IRL-based Prediction of Goals for Dynamic Environments

Fabio Previtali
Sapienza University of Rome

Alejandro Bordallo Micó
University of Edinburgh

Subramanian Ramamoorthy
University of Edinburgh

***Abstract*— Understanding activities of people in a monitored environment is a topic of active research, motivated by applications requiring context-awareness. Inferring future agent motion is useful not only for improving tracking accuracy, but also for planning in an interactive motion task. Despite rapid advances in the area of activity forecasting, many state-of-the-art methods are still cumbersome for use on realistic robots. This is due to the requirement of having good semantic scene and map labelling, as well as assumptions made regarding possible goals and types of motion. Many emerging applications require robots with modest sensory and computational ability to robustly perform such activity forecasting in high density and dynamic environments. We address this by combining a novel multi-camera tracking method, efficient multi-resolution representations of state and a standard Inverse Reinforcement Learning (IRL) technique, to demonstrate performance that is sometimes better than the state-of-the-art in the literature. In this framework, the IRL method uses agent trajectories from a distributed tracker, and the output reward functions, describing the agent's goal-oriented navigation within a Markov Decision Process (MDP) model, can be used to estimate the agent's set of possible future activities. We conclude with a quantitative evaluation comparing the proposed method against others from the literature.**

I. Introduction

Tracking multiple agents in a dense environment, be it humans or robots, is a challenging problem. A multitude of solutions exist for dealing with its different facets, such as overcoming occlusion or motion prediction. Inferring future motion of agents is useful not only for improving tracking accuracy, but also for planning in interactive tasks. Inferring future actions of an agent is known as *activity forecasting*.

Over the past decade, many novel methods have been developed for activity forecasting. These methods include those based on classifiers with structured outputs, which directly discriminate at the level of trajectories albeit with richer representations of the same. Another approach, called Inverse Reinforcement Learning, posits that the motion may be well described as being generated by optimisation within an MDP model, so that learning the motion is the same as inferring the implicit reward function being optimised. In order to keep these algorithms efficient, one often makes assumptions, such as that the scene and environment have been labelled in a semantically meaningful way (thus reducing the sample complexity) or that the space of possible motions is well parametrised and understood.

In many realistic robotics domains, we must apply these methods with a less clear understanding of potential goals (perhaps because the environment is new to the robot, such as in a rescue or rapidly changing construction environment), including in dynamic environments. We address this setting by combining a distributed multi-camera tracking method

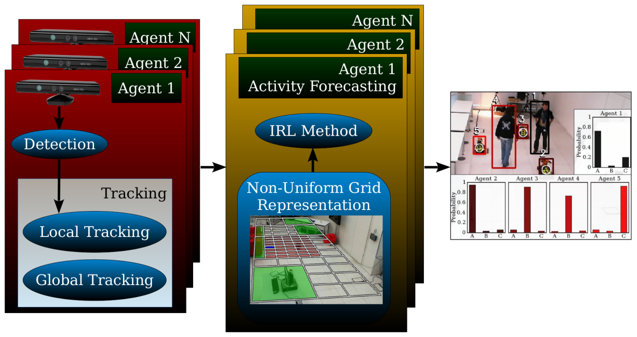


Fig. 1. Activity forecasting using the proposed framework based on IRL. Raw data are acquired from a set of Kinects and then processed by the PTracking algorithm. The agent trajectories in output are subsequently used by an IRL method to generate the agent's set of possible future activities.

and a multi-resolution representation of the environment, with a standard IRL method. This paper uses a relatively standard IRL method, allowing for the possibility that more sophisticated alternatives could be employed in future, within the overall proposed framework.

Contributions. We propose an integrated framework (see Fig. 1) that brings together an IRL technique with distributed tracking and multi-resolution state representation, such that it (1) does not rely on semantic scene labelling before activity forecasting, (2) incrementally updates the IRL model over time as new data becomes available and (3) makes use of non-uniform grids for state representation, making the entire framework linearly scalable with respect to the size of the environment.

We use *PTracking*, our distributed multi-camera multiple object tracker, as the first component. Local and global agent position, as well as velocity estimates, are updated via online tracking, providing trajectories that are then used within an IRL algorithm. This algorithm is fairly standard, and taken directly from the literature, in this first instantiation of our framework. While this allows us to implicitly consider sensor noise and false positives, we do recognise that a more elaborate IRL method that can target a Partially Observable Markov Decision Process motion model - something that is still hard to do and certainly not yet efficient for robot implementation - could be a useful future step. The output of this process is a set of reward functions - one for each goal generated by the algorithm (see Section IV-C) - per agent, describing its goal-oriented navigation policy across space. These reward functions represent the agent's set of possible future activities, forecasted or chosen from a probability distribution function via comparison with real-time observed agent behaviour. Additionally, the proposed approach allows us to perform anomaly detection by adopting a flexible notion of trajectory distance, such as the *Fréchet distance* which allows for graceful degradation even when dealing only with small fragments of the overall motion (see Section IV-C).

