UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Karl Riis

# Forecasting Human Trajectories with Uncertainty Estimation

Master's Thesis (30 ECTS)

Supervisor(s): Meelis Kull, PhD
Novin Shahroudi, MSc

Tartu 2022

## Forecasting Human Trajectories with Uncertainty Estimation

**Abstract:**
Human trajectory forecasting is a task which has been getting increasingly more attention in recent years. It is often used in robotics research as autonomous robots have to be well aware of the movement patterns of surrounding pedestrians to ensure safe and collision-free navigation. Many recent trajectory prediction works have been focused on neural network based solutions which need to be trained on large amounts of data. We propose a new generative trajectory forecasting method which does not need to be previously trained and is algorithmically simple and intuitive. Our method produces a multi-modal output to convey the uncertainty in human motion and is configurable with a set of parameters to adapt it to various environments. We show that our method performs nearly as good and in some cases better than state-of-the-art forecasting models when considering the task of predicting trajectories in an unseen environment. The results indicate that when deploying a forecasting model in an environment for which there is not a lot of data available, a neural network can be rivaled by a simpler approach.

**Keywords:**
Trajectory forecasting, uncertainty estimation

**CERCS:** P176 - Artificial intelligence

## Inimeste trajektooride ennustamine koos määramatuse hindamisega

**Lühikokkuvõte:** Inimeste liikumistrajektooride ennustamine on ülesanne, millele on hiljuti hakatud üha rohkem tähelepanu pöörama. See on eriti aktuaalne teema robootikas, kuna iseliikuvad robotid peavad olema teadlikud enda ümbruses liikuvate inimeste liikumisteekondandest, et tagada ohutu manööverdamine. Paljud hiljutised trajektooride ennustamise lahendused on üles ehitatud tehisnärvivõrkudele, mistõttu vajavad nad treenimist suurel andmemahtudel. Antud töös esitame uue generatiivse trajektooride ennustamise meetodi, mis ei vaja andmetel treenimist ning on algoritmiliselt intuitiivne. Meie esitatud meetod loob multimodaalsed ennustused, et kajastada määramatust inimeste liikumises. Meetod hõlmab endas lihtsasti konfigureeritavaid parameetreid, mille läbi saab optimeerida selle tegevust erisuguste keskkondade jaoks. Viime läbi katsed ülesandel, kus tuleb inimeste liikumistrajektoore ennustada keskkonnas, mille kohta ajaloolised andmed puuduvad. Tulemused näitavad, et ennustamisel tundmatus keskkonnas on meie meetodi sooritus peaaegu sama hea ja kohati parem kui närvivõrkudel põhinevatel tippmudelitel.

**Võtmesõnad:**
Trajektooride ennustamine, määramatuse hindamine

**CERCS:** P176 - Tehisintellekt

# Contents

# 1 Introduction

Human trajectory forecasting is the task of predicting the future path of a person by using the information about their past movement history [Schöller et al., 2020]. This problem has been studied well in the field of robotics research. To ensure the safety of its operation, an autonomous robot has to have some premonition about the future actions of its surrounding pedestrians [Mangalam et al., 2020], thus some sort of trajectory prediction solution is needed.

Many of the latest works in trajectory forecasting have been focused on creating neural network based methods with an emphasis on modeling complex interactions between people [Schöller et al., 2020]. An inherent issue that such an approach has to face is that pedestrian movement datasets contain environmental priors, i.e. obstacles or certain favorable paths in the observable scene which will be learned by the neural network. This can result in poor generalization to new datasets by the model.

Another aspect to consider is that in a realistic setting, a person has many possible future paths they can take [Kosaraju et al., 2019]. For example, when they encounter an obstacle in their path, they may avoid it by going left or right. Similarly, they may change their movement speed at any moment. These are potential future actions which might not be apparent from the person's historical movement data. Due to this, it is often not enough for a trajectory forecasting model to produce a single prediction, instead multi-modal predictions are necessary to cover a distribution of multiple possible future paths.

This thesis aims to show that a competitive trajectory forecasting method can be created without the use of neural networks and the need for large amounts of historical data. We propose a physics-based generative model which does not need to be trained on previous data. The model can be tuned with a set of parameters to adapt it for different environments and it can produce multiple future trajectory predictions with probabilities to convey the uncertainty in human motion. The ultimate motivation behind developing the method was to create a prediction model which could be integrated with an autonomous robot developed by the robotics research group at the University of Tartu Institute of Technology as future work.

This thesis is divided into four main sections. Section 2 provides an overview of the background of human trajectory forecasting. We give a general introduction to the

motion planning component in robots, we formulate the trajectory forecasting problem and discuss the topic of uncertainty in the future trajectories of moving humans.

In Section 3 we explain our main contribution in detail - a new generative method for trajectory forecasting which does not need training and produces a multimodal output only based on the history of the observed instance.

Section 4 covers the datasets, the experimental setup and the evaluation measures which were used to compare our method with other trajectory forecasting works.

Our method is evaluated and compared to other trajectory forecasting methods, including some state of the art works, in Section 5.

# 2 Robotics and trajectory forecasting

The task of forecasting the future trajectories of moving humans is a well-studied topic in robotics research. Having information about the possible future movements of surrounding pedestrians is a crucial skill for autonomous robots to ensure safe and smooth operation without collisions [Kosaraju et al., 2019].

The motivation behind this thesis was to develop a trajectory forecasting method for the purpose of integrating it with a robot being developed by the robotics research group in the University of Tartu. Since trajectory prediction methods are used by the motion planner component of robots, we give a general overview of motion planning in robotics in Section 2.1.

The general task of human trajectory forecasting is formulated in Section 2.2. A brief overview of different approaches in recent trajectory forecasting works is given as well.

Section 2.3 covers the topic of uncertainty in future trajectories and how it can be accounted for when making predictions.

Section 2.4 introduces an easily understandable trajectory forecasting method, the constant velocity model.

## 2.1 Motion planning in robotics

One of the most complex tasks to solve for an autonomous robot is motion planning. The objective of motion planning is to be able to describe some goal for the robot in a high-level language which will be translated to low-level instructions for its mechanical parts, such as a control arm, wheels, etc [Choset et al., 2005]. Motion planning can be separated into four categories: navigation, coverage, localization, and mapping. Navigation is the problem of finding a collision-free path to a specified location. Coverage is the task of moving a tool or a sensor over some desired area. Localization is figuring out the robot's location or pose based on a map and sensor readings. Mapping is the problem of making sense of an unknown environment by the means of exploring and sensing.

Of the aforementioned domains of motion planning, navigation is the main task related to this thesis. The general goal of navigation is path planning - generating a geometric path from the robot's current state to some desired state [Gasparetto et al., 2012]. The simplest case for this is when the path has to be generated in a known static environ-

ment. The problem gets more complex when the robot has to operate in a space it is not familiar with and when kinematic or dynamic constraints are present, such as moving people.

Many recent works have focused on the problem of being able to account for the dynamic motions of surrounding pedestrians. Human trajectory forecasting methods are usually applied to solve this task by giving the robot an idea of how people might move so that a collision-free path could be generated for its motion.

It is also noteworthy that such path planning algorithms are often desirable to be run in real time [Svenstrup et al., 2010]. Thus they should be able to operate in a reasonably bounded time.

## 2.2 Human trajectory forecasting

Human trajectory forecasting is a useful and at times a necessary skill for autonomous robots which operate in populated areas. Anticipating the movement of nearby agents can help robots to plan their path more efficiently and smoothly by finding optimal trajectories for their movement and avoiding any potential colliding paths ahead of time [Rudenko et al., 2020]. Collision avoidance is an especially crucial concern for autonomous vehicles as they have to ensure the safety of vulnerable road users like pedestrians and cyclists.

The goal of forecasting the trajectory of a moving person is to predict the person's future path by taking into account their known movement history [Schöller et al., 2020]. We can imagine that we have a collection of $n$ historical locations of a person ordered by time, denoted as $L_{hist} = (l_1, ..., l_n)$, based on which we wish to predict the following $m$ positions $L_{future} = (l_{n+1}, ..., l_{n+m})$, where $l_i$ is the person's location data at timestep $i$. Usually the position $l_i$ is an x,y-coordinate pair $(x_i, y_i)$ which corresponds to the person's location in a top-down view of the observable area.

A survey in human trajectory forecasting methods by Rudenko et al. provides an insight into different prediction approaches presented in various works in recent years [Rudenko et al., 2020]. They find that methods can be divided into three main groups based on how they approach the modeling of human motion. Physics-based methods predict motion by creating an explicitly defined simulation based on Newton's laws of motion. Usually these methods are sequential, they only produce the prediction

for next timestep. Such prediction methods are then used recurrently to achieve a full trajectory prediction. Examples include the constant velocity model, which predicts that a person will continue moving with the same speed and direction as they were before, and the constant acceleration model, which assumes continual acceleration. Pattern-based methods on the other hand learn to approximate human motion dynamics from training data. These methods can be both sequential and non-sequential, the latter meaning that the distribution over long term trajectories is learned. Examples of this approach include neural networks and hidden Markov models. Planning-based methods reason about the person's long-term goals. They use a cost-function to predict human motion, which is either pre-defined or inferred from observations. These methods often make use of a map of the environment to condition the predictions on impassable obstacles and known end-locations which people want to reach.

