



# Contextual movement models based on normalizing flows

Samuel G. Fadel<sup>1</sup> · Sebastian Mair<sup>2</sup> · Ricardo da Silva Torres<sup>3</sup> · Ulf Brefeld<sup>2</sup>

Received: 3 November 2020 / Accepted: 23 June 2021  
© The Author(s) 2021

## Abstract

Movement models predict positions of players (or objects in general) over time and are thus key to analyzing spatiotemporal data as it is often used in sports analytics. Existing movement models are either designed from physical principles or are entirely data-driven. However, the former suffers from oversimplifications to achieve feasible and interpretable models, while the latter relies on computationally costly, from a current point of view, nonparametric density estimations and require maintaining multiple estimators, each responsible for different types of movements (e.g., such as different velocities). In this paper, we propose a unified contextual probabilistic movement model based on normalizing flows. Our approach learns the desired densities by directly optimizing the likelihood and maintains only a single contextual model that can be conditioned on auxiliary variables. Training is simultaneously performed on all observed types of movements, resulting in an effective and efficient movement model. We empirically evaluate our approach on spatiotemporal data from professional soccer. Our findings show that our approach outperforms the state of the art while being orders of magnitude more efficient with respect to computation time and memory requirements.

**Keywords** Density estimation · Movement models · Normalizing flows · Soccer data · Spatiotemporal data · Sports analytics

---

Samuel G. Fadel and Sebastian Mair have contributed equally to this paper.

---

✉ Sebastian Mair  
mair@leuphana.de

Samuel G. Fadel  
samuel.fadel@ic.unicamp.br

Ricardo da Silva Torres  
ricardo.torres@ntnu.no

Ulf Brefeld  
brefeld@leuphana.de

<sup>1</sup> University of Campinas, Campinas, Brazil

<sup>2</sup> Leuphana University of Lüneburg, Lüneburg, Germany

<sup>3</sup> Norwegian University of Science and Technology, Ålesund, Norway

## 1 Introduction

Movement models are key to spatiotemporal problems. They allow to study player coordination in team sports (Dick and Brefeld 2019) but also generalize to refugee migration patterns (Hübl et al. 2017), collective animal movements (McDermott et al. 2017), and understanding dynamical systems with moving particles (Padberg-Gehle and Schneide 2017). The task of a movement model is to predict all possible movements of a player (or refugee, animal, particle, etc.) in a given situation within a certain amount of time.

Traditional movement (or motion) models focus on simplifications of physical laws (Taki et al. 1996; Taki and Hasegawa 2000; Fujimura and Sugihara 2005) such as the ability to accelerate in every direction equally fast to compute the set of all reachable positions of an agent for a given time horizon. Besides, those models are non-probabilistic and do not account for the fact that positions may be attained with different likelihoods. A simple parametric probabilistic approach, on the other hand, may lead to suboptimal predictive accuracies due to inappropriate choices of the underlying distributions.

Seemingly, Brefeld et al. (2019) solved the problem by proposing nonparametric probabilistic movement models to quantify the desired likelihoods using kernel density estimation (KDE). Although their solution is purely data-driven and renders assumptions on physics obsolete, they need to distinguish initial conditions (e.g., bins of velocities and time horizon intervals). They address this by maintaining many, possibly differently parameterized, models. This turns the advantage of kernel density estimation into a drawback: being nonparametric by design, predictive performance does increase proportionally with data, but every new data point also increases the computation time for the prediction. The same holds true for memory requirements. This is rather impractical.

In this paper, we turn conditional normalizing flows into novel movement models that (i) consist of only a single contextualized probabilistic model, (ii) better adapt to different contexts, such as movement speed, and (iii) allow predictions whose computation time is independent of the amount of training data, allowing for real-time applications. Normalizing flows (Tabak and Vanden-Eijnden 2010; Tabak and Turner 2013; Rippel and Adams 2013; Dinh et al. 2015; Rezende and Mohamed 2015) provide a state-of-the-art framework for learning densities using invertible deep neural networks. A normalizing flow transforms complex data distributions into simpler ones by an invertible chain of transformations. This chain consists of parametrized bijective functions that transform the data into a representation that follows a known base distribution, usually a Gaussian.

The class of conditional normalizing flows (Winkler et al. 2019; Lu and Huang 2020) additionally offers to model a conditional distribution. We extend their approach in the remainder to incorporate context into flow-based movement models. Hence, our flow-based movement model is actually only *a single model* which can be conditioned on several kinds of contexts, particularly more complex ones than just bins of velocities and time horizon intervals. Moreover, our

contribution is efficient and allows for (near) real-time predictions independently of the amount of data.

The remainder is organized as follows. Section 2 reviews related work and Section 3 introduces preliminaries. Section 4 presents our contextual movement models based on normalizing flows. Section 5 reports on empirical results, and Section 6 provides a discussion of the findings. Section 7 concludes.

## 2 Related work

Spatiotemporal analyses are often fundamental when processing data from wearables like smartphones or dedicated GPS-based tracking devices (Zheng 2015; Mazimpaka and Timpf 2016). While there are straightforward descriptive spatiotemporal tasks like the identification of road defects (Byrne et al. 2013; Mohan et al. 2008) or the discrimination of driving styles (Paefgen et al. 2011), many problems ground on accurate predictions of whereabouts of agents in the near future.

A great deal of these approaches has been published in the context of sports analytics and athlete motion. Often, the focus lies on identifying motion patterns over time for groups of agents (e.g., teams) (Laube et al. 2005; Sprado and Gottfried 2009; Gottfried 2008, 2011). Unfortunately, most of these contributions impose qualitative measures and lack generality. Computational approaches have been proposed by Knauf et al. (2016) who study spatiotemporal convolution kernels or Janetzko et al. (2014) who group attacking patterns on the example of soccer. A frequent pattern mining approach for trajectory data has been proposed in Haase and Brefeld (2014). Recently, neural networks have been applied to athlete trajectories to remedy the need for sufficient statistics and (possibly hand-crafted) feature representations describing the situation on the track or pitch (Zheng et al. 2016; Le et al. 2017).

One of the first movement models has been proposed by Taki and Hasegawa (2000). Their approach, however, simplifies physical laws and allows for unbounded velocities due to constant acceleration. Fujimura and Sugihara (2005) counterbalance this limitation by adding a resistive force to prevent unbounded velocities. Nevertheless, this is insufficient to achieve a realistic model physical model. Recently, Brefeld et al. (2019) circumvent the difficulties in deriving a realistic model from physics by proposing a purely data-driven approach, although their probabilistic model suffers from expensive kernel density estimations in practice.

Naturally, sports analytics is not the only domain where movement models are prominently deployed. Besides the already mentioned areas, other applications include vehicle trajectory analysis (Besse et al. 2018), pedestrian trajectory estimation from videos (Zhong et al. 2020), and real-time robot trajectory prediction (Gomez-Gonzalez et al. 2020).

Normalizing flows recently emerged as an attractive approach to learning densities. A significant advantage over other methods for learning densities using neural networks is that they allow the direct maximization of the log-likelihood. Alternatives such as generative adversarial networks (GANs) (Goodfellow et al. 2014) and variational autoencoders (VAEs) (Kingma and Welling 2014; Rezende et al. 2014)

use instead surrogate learning objectives for this task, such as the GAN adversarial loss and the VAE evidence lower bound (ELBO). While this does not hamper their use as generative models, they are not suitable for estimating the likelihood of a given data point.

In the last few years, a number of different flow-based models were proposed. A particular family of models, derived from NICE (Dinh et al. 2015), is appealing due to their computationally efficient nature while being simple to implement. Improvements such as RealNVP (Dinh et al. 2017) and Glow (Kingma and Dhariwal 2018) were proposed over the years, achieving state-of-the-art performance to model complex, high-dimensional distributions, while keeping the attractive properties of NICE. Similarly, autoregressive flow models such as NAF (Huang et al. 2018) and B-NAF (De Cao et al. 2019) have been proposed. Particularly interesting are conditional normalizing flows (Lu and Huang 2020), a simple improvement over Glow which allows them to model distributions conditioned on continuous variables. For an in-depth analysis of these and their relation to other flow-based models, we refer the reader to Papamakarios et al. (2021).

