Chapter 11
Pedestrian Interaction in Tracking: The Social Force Model and Global Optimization Methods

Laura Leal-Taixé and Bodo Rosenhahn

Abstract Multiple people tracking consists in detecting the subjects at each frame and matching these detections to obtain full trajectories. In semi-crowded environments, pedestrians often occlude each other, making tracking a challenging task. Tracking methods mostly work with the assumption that each pedestrian moves independently unaware of the objects or the other pedestrians around it. In the real world though, it is clear that when walking in a crowd, pedestrians try to avoid collisions, keep a close distance to a group of friends or avoid static obstacles in the scene. In this chapter, we present an overview of methods that include pedestrian interaction in a tracking framework. This interaction can be expressed in two ways: first, including social and grouping behavior as a physical model within the tracking system, and second, using a global optimization scheme which takes into account all trajectories and all frames to solve the data association problem.

11.1 Introduction

Multiple people tracking is a key problem for many computer vision tasks, such as surveillance, animation or activity recognition. In crowded environments occlusions and false detections are common, and although there have been substantial advances in the last years, tracking is still a challenging task. Tracking is often divided in two steps: detection, finding the objects of interest on every frame, and data association, matching the detections to form complete trajectories in time. Researchers have presented improvements on the object detector [9, 13, 34, 36] as well as on the optimization techniques [18, 23] and even specific algorithms have been developed

L. Leal-Taixé (✉) • B. Rosenhahn
Leibniz University Hannover, Appelstr. 9A, Hannover, Germany
e-mail: leal@tnt.uni-hannover.de; rosenhahn@tnt.uni-hannover.de

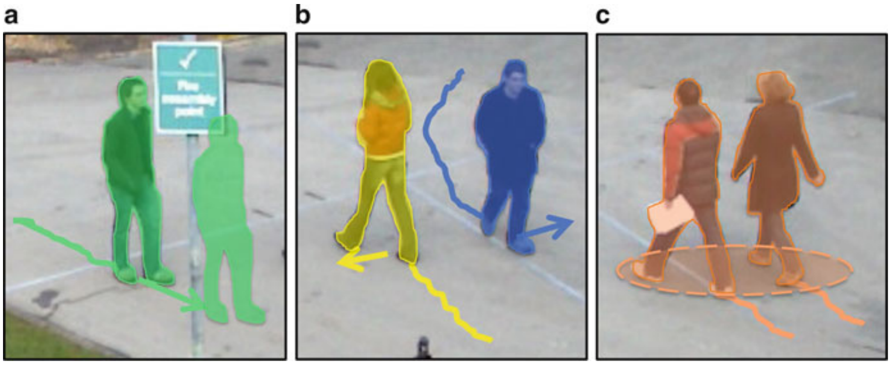

Fig. 11.1 Terms of the social force model. (**a**) Constant velocity assumption. (**b**) Avoidance forces. (**c**) Group attraction forces

for tracking in crowded scenes [2,32]. Though each object can be tracked separately, recent works have proven that tracking objects jointly and taking into consideration their interaction can give much better results in complex scenes. Current research is mainly focused on two aspects to exploit the interaction between pedestrians: the use of a global optimization strategy [7,21,40] and a social motion model [30,38]. The focus of this chapter is to give a detailed overview of multiple people trackers which include either a global optimization method or social behavior information to improve tracking results in crowded scenarios. Finally, the chapter discusses an approach to marry both concepts and include the social behaviors in a global optimization tracking system (Fig. 11.1).

11.1.1 Related Work

Current research is mainly focused on two aspects to exploit the interaction between pedestrians: the use of a global optimization strategy and a social motion model. In this section, we discuss both research trends.

Global Optimization: The optimization strategy deals with the data association problem, which is usually solved on a frame-by-frame basis or one track at a time. Several methods can be used such as Markov Chain Monte Carlo (MCMC) [19], multi-level Hungarian [20], inference in Bayesian networks [27] or the Nash Equilibrium of game theory [39]. In [6] an efficient approximative Dynamic Programming (DP) scheme is presented, in which trajectories are estimated one after the other. This means that if a trajectory is formed using a certain detection, the other trajectories which are computed later will not be able to use that detection anymore. This obviously does not guarantee a global optimum for all trajectories. Recent works show that global optimization can be more reliable in crowded scenes as it solves the matching problem jointly for all tracks. The multiple object tracking problem is defined as a linear constrained optimization flow problem and Linear Programming (LP) is commonly used to find the global optimum. The idea was first used for people tracking in [16], although this method needs to know a priori the number of targets to track, which limits its application in real tracking situations. In [7], the scene is divided into identical cells, each represented by a node in the constructed graph. Using the information of the Probability Occupancy Map, the problem is formulated either as a max-flow and solved with Simplex, or as a min-cost and solved using k-shortest paths, which is a more efficient solution. Both methods show a far superior performance when compared to the same approach with DP [6]. The authors of [3] also define the problem as a maximum flow on an hexagonal grid, but instead of using matching individual detections, they make use of tracklets. This has the advantage that they can precompute the social forces for each of these tracklets, nonetheless, the fact that the tracklets are chosen locally, means the overall matching is not truly global, and if errors occur during the creation of the tracklets, these cannot be overcome by the global optimization. In [40] the tracking problem is formulated as a Maximum A-Posteriori (MAP) problem, which is mapped to a minimum-cost network flow and then efficiently solved using LP. In this case, each node represents a detection, which means the graph is much smaller compared to [3, 7]. Finally, [37] propose to combine global and local methods to match trajectories across cameras and across time, while a unique global formulation for the multi-view multi-object is presented in [22].

Social Behavior for Tracking: Most tracking systems work with the assumption that the motion model for each target is independent. This simplifying assumption is especially problematic in crowded scenes: imagine the chaos if every pedestrian followed his or her chosen path and completely ignored the other pedestrians in the scene. In order to avoid collisions and reach the chosen destination at the same time, a pedestrian follows a series of social rules or social forces. These have been defined in what is called the Social Force Model (SFM) [15], which has been used for abnormal crowd behavior detection [26], crowd simulation [28] and has only recently been applied to multiple people tracking: in [33], an energy minimization approach is used to predict the future position of each pedestrian considering all the terms of the social force model. In [30] and [24], the social forces are included in the motion model of the Kalman or Extended Kalman filter, while the authors in [4] discuss the type of energy needed to include information about the dynamic model, repulsion, etc. and how to optimize it using the standard conjugate gradient method. In [14] a method is presented to detect small groups of people in a crowd, but it is only recently that grouping behavior has been included in a tracking framework [10, 29, 38]. In [29] groups are included in a graphical model which contains cycles and, therefore, Dual Decomposition [8] is needed to find the solution, which obviously is computationally much more expensive than using Linear Programming. Moreover, the results presented in [29] are only for short time windows. On the other hand, the formulations of [10, 38] are predictive by nature and therefore too local and unable to deal with trajectory changes (e.g. when people meet and stop to talk).

Recently, a new approach [21] includes social and grouping models into a global optimization framework, allowing for a better estimate of the true maximum a-posteriori probability of the trajectories and therefore further improving tracking results, especially in crowded scenes.

11.2 Multiple People Tracking

Tracking is commonly divided in two steps: object detection and data association. First, the objects are detected in each frame of the sequence and second, the detections are matched to form complete trajectories. In this section we define the data association problem and describe how to convert it to a minimum-cost network flow problem, which can be efficiently solved using Linear Programming.

The idea is to build a graph in which the nodes represent the pedestrian detections. These nodes are fully connected to past and future observations by edges, which determine the relation between two observations with a cost. Thereby, the matching problem is equivalent to a minimum-cost network flow problem: finding the optimal set of trajectories is equivalent to sending flow through the graph so as to minimize the cost. This can be efficiently computed using the Simplex algorithm or k-shortest paths [11].

11.2.1 Problem Statement

Let $\mathscr{O} = \{\mathbf{o}_k t\}$ be a set of object detections with $\mathbf{o}_k^t = (\mathbf{p}_k, t)$, where $\mathbf{p}_k = (x, y, z)$ is the 3D position and t is the time stamp. A trajectory is defined as a list of ordered object detections $T_k = \{\mathbf{o}_k^1, \mathbf{o}_k^2, \cdots, \mathbf{o}_k^N\}$, and the goal of multiple object tracking is to find the set of trajectories $\mathscr{T}* = \{T_k\}$ that best explains the detections.