## 2.3 Uncertainty in future trajectories

It is important to acknowledge that it is difficult to forecast the future trajectory of a moving person perfectly since human motion is stochastic [Mangalam et al., 2021]. Uncertainty in trajectory futures can come from two main factors: aleatoric variability and epistemic uncertainty.

Aleatoric variability is the randomness in human decisions. This means that at any point in time a person might make an unexpected turn to avoid an obstacle, stop moving or take any other action which is not foreseeable from their past movement history. Epistemic uncertainty is the unknown reasoning behind the path a human has chosen to take. People often have long-term goals for their paths, e.g. walking towards an entrance of a building they wish to visit which a forecasting model is not able to infer.

The uncertainty in human motion can be accounted for by incorporating multi-modality in trajectory predictions [Mangalam et al., 2020]. This can be achieved by predicting many possible path continuations or by predicting the distribution of potential continuations.

## 2.4 Constant velocity model

One of the simplest methods for trajectory forecasting is the constant velocity model or CVM. This physics-based method assumes that a person will continue walking with the same velocity and direction as they have been for some number of their latest timesteps [Schöller et al., 2020]. Despite its simplicity, CVM has shown fairly good results when compared against neural network based works and is often included as a baseline when evaluating new methods.

The method takes one person's location history $L_{hist} = (l_1, ..., l_n)$ as its input. Based on the history, a constant velocity $\Delta_{CONST}$ is calculated which will be recurrently added to the end of the last seen position of the person for $m$ future steps to attain the predicted trajectory continuation:

$$L_{pred} = (l_n + 1 * \Delta_{CONST}, l_n + 2 * \Delta_{CONST}, ..., l_n + m * \Delta_{CONST}) \tag{1}$$

The constant velocity $\Delta_{CONST}$ can be calculated in various ways. A common approach for calculating it is averaging the velocities of $n$ latest historical timesteps.

$$\Delta_{CONST} = \frac{1}{n-1} \sum_{i=1}^{n-1} \Delta_i \tag{2}$$

where $\Delta_i$ is the velocity between two subsequent positions of a person:

$$\Delta_i = l_{i+1} - l_i = (x_{i+1} - x_i, y_{i+1} - y_i) \tag{3}$$

Another approach is to calculate the constant velocity strictly on the two latest timesteps. This is more susceptible to measurement errors: if one of the two timesteps is noisy then the magnitude and direction of the velocity can be more affected by it than with the averaging method.

$$\Delta_{CONST} = l_n - l_{n-1} = (x_n - x_{n-1}, y_n - y_{n-1}) \tag{4}$$

An example of the constant velocity model's prediction can be seen in Figure 1. The figure depicts a top-down view of an observable scene, where the blue line signifies the historical movement path of a person over five timesteps. The distances between the historical locations are marked as $(\Delta_1, ..., \Delta_4)$. An average velocity $\Delta_{CONST}$ is

calculated based on these and then added to the end of the last known location of the person for four times to achieve a short-term prediction.

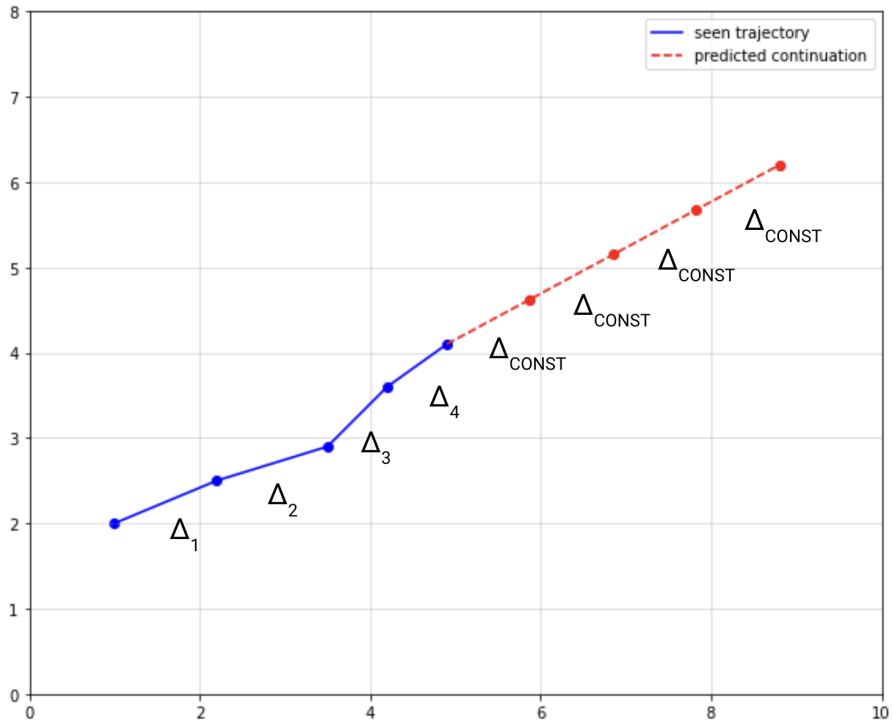


Figure 1. Example of the constant velocity model's prediction.

# 3 Physics based generative method

This section covers our new generative trajectory forecasting method. The purpose of developing the new method was to create a prediction algorithm which could be integrated with a robot's motion planner in future work, so that it would be able to account for the movement of surrounding pedestrians when calculating its own motion path. One of the main requirements for the method was that its output should be multimodal since human motion is uncertain. Another important constraint was that there was no historical data available for the target environment in which the robot would be deployed. This meant that the prediction method should perform well even in the case when there is no data or a minimal amount of data available for the deployment environment.

Due to the lack of historical data, we decided to develop a physics-based generative method for trajectory forecasting. It is based on physics in the sense that it leverages the constant velocity model to create future trajectories. We decided against a neural network oriented approach due to the concern that we would have had to train it on public datasets, based on which the network might not be able to generalize well to our target environment. Another reasoning was to develop an easily understandable solution, every step of our algorithm is well apprehensible to a human, as opposed to a black box solution a neural network would provide.

Our method takes one person's historical location data as input and produces multiple future trajectory predictions with probabilities as its output. The method is configurable with a set of parameters which control the number of predictions and manipulate the inner workings of the algorithm's different steps to adapt to varying environments.

The method consists of two main parts. Firstly, a large number of possible future trajectories are sampled based on a person's historical movement data. This is done by running a constant velocity model based generation algorithm many times with different initial conditions. The second step is creating a limited number of representative predictions based on the sampled trajectories. This is done by running kernel density estimation on the sampled trajectories, dividing them into multiple groups of probability ranges based on their densities, and using K-means clustering on each group to choose representative predictions. These steps are explained in greater detail in the following subsections. The code for our method is publicly available in a GitHub repository [1].

[1] https://github.com/karlriis/trajectory-forecasting

## 3.1 Sampling possible future trajectories

The first step of our generative method is sampling many possible future trajectories for a pedestrian given their historical movement data to cover a large space of possible trajectories. An example of the result of this step can be seen in Figure 2. The figure shows a top-down view of an observable area, where the blue dotted line is the historical motion path of a person and the colored thinner lines are 1000 future trajectories generated by our method. All future figures in this chapter will depict a top-down view of a single person as well unless stated otherwise.

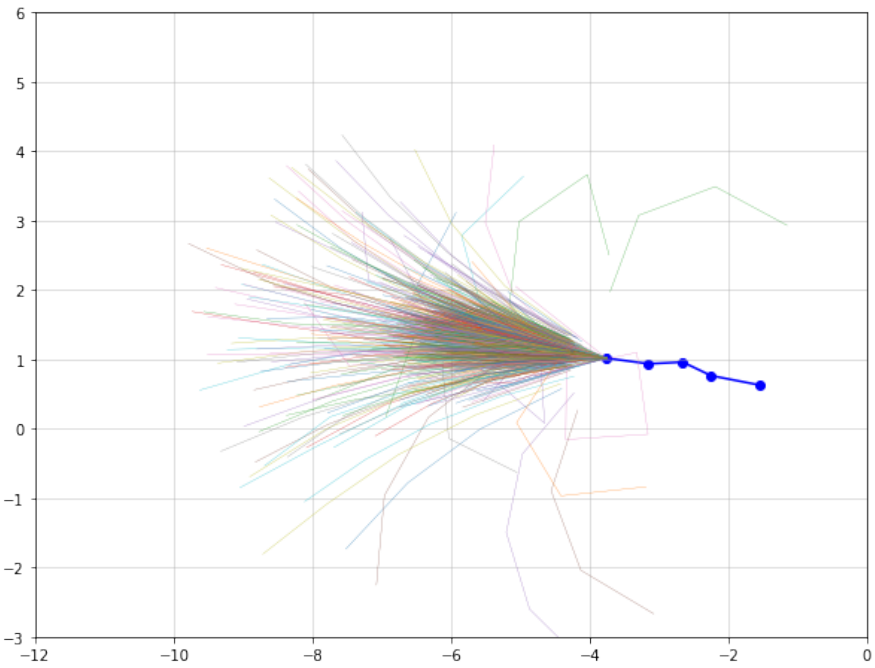


Figure 2. Example of the result of sampling 1000 possible future trajectories based on one person's movement history.

