WILDTRACK: A Multi-camera HD Dataset for Dense Unscripted Pedestrian Detection

Tatjana Chavdarova¹, Pierre Baqué², Stéphane Bouquet², Andrii Maksai², Cijo Jose¹, Timur Bagautdinov², Louis Lettry³, Pascal Fua², Luc Van Gool³, and François Fleuret¹

¹Machine Learning group, Idiap Research Institute & École Polytechnique Fédérale de Lausanne
²CVLab, École Polytechnique Fédérale de Lausanne
³Computer Vision Lab, ETH Zurich

{firstname.lastname}@{idiap1, epfl2, vision.ee.ethz3}.ch

Abstract

People detection methods are highly sensitive to occlusions between pedestrians, which are extremely frequent in many situations where cameras have to be mounted at a limited height. The reduction of camera prices allows for the generalization of static multi-camera set-ups. Using joint visual information from multiple synchronized cameras gives the opportunity to improve detection performance.

In this paper, we present a new large-scale and high-resolution dataset. It has been captured with seven static cameras in a public open area, and unscripted dense groups of pedestrians standing and walking. Together with the camera frames, we provide an accurate joint (extrinsic and intrinsic) calibration, as well as 7 series of 400 annotated frames for detection at a rate of 2 frames per second. This results in over 40000 bounding boxes delimiting every person present in the area of interest, for a total of more than 300 individuals.

We provide a series of benchmark results using baseline algorithms published over the recent months for multi-view detection with deep neural networks, and trajectory estimation using a non-Markovian model.

1. Introduction

Pedestrian detection is an important computer vision problem with numerous applications in security, surveillance, robotics, autonomous driving, and crowdsourcing. The variation of pedestrians appearance greatly increases the difficulty of this problem. With the availability of large-scale monocular datasets of annotated pedestrians and the advances in detection algorithms, the accuracy of the pedestrian detectors has improved significantly in the past few years. Moreover, modern detection algorithms using deep learning allow us to learn discriminative features which are transferable across datasets. Impressively, recently developed deep learning based monocular detectors are approaching human-level performance [49] on common benchmark datasets [17].

However, many situations of practical interest require detection in highly crowded and cluttered scenes. Severe occlusions make monocular pedestrian detection insufficient in these scenarios. Luckily, in real-world applications, image feeds from multiple cameras with overlapping fields of view are often available. Most commonly, the cameras are positioned slightly above the average human height. Hence designing pedestrian detectors by exploiting multiple views and the geometry of the scene will provide reliable detection estimates in crowded scenes.

Surprisingly, to nowadays standards, there is no large scale and good quality public dataset that replicate this setup. The most frequently used one, PETS 2009 [21] is only in the order of several hundreds of frames long. The provided camera calibration is inaccurate which makes it difficult to exploit geometrical constraints jointly. Moreover, it is recorded in a so-called actor set-up, meaning that throughout the sequence, a very limited total number of different individuals actually appear. All other standard datasets are either much shorter and scarcely crowded [22, 48, 7], have a very constrained scenario [3] or use non-overlapping cameras [40].

The lack of such a dataset seriously limits the development of multi-camera detection methods. Recent improve-
ments made by the community call for a more realistic and challenging benchmark that could be used to compare multi-camera detection methods. For example, [12] shows that utilizing the multi-camera input for deep learning based detectors improves both the accuracy and the classification confidence. However, these methods are severely limited by the lack of a large-scale dataset to train on without over-fitting. As a direct consequence, most of the existing joint methods use ad-hoc techniques to combine information extracted using pre-trained monocular detectors.

To help in accelerating the research on methods taking advantage of the multi-camera set-ups, we introduce a large-scale person dataset acquired with seven static cameras, with overlapping fields of view. It captures a realistic unscripted scenario where pedestrians often occlude each other. We provide a very precise joint (extrinsic and intrinsic) calibration and synchronization of sequences from the 7 views as well as 7 series of 400 annotated frames for detection at a rate of 2 frames per second. This results in over 40,000 bounding boxes delimiting every person present in the area of interest, for a total of more than 300 individuals. The annotations of the individual tracks are provided both as 3D locations on the ground plane and 2D bounding-boxes projected in each of the 7 views. Although our dataset is designed to benchmark 3D multi-camera detection, it can also be used for monocular detection. In the monocular case, the size of our dataset increases seven-fold which makes it comparable to the widely used Caltech-USA dataset [15]. Compared to the latter, our dataset has much higher image resolution.

We make the source code of our web-based annotation platform public in order to encourage other researchers to collect and annotate other multi-camera datasets. Finally, we also provide annotations for evaluating camera calibration methods.

This paper is organized as follows: in Section 2 we make a review of multi-camera person datasets and related methods. Section 3 enumerates details of the new dataset, including our camera calibration procedure. We benchmark several state-of-the-art multi-camera detection methods in Section 4 and finally in Section 5 we present potential future research directions.

2. Related work

2.1. Datasets

In Table 1 we list the commonly used pedestrian datasets with a focus on the multi-camera ones. For a more exhaustive listing of the monocular datasets, we refer the reader to [16, chap. 2.4]. As overlapping we refer to multi-camera datasets whose camera’s fields of view strictly overlap. The DukeMTMC [40] dataset does not belong to this category, as only 2 of its camera’s fields of view slightly overlap.

The most widely used dataset with an overlapping camera set-up is the PETS 2009 S2.L1 [21] sequence. In part due to the presence of a slope in the scene, the provided calibration poses large homography mapping deterioration and inconsistencies when projecting 3D points across the views (as noted also in [38, p. 10], [23, p. 10], [12, p. 3]). Besides being a small scale dataset, the PETS 2009 S2.L1 is acquired in an actor set-up. Hence it does not allow for good generalization and fair benchmarking of appearance-based methods.

