Long-term Prediction of Motion Trajectories Using Path Homology Clusters

J. Frederico Carvalho[1], Mikael Vejdemo-Johansson[2,3], Florian T. Pokorny[1] and Danica Kragic[1]

***Abstract*— In order for robots to share their workspace with people, they need to reason about human motion efficiently. In this work we leverage large datasets of paths in order to infer local models that are able to perform long-term predictions of human motion. Further, since our method is based on simple dynamics, it is conceptually simple to understand and allows one to interpret the predictions produced, as well as to extract a cost function that can be used for planning. The main difference between our method and similar systems, is that we employ a map of the space and translate the motion of groups of paths into vector fields on that map. We test our method on synthetic data and show its performance on the Edinburgh forum pedestrian long-term tracking dataset [1] where we were able to outperform a Gaussian Mixture Model tasked with extracting dynamics from the paths.**

I. Introduction

In order for robots to share their environment with people, it is necessary for them to reason about human motion, and use this capability to not inconvenience those they share their workspace with. We propose to address this problem through a motion prediction algorithm that leverages previously observed paths to predict the future motion of partial paths in an unsupervised manner.

Intuitively, we can formulate the problem as a model learning problem, where we hypothesize that human paths follow a certain model $M(\theta)$ and we want to find θ so that the error between the path predicted by the model, and the original path is minimized.

Current methods for this type of motion prediction fall into two categories, i) those which rely on the dynamics of motion, assuming that a subject moves in efficient paths, i.e. straight lines, and any deviations from such motions are due to obstacle avoidance which may include other agents [2]–[5]. ii) those that rely on training parametric models such as neural networks or Hidden Markov Models (HMM) that produce a prediction given past observations [6]–[9].

In the first instance, by assuming that human motion only deviates from a straight line to avoid obstacles, the methods ignore the possibility that subjects may have preferred pathways, and place the onus of predicting deviations from efficiency on the exactness of the tracker, or the environment model. Whereas in the second case, the obtained predictions may be represented in the form of a probability density as in [9] or a static model of occupancy as in [6] which necessitate one to be aware of the threshold of occupancy probabilities when using such methods for planning.

[1] CAS/RPL, KTH, Royal Institute of Technology, Stocholm, Sweden. {jfpbdc,dani,ftpokorny}@kth.se
[2] Mathematics Department, CUNY College of Staten Island, New York, USA. mvj@math.csi.cuny.edu
[3] Computer Science, CUNY Graduate Center, New York, USA.

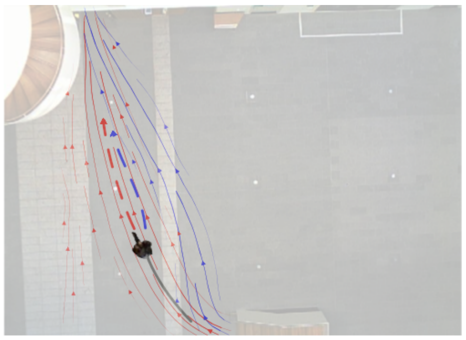
Fig. 1. Illustration of our method, a pedestrian path in an environment, together with two vector fields generated by pedestrian paths (red and blue), and the resulting different predicted paths (dashed).

In this paper we expand our previous work in path clustering [10], [11] and compute vector fields which represent the average motion of paths in each cluster. We exploit the nature of our distance metric, which ensures a cluster should cover a small area of the environment to ensure that different vector fields encode motions in different parts of the workspace. We employ the resulting fields to predict motion by integrating the field that most closely resembles the path observed so far.

To the best of our knowledge, our method is the first to employ insights obtained from the topology of the space of paths to extract higher order dynamics and use them for motion prediction.

II. Related work

Motion prediction is a fairly well-studied task and is closely related to object tracking, therefore rather similar methods can be used to address it. In [5] the authors present a Kalman filter based method for predicting the motion of pedestrians in 2D scene. In [8] The authors use HMMs to predict the motion of construction workers in a cluttered environment in order to improve building site safety. Similarly, in [9] the authors employ an HMM to the motion of people in an office environment, using both the immediate and far past observations to obtain a distribution over future observations. Similarly recurrent neural network models, have been used in [7] to infer driver intention in a roundabout. In [12] the authors employ Support Vector Machines (SVM) and K-means clustering with respect to the edit distance on a dataset of paths as a training step, subsequently future motions are predicted from a partial path by matching it to one fo the clusters.

One important application of motion prediction is the task of compliant motion planning, since by predicting the motion of a pedestrian in a scene, for example, a robot is able to plan so as to not interfere with the pedestrian's future path. In [13] the authors cluster paths to extract motion patterns using an expectation-maximization (EM) algorithm, these models are then used for prediction using a HMM which are used to synthesize compliant motion. An alternative approach was presented in [6] where the authors use a CNN to infer which parts of an environment are most traversed by humans, and use this in order to generate a robot motion that is non-obstructive. In [14] the authors use trajectory clustering in order to predict the motion of an agent, so that they are able to plan an intersecting path.

An area related to motion prediction is learning by demonstration. Here instead of using a model to infer where an object is moving toward, the model is used to instruct a robot how to move in order to perform some task. Popular approaches to this problem employ Gaussian Processes (GP) such as in [15] where the authors use local GP regression to perform a given task from a handful of demonstrations. Another common choice of model is a Gaussian Mixture Model (GMM) such as in [16] where the authors train a GMM from a handful demonstrations in order to obtain a compact model that generalizes a given task.

One way to extract motion primitives is through path clustering, where one intends to group a dataset of paths into subsets of paths that are pairwise similar. Generally, this is achieved through distance-based clustering methods, and a recent survey of such distance based methods can be found in [17]. Another possibility is to concentrate on path clusters that are fundamentally different with respect to their relationship to obstacles in the environments as in [10], and such methods can even be combined with a compatible distance function [18] to obtain more efficient clustering, as we showed in [11].