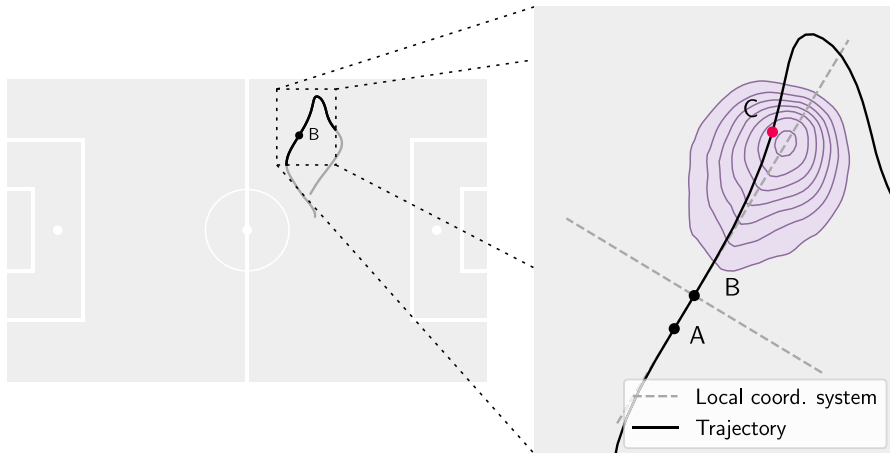
### 3 Preliminaries

Let  $\mathcal{T} = (\mathbf{x}^{(t)})_{t \in \mathbb{R}^+}$  be the trajectory of positions  $\mathbf{x}^{(t)} \in \mathbb{R}^d$  at time  $t$  of an object. In the remainder, we deal with two-dimensional movements, i.e.,  $\mathbf{x}^{(t)} = (x_1^{(t)}, x_2^{(t)})^\top$ , but the following definitions also hold for higher-dimensional movements. The positional data might come with additional information, the so-called context  $\mathbf{c}^{(t)}$  at time  $t$ . An example for context is the velocity vector  $\mathbf{v}^{(t)} \in \mathbb{R}^d$ , or its magnitude, which is known as speed  $v^{(t)} = \|\mathbf{v}^{(t)}\|_2$ , respectively. Our goal is to model the distribution of the position  $\mathbf{x}^{(t+t_\Delta)}$ , which is  $t_\Delta > 0$  seconds in the future, given the current position  $\mathbf{x}^{(t)}$  and context  $\mathbf{c}^{(t)}$ .

To make the movement model location-invariant, we consider a local coordinate system centered at the object's current position and along the last movement direction. Figure 1 shows an example. The object under consideration is currently at position  $B$  and moves along the solid trajectory. The dashed lines show the local coordinate system which is aligned using the last direction, estimated by a position  $A$ , which is in the past. In  $t_\Delta = 1$  seconds, the position  $C$  is reached. The distribution of the next position, which is what we want to model, is depicted as a red contour plot.

#### 3.1 Movement models based on kernel density estimates

Our movement model is based on concepts of Brefeld et al. (2019). They consider movement models for trajectories in soccer and employ a kernel density estimate for their model. The idea is as follows. Given a trajectory (or a set of many trajectories) one first extracts triplets as shown in Figure 1. Let  $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C)$  be such a triplet of positions with timestamps  $t_A, t_B$ , and  $t_C$  such that  $t_A < t_B < t_C$ . We denote the time difference of  $t_A$  and  $t_B$  as  $t_\delta = t_B - t_A$  and the difference of  $t_B$  and  $t_C$  as  $t_\Delta = t_C - t_B$ . Here,  $\mathbf{x}_B$  describes the current position,  $\mathbf{x}_A$  is used to estimate the current direction in



**Fig. 1** Illustration of a movement model (purple contour lines). The solid black line depicts a trajectory. The last, current, and next position of a player are marked as A, B, and C, respectively. The movement model shown in purple models the distribution of the next position C (red dot) using the information of the past and present (A and B). The dashed lines denote a local coordinate system centered at the player and aligned in the current direction of movement

which the object is moving and  $\mathbf{x}_C$  denotes the position in the future. Hence, the next position  $\mathbf{x}_C$  describes the ability to move within a given time horizon  $t_\Delta$ . Collecting multiple next positions  $\mathbf{x}_C$ , represented in the local coordinate system as  $\mathbf{x}_C^{\text{local}}$ , allows for creating a movement model. Such a model is then able to quantify the likelihood of a possible next position relative to its current position.

To obtain the next position  $\mathbf{x}_C^{\text{local}}$ , the triplet is transformed into the local coordinate system. The transformation realizing this first subtracts the current position  $\mathbf{x}_B$  from all points of the triplet. This way, the triplet is centered at the current position. Then, the triplet is rotated such that the last position, i.e.,  $\mathbf{x}_A$ , is aligned with the x-axis of the local coordinate system. Let  $\mathbf{g}(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C) = \mathbf{x}_C^{\text{local}}$  be this transformation. Mathematically, this transformation can be expressed as

$$\mathbf{x}_C^{\text{local}} = \begin{bmatrix} r \cdot \cos(\theta) \\ r \cdot \sin(\theta) \end{bmatrix},$$

where  $r$  is a distance given by  $r = \|\overline{\mathbf{x}_B \mathbf{x}_C}\|_2$  and  $\theta$  is a signed angle given by  $\theta = \angle(\overline{\mathbf{x}_A \mathbf{x}_B}, \overline{\mathbf{x}_B \mathbf{x}_C})$ . After processing multiple triplets along a trajectory for fixed time differences  $t_\delta$  and  $t_\Delta$  and storing the corresponding next positions in the local coordinate system in a set  $\mathcal{S}_{t_\Delta}$ , a kernel density estimate is employed on  $\mathcal{S}_{t_\Delta}$ .

Brefeld et al. (2019) construct several movement models for various time horizons  $t_\Delta$ . As a context, they use the current speed of a player. To add this contextual information, they filter the triplets by current speed. This is done by first defining a binning of the range of possible speeds. Specifically, they use the following binning: [0, 1) (standing), [1, 7) (walking), [7, 14) (jogging), [14, 20) (running), and [20, 40) (sprinting), where all values are in kilometers per hour. Then, movement models are

constructed for multiple time horizons per speed range  $S$ , yielding  $\mathbb{S}_{t_\Delta}^S$ . An example is depicted in Fig. 2. The left shows the set  $\mathbb{S}_{t_\Delta}^S$  holding the next positions within the local coordinate system for a time horizon of  $t_\Delta = 1s$ . We apply the context of a speed range  $S = [14, 20)$ , where the values are again in kilometers per hour. The corresponding kernel density estimate is shown on the right. Following Brefeld et al. (2019), we deploy a Gaussian kernel and select the bandwidth using Scott's rule (Scott 2015).

The time complexity of this approach is its biggest drawback. Computing the likelihood of a single  $d$ -dimensional data point with a kernel density estimate built on  $n$  triplets  $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C)$  takes  $\mathcal{O}(dn)$  time. Analogously, the approach has also a large memory footprint, as every known data point must be accessible during prediction. The space complexity also scales in  $\mathcal{O}(dn)$ .

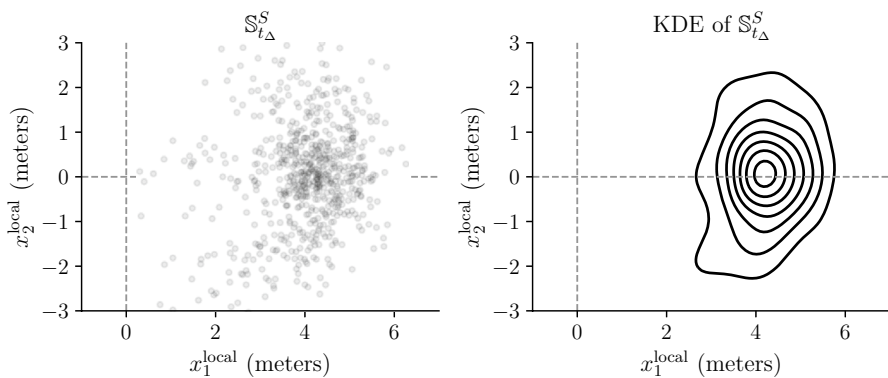
### 3.2 Normalizing flows

Let  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^d$  be instances drawn from an unknown distribution  $p_x(\mathbf{x})$ . The goal is to estimate an accurate model of  $p_x(\mathbf{x})$ . This is done by expressing  $p_x(\mathbf{x})$  in terms of a simpler, known distribution  $p_z(\mathbf{z})$ , and learning a bijective map between them. Formally, let  $\mathbf{f} : \mathbb{R}^d \rightarrow \mathbb{R}^d$  be a bijective and differentiable function. Using the change of variable theorem and  $\mathbf{z} = \mathbf{f}(\mathbf{x})$ , we can express  $p_x(\mathbf{x})$  as

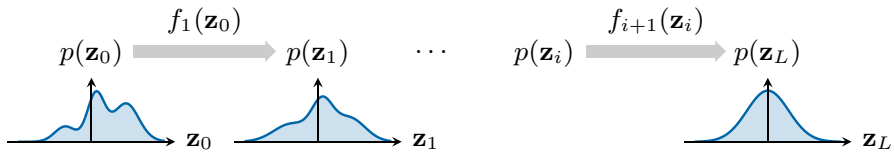
$$p_x(\mathbf{x}) = p_z(\mathbf{f}(\mathbf{x})) |\det J_{\mathbf{f}}(\mathbf{x})|,$$

where  $J_{\mathbf{f}}(\mathbf{x})$  is the Jacobian matrix of the diffeomorphism  $\mathbf{f}$ . The distribution  $p_z(\mathbf{z})$  is often referred to as a *base distribution* (Papamakarios et al. 2021). We henceforth drop the subscript of the distribution  $p$  whenever it is clear from the context.