The three sequences shot at the EPFL campus [22]: Laboratory, Terrace and Passageway, as well as SALSA [3] and Campus [48] are overlapping multi-camera datasets as well. However, they have a small number of total identities and are relatively sparsely crowded. As we can see from Table 1, Laboratory, Terrace and Passageway are of small size and low image quality. In SALSA [3], a cocktail party of 30 minutes is recorded, where the people are static most of the time, making this dataset less challenging for tracking. Finally, Campus does not provide the annotations of the 3D locations of the people.

The EPFL-RLC [12] dataset demonstrates improved joint-calibration accuracy and synchronization compared to PETS. However, rather than providing a complete ground-truth, this dataset represents a collection of a balanced set of positive and negative multi-view annotations and is used for classification of a position as occupied by a pedestrian or not. Full ground-truth annotations are provided solely for a small subset of the last 300 of the total 8000 frames, originally used for testing [12]. Moreover, in comparison to ours, it is acquired with only three cameras which have a much more limited field of view. This results in a ~10-fold smaller number of detections per frame on average.

Hence, WILDTRACK improves upon other multi-camera person datasets thanks to: (i) the high precision calibration and synchronisation between the cameras (see § 3.2), (ii) the large number of annotations that allows for developing deep learning based multi-view detectors. With regard to the state-of-the-art monocular datasets [15]: (i) it exceeds the total number of annotations; and (ii) the regions of interest (ROIs) are of significantly larger resolution.

2.2. Methods

Probabilistic Occupancy Map method (POM) [22] is a generative model which estimates the probabilities of occupancy on the ground plane by exploiting the geometrical constraints from multiple views. POM is formulated in the framework of mean-field inference which naturally handles the occlusions. To leverage time consistency it can be combined with a convex max-cost flow optimization [6] for tracking.

In [2], multi-view detection is re-casted as a linear in-
verse problem. Their model is regularized by enforcing a sparsity constraint on the occupancy vector. It uses a dictionary whose atoms approximate multi-view silhouettes. To alleviate the need for the time demanding Lasso-based [11] computations, a regression model is derived by [23]. This model solely comprises of Boolean arithmetic and sustains the sparsity assumption of [2]. In addition, the iterative method of [2] is replaced with a greedy algorithm based on set covering.

In [38] the occlusions are modeled explicitly per view by a separate Bayesian Network. A multi-view network is then constructed by combining them, based on the ground locations and the geometrical constraints.

Considering crowd analysis, in [22] the multi-view image generation is modeled with a stochastic generative process of random crowd configurations and then maximum a posteriori (MAP) estimate is used to find the best fit with respect to the image observations.

The Deep Multi-camera Detection (DeepMCD) [12] method which integrates CNN features demonstrated state-of-the-art results and showed that the accuracy and the confidence of a CNN classifier increase as more views are used. To mitigate the data requirement problem and improve generalization, the authors first used the larger monocular dataset Caltech [15] to train a base network. The multi-view CNN architecture is adapted so as to use the weights from the base network and produces joint estimates by processing the multi-view streams in parallel. To increase the sharpness and the localization accuracy, [12] also proposes two particular schemes of multi-view hard negative mining to train such a model.

The Deep-Occlusion Reasoning method [5] uses a joint CNN-CRF architecture and Mean-Fields inference to produce a Probabilistic Occupancy Map (POM) as in [22] while leveraging discriminative features extracted by a CNN. It introduces a Higher Order CRF, where unary potentials are produced by ROI pooling CNN [39]. Higher-order potentials are computed as a measure of the consistency between pixel labels produced by a fully convolutional network and a generative model which accounts for geometry and occlusions.

Tracking multiple objects in multiple overlapping cameras, as well as tracking in a single view, mostly follows the tracking-by-detection paradigm [4]. However, due to the scarcity of datasets with multiple overlapping cameras, tracking multiple objects in multiple overlapping cameras has received relatively little attention compared to tracking from a single view. For example, 3DMOT2015 benchmark [31] for 3D tracking lists only a couple methods that exceed a simple linear programming baseline that was proposed with the benchmark. [32] models the motion of people using social forces, and solves a linear program to es-

