8-9-2018

Cross-Scene Trajectory Level Intention Inference using Gaussian Process Regression and Naive Registration

Debasmit Das
Purdue University, das35@purdue.edu

C.S. George Lee
Purdue University, csglee@purdue.edu

ARTICLE TEMPLATE

Cross-Scene Trajectory Level Intention Inference using Gaussian Process Regression and Naive Registration

Debasmit Das[a] and C.S.George Lee[a]

[a]School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA

ARTICLE HISTORY
Compiled August 9, 2018

ABSTRACT
Human intention inference is the ability of an artificial system to predict the intention of a person. It is important in the context of human-robot interaction and homeland security, where proactive decision making is necessary. Human intention inference systems at test time is given a partial sequence of observations rather than a complete one. At a trajectory level, the observations are 2D/3D spatial human trajectories and intents are 2D/3D spatial locations where these human trajectories might end up. We study a learning approach where we train a model from complete spatial trajectories, and use partial spatial trajectories to test intention predictions early and accurately. We use non-parametric Gaussian Process Regression (GPR) as the learning model since GPR has been shown to model subtle aspects of human trajectory very well. We also develop a simple geometric transfer technique called Naive Registration (NR) that allows us to learn the model using training data in a source scene and then reuse that model for testing data in a target scene. Our results on synthetic and real data suggests that our transfer technique achieves comparable results as the technique of training from scratch in the target scene.

KEYWORDS
Intention Inference; Gaussian Process Regression; Transfer Learning; Trajectory prediction

1. Introduction

There is growing research interest on human intention inference because of its importance in human-robot collaboration/interaction and homeland security. Human intention inference is different from human activity recognition because at test time, the system is given a *partial* sequence of observations rather than a complete one and the goal is to identify the intention of the human. The observations can be a time-series of human pose, facial expressions, environmental contextual cues etc. Oztop, Wolpert, and Kawato (2005) suggested that similar parts of the brain are involved in movement planning and intention inference and therefore we will use only human motion information for intention inference. From now, throughout the paper, we shall use the terms intention and intent interchangeably.

The problem of human intention inference can be approached at different levels: *Trajectory* - where humans are treated as 2D/3D points moving in space with an intent. The intentions are salient locations in space. In a 2D scenario, this can be pedestrians moving in a scene towards destinations like buildings, cars etc, as observed from a bird's eye view. The locations of these destinations are the intentions in the scene. In a 3D scenario, trajectory level human intention inference can represent the human reaching target prediction problem where the trajectory of the human arm is used to infer the target locations. The goal would be to identify the most probable intent as *quickly* and *accurately* as possible. The other level is *Activity Level* Intent Inference, where the goal is to identify the activity of a person as quickly and accurately as possible. Some well know work in this area is activity level intent inference which has been called *Activity Prediction* (Ryoo, 2011) and *Activity Anticipation* (Koppula & Saxena, 2016) in the

CONTACT Debasmit Das. Email: das35@purdue.edu

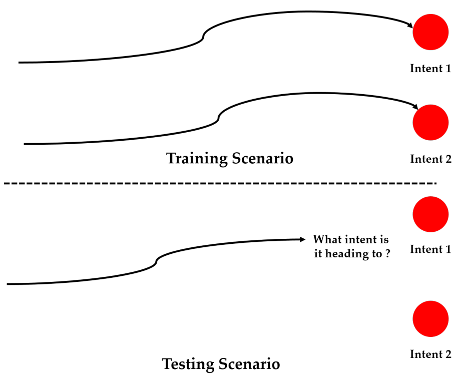


Figure 1. For Trajectory Level Intention inference we have complete trajectories towards the intents as training data. For test data, we have partial trajectories and our goal is to obtain the most probable intent

literature. Prior work in trajectory level intention inference has been carried out in both human reaching target prediction and pedestrian destination prediction problem domains. In Fig. 1 we see an example where training trajectories are completely reaching the intents. For the testing scenario, we only have partial trajectories reaching towards an intent.

The human reaching target prediction, which is human intention inference at 3D trajectory level, has generally been framed as a supervised-learning problem. Ravichandar and Dani (2015) used neural networks to model the transition dynamics of the human arm motion. Pérez-D'Arpino and Shah (2015) used a Bayesian approach to target prediction but realized training and testing on the same target-setup and did not generalize to arbitrary target setups and reaching trajectory starting locations. On the other hand, pedestrian destination prediction has generally been approached by the research community as an end result of pedestrian trajectory prediction. Xie, Shu, Todorovic, and Zhu (2016) formulates intents as *dark matter* where people travel to satisfy their needs. The trajectory model is formulated within a bayesian framework. Yi, Li, and Wang (2015) carries out trajectory prediction by using an energy map calculated using the scene layout, moving pedestrians and stationary groups. Other works in pedestrian trajectory prediction include (Ziebart et al., 2009), (Alahi et al., 2016), (Karasev, Ayvaci, Heisele, & Soatto, 2016). Our work solely focuses on pedestrian destination prediction as a primary application of 2D trajectory level intention inference without regard to trajectory prediction.

Our observation is that prior work on these trajectory level intention inference problems never addresses the issue of re-usability of models in different scenes. Most of these works only deal with models trained in one scene but do not talk about transferability of the model to different scenes. Transferability of models from a source scene to a target scene is important because it saves the much needed effort of training the model from scratch, provided we can guarantee competitive performance in our transfer methodology. We contribute by proposing our own transfer methodology. To our end, we propose the base model for trajectory level intention inference as Gaussian Process Regression (GPR). There are many advantages of selecting GPR as the base learning model. Firstly, GPR is very powerful in being able to quantify uncertainity over the intents using its predictive variance. Secondly, for low-dimensional problems like trajectory level intention inference, GPR can capture suble apsects of the trajectory space using its covariance function. Probabilistic inference is carried out over each intent using the Bayesian-update rule as the trajectory proceeds towards each intent. A forgetting factor is introduced in the Bayesian-update to measure the forgetfulness of the past belief information. Previously, Wang et al. (2013) used a similar GPR-based dynamical model for intention inference in human-robot table tennis. . For transferability between two scenes, we need to understand the difference between two scenes at a trajectory level. At a trajectory level, we assume that there is not much difference in the trajectory behaviour of the humans. The main difference is in terms of the location and quantity of intents in the two different scenes. We use this assumption to develop a simple geometric non-rigid registration technique that allows us to re-use models learnt in source scene and apply it to target scene. In our case, we do zero training on the target scene. The quantity and location arrangement of intents in the source and target scene can be arbitrary. Experiments are conducted for both synthetic and real datasets to see how our transfer technique performs on a target scene compared to a model trained from scratch in the target scene.

The structure of the paper is as follows. Section 2 describes the Gaussian Process Regression training and the prediction procedure. Section 3 discusses the experiments. Section 4 summarizes the results.