Figure 3 depicts a one-dimensional example. The left-hand side shows the data distribution, which is usually unknown. The goal is to map from this distribution to a known, base distribution, depicted on the right side. The usual choice is a standard Gaussian.



**Fig. 2** An example of the set  $\mathbb{S}_{t_\Delta}^S$  with a time horizon of  $t_\Delta = 1s$  and a speed range of 14-20 km/h and the corresponding kernel density estimate



**Fig. 3** A simple normalizing flow on one-dimensional data. The plot to the left depicts the usually unknown data distribution, while the plot to the right shows the base distribution (standard Gaussian)

In practice,  $\mathbf{f}$  is not just a fixed transformation, but rather a chain of parameterized transformations. Hence, we deal with  $\mathbf{f}^{(\theta)}$ , which is represented by a chain of bijective transformations  $\mathbf{f}^{(\theta)} = \mathbf{f}_L^{(\theta)} \circ \mathbf{f}_{L-1}^{(\theta)} \circ \dots \circ \mathbf{f}_1^{(\theta)}$ , where  $L$  is the total number of transformations. This composition defines a *normalizing flow* between  $\mathbf{x} = \mathbf{z}_0$  and  $\mathbf{z} = \mathbf{z}_L$  and is precisely what enables flows to be computationally and analytically tractable. Let  $\mathbf{z}_i = f_i(\mathbf{z}_{i-1})$  be the intermediate variable, where  $i = 1, \dots, L$ . Then,  $\log p(\mathbf{x})$  can be expressed as the log-likelihood of the base distribution and the log-determinant of the Jacobians of each bijection

$$\log p(\mathbf{x}) = \log p(\mathbf{z}) + \sum_{i=1}^L \log \left| \det J_{\mathbf{f}_i^{(\theta)}}(\mathbf{z}_{i-1}) \right|.$$

Thus, we can learn the transformation from the data  $\mathcal{X}$  to its base representation by optimizing the parameters  $\theta$  that minimize the negative log-likelihood

$$\min_{\theta} - \sum_{\mathbf{x} \in \mathcal{X}} \log p(\mathbf{x}).$$

In the remainder, we further simplify the notation by dropping the superscript  $\theta$  from  $\mathbf{f}$ . Note that the example in Fig. 3 described above shows that a normalizing flow defines a generative model. After learning the transformation  $\mathbf{f}$  on data  $\mathcal{X}$ , we can sample from the base distribution and obtain new samples, distributed according to  $p_x$ , by using the inverse transformation  $\mathbf{f}^{-1}$ .

## 4 Learning movements with flow-based models

The choice of transformations for building the flow  $\mathbf{f}$  involves trade-offs between computational efficiency during learning or data generation and its ability to model  $p_x$ . In the last few years, several different approaches emerged. To turn a normalizing flow into a movement model, we focus on only a few characteristic properties. As movement data tends to be low-dimensional, it can often be sampled at a high frequency, resulting in millions of observations.

### 4.1 Flow-based models without context

As an architecture for our flow model, we consider Glow (Kingma and Dhariwal 2018), which is itself based on RealNVP (Dinh et al. 2017). Glow is built from

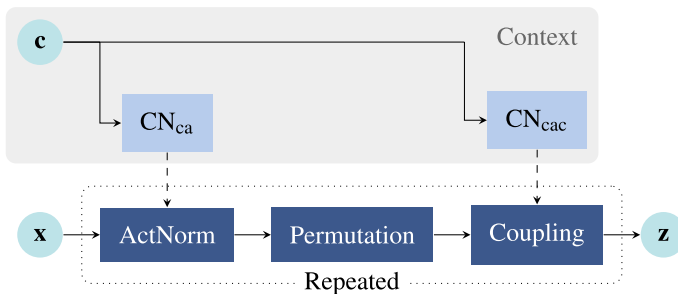
three main transformations: activation normalization (actnorm),  $1 \times 1$  invertible convolutions, and affine coupling. Those transformations are employed in a multi-scale architecture, reshaping the image tensors to have fewer pixels with more channels, referred to as *squeezing*. The channels are then split and further operations are only performed on half of them. This squeezing and splitting scheme is performed several times for scalability. Squeezing, however, is not suitable for vectorial data including  $d$ -dimensional positions. Furthermore, the dimensionality of our setting curtails the computational performance gains from splitting, allowing us to use the entire vector in every transformation of the flow.

An overview of the non-contextualized architecture we use is depicted in Fig. 4 (non-grayed area). We detail the actnorm and affine coupling transformations next. For both of them, let  $\mathbf{z}, \mathbf{y} \in \mathbb{R}^d$  be  $d$ -dimensional vectors, where  $\mathbf{z}$  is the input of the current transformation and  $\mathbf{y}$  is its output, being directly fed into the next transformation in the chain.

*Actnorm.* Let  $\mathbf{a} \in \mathbb{R}^d$  be a scaling vector and  $\mathbf{b} \in \mathbb{R}^d$  an offset vector. The actnorm transformation Kingma and Dhariwal (2018), its inverse, and log-determinant of its Jacobian are given by

$$\begin{aligned}\mathbf{y} &= \mathbf{f}(\mathbf{z}) = \mathbf{a} \odot \mathbf{z} + \mathbf{b}, \\ \mathbf{z} &= \mathbf{f}^{-1}(\mathbf{y}) = (\mathbf{y} - \mathbf{b})/\mathbf{a}, \\ \log |\det J_{\mathbf{f}}(\mathbf{z})| &= \sum_{i=1}^d \log |a_i|,\end{aligned}$$

respectively, where  $\odot$  denotes the Hadamard product. Particularly,  $\mathbf{a}$  and  $\mathbf{b}$  are learned as part of the transformation and are initialized with the first batch of data. This initialization is such that the mean and standard deviation of  $\mathbf{y}$  are zero and one, respectively.



**Fig. 4** Overview of the flow model: the composition of actnorm, permutation and coupling is repeated multiple times to build the flow. The contextual part is shaded in gray.  $\mathbf{x}$  is the input to the flow, in our setting the local coordinates ( $\mathbf{x}_C^{\text{local}}$ );  $\mathbf{z}$  is the output of the transformation into the base representation, which follows a predefined (base) distribution; in the conditional variant,  $\mathbf{c}$  is the context vector, which can be used to characterize the movement being modeled, such as speed.  $\text{CN}_{ca}$  (conditional actnorm) and  $\text{CN}_{cac}$  (conditional affine coupling) augment the respective transformations with contextual information



