DensePeds: Pedestrian Tracking in Dense Crowds Using Front-RVO and Sparse Features

Rohan Chandra, Uttaran Bhattacharya, Aniket Bera, and Dinesh Manocha

University of Maryland, University of North Carolina

Abstract—We present a pedestrian tracking algorithm, DensePeds, that tracks individuals in highly dense crowds (>2 pedestrians per square meter). Our approach is designed for videos captured from front-facing or elevated cameras. We present a new motion model called Front-RVO (FRVO) for predicting pedestrian movements in dense situations using collision avoidance constraints and combine it with state-of-the-art Mask R-CNN to compute sparse feature vectors that reduce the loss of pedestrian tracks (false negatives). We evaluate DensePeds on the standard MOT benchmarks as well as a new dense crowd dataset. In practice, our approach is 4.5× faster than prior tracking algorithms on the MOT benchmark and we are state-of-the-art in dense crowd videos by over 2.6% on the absolute scale on average.

I. INTRODUCTION

Pedestrian tracking is the problem of maintaining the consistency in the temporal and spatial identity of a person in an image sequence or a crowd video. This is an important problem that helps us not only extract trajectory information from a crowd scene video but also helps us understand high-level pedestrian behaviors [1]. Many applications in robotics and computer vision such as action recognition and collision-free navigation and trajectory prediction [2] require tracking algorithms to work accurately in real time [3]. Furthermore, it is crucial to develop general-purpose algorithms that can handle front-facing cameras (used for robot navigation or autonomous driving) as well as elevated cameras (used for urban surveillance), especially in densely crowded areas such as airports, railway terminals, or shopping complexes.

Closely related to pedestrian tracking is pedestrian detection, which is the problem of detecting multiple individuals in each frame of a video. Pedestrian detection has received a lot of traction and gained significant progress in recent years. However, earlier work in pedestrian tracking [4], [5] did not include pedestrian detection. These tracking methods require manual, near-optimal initialization of each pedestrian’s state information in the first video frame. Further, sans-detection methods need to know the number of pedestrians in each frame apriori, so they do not handle cases in which new pedestrians enter the scene during the video. Tracking by detection overcomes these limitations by employing a detection framework to recognize pedestrians entering at any point of time during the video and automatically initialize their state-space information.

However, tracking pedestrians in dense crowd videos where there are 2 or more pedestrians per square meter remains a challenge for the tracking-by-detection literature. These videos suffer from severe occlusion, mainly due to the pedestrians walking extremely close to each other and frequently crossing each other’s paths. This makes it difficult to track each pedestrian across the video frames. Tracking-by-detection algorithms compute a bounding box around each pedestrian. Because of their proximity in dense crowds, the bounding boxes of nearby pedestrians overlap which affects the accuracy of tracking algorithms.

Main Contributions. Our main contributions in this work are threefold.

1) We present a pedestrian tracking algorithm called DensePeds that can efficiently track pedestrians in crowds with 2 or more pedestrians per square meter. We call such crowds “dense”. We introduce a new motion model called Frontal Reciprocal Velocity Obstacles (FRVO), which extends traditional RVO [6] to work with front- or elevated-view cameras. FRVO uses an elliptical approximation for each pedestrian and estimates pedestrian dynamics (position and velocity) in dense crowds by considering intermediate goals and collision avoidance constraints.

2) We combine FRVO with the state-of-the-art Mask R-CNN object detector [7] to compute sparse feature vectors. We show analytically that our sparse feature vector computation reduces the probability of the loss of pedestrian tracks (false negatives) in dense scenarios. We also show by experimentation that using sparse feature vectors makes our method outperform state-of-the-art tracking methods on dense crowd videos, without any additional requirement for training or optimization.

3) We make available a new dataset consisting of dense crowd videos. Available benchmark crowd videos are extremely sparse (less than 0.2 persons per square meter). Our dataset, on the other hand, presents a more challenging and realistic view of crowds in public areas in dense metropolitans.