---

**Table 1.** Commonly used person datasets with a focus on the multi-camera ones. FPS, pos and imp stand for frames per second, positive images and image pairs, respectively. Where applicable, with †+ we denote the pre-defined splits to train and test data. See § 2.1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Resolution</th>
<th>Cameras</th>
<th>FPS</th>
<th>Mobile Static Overlapping Video</th>
<th>IDs</th>
<th>Annotations</th>
<th>Size/Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>INRIA [13]</td>
<td>high</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>No</td>
<td>-</td>
<td>1200+566</td>
</tr>
<tr>
<td>ETH [19, 20]</td>
<td>640×480</td>
<td>1</td>
<td>15</td>
<td>M</td>
<td>Yes</td>
<td>2853 (~4fps) &amp; 10398 (15fps)</td>
<td>4203 frames</td>
</tr>
<tr>
<td>TUD-Brussels</td>
<td>7720×576; 640×480;</td>
<td>1</td>
<td>M</td>
<td>n/a</td>
<td>Yes</td>
<td>-</td>
<td>1776+1326</td>
</tr>
<tr>
<td>Daimler</td>
<td>640×480</td>
<td>1†</td>
<td>15</td>
<td>M</td>
<td>Yes</td>
<td>-</td>
<td>2400+1600</td>
</tr>
<tr>
<td>Daimler-s stereo [29]</td>
<td>640×480</td>
<td>1†</td>
<td>15</td>
<td>M</td>
<td>Yes</td>
<td>-</td>
<td>3915</td>
</tr>
<tr>
<td>Caltech-USA</td>
<td>640×480</td>
<td>1</td>
<td>30</td>
<td>M</td>
<td>Yes</td>
<td>-</td>
<td>~2·10^4(10fps)+1·10^3(1fps)</td>
</tr>
<tr>
<td>APIDIS [14]</td>
<td>1600×1200</td>
<td>7</td>
<td>22</td>
<td>S</td>
<td>Yes</td>
<td>12</td>
<td>86870 (25fps)</td>
</tr>
<tr>
<td>PETS 2009</td>
<td>768×576; 720×576;</td>
<td>7</td>
<td>7</td>
<td>S</td>
<td>Yes</td>
<td>19</td>
<td>4650 (7fps)</td>
</tr>
<tr>
<td>DukeMTMC [40]</td>
<td>1920×1080</td>
<td>8</td>
<td>60</td>
<td>S</td>
<td>No</td>
<td>~2000</td>
<td>4077132 (60fps); 9668 trajectories.</td>
</tr>
<tr>
<td>Laboratory</td>
<td>320×240</td>
<td>4</td>
<td>25</td>
<td>S</td>
<td>Yes</td>
<td>6</td>
<td>476 (1fps)</td>
</tr>
<tr>
<td>Terrace [22]</td>
<td>320×240</td>
<td>4</td>
<td>25</td>
<td>S</td>
<td>Yes</td>
<td>9</td>
<td>1023 (1fps)</td>
</tr>
<tr>
<td>Passageway [22]</td>
<td>320×240</td>
<td>4</td>
<td>25</td>
<td>S</td>
<td>Yes</td>
<td>13</td>
<td>226 (1fps)</td>
</tr>
<tr>
<td>Campus [48]</td>
<td>1920×1080</td>
<td>4</td>
<td>25</td>
<td>S</td>
<td>Yes</td>
<td>25</td>
<td>240 (1fps)</td>
</tr>
<tr>
<td>SALSA [3]</td>
<td>1024×768</td>
<td>4</td>
<td>15</td>
<td>S</td>
<td>Yes</td>
<td>18</td>
<td>1200 (0.3fps)</td>
</tr>
<tr>
<td>EPFL-RLC [12]</td>
<td>1920×1080</td>
<td>3</td>
<td>60</td>
<td>S</td>
<td>Yes</td>
<td>-</td>
<td>~3·2044a;+300frames</td>
</tr>
<tr>
<td>WILDTRACK (ours)</td>
<td>1920×1080</td>
<td>7</td>
<td>60</td>
<td>S</td>
<td>Yes</td>
<td>313</td>
<td>~7×9518 (2fps)</td>
</tr>
</tbody>
</table>

* No color channels.
† 2 color and 2 grayscale cameras.
‡ Stereo camera(s).
estimate the tracks over the whole sequences. [30] uses joint probabilistic data association for online tracking. The approach of [7] can be used both for regularizing detections and tracking. It formulates tracking as a problem of finding K-Shortest paths in a graph of detections.

Some of the recently proposed approaches, such as [35] or [42] can be applied for multiple object tracking in 3D. [35] learns weights of the Recurrent Neural Network (RNN) to predict the motion of objects for data association. [42] learns several RNNs to combine motion, appearance, and social information throughout time for data association. While both of these approaches could be trained for 3D scenes, combining the appearance information from multiple cameras for [42] is non-trivial. Training the model of [35] requires a lot more annotated data than what is currently available for 3D tracking. A recently proposed approach of [34] can be used as a post-processing step to improve the results of other tracking approaches by learning the patterns of human motion in specific scenes and modifying the tracks to follow such patterns.

3. The dataset

3.1. Hardware and data acquisition

Hardware. The dataset was recorded using seven statically positioned HD cameras. In particular, we used four GoPro Hero 3 and three GoPro Hero 4 cameras (Fig. 1). All the seven sequences are of resolution 1920×1080 pixels and were recorded with a frame rate of 60 frames per second (fps). The synchronization between the seven sequences was obtained with a frame rate of 60 frames per second (fps). The synchronization between the seven sequences and were recorded with a frame rate of 60 frames per second (fps). The synchronization between the seven sequences was obtained with a frame rate of 60 frames per second (fps).

Camera placement and layout. The camera layout is highly overlapping as shown in Fig. 1, and the cameras are positioned above the humans’ average height. In Fig. 2 we illustrate from a top view perspective the amount of overlap between the fields of view of the cameras, where the darker the shading, the higher the number of cameras that capture that area. To obtain this illustration, we considered points on the ground plane and counted the number of cameras for which a given point is in their field of view. The circles in Fig. 2 denote the position of the cameras.

Data acquisition. The data acquisition took place in front of the main building of the university ETH Zurich in Switzerland, during nice weather conditions.

3.2. Calibration of the cameras

To calibrate the cameras we used the Pinhole camera model [45], due to its widespread usage and support in multiple libraries, including OpenCV [9]. Primarily, we obtained the intrinsic and the extrinsic parameters, and for the latter, we used points on the ground whose distance was measured by hand.

We put a major focus on providing jointly optimal 3D reconstruction between the cameras. To this end, we manually annotated precisely |D| = 1398 3D points by marking corresponding 2D points across the seven views and throughout multiple frames; which we used to perform Bundle adjustment [44] as we explain below.

Let I and E denote the intrinsic and the extrinsic parameters of all of the cameras, respectively. Given a dataset D of corresponding projections across the views, more precisely: \( D = \{p_i\} \), where \( p_i = \{p_i^1, \ldots, p_i^C\} \), with \( p_i^c \in \mathbb{R}^2 \) and \( C \) denoting the number of cameras, the goal is to find projection matrices \( P_c \), whose parameters are contained in \( \{I, E\} \) and the 3D points \( M = \{m_i\}, m_i \in \mathbb{R}^3, i = 1, \ldots, |D| \), such that:

\[
I^*, E^* = \arg\min_{I, E, M} \sum_{i=1}^{|D|} \sum_{c=1}^C w_c^i \|p_i - P_c m_i\|^2, \tag{1}
\]

where \( \| \cdot \| \) denotes the Euclidean image distance, and \( w_c^i \) is the indicator variable equal to 1 when the point \( p_i \) is visible in view \( c \), and equal to 0 otherwise. In other words,
we formulated the calibration as a non-linear least squares problem, where the error is the squared $L_2$ norm of the difference between the observed feature location and the projection of the corresponding 3D point on the image plane of the camera.