*Affine coupling.* This transformation (Dinh et al. 2017) is slightly more involved. The input  $\mathbf{z} \in \mathbb{R}^d$  is initially split into two parts  $(\mathbf{z}_1, \mathbf{z}_2)$ , each of which is  $d'$ -dimensional, where  $d' = d/2$ . Then, the second part is simply copied,  $\mathbf{y}_2 = \mathbf{z}_2$ , while  $\mathbf{z}_1$  is transformed based on information from  $\mathbf{z}_2$  as follows:

$$\begin{aligned}(\log \mathbf{a}, \mathbf{b}) &= \text{NN}_{\text{ac}}(\mathbf{z}_2), \\ \mathbf{y}_1 &= \exp(\log \mathbf{a}) \odot \mathbf{z}_1 + \mathbf{b},\end{aligned}\tag{1}$$

where  $\text{NN}_{\text{ac}} : \mathbb{R}^{d'} \rightarrow \mathbb{R}^d$  is a nonlinear mapping, generally realized by a neural network. The output is then  $\mathbf{y} = \text{concat}(\mathbf{y}_1, \mathbf{y}_2)$ . This inverse transformation can be computed by first splitting  $\mathbf{y}$  into  $(\mathbf{y}_1, \mathbf{y}_2)$ , copying the second part as before, i.e.,  $\mathbf{z}_2 = \mathbf{y}_2$ , computing the log-scale and offset parameters using  $\text{NN}_{\text{ac}}(\cdot)$  as in Eq. (1), and computing the inverse affine linear transformation via

$$\mathbf{z}_1 = (\mathbf{y}_1 - \mathbf{b}) / \exp(\log \mathbf{a}).$$

Finally, the two parts are merged with  $\mathbf{z} = \text{concat}(\mathbf{z}_1, \mathbf{z}_2)$ . The log-determinant of the Jacobian of this transformation is

$$\log |\det J_f(\mathbf{z})| = \sum_{i=1}^{d'} \log |a_i|.$$

*Permutation.* Not using squeezing and splitting entails a few additional considerations that directly lead us to  $1 \times 1$  convolutions that are particularly designed for these cases. Squeezing increases the number of channels and affect the way affine coupling layers perform. Hence, a permutation operation over the channels is usually performed before an affine coupling transformation. This allows the squeezing to affect a broader range of channels than it is designed to, lifting the flow to perform more sophisticated transformations. As such, having a permutation-like operation over the channels plays a significant role. Recall that this is not the case in our scenario as we cannot apply squeezing at all. Instead, we propose to employ a permutation operation that simply reverses the dimensions of the input. This operation has an inverse transformation and its log-determinant is equal to zero.

A major benefit of this approach is the ability to compute predictions independently of the data used for training. It only requires a number of computations proportional to the data dimensionality  $d$  and the number of transformations  $L$  employed. In other words, computing the likelihood of a data point takes  $\mathcal{O}(dL)$  time. Additionally, only the model parameters have to be stored in memory, resulting in a significantly smaller memory footprint than the KDE-based movement model when dealing with large data sets.

## 4.2 Contextualized flow-based movement models

In this setting, we are particularly interested in contextual information associated with movement data, such as speed. Therefore, the transformations employed in the flows should be able to take into account this additional information. A

conditional flow-based model that appropriately addresses those requirements is conditional Glow (c-Glow) (Lu and Huang 2020). As the name suggests, c-Glow is based on Glow (Kingma and Dhariwal 2018), which is itself based on Real-NVP (Dinh et al. 2017). Nevertheless, the goal is to derive a contextualized model that can be conditioned on arbitrary contexts. The most significant change is the addition of a *conditioning network*, denoted by  $\text{CN}(\cdot)$ , which replaces the parameters of the actnorm and affine coupling layers with surrogates that are predicted from the context  $\mathbf{c} \in \mathbb{R}^{d_c}$ . Hence, a conditional probability density can be devised whose log-variant is given by

$$\log p(\mathbf{x}|\mathbf{c}) = \log p(\mathbf{z}) + \sum_{i=1}^L \log \left| \det J_{\tilde{\mathbf{f}}_i^{\theta(\mathbf{c})}}(\mathbf{x}) \right|.$$

The parameters  $\theta(\mathbf{c})$  of the transformations  $\mathbf{f}_i$  are now dependent on the context  $\mathbf{c}$ . Note that this does not change the computation of the log-determinant of the Jacobian in both cases. The new transformations with the addition of the CN are detailed below.

*Conditional actnorm.* The scaling and offset vectors  $\mathbf{a}$  and  $\mathbf{b}$  are now computed by  $\text{CN}_{\text{ca}} : \mathbb{R}^{d_c} \rightarrow \mathbb{R}^{2d}$  as

$$(\mathbf{a}, \mathbf{b}) = \text{CN}_{\text{ca}}(\mathbf{c}),$$

and the transformation is then carried out as before. The original initialization procedure for  $\mathbf{a}, \mathbf{b}$  is no longer needed.

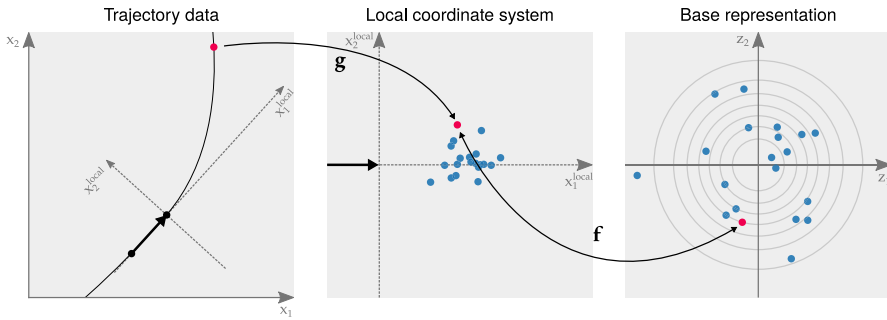
*Conditional affine coupling.* This transformation already computes its scaling  $\mathbf{a}$  and offset  $\mathbf{b}$  parameters from  $\text{NN}_{\text{cac}} : \mathbb{R}^{d+d} \rightarrow \mathbb{R}^d$ . The change introduced by conditioning on  $\mathbf{c}$  is an additional input to  $\text{NN}_{\text{cac}}(\cdot)$ , computed by  $\text{CN}_{\text{cac}} : \mathbb{R}^{d_c} \rightarrow \mathbb{R}^d$ , as

$$(\log \mathbf{a}, \mathbf{b}) = \text{NN}_{\text{cac}}(\mathbf{z}_2, \text{CN}_{\text{cac}}(\mathbf{c})),$$

followed by the same operations as before.

Figure 4 provides an overview of the complete normalizing flow model with and without the conditioning on  $\mathbf{c}$ . As shown in Fig. 5, learning contextual flow-based probabilistic movement models proceeds as follows: Given positional data, triplets  $(A, B, C)$  are extracted from the trajectories. The future position  $C$  in a time horizon  $t_\Delta$  of interest is then represented in a local coordinate system for spatial invariance. The positions which can be reached in time  $t_\Delta$  are then transformed, using our normalizing flow model, into a representational space that follows a standard Gaussian.

The additional context taken into account by the model, while crucial to its flexibility in dealing with a variety of movements observed in the data, incurs little additional computational cost. As in the non-contextual case, the time to compute the likelihood of a data point is  $\mathcal{O}(dL)$  and the amount of memory required is proportional to the number of parameters of the model.



**Fig. 5** A summary of our approach. Left: Triplets are extracted from a trajectory using some current and previous position (black) as well as a future position (red) reached within the time horizon  $t_{\Delta}$ . Center: The future position is remapped using  $\mathbf{g}$  into a local coordinate system. The coordinates are centered at the current position and aligned with the previous position. Right: Points reached in time  $t_{\Delta}$  are transformed using a conditional normalizing flow  $\mathbf{f}$ . The flow  $\mathbf{f}$  is optimized to ensure that points under this transformation are distributed according to a Gaussian base distribution

## 5 Experiments

We now evaluate our flow-based probabilistic movement models. First, we conduct an in-depth analysis of our proposed approach and compare its performance with several baselines. Second, we investigate whether using the positions of the remaining players as additional contextual information improves our movement model. For all experiments, we use data from professional soccer.

### 5.1 Data

The tracking data from soccer consist of coordinates of each player and the ball, recorded with camera-based systems at 25Hz for five professional games from the German Bundesliga. Each game is encoded as a sequence of triplets  $(x_1^{(t)}, x_2^{(t)}, v^{(t)})$  describing the  $x$  and  $y$  coordinates on the pitch and the current speed  $v^{(t)}$  in km/h at time  $t$  for every player. Thus, every game consists of about  $25 \cdot 60 \cdot 90 = 135,000$  positions per player. The  $x$  and  $y$  positions are relative to the origin. Hence, the coordinates are within  $[-52.5, 52.5] \times [-34.0, 34.0] \subset \mathbb{R}^2$ , since the dimensions of a standard soccer field are  $105 \times 68$  meters. We use the first four games for training and the last game for testing.