This is equivalent to maximizing the a-posteriori probability of $\mathscr{T}$ given the set of detections $\mathscr{O}$, which is known as *maximum posterior* or *MAP* problem.

$$\mathscr{T}* = \underset{\mathscr{T}}{\mathbf{argmax}}\, P(\mathscr{T}|\mathscr{O}) \tag{11.1}$$

Further assuming that detections are conditionally independent, the objective function is expressed as:

$$\mathscr{T}* = \underset{\mathscr{T}}{\mathbf{argmax}}\, P(\mathscr{O}|\mathscr{T})P(\mathscr{T}) = \underset{\mathscr{T}}{\mathbf{argmax}} \prod_k P(\mathbf{o}_k|\mathscr{T})P(\mathscr{T}) \tag{11.2}$$

$P(\mathbf{o}_k|\mathscr{T})$ is the likelihood of the detection. Optimizing Eq. (11.2) directly is intractable since the space of $\mathscr{T}$ is huge, nonetheless we make the assumption that the trajectories cannot overlap (i.e., a detection cannot belong to two trajectories) to obtain:

$$\mathscr{T}* = \underset{\mathscr{T}}{\mathbf{argmax}} \prod_k P(\mathbf{o}_k|\mathscr{T}) \prod_{T_k \in \mathscr{T}} P(T_k) \tag{11.3}$$

where the trajectories are represented by a Markov chain:

$$P(\mathscr{T}) = \prod_{T_k \in \mathscr{T}} P_{\text{in}}(\mathbf{o}_k^1) P(\mathbf{o}_k^2|\mathbf{o}_k^1) \dots P(\mathbf{o}_k^t|\mathbf{o}_k^{t-1}) \dots P(\mathbf{o}_k^N|\mathbf{o}_k^{N-1}) P_{\text{out}}(\mathbf{o}_k^N) \tag{11.4}$$

where $P_{\text{in}}(\mathbf{o}_k^t)$ is the probability that a trajectory is initiated with detection $\mathbf{o}_k^t$, $P_{\text{out}}(\mathbf{o}_k^t)$ the probability that the trajectory is terminated at $\mathbf{o}_k^t$ and $P(\mathbf{o}_k^t|\mathbf{o}_k^{t-1})$ is the probability that $\mathbf{o}_k^{t-1}$ is followed by $\mathbf{o}_k^t$ in the trajectory.

11.2.2 *Tracking with Linear Programming*

In this section, we explain how to convert the MAP problem into a Linear Program, which is a particularly interesting since it can be efficiently solved in polynomial time using any of the available techniques from the optimization community [1].

A linear programming problem consists in minimizing or maximizing a linear function in the presence of linear constraints, which can be both equalities and inequalities.

$$\text{Minimize} \qquad c_1 f_1 + c_2 f_2 + \ldots + c_n f_n \tag{11.5}$$

$$\begin{aligned} \text{Subject to} \qquad & a_{11} f_1 + a_{12} f_2 + \ldots + a_{1n} f_n \geq b_1 \\ & a_{21} f_1 + a_{22} f_2 + \ldots + a_{2n} f_n \geq b_2 \\ & \vdots \qquad \vdots \qquad \qquad \vdots \\ & a_{m1} f_1 + a_{m2} f_2 + \ldots + a_{mn} f_n \geq b_m \end{aligned} \tag{11.6}$$

where Eq. (11.5) is the *objective function* and Eq. (11.6) are the *constraints*.

(continued)

(continued)
$c_1, c_2, \ldots, c_n$ denote the known *cost coefficients* and $f_1, f_2, \ldots, f_n$ are the *decision variables* to be determined.

To convert our problem into a linear program, we linearize the objective function by defining a set of flow flags $f_{i,j} = \{0, 1\}$ which indicate if an edge (i, j) is in the path of a trajectory or not.

In a minimum cost network flow problem, the objective is to find the values of the variables that minimize the total cost of the flows over the network. Defining the costs as negative log-likelihoods, and combining Eqs. (11.3) and (11.4), the following objective function is obtained:

$$\begin{aligned} \mathcal{T}* &= \mathop{\mathbf{argmin}}_{\mathcal{T}} \sum_{T_k \in \mathcal{T}} -\log P(T_k) - \sum_k \log P(\mathbf{o}_k|\mathcal{T}) \\ &= \mathop{\mathbf{argmin}}_{\mathcal{T}} \sum_i C_{\text{in},i} f_{\text{in},i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_i f_i + \sum_i C_{i,\text{out}} f_{i,\text{out}} \end{aligned} \tag{11.7}$$

subject to the following constraints:

- Edge capacities: assuming each detection can only correspond to one trajectory, the edge capacities have an upper bound of $u_{ij} \leq 1$ and:

$$f_{\text{in},i} + f_i \leq 1 \qquad f_{i,\text{out}} + f_i \leq 1 \tag{11.8}$$

- Flow conservation at the nodes:

$$f_{\text{in},i} + f_i = \sum_j f_{i,j} \qquad \sum_j f_{j,i} = f_{i,\text{out}} + f_i \tag{11.9}$$

- Exclusion property:

$$f_{i,j} = \{0, 1\} \tag{11.10}$$

The condition in Eq. (11.10) requires us to solve an integer program, which is known to be NP-complete. Nonetheless, we can relax the condition to have the following linear equation:

$$0 \leq f_{i,j} \leq 1. \tag{11.11}$$

Now the problem is defined and can be solved as a linear program. If certain conditions are fulfilled, the solution $\mathcal{T}*$ will still be integer, and therefore will also be the optimal solution to the initial integer program. We discuss the integrality of the solution in more detail in Sect. 11.4.

To map this formulation into a cost-flow network, we define $G = (N, E)$ to be a directed network with a cost $C_{i,j}$ and a capacity u_{ij} associated with every edge

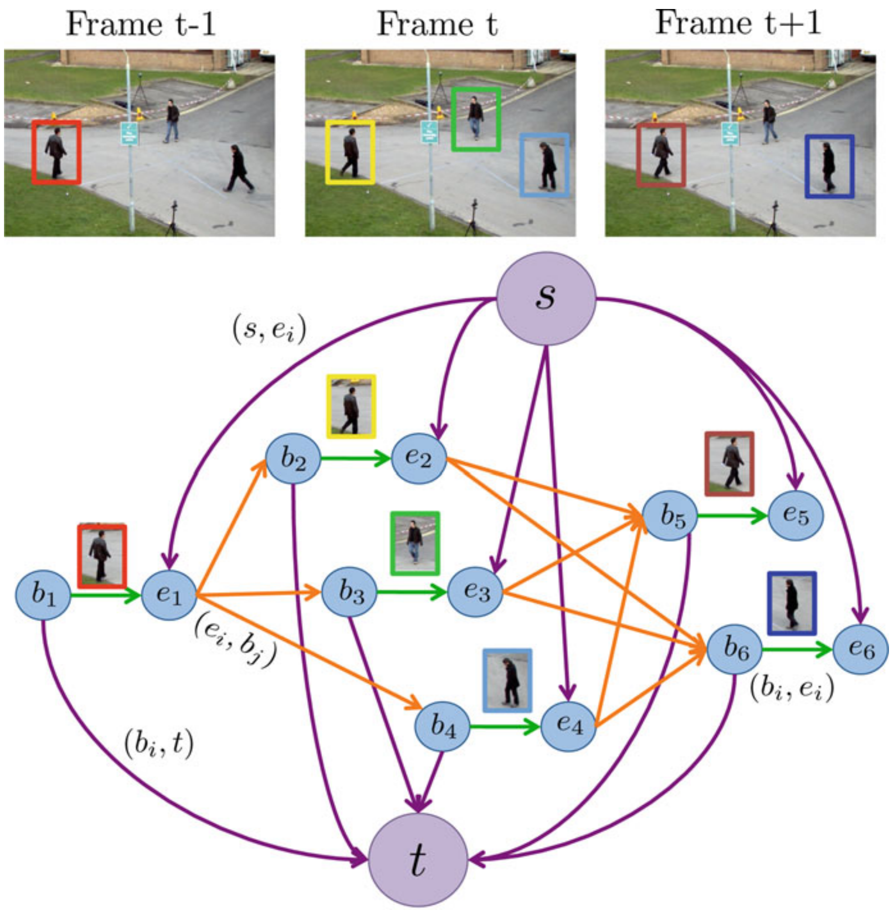

Fig. 11.2 Example of a graph with the special source s and sink t nodes, six detections which are represented by two nodes each: the beginning b_i and the end e_i

$(i, j) \in E$. An example of such a network is shown in Fig. 11.2; it contains two special nodes, the source s and the sink t; all flow that goes through the graph starts at the s node and ends at the t node. Thereby, each flow represents a trajectory T_k and the path that each flow follows indicates which observations belong to each of the trajectories. Each observation $\mathbf{o}_i$ is represented with two nodes, the beginning node $b_i \in N$ and the end node $e_i \in N$ (see Fig. 11.2). A detection edge connects b_i and e_i.

Below we detail the three types of edges present in the graphical model and the cost for each type:

Link Edges: The edges (e_i, b_j) connect the end nodes e_i with the beginning nodes b_j in following frames, with cost $C_{i,j}$ and flow $f_{i,j}$, defined as:

$$f_{i,j} = \begin{cases} 1, & \mathbf{o}_i \text{ and } \mathbf{o}_j \text{ belong to } T_k \text{ and } \Delta f \leq F_{\max} \\ 0, & \text{otherwise} \end{cases} \tag{11.12}$$

where Δf is the frame number difference between nodes j and i and $F_{\max}$ is the maximum allowed frame gap.

The costs of the link edges represent the spatial relation between different subjects. Assuming that a subject cannot move a lot from one frame to the next, we define the costs to be a decreasing function of the distance between detections in successive frames. The time gap between observations is also taken into account in order to be able to work at any frame rate, therefore velocity measures are used instead of distances. The velocities are mapped to probabilities with a Gauss error function as shown in Eq. (11.13), assuming the pedestrians cannot exceed a maximum velocity $V_{\max}$. The effect of parameter $V_{\max}$ is detailed in Sect. 11.5.1.

$$E(V_t, V_{\max}) = \frac{1}{2} + \frac{1}{2}\operatorname{erf}\left(\frac{-V_t + \frac{V_{\max}}{2}}{\frac{V_{\max}}{4}}\right) \tag{11.13}$$

As we can see in Fig. 11.3, the advantage of using Eq. (11.13) over a linear function is that the probability of lower velocities decreases more slowly, while the probability for higher velocities decreases more rapidly. This is consistent with the probability distribution of speed learned from training data.