We show that our method infers agent goals accurately in a varied set of environments. We present empirical comparisons of our framework against state-of-the-art alternatives from the literature. We use as baselines, a method of Ziebart et al. [1], the *Maximum Entropy Markov Model* (*MEMM*) and a random walk method. A quantitative evaluation using multiple data sets shows how our approach performs competitively.

II. Related Work

The problem of understanding activities of people in a monitored environment is becoming increasingly well studied by researchers in multiple communities. The focus is often on two types of challenges - *human activity classification* and *activity recognition*. Human activity classification is an important yet difficult problem in computer vision [2], whose aim is to determine what people are doing, given a set of observations. It finds wide applicability in video surveillance [3], human-computer interfaces [4], sport video analysis [5] and content-based video retrieval [6]. Activity recognition, on the other hand, has as its goal the estimation of a belief-state from observations over time. It follows then that, activity recognition is a temporal classification problem; an agent must generate a sequence of labels, identifying the roles or behaviours of the other agents, given a sequence of observations [7]. Typically methods addressing both activity classification [8] and activity recognition [7] require a substantial amount of captured data during the training phase to generate useful models, a requirement that is not satisfied in many realistic applications. Additionally, even when such data is available, such frameworks may not be able to adapt the learnt model to changes in the monitored environment or, as is needed in some cases, adapt to the dynamics of the environment.

The focus of our work is on trajectory-based human activity analysis. As a proof-of-concept, we propose a *Trajectory-Based Inverse Reinforcement Learning* method for estimating future actions of people (or of robots, vehicles and other agents) from noisy visual input. Similar work has been carried out with success by Ziebart et al. [1], who report on activity forecasting by combining IRL algorithm with a semantic representation of the scene. It is not clear how that technique would deal with unstructured, and in particular not previously labelled, environments.

III. Distributed Multi-Camera Multiple Object Tracking

A. Problem Definition

The Distributed Multi-Camera Multiple Object Tracking problem can be formalised as follows. Let $\mathcal{O} = \{o_1, \ldots, o_n\}$ be the set of all moving objects, each one having a different identity, and $\mathcal{S} = \{s_1, \ldots, s_S\}$ be the set of arbitrarily fixed sensors, each one having limited knowledge about the environment (i.e., each camera can monitor only part of the scene). Moving objects are detected by a background subtraction algorithm and the number of objects n is unknown and can change over time. The set of measurements about the objects in the field-of-view of a camera $s \in \mathcal{S}$ at a time t is denoted by $z_{s,t} = \{z_{s,t}^{(1)}, \ldots, z_{s,t}^{(l)}\}$, where a measurement $z_{s,t}^{(i)}$ can be either a real object present in the environment or a false positive. The set of all the measurements gathered by all cameras at time t is denoted by $z_{\mathcal{S},t} = \{z_{s,t} \mid s \in \mathcal{S}\}$. The history in time of all the measurements coming from all cameras is defined as $z_{\mathcal{S},1:t} = \{z_{\mathcal{S},j} : 1 \leq j \leq t\}$. It is worth noticing that, we do not assume the measurements generated by the cameras to be synchronised. The goal is to determine, for each camera s, an estimation $x_{s,t}$ of the position of the objects at time t in a distributed fashion.

B. Distributed Multi-Clustered Particle Filtering

In order to achieve this goal, we estimate, for each camera s, the position $x_{s,t} = \{x_{s,t}^{(1)}, \ldots, x_{s,t}^{(v)}\}$ of the objects by merging all the available information. Although the cameras continuously send information about their observations, the estimation computed by one camera may be different from the others due to noise or delay in communication. Specifically, the overall objective is to determine the likelihood $p(x_{s,t} \mid z_{\mathcal{S},1:t})$ of the global estimation $x_{s,t}$ for each camera s, given the observations $z_{\mathcal{S},1:t}$ collected by all cameras.