In this paper we present an algorithm for extracting motion primitives from datasets of paths in an efficient way. We do this by employing our path clustering pipeline from [11] to produce groups of paths that are deemed to be similar, and aggregating the observed velocities into a set of vector fields. To predict motion of a path up to a point, we choose the vector field which is most similar to the path observed so far, and extrapolate the motion by integrating that vector field.

The remainder of the paper is structured as follows: In Section III we give a brief explanation of simplicial complexes and their homology, followed by a short explanation of our notation conventions on vector fields. These will be used to extend our path clustering pipeline from [11], and to establish the establish the main contributions of the paper, respectively. In Section IV, we provide the aforementioned extensions to our path clustering pipeline, followed by an algorithm to calculate vector fields based on those path clusters as well as describe how we employ these vector fields for motion prediction. In Section V we describe our construction of synthetic datasets, and present the results from running our pipeline on such datasets, followed by the results of running the aforementioned pipeline on the dataset from [1].

III. Background

In this section we give a short overview of the background and terminology that will be used throughout the rest of the paper. Namely, we provide a short introduction to simplicial complexes and their homology, which will be necessary in order to prove that our alteration to the path distance metric from [11] produces the same clustering results. We also provide a short overview of our notation for vector fields which form the basis of our method for motion prediction.

A. Simplicial Complexes

Here we give a very short introduction to the notion of simplicial complexes. More detailed summaries can be found in [11], [19] as well as in more classical sources such as chapter 2 of [20].

Intuitively, a finite *simplicial complex* is a generalization of a triangulation. Formally it can be specified by a set X of subsets of $\{1, \dots, N\}$ such that for any $\sigma \in X$, and any non-empty $\tau \subset \sigma$, $\tau \in X$. Each element $\tau \in X$ is called a *face* of X.

If we consider a set of N points in $\mathbb{R}^2$, X can be seen as a triangulation where $\{x_i \mid i \in \sigma\}$ is the set of *vertices* of a *triangle* or *edge* when $|\sigma| = 3, 2$. However, we have to make a further restriction that convex hulls of $[\sigma] := \mathrm{Conv}(\{x_i \mid i \in \sigma\})$ satisfy $[\sigma] \cap [\tau] = \mathrm{Conv}(\{x_i \mid i \in (\sigma \cap \tau)\})$ only intersect at shared faces. The *dimension* of a simplex $\sigma \in X$ is defined given by $\dim(\sigma) = |\sigma| - 1$ and the dimension of the complex X is given by $\dim(X) = \max_{\sigma \in X} \dim(\sigma)$. We denote $X^{(i)}$ as the i-th *level* of X, containing all the i-dimensional simplices of X and $X^i = \bigcup_{j=0}^{i} X^{(j)}$, which is a simplicial complex called the i-th skeleton.

A simplicial complex X gives rise to a chain complex $\{C_i(X)\}_{i=1}^{\dim X}$. This corresponds to a sequence of vector spaces, where $C_i(X)$ has $X^{(i)}$ as a base together with a family of boundary maps $\partial_i : C_i(X) \to C_{i-1}(X)$ which send a basis element to a formal sum of its boundary elements with signs shifted according to the induced orientation, a more extensive description can be found in chapter 2 of [20]. An element $c \in C_i(X)$ is called an *i-chain* of the complex, furthermore if $\partial_i c = 0$ then c is called a *i-cycle* and if there exists some $b \in C_{i+1}(X)$ such that $\partial_{i+1} b = c$, then c is called a boundary.

It can be shown that in this setup every boundary is a cycle, and that cycles and boundaries form subspaces of the $C_i(X)$. The i-th *homology* group of X consists of equivalence classes of cycles in $C_i(X)$ which are not boundaries.

When dealing with paths, we view these as an ordered list of (oriented) edges on the simplicial complex which are placed end-to-end. These we regard as 1-chains of X, with integer weights. For example, a weight of 2 in a particular edge indicates that the edge is traversed twice, and a weight of -1 indicates that the edge is traversed once but in the negative direction. We call a 1-chain obtained from such a path a *path chain*. Note that our definition of simplicial complex implies that any two vertices share at most one edge, therefore a path chain p is uniquely determined by an ordered list of vertices $p_0, p_1, \ldots, p_n$.

We say that two path chains p, q are homologous if $p-q$ is a boundary, i.e. if there exists a 2-chain $c_{p,q}$ so that $\partial c_{p,q} = p-q$. In this case $c_{p,q}$ is said to be the *bounding chain* of $p-q$. However, the requirement that $p-q$ forms a cycle is in general too restrictive, therefore it will be useful to consider *quotients* of X. The quotient X/A is defined from the simplicial complex X by identifying every simplex in the subcomplex $A \subset X$ with a single point in the quotient.

B. Vector Fields

Given a subset of $X \subseteq \mathbb{R}^2$ we define a vector field V on X as a continuous function $V : X \to \mathbb{R}^2$. Such a vector field can model the motion of a particle by setting $\dot{x} = V(x)$. We define the (positive) flow of a vector field starting at x_0 as a function $\Phi_{V,x_0}(t) : J \to X$ where J is an interval in $\mathbb{R}$, containing 0 and $\Phi_{V,x_0}(t)$ satisfies $\frac{d}{dt}\Phi_{V,x_0}(t) = V(\Phi_{V,x_0}(t))$ and $\Phi_{V,x_0}(0) = x_0$. The existence of J and Φ_{V,x_0} in some region around x_0 is guaranteed by the Picard-Lindelöf theorem [21] provided that the vector field $V(x)$ is Lipschitz continuous in x in some neighborhood.

We employ a discrete approximation of vector fields in the plane that consists of an assignment $P \mapsto \mathbb{R}^2$ where P is a finite subset of $\mathbb{R}^2$. This can then be extrapolated to a vector field in each simplex of a triangulation of P by linear interpolation. Due to this approximation we guarantee that the obtained vector field is Lipschitz continuous.