Therefore, the cost of a link edge is defined as:

$$\begin{aligned} C_{i,j} &= -\log\left(P(\mathbf{o}_j|\mathbf{o}_i)\right) + C(\Delta f) \qquad (11.14) \\ &= -\log E\left(\frac{\|\mathbf{p}_j - \mathbf{p}_i)\|}{\Delta t}, V_{\max}\right) + C(\Delta f) \end{aligned}$$

where $C(\Delta f) = -\log\left(B_j^{\Delta f - 1}\right)$ is the cost depending on the frame difference between detections.

Detection Edges: The edges (b_i, e_i) connect the beginning node b_i and end node e_i, with cost C_i and flow f_i, defined as:

$$f_i = \begin{cases} 1, & \mathbf{o}_i \text{ belongs to } T_k \\ 0, & \text{otherwise} \end{cases} \tag{11.15}$$

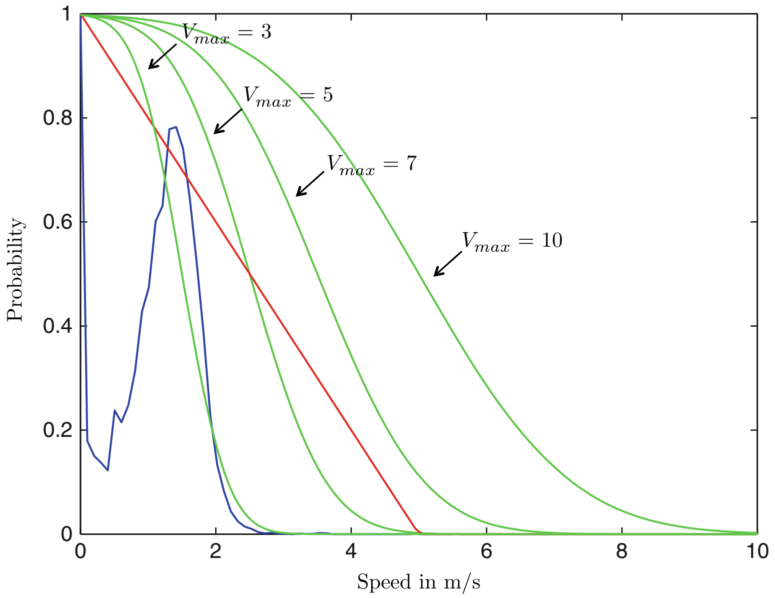

Fig. 11.3 (Color online) *Blue* = normalized histogram of speeds learned from training data. *Red* = probability distribution if cost depends linearly on the velocity. *Green* = probability distribution if the relation of cost and velocities is expressed by Eq. (11.13). An $V_{\max} = 7\,$m/s is used in the experiments

If all the costs of the edges are positive, the solution to the minimum-cost problem is the trivial null flow. Consequently, we represent each observation with two nodes and a detection edge with negative cost:

$$C_i = \log\left(1 - P_{det}(\mathbf{o}_i)\right) + \log\left(\frac{\mathrm{BB}_{\min}}{\|\mathbf{p}_{\mathrm{BB}} - \mathbf{p}_i)\|}\right). \tag{11.16}$$

The higher the likelihood of a detection $P_{det}(\mathbf{o}_i)$ the more negative the cost of the detection edge, hence, confident detections are likely to be in the path of the flow in order to minimize the total cost. If a map of the scene is available, we can also include this information in the detection cost. If a detection is far away from a possible entry/exit point, we add an extra negative cost to the detection edge, in order to favor that observation to be matched. The added cost depends on the distance to the closest entry/exit point $\mathbf{p}_{\mathrm{BB}}$, and is only computed for distances higher than $\mathrm{BB}_{\min} = 1.5m$. This is a probabilistic simple way of including other information present in the scene, such as obstacles or attraction points (shops, doors, etc.).

Entrance and Exit Edges: The edges (s, e_i) connect the source s with all the end nodes e_i, with cost $C_{\mathrm{in},i}$ and flow $f_{\mathrm{in},i}$. Similarly, (b_i, t) connects the end node b_i with sink t, with cost $C_{i,\mathrm{out}}$ and flow $f_{i,\mathrm{out}}$. The flows are defined as:

$$f_{\mathrm{in},i}\ (\text{or } f_{i,\mathrm{out}}) = \begin{cases} 1, & T_k \text{ starts (or ends) at } \mathbf{o}_i \\ 0, & \text{otherwise} \end{cases} \tag{11.17}$$

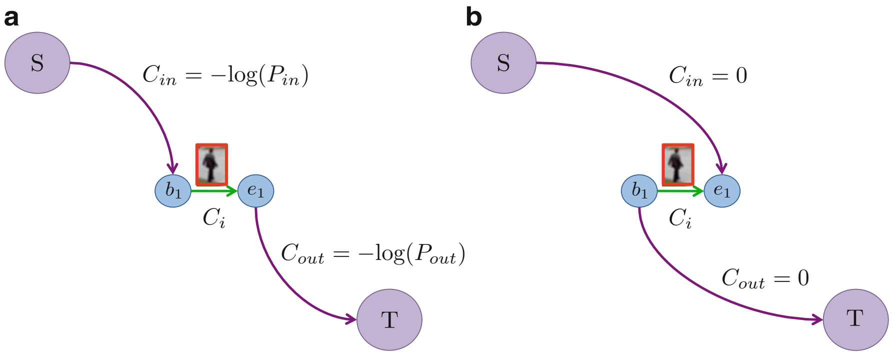

Fig. 11.4 (**a**) Graph structure as used in [40], which requires the computation of P_{in} and P_{out} in an Expectation-Maximization step during optimization. (**b**) Graph structure as used in [21] which does not require the computation of these two parameters; the trajectories are found only with the information of the link and detection edges

This connection, as shown in Fig. 11.4b, was proposed in [21] so that when a track starts (or ends) it does not benefit from the negative cost of the detection edge. Setting $C_{\text{in}} = C_{\text{out}} = 0$ and taking into account the flow constraints of Eqs. (11.8) and (11.9), the trajectories are only created with the information of the link edges.

In contrast, the authors in [40] propose to create the opposite edges (s, b_i) and (e_i, t), which means tracks entering and leaving the scene go through the detection node and therefore benefiting from its negative cost (see Fig. 11.4a). If the costs C_{in} and C_{out} are then set to zero, a track will be started at each detection of each frame, because it will be cheaper to use the entrance and exit edges than the link edges. On the other hand, if C_{in} and C_{out} are very high, it will be hard for the graph to create any trajectory. Therefore, the choice of these two costs is extremely important. In [40], the costs are set according to the entrance and exit probabilities P_{in} and P_{out}, which are data dependent terms that need to be calculated during optimization.

11.3 Modeling Social Behavior

If a pedestrian does not encounter any obstacles, the natural path to follow is a straight line. But what happens when the space gets more and more crowded and the pedestrian can no longer follow the straight path? Social interaction between pedestrians is especially important when the environment is crowded. In this section we consider how to include the social behavior [15], which we divide into the Social Force Model (SFM) and the Group behavior (GR), into the minimum-cost network flow problem.

11.3.1 New MAP and Linear Programming Formulation

The original social force model [15] describes a physical system that predicts the position of a pedestrian in a continuous way, which has been successfully used for crowd simulation [28]. Nonetheless, we use the social information within another paradigm: in our Linear Programming system, we have a set of hypothetical pedestrian positions (in the form of nodes) and we apply the social forces to find out the probability of a certain match (i.e. a certain trajectory being followed by a pedestrian).

When including social and grouping information in the Linear Programming formulation, we can no longer assume that the motion of each subject is independent, which means we have to deal with a much larger search space of $\mathscr{T}$.

We extend this space by including the following dependencies for each trajectory T_k:

- Constant velocity assumption: the observation $\mathbf{o}_k^t \in T_k$ depends on past observations $[\mathbf{o}_k^{t-1}, \mathbf{o}_k^{t-2}]$
- Grouping behavior: If T_k belongs to a group, the set of members of the group $\mathscr{T}_{k,\mathrm{GR}}$ has an influence on T_k
- Avoidance term: T_k is affected by the set of trajectories $\mathscr{T}_{k,\mathrm{SFM}}$ which are close to T_k at some point in time and do not belong to the same group as T_k

The first and third dependencies are grouped into the SFM term. The sets $\mathscr{T}_{k,\mathrm{SFM}}$ and $\mathscr{T}_{k,\mathrm{GR}}$ are disjoint, i.e., for a certain pedestrian k, the set of pedestrians that have an attractive effect (the group to which pedestrian k belongs to), is different from the pedestrians that have a repulsive effect on k. Therefore, we can assume that these two terms are independent and decompose $P(\mathscr{T})$ as:

$$\begin{aligned} P(\mathscr{T}) &= \prod_{T_k \in \mathscr{T}} P(T_k \cap \mathscr{T}_{k,\mathrm{SFM}} \cap \mathscr{T}_{k,\mathrm{GR}}) \\ &= \prod_{T_k \in \mathscr{T}} P(\mathscr{T}_{k,\mathrm{SFM}} | T_k) P(\mathscr{T}_{k,\mathrm{GR}} | T_k) P(T_k) \end{aligned} \tag{11.18}$$