On our dense crowd videos dataset, our method outper-
forms the state-of-the-art, by 2.6% and reduce false negatives by 11% on our dataset. We also validate the benefits of FRVO and sparse feature vectors through ablation experiments on our dataset. Lastly, we evaluate and compare the performance of DensePeds with state-of-the-art online methods on the MOT benchmark [8] for a comprehensive evaluation. The benefits of our method do not hold for sparsely crowded videos such as the ones in the MOT benchmark since FRVO is based on reciprocal velocities of two colliding pedestrians. Unsurprisingly, we do not outperform the state-of-the-art methods on the MOT benchmark since the MOT sequences do not contain many colliding pedestrians, thereby rendering FRVO ineffective. Nevertheless, our method still has the lowest false negatives among all the available methods. Its overall performance is state of the art on dense videos and is in the top 13% of all published methods on the MOT15 and top 20% on MOT16.

II. RELATED WORK

The most relevant prior work for our work is on object detection, pedestrian tracking, and motion models used in pedestrian tracking. Each of these fields is well-studied, and a comprehensive overview for each of them is well beyond the scope of this paper. We only present a brief overview of prior work on these fields that are most closely related to our work in this paper.

A. Object Detection


More recently, the prevalence of CNNs has led to the development of Mask R-CNN [7], which extends Faster R-CNN to include pixel-level segmentation. The use of Mask R-CNN in pedestrian tracking has been limited, although it has been used for other pedestrian-related problems such as pose estimation [14].

B. Pedestrian Tracking

There have been considerable advances in object detection since the advent of deep learning, which has led to substantial research in the intersection of deep learning and the tracking-by-detection paradigm [15], [16], [17], [18]. However, [18], [16] operate at less than one fps on a GPU and have low accuracy on standard benchmarks while [15] sacrifices accuracy to increase tracking speed. Finally, deep learning methods require expensive computation, often prohibiting real-time performance with inexpensive computing resources. Some recent methods [19], [20] achieve high accuracy but may require high-quality, heavily optimized detection features for good performance. For an up-to-date review of tracking-by-detection algorithms, we refer the reader to methods submitted to the MOT [8] benchmark.

C. Motion Models in Pedestrian Tracking

Motion models are commonly used in pedestrian tracking algorithms to improve tracking accuracy [21], [22], [23], [24]. [22] presents a variation of MHT [25] and shows that it is at par with the state-of-the-art from the tracking-by-detection paradigm. [24] uses a motion model to combine fragmented pedestrian tracks caused by occlusion. These methods are based on linear constant velocity or acceleration models. Such linear models, however, cannot characterize pedestrian dynamics in dense crowds [26]. RVO [6] is a non-linear motion model that has been used for pedestrian tracking in dense videos to compute intermediate goal locations, but it only works with top-facing videos and circular pedestrian representations. Other motion models used in pedestrian tracking are the Social Force model [27], LTA [28], and ATTR [29].

There are also many discrete motion models that represent each individual or pedestrian in a crowd as a particle (or as a 2D circle on a plane) to model the interactions. These include models based on simple interaction rules [30], repulsive forces [31] and velocity-based optimization algorithms [32], [6]. More recent discrete approaches are based on short-term planning using a discrete approach [33] and cognitive models [34]. However, these methods are based on circular agent representation and do not work well for front-facing pedestrians in dense crowd videos as they are overly conservative in terms of motion prediction.

III. FRVO: LOCAL TRAJECTORY PREDICTION

When walking in dense crowds, pedestrians keep changing their velocities frequently to avoid collisions with other pedestrians. Pedestrians also exhibit local interactions and other collision-avoidance behaviors such as side-stepping, shoulder-turning, and backpedaling [35]. As a result, prior motion models with constant velocity or constant acceleration assumptions do not accurately model crowded scenarios. Further, defining an accurate motion model gets even more challenging in front-view videos due to occlusion and proximity issues. For example, in top-view videos, one uses circular representations to model the shape of the heads of pedestrians. It is, generally, physically impossible for two individual’s heads to occlude or occupy the same space. In front facing crowd videos, however, occlusion is a common problem. In this section, we introduce our non-linear motion model, Frontal RVO (FRVO), which is designed to work with crowd videos captured using front- or elevated-view cameras.