IV. Method

We begin by describing a small alteration to our path clustering pipeline. In [11] we used a model of the quotient of the base space X by a subspace $A \subset X$ which encompasses the endpoints of the paths in the dataset. As noted Section III-A, this allows us to treat path chains as cycles and use that to find if they are homologous. We then defined the distance between the paths to be either the area of the bounding chain if tey are homologous, and infinity otherwise.

In the following we show that under mild assumptions a pair of paths in X/A can only be homologous if they begin in the same component of A and end in the same component of A. Therefore, under this condition the results from [11] are valid when instead of considering X/A we use the decomposition of A into its connected components $A_1 \cup, \ldots, \cup A_k$ and consider instead the *iterated quotient* $(\ldots(X/A_1)/\ldots)/A_k$ which we denote by $X/(A_1, \ldots, A_k)$. All proofs except for Lemma 1 are contained in appendix.

The following lemma establishes that the results we obtain are independent of the ordering of $A_1, \ldots, A_k$.

Lemma 1: Let $A_1, \ldots, A_k$ be disjoint subsets of X and let $\mathbb{S}_k$ be the set of permutations of k elements, then for any $\sigma \in \mathbb{S}_k$ we have $X/(A_1, \ldots, A_k) \cong X/(A_{\sigma_1}, \ldots, A_{\sigma_k})$.

For convenience, we will further restrict the subcomplexes $A_1, \ldots, A_k$ to satisfy the following:

Definition 1: We say that a sequence of subcomplexes $A_1, \ldots, A_k \subseteq X$ are *well disconnected* if for every simplex $\sigma \in X$, $\sigma \cap A_i \neq \emptyset$ implies $\sigma \cap A_j = \emptyset$ for all $j \neq i$.

The following proposition is a reformulation of Proposition 2 from [11] which establishes the uniqueness of bounding chains in our setting and guarantees that the distance function is well defined.

Proposition 1: Let X be a simplicial complex which can be embedded in $\mathbb{R}^2$ and $A_1, \ldots, A_k \subseteq X$ disconnected subcomplexes such that $H_1(A_i) = 0$, then given any 1-cycle c of $X/(A_1, \ldots, A_k)$, there exists at most one 2-chain s such that $\partial s = c$.

The following lemmas characterize the notion of cycle and boundary arising from path chains in an iterated quotient $X/(A_1, \ldots, A_k)$ when compared to the quotient X/A.

Lemma 2: Given two path chains $p = p_0, \ldots, p_n$ and $q = q_0, \ldots, q_m$ in a simplicial complex, then $p-q$ is a cycle if and only if both $p_0 = q_0$ and $p_n = q_m$ or p and q are both cycles.

Lemma 3: Let X be a simplicial complex embedded in $\mathbb{R}^2$ and $A_1, \ldots, A_k$ be well disconnected subcomplexes so that $H_1(A_i) = 0$. Let c be a path chain in X, then the associated chain $[c]$ is a boundary in $C_1(X/(A_1 \cup \ldots \cup A_k))$ if and only if it is a boundary in $C_1(X/(A_1, \ldots, A_k))$.

These results summarize the situation for path chains in the iterated quotient $X/(A_1, \ldots, A_k)$ for $X \subset \mathbb{R}^2$ and $A_1, \ldots, A_k$ well disconnected. Namely, two path chains p, q are homologous if and only if they were homologous in X/A, and if they are homologous, there is at most one bounding chain. This entails that our notion of area homology distance carries over to this setting without change.

A. Calculating Vector Fields

We define a vector field starting from the first order differences of the path and extrapolating to the rest of the space by using an underlying simplicial complex. In Algorithm 1 we provide the pseudocode for generating such a vector field from a path. The remainder of this subsection is dedicated to explaining how this algorithm works, and we defer to the next subsection to explain how we use the vector fields for motion prediction.

Definition 2: Let $\gamma = (t_0, p_0), (t_1, p_1), \ldots, (t_n, p_n)$ be a path, then we define the *differential* of γ at a given point p_i to be given by:

$$d\gamma(t_i) = \frac{p_{i+1} - p_i}{t_{i+1} - t_i}.$$

The differential defines a vector field associated to γ but only at the points p_i. To extrapolate this field to other points in $\mathbb{R}^2$ we first extrapolate it to neighboring points by averaging the contributions from neighbors (as illustrated in Fig. 2).

The main idea of Algorithm 1 is to establish the vector field along the path, and then use the edges of the mesh S to expand it to the rest of the space. This allows one to define the vector field as a simple $N \times 2$ matrix where N is the number of vertices S. For points which are not vertices of S the value of the vector field can be established through linear interpolation.

Algorithm 1: Calculate the vector field associated to a path

```
     input  : $S = (V, E)$
     input  : $P = (p_1, t_1), ..., (p_n, t_n)$
 1   $Lv_i \leftarrow \infty$                where $i \leftarrow 1, \ldots, \text{size}(V)$;
 2   $Npts_i \leftarrow 0$                  where $i \leftarrow 1, \ldots, \text{size}(V)$;
 3   $dP_i \leftarrow (p_i, \frac{p_{i+1}-p_i}{t_{i+1}-t_i})$     where $i \leftarrow 1, \ldots, n$;
 4   $vP_i \leftarrow (0, 0)$              where $i \leftarrow 1, \ldots, \text{size}(V)$;
 5   for $(p, dp) \leftarrow dP_0, dP_1, \ldots$ do
 6       $q \leftarrow \text{nearest}(V, p)$;
 7       $i \leftarrow \text{index}(q, V)$;
 8       $d \leftarrow dist(q, p)$;
 9       $vP_i \leftarrow vP_i + \text{DECAY}(d)dp$;
10       $Npts_i \leftarrow Npts_i + 1$;
11   $NextLv \leftarrow \{\}$;
12   for $i \leftarrow 0, \ldots, \text{size}(V)$ do
13       $n \leftarrow Npts_i$;
14       if $n > 0$ then
15           $vP_i \leftarrow vP_i/n$;
16           $Lv_i \leftarrow 0$;
17           append$(NextLv, neighborIdx(S, V_i))$;
18   $CurrLv \leftarrow 1$;
19   while $notEmpty(NextLv)$ do
20       $NewLv \leftarrow \{\}$;
21       for $i \leftarrow NextLv_0, NextLv_1, \ldots$ do
22           if $Lv_i < CurrLv$ then
23               skip
24           $Nbh \leftarrow neighborIdx(S, V_i)$;
25           $N \leftarrow 0$;
26           for $j \leftarrow Nbh_0, Nbh_1, \ldots$ do
27               if $Lv_j < CurrLv$ then
28                   $d \leftarrow dist(V_i, V_j)$;
29                   $vP_i \leftarrow vP_i + \text{DECAY}(d)vP_j$;
30                   $N \leftarrow N + 1$;
31               else
32                   append$(NewLv, j)$;
33           $vP_i \leftarrow \frac{vP_i}{N}$;
34       $NextLv \leftarrow eliminateRepeated(NewLv)$;
35       $CurrLv \leftarrow CurrLv + 1$;
36   return $vP$;
```

