gradient problem. Unlike other methods, such as Fully-Connected (FC) and Convolutional Neural Networks (CNN), that produce independent outputs at two consecutive time instants, LSTM nets introduce a temporal dependency among the outputs by keeping a sort of "memory" of their state after each interrogation. The LSTM Encoder-Decoder architecture, also known as Sequence-to-Sequence [8] allows to map fixed size input sequences to output sequences with different length. Its success derives mostly from its application to Natural Language Processing (NLP), and in particular to machine translation (text-to-text) [8], speech recognition (audio-to-text) [9] and image/video captioning (image-to-text) [10]. Another application has been the trajectory prediction of surrounding vehicles proposed in [11], where the authors combined Encoder-Decoder LSTM and *beam search* to continuously predict a set of K most likely paths. With the growing interest for neuromorphic and spiking computation, event-cameras have been used for several tasks characterised by high dynamics such as camera pose relocalisation [12], visual-inertial odometry [13], gesture recognition [14], tracking [6], [15] and trajectory estimation [13]. In this work, the authors aimed at estimating the position of the vision sensor in space, parameterising the trajectory by means of B-splines. Event-based vision has been used also for predicting the steering angle of an autonomous vehicle [16]. Here, classical deep learning methods have been applied on "frames of events", analysing the performance of the networks under different light conditions and event integration time. In [17] low latency has been exploited for high-speed quadrotors mounting event-cameras in order to sense potential obstacles and perform avoidance maneuvers.

II. Methodology

This work considers a handover task, in which a person is handing an object to the robot and the robot must grasp the object at the optimal location and at the right time, in order to achieve smooth human-robot interaction. We focus on the trajectory prediction, exploiting existing work on tracking [6] and leaving the robot control for future work. To solve the prediction problem, given the current position and past positions of a target, the most likely future trajectory is estimated. The visual pipeline, shown Fig. 1, consists of the following components:

- The ATIS [3] event-camera mounted on iCub;
- The particle filter circle tracker [6] that estimates the object trajectory from the raw events from the camera;
- The LSTM-based prediction system that takes the current and past positions of the target and estimates the future points in space and in time.

a) Event-camera: Whenever light in the scene changes beyond a certain threshold, events are produced only from those pixels where the change happened. The information is a vector containing the $< x_i, y_i >$ pixel location of the i-th event in the image plane, the timestamp t_i with microsecond resolution and the polarity p_i of the change (describing

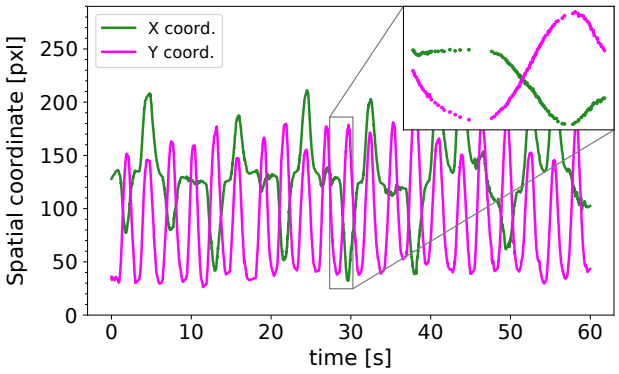


Fig. 2: Dataset: one minute (over the total 40) of the tracking algorithm output, as the target object is handed to the robot and then retracted. The inset shows the spatial sampling strategy of event-based data with more samples for fast-motion, and less samples when the target is slowing down.

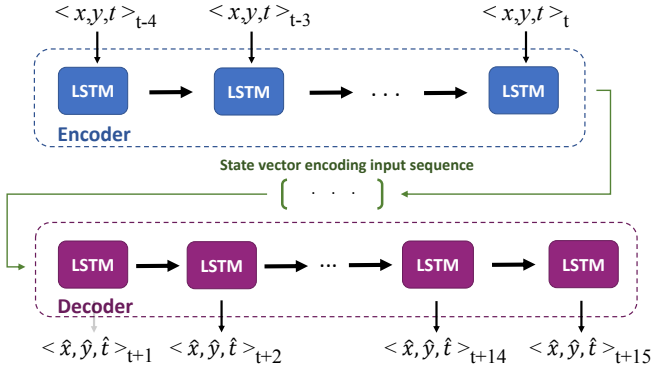


Fig. 3: LSTM Encoder-Decoder: it consists of an encoder, that receives an input sequence of length w_{in}, followed by a decoder that outputs a sequence of length w_{out}

whether the light intensity has increased or decreased). The resolution of the camera is 304×240 pixels. The event-camera is integrated into the neuromorphic iCub [5], using the event-driven library [18] integrated with YARP - the iCub middleware.