In accordance with the selected model, the set $\{I, E\}$ in our implementation consists of $7 \times 15$ parameters for each camera: 3 for rotation, 3 for translation, 2 for focal length ($x$ and $y$), 2 for the principal point, 3 for radial distortion and 2 for tangential distortion. To optimize Eq. 1, we used the open source C++ library Ceres \[1\] (see App. 3 for further details on the implementation).

We experimented with fixing one of the sets of parameters and sequentially optimizing each. Nonetheless, jointly optimizing $I$ and $E$ as formalised in Eq. 1 provided best results and this final step significantly improved the joint projection accuracy.

Fig. 4 depicts cropped regions taken from synchronized images and of a different view. To illustrate the precision of the camera calibration, we first manually marked a point projections in two of the views, shown in blue color in the left column of Fig. 4. Using the provided camera calibration, we then compute the 3D location of the point as an intersection between the rays defined by those projections. Finally, the resulting 3D point is projected in the remaining views, displayed as red points in Fig. 4.

### 3.3. Annotation process

To make sure that our annotations are as precise as possible, we made use of the accurate camera calibration. Namely, we formulated the annotation task as adjusting the position of a 3D cylinder on the ground, such that its projections (rectangles) across all of the views overlap the person being marked by the annotator. In summary, we considered this approach as: (i) more time-effective, as one has to perform one adjustment instead of marking bounding boxes and adjusting each separately; as well as that it allows for obtaining (ii) higher precision, since the position is jointly adjusted. The latter follows from the fact that the best position of a bounding box when annotating in one view is likely to be ambiguous, and in cases of more severe occlusions, it could also be infeasible to guess it.

To this end, we designed a new multi-camera annotation tool, whose Graphical User Interface (GUI) is illustrated in Fig. 5. It was written using Python, Django, and Javascript. The tool was deployed through Amazon Mechanical Turk \[10\] so that external people would be remunerated to help us carry out this time-consuming procedure. To ensure that profit is not prioritized over the accuracy and the precision of the annotations, we were highly involved in the process, and particular annotators were carefully selected. For details regarding how the annotation process was carried out, and the file format of the annotations, please refer to App. 1 and App. 2, respectively.
3.4. Statistics

Annotated frames. Following practices amassed in monocular pedestrian detection with CNNs, we used frame rate of 10 fps to extract the frames. Herein, as a frame $I_t$ we refer to a set of synchronized images from the seven views $(C=7)$, i.e. $\{I^1_t, \ldots, I^C_t\}$. We provide annotations for the first 2000 frames. However, in our observations annotating every 5th suffices the speed of movement of the persons, and hence interpolation can be used to further enlarge the dataset. To summarize, this amounts to total of $400 \times 400$ manually annotated images. Below we always list total number of annotations at 2 fps.

Multi-view annotations. There are 9518 multi-view single person annotations in total. In Fig. 6, we illustrate an example of a multi-view annotation of our dataset, visible in all of the seven views at the same time. On average, each frame $I_t$ captures 23.8 people. Considering a frame rate of 2 fps, each person is seen in 30.41(47.87) frames, with a mode of 22 frames (see App. 4).

Monocular annotations. Since a multi-view detection may not be visible in each of the cameras’ fields of view, the number of monocular examples is precisely $8731, 7875, 6703, 2239, 3920, 9408, 3731$, respectively for each view. This amounts to a total of 42607 detections at 2 fps.

Occlusion levels. As the annotations are obtained in 3D, the occlusion of each pedestrian in each view can be automatically obtained. Following the work of [15] in Fig. 3 we illustrate the level of occlusions per each view separately, by calculating the number of occluded pixels over the number of total pixels for each detection. Similarly, we calculate such normalized occlusion levels for the multi-view examples across all of the views. As can be noticed from Fig. 3, if multi-view samples are used, the probability that a person will be completely occluded in all of the views simultaneously significantly goes down.

4. Benchmarks

In the sequel, we evaluate detection and tracking methods. In our experiments, we used a frame rate of 2 fps.

4.1. Evaluation protocol

We compute false positives (FP), false negatives (FN) and true positives (TP) by assigning detections to ground truth using Hungarian matching. Since we operate on the ground plane, we impose that a detection can be assigned to a ground truth annotation if and only if it is closer than a distance $r$. Given FP, FN and TP, we calculate:

- **Multiple Object Detection Accuracy (MODA)** accounting for the normalized missed detections and false positives, as well as the **Multiple Object Detection Precision (MODP)** metric which assesses the localization precision [28].

- **Precision & Recall.** We estimate the empirical precision and recall, calculated by $P = TP/(TP + FP)$ and $R = TP/(TP + FN)$ respectively.

We report MODP, Precision, and Recall for radius $r=0.5m$, which roughly corresponds to the width of a human body.

Unless otherwise emphasized, the used metrics are always calculated in terms of the Euclidean distance between the detection and the annotation on the ground plane in world coordinates, or alternatively from top-view. Thus note that such metrics are unforgiving in terms of projection errors as we measure distances in the ground plane, which would not be the case if we evaluated overlap in the image plane as is often done in the monocular case.

Regarding multiple object tracking, we report metrics which are also reported for MOTChallenge benchmark [31]: a set of CLEAR MOT [8] metrics, as well as identity-aware metrics of [41]. For evaluation we used the devkit provided with [31], and we similarly report metrics for radius $r=1m$. Below we briefly review some of the reported tracking metrics:

- **Multiple Object Tracking Accuracy (MOTA)** accounts for missed detections (False Negatives (FN)),
False Positives (FP), and Identity Switches (IDs) between current and next frame of tracking. We also report Multiple Object Tracking Precision (MOTP).