The basis of the trajectory generation is the constant velocity model. For our method, we further enhance the CVM by incorporating rotations into its predictions and inducing various random events at the generation of each future timestep. To attain many potential future trajectories for a person, we run the CVM-based generation many times for their historical trajectory. We introduce some noise to the historical data for each run, which

combined with the random events, produces varying future trajectories.

The calculation of the base velocity of the CVM and the random events happening at each generation timestep are controlled by parameters given to our method. These aspects are described in the subsections below.

### 3.1.1 Constant velocity model variations

We use two variations of the constant velocity model to generate the future trajectory samples. The first variation is the regular method described in Section 2.4. We use the approach of averaging the velocities in historical data to calculate the constant velocity $\Delta_{CONST}$. The second variation is an enhancement to the regular method by incorporating a constant angle change in predictions in addition to the velocity change. The angle is used in a similar way to the velocity, a constant angle is calculated based on the angle changes in the historical data, and then the angle is used to rotate all new predicted points. Figure 3 shows an example of a trajectory prediction by the CVM with the constant angle change included. Applying rotations for some of the samples allows our method to consider curving motion paths and thus cover a larger distribution of potential paths a person might take.

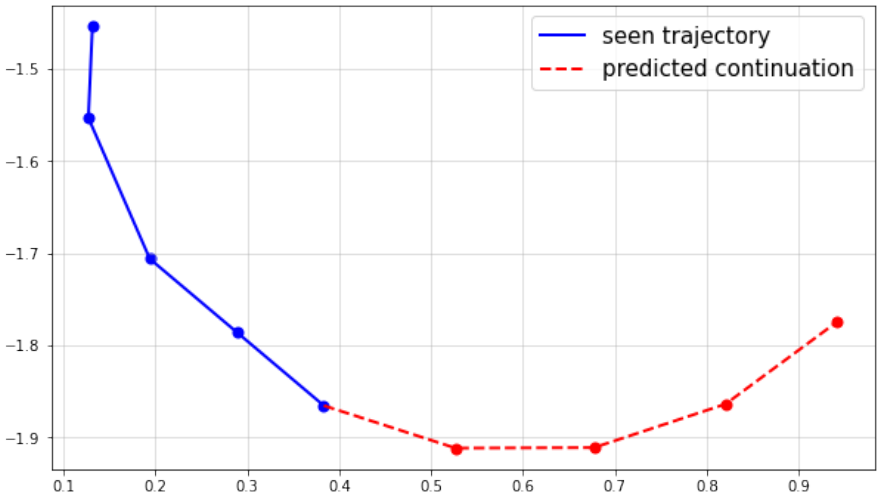


Figure 3. Example of the constant velocity model with constant rotation.

Each future trajectory is generated by using only one of the two aforementioned methods. The decision of which method to use is made by a parameter of our method - the probability of choosing the method with rotation $p_{rotation}$ which takes a value from the range $[0.0, 1.0]$. For each trajectory generation, the CVM with constant rotation is used with the probability of $p_{rotation}$, and the regular method is used with the probability of $1 - p_{rotation}$.

### 3.1.2 Calculating the initial velocity and angle

To use the constant velocity model to generate trajectories, a base velocity has to be calculated based on historical movement data which will be added to the end of the last known position of a pedestrian recurrently.

Prior to calculating the base velocity, our method adds some noise to the historical data $L_{hist} = (l_1, ..., l_n)$ for the generation of each future trajectory to get a new history $L'_{hist}$. This causes the historical data to be slightly different for the generation of each sample, introducing additional variance to the resulting distribution. The additional variance means that the generated future trajectories will differ from each other, covering a greater space of potential human movement. The noise is added to the x and y positions of each historical datapoint $l_i = (x_i, y_i)$ by sampling a new value for them from a Gaussian distribution with a mean of the previous value and a standard deviation of $\sigma_{history}$. The value for the standard deviation $\sigma_{history}$ is specified as a parameter for our method.

$$L'_{hist} = (l'_1, ..., l'_n) \tag{5}$$

where

$$l'_i = (x'_i, y'_i) \tag{6}$$

where

$$\begin{aligned} x'_i &\sim N(x_i, \sigma^2_{history}) \\ y'_i &\sim N(y_i, \sigma^2_{history}) \end{aligned} \tag{7}$$

Once the noise is added, we calculate the base velocity for the CVM by taking the weighted average of the location displacements of the historical data. We use exponentially decaying weights for the weighted averaging, which causes the latest velocities to have greater weights, thus contributing more to the result of the calculation.

$$\overline{\Delta} = \frac{\sum_{i=1}^{n-1} w_i \Delta_i}{\sum_{i=1}^{n-1} w_i}, \tag{8}$$

where

$$\Delta_i = l'_{i+1} - l'_i = (x'_{i+1} - x'_i, y'_{i+1} - y'_i) \tag{9}$$

and

$$w_1 = w_b^{n-2}, w_2 = w_b^{n-3}, ..., w_{n-2} = w_b^1, w_{n-1} = w_b^0 \tag{10}$$

and

$$0 < w_b \leq 1 \tag{11}$$

Notice that there are $n-1$ velocities it total since the number of historical locations is $n$.

The base weight $w_b$ used in the weighted averaging is sampled uniformly randomly for the generation of each future trajectory. Our model accepts a parameter for the lower bound $w_{low}$ of the weight, which is used to sample $w_b$ uniformly randomly from the range $[w_{low}, 1]$. When the lower bound is set to 1, then the weighted average becomes a regular average, where all elements contribute to the result equally. As the weight gets closer to 0, the more the latest timesteps contribute to the calculation of the average, making the most recent timesteps more important. Since the base weight is sampled separately for the generation of each trajectory, it causes some of the generated trajectories to make more use of the earliest timesteps in the historical data than others. This further introduces additional variety to the resulting distribution.

To use the CVM with constant rotation, a base angle $\overline{\alpha}$ has to be calculated to rotate all future points. This is done in a similar way as the constant velocity calculation, by taking the weighted average of all angle changes in the historical data:

$$\overline{\alpha} = \frac{\sum_{i=1}^{n-2} w_i \alpha_i}{\sum_{i=1}^{n-2} w_i} \tag{12}$$

where

$$w_1 = w_b^{n-3}, w_2 = w_b^{n-4}, ..., w_{n-3} = w_b^1, w_{n-2} = w_b^0 \tag{13}$$

Each angle $\alpha_i$ is calculated between two subsequent velocities, thus there are $n-2$ angles $\alpha_i$ in total since the number of velocities is $n-1$. The angle $\alpha_i$ is calculated by using the 2-argument arctangent, a function for the calculation of the angle between two vectors available in various programming languages, including Python [Zhang et al., 2021]. The

determinant and dot product of two subsequent velocities are passed to the function as arguments:

$$\alpha_i = atan2(\det(\Delta_i, \Delta_{i+1}), \Delta_i \cdot \Delta_{i+1}), \tag{14}$$

### 3.1.3 Inducing random events

During the generation of each future trajectory, some random events are applied at the creation of each new timestep. These random events are changing the velocity of the CVM, changing the rotation angle of the CVM with constant rotation, and setting the velocity to zero to imitate sudden stops. The goal of the events is to introduce additional variety for each sampled trajectory. They simulate possible changes in walking pace and direction a person might make in the future.

The random event for changing the base velocity of the CVM is meant to simulate a person speeding up or slowing down at some point in the future. The event is specified by two parameters for our method. The first parameter is the probability for the event to happen, denoted by $p_{velocity\_change}$, which takes a value from the range $[0.0, 1.0]$. At the algorithm's runtime, this probability is applied at the generation of each future timestep to decide whether to change the base velocity of the CVM. When the event is applied, the base velocity $\overline{\Delta} = (\delta_x, \delta_y)$ is updated by adding a specified amount of noise to its $x$ and $y$ directions. This is done by sampling new x and y values $\delta'_x$ and $\delta'_y$ from a Gaussian distribution with a mean of the previous value and a standard deviation of $\sigma_{velocity}$, which is specified as the second parameter for this event.

$$\overline{\Delta}' = (\delta'_x, \delta'_y) \tag{15}$$

where

$$\begin{aligned} \delta'_x &\sim N(\delta_x, \sigma^2_{velocity}) \\ \delta'_y &\sim N(\delta_y, \sigma^2_{velocity}) \end{aligned} \tag{16}$$

The event for changing the angle of the CVM with constant rotation is similarly specified by two parameters - the probability for the event to happen, denoted by $p_{angle\_change}$ and a value for the standard deviation used for sampling noise to add to the existing angle, denoted by $\sigma_{angle}$. If the event is applied, then noise is added to the existing angle. This is done by sampling a new value $\overline{\alpha'}$ from a Gaussian distribution with a mean of the

previous value and a standard deviation of $\sigma_{angle}$.

$$\overline{\alpha}' \sim N(\overline{\alpha}, \sigma^2_{angle}) \tag{17}$$

The event for randomly stopping is specified by a single parameter - the probability for the event happening which takes a value from the range $[0.0, 1.0]$. When the event is triggered, then the next future location which is generated will have the same exact value as the previous one, to signify that no movement happened.