b) Visual tracking: To track the position of the object (a round paddle) handed to the robot, we applied a known, robust approach [6] tuned to circles that rejects clutter generated by the robot's ego-motion and other moving agents. The position $< x_c, y_c >$ of the target and the associated event timestamp t_c are updated by the tracker every time the target moves of a set amount of pixels, implementing a spatial sampling. As in [4], we choose a spatial sub-sampling of 3 pixels.

c) Prediction baselines: Prediction is, at a basic level, a regression task: from the current and past data available, plus some understanding (i.e. a model) of the system, the future states of the system can be extrapolated. However, regression methods need to be based on a good model of the process and might not scale to more complex scenarios. In this work, we compare the LSTM-based predictor to several hand-tuned regression baselines, with the aim to understand if LSTM is a good approach to the task and, in particular, with the event-driven data. From an inspection of the data, see Fig. 2, we chose a linear, a quadratic, and a mixed-sinusoidal models. Given w_{in} past points (including the current point) we fit each model and use it to predict the position future instants and spatial points of the trajectory, always assuming that time increases linearly. A **linear** model $x(t) = a_{1,x}\ t + a_{0,x}$ and $y(t) = a_{1,y}\ t + a_{0,y}$ is perhaps unsuited to the data, especially at "turning points", but is used as an absolute baseline of prediction. A **quadratic** function $x(t) = a_{2,x}\ t^2 + a_{1,x}\ t + a_{0,x}$ and $y(t) = a_{2,y} t^2 + a_{1,y}\ t + a_{0,y}$ should better model the "turning points" (assuming they are parabolic), and, in general, better model non-linear data. The parameters $a_{i,x}, a_{i,y}$ are the coefficients optimised by the fitting procedure. Finally, as the y-axis shows a periodic motion, we used a **mixed-sinusoidal** model $y(t) = A \sin(2\pi f t + \phi) + B$. In this case, we use a linear model for the temporal axis and the x-axis. The amplitude A, the frequency f, the phase shift ϕ and the offset B are the parameters optimised with a least squares regression procedure. Each of the baseline models is recalculated for each prediction point: when the predictor is queried, the past w_{in} points are taken, the model parameters are estimated (with either polynomial or least-squares regression) and the future points are predicted given the model as it is fit to the data. The number of input points, w_{in}, can be as low as 5 for the linear model, but has to be higher in the parabolic model (15 points) and in the mixed-sinusoidal model (100 points), in order to better capture the trajectory shape.

d) LSTM Encoder-Decoder: LSTM was adopted as the state-of-the-art in recurrent neural networks (RNN) in order to solve the prediction problem. RNNs fully take advantage of the temporal information associated with the events by exploiting the sequential information coming from queries over time. In this work, we introduce a LSTM compatible with the asynchronous event data by directly feeding the time as input feature to the network. As the event-camera is not polled at a fixed-rate, and the tracker follows the same sampling strategy, the time between sample points is not constant, hence the LSTM is asynchronously queried and the input state is $< x, y, dt >$, where $< x, y >$ is the point on the trajectory and $< dt >$ is the time period between sample points. We use an Encoder-Decoder architecture [8] that consists of two main components, as shown in Fig. 3. The Encoder is fed with the input data and attempts to represent the information with a lower dimensionality state vector. The output of the encoder is then fed into the Decoder. The Decoder is queried and, from its initial state, produces the output sequence associated to the Encoder input. The separation in two sub-networks enables the input sequences of length w_{in} to be different from the output sequence length, w_{out}, which becomes the predicted trajectory. Any time the network is queried, it is fed with the most recent w_{in} points, and the predicted trajectory of length w_{out} is obtained.