We assume that the acquired observations are affected by an unknown noise that is conditionally independent among the cameras. During the acquisition process, each camera does not interact with the others, thus allowing for a factorisation of the likelihood of the global estimation that can be expressed by the following joint likelihood:

$$p(z_{\mathcal{S},t}|x_{s,t}) = \prod\nolimits_{s \in \mathcal{S}} p(z_{s,t}|x_{s,t}) \tag{1}$$

Given the assumption in Eq. (1), a fusion algorithm can be described using Bayesian Recursive Estimation:

$$p(x_{s,t}|z_{\mathcal{S},1:t}) = \frac{p(z_{\mathcal{S},t}|x_{s,t})p(x_{s,t}|z_{\mathcal{S},1:t-1})}{\int p(z_{\mathcal{S},t}|x_{s,t})p(x_{s,t}|z_{\mathcal{S},1:t-1})dx_{s,t}} \tag{2}$$

$$p(x_{s,t}|z_{\mathcal{S},1:t-1}) = \int p(x_{s,t}|x_{s,t-1})p(x_{s,t-1}|z_{\mathcal{S},1:t-1})dx_{s,t-1} \tag{3}$$

Eq. (2) and (3) represent a global recursive update that can be computed if and only if complete knowledge about the environment is available. Therefore, we propose to approximate the exact optimal Bayesian computation - Eq. (2) and (3) - by using a Distributed Particle Filter-based algorithm. To this end, we devise a novel method, called *PTracking*, based on Distributed Multi-Clustered Particle Filtering. The algorithm is divided into two phases, namely a *local estimation* phase and a *global estimation* phase (Algorithm 1). Each camera performs the local and global computation, sharing the obtained results in order to achieve a better representation of the current scene.

Local estimation. The local estimation phase (Algorithm 1, lines 1-7) contains three steps: 1) A particle filtering step, that computes the evolution of the local estimations given the local observations $z_{s,t}$ provided by the sensor; 2) A clustering step that determines the *Gaussian Mixture Model* (*GMM*) parameters of this distribution; 3) A *data association* step to assign an identity to each object $o \in \mathcal{O}$.

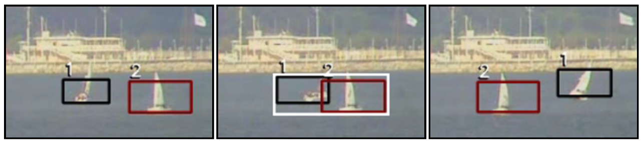
```
Algorithm 1: PTracking
  Input: perceptions $z_{s,t}$, local track numbers $i_{s,t-1}$, global
         track numbers $I_{s,t-1}$
  Data: set of local particles $\tilde{\xi}_{s,t}$, set of global particles $\tilde{\xi}_{\mathcal{S}',t}$,
        local GMM set $\mathcal{L}$, global GMM set $\mathcal{G}$
  Output: global estimations $x_{s,t} = (\boldsymbol{I}_{s,t}, \boldsymbol{\Lambda}_{s,t}, \boldsymbol{M}_{s,t}, \boldsymbol{\Sigma}_{s,t})$
1  begin
2      $\tilde{\xi}_{s,t} \sim \pi_t(x_{s,t}|x_{s,t-1}, z_{s,t})$
3      Re-sample by using the SIR principle
4      $\mathcal{L} = KClusterize(\tilde{\xi}_{s,t})$
5      $(\boldsymbol{i}_{s,t}, \boldsymbol{\lambda}_{s,t}, \boldsymbol{\mu}_{s,t}, \boldsymbol{\sigma}_{s,t}) = DataAssociation(\mathcal{L}, i_{s,t-1})$
6      Communicate belief $(\boldsymbol{i}_{s,t}, \boldsymbol{\lambda}_{s,t}, \boldsymbol{\mu}_{s,t}, \boldsymbol{\sigma}_{s,t})$ to other agents
7  end
8  begin
9      Collect $\mathcal{L}_{\mathcal{S}'}$ from a subset $\mathcal{S}' \subseteq \mathcal{S}$ of cameras within a $\Delta t$
10     $\tilde{\xi}_{\mathcal{S}',t} \sim \tilde{\pi} = \sum_{s \in \mathcal{S}'} \boldsymbol{\lambda}_{s,t} \mathcal{N}(\boldsymbol{\mu}_{s,t}, \boldsymbol{\sigma}_{s,t})$
11     Re-sample by using the SIR principle
12     $\mathcal{G} = KClusterize(\tilde{\xi}_{\mathcal{S}',t})$
13     $(\boldsymbol{I}_{s,t}, \boldsymbol{\Lambda}_{s,t}, \boldsymbol{M}_{s,t}, \boldsymbol{\Sigma}_{s,t}) = DataAssociation(\mathcal{G}, I_{s,t-1})$
14 end
```