2. Proposed Gaussian Process Regression Method

Our goal is to predict the most probable intent location the human is going towards by obtaining a probability score over all the intent locations present in the scene. The probability score determines how likely each intent is to be reached by the human. This uncertainty over the intents can be quantified using the predictive distribution of GPR. For training the GPR model, data in the form of human trajectories directed towards different intents are recorded with each trajectory being time-sampled at the sensor frequency. In the case of pedestrian destination prediction, the sensor is a 2D surveillance camera. In the case of human reaching target prediction, the sensor is a 3D RGB-D sensor like Microsoft Kinect. We denote the position of the human trajectory at time t by $\mathbf{p}_t \in \mathbb{R}^d$, $d = 2, 3$ for the 2D and 3D case respectively. A trajectory towards the intent is a time-series of such hand positions. For intent prediction, we assume a state (position) transition model:

$$\mathbf{p}_{t+1} = \mathbf{f}(\mathbf{p}_t, g) + \boldsymbol{\epsilon}, \tag{1}$$

where $\mathbf{f}(\cdot)$ is the deterministic state transition function, and $\mathbf{p}_{t+1}$ and $\mathbf{p}_t$ are the next and current sampled trajectory points, respectively. $\boldsymbol{\epsilon} \in \mathbb{R}^d$ is a Gaussian distributed noise vector, and g is the intent index given as a positive integer, for G intents, $g \in \{1, 2, ...G\}$. Our intermediate goal is to learn the predictive distribution $p(\mathbf{p}_{t+1}|\mathbf{p}_t, g)$ using Gaussian Process Regression.

2.1. Gaussian Process Regression

This subsection briefly introduces Gaussian Process Regression (See Rasmussen (2006) for a detailed description). Suppose we have a dataset $D = \{(\mathbf{x}_i, y_i)_{i=1}^n\} \equiv (\mathbf{X}, \mathbf{y})$ of n training points, where $\mathbf{x}_i$ and y_i are inputs and desired outputs and $\mathbf{X} \in \mathbb{R}^{n\times d}$ (where d being the dimension of the input) and $\mathbf{y} \in \mathbb{R}^n$ are the result of stacking them vertically, respectively. Our model is as follows:

$$y_i = f(\mathbf{x}_i) + \epsilon_i, f \sim GP(\cdot|0, K), \epsilon_i \sim N(\cdot|0, \sigma^2), \tag{2}$$

where the prior on f is a Gaussian Process, which implies that $p(f)$ is a Gaussian process with zero mean and covariance function K. The likelihood $p(y|f)$ (ϵ_i) is Gaussian and therefore posterior on f is also Gaussian. Gaussian processes are parameterized by a mean function, $\mu(\mathbf{x})$, and a covariance or kernel function, $K(\mathbf{x}_i, \mathbf{x}_j)$. An example of mean function can be $\mathbf{a}^T\mathbf{x} + c$, $\mathbf{a} \in \mathbb{R}^d$ and $c \in \mathbb{R}$. A popular example of covariance function is $\alpha_1\text{exp}\{-\frac{|\mathbf{x}-\mathbf{x}'|^{\alpha_3}}{\alpha_2}\} + \alpha_4 + \alpha_5\delta_{ij}$, where δ_{ij} is the Kronecker delta function and $\boldsymbol{\alpha}$=$[\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5]^T$ are hyperparameters that need to be learnt.

There are two steps in Gaussian Process Regression:

- *Training:* The training process tunes the hyperparameters α_i's in the marginal likelihood $p(\mathbf{y}|\mathbf{X}, \boldsymbol{\alpha}) = N(\mathbf{0}, \boldsymbol{\Sigma}_n + \sigma^2\mathbf{I}_{n\times n})$, where $\boldsymbol{\Sigma}_n \in \mathbb{R}^{n\times n}$ is the kernel matrix such that $[\boldsymbol{\Sigma}_n]_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{x}_i, \mathbf{x}_j$ are instances of training inputs. $\mathbf{I}_{n\times n}$ is the identity matrix. The logarithm of the marginal likelihood is optimized with respect to σ and $\boldsymbol{\alpha}$.
- *Prediction:* After the hyperparameters are tuned, prediction is carried out for a test point $\mathbf{x}^\dagger$. The predictive distribution is then $p(y^\dagger|\mathbf{x}^\dagger) = N(\mu^\dagger, \sigma^{\dagger 2})$ where $\mu^\dagger = \boldsymbol{\Sigma}_n^{\dagger T}(\boldsymbol{\Sigma}_n + \sigma^2\mathbf{I}_{n\times n})^{-1}\mathbf{y}$ and

$\sigma^{\dagger 2} = \Sigma^{\dagger} - \mathbf{\Sigma}_n^{\dagger T}(\mathbf{\Sigma}_n + \sigma^2 \mathbf{I}_{n \times n})^{-1}\mathbf{\Sigma}_n^{\dagger} + \sigma^2$. $\mathbf{\Sigma}_n^{\dagger} \in \mathbb{R}^n$ are vector of covariances between test point $\mathbf{x}^{\dagger}$ and n training points, and $\Sigma^{\dagger} = K(\mathbf{x}^{\dagger}, \mathbf{x}^{\dagger})$.

In our problem, after having learnt GP's for the 3 coordinates in the 3D case, we have a predictive distribution of $p(x_{t+1}|\mathbf{p}_t, g)$, $p(y_{t+1}|\mathbf{p}_t, g)$, $p(z_{t+1}|\mathbf{p}_t, g)$ and combine them with independence assumptions to form:

$$p(\mathbf{p}_{t+1}|\mathbf{p}_t, g) = p(x_{t+1}|\mathbf{p}_t, g)p(y_{t+1}|\mathbf{p}_t, g)p(z_{t+1}|\mathbf{p}_t, g). \tag{3}$$

For the 2D case, we will have

$$p(\mathbf{p}_{t+1}|\mathbf{p}_t, g) = p(x_{t+1}|\mathbf{p}_t, g)p(y_{t+1}|\mathbf{p}_t, g). \tag{4}$$

For training, there can be two different approaches, which we will call as *holistic* and *atomistic*, respectively. In the holistic approach, we consider the intent index g as an input feature to the learning of the predictive distribution of each coordinate. For example in the 3D case, when learning the factor $p(x_{t+1}|\mathbf{p}_t, g)$, the input is (x_t, y_t, z_t, g) and the output is x_{t+1} and so on for the other co-ordinates. In this case we will have d GPR models for d dimensions. The alternative atomistic approach is to consider a separate model for each co-ordinate and for each intent g. For example in the 3D case, when learning the factor $p(x_{t+1}|\mathbf{p}_t, g)$, the input is (x_t, y_t, z_t) and the output is x_{t+1} and so we learn different models for other co-ordinates and other intents g. For this atomistic approach, we will have dG models for d coordinates and G intents.

Table 1. Worst Case Time-Complexity of different training approaches

Time Complexity	Holistic	Atomistic
Training (Serial)	dn^3	$dG(\frac{n}{G})^3$
Prediction (Serial)	dn^2	$dG(\frac{n}{G})^2$
Training (Parallel)	n^3	$(\frac{n}{G})^3$
Prediction (Parallel)	n^2	$(\frac{n}{G})^2$

For n training samples, we show in Table 1, that the atomistic approach is faster for prediction and training. We assume that training data is equally distributed among intent classes. For a parallel implementation, where model leaning and inference can be done simultaneously and for large number of intents G, the holistic approach becomes very slow compared to the atomistic approach. In terms of accuracy, we experimented and found that they do not have significant difference in performance. So we will use the atomistic approach for training and prediction, as it will allow deployment in future real-time systems easily.