Let us assume that we are analyzing observation $\mathbf{o}_k^t$. In Fig. 11.5 we summarize which observations influence the matching of $\mathbf{o}_k^t$. Typical approaches [40] only take into account distance (DIST) information, that is, the observation in the previous frame $\mathbf{o}_k^{t-1}$. We introduce the social dependencies (SFM) given by the constant velocity assumption (green nodes) and the avoidance term (yellow nodes). In this case, two observations, $\mathbf{o}_q^t$ and $\mathbf{o}_r^t$ that do not belong to the same group as $\mathbf{o}_k^t$, will be considered to create a repulsion effect on $\mathbf{o}_k^t$. On the other hand, the orange nodes which depict the grouping term (GR), are two other observations $\mathbf{o}_m^t$ and $\mathbf{o}_n^t$ which

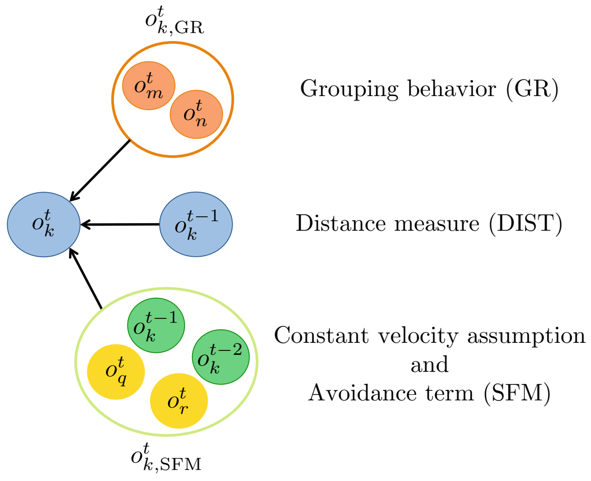

Fig. 11.5 Diagram of the dependencies for each observation $\mathbf{o}^t_k$

do belong to the same group as $\mathbf{o}^t_k$ and therefore have an attraction effect on $\mathbf{o}^t_k$. Note that all these dependencies can only be modeled by high order terms, which means that either we use complex solvers [29] to find a solution in graphs with cycles, or we keep the linearity of the problem by using an iterative approach as we explain later on.

The objective function is accordingly updated:

$$\begin{aligned}\mathscr{T}* &= \underset{\mathscr{T}}{\mathbf{argmin}} \sum_{T_k \in \mathscr{T}} -\log P(T_k) - \log P(\mathscr{T}_{\mathrm{SFM}}|T_k) \\ &\quad - \log P(\mathscr{T}_{\mathrm{GR}}|T_k) + \sum_k -\log P(\mathbf{o}_k|\mathscr{T}) \\ &= \underset{\mathscr{T}}{\mathbf{argmin}} \sum_i C_{\mathrm{in},i} f_{\mathrm{in},i} + \sum_i C_{i,\mathrm{out}} f_{i,\mathrm{out}} \\ &\quad + \sum_{i,j} (C_{i,j} + C_{\mathrm{SFM},i,j} + C_{\mathrm{GR},i,j}) f_{i,j} + \sum_i C_i f_i\end{aligned} \tag{11.19}$$

11.3.2 Social Force Model

The social force model states that the motion of a pedestrian can be described as if they were subject to "social forces". There are three main terms that need to be considered: the desire of a pedestrian to maintain a certain speed, the desire to keep a comfortable distance from other pedestrians and the desire to reach a destination. Since we cannot know a priori the destination of the pedestrian in a real tracking system, we focus on the first two terms.

Constant Velocity Assumption: The pedestrian tries to keep a certain speed and direction, therefore we assume that in $t+\Delta t$ we have the same speed as in t and predict the pedestrian's position in $t+\Delta t$ accordingly.

$$\tilde{\mathbf{p}}_i^{t+\Delta t} = \mathbf{p}_i^t + \mathbf{v}_i^t \Delta t$$

Avoidance Term: The pedestrian also tries to avoid collisions and keep a comfortable distance from other pedestrians. This term is modeled as a repulsion field with an exponential distance-decay function with value α learned from training data.

$$\mathbf{a}_i^{t+\Delta t} = \sum_{g_m \neq g_i} \exp\left(-\frac{\|\tilde{\mathbf{p}}_i^{t+\Delta t} - \tilde{\mathbf{p}}_m^{t+\Delta t}\|}{\alpha \Delta t}\right) \tag{11.20}$$

To compute the cost of edge (i, j), the constant velocity assumption is used to predict the position of $\mathbf{o}_i$ and $\mathbf{o}_j$ as well as the rest of pedestrians $\tilde{\mathbf{p}}_m^{t+\Delta t}$, and the repulsion acceleration each pedestrian has on i is also taken into account. The only pedestrians that have this repulsion effect on subject i are the ones which do not belong to the same group as i and $\|\tilde{\mathbf{p}}_i^{t+\Delta t} - \tilde{\mathbf{p}}_m^{t+\Delta t}\| \leq 1m$. The different avoidance terms are combined linearly.

Now the prediction of the pedestrian's next position is also influenced by the avoidance term (acceleration) from all pedestrians:

$$\tilde{\mathbf{p}}_i^{t+\Delta t} = \mathbf{p}_i^t + (\mathbf{v}_i^t + \mathbf{a}_i^{t+\Delta t} \Delta t)\Delta t \tag{11.21}$$

The distance between prediction and real measurements is used to compute the cost:

$$C_{\text{SFM},i,j} = -\log E\left(\frac{\|\tilde{\mathbf{p}}_i^{t+\Delta t} - \mathbf{p}_j^{t+\Delta t}\|}{\Delta t}, V_{\max}\right) \tag{11.22}$$

where the function E is detailed in Eq. (11.13).

In Fig. 11.6 we plot the probability distribution computed using different terms. Note, this is just for visualization purposes, since we do not compute the probability for each point on the scene, but only for the positions where the detector has fired. There are four pedestrians in the scene, the purple one and three green ones walking in a group. As shown in Fig. 11.6b, if we only use the predicted positions (yellow heads) given the previous speeds, there is a collision between the purple pedestrian and the green marked with a 1 collide. The avoidance term shifts the probability mode to a more plausible position.

11.3.3 Group Model

The social behavior [15] also includes an attraction force which occurs when a pedestrian is attracted to a friend, shop, etc. In this section, we show how to model

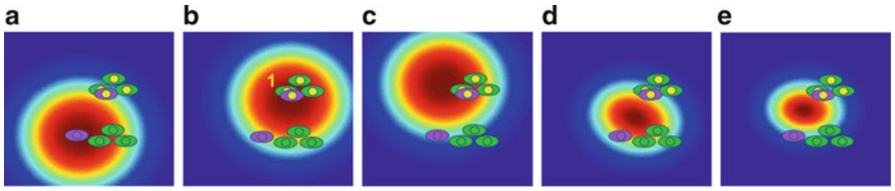

Fig. 11.6 (Color online) Three green pedestrians walk in a group, the predicted positions in the next frame are marked by *yellow heads*. The purple pedestrian's linearly predicted position (*yellow head*) clearly interferes with the trajectory of the group. Representation of the probability (blue is 0 red is 1) distribution for the purple's next position using: (**a**) only distances, (**b**) only SFM (constant velocity assumption and avoidance term), (**c**) only GR (considering the purple pedestrian belongs to the group), (**d**) distances+SFM and (**e**) distances+SFM+GR

the attraction between members of a group. Before modeling group behavior we need to determine which tracks form each group and at which frame the group begins and ends (to deal with splitting and formation of groups). The idea is that if two pedestrians are close to each other over a reasonable period of time, they are likely to belong to the same group. From the training sequence in [30], the distance and speed probability distributions of the members of a group P_g vs. individual pedestrians P_i is learned. If m and n are two trajectories which appear on the scene at $t = [0, N]$, we compute the flag $G_{m,n}$ that indicates if m and n belong to the same group.

$$G_{m,n} = \begin{cases} 1, & \sum_{t=0}^{N} P_g(m,n) > \sum_{t=0}^{N} P_i(m,n) \\ 0, & \text{otherwise} \end{cases} \tag{11.23}$$

For every observation $\mathbf{o}_i$, we define a group label g_i which indicates to which group the observation belongs to, if any. If several pedestrians form a group, they tend to keep a similar speed, therefore, if i belongs to a group, we can use the mean speed of all the other members of the group to predict the next position for i:

$$\tilde{\mathbf{p}}_i^{t+\Delta t} = \mathbf{p}_i^t + \sum_{g_m = g_i} \mathbf{v}_m^t \Delta t \tag{11.24}$$

The distance between this predicted position and the real measurements is used in (11.13) to obtain the cost for the grouping term.

An example is shown in Fig. 11.6c, where we can see that the maximum probability provided by the group term keeps the group configuration. In Fig. 11.6d we show the combined probability of the distance and SFM information, which narrows the space of probable positions. Finally, Fig. 11.6e represents the combined probability of DIST, SFM and GR. As we can see, the space of possible locations for the purple pedestrian is considerably reduced as we add the social and grouping behaviors, which means we have less ambiguities for data association. This is specially useful to decrease identity switches as we present in Sect. 11.5.

11.4 Optimization

To compute the SFM and grouping costs, we need to have information about the velocities of the pedestrians, which can only be obtained if we already have the trajectories. This chicken-and-egg problem is solved iteratively as shown in Algorithm 3; on the first iteration, the trajectories are estimated only with the information defined in Sect. 11.2.2, for the rest of iterations, the SFM and GR is also used. The algorithm stops when the trajectories do not change or when a maximum number of iterations M_i is reached.