Algorithm 1 begins by initializing the vector field vP in line 3 to be zero at every point, and the variable dP in line 2 which stores the differential of the path P. Two further auxiliary variables are set: $Npts_i$, which is the number of points that are averaged when producing the estimation at the i-th vertex of the mesh, and Lv_i which indicates at which "level" the i-th point has been calculated. Here i is assumed to range over the index of all points in V.

In lines 5 to 10, the vector field is calculated at the nearest neighbors of the points in the path, using to this effect the values of dP. The expansion of the vector field to the mesh begins in earnest with the loop in lines 12 to 17. In it, the points which were previously assigned a vector field value through the nearest neighbor query, are assigned level zero, whereas their neighbors are inserted into the variable $NextLv$ which contains the next "level" of points to be analyzed.

Finally, the main loop in lines 19 to 35 proceeds by checking the neighbors of each element of $NextLv$ which has not been previously analyzed, and averaging the values of the vector field vP at those neighbors that are at a level lower than the current level $CurrLv$. The neighbors that have not been assigned a level yet are then added to the new level set $NewLv$. At the end of this loop the new level set replaces the level set and the current level is incremented.

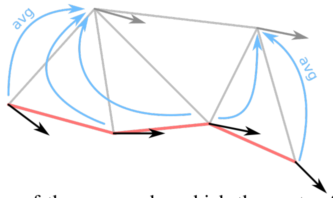

Fig. 2. Illustration of the process by which the vector field is expanded to neighboring vertices. The vertices along the line in red are in the path, and the black arrows on them indicate the value of the vector field along that path. The lines in light gray are the edges of the mesh, and the arrows in blue indicate which elements are averaged together before into the arrow in dark gray that represents the vector field at the opposite (w.r.t. the red path) end of the edge.

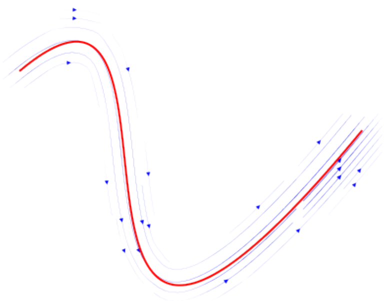

Fig. 3. Example of a path and the associated vector fields. Depicted are a simple curve γ in red with its associated vector field V_γ in blue computed by Algorithm 1. As we can see, the field forms a "corridor" around the path.

Finally, in order to obtain the vector field associated to a cluster of paths $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ we will have to average the different vector fields. To this end, we need to make the different fields comparable, which we achieve by renormalizing them so that the highest velocity is 1:

$$W_\gamma(x) = \frac{V_\gamma(x)}{\sup_{y \in X} V_\gamma(y)}$$

Similarly, we obtain the vector field V_Γ associated to the cluster Γ by combining the vector fields W_γ and renormalizing them the same way:

$$W_\Gamma(x) = \sum_{\gamma \in \Gamma} W_\gamma(x) \qquad V_\Gamma(x) = \frac{W_\Gamma(x)}{\sup_{y \in X} W_\Gamma(y)}$$

B. Cluster assignment and motion prediction

With the vector fields calculated we can finally assign a hypothesis cluster to a path that has only been partially observed. Namely, for each vector field we measure "how parallel" it is to the path that is being examined, and the corresponding cluster is then the cluster that is assigned i.e. for a given path $\eta = (t_1, \eta_1), \ldots, (t_k, \eta_l)$ we assign the cluster:

$$\Xi(\eta) = \underset{i=1,\ldots,k}{\arg\max} \sum_{j=1}^{l} \langle d\eta(t_j), V_{\Gamma_i}(\eta(t_j)) \rangle$$

and we predict the path to be the integral path of the vector field starting at the last point in the path while maintaining the current velocity by (numerically) solving the integral equation:

$$\tilde{\eta}(t) = \eta(t_l) + \|d\eta(t_l)\| \int_{t_l}^{t} \frac{V_{\Xi(\eta)}(\tilde{\eta}(s))}{\|V_{\Xi(\eta)}(\tilde{\eta}(s))\|} ds$$

V. Experiments

In this section, we present our experimental results. We start with two synthetic examples, where the data is drawn from a known distribution. The first set exemplifies an ideal scenario where the data is well separated, and our score function is ideally suited to distinguish between the different sets of data. The second dataset contains more complex dynamics, and showcases situations in which our score function can underperform and be led to produce misclassification errors. Finally, in the real world data example, we show how our method performs on the dataset from [1].