- Most Tracked (MT), Partially Tracked (PT), Mostly Lost (MT) trajectories, as well as the number of Fragmentations (FM).

- IDF1 accounts for missed detections, false positives, and identity switches throughout tracking, after finding a global one-to-one assignment between ground truth and predicted trajectories. It is calculated similarly to F1 score, using the Identity Precision (IDP) and Identity Recall (IDR) as proxies for precision and recall.

4.2. Evaluated methods

We evaluated the following detection methods:

- POM-CNN. The multi-camera detector [22] described in § 2.2, uses background subtraction pre-processing and takes such segmented images as input. In its original implementation the input is obtained using traditional algorithms [50, 36]. Hence, for a fair comparison reflecting the progress that has occurred since then, we use a CNN-based background subtraction [33].

- DeepMCD (described in § 2.2), which is an end-to-end deep learning method. We used its implementation that uses GoogLeNet [43] by: (i) testing on the WILDTRACK dataset with the provided pre-trained model on the PETS dataset–Pre-DeepMCD; as well as (ii) training solely the top classifier on the WILDTRACK dataset–TopDeepMCD. Its implementation uses monocular Non Maxima Suppression (NMS) and the performance measures are calculated using the first view, rather than the world coordinates.

- ResNet-DeepMCD & DenseNet-DeepMCD are our implementations of DeepMCD in PyTorch [37], which use ResNet-18 [26] and DenseNet-121 [27], respectively. As WILDTRACK is of larger size, we omitted the step of pre-training on the Caltech dataset. Our implementation uses top-view NMS to select the final detections of the candidates while prioritizing the detections with higher probability estimates, as opposed to [12]. We use NMS threshold of 0.4 and $r=0.5$m. We used 90% and 10% of the frames for training and testing, respectively and grid density of 60×180. Analogously, we also train a monocular detector ResNet-18 [26] while using samples from the training frames from all of the views, denoted as ResNet-Monocular.

- Deep-Occlusion (see § 2.2), which is a hybrid CNN-CRF method to use information about geometry and calibration while leveraging on the discriminative power of a pre-trained monocular CNNs.

- RCNN-projected. The recent work of [47] proposes a MCMT tracking framework that relies on a powerful CNN for detection purposes [39]. Since the code of [47] is not publicly available, we reimplemented their detection methodology without the tracking component for a fair comparison with the detection methods that operate on images acquired at the same time. More precisely, we first run the 2D detector proposed by [39] on each image. We then project the bottom of the 2D bounding box onto the ground reference frame as in [47] to get 3D ground coordinates. Finally, we cluster all the detections from all the cameras using 3D proximity to produce the final set of detections.

For multiple object tracking, we provide results of the following methods:

- KSP is a simple baseline approach of [7] which finds the most likely sequence of occupancies of ground floor locations of POM given probabilities of occupancies of each location given by a detector. To do so it solves the problem of finding the most likely trajectories given the detections as finding the K-Shortest Paths in a graph. This method does not use appearance information for tracking, but is a suitable baseline as it can be applied out of the box in multi-camera scenario.

- ptrack is a recent approach of [34] which improves the results of other tracking approaches by learning the motion patterns of the scene and modifying the tracks of people so as to follow those patterns. This is suitable for our scenario both because in our scene there are several clearly identifiable patterns of motion, as well as because it can handle tracking in the ground plane. We therefore use this approach on top of the trajectories found by KSP.

4.3. Benchmark results

Detection. The results of the enumerated methods in § 4.2 are given in Tab. 3. We observe that joint utilization of the multiple views largely improves detection performance. More explicit occlusion reasoning further improves the MODA metric, even for small values of $r$, Fig. 7. In Tab. 4 we list the classification results obtained when training a monocular ResNet-18 using samples extracted from all of the seven views, as well as multi-view training. The results indicate that even though the size of the dataset is seven-fold larger for the monocular case, using the multiple views increases both the accuracy and the confidence of the classifier.
Table 2. Benchmark tracking results on the WILDTRACK dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>IDF1</th>
<th>IDP</th>
<th>IDR</th>
<th>MT</th>
<th>PT</th>
<th>ML</th>
<th>FP</th>
<th>FN</th>
<th>IDs</th>
<th>FM</th>
<th>MOT</th>
<th>MOTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepOcclusion+KSP</td>
<td>73.2</td>
<td>83.8</td>
<td>65.0</td>
<td>49</td>
<td>79</td>
<td>43</td>
<td>1095</td>
<td>7503</td>
<td>85</td>
<td>92</td>
<td>69.6</td>
<td>61.5</td>
</tr>
<tr>
<td>DeepOcclusion+KSP+ptrack</td>
<td>78.4</td>
<td>84.4</td>
<td>73.1</td>
<td>72</td>
<td>74</td>
<td>25</td>
<td>2007</td>
<td>5830</td>
<td>103</td>
<td>95</td>
<td>72.2</td>
<td>60.3</td>
</tr>
</tbody>
</table>

Table 3. Benchmark detection results on WILDTRACK.