Algorithm 1 Iterative optimization

```
while $\mathscr{T}_i \neq \mathscr{T}_{i-1}$ and $i \leq M_i$ do
    if $i == 1$ then
        1.1. Create the graph using only DIST information
    else
        1.2. Create the graph using DIST, SFM and GR information
    end if
    2. Solve the graph to find $\mathscr{T}_i$
    3. Compute velocities and groups given $\mathscr{T}_i$
end while
```

11.4.1 Linear Programming Solvers

The minimum cost solution is found using the Simplex algorithm [11], with the implementation given in [25]. Though Simplex has an exponential worst-case complexity, most sequences can be tracked in just a few seconds; this is because each node represents one detection, and therefore the dimension of the graph is quite small. For larger graphs [7] or more crowded environments, we can use the k-shortest paths solver [7, 31] which has a worst case complexity of $O(k(m+n\cdot\log(n)))$. For more details on network flows and Simplex we refer the reader to [1], and to [35] for more information on the k-shortest path algorithm.

11.4.2 Integrality of the Solution

When defining the program to be solved, we saw that Eq. (11.10) defined an integer program, which is known to be NP-complete. The condition is relaxed into Eq. (11.11) in order to use efficient Linear Programming solvers to find the optimum solution to our problem. If the solution to the relaxed version of the program is integer, then we know it is an optimal solution of the original problem [1]. The question is, can we guarantee that the solution will be always integer?

Let us assume the conditions of the Linear Program are expressed as: $Ax = b$. If all entries of A and b are integer, as it is our case, we can determine that $Ax = b$ has an integer solution by Cramer's rule:

$$Ax = b \quad \Leftrightarrow \quad x = A^{-1}b \quad \Leftrightarrow \quad \forall i : x_i = \frac{\det(A^i)}{\det(A)} \tag{11.25}$$

where A^i is equal to A except on the i-th column where it is equal to b. From here, we can determine that x will be integer when $\det(A)$ is equal to +1 or -1. A matrix $A \in Z^{m\times n}$ is *totally unimodular* if the determinant of all the subsquare matrices of A is either 0, +1 or -1.

Theorem 11.1. *If A is totally unimodular, every vertex solution of $Ax \leq b$ is integer.*

A well-known case of totally unimodular matrices are the node arc incidence matrices N of a directed network. Therefore, our defined constraint matrix is totally unimodular, and the solutions we will obtain will always be integer.

11.4.3 Computationally Reduction

To reduce the computational cost, the graph can be pruned using the physical constraints represented by the edge costs. If any of the costs C_{ij}, $C_{\mathrm{SFM},i,j}$ or $C_{\mathrm{GR},i,j}$ is infinite, the two detections i and j are either two far away to belong to the same trajectory or they do not match according to social and grouping rules, therefore the edge (i, j) is erased from the graphical model. For long sequences, the video can be divided into several batches and optimize for each batch. For temporal consistency, the batches have an overlap of $F_{\max} = 10$ frames. The runtime of [21] for a sequence of 800 frames (114 s), 4,837 detections, batches of 100 frames and 6 iterations is 30 s on a 3 GHz machine.

11.5 Experimental Results

In this section we show the tracking results of several state-of-the-art methods on three publicly available datasets and compare them using the CLEAR metrics [17], which split the measuring scores into *accuracy* and *precision*:

- Detection Accuracy (DA): measures how many detections where correctly found and therefore is based on the count of missed detections m_t and false alarms f_t for each frame t.

$$DA = 1 - \frac{\sum_{t=1}^{N_f} m_t + f_t}{\sum_{t=1}^{N_f} N_G^t}$$

where N_f is the number of frames of the sequence and N_G^t is the number of ground truth detections in frame t. A detection is considered to be correct when it is found within 50 pixels from the ground truth and the bounding boxes of both ground truth and detection have some overlap.

- Tracking Accuracy (TA): similar to DA but also including the identity switches i_t. In this case, the measure does not penalize identity switches as much as a missing detection or a false alarm as we use a $\log_{10}$ weight.

$$DA = 1 - \frac{\sum_{t=1}^{N_f} m_t + f_t + \log_{10}(1 + i_t)}{\sum_{t=1}^{N_f} N_G^t}$$

- Detection Precision (DP): precision measurements represent how well the bounding box detections match the ground truth. For this, an overlap measure between bounding boxes is used:

$$Ov^t = \sum_{i=1}^{N_{\text{mapped}}^t} \frac{|G_i^t \cap D_i^t|}{|G_i^t \cup D_i^t|}$$

where N_{mapped}^t is the number of mapped objects in frame t, i.e., the number of detections that are matched to some ground truth object. G_i^t is the ith ground truth object of frame t and D_i^t the detected object matched to G_i^t. The DP measure is then expressed as:

$$DP = \frac{\sum_{t=1}^{N_f} \frac{Ov^t}{N_{\text{mapped}}^t}}{N_f}$$

- Tracking Precision (TP): measures the spatiotemporal overlap between ground truth trajectories and detected ones, taking into account also split and merged trajectories.

$$TP = \frac{\sum_{i=1}^{N_{\text{mapped}}^t} \sum_{t=1}^{N_f} \frac{|G_i^t \cap D_i^t|}{|G_i^t \cup D_i^t|}}{\sum_{t=1}^{N_f} N_{\text{mapped}}^t}$$

11.5.1 Analysis of the Effect of the Parameters

All parameters defined in previous sections are learned from training data using one sequence of the publicly available dataset [30]. In this section we study the effect of the few parameters needed in [21], and show the method works well for a wide range of these parameters and therefore no parameter tuning is needed to obtain a good performance. The analysis is done on two publicly available datasets: a crowded town center [5] and the well-known PETS2009 dataset [12], to see the different effects of each parameters on each dataset.

11.5.2 Number of Iterations

The first parameter we analyze is the number of iterations M_i allowed. This determines how many times the loop between computing social forces and computing trajectories is performed as explained in Algorithm 1. Looking at the results on the PETS 2009 dataset in Fig. 11.7b, we can see that after just two iterations the results remain very stable. Actually, the algorithm reports no changes in the trajectories after three iterations, and therefore stops even though the maximum number of iterations allowed is higher. The result with one and two iterations is also not very different, which means the social and grouping behavior do not significantly improve the results for this particular dataset. This is due to the fact that this dataset is very challenging from a social behavior point of view, with subjects often changing direction and groups forming and splitting frequently. More details and comments on these results can be found in Sect. 11.5.6.2. On the other hand, we observe a different effect on the TownCenter dataset, shown in Fig. 11.7a. In this case, there is a clear improvement when using social and grouping behavior (i.e. the result improves when we use more than one iteration. We also observe a pattern on how the Tracking Accuracy of the dataset evolves: there is a cycle of three iterations for which the accuracy increases and decreases in a similar pattern. This means that the algorithm is jumping between two solutions and will not converge to neither one of them. This happens when pedestrians are close together for a long period of time but are not forming a group, which means that even with social forces, it is hard to say which paths they will follow.

11.5.3 Maximum Speed

This is the parameter that determines the maximum speed of the pedestrians that we are observing. In this case, we can see in Fig. 11.7c, d a clear trend in which the results are very bad when we force the pedestrians to walk more slowly that they actually do, since we are artificially splitting trajectories. The results converge when