A. Notations

Table I lists the notations used in this paper. A few important points to keep in mind are:

- While $h_j$ denotes a detected pedestrian, $f_{h_j}$ denotes the feature vector extracted from the segmented box of $h_j$.
- $H_i = \{h_j : |p_i - h_j| \leq \rho, p_i \in \mathcal{P}\}$

B. Analytical Comparison with RVO

Our motion model is an extension of the RVO (Reciprocal Velocity Obstacle) [6] approach, which can accurately model the trajectories of pedestrians using collision avoidance constraints. However, RVO models pedestrians using circular shapes which result in false overlaps in front- and elevated-view camera videos (Figure 3). A false overlap is essentially a false positive wherein RVO would signal a collision while the actual scene would not contain a collision. We show here that false overlaps cause the accuracy of RVO to drop by an error margin of $\Delta$. 

### Table I: Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_j$</td>
<td>Detected pedestrian</td>
</tr>
<tr>
<td>$f_{h_j}$</td>
<td>Feature vector of $h_j$</td>
</tr>
<tr>
<td>$H_i$</td>
<td>Set of pedestrians within a certain radius $\rho$ from $p_i$</td>
</tr>
<tr>
<td>$\mathcal{P}$</td>
<td>Set of all pedestrians in the scene</td>
</tr>
</tbody>
</table>

- While $h_j$ denotes a detected pedestrian, $f_{h_j}$ denotes the feature vector extracted from the segmented box of $h_j$.
- $H_i = \{h_j : |p_i - h_j| \leq \rho, p_i \in \mathcal{P}\}$

### B. Analytical Comparison with RVO

Our motion model is an extension of the RVO (Reciprocal Velocity Obstacle) [6] approach, which can accurately model the trajectories of pedestrians using collision avoidance constraints. However, RVO models pedestrians using circular shapes which result in false overlaps in front- and elevated-view camera videos (Figure 3). A false overlap is essentially a false positive wherein RVO would signal a collision while the actual scene would not contain a collision. We show here that false overlaps cause the accuracy of RVO to drop by an error margin of $\Delta$. 

- While $h_j$ denotes a detected pedestrian, $f_{h_j}$ denotes the feature vector extracted from the segmented box of $h_j$.
- $H_i = \{h_j : |p_i - h_j| \leq \rho, p_i \in \mathcal{P}\}$
TABLE I

<table>
<thead>
<tr>
<th>Notations We Use in This Paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_i )</td>
</tr>
<tr>
<td>( h_j )</td>
</tr>
<tr>
<td>( f p_i ) &amp; ( H_j )</td>
</tr>
<tr>
<td>( l )</td>
</tr>
<tr>
<td>( \mathcal{H} ) &amp; ( \mathcal{P} )</td>
</tr>
<tr>
<td>( \rho )</td>
</tr>
<tr>
<td>( H_i )</td>
</tr>
<tr>
<td>( N_i )</td>
</tr>
<tr>
<td>( d(a, b) )</td>
</tr>
</tbody>
</table>

\( \Delta \leq \) const represents the set of velocities for \( p_i \) for a pedestrian \( \in \mathcal{H}_i \). The set of permitted velocities \( \Psi_t \) is the velocity the pedestrian would have taken in the absence of obstacles or colliding pedestrians, computed using the standard RVO formulation.

C. Computing Predicted Velocities

Each pedestrian \( p_i \) is represented using the following 8-dimensional state vector, \( \Psi_t = [x, v, v_{\text{pref}}, l, w] \), where \( x, v \), and \( v_{\text{pref}} \) denote the current position of the pedestrian’s center of mass, the current velocity, and the preferred velocity, respectively. \( l \) is the height of the pedestrian’s visible face, and \( w \) captures the shoulder length. \( v_{\text{pref}} \) is the velocity the pedestrian would take to move without the need for collision avoidance. We use \( \Psi_t \) to compute the set of permitted velocities for each pedestrian.