### 5.2 Baselines

We compare our model against the following baselines. The first baseline, denoted as **KDE**, follows the idea of Brefeld et al. (2019) and deploys kernel density estimates for the movement models. We use a Gaussian kernel and select the bandwidth using Scott's rule (Scott 2015). A second, straightforward baseline simply estimates a two-dimensional Gaussian (mean and covariance) on the

point cloud  $\mathbb{S}$  as introduced in Sect. 3, see for instance Figure 2. We refer to this baseline as **Gaussian**. Third, we test a two-dimensional histogram, again on the point cloud mentioned above. The histogram uses an equally spaced grid over  $[-10, 30] \times [-20, 20]$  with 1600 cells of size  $1 \times 1$  meter. Note that this range covers a larger space than depicted in Figure 2 and is suitable for all velocity ranges. We refer to this baseline as **Histogram**.

Depending on the experiment we use different configurations. We experiment with time horizons  $t_\Delta \in \{1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}$ , where every value is in seconds. The baselines use the same speed discretization as in Brefeld et al. (2019) which is  $\{[0, 1), [1, 7), [7, 14), [14, 20), [20, 40]\}$ . Those values are in kilometers per hour. Consequently, we train per baseline  $7 \cdot 5 = 35$  different configurations, one per time horizon and speed range. As for the  $t_\delta$ , denoting the time difference for estimating the direction in which a player is moving, we again follow Brefeld et al. (2019) and use  $t_\delta = 0.2$  seconds. We show the influence of  $t_\delta$  in "Appendix 1". The code for all baselines is written in Python using NumPy (Harris et al. 2020) and SciPy (Virtanen et al. 2020). All experiments run on a machine with an Intel Xeon CPU, 256GB of RAM, and an NVIDIA V100 GPU.

### 5.3 Evaluation metrics

When evaluating and comparing movement models, it is important to quantify how well a model explains the observed movements. In other words, a movement model should be capable of showing where the agent or object of interest will be in the near future. Consider a model that predicts a large area of future positions. Although this model explains all future positions by its sheer broadness, it is not concise. Hence, a good movement model needs to be concise and accurate and, to fulfill both, the model has to find an optimal trade-off between area and accuracy to estimate future positions in the smallest area possible.

A natural measure for this trade-off is the (log-)likelihood. Since densities are normalized by definition, larger areas possess lower point-wise likelihoods while compact densities capture only trivial movements and do not generalize well. As a consequence, a model achieving a good trade-off will also achieve higher likelihoods.

We additionally aim to measure the complexity of the different approaches. Section 3 shows that the complexity of predicting a likelihood either scales in the size of training data (KDE) or in the complexity of the neural networks (flow-based models). We thus measure the average evaluation time and report its quantity in seconds. Note that the prediction speed remains constant for both the Histogram and the Gaussian baseline.

A similar argument holds for memory footprints of the different approaches. The predictive performance of KDE is not only expected to deteriorate for data at large scales, it is also expected to require an excessive amount of memory. Hence, we also analyze the memory requirements of all evaluated methods and provide the number of variables (i.e., floats or double precision) that need to be stored.

## 5.4 Setup of the flow-based movement models

For a thorough comparison of the unconditional and the conditional flow-based movement models, we experiment with several configurations of the proposed models. The unconditional **Flow** uses the same time horizons and speed discretizations as the baselines above. Hence, we consider as many different models as the baselines for obtaining a fair comparison. With Flow, we intend to show that movement models based on normalizing flows achieve the same or better predictive performance as the state of the art. We introduce contextual information, given by  $\mathbf{c}^{(t)} = (v^{(t)}, t_\Delta)$ , directly as a part of a single model in **CFlow**. Here, we use the current speed  $v^{(t)}$  and the time horizon  $t_\Delta$  as context, just as for the baselines. Note that the baselines are implicitly conditioned on the time horizon as well due to maintaining different models for different intervals. Hence, the purpose of CFlow is to have a unified model instead of many different ones. We further evaluate a model called **CFlow-extended** which leverages the positions of other players as additional contextual information. This model uses as context  $\mathbf{c}^{(t)} = (v^{(t)}, t_\Delta, \bar{x}_{\text{rel}}^{(t)})$ , where  $\bar{x}_{\text{rel}}^{(t)} \in \mathbb{R}^{\frac{1}{2}d_{\text{hidden}}}$  is a permutation invariant representation of the relative positions of the remaining players. We provide details of how we compute this vector in "Appendix 2".

All models follow the architecture shown in Figure 4, repeating the three depicted transformations (actnorm, permutation, and coupling)  $L = 8$  times in sequence. The same applies to their contextual alternatives. For unconditional and conditional flow-based models, the architectures of the neural networks are outlined in Table 1. All networks use SELU (Klambauer et al. 2017) activations,  $d$  denotes the dimensionality of the input  $\mathbf{x}$  and  $d_c$  is the dimensionality of the context which is for CFlow and CFlow-extended,  $d_c = 2$  and  $d_c = 2 + \frac{1}{2}d_{\text{hidden}}$ , respectively. The size of the hidden dimensionality is set to  $d_{\text{hidden}} = 16$ . Just as the authors of Glow (Kingma and Dhariwal 2018), we initialize the weights and biases of the last layer of each neural network in Table 1 to zeros for stability. This implies the actnorm and affine coupling layers are initialized to identity transformations before training.

The code is written in Python using JAX (Bradbury et al. 2018). We employ the Adam optimizer (Kingma and Ba 2015) with a learning rate of  $10^{-3}$ . For stability,

**Table 1** Neural network architectures detailed as a sequence of fully connected layers, where the constants before and after arrows indicate the dimensionalities of input and outputs for that layer, respectively, while SELU denotes the activation function Klambauer et al. (2017). The networks are part of the affine coupling  $\text{NN}_{\text{ac}}(\mathbf{z}_2)$  that operates on the second part of the intermediate representation  $\mathbf{z}_2 \in \mathbb{R}^{d/2}$ , the conditional actnorm  $\text{CN}_{\text{ca}}(\mathbf{c})$  which uses the context  $\mathbf{c} \in \mathbb{R}^{d_c}$ , and the conditional affine coupling  $\text{NN}_{\text{cac}}(\mathbf{z}_2, \tilde{\mathbf{c}})$ , where  $\tilde{\mathbf{c}} = \text{CN}_{\text{ca}}(\mathbf{c}) \in \mathbb{R}^d$

Network	Architecture and activation functions
$\text{NN}_{\text{ac}}$	$(d/2) \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d$
$\text{CN}_{\text{ca}}$	$d_c \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} 2d$
$\text{NN}_{\text{cac}}$	$(d/2 + d) \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d$
$\text{CN}_{\text{cac}}$	$d_c \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d$

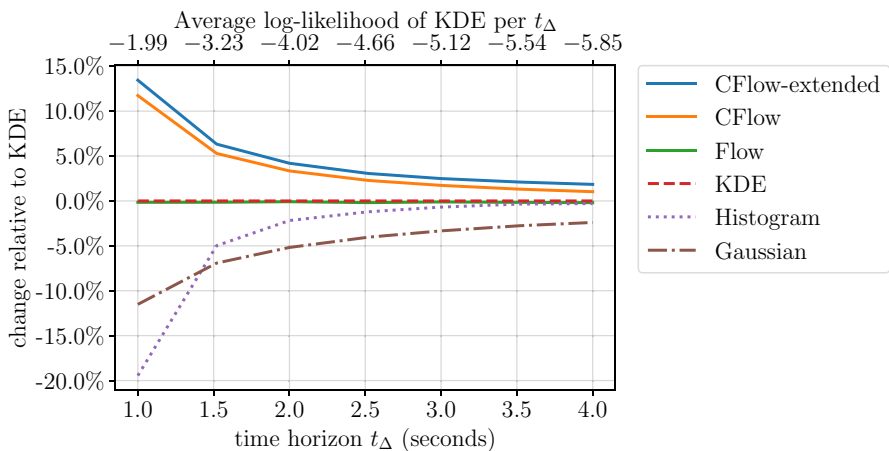
we clip gradients (Pascanu et al. 2013) with norms larger than 20. The batch size is set to 1,024 for all models except the conditional models which condition on time horizon, in which a larger batch size of  $1,024 \cdot N_{t_\Delta}$  is used to accommodate for different time horizons, where  $N_{t_\Delta}$  denotes the number of different time horizons. All models are trained for 100 epochs with no early stopping.

## 5.5 Results

We first evaluate the predictive performance of the baseline movement models and compare them to the unconditional Flow. To have a fair comparison, we deal with 35 different configurations per model. We proceed as follows. From the test game, we randomly sample a trajectory over three minutes for every player, leaving us with 22 such trajectories. Then, for several time horizons ranging from 1s to 4s, we compare the average log-likelihood per trajectory and model as well as the corresponding computation time for the prediction.