The prediction step of the PF uses an initial guessed distribution, based on a *transition state* model π. Such a transition model makes a prediction of the next state based on the sensor movement. Then, using the previously computed state $x_{s,t-1}$, the transition model, given by the measurements $z_{s,t}$, is applied. Afterwards, from this hypothesised distribution, a set of samples is drawn and weighted exploiting the current local perception $z_{s,t}$. Finally, the *Sampling Importance Resampling* (*SIR*) principle is used to re-sample the particles which are then clustered in order to determine the parameters of the final GMM model. It is worth noticing that, in contrast to other related approaches, this step enables the creation of a more compact information structure allowing us to drastically reduce the communication overhead. A *data association* step is then applied to assign an identity (track number) to each object.

When the final GMM set has been computed, each camera broadcasts the set of GMM parameters describing all the objects detected.

KClusterize. The clustering phase is performed by using a novel clustering algorithm, called *KClusterize*, aiming at fulfilling the following three requirements: 1) number of objects to be detected cannot be known a priori, 2) low computational load for real-time applications and 3) Gaussian distribution for each cluster. Alternative clustering methods are not adequate since they either need to know the number of clusters in advance (e.g., *k-means*), or they are computationally expensive and not real-time (e.g., free-clustering algorithms like *Expectation-Maximization*, *BSAS* or *QT-Clustering*). KClusterize does not require any initialisation, it has a linear complexity and all the obtained clusters reflect a Gaussian distribution.

More specifically, KClusterize first clusters the particles trying to find all the possible Gaussian distributions. Then, a post-processing step is applied to verify that each cluster



(a) (b) (c)

Fig. 2. Group tracking. Two sailing boats are going to cross each other. Occlusions are handled considering the collapsing tracks to form a group, instead of tracking them separately.

actually represents a Gaussian distribution. To this end, all the non-Gaussian clusters are split (if possible) into Gaussian clusters. It is worth noticing that, the final number of Gaussian distribution components provided as output can be different from the one found during the first step. Finally, using such clusters a GMM set $(\boldsymbol{\lambda}_{s,t}, \boldsymbol{\mu}_{s,t}, \boldsymbol{\sigma}_{s,t})$, representing the estimations performed by the camera s, is created.

Global estimation. The global estimation phase (Algorithm 1, lines 8-14) starts receiving information from other cameras. Notice that, as already mentioned, the proposed method is asynchronous and the collection of information is limited to a small amount of time Δt. During this time information is received from a subset $\mathcal{S}' \subseteq \mathcal{S}$ of cameras. This mechanism is thus robust to communication delays and dead nodes, since the global estimation phase proceeds even if some node is not communicating or the communication channel is not reliable. Once data have been gathered, a particle set $\tilde{\xi}_{\mathcal{S}',t}$ is updated using the received GMM parameters $(\boldsymbol{i}_{s,t}, \boldsymbol{\lambda}_{s,t}, \boldsymbol{\mu}_{s,t}, \boldsymbol{\sigma}_{s,t})$ for $s \in \mathcal{S}'$. These particles are re-sampled in order to extract reliable quality information about the global estimates. Then, a weighting procedure is applied to the set. Instead of weighting the particles by using the whole pool of GMM parameters, we cluster them by again using *KClusterize* to obtain a new GMM pool. The weighting of particles is performed using such a new GMM pool. In this way the assigned weights are more consistent since only the most relevant parameters are considered. The global estimation phase determines the GMM parameter set of the tracked objects considering all the information available at time t (local observations and information received by other cameras). Finally, a *data association* step is applied to assign an identity to each object considering all the available information received by other cameras.

Data association. An identity (i.e., a track number) has to be assigned to each object, by associating the new observations to the existing tracks. This is the most difficult and fundamental step for any tracking algorithm. In our approach, we consider as features for data association the direction, the velocity and the position of the objects. Complete and partial occlusions can occur when objects are aligned with respect to the camera view or when they are very close to each other, making visual tracking hard. Our solution is to consider collapsing tracks to form a group, instead of tracking them separately (see Fig. 2). When multiple tracks have their bounding boxes moving closer to each other (Fig. 2a), the tracker saves their color histograms and it merges them into a group (Fig. 2b) - the histograms are used as models for re-identifying the objects when the occlusion phase is over

TABLE I

QUANTITATIVE COMPARISON ON PETS 2009 WITH STATE-OF-THE-ART METHODS. RESULTS TAKEN FROM CORRESPONDING PAPER OF THE AUTHORS.