2.2. Bayesian Inference Rule

After learning the predictive distribution model, we apply it to a test trajectory moving towards each intent used in the training phase. The output will be intent probabilities (belief) for each sample of the trajectory. We choose the *most probable intent* as the predicted intent. For calculating intent probabilities, we use Bayes rule and the first-order Markov assumption:

$$p(g|\mathbf{p}_{1:t+1}) \propto p(\mathbf{p}_{t+1}|\mathbf{p}_t, g)p(g|\mathbf{p}_{1:t}). \tag{5}$$

The factor $p(\mathbf{p}_{t+1}|\mathbf{p}_t, g)$ is learnt using GPR as discussed before while the factor $p(g|\mathbf{p}_{1:t})$ is recursively evaluated from the initial belief; that is, the prior $p(g|\mathbf{p}_1)$. The normalization is done after summing out the posterior probabilities $p(g|\mathbf{p}_{1:t+1})$ over all the intents.

In the Bayesian belief update, we introduce a forgetting factor $0<\lambda<1$, which measures how much past belief information, that is, $p(g|\mathbf{p}_{1:t})$, the model needs to forget. $\lambda = 1$ implies taking only the present likelihood $p(\mathbf{p}_{t+1}|\mathbf{p}_t, g)$ and forgetting the past belief. The modified equation would be:

$$p(g|\mathbf{p}_{1:t+1}) \propto p(\mathbf{p}_{t+1}|\mathbf{p}_t, g)p(g|\mathbf{p}_{1:t})^{1-\lambda}. \tag{6}$$

After comparing the performance (accuracy, loss etc.) of the model with different λ, we could choose the λ that produces the best performance. This technique has been used in Wang et al. (2013), where the performance was shown to be insensitive to λ. Alternatively, we use a novel *stationary state condition* to find the acceptable value of λ. The stationary state condition (S.S.C.) suggests that the target probabilities should not change if the states do not change in the next time step. Mathematically, it implies that whenever $\mathbf{p}_{t+1} = \mathbf{p}_t$, after applying Bayes update rule, we should have $p(g|\mathbf{p}_{1:t+1}) = p(g|\mathbf{p}_{1:t})$ for $g \in \{1, 2\}$. To study how the probability dynamics changes with λ and how it relates to S.S.C., we consider the 2 target case, where $g \in \{1, 2\}$. Normalizing Eq. (6) for target 1, we obtain the following:

$$p(1|\mathbf{p}_{1:t+1}) = \frac{p(\mathbf{p}_{t+1}|\mathbf{p}_t, 1)p(1|\mathbf{p}_{1:t})^{(1-\lambda)}}{\sum_{g=1}^{2} p(\mathbf{p}_{t+1}|\mathbf{p}_t, g)p(g|\mathbf{p}_{1:t})^{(1-\lambda)}}. \tag{7}$$

Changing the notation such that $p(1|\mathbf{p}_{1:t+1}) = p_{t+1}$ and $p(1|\mathbf{p}_{1:t}) = p_t$, we have $p(2|\mathbf{p}_{1:t}) = 1 - p_t$ and the likelihoods be $p(\mathbf{p}_{t+1}|\mathbf{p}_t, 1) = l_{1t}$ and $p(\mathbf{p}_{t+1}|\mathbf{p}_t, 2) = l_{2t}$ and $r_t = \frac{l_{2t}}{l_{1t}}$ be the likelihood ratio. Putting notations back into Eq. (7) and rearranging, we get the recurrence relation of the probability as

$$p_{t+1} = \frac{1}{r_t(\frac{1}{p_t} - 1)^{1-\lambda} + 1}. \tag{8}$$

As motion starts, the prior will be p_0. If the states (positions) do not change, the likelihoods (l_{1t},l_{2t}) and therefore the likelihood ratios (r_t) will be the same throughout the recurrences. Setting $r_t = r$, if we want the steady state of the sequence (p_∞), we set $p_{t+1} = p_t$ in Eq. (8) and obtain,

$$p_\infty = \frac{1}{r^{\frac{1}{\lambda}} + 1}. \tag{9}$$

If we have a prior $p_0 = 0.5$, and if $r = 1$; that is, the likelihood of either target are equal(physically represents human arm in an ambiguous position), p_∞ will be 0.5 and S.S.C. holds eventually. Practically, r will never be 1. It will be either $r > 1$ or $r < 1$. When r is just greater than 1; that is, 1.1, for $\lambda = 0$, $p_\infty = 0$ and for $\lambda = 1$, $p_\infty = 0.47$, which is close to prior 0.5. When r is just less than 1; that is, 0.9, for $\lambda = 0$, $p_\infty = 1$ and for $\lambda = 1$, $p_\infty = 0.52$, which is close to prior 0.5. Thus, higher λ yields less divergence from the prior and is closer to follow S.S.C. Fig. 2 depicts the situation for $r > 1$, where we

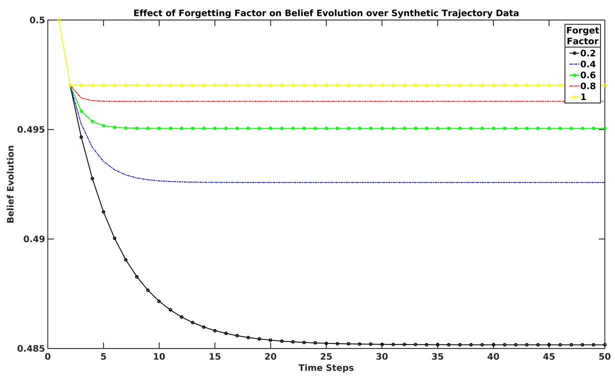


Figure 2. Belief evolution for different forgetting factor λ

also see that the belief settles very quickly for $\lambda = 1$, and for $\lambda = 0.2$ it takes more time to settle. Since we want to follow the S.S.C. as closely as possible but also not to forget all the past belief completely ($\lambda = 1$), we therefore settle with an intermediate value of $\lambda = 0.8$ for our experiments.

2.3. Transfer Technique

Traditionally, transfer learning (See (Pan & Yang, 2010) for a survey) between two scenes occurs with availability of data in source scene as well as little availability of data in target scene, during training the model. In our case we assume availability of data in source scene but no data in target scene during training. We assume that the only information about target scene we have after training is the location of intents in that target scene. This allows us to devise a transfer technique where we can reuse the learned model from the source scene on a number of target scenes after knowing the intent locations in those target scenes. This will be advantageous as we do not have to train a new model from scratch using data in the target scene. Before describing our method, we introduce some notations and definitions regarding transfer learning.