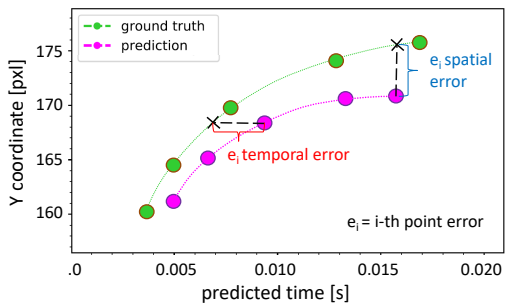


Fig. 4: The error is measured between the predicted trajectory and the ground-truth as a spatial error (at a fixed time) and a temporal error (at a fixed location).

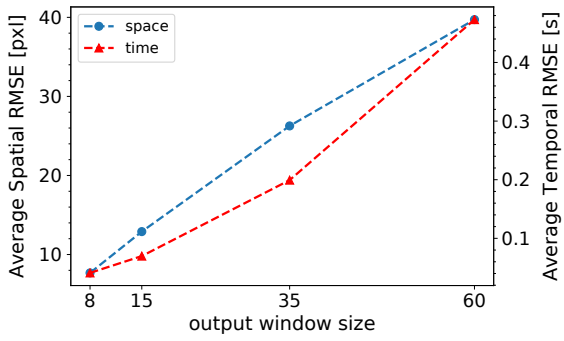
III. Experiments and Results

a) Dataset: A dataset of 40 minutes was created as a subject moved the paddle towards and away from the robot, as if periodically handing an object to the robot. The paddle trajectory was always in the field of view of the camera and the output of the tracker $< x_c, y_c, t_c >$ was recorded. As the camera was not moving, the data mostly consists of the paddle and the person moving. However, as the tracker is robust to clutter, the data processed by the predictor is limited to the trajectory of the target itself, excluding the events generated by the person. A 1 minute section of the dataset is shown in Fig. 2. Backward movements, where the human brings back the object away from the robot, were also included in the dataset, as it is important to understand human interactions beyond that of only giving the object. Understanding that the robot should not grasp under such circumstances is equally important.

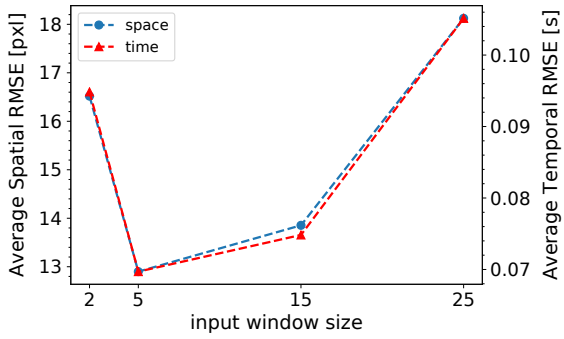
b) LSTM network and training parameters: The spatial sampling of the tracker was 3 pixels, i.e. the tracker sent the location of the target to the input of the LSTM network every time the target position changed of 3 pixels. In the following, a single "point" corresponds to a movement of 3 pixels. The architecture considered has an input layer with 3 neurons, 2 to input the (X,Y) coordinates and one for the time interval dt between the events, 25 neurons for both the Encoder and the Decoder networks, and 3 neurons in the output layer, in order to predict future (X,Y) spatial coordinates and the relative estimated times of arrival. To train the model, the full 40 minute recording was first windowed accordingly to the w_{in} and w_{out} values. Then, the total amount of sequences was shuffled and divided in 70% training and 30% validation. The final test was performed on a third dataset of 60 seconds. The network was trained using the Adam optimisation algorithm with learning rate $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. For regression, the Mean Squared Error (MSE) loss function was adopted. The training lasted for 200 epochs using batches of size 128. No dropout was required.

A. LSTM input-output characteristics

We characterised the system performance in terms of prediction accuracy for different w_{out} and w_{in}. The model is continuously queried inputting the latest w_{in} points received from the tracker and outputs a sequence of w_{out} points, representing the estimated future positions in the image plane and their temporal distance from the current time. We define a sequence-to-sequence error measuring how much the predicted sequence overlaps with the ground truth. Prediction error - obtained as average error over the entire dataset - was measured separating space and time, as described in Fig. 4. The error increases linearly with w_{out}, i.e. further into the future, as shown in Fig. 5a. As such there is no optimal output value to use, and instead the value can be chosen to trade-off the acceptable error and the required prediction look-ahead. Fig. 5b shows that the length of the input sequence, w_{in} has an optimal point at 5 points (a movement of 15 pixels), as too few points do not contain enough information and too many points lead to an increased error.