Unsurprisingly, the average log-likelihoods decrease for larger time horizons  $t_\Delta$ , as the models become progressively more uncertain with increasing  $t_\Delta$ . Figure 6 shows the relative improvement of each model compared to the KDE. It can be seen that the Flow model (green) performs on par with the KDE (dashed red) while the Gaussian approximation (brown) is almost consistently the worst. In addition, the performance gap of the Histogram (dotted purple) closes for larger time horizons, however, it underperforms compared to the KDE.

The relative improvement of CFlow (orange) and CFlow-extended (blue), each of which are single contextual models, shows that they clearly dominate all baselines by far. We credit this observation to conditioning. While the other models use a predefined velocity discretization, both CFlow and CFlow-extended adapt specifically to any velocity without the need for binning intervals with identical model output.

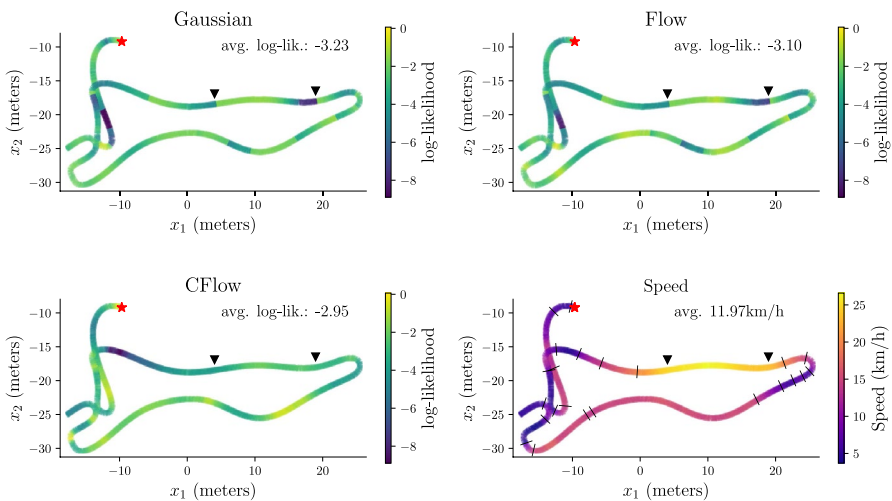


**Fig. 6** Average log-likelihood values (22 players) per model relative to the KDE baseline for various time horizons  $t_\Delta$ . The specific values of the KDE are stated in the upper x-axis

If we additionally include the relative positions of the other players into the context and condition on it, the performance increases further.

The flexibility afforded by conditioning can also be seen in Fig. 7, which visualizes a trajectory of a soccer player drawn from the data. The bottom-right part of the figure depicts the current velocities of the player. In the remaining parts, the trajectory is colored according to the log-likelihood values using the Gaussian, Flow, and CFlow models for a time horizon of  $t_{\Delta} = 1s$ . We also evaluate the average log-likelihood of this trajectory showing that CFlow has the highest log-likelihood, meaning it predicts the movement best. Once again, the Gaussian model yields the worst log-likelihood. We credit this due to its implicit symmetry. While being symmetric for moving left or right might be acceptable in this use case, the symmetry in the orthogonal direction is clearly not present, as seen in Fig. 2. Visual inspection shows the effect of binning the velocities on the performance of the Gaussian and Flow; the coloring exhibits a block structure due to velocity transitions across bins. By contrast, the conditional CFlow distinguishes by a continuous coloring, indicating a much better adaptation to the trajectory at hand. CFlow and CFlow-extended can smoothly adjust to the current velocity at each point in time.

Table 2 compares the predictive performance of the competitors in terms of computation time (in seconds). The numbers are again averaged over 22 individual trajectories and are shown with their standard errors. The baselines Gaussian and Histogram take almost no time to compute the predicted movements. KDE is consistently the slowest model by orders of magnitude. The result shows the limitations



**Fig. 7** A sample trajectory from soccer data. The bottom right plot shows speed along the trajectory. Other plots depict log-likelihood values along the trajectory for the Gaussian, Flow, and CFlow models for a time horizon of  $t_{\Delta} = 1s$ . Darker colors in these other plots indicate the model estimates that position as less likely. We also provide avg. log-likelihood values for each model in this trajectory. The red star denotes the start of the trajectory. Black lines indicate model transitions for the Gaussian and Flow models, which rely on speed binning. Black triangle markers indicate an example where models behave distinctively: at those points in the trajectory the player starts accelerating and then slows down and CFlow maintains a consistent quality of predictions, while the other two models do not

**Table 2** Evaluation time averaged over 22 players (incl. standard errors) in seconds

Model	$t_d = 1.0s$
KDE	774.3s $\pm$ 23.0
Histogram	< 0.1s $\pm$ 0.0
Gaussian	< 0.1s $\pm$ 0.0
Flow	0.5s $\pm$ 0.0
CFlow	0.1s $\pm$ 0.0
CFlow-extended	0.7s $\pm$ 0.0

of the KDE for (near) real-time applications and/or large data sets. Our proposed flow-based models are only marginally slower than the straightforward competitors and, as expected, enable predictions that can be computed very efficiently.

The KDE baseline is not only the slowest approach when comparing evaluation time, it has also the highest memory demand, as Table 3 shows. This is due to the nonparametric nature of the approach. The more training data is observed and integrated, the better the model. Trivially, integrating more data into a KDE also means to store exactly these additional data. In comparison, the Histogram baseline has a fixed grid and velocity binning, and, hence, a fixed memory footprint. The Gaussian exploits strong assumptions (e.g., unimodality, symmetry) and needs the least amount of memory, but is quite limited in expressiveness. Our proposed flow-based models have moderate memory requirements which neither grow with training data as the KDE, nor with a higher precision or finer-grained bins as the Histogram.

## 6 Discussion

Binning velocities are common trick in histogram-based movement models since continuous movements can be treated as discrete events that are instances of one or another bin. A major issue that is usually not addressed properly is how to find the optimal size of bins for a problem at hand. Often, optimal sizes are not equidistant but grow with the values of the variable of interest, such as speed. The bins also induce hard thresholds that may lead to treating similar values very differently, if

**Table 3** Memory footprint of the evaluated approaches. The memory demand is stated in terms of floating point numbers which need to be saved

Model	No. of models	Memory demand	
		Per model	Total
KDE	35	variable	175,177,098
Histogram	35	1,600	56,000
Gaussian	35	6	210
Flow	35	2,360	82,600
CFlow	1	11,704	11,704
CFlow-extended	1	15,056	15,056



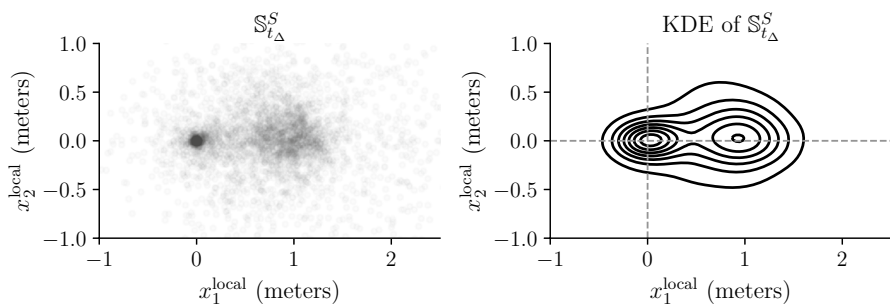
they end up in two neighboring bins. The presented contextual models do not have these limitations as they work directly on the continuous signal.

In the case of movement models, binning leads to maintaining several models and renders the space and time complexity of the models highly inefficient. By contrast, the contextual models do not rely on binning and simply condition on all variables of interest such as current velocity, time horizon, and even the position of the other players. We thus end up with only a single model which is efficient and allows for (near) real-time processing. Our experiments show that both contextual models are moreover achieving superior performance in terms of log-likelihood and clearly outperform all other approaches.