<table>
<thead>
<tr>
<th>Method</th>
<th>MODA</th>
<th>MODP</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep-Occlusion+KSP</td>
<td>0.752</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Deep-Occlusion</td>
<td>0.741</td>
<td>0.538</td>
<td>0.95</td>
<td>0.80</td>
</tr>
<tr>
<td>ResNet-DeepMCD</td>
<td>0.678</td>
<td>0.642</td>
<td>0.85</td>
<td>0.82</td>
</tr>
<tr>
<td>DenseNet-DeepMCD</td>
<td>0.635</td>
<td>0.666</td>
<td>0.87</td>
<td>0.74</td>
</tr>
<tr>
<td>POM-CNN</td>
<td>0.232</td>
<td>0.305</td>
<td>0.75</td>
<td>0.55</td>
</tr>
<tr>
<td>RCNN-projected</td>
<td>0.113</td>
<td>0.184</td>
<td>0.68</td>
<td>0.43</td>
</tr>
<tr>
<td>Pre-DeepMCD</td>
<td>0.334*</td>
<td>0.528*</td>
<td>0.93</td>
<td>0.36</td>
</tr>
<tr>
<td>Top-DeepMCD</td>
<td>0.601*</td>
<td>0.642*</td>
<td>0.80</td>
<td>0.79</td>
</tr>
<tr>
<td>ResNet-View 1</td>
<td>-1.823</td>
<td>0.598</td>
<td>0.26</td>
<td>1.00</td>
</tr>
<tr>
<td>ResNet-View 2</td>
<td>-1.050</td>
<td>0.607</td>
<td>0.32</td>
<td>0.99</td>
</tr>
<tr>
<td>ResNet-View 3</td>
<td>-1.036</td>
<td>0.583</td>
<td>0.32</td>
<td>0.98</td>
</tr>
<tr>
<td>ResNet-View 4</td>
<td>-0.251</td>
<td>0.723</td>
<td>0.42</td>
<td>0.71</td>
</tr>
<tr>
<td>ResNet-View 5</td>
<td>0.466</td>
<td>0.623</td>
<td>0.67</td>
<td>0.91</td>
</tr>
<tr>
<td>ResNet-View 6</td>
<td>-1.841</td>
<td>0.591</td>
<td>0.26</td>
<td>1.00</td>
</tr>
<tr>
<td>ResNet-View 7</td>
<td>-0.122</td>
<td>0.701</td>
<td>0.47</td>
<td>0.97</td>
</tr>
</tbody>
</table>

* Monocular calculation of the metric, using the first view.

Table 4. Classification results (accuracy and area under ROC curve) on the test frames of the WILDTRACK dataset using ResNet-18 [26]. The results obtained for monocular classification are averaged over all of the views.

<table>
<thead>
<tr>
<th>Training</th>
<th>Accuracy (%)</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monocular</td>
<td>84.57 (1.718)</td>
<td>0.91 (0.028)</td>
</tr>
<tr>
<td>Multi-view</td>
<td>95</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Tracking. Tab. 2 lists benchmark results of the tracking methods described in § 4.2 (see App. 6 for additional results). Parameters of the methods were optimized for IDF1 metric on the same training data as the appropriate detector was trained on. As shown, the dataset presents a significant tracking challenge, with IDF1 metric results lower than those seen on the [31] benchmark on [40] dataset, where each individual camera mostly observes a separate, and somewhat simpler scene.

5. Discussion

The development of new multi-view people tracking methods is hampered by the surprising lack of appropriate datasets. We provide a new large scale, high-resolution, and highly accurately calibrated multi-camera pedestrian dataset, which is more realistic than any of the previously published ones.

Figure 7. Benchmark of the multi-view detectors using the MODA metric (y-axis) for different radius r (x-axis) on WILDTRACK.

Our initial benchmarks show that deep learning person detection indeed largely benefits from a multi-camera set-up and this motivates further work in that direction. Our dataset will motivate and facilitate such research. While performance of monocular pedestrian detectors saturates on common benchmarks, our densely crowded realistic set-up, which results in complex dynamics and constant occlusions among persons, and the high resolution, will prove useful for further improving monocular detection, as well as other problem of inference such as tracking, or crowd analysis at large.

Finally, in addition to the large number of individual detections, the WILDTRACK dataset also has a large fraction of unlabelled frames. This will be precious for unsupervised methods, with the possibility to be benchmarked on the annotated portion.

Acknowledgment

This work was supported by the Swiss National Science Foundation, under grant CRSII2-147693 “WILDTRACK” and by the Hasler Foundation through the “MEMUDE” project. We gratefully acknowledge NVIDIA’s support through their academic GPU grant program. We also thank Salim Kayal and Florent Monay for their help and insightful discussions regarding the calibration of the cameras.
References


Foremost, we elaborate in more detail the annotation procedure in Section 1. We then list details regarding the provided annotations in Section 2. Details on our implementation of the camera calibration are given in Section 3. Section 4 provides further details of the statistics summarized in Section 3 of the paper. In Section 5 we discuss recommended training and testing splits of the WILDTRACK dataset. Finally, we provide additional tracking results in Section 6.

1 Annotation process

Annotation tool. As an area of interest we consider a $12 \times 36m$ ground plane of the 3D space lying in the intersection of the fields of view of the seven cameras. We discretize this ground surface as a grid of 480×1440 points, what corresponds to an offset of 2.5cm in both directions. Given such a regular high-density grid of, at each location we construct a cylinder volume whose height and width correspond to the humans’ average height and width. Each such cylinder projects into the separate 2D views as a rectangle. The position of these rectangles in all of the views is then calculated in pixel coordinates using the camera calibration. Finally, we use these pre-calculated projections to integrate them into our annotation tool.

The labelling tool is a Python-based web application. It is built with a very responsive design, and its graphical user interface (GUI) is illustrated in Fig. 1. Our annotation tool is hosted on a website, which was created and managed using Django. The source-code is also available for download.

For the selected frame to be annotated, the tool displays the seven corresponding images at the same time (see Fig. 1). In order to provide a multi-view annotation, the user of the tool first has to mark the placement of the bounding boxes. This is achieved by a single click, whose location should be at the feet of the person to be annotated, in either of the views where it is visible. Instantaneously, the boxes automatically appear in the views in which the person is visible. To complete the multi-view annotation, the user shall next adjust the position of the bounding boxes.