### 3.1.4 Overview of parameters and pseudocode

Table 1 shows an overview of all of the parameters of our method's generative step which were described in the previous sections. A shorthand value is given for each parameter and their definitions are summarised. The shorthands are referenced in some of the following chapters when discussing the use of the parameters to avoid repetition.

Table 1. Overview of the parameters of our method's generative step

| **Parameter name** | **Definition** |
|---|---|
| $N_{samples}$ | The number of future trajectory samples generated for one instance of historical data |
| $\sigma_{history}$ | Standard deviation for the amount of noise sampled from a Gaussian distribution which is added to every timestep of the current instance of historical data |
| $w_{low}$ | The lower bound used for uniformly randomly sampling a base weight for weighted average calculation of the constant velocity and angle |
| $p_{rotation}$ | The probability of using the constant velocity model with constant rotation when generating a sample |
| $p_{velocity\_change}$ | The probability of applying the velocity change event at the generation of any future timestep |
| $p_{angle\_change}$ | The probability of applying the angle change event at the generation of any future timestep |
| $\sigma_{velocity}$ | Standard deviation for the amount of noise sampled from a Gaussian distribution which is added to the x and y directions of the velocity $\overline{\Delta}$ when the velocity change event is triggered |
| $\sigma_{angle}$ | Standard deviation for the amount of noise sampled from a Gaussian distribution which is added to the angle $\overline{\alpha}$ when the angle change event is triggered |
| $p_{stop}$ | The probability of applying the movement stopping event at the generation of any future timestep |

Algorithm 1 contains the pseudocode for the future trajectory generation process of our method to provide a general overview of how the aforementioned steps of the trajectory sampling algorithm come together. The algorithm depicts the future trajectory generation for one instance of observed historical data $L_{hist}$. The function $uniform\_random(low, high)$ signifies uniformly random sampling of a floating point value from the range $[low, high]$. The values attained from it are compared to the probability parameters of the random events to decide whether to apply them. Some of the steps of the algorithm are abstracted as they were discussed in mathematical details previously. This includes adding noise to the historical data denoted by $add\_noise()$, the initial calculation of the base velocity $\overline{\Delta}$ denoted by $calculate\_average\_velocity()$, and the calculation of the base angle $\overline{\alpha}$ denoted by $calculate\_average\_angle()$.

**Algorithm 1:** Our method's future trajectory generation process

**Input:** One person's historical trajectory: $L_{hist}$
Parameters for the sampling: $N_{samples}$, $w_{low}$, $p_{rotation}$, $p_{velocity\_change}$, $p_{angle\_change}$, $\sigma_{history}$, $\sigma_{velocity}$, $\sigma_{angle}$, $p_{stop}$

**Output:** Array of possible future trajectories: $samples$

samples = [];

**for** $n := 1$ *to* $N_{samples}$ **do**

/* Setting the initial state */

$L'_{hist} = add\_noise(L_{hist}, \sigma_{history})$;

last_datapoint = $L'_{hist}.take\_last\_element()$ ;

$\overline{\Delta} = calculate\_average\_velocity(L'_{hist}, w_{low})$;

$\overline{\alpha} = calculate\_average\_angle(L'_{hist}, w_{low})$;

current_sample = [];

is_rotation_enabled = $uniform\_random(0.0, 1.0) < p_{rotation}$;

**for** $m := 1$ *to* $N_{sample\_length}$ **do**

/* Applying random events to modify the state */

**if** *uniform_random(0.0, 1.0)* < $p_{velocity\_change}$ **then**

$\overline{\Delta} = (\delta_x + N(0, \sigma^2_{velocity}), \delta_y + N(0, \sigma^2_{velocity}))$;

**else if** *uniform_random(0.0, 1.0)* < $p_{angle\_change}$ **then**

$\overline{\alpha} = \overline{\alpha} + N(0, \sigma^2_{angle})$;

**else if** *uniform_random(0.0, 1.0)* < $p_{stop}$ **then**

stop = $True$;

/* Creating the new timestep */

**if** *stop == True* **then**

new_datapoint = last_datapoint;

stop = $False$;

**else**

new_datapoint = last_datapoint + $\overline{\Delta}$;

**if** *is_rotation_enabled == True* **then**

new_datapoint = $rotate$(new_datapoint, $\overline{\alpha}$);

current_sample.insert(new_datapoint);

last_datapoint = new_datapoint;

samples.insert(current_sample);

## 3.2 Creating representative predictions

The second part of our method is making use of the large number of future trajectories created for a person by the generative step. The goal of this step is to choose a limited number of representative predictions to produce as the method's output. The process of selecting these predictions is explained step by step in the next sections.

### 3.2.1 Kernel density estimation on future trajectories

When the possible future trajectories for a person are sampled, kernel density estimation (KDE) is run on the final points of all generated trajectories. Scott's Rule is used as the KDE bandwidth. KDE estimates the probability density function of the generation of the final trajectory points. This gives an insight into which future paths are more likely to happen according to our generation logic. The estimation results are used to give each generated trajectory a probability based on which density region the final point of the trajectory is in.

An example KDE distribution of the final points of generated trajectories can be seen in Figure 4. The figure displays the distribution of the 1000 trajectories shown previously in Figure 2.

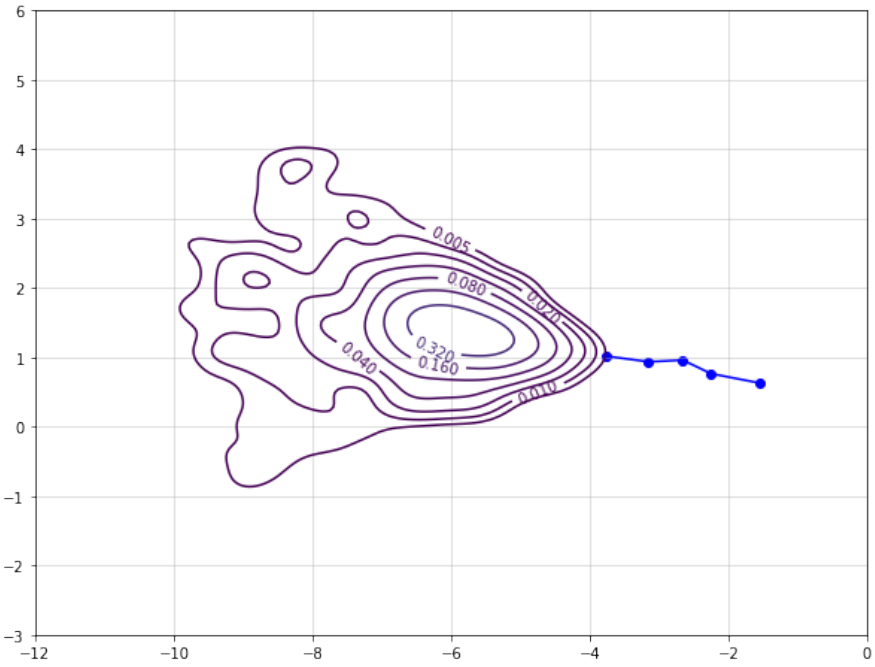


Figure 4. Example of the result of running kernel density estimation on the final points of 1000 sampled future trajectories.

### 3.2.2 Separating sampled trajectories into groups by KDE

Once the kernel density estimation is run, the sampled trajectories are ordered decreasingly by the density estimations of their final points. Then the sorted trajectories are separated into multiple groups of varying sizes. The goal of this is to create representative sections of the sampled data, with each group containing possible future trajectories of some certain density range. The first group will contain the densest or most likely trajectories, the second will contain less likely ones and so on. This will allow us to create future trajectory predictions for each group separately in the next step.

The number of groups and their sizes are specified as parameters for our method. A list of quantiles $\mathbf{q} = [q_1, ..., q_m]$ is provided as an argument where $0\% < q_1 < q_2 < ... < q_m \leq 100\%$ . The range between any two subsequent quantiles creates a group of trajectories containing the corresponding percentage range of the trajectories sorted by KDE. For example, if two adjacent quantiles were 20% and 40%, then the group corresponding to these values will contain the top 20% to top 40% of the sorted

trajectories.

An example of the described partitioning is showcased in Figures 5-8. The example builds upon the 1000 sampled trajectories shown in Figure 2. The generated trajectories are sampled into four groups, with the first group containing the top $0\% - 20\%$ of trajectories by KDE, the second containing the top $20\% - 48\%$, the third containing the top $48\% - 65\%$ and the fourth group containing the final $65\% - 100\%$. The argument $\mathbf{q}$ in this example receives a value of $[20\%, 48\%, 65\%, 100\%]$. Each figure showcases all of the trajectories belonging to one of the groups.