	Leal-Taixé et al. [9]	Berclaz et al. [10]	Sharma et al. [11]	Breitenstein et al. [10]	Yang et al. [12]	PTracking Mono	PTracking Multi
MOTA	67.0%	73.2%	67.5%	74.5%	75.9%	**76.0%**	**87.4%**
MOTP	53.4%	60.3%	48.2%	56.3%	53.8%	**63.0%**	**72.2%**

(Fig. 2c). A group evolves considering both the estimated trajectory and the observations coming from the detector. When an occluded object becomes visible again, the stored histograms are used to re-assign the correct identification number, belonging to the corresponding registered track.

Quantitative analysis. We use the CLEAR MOT [13] metrics *MOTA* and *MOTP* to quantitatively measure the performance of the proposed tracking method. We use the ground-truth used in [14] and the CLEAR MOT metrics have been computed using the publicly available code provided by Zhang et al. [15]. The assignment of tracking output to ground-truth is done using the Hungarian algorithm with an assignment cut-off at 1 meter. *MOTP* is normalized to this cut-off threshold. Table I shows the quantitative comparison with state-of-the-art approaches on the PETS 2009 data set. It is worth noticing that this data set is one of the most challenging one for tracking systems. Finally, we use View1, View3 and View8 to perform the distributed tracking in the "PTracking Multi" setup.

IV. ACTIVITY FORECASTING FROM NOISY VISUAL OBSERVATIONS

A. Problem Definition

Our objective in *activity forecasting* is the task of estimating future actions of people (or of robots, vehicles and other agents), from noisy visual input. We address this using an IRL procedure which works on the output of the PTracking algorithm described in Section III. In a certain sense, the problem formulation ought to acknowledge that states are not known exactly, instead being estimated through a visual tracking process. So, the temporal modelling of activity may perhaps be described in the language of partially observable models. However, such models are rarely easy to work with, especially so when the goal is efficient real-time implementation on resource constrained robots. So, we proceed by making an assumption akin to 'certainty equivalence', modelling the temporal dynamics in terms of an MDP, which is fed the output of the PTracking algorithm which acts as state estimator. To the extent that the tracker maintains a fully probabilistic representation of objects and their motion and that the set of possible goals estimated by the IRL algorithm can be incrementally grown, this is a useful compromise that could be further relaxed in future work.

A finite discrete-time MDP is a tuple $(\mathbf{S}, \mathbf{A}, \mathbf{P}_{sa}, \mathbf{R}, \gamma)$ where: $\mathbf{S}$ is a finite set of N states, $\mathbf{A} = \{a_1, a_2, \ldots, a_k\}$ is a set of k actions (i.e., North, West, South, East), $\mathbf{P}_{sa}(\cdot)$ are the state transition probabilities upon taking action a in a state s, $\mathbf{R} : \mathbf{S} \times \mathbf{A} \mapsto \mathbb{R}$ is the reinforcement function, bounded in absolute value by R_{max} and $\gamma \in [0, 1)$ is the discount factor.

We adopt a standard IRL algorithm, due to Russel et al. [16], in this work. Their work makes use of the *policy optimality* theorem to derive a linear programming formulation of IRL as follows:

$$\min \textstyle\sum_{i=1}^{N} -x_i + \lambda(r_i^+ - r_i^-)$$
$$s.t.$$
$$\begin{cases} x_i \le (\mathbf{P}_{a_*} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_*})^{-1}\mathbf{R} & \\ \qquad\qquad\qquad\qquad \forall\, a \in \mathbf{A},\ i \in \{1, \ldots, N\} & \\ x_i \ge 0 & i \in \{1, \ldots, N\} \\ r_i = r_i^+ + r_i^- & i \in \{1, \ldots, N\} \\ |\mathbf{R}_i| \le R_{max} & i \in \{1, \ldots, N\} \end{cases} \tag{4}$$

Here, $\mathbf{P}_a$ denotes a NxN matrix in which element (i, j) gives the probability of transitioning to state j upon taking action a in state i, $\mathbf{P}_{a_*}$ denotes the current *optimal* policy, obtained by combining the optimal policy at the previous step and the output of the PTracking algorithm. In Eq. (4), the non-linear 1-norm operator $||\mathbf{R}||_1$ of the original problem stated by Russel et al. has been linearised by adding two more variables r_i^+ and r_i^- for every r_i variable with $i \in \{1, \ldots, N\}$. The non-linear operator $\min_{a \in \mathbf{A}}$, instead, has been linearised by introducing N new variables x_i where $i \in \{1, \ldots, N\}$. The linear programming problem defined in Eq. (4) can be easily and efficiently solved using standard techniques, such as the Simplex algorithm.