A *domain* $\mathcal{D}$ consists of feature space $\mathcal{X}$ and probability distribution $P(X)$, such that $X = \{x_1, x_2, ..., x_n\} \in \mathcal{X}$ For particular domain, $\mathcal{D} = \{\mathcal{X}, P(X)\}$ a *task* $\mathcal{T}$ consists of 2 components: a label space $\mathcal{Y}$ and an objective function $f(\cdot)$ with $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$. The $f(\cdot)$ is not observed but is learned from training data consisting of pairs of $\{x_i, y_i\}$, where $x_i \in X$ and $y_i \in \mathcal{Y}$. The function $f(\cdot)$ is used to predict the label of an instance x. From a probabilistic view $f(\cdot)$ is written probabilistically as the discriminative model $P(y|x)$. We define a *scene* as $\mathcal{S} = \{\mathcal{D}, \mathcal{T}\}$. In our problem of trajectory level intention inference the feature space is the $d-$dimensional space containing trajectories. The marginal $P(X)$ is generally unknown. The label space $\mathcal{Y}$ is the space of possible intents. So $|\mathcal{Y}|$ will be equal to the number of intents. Likewise, we will have two scenes - the source scene $\mathcal{S}_S$ and the $\mathcal{S}_T$. Accordingly, the definition of transfer learning is as follows -

Definition (*Transfer Learning*) Given a source scene $\mathcal{S}_S = \{\mathcal{D}_S, \mathcal{T}_S\}$ and a target scene $\mathcal{S}_T = \{\mathcal{D}_T, \mathcal{T}_T\}$, transfer learning aims to help improve learning of predictive function $f_T(\cdot)$ in $\mathcal{S}_T$ using knowledge learnt in scene $\mathcal{S}_S$, where $\mathcal{S}_S \neq \mathcal{S}_T$. For our problem of trajectory level intention inference, the source scene $\mathcal{S}_S$ differs from the target scene $\mathcal{S}_T$ in terms of the quantity and location of intents. If we directly apply the model learned from the source scene $\mathcal{S}_S$ to the target scene $\mathcal{S}_T$, there will be lot of error. The error will be due to difference in spatial distribution of trajectory data between the source and target scene. Our approach is to find a transformation from the target scene data space so that we can carry out inference in the source scene data space. For our problem of trajectory level intention inference we are learning the factor $p(\mathbf{p}_{t+1}|\mathbf{p}_t, g)$ (g is the intent number) which is equivalent to learning the generative model $P(x|y)$ according to the notations of transfer learning. In our case we will have a separate generative model for each intent defined using the atomistic approach previously. In a target scene, the intents will be located differently. Therefore, $P(x_S|y_S) \neq P(x_T|y_T)$. However since we have a separate model for each intent, we could find a spatial transformation from the target scene feature space to the corresponding source scene feature space for the corresponding intents respectively. We had previously used a feature transformation technique in our work on domain adaptation (Das & Lee, 2018b, 2018a). In that case we had unlabelled data in the target domain. In this case, the only information that we will use in the target scene are the intent locations. The notion of spatial transformation is intuitive because the target scene intent is different from the source scene intent only in terms of the location in space. We would carry out this transformation for all the intents and then carry out inference (See Eq. (6)).This will allow the source scene model re-usability and we would not have to train a model from scratch, using the data in the target scene. We will now describe our transfer technique for both $|\mathcal{Y}_S| = |\mathcal{Y}_T|$ and $|\mathcal{Y}_S| \neq |\mathcal{Y}_T|$

2.3.1. Same Number of Intents ($|\mathcal{Y}_S| = |\mathcal{Y}_T|$), Different Locations

Our goal is to reuse the model learned in the source scene to the target scene. Suppose, we have already learnt the predictive distribution model $p(\mathbf{p}_{t+1}|\mathbf{p}_t, g_S)$ in the source scene. g_S is the intent in the source scene located at $\mathbf{g}_S \in \mathbb{R}^d$. g_T is the corresponding intent in the target scene located in the $\mathbf{g}_T \in \mathbb{R}^d$. Let the starting point of the test trajectory in the target scene be located at $\mathbf{s}_T \in \mathbb{R}^d$. We want to find the corresponding starting point $\mathbf{s}_S \in \mathbb{R}^d$ in the source scene. For that we conduct a 1-Nearest Neighbour search from among the starting points of trajectories going towards intent g_S in the source scene. From among these starting points, we choose $\mathbf{s}_S$ such that the vector $\mathbf{s}_S - \mathbf{g}_S$ is the closest to $\mathbf{s}_T - \mathbf{g}_T$ in the $d-$dimensional space. The reason behind doing this is that we want to transform the test trajectory in target scene to the closest located corresponding trajectory in the source scene. Let's call this correspondence transformation $\mathbf{m}(\cdot) : \mathbb{R}^d \mapsto \mathbb{R}^d$ that maps trajectory points going towards intent g_T to possible trajectory points as if it were going towards intent g_S. To carry out the transition inference $p(\mathbf{p}_{t+1}|\mathbf{p}_t, g_T)$, where $\mathbf{p}_{t+1}$, $\mathbf{p}_t \in \mathbb{R}^d$ are the next and current location of trajectory points towards intent g_T, we map these points using $\mathbf{m}(\cdot)$ and carry out inference in our already learned model $p(\mathbf{m}(\mathbf{p}_{t+1})|\mathbf{m}(\mathbf{p}_t), g_S)$. Thus, we assume that $p(\mathbf{p}_{t+1}|\mathbf{p}_t, g_T) \approx p(\mathbf{m}(\mathbf{p}_{t+1})|\mathbf{m}(\mathbf{p}_t), g_S)$ We follow the similar approach for for all the corresponding intents in the source and the target scene. Once this is done, we use Eq. (6) to find the posteriors $p(g_T|\mathbf{m}(\mathbf{p}_{1:t+1}))$ and then normalize over all the intents to find the net probability scores. Now, we describe a method to find $\mathbf{m}(\cdot)$.We consider two virtual sticks, one that connects from $\mathbf{s}_S$ to $\mathbf{g}_S$ and another that connects $\mathbf{s}_T$ to $\mathbf{g}_T$ and call them vectors $\mathbf{l}_S$ and $\mathbf{l}_T$, respectively. We find the rotation matrix $\mathbf{R}_{ST}$ that transforms the unit vector $\hat{\mathbf{l}}_S$ to $\hat{\mathbf{l}}_T$. To transform point $\mathbf{p}_T$ lying on the trajectory towards intent g_T to a corresponding point $\mathbf{p}_S$ as if it is going towards intent g_S, we use the following equation:

$$\mathbf{p}_S = \mathbf{m}(\mathbf{p}_T) = \mathbf{R}_{ST}(\mathbf{p}_T - \mathbf{s}_T)\frac{||\mathbf{l}_S||}{||\mathbf{l}_T||} + \mathbf{s}_S, \tag{10}$$

which describes the transformation between the relative position vectors of the starting locations and