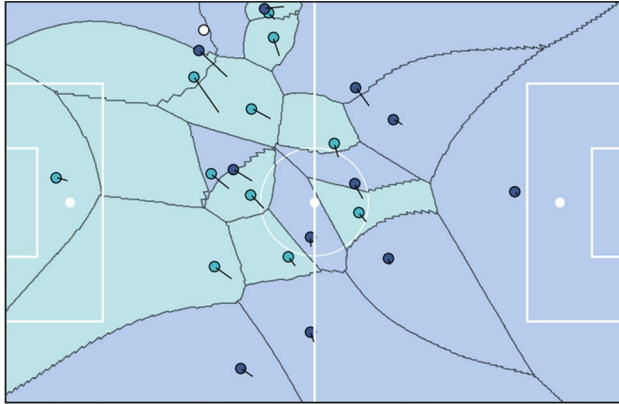
As another (straightforward) baseline, we incorporated a Gaussian in the experiments. Naturally, a Gaussian is trivial to estimate and turns out efficient in time and space. However, these advantages imply assumptions on the densities being unimodal and symmetric. While some scenarios may support these claims, others may suffer from this oversimplification, as shown in Fig. 8. It depicts the movements of a goal keeper and the point cloud has clearly two modes. Estimating the density of the points with a Gaussian introduces unnecessary errors by assigning a great deal of probability mass to unlikely events. Again, the proposed contextual models do not make any assumption on the nature of the densities and adapt to any multi-modal distribution of points.

A practical application for movement models is to compute so-called zones of control that describe the areas which are controlled by a player. The underlying idea is that the payer who controls the zone is expected to arrive first at every position within this area (Taki and Hasegawa 2000; Gudmundsson and Wolle 2014; Horton et al. 2015; Spearman et al. 2017; Brefeld et al. 2019). Figure 9 shows an example. Using any of the CFlow models to compute the zones of control removes the necessity of binning velocities and time horizons; computation time and memory requirements are very low, even when training on massive amounts of data, compared to KDE-based approach in Brefeld et al. (2019). Our models thus allow for (near) real-time processing of live data and may be used in live analysis and broadcasts to visualize key situations.

Finally, while movement models, by definition, compute the area of possible (near-) future positions, they do not allow for predicting positions. Our contextual movement



**Fig. 8** An example of the set  $S_{t_{\Delta}}^S$  (left) and corresponding kernel density estimate (right) for a goal keeper



**Fig. 9** Soccer pitch showing players and the controlled spaces per player derived from the individual movement models

models, however, may overcome this issue by explicitly conditioning the models also on positions of other players (CFlow-extended) and the ball, ball possession indicators, etc. In principle, any CFlow should adapt to the current positioning of players and ball on the pitch, hence, shifting the probability mass toward areas that better represent motions of professional athletes given the current situation. Although these assumptions need to be confirmed in additional (empirical) studies and are clearly out of scope for the present paper, the sheer possibility indicates the potential impact and importance of this line of work.

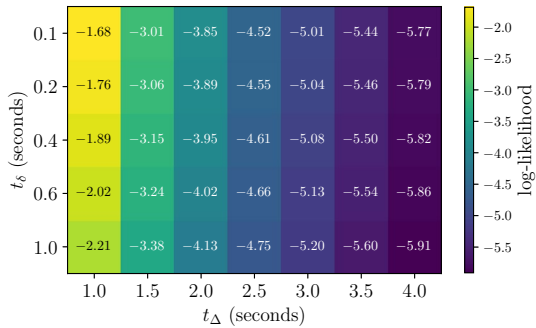
## 7 Conclusion

In this paper, we studied the problem of learning movement models in a purely data-driven fashion, which we evaluated on data from professional soccer matches. We circumvented the limitations of previous approaches and cast the problem as a conditional density estimation task. We exploited characteristics of the low-dimensional problem formulation to devise conditional normalizing flows for modeling movements. In contrast to the state of the art, our contextual model consists of only a single model. Having a conditional model proved important for the integration of contextual information, such as velocities of movements. In principle, any relevant information can be used as context. Moreover, they outperformed all competitors by far when it came to predictive performance and turned out very efficient. Their computation times were comparable to trivial baselines and orders of magnitude faster than the state of the art.

### Appendix 1: Influence of $t_\delta$

Figure 10 shows the log-likelihood values of CFlow for the same setting as Fig. 6, but now also varying  $t_\delta$ . The figure shows there is little influence from varying  $t_\delta$  when compared to  $t_\Delta$ . Although a choice of  $t_\delta = 0.1s$  seems to be better, we follow previous work of Brefeld et al. (2019) and use  $t_\delta = 0.2s$ .

**Fig. 10** Log-likelihood values computed by CFlow in the same setting as Figure 6



### Appendix 2: Details of CFlow-extended

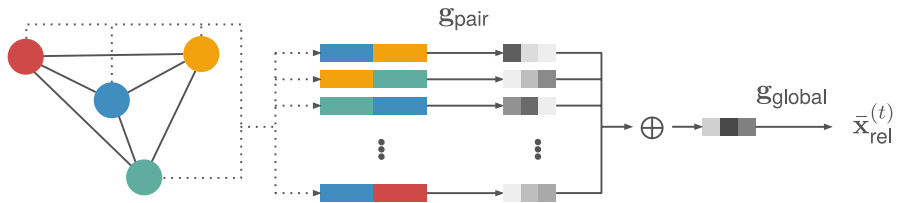
CFlow-extended uses contextual information given by  $\mathbf{c}^{(t)} = (v^{(t)}, t_\Delta, \bar{\mathbf{x}}_{rel}^{(t)})$ , which extends the current speed of the player  $v^{(t)}$  and the time horizon  $t_\Delta$  by the positions of the other players, relative to the player under consideration, i.e.,  $\bar{\mathbf{x}}_{rel}^{(t)}$ . For soccer, there are 21 remaining players whose relative positions are summarized in the set  $\mathcal{X}_{rel}^{(t)} = \{\mathbf{x}_1^{(t)}, \dots, \mathbf{x}_{21}^{(t)}\}$ . However, instead of simply stacking those positions into a vector, we leverage an aggregated representation  $\bar{\mathbf{x}}_{rel}^{(t)} \in \mathbb{R}^{\frac{1}{2}d_{hidden}}$  to circumvent the problem of ordering those players. Here,  $d_{hidden}$  is the dimensionality of the aggregated representation  $\bar{\mathbf{x}}_{rel}^{(t)}$ . The latter is obtained by employing a relation network Santoro et al. (2017) and defined as

$$\bar{\mathbf{x}}_{rel}^{(t)} = \mathbf{g}_{global} \left( \frac{1}{n_{rel}^2} \sum_{i=1}^{n_{rel}} \sum_{j=1}^{n_{rel}} \mathbf{g}_{pair}(\mathbf{x}_i^{(t)}, \mathbf{x}_j^{(t)}) \right),$$

where  $n_{rel} = |\mathcal{X}_{rel}^{(t)}|$  and both  $\mathbf{g}_{pair} : \mathbb{R}^{2d} \rightarrow \mathbb{R}^{d_{hidden}}$ ,  $\mathbf{g}_{global} : \mathbb{R}^{d_{hidden}} \rightarrow \mathbb{R}^{\frac{1}{2}d_{hidden}}$  are small feedforward neural networks with architectures outlined in Table 4. The output of this aggregation step,  $\bar{\mathbf{x}}_{rel}^{(t)}$ , is then used by the conditioning networks  $CN_{ca}(\mathbf{c})$  and  $CN_{cac}(\mathbf{c})$  whose input dimensionalities increase by  $\frac{1}{2}d_{hidden}$ . An example of the relation network is depicted in Fig. 11. The remainder of the flow is then carried out in the same way as CFlow.

**Table 4** Network architectures for CFlow-extended

Network	Architecture and activation functions
$\mathbf{g}_{pair}$	$2d \xrightarrow{SELU} d_{hidden} \xrightarrow{SELU} d_{hidden} \xrightarrow{SELU} d_{hidden}$
$\mathbf{g}_{global}$	$d_{hidden} \xrightarrow{SELU} d_{hidden} \xrightarrow{SELU} d_{hidden} \xrightarrow{SELU} \frac{1}{2}d_{hidden}$



**Fig. 11** An example of the relation network with only four objects  $\mathbf{x}_i^{(t)}$  for simplicity. The colored nodes represent the objects  $\mathbf{x}_i^{(t)}$