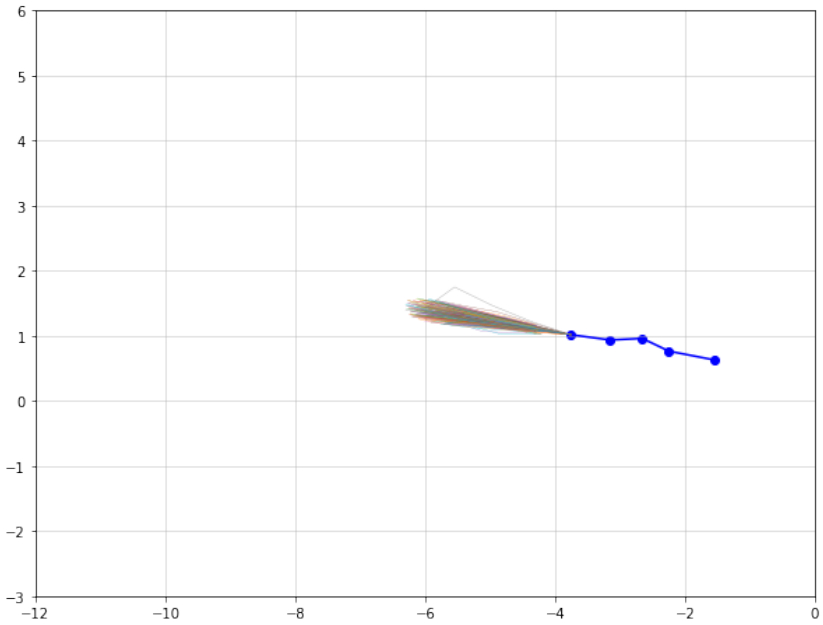


Figure 5. Sampled trajectories from the 0% to 20% quantile

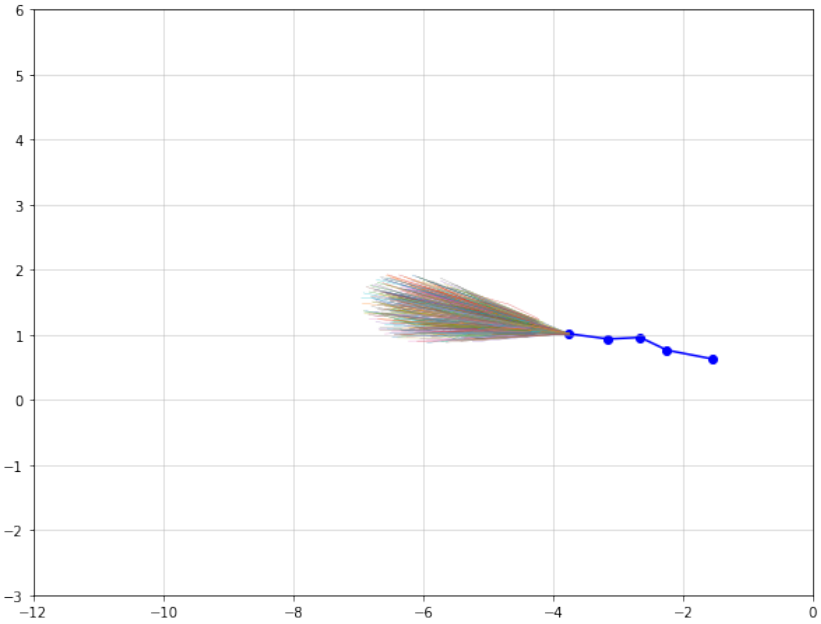


Figure 6. Sampled trajectories from the 20% to 48% quantile

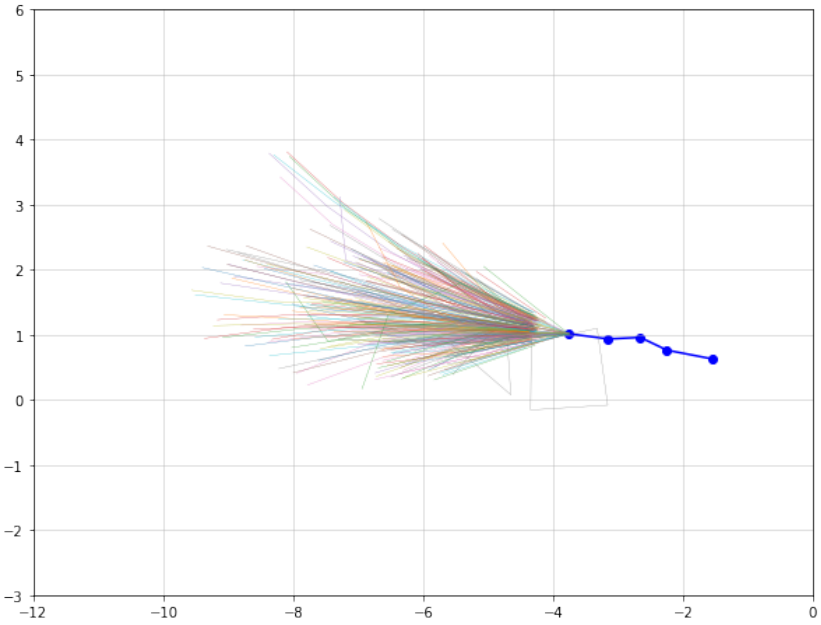


Figure 7. Sampled trajectories from the 48% to 65% quantile

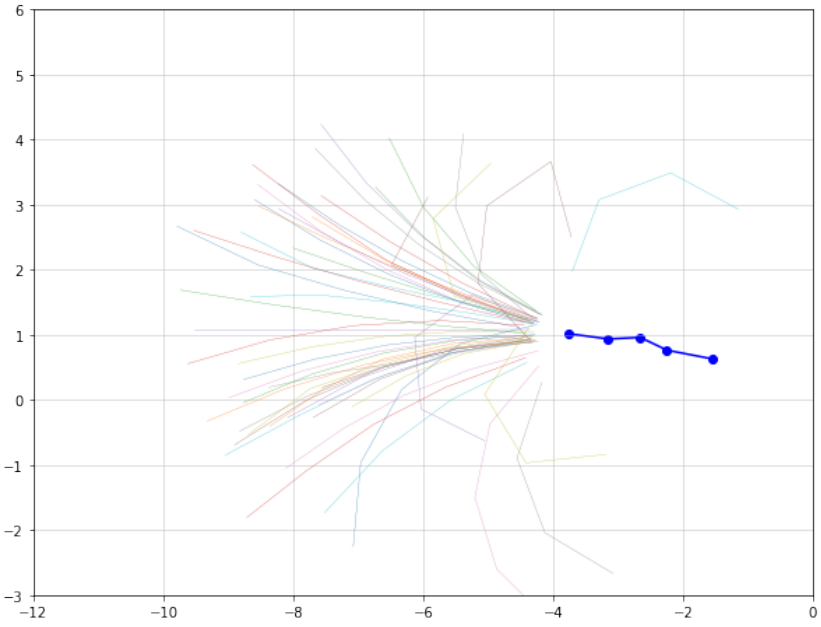


Figure 8. Sampled trajectories from the 65% to 100% quantile

### 3.2.3 K-means clustering for choosing representative predictions

As the final step of our algorithm, representative predictions are chosen from the previously created groups of samples. This is done by running K-means clustering on the final points of the generated samples on each sample group separately. The K-means clustering algorithm groups data into $k$ clusters where each cluster contains similar entities by minimizing squared euclidean distances within the cluster [Na et al., 2010].

The parameter $k$ for specifying the number of clusters for K-means is passed as an argument to our method for each group: $\mathbf{k} = [k_1, ..., k_n]$ where $k_i$ marks the number of desired clusters for the $i$-th group.

By running K-means clustering on the sampled trajectories in each group, we further divide each group into multiple partitions. As the last step of the algorithm, each cluster in each group is used to attain one representative prediction. This is done by taking the pointwise mean of all trajectories belonging to a cluster.

Each prediction also receives a probability $p_i$ which is calculated by finding the proportion of the samples included in the corresponding cluster over all samples. The

idea behind including the probabilities is that a robot's planner can use them for risk assessment for collision avoidance purposes [Kim and Kum, 2018]. Future trajectories with very small probabilities can potentially be ignored as they are unlikely to happen and ones with high probabilities can receive a higher priority.

In total, the number of predictions created by our method is the sum of the number of clusters generated for each group:

$$n_{preds} = \sum_{i=1}^{n} k_i \tag{18}$$

The result of this step is seen in Figure 9. The blue dotted line signifies the historical path of a person and the colored lines are the representative predictions chosen by our method. This example builds upon the previous example shown in Figures 5-8, where the sampled trajectories are divided into four groups. We choose one representative prediction from the first group and five from every other group. This means that the aforementioned K-means parameter $\mathbf{k}$ has a value of $[1, 5, 5, 5]$. The first group's prediction is denoted by red, the second group's predictions by green, the third group's by orange, and the fourth group's by purple. Every representative prediction also has its probability annotated at its endpoint. It can be seen that the representative trajectories from the groups with a higher probability range (i.e. groups 1 and 2) are closer together and focused on the center. The groups with the lower probability ranges (groups 3 and 4) produce trajectories which are more fanned out.