We compare our method with simple dynamics based motion prediction methods, the first one simply integrates the current speed of the path into the future, i.e. given the path γ up to time t we predict that at time $t+s$ the path is at $\gamma(t) + s\dot{\gamma}(t)$. The second can only be used when there is a known goal region, which is the case for the synthetic datasets, and works by setting an attractor on a target in the S in the goal region but mantains the norm of the velocity, so in essence we solve the initial value problem:

$$\frac{d}{dt}\begin{pmatrix}\gamma(t+s)\\ \dot{\gamma}(t+s)\end{pmatrix} = \frac{\|\dot{\gamma}(t)\|}{\|\dot{\gamma}(t+s)\|}\begin{pmatrix}\dot{\gamma}(t+s)\\ \beta(S-\gamma(t+s))\end{pmatrix}$$

where the parameter β is chosen to minimize the prediction error. In the real world dataset we use an alternative vector field model, that consists of modeling the vector field associated to each path cluster using a GMM as in [22]. All integrations are calculated using an explicit fourth order Runge-Kutta method with a step size of 0.01 on the synthetic datasets, and $(1/90)s^{-1}$ on the real-world dataset, this corresponds to ten steps per frame, as the paths were drawn from a tracking process running at 9 frames per second.

A. Simple Synthetic Data

In this test we employed a simple model where we draw paths in a 100×100 square in $\mathbb{R}^2$ (the distance units can be taken to be meters) and demark four regions:

- left near point $(0, 50)$ where all paths begin.
- right near point $(100, 50)$ where all paths end.
- top near point $(50, 100)$ which contains the midpoints of all paths in cluster 1.
- bottom near point $(50, 0)$ which contains the midpoints of all paths in cluster 2.

Each path is interpolated from three points (begining, middle and end with timestamps 0, 0.5 and 1 respectively) using a GP, the resulting paths can be seen in Figure 4. To draw a path from the distribution we first randomly choose either the top or bottom region, then we draw the prescribed three way points that define the path, and interpolate them as specified.

To test in this scenario we sampled randomly 1000 paths from the distribution and clustered them into two clusters, and calculated the associated vector fields. We tested the performance of the resulting model on 100 more paths also drawn from the distribution.

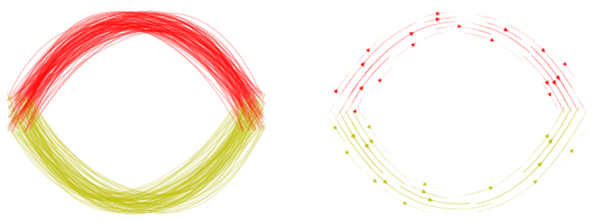

Fig. 4. Simple synthetic data set: The first cluster of paths is depicted in red, and the second one is depicted in yellow. To the right we show the stream plot with the vectorfields associated to each cluster in the respective color.

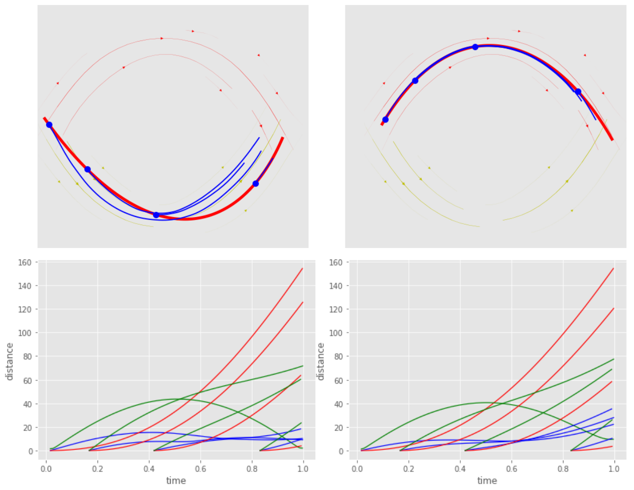

Fig. 5. Top row: Both plots contain a path in red drawn from the same distribution as the dataset used to calculate the vector fields. Each of the blue paths is predicted with our method starting from the large blue point at its extremity. Bottom row: Error for the predicted paths in the top row. The horizontal axis represents time, and the vertical axis represents the distance from the ground truth. Each of the blue,red and green lines represents the measured error when motion is predicted using our method (blue), constant velocity prediction (red) and an attractor at the goal region (green).

In Figure 5 we show an example of predictions from our method, together with the error incurred by both our method, and the constant-velocity and the attractor-towards goal models. The error is represented in time (of the path/predicted path which starts at the point where the prediction starts) versus distance between the ground-truth and the predicted path. As we can see our method is able to track the overall shape of a simple path, which on these instances allows it to outperform both other predictors.

In Figure 6 we present the statistics of the errors for the three different models, namely we plot the prediction horizon (that is, amount of time *after* the beginning of the prediction) versus error (again, distance between the predicted path and the ground truth at the given prediction horizon). For each model we plot the average (solid line) and standard deviation (shaded region) of the error at any given time. As we can see our observations from Figure 5 remain valid when we look at statistics of the error.

Note that in this case we used partial information from how the dataset was generated to calculate the vector fields, namely we used two path clusters, which generated two vector fields, each of which closely tracks the associated cluster, therefore

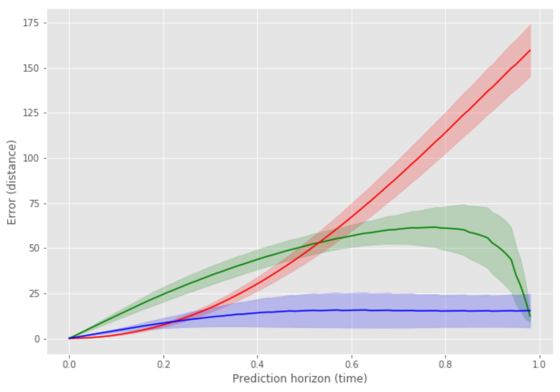

Fig. 6. Error statistics for the prediction models for the dataset in Figure 4. Here we plot the statistics for the three different models, our model is in blue, the constant velocity prediction is in red, and the attractor model is in green. The statistics shown are the mean error (solid line) and the standard deviation region (shaded region).

as long as the correct vector field is identified, the prediction error should remain low. Furthermore, since the vector fields almost do not overlap in space, and when they do, they are mostly perpendicular to each other, identifying the correct vector field becomes a simple task. Next we show another synthetic example where this separation between fields is not well demarcated, and therefore incurs in higher error predictions for our model.