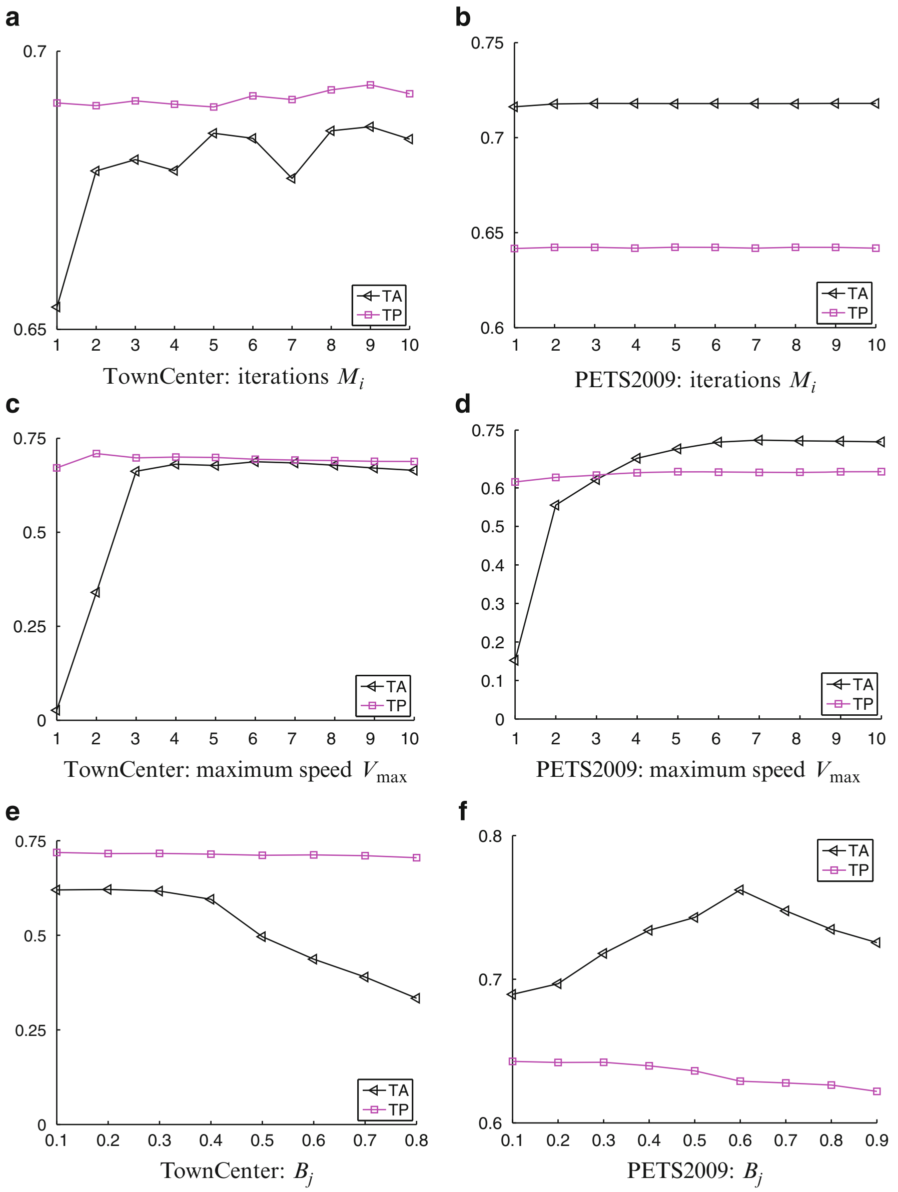

Fig. 11.7 (Color online) Tracking accuracy (*black*) and precision (*magenta*) obtained for the Town Center dataset (*left column*) and the PETS 2009 dataset (*right column*) given varying parameter values. (**a**) TownCenter: iterations M_i. (**b**) PETS2009: iterations M_i. (**c**) TownCenter: maximum speed $V_{\max}$. (**d**) PETS2009: maximum speed $V_{\max}$. (**e**) TownCenter: B_j $V_{\max}$. (**f**) PETS2009: B_j

the maximum speed allowed is around 3–5 m/s, which is the reported mean speed of pedestrians in a normal situation. More interestingly, we observe that the results are kept constant when using higher maximum speed values. This is a positive effect of the global optimization framework, since we can use a much higher speed limit and this will still give us good results and will allow us to track a person running through the scene, a case of panic when people start running, etc.

11.5.4 Cost for the Frame Difference

The last parameter, B_j, appears in Eq. (11.15) and represents the penalization term that we apply when the frame difference between two detections that we want to match is larger than 1. This term is used in order to give preference to matches that are close in time. Here we can again see different effects on the two datasets. In Fig. 11.7e, we see that the results are stable until a value of 0.4. The lower the value, the higher is the penalization cost for the frame difference, which means it is more difficult to match those detections which are more than 1 frame apart. When the value of B_j is higher than 0.4, there are more ambiguities in the data association process because it is easier to match detections which are many frames apart. In the TownCenter dataset, there is no occluding object in the scene, which means missing detections are sporadic within a given trajectory. In this scenario, a lower value for B_j is better, since small gaps can be filled and there are less ambiguities. Nonetheless, we see different results in the PETS 2009 dataset in Fig. 11.7f, since here there is a clear occluding object in the middle of the scene (see Fig. 11.8) which occludes the pedestrians for longer periods of time. In this case, a higher value of B_j allows to overcome these large gaps of missing data, and that is why the best value for this dataset is around 0.6.

11.5.5 Evaluation with Missing Data, Noise and Outliers

We evaluate the impact of every component of the approach in [21] with one of the sequences of the dataset [30], which contains images from a crowded public place, with several groups as well as walking and standing pedestrians. The sequence is 11,601 frames long and contains more than 300 trajectories. First of all, the group detection method is evaluated on the whole sequence with ground truth detections: 61% are correctly detected, 26% are only partially detected, 13% are not found and an extra 7% groups are detected wrongly. All experiments are performed with 6 iterations, a batch of 100 frames, $V_{\max} = 7\,\text{m/s}$, $F_{\max} = 10$, $\alpha = 0.5$ and $B_j = 0.3$.

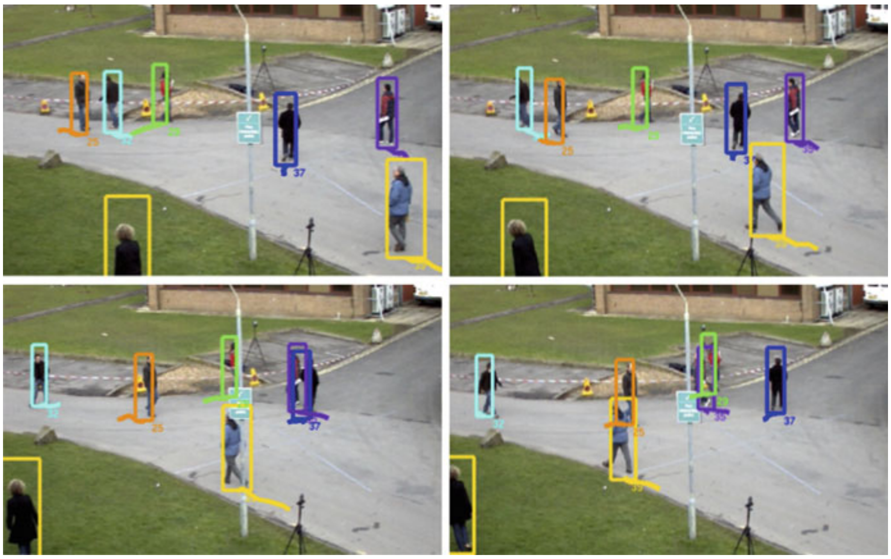

Fig. 11.8 Four frames of the PETS2009 sequence (separation of nine frames), showing several occlusions, both created by the obstacle on the scene and between pedestrians. All the occlusions can be recovered with the proposed method

Using the ground truth (GT) pedestrian positions as the baseline for our experiments, we perform three types of tests, missing data, outliers and noise, and compare the results obtained with:

- DIST: proposed network model with distances
- SFM: adding the Social Force Model (Sect. 11.3.2)
- SFM+GR: adding SFM and grouping behavior (Sect. 11.3.3)

Missing Data: This experiment shows the robustness of our approach given missed detections. This is evaluated by randomly erasing a certain percentage of detections from the GT set. The percentages evaluated are $[0, 4, 8, 12, 16, 20]$ from the total number of detections over the whole sequence. As we can see in Fig. 11.9, both SFM and SFM+GR increase the tracking accuracy when compared to DIST.

Outliers: With an initial set of detections of GT with 2% missing data, tests are performed with $[0, 10, 20, 30, 40, 50]$ percentage of outliers added in random positions over the ground plane. In Fig. 11.9, the results show that the SFM is especially important when the tracker is dealing with outliers. With 50% of outliers, the identity switches with SFM+GR are reduced 70% w.r.t the DIST results.

Noise: This test is used to determine the performance of our approach given noisy detections, which are very common mainly due to small errors in the 2D-3D mapping. From the GT set with 2% missing data, random noise is added to every detection. The variances of the noise tested are $[0, 0.002, 0.004, 0.006, 0.008, 0.01]$

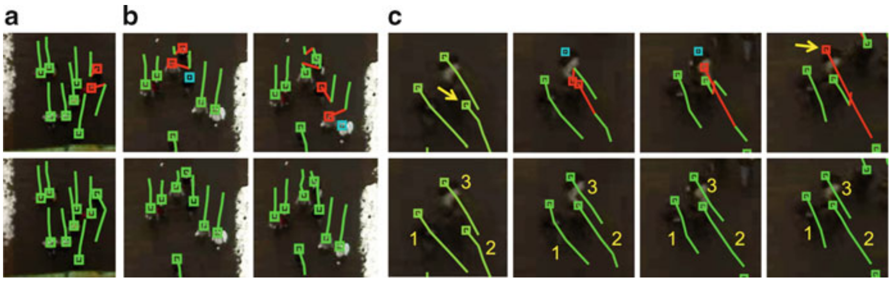

Fig. 11.9 (Color online) Experiments are repeated 50 times and average result, maximum and minimum are plotted. *Blue star* = results with DIST, *Green diamond* = results with SFM, *Red square* = results with SFM+GR. *From left to right*: Experiment with simulated missing data, with outliers, and with random noise

of the size of the scene observed. As expected, group information is the most robust to noise; if the position of pedestrian A is not correctly estimated, other pedestrians in the group will contribute to the estimation of the true trajectory of A.

These results corroborate that having good behavioral models becomes more important as the observations deteriorate. In Fig. 11.10 we plot the tracking results of a sequence with 12% simulated missing data. Only using distance information can see identity switches as shown in Fig. 11.10a. In Fig. 11.10b we can see how missing data affects the matching results. The matches are shifted, this chain reaction is due to the global optimization. In both cases, the use of SFM allows the tracker to interpolate the necessary detections and find the correct trajectories. Finally, in Fig. 11.10c we plot the wrong result which occurs because track 3 has two consecutive missing detections. Even with SFM, track 2 is switched for 3, since the switch does not create extreme changes in velocity. In this case, the grouping information is key to obtaining good tracking results. More results are shown in Fig. 11.13, first row.

11.5.6 Tracking Results

In this section, we compare results of several state-of-the-art methods on two publicly available datasets: a crowded town center [5] and the well-known PETS2009 dataset [12]. We compare results obtained with:

- Benfold et al. [5]: using the results provided by the authors for full pedestrian detections. The HOG detections are also given by the authors and used as input for all experiments.
- Zhan et al. [40]: globally optimum tracking based on network flow linear programming.
- Pellegrini et al. [30]: tracker based on Kalman Filter which includes social behavior.