(a)



(b)

Fig. 5: Prediction error for (a) different w_{out} values assuming $w_{in} = 5$ points and (b) different w_{in} values assuming $w_{out} = 35$ points.

B. Baseline comparison

We compared the LSTM performance against all regression models. The parameters for the LSTM were set to be $w_{in} = 5$ (15 pixels) and $w_{out} = 15$ (45 pixels), as it is approximately one third of the maximum swing along the y-axis (as shown in Fig. 2), offering a good compromise between prediction window and prediction error. Fig. 6 shows example predictions for the LSTM and all regression

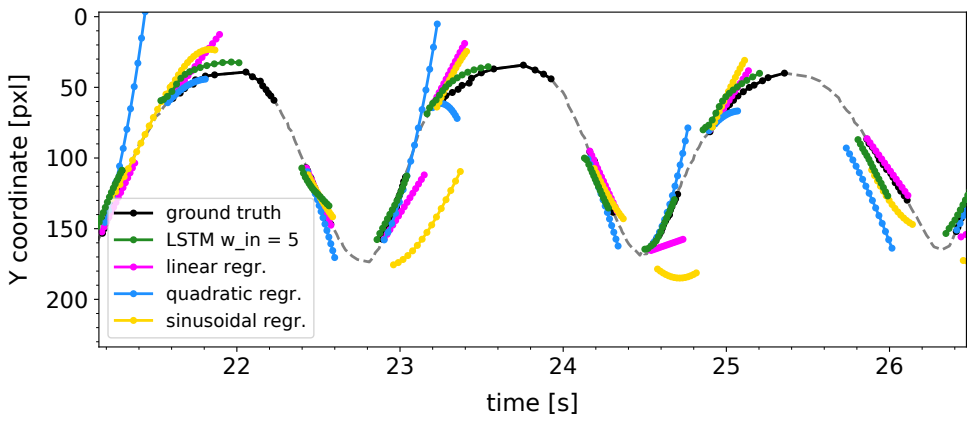


Fig. 6: Example predictions in linear and non-linear segments of the trajectory for all methods.

baselines. The linear model performs well in linear sections of the dataset, but, as expected, cannot correctly fit the "turning-points" present in the data. The quadratic model can better fit the "turning-points" (where the slope changes), but the prediction diverges from the true trajectory in presence of small sources of noise in the linear section. Finally, the sinusoidal model can fit both the linear section and "turning-points", but it requires a long w_{in} to estimate the frequency. It is not sensitive to small, local, variations - for example the starting point of the prediction is often not at the current position of the target because the best fit model to the long history does not intersect the current point. The LSTM is not a perfect predictor, but it is able to predict both linear and non-linear sections of the trajectory and might scale to more complex trajectories. Fig. 7a summarises the results, showing the prediction error at different points of the w_{out} sequence, averaged over the whole dataset. The linear regression presents a low error for the first 10 points but degrades quickly, as it is not complex enough to represent the data. The quadratic regression achieves better results, but it is still unable to match the LSTM performance. The sinusoidal model is the second best fit for the last points, proving that the Y coordinate exhibits a periodic pattern. The large initial error is due to the fact that we are forcing the prediction to have a periodic behavior. The model therefore performs worse at local predictions, but better on the overall trajectory. Fig. 7b shows the analysis performed above, but only for the linear parts of the trajectories, where the linear regression and the LSTM perform best. As the sinusoidal model does not always pass through the current point, it is often offset by a small amount, despite estimating a correct trajectory shape. Fig. 7c shows the analysis for the non-linear segments of the trajectories, confirming the observation that all the predictors, except for the linear one, can achieve good results, with the LSTM showing better accuracy for long-term predictions overall.

C. Prediction of direction change

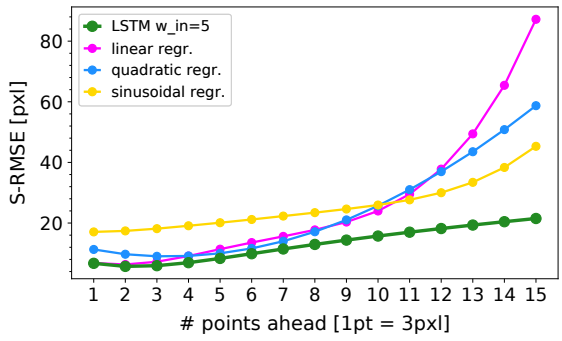
When attempting to predict the "turning-point" of a trajectory, it is not directly known when the subject will decide to change direction. We therefore do not expect the LSTM to predict perfectly. While some cues can be used, such as decrease in velocity, the system can only predict the "turning-point" after some observations that it is occurring. What is important therefore is how quickly the model corrects given only a small amount of evidence. Fig. 8 shows the error statistics for some points before and some points after a turn, and indicates that within 2 points after a motion change, the LSTM trajectory error is within 20 pixels, - a typical value for the model in general.