B. More Complex Synthetic Data

With this example we aim to show that our model is able to encode more complex dynamics than a simple single curved motion towards a target, and to show where prediction errors may arise from in our model. To generate the data we use an almost identical procedure to the one before, except we have an extra "foreword" region near the point $(25, 0)$ where all paths go through. Once again, to draw a path from one cluster we draw four points (one from the beginning region, one from the foreword region, one from the top or bottom region, and one from the end region). The points are then given timestamps 0, 0.2 , 0.7 and 1 respectively, and are interpolated with a Gaussian process.

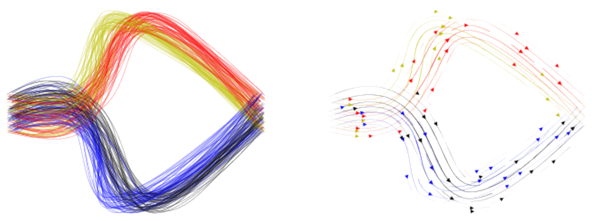

Fig. 7. A more complicated dataset, comprising four path clusters (red, blue, black and yellow). To the right we see the associated vector fields.

From the description of the generated dataset it would be expected that there would be only two clusters as in the previous example, however, our clustering method proposed a model with four clusters as seen in Figure 7 when maximizing the ratio of inter-cluster distance to in-cluster distance. In Figure 8 both demonstrate our method in the top row, and

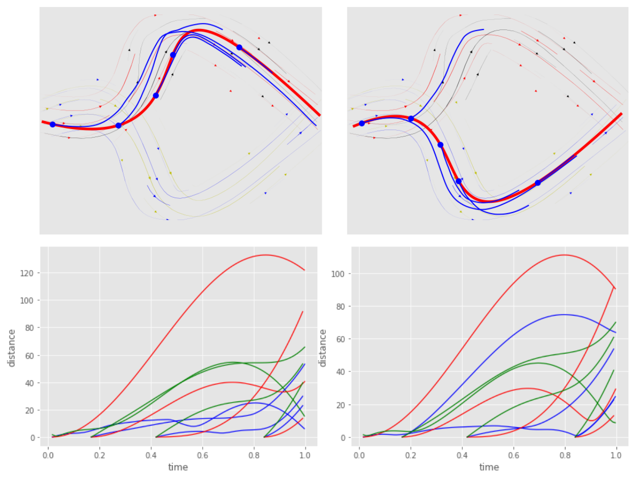

Fig. 8. Top row: Two paths from the dataset Figure 7 together with predictions performed using our method, as in the top row of Figure 5. Bottom row: Error for the predicted paths in the top row, the legend is the same as in the bottom row of Figure 5.

compare it with the two simple dynamics methods used in the previous example in the bottom row. From loooking at the predictions we notice that one of them follows the wrong vector field leading to a gigh prediction error. This happens because at the point of prediction all the fields are similar, this implies that the score functions will also have a similar value and leads to the possibility of missclassifications which may result in higher error than the simple models.

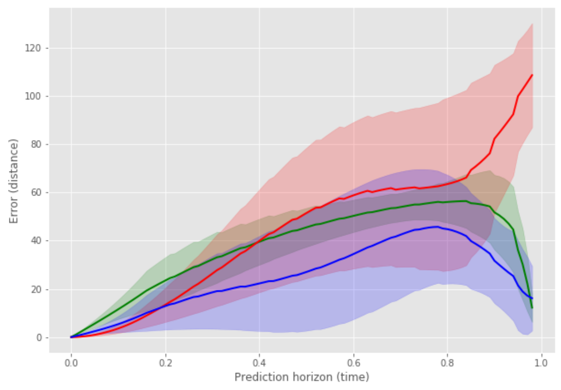

Fig. 9. Error statistics for the prediction models for the dataset in Figure 7. Here we plot the statistics for the three different models, our model is in blue, the constant velocity prediction is in red, and the attractor model is in green. The statistics shown are the mean error (solid line) and the standard deviation region (shaded region).

However, when we once again analyze the error statistics for the predictions in this dataset, we see that our model once again outperforms the simple dynamics models. However, the mean and standard deviations of the error at all horizons are now larger than they were in the simple dataset which comes from the fact that the possibility of misclassification leads to larger errors at all prediction horizons.

C. Real-world Data

We now repeat the analysis performed for the synthetic datasets, using instead the Edinburgh dataset [1]. This dataset comprises tracks of pedestrians in Edinburgh University, and part of it is pictured in Figure 10 (left).

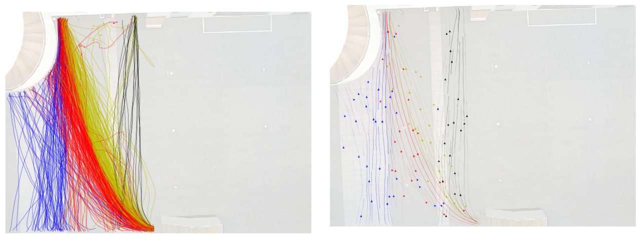

Fig. 10. Subset of the path clusters (left) and corresponding vector fields (right) associated to paths from the Edinburgh pedestrian tracking dataset [1] which share origin and destination regions.

However, due to high-frequency noise from measurement errors. For example, the positions output by the tracker are measured in pixels, so even when an object is stationary the position is expected to change due to naturally shifting lighting conditions. We thus need to preprocess the data so that velocities vary smoothly as expected. To this end we assume the noise is white, so that it can be filtered by using a simple 5-point moving average ($\hat{x}_t = (x_{t-2} + \cdots + x_{t+2})/5$).