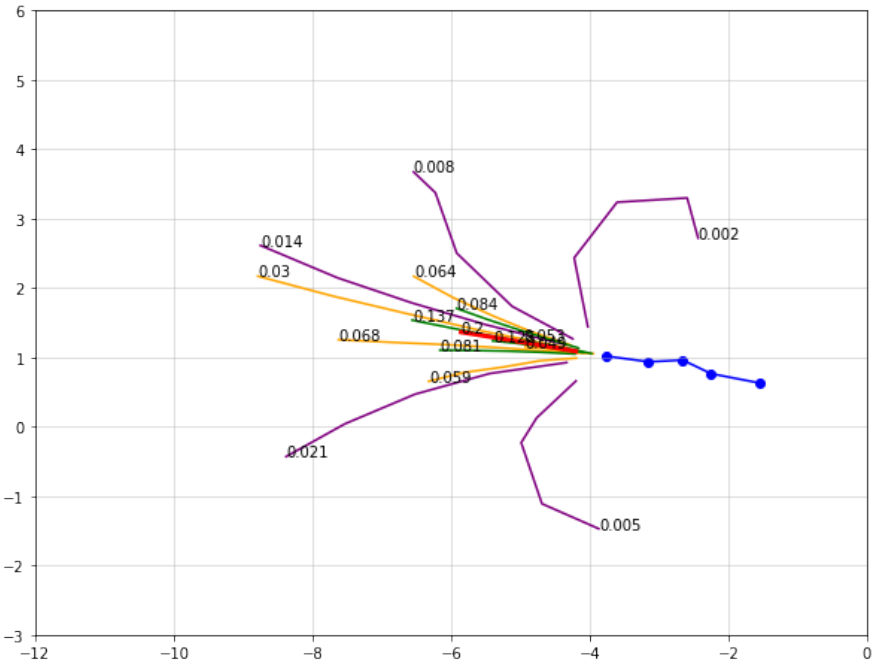


Figure 9. Example of the result of running K-means clustering on the representative groups of the sampled trajectories.

# 4 Experimental setup

This section describes the datasets, evaluation measures, and the baseline comparisons which were used to evaluate our generative method.

## 4.1 Datasets

We evaluated our method and compared it to other trajectory forecasting methods on the ETH [Ess et al., 2007] and UCY [Lerner et al., 2007] datasets which are commonly used for comparison in other works on the trajectory prediction topic [Salzmann et al., 2020] [Gupta et al., 2018]. The datasets contain 5 different sets of real world pedestrian data consisting of 4 unique scenes. The scenes are filmed from a top or close to top view and the data is annotated at 2.5 frames per second.

We use the commonly used approach for evaluating the methods on these datasets [Salzmann et al., 2020] [Gupta et al., 2018]. We observe a trajectory for 8 timesteps (3.2 seconds) and predict the following 12 timesteps (4.8 seconds). The observations are made in a rolling manner, i.e. if a pedestrian's path is longer than 20 timesteps then all sections of the trajectory with the length of 8 + 12 timesteps are used for evaluation.

Any methods which require training are trained with a leave-one-dataset-out cross-validation approach. This means that a method is trained on four sets of data and evaluated on the held out fifth, ensuring that test data remains unseen before evaluation.

## 4.2 Evaluation measures

As is common in prior work [Gupta et al., 2018] [Salzmann et al., 2020], our method is evaluated with two error metrics - final displacement error and average displacement error.

Final displacement error is the Euclidean distance between the final point of the predicted trajectory and the final point of the ground truth trajectory. This measures how well the end-target of the ground truth trajectory is predicted.

$$FDE = \sqrt{(x - \hat{x})^2 + (y - \hat{y})^2} \tag{19}$$

Average displacement error is the average Euclidean distance between all points of the predicted trajectory and the ground truth trajectory. This measures how well the

predicted trajectory follows the ground truth throughout all timesteps.

$$ADE = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2} \tag{20}$$

We consider two separate cases of evaluation in our experiments, evaluation on a single prediction and evaluation on the best of 20 predictions. In the case of a single prediction, the FDE and ADE metrics are used directly on the prediction by comparing it against the ground truth trajectory. In the case of best of 20 predictions, we calculate the metric values for each trajectory, and choose the one with the lowest error for evaluation. This is the common practice for multiple trajectory prediction in existing works [Gupta et al., 2018] [Salzmann et al., 2020].

## 4.3 Comparison baselines

We use several other trajectory prediction methods as baseline evaluation comparisons to our method. We use the constant velocity model in two variations and two state of the art generative methods designed for multi-modal predictions, Social GAN and Trajectron++.

The constant velocity model is often included in trajectory forecasting works as a comparison baseline due to its simplicity. A recent study has also shown that CVM is a reasonable prediction method by itself, it achieved as good or better results than several neural network based approaches and state of the art methods when evaluated on unseen datasets [Schöller et al., 2020]. Since the generative step of our method is based on this, it is also useful to see if and by how much the addition of random events and rotations have improved its performance. We include two variations of the method, one which calculates the constant velocity from the two most recent historical timesteps and one which uses the entire historical data for the calculation. To make use of CVM in experiments with multi-modal predictions, we run the method several times whilst adding noise to each historical timestep during each run to introduce variability in predictions. The noise is sampled uniformly randomly from a Gaussian distribution with a mean of 0 and a standard deviation of 1.

Trajectron++ [Salzmann et al., 2020] is a state of the art LSTM-based generative trajectory forecasting model which has been one of the best performing methods in recent works. The method contains some complex ideas to improve its performance.

It can model the dynamic interactions between multiple agents in the scene and it can make use of heterogeneous data such as a map of the scene to have more context for predictions. It is also designed to produce multi-modal output. To keep comparisons fair we use Trajectron++ without its dynamic interactions integration since at its current state our method doesn't account for interactions between people. It should be noted though that including the dynamic interactions integration had a marginal effect on the model's performance in terms of FDE and ADE. The model was trained and evaluated as per the authors' code and instructions available in their public repository in GitHub[2].

We also include Social GAN [Gupta et al., 2018] in the comparisons. It is a highly cited method created in 2018 which is often included as a baseline in recent works. It is a generative method which aims to predict socially acceptable future trajectories. This means that it incorporates social norms into its predictions, like modeling the respect for other people's personal space. The method is also optimized to produce multiple predictions per each person.

## 4.4 Our method's parameters

We tuned the parameters for our method separately for each set of data in the ETH and UCY datasets. A cross validation approach was used as has been the common approach in other works. This means that when the method was being tuned for one set of data, it was validated on the other four sets so as to not train directly on the dataset it was going to be evaluated on. A grid search was run for each parameter to attain the optimal values.

Table 2 shows the parameter values for our method for the scenes in the ETH dataset and Table 3 shows the parameter values for the scenes in the UCY dataset. It can be seen that the resulting parameter values are roughly similar over all datasets, with minor differences in some of the event probabilities and the grouping and clustering parameters.

[2]https://github.com/StanfordASL/Trajectron-plus-plus

Table 2. Our method's parameters for the scenes in the ETH dataset. The values for each scene were tuned only on other scenes.

| **Parameter** | **ETH** | **Hotel** |
|---|---|---|
| $N_{samples}$ | 500 | 500 |
| $\sigma_{history}$ | 0.05 | 0.05 |
| $p_{rotation}$ | 0.5 | 0.5 |
| $p_{velocity\_change}$ | 0.1 | 0.1 |
| $\sigma_{velocity}$ | 0.1 | 0.1 |
| $p_{angle\_change}$ | 0.2 | 0.2 |
| $\sigma_{angle}$ | 2 | 2 |
| $p_{stop}$ | 0.025 | 0.0 |
| $w_{low}$ | 0.15 | 0.1 |
| Group quantiles $\mathbf{q}$ | [0.05, 0.5, 0.75, 1.0] | [0.05, 0.60, 1.0] |
| Clusters per group $\mathbf{k}$ | [1, 9, 6, 4] | [1, 11, 8] |

Table 3. Our method's parameters for the scenes in the UCY dataset. The values for each scene were tuned only on other scenes.

| **Parameter** | **Zara 1** | **Zara 2** | **University** |
|---|---|---|---|
| $N_{samples}$ | 500 | 500 | 500 |
| $\sigma_{history}$ | 0.05 | 0.1 | 0.025 |
| $p_{rotation}$ | 0.5 | 0.5 | 0.5 |
| $p_{velocity\_change}$ | 0.1 | 0.1 | 0.1 |
| $\sigma_{velocity}$ | 0.1 | 0.1 | 0.1 |
| $p_{angle\_change}$ | 0.1 | 0.2 | 0.2 |
| $\sigma_{angle}$ | 2 | 2 | 2 |
| $p_{stop}$ | 0.0 | 0.025 | 0.05 |
| $w_{low}$ | 0.15 | 0.1 | 0.1 |
| Group quantiles $\mathbf{q}$ | [0.05, 0.60, 1.0] | [0.05, 0.60, 1.0] | [0.05, 0.5, 0.75, 1.0] |
| Clusters per group $\mathbf{k}$ | [1, 14, 5] | [1, 14, 5] | [1, 9, 6, 4] |

# 5 Experiment results

This section describes the experiments which were conducted for our method and provides an analysis of the results. We ran three different experiments, of which two are comparisons of our method against other trajectory forecasting methods. We compare our method against others in the task of predicting one future trajectory for every person and in the task of predicting twenty future trajectories. We also perform an ablation study of our method to evaluate the effect of the random events at the generative step of the algorithm and the effect of the grouping and K-means clustering on the generated samples.