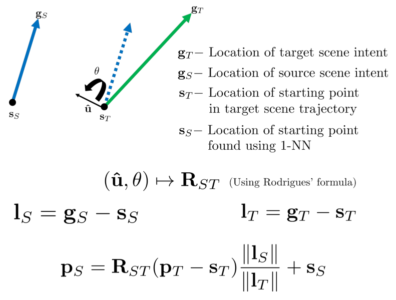

$\mathbf{g}_S$
$\mathbf{g}_T$
θ
$\hat{\mathbf{u}}$
$\mathbf{s}_S$
$\mathbf{s}_T$
$\mathbf{g}_T-$ Location of target scene intent
$\mathbf{g}_S-$ Location of source scene intent
$\mathbf{s}_T-$ Location of starting point in target scene trajectory
$\mathbf{s}_S-$ Location of starting point found using 1-NN
$(\hat{\mathbf{u}}, \theta) \mapsto \mathbf{R}_{ST}$ (Using Rodrigues' formula)
$\mathbf{l}_S = \mathbf{g}_S - \mathbf{s}_S$
$\mathbf{l}_T = \mathbf{g}_T - \mathbf{s}_T$
$\mathbf{p}_S = \mathbf{R}_{ST}(\mathbf{p}_T - \mathbf{s}_T)\frac{||\mathbf{l}_S||}{||\mathbf{l}_T||} + \mathbf{s}_S$

Figure 3. Proposed naive registration.

intent locations for the source and target intents. $\frac{||\mathbf{l}_S||}{||\mathbf{l}_T||}$ is the scale factor depending on how far the intents are located with respect to the starting position. We use the Rodriguez formula for deriving the rotation matrix. If $\hat{\mathbf{u}}$ is the unit vector representing the axis of rotation and θ is the angle of rotation, the rotation matrix $\mathbf{R}$ is given as

$$\mathbf{R} = \mathbf{I} + (\sin\theta)\mathbf{U} + (1 - \cos\theta)\mathbf{U}^2 \tag{11}$$

where $\mathbf{U}$ is the skew-symmetric form of $\hat{\mathbf{u}}$. If $\hat{\mathbf{u}} = [u_1, u_2, u_3]^T$, then the skew-symmetric matrix,

$$\mathbf{U} = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \tag{12}$$

Fig. 3 illustrates and explains this approach. We call this method as *Naive Registration* (NR). This NR is carried for all the corresponding intents in the source and target scene. Now, we discuss how we apply this approach for different number of intents in the source and target scene.

2.3.2. Different Number of Intents ($|\mathcal{Y}_S| \neq |\mathcal{Y}_T|$)

We consider two cases, one when $|\mathcal{Y}_S| > |\mathcal{Y}_T|$ and $|\mathcal{Y}_S| < |\mathcal{Y}_T|$. When $|\mathcal{Y}_S| > |\mathcal{Y}_T|$, that is when the number of intents in the source scene is more than the number of intents in the target scene, we just use $|\mathcal{Y}_T|$ of the $|\mathcal{Y}_S|$ models learned in the source scene. Then, we apply Naive Registration (NR) as described in the previous section to those $|\mathcal{Y}_T|$ models and use Bayesian Update rule (6) to carry out inference over the $|\mathcal{Y}_T|$ intents in the target scene.

The case with $|\mathcal{Y}_S| < |\mathcal{Y}_T|$ is more complex. In classification, we generally extend binary classifiers to multi-class classification using the one-vs.-one pair approach or the one-vs.-all approach. So, we could use similar approach for extending intention inference from a scene with $|\mathcal{Y}_S|$ intents to a scene with $|\mathcal{Y}_T|$ intents. However, the problem with these approaches is that they do not have a common reference to compare against. Using such approaches might yield incorrect inference results. Therefore, we propose a method of intention inference for the $|\mathcal{Y}_S| < |\mathcal{Y}_T|$ case which we call the *pivot* approach. The pivots are reference dummy intents that we choose in the target scene. We choose $|\mathcal{Y}_S| - 1$ pivots in the target scene. We chose the location of pivots to be far away from the scene. For example if the scene was normalised within limits of $[0, 1]$ in all the d dimensions, we could choose pivots randomly at 5 units. The main advantage of choosing pivots far away is that they do not become the most probable intents and interfere with the inference procedure but can still act as a reference for final probabilistic inference over the target scene intents. The process can be described in the following algorithm.

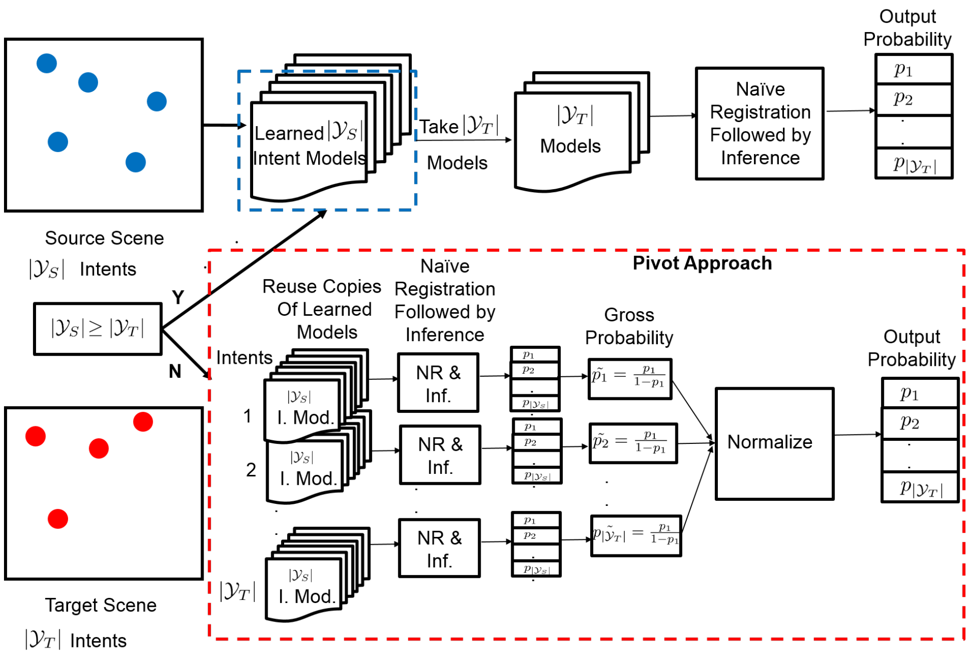
Algorithm 1: Pivot Approach

```
Select $|\mathcal{Y}_S| - 1$ pivots in target scene
for each point in test trajectory do
    for intent $i = 1, 2, ...|\mathcal{Y}_T|$ in target scene do
        Choose intent $i$ and $|\mathcal{Y}_S| - 1$ pivots as intent set
        Carry Out Naive Reg. (10) & Bayes Update
        Set gross probability $\tilde{p_i} \leftarrow \frac{p}{1-p}$
        Where $p$ is the belief of intent
    end
    Normalize :$p_i = \frac{\tilde{p_i}}{\sum_i \tilde{p_i}}$
end
```

As an example, consider the case where we have $|\mathcal{Y}_S| = 2$ intents in the source scene and $|\mathcal{Y}_T| = 3$ in the target scene. Therefore, we will have $|\mathcal{Y}_S| - 1 = 1$ pivot in the target scene. At a particular instance of the Bayesian inference of a test trajectory in the target scene, let the probability of Intent 1 vs. the pivot be [0.7; 0.3], Intent 2 vs. the pivot be [0.6; 0.4] and Intent 3 vs. the pivot be [0.5; 0.5]. Then, the gross-probability of Intent 1, Intent 2 and Intent 3 are $\frac{0.7}{0.3} = 2.33$, $\frac{0.6}{0.4} = 1.5$ and $\frac{0.5}{0.5} = 1$. They are normalized to obtain net probabilities of 0.482, 0.31, 0.207 respectively. The full transfer methodology for $|\mathcal{Y}_S| \neq |\mathcal{Y}_N|$is described in Fig. 4

3. Experiments and Experimental Results

All the simulations pertaining to the Gaussian Process Regression (GPR) model has been performed in MATLAB. For training the GPR, we chose the *matern* covariance function because it has been shown to describe spatial data like human motion quite well (Stein, 2012). For evaluation purpose we use plots of Accuracy v/s Observation ratio (AvOR). Observation Ratio is the fraction of time observed to the total time of a trajectory. Accuracy is obtained as the averaged accuracy over all intent classes. A model has better performance on a dataset if it has higher accuracies at lower observation ratios.