B. Inverse Reinforcement Learning Model

The result of solving the problem in Eq. (4) is the reward function (for each goal), that captures both interactions and movements of objects in the monitored environment. Future actions may be forecasted and suspicious trajectories (i.e., anomalies from the model obtained via IRL) recognised in advance. Since the problem defined by equation 4 is a discrete optimisation problem, we need to discretise continuous visual observations coming from the PTracking algorithm (see Section III). To this end, we propose a representation of the monitored environment based on a set of non-uniform grids, thus allowing for an effective and efficient representation of the monitored environment. In this manner, portions of the environment in which there are multiple interactions are represented by a set of dense grids, while parts of the environment that do not have particular interactions are described by sparse grids.

Grid update. First, a uniform grid mapping the monitored environment is constructed. Then, such a grid is periodically updated based on the information provided by the PTracking algorithm. High density locations are described richly by increasing the granularity of the grid mapping around that location, whereas parts of the environment having fewer

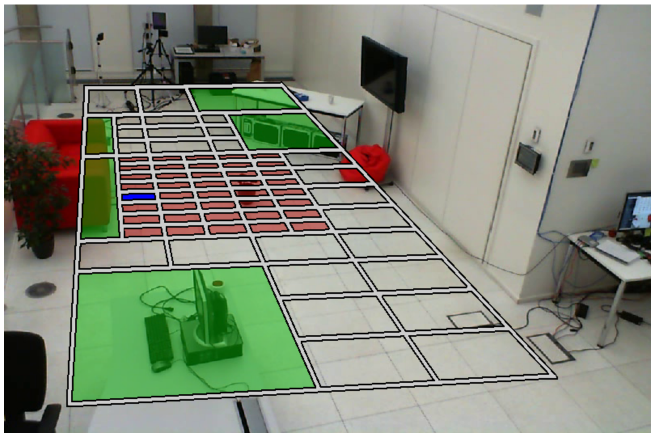


Fig. 3. Set of non-uniform grid representation of the environment. Dynamic regions (in red) are mapped with a set of dense grids, while the ones having few interactions (in green) are represented with sparse and small grids. The goal is highlighted by a blue cell of the grid.

interactions are described sparsely. By adopting the non-uniform grid representation, the proposed approach linearly scales with respect to the size of the environment as well as the computational resources required for solving the optimisation problem. An example of the non-uniform grid representation is depicted in Fig. 3, while a possible IRL model associated with it is shown in Fig. 4, in which the end goal is represented by the function's global maximum.

C. Activity Forecasting and Anomaly Detection

Our approach can be summarised in terms of the following key steps. The output of the optimisation process of Eq (4) is a set of G reward function models, one for each goal, $\mathcal{M}_s^G$ per agent s, where $s \in \mathcal{S}$ (see Section III-A). This allows us to generate a policy $\pi_s^{G_i}$ for each goal $G_i \in G$ per agent s. We can now build likelihoods over the set of goals G as follows.

Observed trajectory extraction. We gather object estimates from the PTracking algorithm considering an arbitrary temporal window ($5s$ for our experimental evaluation). Having acquired a set of trajectories $\mathcal{U}$, we ground each trajectory $u \in \mathcal{U}$ in every policy π_s^G.

Policy comparison. At this stage, we get the best fitting policy by comparing the target trajectory against a set of trajectories drawn from potential optimal policies. We do this using a combination of *Fréchet distance* and *cosine similarity*, to allow for the possibility that the target trajectory is merely a fragment of the overall optimal policy so that this notion of geometric similarity is one that better captures our notion of activity membership.

Goal prediction. We are finally able to predict in real-time the goal toward which each moving object is likely to be headed, by executing the policy that best matches the movement pattern of every object. It could happen that a trajectory $u \in \mathcal{U}$ does not match any model. This can happen for two main reasons. The first is that the trajectory fragment u is anomalous and refers to a suspicious activity pattern. The second, which we cannot always rule out, is that there may be a multitude of optimal policies that represent the activity class or that the environment has changed leading to

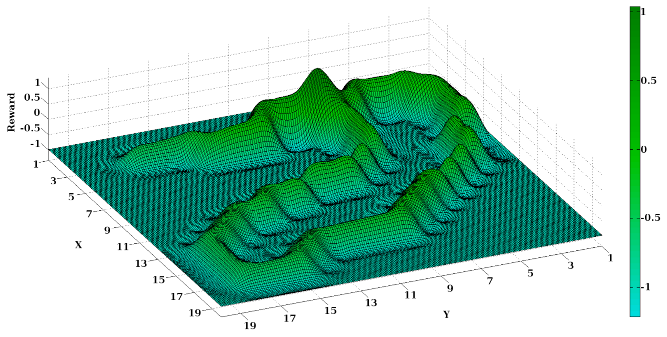


Fig. 4. Trajectory-based model learnt by solving the IRL problem defined in Eq. (4). The goal is represented by the highest point.

new types of motion. The latter, however, can be recognised by analysing the foreground model provided by the detector algorithm (see Section III-A) because it will hugely differ from the background model learnt so far. All learnt models are discarded until new models are available.