In Figure 10 (right) we show a picture of the vector fields generated from the paths in the left, as we can see the vectorfields have a large overlap, which results in misclassification errors, as we previously pointed out. As we also pointed out before, we compare our method to using the simple constant velocity prediction and a GMM vector field approach [22]. In this model we calculate all pairs $(\gamma(t), \dot{\gamma}(t))$ from paths γ in some cluster Γ, which are then assumed to be samples drawn from a random variable (X_Γ, V_Γ) with domain $\mathbb{R}^2 \times \mathbb{R}^2$ and with according to a GMM. To choose the vector field to integrate in order to predict a new path γ we select the cluster Γ that maximizes the likelihood $\sum_{t=0}^{T} \log(p_{(X_\Gamma, V_\Gamma)}(\gamma(t), \dot{\gamma}(t)))$ and we calculate the value of eth vector field at point x as being the expected value $E(V_\Gamma | X_\Gamma = x)$.

However, when analyzing the predictions in Figure 11, we see that once again, in the long term our method is able to attain a smaller error than simply following the current velocity or using the GMM vector field model. Upon a closer inspection and comparing with Figure 10, we see that the paths on the top row are not always attributed to the same vector field. Namely the first two predictions of the path on the right follow the red cluster, the third prediction follows the blue cluster as the path curves upward more abruptly in a region where the red cluster does not.

Finally, the statistics in Figure 12, second our observations from the analysis of Figure 11 and we we can see that our method outperforms on average constant-velocity extrapolation and integrating the GMM vector field model, while also attaining lower variance.

VI. Conclusions and Future Work

In this paper we presented a method to extract motion primitives in a scalable way from large datasets of paths using piece-wise linear vector fields. We compared our method to other dynamics based methods, and observed that our approach is able to effectively capture more complex dynamics than conceptually simpler primitives, and in the tested datasets lead to more faithful predictions than a GMM vector field approach at all time horizons.

In future work we hope to improve our framework in several ways, namely by using sparse vector field models, instead of the dense model currently used, and using insights from sheaf theory [23] in order to provide more resilient prediction dynamics. We also intend to integrate our method into a tracking framework, for example a bayesian filter to test the system performance with our system dynamics.

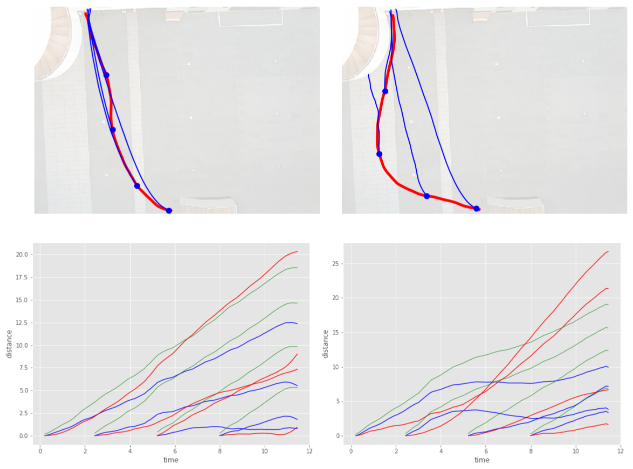

Fig. 11. Top row: two paths from the Edinburgh pedestrian tracking dataset and associated predictions using our method, starting at different points along the path. Bottom row: comparison of the prediction error using our method (blue) and constant velocity prediction (red), and GMM vector field integration (green).

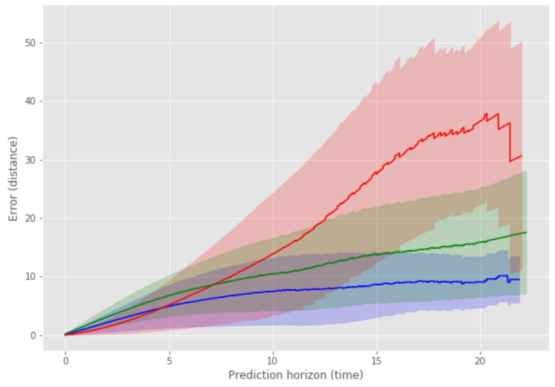

Fig. 12. Error statistics for the prediction models for the Edinburgh pedestrian tracking dataset. Depicted are the error statistics for our model (blue), constant velocity prediction (red) and the GMM vector field (green). The statistics shown are the mean error (solid line) and the standard deviation region (shaded region).

Appendix
Proofs of results from Section IV

A. *Proof of Proposition 1:*

We follow an argument similar to the proof of Proposition 2 in [11]. Namely when $A \subseteq X$ is a subcomplex, we can build the exact sequence of the pair [20].

$$\cdots \to H_n(A) \to H_n(X) \to H_n(X/A) \to H_{n-1}(A) \to \cdots$$

Since this sequence is in reduced homology, and each A_i is of dimension 2 and embedded in $\mathbb{R}^2$, therefore has $H_2(A_i) = 0$, furthermore it satisfies $H_1(A_i) = 0$ and since it is connected, it also satisfies $H_0(A_i) = 0$ (since it is reduced homology). Therefore $H_n(A_i) = 0$ for every n and there is an isomorphism in homology between $H_n(X)$ and $H_n(X/A_i)$ furthermore since the A_i are disjoint, the subcomplex $A_i/A_j \subseteq X/A_j$ is isomorphic to A_i and so still satisfies $H_n(A_i) = 0$.

Now, this implies that in the conditions of the proposition $H_n(X/(A_1, \dots, A_k)) \cong H_n(X)$ for every n, and therefore $H_2(X/(A_1, \dots, A_k)) = 0$ which implies that for any 1-chain c there exists at most one 2-chain s so that $\partial s = c$. ■

B. Proof of Lemma 2:

Since p is a path chain, $\partial p = (p_1 - p_0) + (p_2 - p_1) + \dots + (p_n - p_{n-1}) = p_n - p_0$. In a similar fashion we have $\partial q = q_m - q_0$ and so $\partial(p - q) = 0$ if and only if $p_n - p_0 = q_m - q_0$, i.e. if and only if $p_n = q_m$ and $p_0 = q_0$ or $p_n = p_0$ and $q_m = q_0$. ■