For this purpose, the keyboard arrows shall be used. More precisely, the left, right, up and down keys should be used in order to shift the 3D imaginary cylinder on the ground plane. To help annotators, the correspondence “key-direction” for each of the views is also depicted in the tool and optionally visible while annotating. In addition, a zooming feature can be used, that once a multi-view annotation is selected, allows for zooming-in the corresponding bounding boxes. This was implemented in order to make it easier for the annotators to obtain more precise locations of the annotations. The arrow key presses that translate into a movement of the 3D cylinder, are instantly visible in all of the views that capture the person currently being annotated. After getting used to the annotation process, annotators become more and more precise on the first step: placing the bounding boxes, which significantly reduces the time required to annotate as less adjustments are required.

1 https://pedestrian_tag.epfl.ch/
2 https://github.com/cvlab-epfl/multicam-gt
Once the frame has been fully labelled and the user has moved to the next frame, optionally (s)he is able to reload the annotations from the previous frame, traverse each of the annotations, and refine their positions. Additional features such as keyboard short-cuts are also supported for these utilities.

Finally, a more elaborate version of these instructions is provided in the annotation tool, accompanied with numerous illustrations.

**Annotating on Mechanical Turk.** We used Amazon Mechanical Turk [3] to obtain our annotations. Due to the risk of the annotators prioritizing profit over quality of the annotations, we were highly involved in the process.

In our experience of annotating frames of our dataset, pre-loading annotations from the previous frame, traversing these, and adjusting each, often proves less time-consuming then starting to annotate each multi-view frame from scratch. This motivated providing the feature of pre-loading annotations explained above. Hence, to help accelerate the annotation process the recruited annotators were assigned frames in batches of size 10. To ensure that this feature is not negatively utilised by the annotators, we also store flags indicating if these “imported” annotations have been adjusted or not.

As explained, annotators were found via Mechanical Turk. However, since the dataset is quite challenging, annotating locations in 3D for crowded scenes may require substantial attention and dedication. Despite all our efforts to make the tool easy to use, it turned out that most MT workers were reluctant to provide this level of effort and they were almost never achieving the required quality. We therefore had to select few workers to whom we personally explained the level of detail needed. They were then able to annotate with higher accuracy.

On average, annotating one frame takes $\sim 10$ minutes for a trained annotator, and approximately half of that when importing the annotations from the previous frame.

## 2 Annotations

### 2.1 File formats

The annotations are provided in a separate file per each multi-view frame. Each annotation file is provided in the JavaScript Object Notation (JSON) open-standard file format. This format is human readable and programming language independent. Many programming languages integrate libraries that offer support for working with these files, including Python.

Each multi-view annotation contains the following information:
- **Person ID**: A unique identifier of a person appearing in the sequence.
- **3D location**: \((X, Y)\) location of the target in meters on the ground plane with respect to the origin.
- **pixel coordinates in each of the views**: For each of the seven cameras, the detection location in pixel coordinates for that view are given: \(\{(x_{c \min}^{c}, y_{c \min}^{c}, x_{c \max}^{c}, y_{c \max}^{c})\}, c = 1, \ldots, 7\).

2.2 Memory size

**Images.** We refer as a frame a set of 7 images, synchronized with the same time stamp. The extracted and pre-processed frames with removed distortions contain 36000 × 7 images, while each image is of size \(\sim 2.9\) MB. This corresponds to 10 frames per second for 1h and 7 cameras. Currently there are 400 annotated frames, at 2fps (see Section 3.4).

**Videos.** Each of the 7 videos is approximately 1:50h long, and of size \(\sim 25GB\).

3 Camera calibration

**Intrinsic.** The intrinsic calibration was obtained for each camera separately. For this purpose we used the OpenCV function `calibrateCamera` which provides also the distortion coefficients. Precisely, we used 3 radial distortion coefficients. In particular, we used the asymmetric circle grid provided by OpenCV with sizes of 4 × 11, and 20 frames to obtain each camera’s intrinsic matrix. To obtain higher accuracy, we made sure that the target (the grid of circles), is captured in as many parts of the field of view of the camera as possible.

**Extrinsic.** In our implementation, for each of the seven views we used 23, 26, 15, 19, 21, 28 and 19 pairs of points, respectively. We used the OpenCV’s module `solvePnP` [2], which given the intrinsics provides the rotation and the translation vector. The 3D measurements and the annotated corresponding points will also be made available, so as camera calibration methods could make use of these.

**Bundle adjustment.** In our implementation, we used the open source C++ library Ceres [1], which offers extensive support for bundle adjustment problems. We used linear optimisation which in Ceres is referred to as *Iterative Schur*.

4 Additional statistics

![Figure 2: Histogram of the number of frames in which one person appears: the normalized number of different identities (y-axis) that appear within a range of number of frames (x-axis).](image)

Fig. 2 depicts the number frames in which a person appears. In particular, we consider a frame rate of 2 fps, a total number of frames of 400, and 313 different identities. On average, each person appears in 30.41(47.87) frames, and the mode is 22 frames.
5 Recommended splits of the WILDTRACK dataset

We regard two use-cases of the WILDTRACK dataset, and we discuss recommended partitions for each. Please consider visiting the website for downloading the dataset\(^3\), for up to date details.

**Scenario A: Supervised methods.** We recommend that the last 10% of the annotated frames at 2 fps are used for testing. This amounts to a total of 40 frames at 2 fps. For training one shall use the remaining portion of the dataset, with optional sampling frame rate.

**Scenario B: Unsupervised methods.** In this case, we recommend that the entire annotated portion at a fixed frame rate of 2 fps is used for benchmarking unsupervised methods. The remaining portion for which annotations are not provided can be used for training, using an optional sampling rate.

6 Additional tracking benchmarks

Table 1 shows additional tracking results, where we use the same notation for the methods as in the paper (see Section 4.2 in the paper).