Goal sampling. In the general case, goals may not be defined ahead of time. In a setting such as a home environment, goals could be associated with routine activity and, say, tools or objects frequently used by a human user. Here, static points of interest can be conjectured from a relatively inexpensive scene analysis, providing much needed contextual structure of the environment. However, in a dynamic scenario like an airport, train station or even a fast changing construction or rescue site, this process may be infeasible and the set of potential goals identified from surface level analysis may be very large. This problem is compounded by the lack of clear and persistently identifiable structure in these rapidly changing environments. In these cases, we could generate potential goals by analysing the information provided by a tracking system (i.e., analysing trajectories of all moving agents), and work with respect to this set.

V. Experimental Evaluation

The framework has been tested in three different environments: Our HRI laboratory, the main entrance to our Informatics Forum and the VIRAT data set. A quantitative comparison, depicted in Fig. 5, demonstrates how our proposed framework outperforms alternatives in terms of NLL - Eq. (5), including a state-of-the-art approach of Ziebart et al., a Maximum Entropy Markov Model algorithm and a random walk baseline procedure.

Comparison metric. In each experiment, we have one demonstrated path, a sequence of states s_t and actions a_t, generated by all agents for a specific configuration of a scene. We compare the demonstrated path with the probabilistic distribution over paths generated by our IRL algorithm using the *Negative Log-Loss* (*NLL*) of a trajectory, as in [1], defined as follows:

$$NLL(s) = E_{\pi(a\,|\,s)}\Big[-\log \prod\nolimits_t \pi(a_t\,|\,s_t)\Big] \tag{5}$$

The NLL represents the expectation of the log-likelihood of a trajectory s under a policy $\pi(a\,|\,s)$. This metric measures the probability of drawing the demonstrated trajectory from the learnt distribution over all possible trajectories.

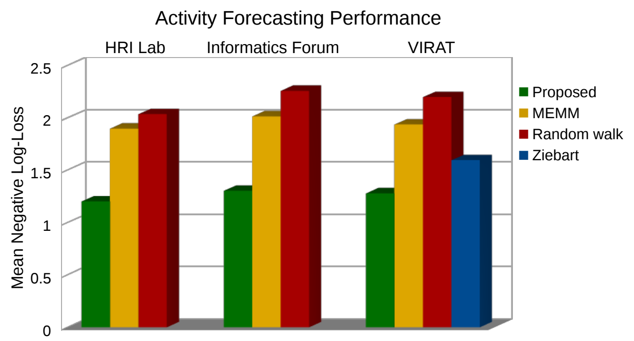


Fig. 5. Mean NLL of activity forecasting performance on chosen data sets.

Experimental setup. We compare our framework against a Maximum Entropy Markov Model based algorithm and a random walk baseline procedure in all scenarios. Moreover, in the VIRAT data set, we also compare against the approach proposed by Ziebart et al. in [1]. We use the same input data and the same state representation for the random walk baseline, the MEMM method and the proposed approach, in all the chosen scenarios. While for the Ziebart et al., we report the result taken from the corresponding paper.

HRI laboratory. This scenario simulates a small home environment in which points of interest are extracted by analysing the tracking data. Robot and people's position and velocity estimates are provided by the distributed tracker using two overhead cameras, facing opposite directions but with overlapping fields of view over the environment. In high density scenarios, the activity forecasting task is made challenging by the limited collision-free space in proportion to the physical size of the agents involved. Such a constraint could force an agent to dramatically change bearing to avoid a dynamic obstacle while still approaching one's target goal.

Informatics forum. We evaluate our approach according to its performance in real-time tracking and activity forecasting in a natural human environment. This is challenging due to numerous aspects, such as the presence of agents with changing intentions, or agents that are navigating with other latent constraints (e.g. maintaining a spatial formation with respect to other agents). For this scenario, possible goals have been conjectured by analysing the information provided by the tracking algorithm. Our results show that, our activity forecasting algorithm provides accurate beliefs over the possible set of goals.