## 5.1 Single trajectory prediction

Let's first consider the task of producing a single trajectory prediction for each person. We evaluate our method and the comparison baselines on the ETH and UCY datasets.

Since our method is conditioned to produce multiple trajectory predictions with varying probabilities, we have to choose a single prediction to keep the evaluation fair. To do this we choose the prediction with the highest probability to be used for evaluation.

Table 4. FDE on single predictions

| Dataset | CVM (2pt) | CVM (8pt) | S-Gan | Trajectron++ | Ours |
|---|---|---|---|---|---|
| ETH | 2.303 | 2.282 | 2.210 | **1.615** | 2.019 |
| Hotel | 0.462 | 0.614 | 2.180 | 0.499 | **0.401** |
| Univ | 1.576 | 1.369 | 1.280 | 1.205 | **1.196** |
| Zara 1 | 1.132 | 0.952 | 0.910 | **0.770** | 1.016 |
| Zara 2 | 0.860 | 0.724 | 1.110 | **0.589** | 0.780 |
| Average | 1.267 | 1.188 | 1.538 | **0.936** | 1.082 |

The mean final displacement error per dataset on single trajectory predictions can be seen in Table 4. On average, Trajectron++ is the best performer with a mean FDE of 0.936. It produced the best results in 3 out of 5 datasets. Our method comes second

with a mean FDE of 1.082 whilst having the best results on 2 sets of data. The constant velocity model takes the next 2 spots with the variant using the full 8 points of history having a mean FDE of 1.188 and the variant using 2 latest history points having a mean FDE of 1.267. Social GAN is further off with a mean result of 1.538.

Table 5. ADE on single predictions

| Dataset | CVM (2pt) | CVM (8pt) | S-Gan | Trajectron++ | Ours |
|---|---|---|---|---|---|
| ETH | 1.102 | 1.075 | 1.130 | **0.695** | 0.953 |
| Hotel | 0.243 | 0.319 | 1.010 | 0.224 | **0.213** |
| Univ | 0.781 | 0.618 | 0.600 | **0.468** | 0.570 |
| Zara 1 | 0.552 | 0.427 | 0.420 | **0.297** | 0.483 |
| Zara 2 | 0.421 | 0.324 | 0.520 | **0.227** | 0.381 |
| Average | 0.620 | 0.553 | 0.736 | **0.382** | 0.520 |

Table 5 showcases the methods' performance in terms of average displacement error. Trajectron++ is the favorite in this case as well, outperforming other methods on four datasets with a mean ADE of 0.382. Our method performs best on one dataset and second best on two sets of data with a mean ADE of 0.520. The constant velocity model variations are 3rd and 4th best again with mean ADE-s of 0.553 and 0.620.

In this experiment, our method performed noticeably worse than Trajectron++. This is somewhat expected as choosing a single prediction from our method's multi-modal output loses some of the value created by the algorithm, the results of grouping and K-means clustering of the sampled trajectories are neglected.

## 5.2 Multiple trajectory predictions

Let us now consider the task of predicting multiple trajectories for each person. This fits the use case of our method well. It allows for evaluating a major part of the algorithm which was unused in the previous experiment, choosing many representative predictions out of the generated samples. This task is evaluated on the ETH and UCY datasets as well. In this experiment, each method predicts 20 possible future trajectories for

every pedestrian, of which the one with the lowest error is compared to the ground truth trajectory for evaluation.

Table 6. FDE on the best of 20 predictions

| Dataset | CVM (2pt) | CVM (8pt) | S-Gan | Trajectron++ | Ours |
|---|---|---|---|---|---|
| ETH | 1.108 | 1.758 | 1.520 | 0.812 | **0.642** |
| Hotel | 0.363 | 0.590 | 1.610 | 0.197 | **0.180** |
| Univ | 0.971 | 1.220 | 0.690 | 0.450 | **0.448** |
| Zara 1 | 0.793 | 0.884 | 1.260 | **0.342** | 0.416 |
| Zara 2 | 0.555 | 0.656 | 0.840 | **0.253** | 0.309 |
| Average | 0.758 | 1.022 | 1.184 | 0.411 | **0.399** |

Table 6 shows the average FDE results of each method for each set of data. Here our method outperforms others on three out of five datasets, with Trajectron++ being its closest rival. Our method has the greatest advantage on the ETH dataset, a slight advantage on the Hotel dataset, and a marginal lead on the University dataset. Trajectron++ performs better on the Zara 1 and Zara 2 datasets and is second best on the others. On average, our method still retains a slight lead over all datasets with a mean FDE of 0.399 against Trajectron's 0.411. The rest of the methods have noticeably larger errors. The constant velocity model achieves an average FDE of 0.758 when using the latest 2 timesteps as its input and an average FDE of 1.022 when using all 8 historical points. Social GAN is the worst performer with an average FDE of 1.184.

Table 7. ADE on the best of 20 predictions

| Dataset | CVM (2pt) | CVM (8pt) | S-Gan | Trajectron++ | Ours |
|---|---|---|---|---|---|
| ETH | 0.641 | 0.899 | 0.810 | **0.396** | 0.440 |
| Hotel | 0.205 | 0.308 | 0.720 | **0.115** | 0.125 |
| Univ | 0.527 | 0.572 | 0.600 | **0.205** | 0.264 |
| Zara 1 | 0.416 | 0.409 | 0.340 | **0.155** | 0.248 |
| Zara 2 | 0.294 | 0.304 | 0.420 | **0.115** | 0.182 |
| Average | 0.417 | 0.498 | 0.578 | **0.197** | 0.252 |

Table 7 shows the methods' results in terms of average displacement error. Trajectron++ performs the best on each dataset with a mean ADE of 0.197. Our method is the second best on every dataset with a mean ADE of 0.252. Other methods have greater errors with CVM having a mean ADE of 0.417 when using two most recent timesteps as its input and 0.498 when using all eight timesteps. Social GAN produces a mean ADE of 0.578.

The results of this experiment show that our method can predict the final destination of the predicted trajectory well, as demonstrated by producing the best results in terms of final displacement error. Despite this, the results in terms of average displacement error show that it does not predict the actual path of the trajectory as well as Trajectron++.

## 5.3 Ablation study

We perform an ablation study of our method. We evaluate the method on the ETH and UCY datasets with various steps of the algorithm turned off to judge their effect on performance.

We validate the generative step of the algorithm by turning off the velocity and angle change events, by turning only one of them on, and by turning both of them on. Toggling these events is done by manipulating the probabilities for the velocity and angle change events, parameters $p_{velocity\_change}$ and $p_{angle\_change}$ are set to 0 to disable them.

We also validate the grouping and clustering step by running the aforementioned variations both with the regular setup and with disabling the grouping of the generated

samples. This is done by manipulating the parameters for the grouping and the clustering of the samples. To disable grouping, $\mathbf{q}$ receives a single value of $[100\%]$ and $\mathbf{k}$ receives a single value of $[20]$. These values indicate that samples are left in a single group and K-means clustering with $k = 20$ is run directly on the single group.

The experiment is run with the best of 20 method as used in the multiple trajectory prediction experiment since it showcases the desired use case of our method the best. We analyze the performance in terms of final displacement error and average displacement error separately in the following subsections.

### 5.3.1 Final displacement error

Table 8. Our method FDE in various configurations with sample grouping disabled.

| Dataset | No events | Velocity event | Angle event | Angle & velocity event |
|---|---|---|---|---|
| ETH | 1.469 | 0.968 | **0.818** | 0.836 |
| Hotel | 0.271 | 0.401 | **0.248** | 0.378 |
| Univ | 0.750 | 0.744 | **0.538** | 0.642 |
| Zara 1 | 0.633 | 0.738 | **0.582** | 0.685 |
| Zara 2 | 0.516 | 0.544 | **0.415** | 0.489 |
| Average | 0.728 | 0.679 | **0.520** | 0.606 |

Table 8 shows our method's performance in terms of final displacement error on all datasets with various random event configurations with the grouping of samples disabled. The variation with no events performs the worst on average, with a mean FDE of 0.728. The variation with only the angle change event enabled performs the best on average, with a mean FDE of 0.520. The results of having both events enabled show that in this case including the velocity change event was actually detrimental to performance, by producing a mean FDE of 0.606.