Output Probability
p_1
p_2
$p_{|\mathcal{Y}_T|}$
Learned $|\mathcal{Y}_S|$ Intent Models
Take $|\mathcal{Y}_T|$ Models
$|\mathcal{Y}_T|$ Models
Naïve Registration Followed by Inference
Source Scene
$|\mathcal{Y}_S|$ Intents
$|\mathcal{Y}_S| \geq |\mathcal{Y}_T|$
Y
N
Pivot Approach
Reuse Copies Of Learned Models
Naïve Registration Followed by Inference
Gross Probability
Intents
1
2
$|\mathcal{Y}_T|$
$|\mathcal{Y}_S|$ I. Mod.
NR & Inf.
$\tilde{p}_1 = \frac{p_1}{1-p_1}$
$\tilde{p}_2 = \frac{p_1}{1-p_1}$
$\tilde{p}_{|\mathcal{Y}_T|} = \frac{p_1}{1-p_1}$
Normalize
Target Scene
$|\mathcal{Y}_T|$ Intents

Figure 4. Full Transfer methodology using Naive Registration and Pivot Approach

3.1. Synthetic Data Experiments

We use synthetic data to verify our method before applying them to real data. This is necessary because in real data-sets, we cannot arbitrarily change the number of intents in the source and target scenes and observe the performance variations. For generating synthetic data, we use Bezier curves which has been previously used in modelling human trajectories (Faraway, Reed, & Wang, 2007). We follow suit and model bezier curve with 4 control points. These are also called as cubic Bezier curves. If we have four control points $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, all $\mathbf{p}_i \in \mathbb{R}^d$, the bezier curve $\mathbf{p}(t)$ is given by

$$\mathbf{p}(t) = (1-t)^3\mathbf{p}_0 + 3(1-t)^2t\mathbf{p}_1 + 3(1-t)t^2\mathbf{p}_2 + t^3\mathbf{p}_3 \tag{13}$$

for $0 \leq t \leq 1$. Samples of t are chosen such that $t = \text{Be}(v|\alpha, \beta)$. $\text{Be}(\cdot)$ is the CDF of beta distribution with parameters $\alpha = 2, \beta = 2$. v are uniformly sampled points in the range $[0, 1]$. For our intention inference problem, when generating trajectory data towards a particular intent location, we choose $\mathbf{p}_3$ to be the intent location and $\mathbf{p}_0 \sim \mathcal{U}^d(0, 1)$, $\mathbf{p}_1 \sim \frac{1}{3}\mathbf{p}_0 + \frac{2}{3}\mathbf{p}_3 + \mathcal{U}^d(-0.1, 0.1)$, $\mathbf{p}_2 \sim \frac{1}{3}\mathbf{p}_3 + \frac{2}{3}\mathbf{p}_0 + \mathcal{U}^d(-0.1, 0.1)$. $\mathcal{U}^d(0, 1)$ is a $d-$dimensional uniform distribution. A sample 2D dataset generated using Bezier curves is given in Fig. 5

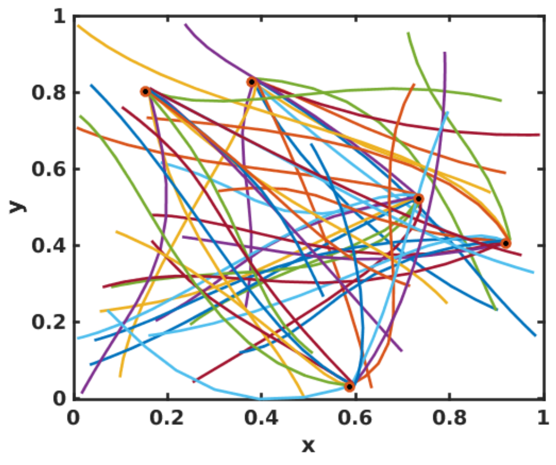


Figure 5. A sample synthetic 2D dataset of 5 intents with 10 trajectories per intent. Black circles with red borders show the intent location

Using such toy datasets, our goal is to report AvOR plots. All the experiments are performed over ten trials. The ten trials were created by randomly dividing the datasets with different seed. Each trial was tested using five-fold cross validation (80% for training and 20% for testing from each intent class). Accuracy is reported as an average over all test trajectories over all intent classes for 50 to 100 percent of observed trajectory time. For cross-scene evaluation, if the source scene is A and the target scene is B, the following three experiments are possible-

- Model trained on Scene A and tested on Scene B without NR transfer technique. (A-B). We do not report results of this because the AvOR performance is poor (<50% accuracy) and the plot will be random.
- Model trained on Scene A and tested on Scene B with NR transfer technique. (A$\rightarrow$B). We will call results of these experiments as Cross-Scene Performance
- Model trained on Scene B and tested on Scene B (B-B). We will call results of these experiments as In-Scene performance.

Our goal is to achieve the Cross-Scene performance as close as possible to the In-Scene performance on the same scene. We also repeat reverse experiments with B as the source scene and A as the target scene. Firstly, we consider the case when the number of intents in the source and target scene are same but the intent locations are different. For that, we synthetically generate 2 scenes with each scene containing 8 intents at different locations from each other. Results of the experiment is shown in the leftmost plot of Fig. 6 . From the results it shows that A-A is better than B$\rightarrow$A but are comparable. On the other hand, results of the experiment A$\rightarrow$B is way better than B-B. In a way, we have achieved cross-scene performance close to (and sometimes better than) in-scene performance (that corresponds to training from scratch), using a simple NR technique that reuses models on target scenes with zero re-training. We now experiment with the situation where the source scene and target scene have different number of intents. We consider 3 scenes with 2, 11, 20 intents respectively. We chose such a variation in the intent quantity to see how our method performs when there is huge difference between source and target scene. As seen in the second from left plot of Fig. 6, the cross-scene and in-scene accuracy is above 97% for the second half of the observation time (0.5 to 1) and they are reasonably close to each other suggesting that Naive Registration works. Also, the accuracy is higher compared to the scene with 8 intents because there are only 2 intents and it is very easy to figure out the most probable intent early on in the trajectory. For the scene with 11 intents, we see that the cross-scene performance of the transfer (2$\rightarrow$11) is way better than the in-scene performance (11-11). The in-scene performance for (11-11) is slightly better than the cross-scene performance (20$\rightarrow$11). Generally, generalization (ie. discrepancy in test and train results) is poor if the test samples lie far apart from the training samples. In our case, when we are carrying out in-scene experiments (11-11) we have test samples that are not near the training