VIRAT. The data set is designed to be realistic, natural and challenging for video surveillance domains in terms of its resolution, background clutter, diversity in scenes and human activity/event categories. In order to fairly compare our IRL approach against the state-of-the-art method of Ziebart et al., we choose goals as done in [1].

Discussion. In this work, we have access to trajectories of normal behaviours that are used for the initial generation of an MDP model through IRL, for each agent. Such a model describes the *preferred* paths of an agent moving toward a certain goal. The generated model is independent in terms of agent's velocity, hence a prediction of future agent motions, having an arbitrary velocity with respect to the observed one, is still possible by applying the IRL model. In the case of dramatic changes in the environment dynamics, the IRL model becomes essentially unusable due to this considerable variation in the structure of the environment. Therefore, an updated IRL model, taking into account these new changes, is needed before forecasting agent intentions. This suggests that the foregoing description is to be viewed as a template of a framework that can be further enhanced with lifelong and continual learning towards efficient activity forecasting for a practical social robot.

VI. Conclusions

We presented a novel framework for estimating the future movement intentions of goal-oriented agents in an interactive multi-agent setting. We achieve this by combining Inverse Reinforcement Learning with a Markov Decision Process model of motion, and a distributed multi-camera tracking algorithm. The resulting reward functions represent the agent's set of possible future activities, on which forecasts are made through a probability distribution function via comparison with real-time observed agent behaviour. This method is evaluated for accuracy and robustness in dense and dynamic environments with autonomously planning robots and pedestrians. Our results show that this is an effective and computationally efficient alternative to models that depend either on offline training of pedestrian trajectory models or on physical scene features and prior knowledge of goals.

References

[1] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, “Activity forecasting,” in *ECCV*. Springer, 2012, pp. 201–214.

[2] J. K. Aggarwal and M. S. Ryoo, “Human activity analysis: A review,” *ACM Computing Surveys*, vol. 43, no. 3, p. 16, 2011.

[3] J. C. Nascimento, M. A. Figueiredo, and J. S. Marques, “Trajectory classification using switched dynamical hidden markov models,” *Transactions on Image Processing*, vol. 19, no. 5, 2010.

[4] A. Jaimes and N. Sebe, “Multimodal human-computer interaction: A survey,” *CVIU*, vol. 108, no. 1, pp. 116–134, 2007.

[5] A. Ekin, A. M. Tekalp, and R. Mehrotra, “Automatic soccer video analysis and summarization,” *Transactions on Image Processing*, vol. 12, no. 7, pp. 796–807, 2003.

[6] C. G. Snoek and M. Worring, “Concept-based video retrieval,” *Foundations and Trends in Information Retrieval*, vol. 2, no. 4, 2008.

[7] D. L. Vail, M. M. Veloso, and J. D. Lafferty, “Conditional random fields for activity recognition,” in *AAMAS*. ACM, 2007, p. 235.

[8] L. Wang, Y. Qiao, and X. Tang, “Latent hierarchical model of temporal structure for complex activity classification,” *Transactions on Image Processing*, vol. 23, no. 2, pp. 810–822, 2014.

[9] L. Leal-Taixé, G. Pons-Moll, and B. Rosenhahn, “Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker,” in *ICCVW*, 2011, pp. 120–127.

[10] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, “Multiple object tracking using k-shortest paths optimization,” *Transactions on PAMI*, vol. 33, no. 9, pp. 1806–1819, 2011.

[11] P. K. Sharma, C. Huang, and R. Nevatia, “Evaluation of people tracking, counting and density estimation in crowded environments,” in *International Workshop on PETS*. IEEE, 2009, pp. 39–46.

[12] J. Yang, Z. Shi, P. Vela, and J. Teizer, “Probabilistic multiple people tracking through complex situations,” in *International Workshop on PETS*. IEEE, 2009, pp. 79–86.

[13] R. Kasturi *et al.*, “Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol,” *Transactions on PAMI*, vol. 31, no. 2, pp. 319–336, 2009.

[14] A. Andriyenko, S. Roth, and K. Schindler, “An analytical formulation of global occlusion reasoning for multi target tracking,” in *ICCVW*. IEEE, 2011, pp. 1839–1846.

[15] J. Zhang, L. L. Presti, and S. Sclaroff, “Online multi-person tracking by tracker hierarchy,” in *AVSS*. IEEE, 2012, pp. 379–385.

[16] A. Y. Ng and S. J. Russell, “Algorithms for inverse reinforcement learning,” in *ICML*, 2000, pp. 663–670.