**Acknowledgements** The authors would like to thank Hendrik Weber, Deutsche Fußball Liga (DFL), and Sportcast GmbH for providing positional data. This research was funded in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil, Finance Code 001 and by FAPESP (grants #2018/19350-5, #2017/20945-0, #2016/50250-1, #2017/24005-2, #2019/17729-0, #2015/24494-8). This research was partially funded by the RFF PastOPol project.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Besse, P.C., Guillouet, B., Loubes, J., Royer, F.: Destination prediction by trajectory distribution-based model. *IEEE Trans. Intell. Transp. Syst.* **19**(8), 2470–2481 (2018)
- Bradbury, J., Frostig, R., Hawkins, P., Johnson, M.J., Leary, C., Maclaurin, D., Wanderman-Milne, S.: JAX: composable transformations of Python+NumPy programs. <http://github.com/google/jax> (2018)
- Brefeld, U., Lasek, J., Mair, S.: Probabilistic movement models and zones of control. *Mach. Learn.* **108**(1), 127–147 (2019)
- Byrne, M., Parry, T., Isola, R., Dawson, A.: Identifying road defect information from smartphones. *Road Trans. Res.* **22**(1), 39–50 (2013)
- De Cao, N., Titov, I., Aziz, W.: Block neural autoregressive flow. In: *35th Conference on Uncertainty in Artificial Intelligence (UAI19)* (2019)
- Dick, U., Brefeld, U.: Learning to rate player positioning in soccer. *Big data* **7**(1), 71–82 (2019)
- Dinh, L., Krueger, D., Bengio, Y.: NICE: non-linear independent components estimation. In: *2015 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings (2015)
- Dinh, L., Shol-Dickstein, J., Bengio, S.: Density estimation using Real NVP. In: *International Conference on Learning Representations* (2017)
- Fujimura, A., Sugihara, K.: Geometric analysis and quantitative evaluation of sport teamwork. *Syst. Comput. Japan* **36**(6), 49–58 (2005)
- Gomez-Gonzalez, S., Prokudin, S., Schölkopf, B., Peters, J.: Real time trajectory prediction using deep conditional generative models. *IEEE Robot. Autom. Lett.* **5**(2), 970–976 (2020)

- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. *Adv. Neural Inf. Process. Syst.* **27**, 2672–2680 (2014)
- Gottfried, B.: Representing short-term observations of moving objects by a simple visual language. *J. Vis. Lang. Comput.* **19**(3), 321–342 (2008)
- Gottfried, B.: Interpreting motion events of pairs of moving objects. *GeoInformatica* **15**(2), 247–271 (2011)
- Gudmundsson, J., Wolle, T.: Football analysis using spatio-temporal tools. *Comput., Environ. Urban Syst.* **47**, 16–27 (2014)
- Haase, J., Brefeld, U.: Mining positional data streams. In: *International Workshop on New Frontiers in Mining Complex Patterns*, pp. 102–116. Springer, Cham (2014)
- Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., et al.: Array programming with NumPy. *Nature* **585**(7825), 357–362 (2020)
- Horton, M., Gudmundsson, J., Chawla, S., Estephan, J.: Automated classification of passing in football. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 319–330. Springer, Cham (2015)
- Huang, C.W., Krueger, D., Lacoste, A., Courville, A.: Neural autoregressive flows. In: *International Conference on Machine Learning*, pp. 2078–2087 (2018)
- Hübl, F., Cvetojevic, S., Hochmair, H., Paulus, G.: Analyzing refugee migration patterns using geo-tagged tweets. *ISPRS Int. J. Geo-Inf.* **6**(10), 302 (2017)
- Janetzko, H., Sacha, D., Stein, M., Schreck, T., Keim, D.A., Deussen, O.: Feature-driven visual analytics of soccer data. In: *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 13–22 (2014)
- Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: *2015 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)
- Kingma, D.P., Dhariwal, P.: Glow: generative flow with invertible 1x1 convolutions. In: *Advances in Neural Information Processing Systems*, pp. 10215–10224 (2018)
- Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: *2014 2nd International Conference on Learning Representations (ICLR)*, Banff, AB, Canada, April 14–16, 2014, Conference Track Proceedings (2014)
- Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks. *Adv. Neural Inf. Process. Syst.* **30**, 971–980 (2017)
- Knauf, K., Memmert, D., Brefeld, U.: Spatio-temporal convolution kernels. *Mach. Learn.* **102**(2), 247–273 (2016)
- Laube, P., Imfeld, S., Weibel, R.: Discovering relative motion patterns in groups of moving point objects. *Int. J. Geogr. Inf. Sci.* **19**(6), 639–668 (2005)
- Le, H.M., Carr, P., Yue, Y., Lucey, P.: Data-driven ghosting using deep imitation learning. In: *MIT Sloan Sports Analytics Conference* (2017)
- Lu, Y., Huang, B.: Structured output learning with conditional generative flows. In: *Thirty-Fourth AAAI Conference on Artificial Intelligence* (2020)
- Mazimpaka, J.D., Timpf, S.: (2016) Trajectory data mining: a review of methods and applications. *J. Spat. Inf. Sci.* **13**, 61–99 (2016)
- McDermott, P.L., Wikle, C.K., Millspaugh, J.: Hierarchical nonlinear spatio-temporal agent-based models for collective animal movement. *J. Agric. Biol., Environ. Stat.* **6**(3), 294–312 (2017)
- Mohan, P., Padmanabhan, V.N., Ramjee, R.: Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. In: *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, SenSys '08*, pp. 323–336 (2008)
- Padberg-Gehle, K., Schneide, C.: Trajectory-based computational study of coherent behavior in flows. *PAMM* **17**(1), 11–14 (2017)
- Paefgen, J., Michahelles, F., Staake, T.: GPS trajectory feature extraction for driver risk profiling. In: *Proceedings of the 2011 International Workshop on Trajectory Data Mining and Analysis*, ACM, New York, NY, USA, TDMA '11, pp. 53–56 (2011)
- Papamakarios, G., Nalisnick, E., Rezende, D.J., Mohamed, S., Lakshminarayanan, B.: Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.* **22**(57), 1–64 (2021)
- Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: *International Conference on Machine Learning*, pp. 1310–1318 (2013)

- Rezende, D., Mohamed, S.: Variational inference with normalizing flows. In: *International Conference on Machine Learning*, PMLR, pp. 1530–1538 (2015)
- Rezende DJ, Mohamed S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. In: *International Conference on Machine Learning*, pp. 1278–1286 (2014)
- Rippel, O., Adams, R.P.: High-dimensional probability estimation with deep density models. arXiv preprint [arXiv:13025125](https://arxiv.org/abs/13025125) (2013)
- Santoro, A., Raposo, D., Barrett, D.G., Malinowski, M., Pascanu, R., Battaglia, P., Lillicrap, T.: A simple neural network module for relational reasoning. In: *Advances in neural information processing systems*, pp. 4967–4976 (2017)
- Scott, D.W.: *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, New York (2015)
- Spearman, W., Pop, P., Basye, A., Hotovy, R., Dick, G.: Physics-based modeling of pass probabilities in soccer. In: *Proceedings of the 11th MIT Sloan Sports Analytics Conference*, pp. 1–14 (2017)
- Sprado, J. and Gottfried, B., 2008, July. What motion patterns tell us about soccer teams. In: *Robot Soccer World Cup*, pp. 614–625. Springer, Berlin, Heidelberg
- Tabak, E.G., Turner, C.V.: A family of nonparametric density estimation algorithms. *Commun. Pure Appl. Math.* **66**(2), 145–164 (2013)
- Tabak, E.G., Vanden-Eijnden, E.: Density estimation by dual ascent of the log-likelihood. *Commun. Math. Sci.* **8**(1), 217–233 (2010)
- Taki, T., Hasegawa, Ji.: Visualization of dominant region in team games and its application to teamwork analysis. In: *Proceedings computer graphics international*, IEEE pp. 227–235 (2000)
- Taki, T., Hasegawa, J., Fukumura, T.: Development of motion analysis system for quantitative evaluation of teamwork in soccer games. In *Proceedings of 3rd IEEE International conference on image processing* **3**, 815–818 (1996)
- Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al.: Scipy 1.0: fundamental algorithms for scientific computing in python. *Nat. methods* **17**(3), 261–272 (2020)
- Winkler, C., Worrall, D., Hooeboom, E., Welling, M.: Learning likelihoods with conditional normalizing flows. arXiv preprint [arXiv:191200042](https://arxiv.org/abs/191200042) (2019)
- Zheng, S., Yue, Y., Hobbs, J.: Generating long-term trajectories using deep hierarchical networks. *Adv. Neur. Inf. Process. Syst.* **29**, 1543–1551 (2016)
- Zheng, Y.: Trajectory data mining: an overview. *ACM Trans. Intell. Syst. Technol.* **6**(3), 1–41 (2015)
- Zhong, J., Sun, H., Cao, W., He, Z.: Pedestrian motion trajectory prediction with stereo-based 3d deep pose estimation and trajectory learning. *IEEE Access* **8**, 23480–23486 (2020)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.