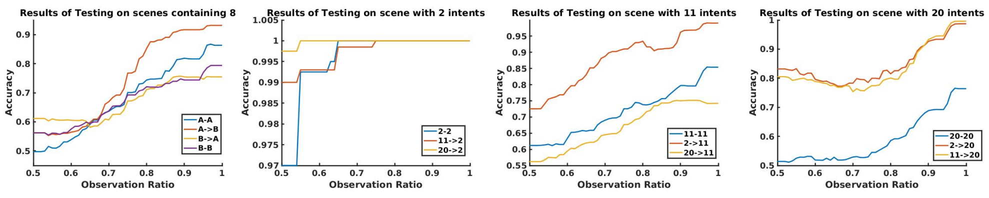


Figure 6. Result plots for the synthetic dataset

samples in the feature space. However, when we are carrying out cross-scene experiments (2→11) we are using NR to transform test sample to the closest training sample (using 1 Nearest Neighbor) making sure that the transformed test sample lies near training samples. This sometimes results into cross-scene performance being better than in-scene performance. This phenomenon is repeated for the scene containing 20 scenes, where both cross-scene performances (2→20,11→20) are better than the in-scene performance (20-20).

3.2. Real Data Experiments for Pedestrian Destination Prediction

For real datasets, intents are not localized as in synthetic datasets. Rather, intents are spread-out in the feature space. This is evident in Fig. 7, where we see that the end-point locations of trajectories are spread out in space. So, we cannot use Naive Registration transfer technique directly since we do not have a point location of an intent in the source and target scene. So, we propose to divide each intent into children sub-intents. This is carried out by clustering over the end points of the trajectories towards each intent. We choose Mean-shift algorithm as the clustering technique (Cheng, 1995). This clustering algorithm does not require pre-defined number of clusters. Also, since it is a mode-seeking algorithm, it assigns sub-intent locations to wherever there is a high-density of trajectory end-points of an intent. It also has good performance for any shape of clusters. The only tuning parameter required for the mean-shift algorithm is the bandwidth. For all our real-datsets, we apply min-max feature scaling so that all d-dimensional data is bounded to a data-space $[0,1]^d$. So, we choose a reasonable choice of bandwidth as 0.1 after careful experimentation. Results of mean-shift clustering are shown in Fig. 7. After clustering is done, the sub-intent locations over all the intents are collected together to form the final intent location set of the scene. Training and prediction is carried out using this final intent set. During prediction, at any instant of the test trajectory, whenever a sub-intent is inferred to be the most probable, we choose its parent intent to be the most probable. Accordingly, accuracy vs observation ratio plots are reported.

For real datasets, more salient intents will have more human trajectories towards it. As an example, in a pedestrian scenario, more people will visit a hospital than a shop. So we would capture more trajectory data for the hospital intent class than the shop intent class. The scenes used in the real-data set are in birds-eye view so that we can safely model pedestrian trajectories as lying in 2D space. Also, for the real data, a lot of pedestrian spatial trajectories are partial and not ending upto the destination eventually. For real data, we use publicly available pedestrian datasets ETH (Pellegrini, Ess, Schindler, & Van Gool, 2009) and UCY (Lerner et al., 2007). The ETH dataset consists of 2 scenes (ETH and Hotel). The UCY dataset consists of 2 scenes (ZARA and UCY). Each ZARA scene consists of 2 datasets ZARA-01 and ZARA-02. Our goal is to learn a model from one scene and transfer and test it to a different scene. Without our proposed transfer techniques, we would obtain accuracies in the range of 50% even at 100% of the observed trajectory. The performance would be even worse if the source and target scene intents are located very far from each other. We will use the notation A $\rightarrow$ B. for the situation when the model is trained on scene A and tested on scene B. The notation A-A implies that we use $\frac{2}{3}$ of the dataset of scene A for training and the remaining $\frac{1}{3}$ of the dataset of scene A for testing. Subsequently, for the case A $\rightarrow$ B, we use the previously trained model on $\frac{2}{3}$ of the dataset of scene A, apply the transfer technique and then test on the full dataset of scene B. This is the methodology, we will apply for all the

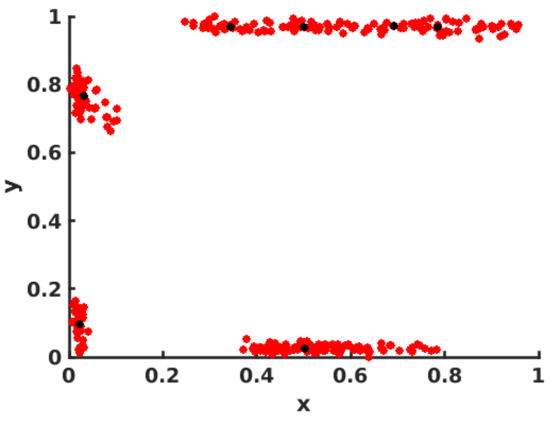


Figure 7. Red dots show the end-points of the trajectories of the normalized UCY (Lerner et al., 2007) Dataset. There are 4 intents. Since the intents are spread out, we estimate the location of sub-intents as shown by black dots. The estimated sub-intents are modes of the mean-shift clustering algorithm with a bandwidth of 0.1

.

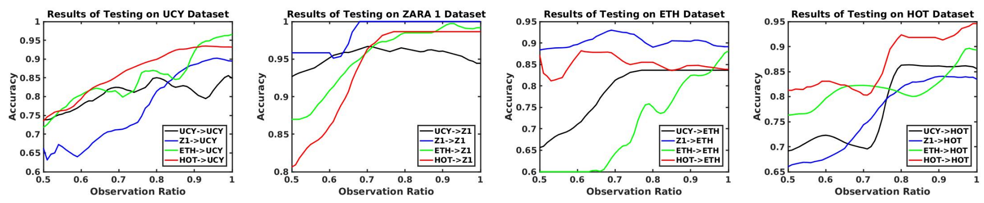


Figure 8. All Results on the Pedestrian Destination Prediction datasets

datasets of the different scenes. We want to study how the performance of A $\rightarrow$ B varies for different transfer scenario. From these plots in Fig. 8, we see that in almost all cases the transferred models are comparable to the baseline (A-A). The results are wiggly, because the pedestrians path towards the destination is not as smooth as the toy data. Also, unexpectedly the intention prediction performance decreases with observation ratio (UCY $\rightarrow$ ZARA 1). This because most of the trajectories might not end up at the destination.

3.3. *Real Data Experiments for Human Reaching Target Prediction*

The training demonstrations are obtained from the data repository used in (Luo & Berenson, 2015) which had been recorded using VICON, sampled at 120 Hz. From it, we use 14 trajectories for each of 2 targets for training. As these trajectories were not annotated with target locations, we used K-means clustering among the end-point locations of all the trajectories ($K = 2$) to obtain the estimated target locations $\mathbf{t}_1, \mathbf{t}_2 \in \mathbb{R}^3$ (See Fig. 9). We also considered a different scene consisting of 5 targets. (See Fig. 10). Results are reported in the same way as in pedestrian destination prediction experiments in Fig. 11. Scene 1 and Scene 2 corresponds to scene with 2 and 5 intents respectively. From, the plots we see that in almost all cases the transferred models are comparable to the baseline (A-A). Expectedly, performance on Scene 1 is better owing to lesser number of intents.