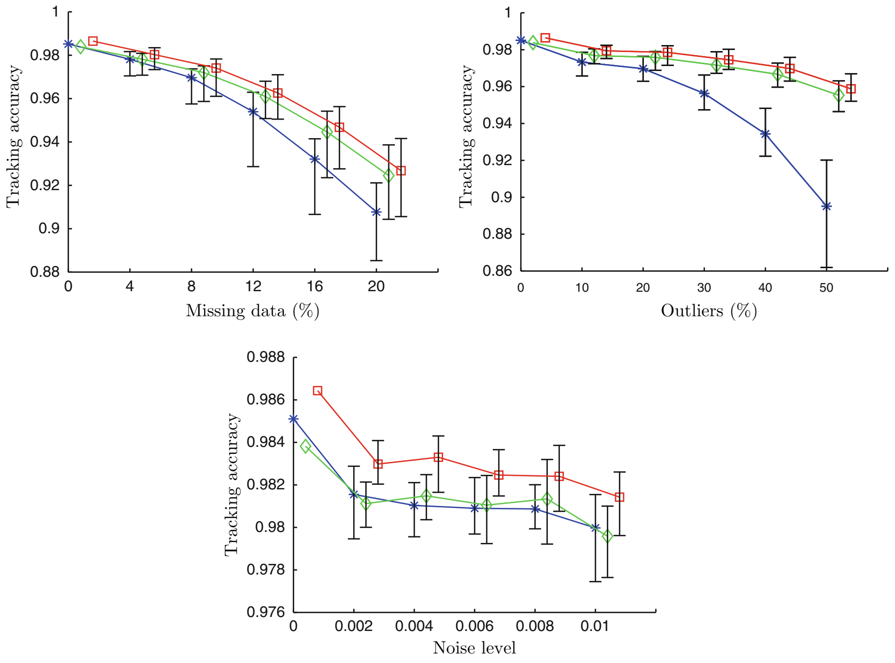

Fig. 11.10 (Color online) *Top row*: Tracking results with only DIST. *Bottom row*: Tracking results with SFM+GR. *Green* = correct trajectories, *Blue* = observation missing from the set, *Red* = wrong match. (**a**) Wrong match with DIST, corrected with SFM. (**b**) Missing detections cause the matches to shift due the global optimization; correct result with SFM. (**c**) Missed detection for subject 3 on two consecutive frames. With SFM, subject 2 in the first frame (*yellow arrow*) is matched to subject 3 in the last frame (*yellow arrow*), creating an identity switch; correct result with grouping information

- Yamaguchi et al. [38]: tracker based on Kalman Filter which includes social and grouping behavior.
- Leal-Taixé et al. [21]: globally optimum tracking based on network flow linear programming and including social and grouping behavior.

For a fair comparison, we do not use appearance information for any method. The methods [5,30,38] are online, while [21,40] processes the video in batches. For these last two methods, all experiments are performed with six iterations, a batch of 100 frames, $V_{\max} = 7\,\text{m/s}$, $F_{\max} = 10$, $\alpha = 0.5$ and $B_j = 0.3$.

11.5.6.1 Town Center Dataset

We perform tracking experiments on a video of a crowded town center [5], using one of every ten frames (simulating 2.5 fps). We show detection accuracy (DA), tracking

Table 11.1 Town Center sequence

	DA	TA	DP	TP	IDsw
HOG detections	63.1	–	71.9	–	–
Benfold et al. [5]	64.9	64.8	**80.5**	**80.4**	259
Zhang et al. [40]	66.1	65.7	71.5	71.5	114
Pellegrini et al. [30]	64.1	63.4	70.8	70.7	183
Yamaguchi et al. [38]	64.0	63.3	71.1	70.9	196
Leal-Taixé et al. [21]	**67.6**	**67.3**	71.6	71.5	**86**

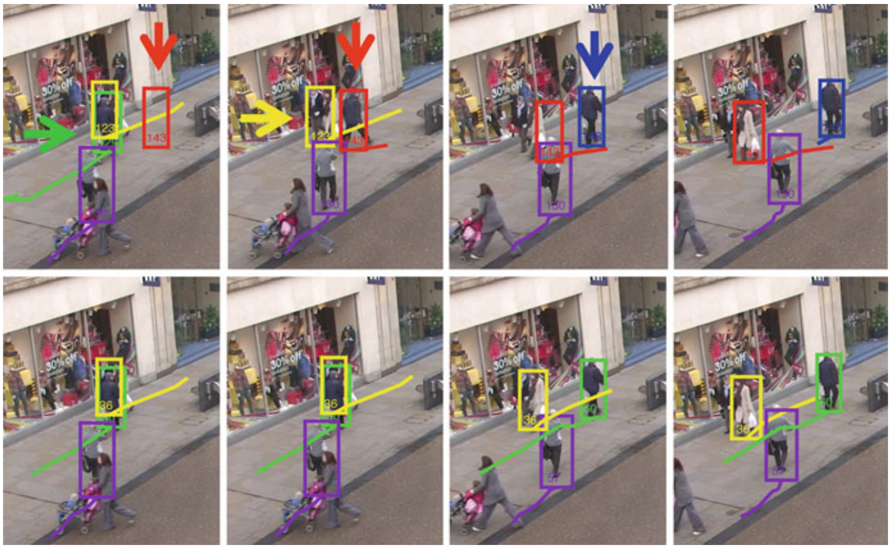

Fig. 11.11 Predictive approaches [30, 38] (*first row*) vs. Proposed method (*second row*)

accuracy (TA), detection precision (DP) and tracking precision (TP) measures as well as the number of identity switches (IDsw).

Note, the precision reported in [5] is about 9% higher than the input detections precision; this is because the authors use the motion estimation obtained with a KLT feature tracker to improve the exact position of the detections, while we use the raw detections. Still, our algorithm reports 64% less ID switches. As shown in Table 11.1, [21] algorithm outperforms [30, 38], both of which include social behavior information, by almost 4% in accuracy and with 50% less ID switches. In Fig. 11.11 we can see an example where [30, 38] fail. The errors are created in the greedy phase of predictive approaches, where people fight for detections. The red false detection in the first frame takes the detection in the second frame that should belong to the green trajectory (which ends in the first frame). In the third frame, the red trajectory overtakes the yellow trajectory and a new blue trajectory starts where the green should have been. None of the resulting trajectories violate the SFM

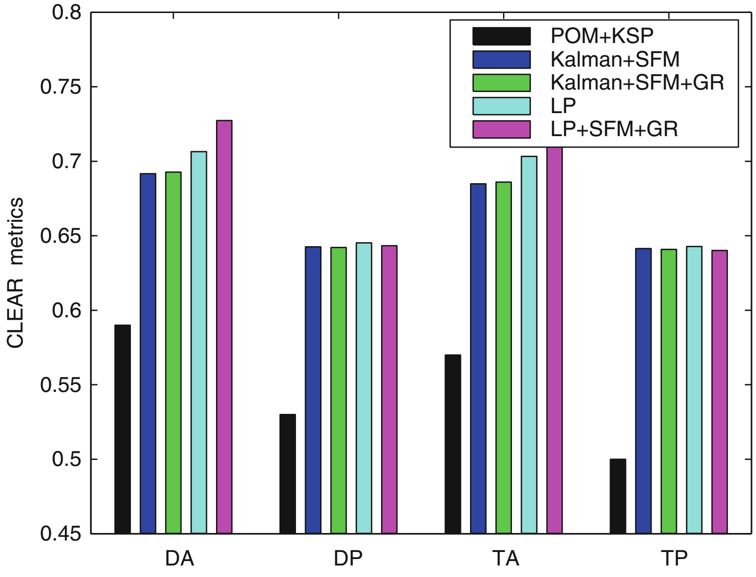

Fig. 11.12 Results of the proposed method on the PETS2009 dataset views 1. (a) Detection accuracy, *DA*. (b) Detection precision, *DP*. (c) Tracking accuracy, *TA*. (d) Tracking precision, *TP*

and GR conditions. On the other hand, a global optimization framework takes full advantage of the SFM and GR information and correctly recovers all the trajectories. More results of the proposed algorithm can be seen in Fig. 11.13, last row.

11.5.6.2 Results on the PETS2009 Dataset

In addition, we present results of monocular tracking on the PETS2009 sequence L1, View 1 with the detections obtained using the Mixture of Gaussians (MOG) background subtraction method. We compare the results with the previously described methods plus the monocular result of View 1 presented in [7], where the detections are obtained using the Probabilistic Occupancy Map (POM) and the tracking is done using k-shortest paths (Fig. 11.12).