Table 9. Our method FDE in various configurations with sample grouping enabled.

| Dataset | No events | Velocity event | Angle event | Angle & velocity event |
|---|---|---|---|---|
| ETH | 1.451 | 0.921 | 0.759 | **0.656** |
| Hotel | 0.222 | 0.293 | **0.172** | 0.258 |
| Univ | 0.719 | 0.598 | 0.531 | **0.502** |
| Zara 1 | 0.556 | 0.606 | **0.465** | 0.544 |
| Zara 2 | 0.475 | 0.419 | 0.364 | **0.363** |
| Average | 0.685 | 0.567 | **0.458** | 0.465 |

Table 9 shows our method's performance in terms of final displacement error on all datasets with various random event configurations with the grouping of samples enabled. The results showcase that using the sample grouping and K-means clustering technique is justified, every event configuration with grouping enabled noticeably outperforms its counterpart with the grouping disabled. For example, enabling both random events without grouping achieved a mean FDE of 0.606, but with grouping achieved a mean FDE of 0.465.

Similarly to the results without sample grouping, the variation with the random generative events disabled performs the worst on average, with a mean FDE of 0.685. The variation with only the angle change event enabled and the variation with both events enabled show the best performance with nearly identical results, with mean FDE-s of 0.458 and 0.465 respectively. In three out of five datasets the latter configuration actually performs better, but the loss on the other two sets is great enough to drive its mean FDE down. An interesting insight is that with grouping and clustering enabled, the velocity event does not degrade the method's performance as in the previous results. Altogether the results show that the random events are justified additions to our method as well, as they boost its performance. The results also demonstrate that each dataset responds differently to certain events, for example using only the angle event works noticeably better on the Hotel dataset than using both events, but the opposite is true for the ETH dataset.

### 5.3.2 Average displacement error

Table 10. Our method ADE in various configurations with sample grouping disabled.

| Dataset | No events | Velocity event | Angle event | Angle & velocity event |
|---|---|---|---|---|
| ETH | 0.730 | 0.542 | 0.528 | **0.498** |
| Hotel | **0.153** | 0.197 | 0.154 | 0.205 |
| Univ | 0.379 | 0.366 | **0.315** | 0.343 |
| Zara 1 | **0.310** | 0.351 | 0.322 | 0.369 |
| Zara 2 | 0.254 | 0.261 | **0.236** | 0.260 |
| Average | 0.365 | 0.343 | **0.311** | 0.335 |

Table 10 showcases the ADE results of our method in various configurations with the sample grouping and clustering disabled. Unlike in terms of FDE, the random events without grouping and clustering do not have as noticeable of an effect on the performance. Without any events, the mean ADE is 0.365, with the velocity event it is 0.343, with the angle event 0.311, and with both events 0.335.

Table 11. Our method ADE in various configurations with sample grouping enabled.

| Dataset | No events | Velocity event | Angle event | Angle & velocity event |
|---|---|---|---|---|
| ETH | 0.708 | 0.525 | 0.501 | **0.440** |
| Hotel | 0.131 | 0.164 | **0.119** | 0.155 |
| Univ | 0.358 | 0.315 | 0.302 | **0.294** |
| Zara 1 | 0.274 | 0.299 | **0.264** | 0.314 |
| Zara 2 | 0.234 | 0.219 | **0.206** | 0.214 |
| Average | 0.341 | 0.304 | **0.278** | 0.283 |

Table 10 displays the ADE results in various event configurations with the sample grouping and clustering enabled. Firstly, as was the case with FDE, it is evident that the addition of grouping and clustering increases the performance of the method. Every configuration outperforms its counterpart with the grouping and clustering disabled.

Also, similarly to the FDE results, the configurations with only the angle change event enabled and with both events enabled perform the best with similar results, with mean ADE of 0.278 and 0.283 respectively. The former is better on three out of five sets of data and the latter on the rest of the two. This further shows that the parameters of our method can be tuned to produce good results in different environments, e.g. using both events works well for ETH and using only the angle event is fitting for Zara 1.

# 6 Conclusion

We proposed a new physics-based generative trajectory forecasting method. The method produces multi-modal predictions based solely on the historical data of the observed person, i.e. it does not need to be trained on previous data. Each step of our algorithm is interpretable and its workflow is trackable, as opposed to a neural network based approach which many state of the art methods take. The method consists of two main parts, generating many potential future trajectories and choosing the representative predictions from them. Both of the steps are configurable by parameters and easily extendable to adapt the method to various environments.

Comparisons against other methods show that our method performs nearly as good as state-of-the-art models and in some cases even better. Our method produces good predictions in terms of the final endpoint of the trajectory, but lacks the accuracy in terms of predicting the whole path when compared against the best performing model in recent works. We demonstrated with the experiments that it is possible to create a relatively well-performing forecasting method whilst keeping the inner workings of its algorithm easy to follow. We also performed an ablation study on our method which proved the added benefit of the various generative steps of our algorithm and the process of choosing representative predictions.

Future work entails integrating our proposed method with a robot being developed by the robotics research group at the University of Tartu, as that was the end goal of the development of the algorithm. Further optimizing can be done to the method as well. With some adjustments, it could make use of an annotated map of a scene when making predictions which could be used to condition the predictions on some known goal locations and obstacles of the environment. Knowledge of goal locations, e.g. a door to a building where a person is likely to head, could allow the method to prioritize the generation of such trajectories which lead to the goal. Likewise, information about non-passable objects, such as a wall or a chair, could be used to reject the generation of such trajectories which try to pass them.

# Acknowledgement

The work was supported in part by the Artificial Intelligence (AI) & Robotics Estonia (AIRE), the Estonian candidate for European Digital Innovation Hub, funded by the Ministry of Economic Affairs and Communications in Estonia.

# References

[Choset et al., 2005] Choset, H., Lynch, K., Hutchinson, S., Kantor, G., and Burgard, W. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Intelligent Robotics and Autonomous Agents series. MIT Press.

[Ess et al., 2007] Ess, A., Leibe, B., and Van Gool, L. (2007). Depth and appearance for mobile scene analysis. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8.

[Gasparetto et al., 2012] Gasparetto, A., Boscariol, P., Lanzutti, A., and Vidoni, R. (2012). Trajectory planning in robotics. *Mathematics in Computer Science*, 6(3):269–279.

[Gupta et al., 2018] Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., and Alahi, A. (2018). Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2255–2264.

[Kim and Kum, 2018] Kim, J. and Kum, D. (2018). Collision risk assessment algorithm via lane-based probabilistic motion prediction of surrounding vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 19(9):2965–2976.

[Kosaraju et al., 2019] Kosaraju, V., Sadeghian, A., Martín-Martín, R., Reid, I., Rezatofighi, H., and Savarese, S. (2019). Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. *Advances in Neural Information Processing Systems*, 32.

[Lerner et al., 2007] Lerner, A., Chrysanthou, Y., and Lischinski, D. (2007). Crowds by example. *Computer Graphics Forum*, 26(3):655–664.

[Mangalam et al., 2021] Mangalam, K., An, Y., Girase, H., and Malik, J. (2021). From goals, waypoints & paths to long term human trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15233–15242.

[Mangalam et al., 2020] Mangalam, K., Girase, H., Agarwal, S., Lee, K.-H., Adeli, E., Malik, J., and Gaidon, A. (2020). It is not the journey but the destination: Endpoint conditioned trajectory prediction. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, pages 759–776, Cham. Springer International Publishing.

[Na et al., 2010] Na, S., Xumin, L., and Yong, G. (2010). Research on k-means clustering algorithm: An improved k-means clustering algorithm. In *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, pages 63–67.

[Rudenko et al., 2020] Rudenko, A., Palmieri, L., Herman, M., Kitani, K. M., Gavrila, D. M., and Arras, K. O. (2020). Human motion trajectory prediction: a survey. *The International Journal of Robotics Research*, 39(8):895–935.

[Salzmann et al., 2020] Salzmann, T., Ivanovic, B., Chakravarty, P., and Pavone, M. (2020). Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision*, pages 683–700. Springer.

[Schöller et al., 2020] Schöller, C., Aravantinos, V., Lay, F., and Knoll, A. (2020). What the constant velocity model can teach us about pedestrian motion prediction. *IEEE Robotics and Automation Letters*, 5(2):1696–1703.

[Svenstrup et al., 2010] Svenstrup, M., Bak, T., and Andersen, H. J. (2010). Trajectory planning for robots in dynamic human environments. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4293–4298.

[Zhang et al., 2021] Zhang, T., Stackhouse, P. W., and Macpherson, B. (2021). A solar azimuth formula that renders circumstantial treatment unnecessary without compromising mathematical rigor: Mathematical setup, application and extension of a formula based on the subsolar point and atan2 function. *Renewable Energy*, 172:1333–1340.

# Appendix

# I. Licence

## Non-exclusive licence to reproduce thesis and make thesis public

I, **Karl Riis**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   **Human Trajectory Forecasting with Uncertainty Estimation**,
   *(*title of thesis)

   supervised by Meelis Kull and Novin Shahroudi.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Karl Riis
***17/05/2020***