4. Conclusions

In this paper, we have developed a trajectory level intention inference system. We used the probabilistic formulation of Gaussian process regression for the prediction since it performs well for regression and

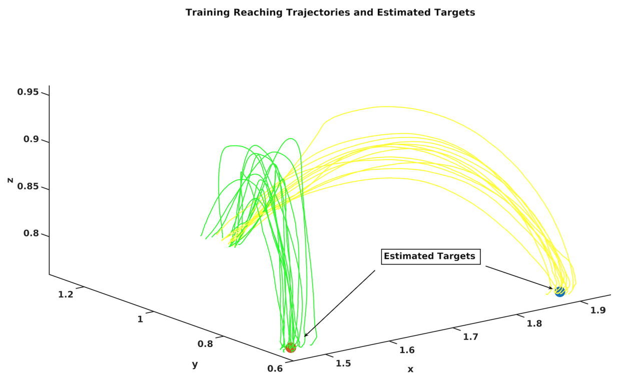


Figure 9. Training demonstrations and estimated targets.

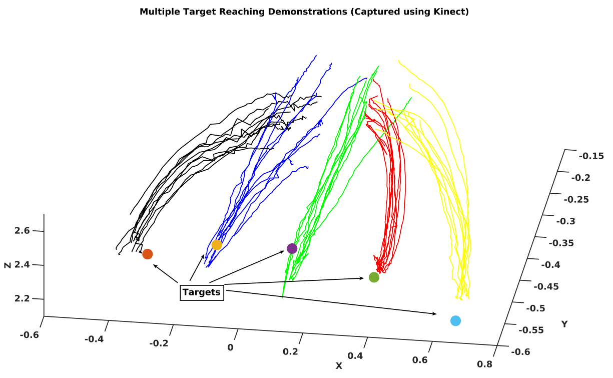


Figure 10. Another scene consisting of 5 targets

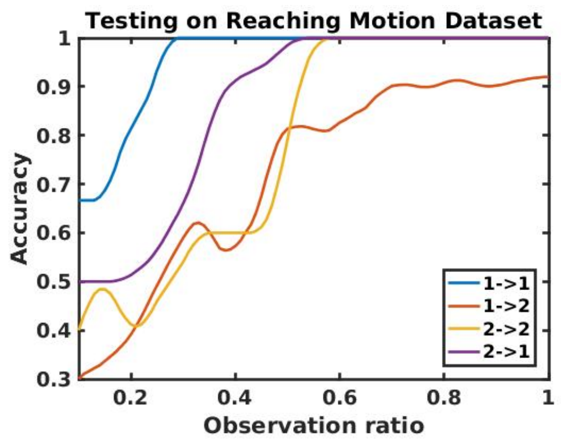


Figure 11. Cross-Scene Performance for reaching data.

uncertainty quantification with less training data. For generalization to different target locations, we proposed a naive registration technique. We also explored the stationary state condition for choosing the forgetting factor. We tested our approach on pedestrian and reaching motion datasets. We find the results to be comparable to the baseline approach. In the future, we would like to study intention inference in presence of obstacles and changing intents.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., & Savarese, S. (2016). Social lstm: Human trajectory prediction in crowded spaces. In *Proc. IEEE conference on computer vision and pattern recognition (cvpr)* (pp. 961–971).

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, *17*(8), 790–799.

Das, D., & Lee, C. (2018b). Unsupervised domain adaptation using regularized hyper-graph matching. *arXiv preprint arXiv:1805.08874*.

Das, D., & Lee, C. G. (2018a). Sample-to-sample correspondence for unsupervised domain adaptation. *Engineering Applications of Artificial Intelligence*, *73*, 80–91.

Faraway, J. J., Reed, M. P., & Wang, J. (2007). Modelling three-dimensional trajectories by using bezier curves with application to hand motion. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *56*(5), 571–585.

Karasev, V., Ayvaci, A., Heisele, B., & Soatto, S. (2016). Intent-aware long-term prediction of pedestrian motion. In *Proc. IEEE int. conf. robot. autom. (icra)* (pp. 2543–2549).

Koppula, H. S., & Saxena, A. (2016). Anticipating human activities using object affordances for reactive robotic response. *IEEE Trans. Pattern Anal. Mach. Intell.*, *38*(1), 14–29.

Lerner, A., Chrysanthou, Y., & Lischinski, D. (2007). Crowds by example. *Computer Graphics Forum*, *26*(3), 655–664.

Luo, R., & Berenson, D. (2015). A framework for unsupervised online human reaching motion recognition and early prediction. In *Proc. IEEE/RSJ int. conf. intell. robot. syst. (iros)* (pp. 2426–2433).

Oztop, E., Wolpert, D., & Kawato, M. (2005). Mental state inference using visual control parameters. *Cognitive Brain Research*, *22*(2), 129–151.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Trans. Knowledge Data Engg.*, *22*(10), 1345–1359.

Pellegrini, S., Ess, A., Schindler, K., & Van Gool, L. (2009). You'll never walk alone: Modeling social behavior for multi-target tracking. In *Proc. IEEE int. conf. computer vision* (pp. 261–268).

Pérez-D'Arpino, C., & Shah, J. A. (2015). Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification. In *Proc. IEEE int. conf. robot. autom. (icra)* (pp. 6175–6182).

Rasmussen, C. E. (2006). *Gaussian processes for machine learning*. The MIT Press.

Ravichandar, H. C., & Dani, A. (2015). Human intention inference and motion modeling using approximate EM with online learning. In *Proc. IEEE/RSJ int. conf. intell. robot. syst. (iros)* (pp. 1819–1824).

Ryoo, M. S. (2011). Human activity prediction: Early recognition of ongoing activities from streaming videos. In *Proc. IEEE int. conf. computer vision* (pp. 1036–1043).

Stein, M. L. (2012). *Interpolation of spatial data: Some theory for kriging*. Springer Science & Business Media.

Wang, Z., Mülling, K., Deisenroth, M. P., Amor, H. B., Vogt, D., Schölkopf, B., & Peters, J. (2013). Probabilistic movement modeling for intention inference in human–robot interaction. *Intern. J. of Robotics Research*, *32*(7), 841–858.

Xie, D., Shu, T., Todorovic, S., & Zhu, S.-C. (2016). Modeling and inferring human intents and latent functional objects for trajectory prediction. *arXiv preprint arXiv:1606.07827*.

Yi, S., Li, H., & Wang, X. (2015, June). Understanding pedestrian behaviors from stationary crowd groups. In *Proc. IEEE conference on computer vision and pattern recognition (cvpr)* (p. 3488-3496).

Ziebart, B. D., Ratliff, N., Gallagher, G., Mertz, C., Peterson, K., Bagnell, J. A., … Srinivasa, S. (2009). Planning-based prediction for pedestrians. In *Proc. IEEE/RSJ int. conf. intell. robot. syst. (iros)* (pp. 3931–3936).