D. Computational time

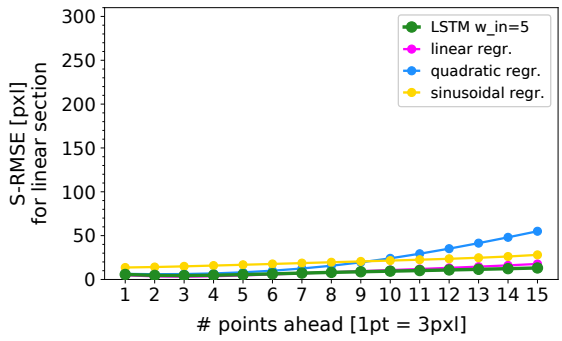
Table I reports the time needed to get a prediction from the LSTM model for different pairs of input and output lengths. The inference time of the LSTM increases with the parameters of w_{in} and w_{out} and may affect design decisions depending on the dynamics of the task, e.g. how much time is available for prediction, and how fast objects are moving. The values tested above, $w_{in} = 5$ and $w_{out} = 15$, result in an update frequency of 500 Hz - when considering pre-processing and detection time, this drops to 250 Hz. Using an event-camera enables a visual input at this rate, also enabling a trajectory prediction at 250 Hz. If using a frame-based camera, the algorithm would be limited by the sensor rate (30 Hz) or tracking algorithm rate and not by the prediction algorithm.

		w_{out}			
		8	**15**	**35**	**60**
w_{in}	**2**	1.7	2.0	3.5	5.0
	5	1.8	2.0	3.5	5.0
	15	2.3	2.5	3.8	5.7
	25	2.5	3.2	4.3	6.3

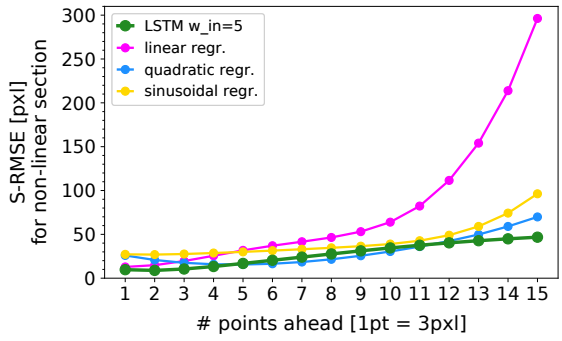
TABLE I: Computational time for different w_{in} and w_{out}, measured in milliseconds [ms]. The average pre-processing is 2 ms which should be added to these values.



(a)



(b)



(c)

Fig. 7: Prediction error for all methods: (a) for the entire test sequence, and split into (b) **linear-only**, and (c) **non-linear-only** segments.

IV. Conclusions

In this work we compare the LSTM learning method to parameterised models for regression, for asynchronous prediction of event-based trajectories. The only model that could achieve similar results as the LSTM is the mixed-sinusoidal model. It could be tuned to better fit the data by optimising w_{in} and introducing a method to more closely fit more recent data compared to past data. However, the method in general is highly specific to the data and therefore does not generalise well, even to similar hand-over tasks, to motions that do not follow the periodic trend. In addition, the linear models for time and x-axis were selected post-hoc, after visualising

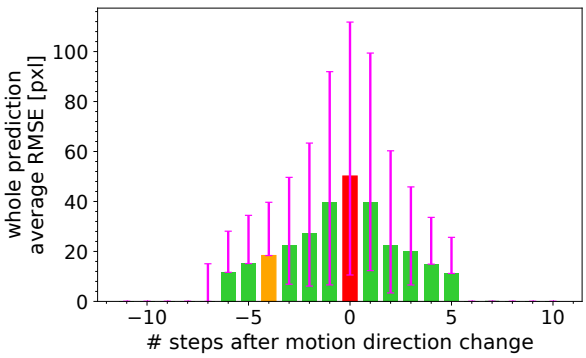


Fig. 8: Error around a direction change point showing 25-50-75 error percentiles.