We assume the pedestrian is oriented towards their direction of motion. For each frame and each pedestrian, we construct the half-plane constraints for each of its neighboring pedestrians and obstacles to predict its motion. We use velocity obstacles to compute the set of permitted velocities for a pedestrian. Each pedestrian \( p_i \), \( j \thinspace \text{th} \) in its set of all detected persons within a circle of radius \( r \) centered around \( p_i \), defined as:

\[
\Psi_t = [x, v, v_{\text{pref}}, l, w] \]

where \( x, v \), and \( v_{\text{pref}} \) denote the current position of the pedestrian’s center of mass, the current velocity, and the preferred velocity, respectively. \( l \) is the height of the pedestrian’s visible face, and \( w \) captures the shoulder length. \( v_{\text{pref}} \) is the velocity the pedestrian would have taken in the absence of obstacles or colliding pedestrians, computed using the standard RVO formulation.

We assume the pedestrians are oriented towards their direction of motion. For each frame and each pedestrian, we construct the half-plane constraints for each of its neighboring pedestrians and obstacles to predict its motion. We use velocity obstacles to compute the set of permitted velocities for a pedestrian. Given two pedestrians \( p_i \) and \( p_j \), the velocity obstacle of \( p_i \) induced by \( p_j \), denoted by \( V\Omega^r_{p_i,p_j} \), constitutes the set of velocities for \( p_i \) that would result in a collision with \( p_j \) at some time before \( \tau \). By definition, \( p_i \) and \( p_j \) are guaranteed to be collision-free for at least time \( \tau \), if \( \Psi_t \notin V\Omega^r_{p_i,p_j} \). An pedestrian \( p_i \) computes the velocity obstacle \( V\Omega^r_{p_i,\mathcal{H}_i} \) for each of its neighboring pedestrians, \( p_j \). The set of permitted velocities for a pedestrian \( p_i \) is simply the convex region given by the intersection of the half-planes of the permitted velocities induced by all the neighboring pedestrians and obstacles. We denote this convex region for pedestrian \( p_i \) and time horizon \( \tau \) as \( FRVO^r_{p_i} \). Thus, \( FRVO^r_{p_i} = \bigcup_{p_j \in \mathcal{H}_i} V\Omega^r_{p_i,p_j} \).

For each pedestrian \( p_i \), we compute the new velocity \( v_{\text{new}} = \arg \max \| v - v_{\text{pref}} \| \) from \( FRVO^r_{p_i} \), that minimizes the deviation from its preferred velocity \( v_{\text{pref}} \) such that \( v \notin FRVO^r_{p_i} \).

In order to compute collision-free velocities, we need to compute the Minkowski Sums of ellipses. In practice, computing the exact Minkowski sums of ellipses is much more expensive as compared to those of circles. To overcome the complexity of exact Minkowski Sum computation, we compute conservative linear approximations of ellipses [35] and represent them as convex polygons. As a result, the collision avoidance problem reduces to linear programming.

We use the predicted velocities to estimate the new position of a pedestrian in the next time-step.

IV. DENSEPeds AND SPARSE FEATURES

In this section, we describe our tracking algorithm, DensePeds, which is based on the tracking-by-detection

\[
\Delta \leq \sin^{-1}\left(\frac{2r}{\|p\|}\right) \leq \sin^{-1}\left(\frac{2r - \delta}{\|p\|}\right) \leq \sin^{-1}\left(\frac{2r}{\|p\|}\right) \leq \sin^{-1}\left(\frac{2r - \delta}{\|p\|}\right)
\]

\( \Delta \) is upper-bounded by a function of \( \delta \). Observe that the error bound increases for higher \( \delta \). This is interpreted as follows: the more we increase the \( \delta \)-overlap, the transparent gray cone will correspondingly shrink, and the distance between the RVO generated velocity, and the ground truth velocity will increase.