The first observation that we make is that the linear programming methods (LP and LP+SFM+GR) clearly outperform predictive approaches in accuracy. This is because this dataset is very challenging from a social behavior point of view, because the subjects often change direction and groups form and split frequently. Approaches based on a probabilistic framework [21, 40] are better suited for unexpected behavior changes (like destination changes), where other predictive approaches fail [30,38]. We can also see that the LP+SFM+GR method has a higher accuracy than the LP method, which does not take into account social and grouping behavior. The grouping term is specially useful to avoid identity switches between

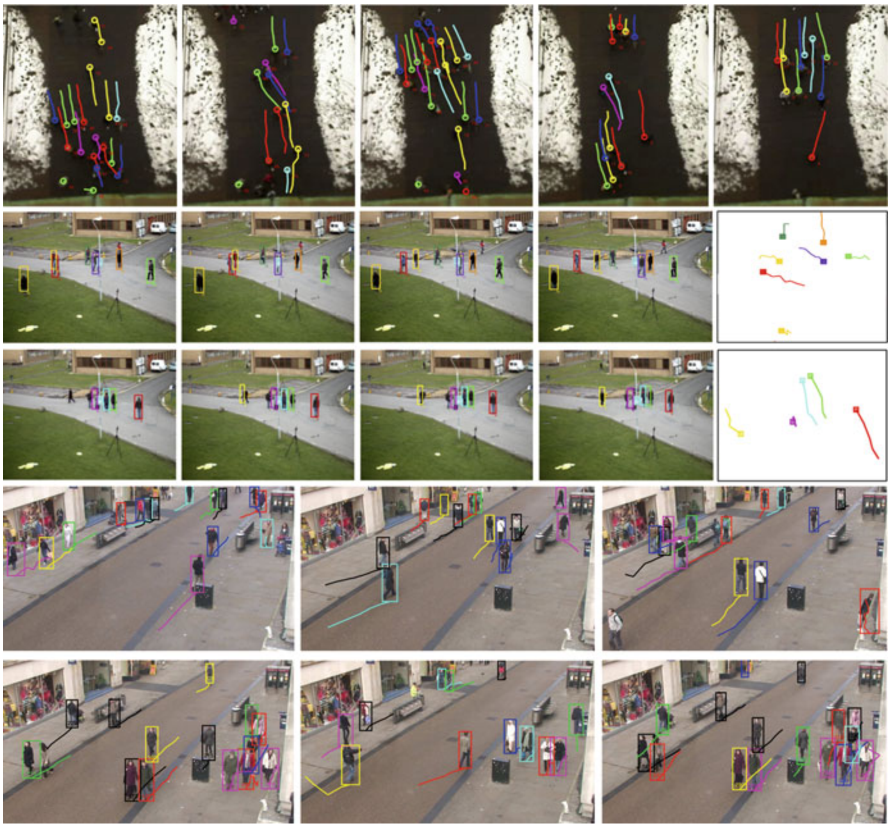

Fig. 11.13 *First row*: Results on the BIWI dataset (Sect. 11.5.5). The scene is heavily crowded, social and grouping behavior are key to obtaining good tracking results. *Second* and *third rows*: Results on the PETS2009 dataset (Sect. 11.5.6.2). *Last two rows*: Results on the Town Center dataset (Sect. 11.5.6.1)

member of a group (see an example in Fig. 11.13, third row, the cyan and green pedestrian who walk together). Precision is similar for all methods since the same detections have been used for all the experiments and we do not apply smoothing or correction of the bounding boxes.

11.6 Conclusions

In this chapter, we presented an overview of methods that integrate pedestrian interaction into a tracking framework in two ways: using a globally optimum solver or improving the dynamic model with social forces. Furthermore, we presented how to combine the strength of both approaches by finding the MAP estimate of the trajectories total posterior including social and grouping models using a minimum-cost network flow with an improved novel graph structure that outperforms existing approaches. People interaction is persistent rather than transient, hence the probabilistic formulation fully exploits the power of behavioral models as opposed to standard predictive and recursive approaches such as Kalman filtering. Experiments on three public datasets reveal the importance of using social interaction models for tracking in difficult conditions such as in crowded scenes with the presence of missed detections, false alarms and noise.

Acknowledgements This work was partially funded by the German Research Foundation, DFG projects RO 2497/7-1 and RO 2524/2-1.

References

1. Ahuja, R., Magnanti, T., Orlin, J.: Network Flows: Theory, Algorithms and Applications. Prentice Hall, Englewood Cliffs (1993)
2. Ali, S., Shah, M.: Floor fields for tracking in high density crowded scenes. In: ECCV, Marseille, France (2008)
3. Andriyenko, A., Schindler, K.: Globally optimal multi-target tracking on an hexagonal lattice. In: ECCV, Crete, Greece (2010)
4. Andriyenko, A., Schindler, K.: Multi-target tracking by continuous energy minimization. In: CVPR, Colorado Springs, USA (2011)
5. Benfold, B., Reid, I.: Stable multi-target tracking in real-time surveillance video. In: CVPR, Colorado Springs, USA (2011)
6. Berclaz, J., Fleuret, F., Fua, P.: Robust people tracking with global trajectory optimization. In: CVPR, New York, USA (2006)
7. Berclaz, J., Fleuret, F., Türetken, E., Fua, P.: Multiple object tracking using k-shortest paths optimization. In: TPAMI (2011)
8. Bertsekas, D.: Nonlinear Programming. Athena Scientific, Belmont (1999)
9. Breitenstein, M., Reichlin, F., Leibe, B., Koller-Meier, E., van Gool, L.: Robust tracking-by-detection using a detector confidence particle filter. In: ICCV, Kyoto, Japan (2009)
10. Choi, W., Savarese, S.: Multiple target tracking in world coordinate with single, minimally calibrated camera. In: ECCV (2010)
11. Dantzig, G.: Linear Programming and Extensions. Princeton University Press, Princenton (1963)
12. Ferryman, J.: Pets 2009 dataset: Proc. 11th IEEE Int'l Workshop performance and evaluation of tracking and surveillance, http:/pets2009.net (2011)
13. Gall, J., Yao, A., Razavi, N., van Gool, L., Lempitsky, V.: Hough forests for object detection, tracking, and action recognition. In: TPAMI (2011)
14. Ge, W., Collins, R., Ruback, B.: Automatically detecting the small group structure of a crowd. In: WACV, Snowbird, Utah (2009)
15. Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Phys. Rev. E **51**, 4282 (1995)
16. Jiang, H., Fels, S., Little, J.: A linear programming approach for multiple object tracking. In: CVPR, Minneapolis, Minnesota (2007)
17. Kasturi, R., Goldgof, D., Soundararajan, P., Manohar, V., Garofolo, J., Boonstra, M., Korzhova, V., Zhang, J.: Framework for performance evaluation for face, text and vehicle detection and tracking in video: data, metrics, and protocol. TPAMI (2009)

18. Kaucic, R., Perera, A., Brooksby, G., Kaufhold, J., Hoogs, A.: A unified framework for tracking through occlusions and across sensor gaps. In: CVPR, San Diego, USA (2005)
19. Khan, Z., Balch, T., Dellaert, F.: Mcmc-based particle filtering for tracking a variable number of interacting targets. In: TPAMI (2005)
20. Leal-Taixé, L., Heydt, M., Rosenhahn, A., Rosenhahn, B.: Automatic tracking of swimming microorganisms in 4d digital in-line holography data. In: IEEE Workshop on Motion and Video Computing (WMVC), Snowbird, Utah (2009)
21. Leal-Taixé, L., Pons-Moll, G., Rosenhahn, B.: Everybody needs somebody: modeling social and grouping behavior on a linear programming multiple people tracker. In: ICCV Workshops. 1st Workshop on Modeling, Simulation and Visual Analysis of Large Crowds, Barcelona, Spain (2011)
22. Leal-Taixé, L., Pons-Moll, G., Rosenhahn, B.: Branch-and-price global optimization for multi-view multi-object tracking. In: CVPR, Providence, USA (2012)
23. Leibe, B., Schindler, K., Cornelis, N., van Gool, L.: Coupled detection and tracking from static cameras and moving vehicles. TPAMI (2008)
24. Luber, M., Stork, J., Tipaldi, G., Arras, K.: People tracking with human motion predictions from social forces. In: ICRA, Anchorage, USA (2010)
25. Makhorin, A.: Gnu linear programming kit (glpk). http://www.gnu.org/software/glpk/ (2010)
26. Mehran, R., Oyama, A., Shah, M.: Abnormal crowd behavior detection using social force model. In: CVPR, Miami, USA (2009)
27. Nillius, P., Sullivan, J., Carlsson, S.: Multi-target tracking – linking identities using bayesian network inference. In: CVPR (2006)
28. Pelechano, N., Allbeck, J., Badler, N.: Controlling individual agents in high-density crowd simulation. In: Eurographics/ACM SIGGRAPH Symposium on Computer Animation, Prague, Czech Republic (2007)
29. Pellegrini, S., Ess, A., van Gool, L.: Improving data association by joint modeling of pedestrian trajectories and groupings. In: ECCV (2010)
30. Pellegrini, S., Ess, A., Schindler, K., van Gool, L.: You'll never walk alone: modeling social behavior for multi-target tracking. In: ICCV (2009)
31. Pirsiavash, H., Ramanan, D., Fowlkes, C.: Globally-optimal greedy algorithms for tracking a variable number of objects. In: CVPR (2011)
32. Rodriguez, M., Sivic, J., Laptev, I., Audibert, J.: Data-driven crowd analysis in videos. In: ICCV (2011)
33. Scovanner, P., Tappen, M.: Learning pedestrian dynamics from the real world. In: ICCV (2009)
34. Shu, G., Dehghan, A., Oreifej, O., Hand, E., Shah, M.: Part-based multiple-person tracking with partial occlusion handling. In: CVPR (2012)
35. Suurballe, J.: Disjoint paths in a network. Networks **4**, 125–145 (1974)
36. Wu, B., Nevatia, R.: Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet part detectors. In: IJCV **75**(2) (2007)
37. Wu, Z., Kunz, T., Betke, M.: Efficient track linking methods for track graphs using network-flow and set-cover techniques. In: CVPR (2011)
38. Yamaguchi, K., Berg, A., Ortiz, L., Berg, T.: Who are you with and where are you going? In: CVPR (2011)
39. Yang, M., Yu, T., Wu, Y.: Game-theoretic multiple target tracking. In: ICCV (2007)
40. Zhang, L., Li, Y., Nevatia, R.: Global data association for multi-object tracking using network flows. In: CVPR (2008)