the data. While the LSTM still requires data to learn, it can adapt to a much wider variety of trajectory types and scales much better to more complex problems. This work was not only a comparison of LSTM to regression techniques, but an exploratory work for predicting in asynchronous frameworks. The event-camera and subsequent event-driven tracking techniques produce data which is spatially sampled, rather than temporally sampled, and therefore presents substantially different information in the sequence itself. We have previously shown that the asynchronous sampling is beneficial for prediction tasks using the LSTM [4]. In this manuscript, we extend the scope of the previous work by showing that not only the LSTM is a good choice to learn to predict object trajectories for handover tasks, but also that it can run as fast as 250 Hz, thanks to the event-driven technology. In particular, the LSTM manages to predict linear and non-linear segments of the data with equal success, requiring many less points in history to make its prediction. While some parts of the data are *unpredictable*, the LSTM is able to quickly correct itself given new evidence of the true trajectory. However, other learning methods exist to which LSTM could be compared. We have avoided a CNN, which is the standard state-of-the-art deep learning technique, as the data does not lend itself to convolutions, but rather is a sequence, and thus more suited to the LSTM model. Our focus was to take the encoder-decoder LSTM as the state-of-the-art in learning methods for data sequences for initial investigation, and integrate improvements once the method was validated as a reasonable choice. These works together present the first steps toward an event-driven framework for human-robot interaction on the neuromorphic iCub. Event-driven and neuromorphic sensing and processing can lead to low-latency, low-computation processing for robots, but the problem of finding the best tools and techniques that fit the shift in sensing paradigm is still open. Given the power of deep learning techniques, we want to understand how these tools can be used with the asynchronous event-driven data, and the asynchronous LSTM appears to have a strong potential to integrate deep-learning with event-cameras in an elegant way.

REFERENCES

[1] S. K. Card, G. G. Robertson, and J. D. Mackinlay, "The information visualizer, an information workspace," in *Proceedings of the SIGCHI Conference on Human factors in computing systems*. ACM, 1991, pp. 181–186.

[2] R. B. Miller, "Response ime in man-computer conversational transactions," in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*. ACM, 1968, pp. 267–277.

[3] C. Posh, 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," in *IEEE Journal of Solid-State Circuits*, vol. 46, Jan. 2011, pp. 259–275.

[4] 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.

[5] C. Bartolozzi, F. Rea, C. Clercq, M. Hofstatter, D. B. Fasnacht, G. Indiveri, and G. Metta, "Embedded neuromorphic vision for humanoid robots," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshop (CVPRW2011)*. IEEE, June 2011, pp. 129–135.

[6] A. Glover and C. Bartolozzi, "Robust visual tracking with a freely-moving event camera," in *2017 IEEE/RSJ International Conference on Intelligent Robots ans Systems (IROS)*. Vancouver, Canada: IEEE, Dec. 2017, pp. 3769–3776.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[8] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 3104–3112.

[9] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, K. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[10] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, "Sequence to sequence - video to text," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015.

[11] S. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. Changshu, China: IEEE, Oct. 2018.

[12] A. Nguyen, T.-T. Do, D. G. Caldwell, and N. G. Tsagarakis, "Real-time 6DOF pose relocalization for event cameras with stacked spatial LSTM networks," *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[13] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza, "Continuous-time visual-inertial odometry for event cameras," *IEEE Transactions on Robotics*, vol. 34, pp. 1425–1440, Dec. 2018.

[14] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. D. Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, and D. Modha, "A low power, fully event-based gesture recognition system," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] T. Delbruck and M. Lang, "Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor," *Frontiers in Neuroscience*, vol. 7, p. 223, Nov. 2013. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2013.00223

[16] A. I. Maqueda, A. Loquercio, G. Gallego, N. Garca, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[17] D. Falanga, S. Kim, and D. Scaramuzza, "How fast is too fast? the role of perception latency in high-speed sense and avoid," *IEEE Robotics and Automation Letters*, vol. 4, pp. 1884–1891, Apr. 2019.

[18] A. Glover, V. Vasco, M. Iacono, and C. Bartolozzi, "The event-driven software library for YARP - with algorithms and iCub applications," *Frontiers in Robotics and AI*, vol. 4, p. 73, 2018.