We therefore require geometric representations for which \( \delta \to 0 \). We propose to model each pedestrian using an elliptical pedestrian representation, with the ellipse axes capturing the height of the pedestrian’s visible face and the shoulder length. We do not manually differentiate between the major and minor axes since ellipses can be tall or fat depending on the pedestrian orientation, and the particular axis to map either face length or shoulder width will vary. We observe that doing so produces an effective approximation. We only use the face and the shoulder in the FRVO formulation, because occlusion makes it difficult to observe other parts of the body in dense videos reliably.
and is calculated as:

\[ \text{IoU overlap of the bounding box of the detected pedestrian}, h, \text{used to solve the following optimization problem to identify Cosine metric and the IoU overlap} [38]. The Cosine metric is [37]. This matching process is performed in two ways: the vectors, which are matched using an association algorithm. Figure 3 and Figure 3. This results in another set of bounding pedestrian segmentation to generate segmented boxes, which want to assign labels for pedestrians in the next frame. We next frame. We illustrate our approach in Figure 4. At current row of Table I) and the Hungarian algorithm [37] and are matched using the Cosine metric [19], to compute binary feature vectors. Since a large portion of a segmented box contains zeros, these feature vectors are mostly sparse. We refer to these vectors as our sparse feature vectors. Our choice of the network parameters governs the size of the feature vectors.

### 2) Reduced Probability of Track Loss: We now show how sparse feature vectors generated from segmented boxes reduce the probability of the loss of pedestrian tracks (false negatives).

We define \( T_i = \{\Psi_{i,t}\} \) to be the set of positively identified states for \( p_i \) until time \( t \). We denote the time since the last update to a track ID as \( \mu \). We denote the ID of \( p_i \) as \( \alpha \) and we represent the correct assignment of an ID to \( p_i \) as \( \Gamma(\alpha) \). The threshold for the Cosine metric is \( \lambda \sim U[0,1] \). The threshold for the track age, i.e., the number of frames before which track is destroyed, is \( \xi \). We denote the probability of an event that uses Mask R-CNN as the primary object detection algorithm with \( P^M(\cdot) \) and the probability of an event that uses a standard Faster R-CNN [13] as the primary object detection algorithm (i.e., outputs bounding boxes without boundary subtraction) with \( P^F(\cdot) \). Finally, \( T_i \sim \{\emptyset\} \) represents the loss of \( T_i \) by occlusion.

We now state and prove the following lemma.

**Lemma IV.1.** For every pair of feature vectors \( (f_{h_i}^M, f_{h_j}^F) \) generated from a segmented box and a bounding box respectively, if \( \|f_{h_i}^M\|_0 > \|f_{h_j}^F\|_0 \), then \( d(f_{p_i}, f_{h_i}^M) < d(f_{p_j}, f_{h_j}^F) \) with probability \( 1 - \frac{B}{A} \), where \( A \) and \( B \) are positive integers and \( A > B \).
Proof. Using the definition of the Cosine metric, the lemma reduces to proving the following,

\[ f_T^p(f_{h_i}^M - f_{h_j}^F) > 0 \]  \hspace{1cm} (3)

We pad both \( f_{h_i}^M \) and \( f_{h_j}^F \) such that \( \| f_{h_i}^M \|_0 > \| f_{h_j}^F \|_0 \).

We reduce \( f_{p_i}^T, f_{h_i}^M, \) and \( f_{h_j}^F \) to binary vectors, i.e., vectors composed of 0s and 1s. Let \( \Delta f = f_{h_i}^M - f_{h_j}^F \). We denote the number of 1s and -1s in \( \Delta f \) as \( A \) and \( B \), respectively. Now, let \( x \) and \( y \) denote the \( L_0 \) norm of \( f_{h_i}^M \) and \( f_{h_j}^F \), respectively. From our padding procedure, we have \( x > y \). Then, if \( y = B \), then \( x = A \) and we trivially have \( A > B \). But if \( y > B \), then \( A = x - (y - B) \Rightarrow A - B = x - y \). From \( x > y \), it again follows that \( A > B \). Thus, \( x > y \Rightarrow A > B \).