C. Proof of Lemma 3:

Assume, without loss of generality that $k = 2$, and therefore the relative region has only two connected components A_1, and A_2. This decomposition induces a factorization of the quotient $X \to X/(A_1 \cup A_2)$ through the iterated quotient $X/(A_1, A_2)$ (which corresponds simply to the identification of the points representing each of the connected components). Call these maps ϕ and ψ:

$$X \xrightarrow{\phi} \frac{X}{A_1, A_2} \xrightarrow{\psi} \frac{X}{A_1 \cup A_2}$$

Since ϕ and ψ are simplicial maps, they induce chain homomorphisms $\phi_*([c]) = [\phi(c)]$, and $\psi_*([d]) = [\psi(d)]$. By definition of chain homomorphism, they commute with the differential, meaning $\partial \circ \psi_*([d]) = \psi_*(\partial[d])$. Therefore if $\phi_*([c])$ is a boundary, then it can be written as $\partial[b]$ for some 2-chain $[b]$ which means.

$$\psi_*(\phi_*([c])) = \psi_*(\partial[b]) = \partial(\psi_*([b]))$$

i.e. $\psi_* \circ \phi_*([c]) = [\psi \circ \phi(c)]$ is also a boundary. To prove the reciprocal assume now that c is a path chain, and $[\psi \circ \phi(c)]$ is a boundary. Note once again, that all that ψ does is identify the points representing A_1 and A_2, leaving every other simplex in $X/(A_1, A_2)$ intact. Furthermore if we let the points in A_1, A_2 be the first two points in the ordering of $(X/(A_1, A_2))^{(0)}$, we note that since A_1, A_2 are well disjoint, so they do not share an edge, and hence ψ preserves simplex orientations. With that in mind, since any $[b]$ such that $\partial[b] = \psi_* \circ \phi_*([c])$ is a 2-chain, we can calculate the preimage of $[b]$ by ψ_*, which is itself a well defined 2-chain $[b^*]$ satisfying:

$$\psi_*(\phi_*([c])) = \partial\psi_*([b^*]) = \psi_*(\partial[b^*])$$

And again since ψ_* identifies the 0-simplices corresponding to A_1 and A_2 it is an isomorphism on 1-chains, and therefore $\phi_*([c]) = \partial[b^*]$ meaning, $\phi_*([c])$ is a boundary. ■

Acknowledgements

The authors would like to thank Patric Jensfelt for his help in the writing of this paper. This work was supported by the Knut and Alice Wallenberg Foundation and the Swedish Research Council.

References

[1] B. Majecka, “Statistical models of pedestrian behaviour in the forum,” Master’s thesis, School of Informatics, University of Edinburgh, 2009.

[2] S. Kim, A. Bera, A. Best, R. Chabra, and D. Manocha, “Interactive and adaptive data-driven crowd simulation,” in *2016 IEEE Virtual Reality (VR)*, Mar. 2016.

[3] D. Helbing and P. Molnár, “Social force model for pedestrian dynamics,” *Physical Review E*, 1995.

[4] S. Kim, S. J. Guy, W. Liu, D. Wilkie, R. W. Lau, M. C. Lin, and D. Manocha, “Brvo: Predicting pedestrian trajectories using velocity-space reasoning,” *The International Journal of Robotics Research*, vol. 34, no. 2, 2015.

[5] A. Bera, S. Kim, T. Randhavane, S. Pratapa, and D. Manocha, “Glmp - realtime pedestrian path prediction using global and local movement patterns,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016.

[6] J. Doellinger, M. Spies, and W. Burgard, “Predicting occupancy distributions of walking humans with convolutional neural networks,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, Jul. 2018.

[7] A. Zyner, S. Worrall, and E. Nebot, “A recurrent neural network solution for predicting driver intentions at unsignalized intersections,” *Robotics and Automation Letters*, vol. 3, no. 3, Jul. 2018.

[8] K. M. Rashid and A. H. Behzadan, “Enhancing motion trajectory prediction for site safety by incorporating attitude toward risk,” in *Computing in Civil Engineering 2017*, 2017.

[9] Z. Wang, P. Jensfelt, and J. Folkesson, “Multi-scale conditional transition map: Modeling spatial-temporal dynamics of human movements with local and long-term correlations,” in *2015 IEEE/RSJ, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*, 2015.

[10] F. T. Pokorny, K. Goldberg, and D. Kragic, “Topological trajectory clustering with relative persistent homology,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2016.

[11] J. F. Carvalho, D. Kragic, M. Vejdemo-Johansson, and F. T. Pokorny, “Path clustering with homology area,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2018.

[12] S. Xiao, Z. Wang, and J. Folkesson, “Unsupervised robot learning to predict person motion,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015.

[13] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, “Learning motion patterns of people for compliant robot motion,” *The International Journal of Robotics Research*, vol. 24, no. 1, 2005.

[14] C. Sung, D. Feldman, and D. Rus, “Trajectory clustering for motion prediction,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012.

[15] M. Schneider and W. Ertel, “Robot learning by demonstration with local gaussian process regression,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010.

[16] S. Calinon, T. Alizadeh, and D. G. Caldwell, “On improving the extrapolation capability of task-parameterized movement models,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov. 2013.

[17] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah, “Time-series clustering – A decade review,” *Information Systems*, vol. 53, Oct. 2015.

[18] E. W. Chambers and M. Vejdemo-Johansson, “Computing minimum area homologies,” *Comput. Graph. Forum*, vol. 34, no. 6, 2015.

[19] J. F. Carvalho, M. Vejdemo-Johansson, D. Kragic, and F. T. Pokorny, “An algorithm for calculating top-dimensional bounding chains,” *PeerJ Computer Science*, vol. 4, May 2018.

[20] A. Hatcher, *Algebraic Topology*. Cambridge University Press, 2002.

[21] S. Lang, *Fundamentals of Differential Geometry*. New York, NY: Springer US, 1999.

[22] S. Calinon, F. Guenter, and A. Billard, “On learning, representing and generalizing a task in a humanoid robot,” *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 37, no. 2, 2007.

[23] M. Robinson, “Sheaves are the canonical data structure for sensor integration,” *Information Fusion*, vol. 36, 2017.