<table>
<thead>
<tr>
<th>Method</th>
<th>IDF1</th>
<th>IDP</th>
<th>IDR</th>
<th>MT</th>
<th>PT</th>
<th>ML</th>
<th>FP</th>
<th>FN</th>
<th>Ids</th>
<th>FM</th>
<th>MOTA</th>
<th>MOTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-DeepMCD+KSP</td>
<td>62.5</td>
<td>84.9</td>
<td>49.5</td>
<td>11</td>
<td>11</td>
<td>19</td>
<td>154</td>
<td>2081</td>
<td>35</td>
<td>30</td>
<td>50.9</td>
<td>75.1</td>
</tr>
<tr>
<td>ResNet-DeepMCD+KSP+ptrack</td>
<td>64.2</td>
<td>93.1</td>
<td>49.0</td>
<td>10</td>
<td>9</td>
<td>22</td>
<td>49</td>
<td>2239</td>
<td>5</td>
<td>5</td>
<td>50.4</td>
<td>75.9</td>
</tr>
<tr>
<td>ResNet-View 1+KSP</td>
<td>28.5</td>
<td>18.7</td>
<td>60.7</td>
<td>34</td>
<td>4</td>
<td>0</td>
<td>10050</td>
<td>257</td>
<td>162</td>
<td>58</td>
<td>-140.9</td>
<td>59.0</td>
</tr>
<tr>
<td>ResNet-View 1+KSP+ptrack</td>
<td>30.2</td>
<td>20.1</td>
<td>60.6</td>
<td>30</td>
<td>8</td>
<td>0</td>
<td>9173</td>
<td>410</td>
<td>131</td>
<td>51</td>
<td>-123.5</td>
<td>58.0</td>
</tr>
<tr>
<td>ResNet-View 2+KSP</td>
<td>29.1</td>
<td>19.4</td>
<td>58.8</td>
<td>28</td>
<td>6</td>
<td>1</td>
<td>8428</td>
<td>265</td>
<td>172</td>
<td>47</td>
<td>-121.3</td>
<td>50.7</td>
</tr>
<tr>
<td>ResNet-View 2+KSP+ptrack</td>
<td>31.4</td>
<td>21.2</td>
<td>60.6</td>
<td>28</td>
<td>6</td>
<td>1</td>
<td>7698</td>
<td>249</td>
<td>128</td>
<td>31</td>
<td>-101.6</td>
<td>50.2</td>
</tr>
<tr>
<td>ResNet-View 3+KSP</td>
<td>25.8</td>
<td>17.0</td>
<td>53.7</td>
<td>35</td>
<td>4</td>
<td>1</td>
<td>9874</td>
<td>286</td>
<td>177</td>
<td>52</td>
<td>-133.5</td>
<td>51.2</td>
</tr>
<tr>
<td>ResNet-View 3+KSP+ptrack</td>
<td>27.2</td>
<td>18.1</td>
<td>54.2</td>
<td>33</td>
<td>5</td>
<td>2</td>
<td>9208</td>
<td>402</td>
<td>150</td>
<td>46</td>
<td>-120.5</td>
<td>49.1</td>
</tr>
<tr>
<td>ResNet-View 4+KSP</td>
<td>20.5</td>
<td>12.6</td>
<td>54.4</td>
<td>14</td>
<td>5</td>
<td>1</td>
<td>4362</td>
<td>137</td>
<td>42</td>
<td>16</td>
<td>-255.3</td>
<td>60.5</td>
</tr>
<tr>
<td>ResNet-View 4+KSP+ptrack</td>
<td>22.1</td>
<td>13.9</td>
<td>54.4</td>
<td>13</td>
<td>2</td>
<td>5</td>
<td>3904</td>
<td>180</td>
<td>32</td>
<td>11</td>
<td>-222.1</td>
<td>60.3</td>
</tr>
<tr>
<td>ResNet-View 5+KSP</td>
<td>39.7</td>
<td>32.6</td>
<td>50.9</td>
<td>20</td>
<td>13</td>
<td>3</td>
<td>2560</td>
<td>598</td>
<td>117</td>
<td>79</td>
<td>5.8</td>
<td>54.2</td>
</tr>
<tr>
<td>ResNet-View 5+KSP+ptrack</td>
<td>41.7</td>
<td>35.0</td>
<td>51.7</td>
<td>18</td>
<td>12</td>
<td>6</td>
<td>2334</td>
<td>672</td>
<td>94</td>
<td>54</td>
<td>10.9</td>
<td>55.2</td>
</tr>
<tr>
<td>ResNet-View 6+KSP</td>
<td>26.6</td>
<td>17.5</td>
<td>55.4</td>
<td>34</td>
<td>4</td>
<td>1</td>
<td>10200</td>
<td>375</td>
<td>172</td>
<td>77</td>
<td>-136.1</td>
<td>52.2</td>
</tr>
<tr>
<td>ResNet-View 6+KSP+ptrack</td>
<td>29.4</td>
<td>19.9</td>
<td>56.4</td>
<td>30</td>
<td>8</td>
<td>1</td>
<td>8860</td>
<td>498</td>
<td>127</td>
<td>51</td>
<td>-108.4</td>
<td>52.8</td>
</tr>
<tr>
<td>ResNet-View 7+KSP</td>
<td>38.6</td>
<td>27.1</td>
<td>67.0</td>
<td>22</td>
<td>3</td>
<td>0</td>
<td>4488</td>
<td>171</td>
<td>72</td>
<td>28</td>
<td>-61.1</td>
<td>65.1</td>
</tr>
<tr>
<td>ResNet-View 7+KSP+ptrack</td>
<td>41.7</td>
<td>30.3</td>
<td>66.8</td>
<td>19</td>
<td>3</td>
<td>3</td>
<td>3791</td>
<td>253</td>
<td>49</td>
<td>21</td>
<td>-39.4</td>
<td>64.9</td>
</tr>
</tbody>
</table>

References


\(^3\)https://cvlab.epfl.ch/data/wildtrack