Next, we define a (1, 1) coordinate in an ordered pair of vectors as the coordinate where both vectors contain 1s. Similarly, a (1, -1) coordinate in an ordered pair of vectors is the coordinate where the first vector contains 1 and the second vector contains -1. Then, let \( p_a \) and \( p_b \) respectively denote the number of (1, 1) coordinates and (1, -1) coordinates in the pair \( (f_{p_i}^T, \Delta f) \). By definition, we have \( 0 < p_a < A \) and \( 0 < p_b < B \). Thus, if we assume \( p_a \) and \( p_b \) to be uniformly distributed, it directly follows that \( P(p_a > p_b) = 1 - \frac{B}{A} \).

Based on Lemma IV.1, we finally prove the following proposition.

**Proposition IV.1.** With probability \( 1 - \frac{B}{A} \), sparse feature vectors extracted from segmented boxes decrease the loss of pedestrian tracks, thereby reducing the number of false negatives in comparison to regular bounding boxes.

Proof. In our approach, we use Mask R-CNN for pedestrian detection, which outputs bounding boxes and their corresponding masks. We use the mask and bounding box pair to generate a segmented box (Section IV-B.1). The correct assignment of an ID depends on successful feature matching between the predicted measurement feature and the optimal detection feature, that is, when the cosine cost between the two features is below a set threshold, \( \lambda \). But since the correct ID assignment depends on multiple factors including a low cosine cost, we instead exploit the equivalency of the contrapositive,

\[ d(f_{p_i}, f_{h_j}^\ast) > \lambda \Leftrightarrow (\alpha = \emptyset) \]  \hspace{1cm} (4)

Equation 4 indicates that when the cosine cost is greater than \( \lambda \), then feature matching fails and therefore an ID fails to be assigned, or equivalently, set to \( \emptyset \). Using Lemma IV.1 and the fact that \( \lambda \), \( \sim \), \( \text{i.i.d.} \), \( U[0, 1] \),

\[ P(d(f_{p_i}, f_{h_i}^M) > \lambda) < P(d(f_{p_i}, f_{h_i}^F) > \lambda) \]

Using the equivalency in Eq. 4, we obtain,

\[ P^M(\alpha = \emptyset) < P^F(\alpha = \emptyset) \]  \hspace{1cm} (5)

Now, in our approach, we set certain fixed conditions that need to be satisfied for a track to be destroyed or lost. Those conditions are listed as follows:

1) DensePeds updates the ID of every pedestrian after each frame. If an update fails to occur for \( \mu > \zeta \) frames, then this leads to loss of the track of that pedestrian.

2) If the first condition is satisfied, and the current ID of a pedestrian is not set, then the track of that pedestrian is lost.

We formalize the two conditions as follows,

\[ (\mu > \zeta) \land (\alpha = \emptyset) \Leftrightarrow T_i \leftarrow \{\emptyset\} \]

Using Eq. 5, it follows that,

\[ P^M(T_i \leftarrow \{\emptyset\}) < P^F(T_i \leftarrow \{\emptyset\}) \]  \hspace{1cm} (6)

Informally, equation 6 tells us that the probability of losing a track using segmented boxes is less than the probability of losing a track if we were to use regular bounding boxes.

To complete the proof, we now show that equation 6 implies that fewer lost tracks leads to fewer false negatives. We start by defining the total number of false negatives (FN) as

\[ FN = \sum_{t=1}^{T} \sum_{p_g \in \mathcal{G}} \delta_{T_t} \]  \hspace{1cm} (7)

where \( p_g \in \mathcal{G} \) denotes a ground truth pedestrian in the set of all ground truth pedestrians at current time \( t \) and \( \delta_z = 1 \) for \( z = 0 \) and 0 elsewhere. This is a variation of the Kronecker delta function. Using Eq. 6 and Eq. 7, we can say that fewer lost tracks \( (T_i \leftarrow \{\emptyset\}) \) indicate a smaller number of false negatives. We empirically demonstrate this analysis in Section V.

The upper bound, \( P^F(T_i) \), in Eq. 6 depends on the amount of padding done to \( f_{p_i} \) and \( f_{h_i} \). A general observed trend
is that a higher amount of padding results in a larger upper bound in Eq. 6.

V. EXPERIMENTAL EVALUATION

A. Datasets

We evaluate DensePeds on both our dense pedestrian crowd dataset and on the popular MOT [8] benchmark. MOT is a standard benchmark for testing the performance of multiple object tracking algorithms, and it subsumes older benchmarks such as KITTI [46] and PETS [47]. When evaluating on MOT, we use the popular MOT16 and MOT15 sequences. However, the main drawback of the sequences in the MOT benchmark is that they do not contain dense crowd videos that we address in this paper.

Our dataset consists of 8 dense crowd videos where the crowd density ranges from 2 to 2.7 pedestrians per square meter. By comparison, the densest crowd video in the MOT benchmark has around 0.2 persons per square meter. All videos in our dataset are shot at an elevated view using a stationary camera. Seven of the eight videos are shot 1080p resolution, and one video is at 480p resolution.

While we agree that the current size of our dataset is prohibitively small for training existing deep learned-based object detectors, we showcase the results of our method on this dataset to validate our claims and analyses. We plan to expand and release a benchmark version of our dataset in the recent future.

B. Evaluation Metrics and Methods

1) Metrics: We use a subset of the standard CLEAR MOT metrics [48] for evaluating the performance of our algorithm. Specifically, the metrics we use are:

- **Number of mostly tracked trajectories (MT).** Correct ID association with a pedestrian across frames for at least 80% of its life span in a video.
- **Number of mostly lost trajectories (ML).** Correct ID association with a pedestrian across frames for at most 20% of its life span in the video.
- **False Negatives (FN).** Total number of false negatives in object detection (i.e., pedestrian tracks lost) over the video.
- **Identity Switches (IDSW).** The total number of identity switches over pedestrians in the video.
- **MOT Accuracy (MOTA).** Overall tracking accuracy taking into account false positives (FP), false negatives (FN), and ID switches (IDSW). It is defined as

\[
MOTA = 1 - \frac{FP + FN + IDS}{GT},
\]

where \(GT\) is the sum of annotated pedestrians in all the frames in the video.

These are the most important metrics in CLEAR MOT for evaluating a tracker’s performance. We exclude metrics that evaluate the performance of detection since object detection is not a contribution of this paper. We have also excluded the number of false positives from our tables since it is already included in the definition of MOTA and is not the focus of this paper. On the other hand, we highlight the results of false negatives to validate the theoretical formulation of our algorithm in Section III-B.

2) Methods: There is a large number of tracking-by-detection methods on the MOT benchmark, and it is not feasible to evaluate our method against all of them in the limited scope of this paper. Further, many methods submitted on the MOT benchmark server self-defense strategy. Thus, to keep our evaluations as fair and competitive as possible, we compare with state-of-the-art online trackers on the MOT16 benchmark server. MOT16 is the best online published tracker on the MOT benchmark with available code, and MDP is the second-best method with available code. We use their off-the-shelf implementations, with weights pre-trained on the MOT benchmark, to compare with DensePeds.

We observe that DensePeds produces the lowest false negatives of all the methods. Thus, by Proposition IV.1 and Table II, DensePeds provably reduces the number of false negatives on dense crowd videos.

The low MT and high ML scores for the sequences are not an indicator of poor performance; rather, they are a consequence of the strict definitions of the metrics. For example, in dense crowd videos, it is challenging to maintain a track for 80% of a pedestrian’s life span. DensePeds has the highest MT and lowest ML percentages of all the methods. This explains our high number of ID switches as a higher number of pedestrians being tracked correspondingly increase the likelihood of ID switches. Most importantly, the MOTA of DensePeds is 2.9% more than that of MOTDT and 2.6% more than that of MDP. Roughly speaking, this is equivalent to an average rank difference of 19 from MOTDT and 17 from MDP on the MOT benchmark. We do not include the tracking speed metric (Hz) as shown in Table III since that is a metric produced exclusively by the MOT benchmark server.

2) MOT Benchmark: Although our algorithm is primarily targeted for dense crowds, in the interest of thorough analysis, we also evaluate DensePeds on sparser crowds. We select a wide range of methods (as described in Section V-B.2) to compare with DensePeds and highlight its relative advantages and disadvantages in Table III.

As expected, we do not outperform state of the art on the MOT sequences due to its sparse crowd sequences. We specifically point to our low MOTA scores which we attribute to our algorithm requiring a more accurate ground truth than the ones provided by the MOT benchmark. Our detection and tracking are highly sensitive — they can track pedestrians that are too distant to be manually labeled. This erroneously leads to a higher count of false positives and reduces the MOTA. This was observed to be true for the methods we
Next, we replace FRVO in turn with a constant velocity motion model [19], the Social Forces model [31], and the IDS [15].

However, following Proposition IV.1, our method achieves approximately 4.5× the lowest number of false negatives among all the methods on the MOT benchmark as well. In terms of runtime, we are approximately 4.5× faster than the state-of-the-art methods on an NVIDIA Titan Xp GPU.

3) Ablation Experiments: In Table IV, we validate the benefits of sparse feature vectors (obtained via segmented boxes) in detection and FRVO in tracking in dense crowds with ablation experiments.

First, we replace segmented boxes with regular bounding boxes (boxes without background subtraction), and compare its performance with DensePeds on the MOT benchmark. In accordance with Proposition IV.1, DensePeds reduces the number of false negatives by 20.7% on relative.

Next, we replace FRVO in turn with a constant velocity motion model [19], the Social Forces model [31], and the standard RVO model [6], all without changing the rest of the DensePeds algorithm. All of these models fail to track pedestrians in dense crowds (ML nearly 100%). Note that a consequence of failing to track pedestrians is the unusually low number of ID switches that we observe for these methods. This, in turn, leads to their comparatively high MOTA, despite being mostly unable to track pedestrians. Using FRVO, we decrease ML by 55% on relative. At the same time, our MOTA improves by a significant 12% on relative and 8.5% on absolute.

VI. ACKNOWLEDGEMENT

This research is supported in part by ARO grant W911NF19-1-0069, Alibaba Innovative Research (AIR) program and Intel.

REFERENCES


TABLE IV
ABLATION STUDIES WHERE WE DEMONSTRATE THE ADVANTAGE OF SEGMENTED BOXES AND FRVO. WE REPLACE SEGMENTED BOXES WITH REGULAR BOUNDING BOXES (BBOX) AND COMPARE ITS PERFORMANCE WITH DENSEPeds ON THE MOT BENCHMARK. WE ALSO REPLACE THE FRVO MODEL IN TURN WITH A CONSTANT VELOCITY MODEL (CONST. VEL.), THE SOCIAL FORCES MODEL (SF) [31] AND THE STANDARD RVO MODEL (RVO) [6] AND COMPARE THEM ON OUR DENSE CROWD DATASET. BOLD IS BEST. ARROWS (∧, ∨) INDICATE THE DIRECTION OF BETTER PERFORMANCE. OBSERVATION:
Using FRVO IMPROVES THE MOTA BY 13.5% OVER THE NEXT BEST ALTERNATIVE. USING SEGMENTED BOXES REDUCES THE FALSE NEGATIVES BY 20.7%.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BBox</td>
<td>14.3</td>
<td>34.3</td>
<td>74.2</td>
<td>76.4</td>
<td>17.6</td>
<td>18.0</td>
</tr>
<tr>
<td>Densespds</td>
<td>19.0</td>
<td>37.2</td>
<td>74.9</td>
<td>75.9</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>Const. VEL</td>
<td>0</td>
<td>68.0</td>
<td>82.4</td>
<td>64.2</td>
<td>67.1</td>
<td></td>
</tr>
<tr>
<td>SF</td>
<td>0</td>
<td>68.1</td>
<td>81.9</td>
<td>64.4</td>
<td>67.3</td>
<td></td>
</tr>
<tr>
<td>RVO</td>
<td>0</td>
<td>69.1</td>
<td>81.8</td>
<td>66.4</td>
<td>67.8</td>
<td></td>
</tr>
<tr>
<td>Densespds</td>
<td>7.0</td>
<td>64.4</td>
<td>1.208</td>
<td>58.225</td>
<td>56.7</td>
<td>78.3</td>
</tr>
</tbody>
</table>


