Scientific Research Publishing

Open Journal of Modelling and Simulation, 2019, 7, 1-18
http://www.scirp.org/journal/ojmsi
ISSN Online: 2327-4026
ISSN Print: 2327-4018

# Computer Model for Evaluating Multi-Target Tracking Algorithms

**Garret Vo, Chiwoo Park**

Department of Industrial and Manufacturing Engineering, Florida State University, Tallahassee, FL, USA
Email: gdv12@my.fsu.edu, cpark5@fsu.edu

**How to cite this paper:** Vo, G. and Park, C. (2019) Computer Model for Evaluating Multi-Target Tracking Algorithms. *Open Journal of Modelling and Simulation*, **7**, 1-18.
https://doi.org/10.4236/ojmsi.2019.71001

**Received:** October 8, 2018
**Accepted:** November 5, 2018
**Published:** November 8, 2018



## Abstract

Public benchmark datasets have been widely used to evaluate multi-target tracking algorithms. Ideally, the benchmark datasets should include the video scenes of all scenarios that need to be tested. However, a limited amount of the currently available benchmark datasets does not comprehensively cover all necessary test scenarios. This limits the evaluation of multitarget tracking algorithms with various test scenarios. This paper introduced a computer simulation model that generates benchmark datasets for evaluating multi-target tracking algorithms with the complexity of multitarget tracking scenarios directly controlled by simulation inputs such as target birth and death rates, target movement, the rates of target merges and splits, target appearances, and image noise types and levels. The simulation model generated a simulated video and also provides the ground-truth target tracking for the simulated video, so the evaluation of multitarget tracking algorithms can be easily performed without any manual video annotation process. We demonstrated the use of the proposed simulation model for evaluating tracking-by-detection algorithms and filtering-based tracking algorithms.

## Keywords

Performance Evaluation, Multi-Target Tracking, Computer Model, Simulation

## 1. Introduction

A multi-target tracking problem is to estimate the trajectory of multiple targets as they move and interact with each other in a sequence of images, estimating targets' locations and velocities [1] [2]. A multi-target tracking problem was driven primarily by aerospace and defense applications, such as radar, sonar, guidance, and navigation [1] [3] [4]. With the advancement in high performance

computing and the availability of inexpensive sensors and cameras, multi-target tracking problem has become popular, and it has grown into an established discipline [5] [6] [7]. Currently multi-target tracking algorithms find many applications in computer vision [2], oceanography [8] [9], robotics [10] [11], and remote sensing [12].

Many multi-target tracking approaches have been extensively studied. A popular approach is a tracking-by-detection method that uses an existing target detection algorithm to detect targets in images and solves an optimization problem for associating and tracing the detected targets over a time horizon [13]-[22]. Another popular approach is a filtering-based approach, which explicitly models the behavior of targets with a linear or non-linear state space model and estimates the system states (typically target locations) using Kalman filtering [23] [24] [25], particle filtering [26] [27] [28] or other MCMC samplers [29] [30] [31] [32].

The performances of the existing approaches vary depending on the complexities and types of the video scenes which they applied to, and there is no single method that universally works best for all test videos. It is often very difficult to choose a good tracking algorithm among many algorithms that can handle the video complexities existing in an application of interest. The best way of evaluating and choosing a multitarget tracking algorithm is to use test video datasets collected directly from the application of interest. However, the evaluation with the test video is often very time-consuming because mostly no ground-truth is available for the test video datasets, for which users may need to spend significant time on manually tracking and annotating targets in the test video. The manual annotation is subject to many human operators' errors. Using public benchmark datasets coming with ground-truth, such as PETS [21] [33] [34], ETH dataset [35] [36], and Technical University of Darmstadt (TUD) dataset [37] [38] [39], might be an alternative, but the benchmark datasets sometimes do not give any good guidance on the evaluation, because the types of targets and the complexities of the tracking events in the public benchmark datasets are not comparable to those existing in the application of interest. In addition, the ground-truth of the public benchmark datasets was manually annotated by human operators using video annotation tools, and the quality and information in the ground-truth vary significantly [40].

In this paper we propose a simulation model that generates benchmark data for multitarget tracking with a fully controlled setting of target appearances, motion, target split and merge, target birth and death, and noise level in video scenes. Each generated benchmark data include a simulated video, and the groud truth target tracking, including individual target locations and the time traces of their merge and split events. We believe that using the benchmark dataset simulated with the complexity comparable to the application of interest would lead to a more accurate evaluation of the existing multitarget tracking algorithms and thus a better choice of the algorithm for the application.

The remaining of this paper is organized as follows. In Section 2, we review the public benchmark datasets popularly used for multitarget tracking, and the performance metrics that have been used for evaluating multitarget tracking algorithms. In Section 3, we describe our simulation model. In Sections 4 and 5, we present an example of applying the simulator for evaluating a multitarget tracking algorithm.

## 2. Limitation of Public Benchmark Datasets

There are public benchmark datasets in computer vision for testing and evaluating multitarget tracking algorithms. This section will briefly review some of the popular benchmark datasets as listed below:

- Performance Evaluation of Tracking and Surveillance (PETS) [21] [33] [34];
- Technical University of Darmstadt (TUD) [37] [38] [39].
- ETH dataset:
- BIWI Walking Pedestrian dataset [35] [36];
- Pedestrian Mobile Scene Analysis [41] [42] [43];
- Caviar [33] [44] [45].

The first dataset has been provided as a part of the paper competition for the International Workshop on Performance Evaluation of Tracking and Surveillance. Every year different test video scenarios are provided, which are mostly for video surveillance. For example, PETS 2000 and PETS 2001 datasets are designed for tracking outdoor people and vehicles. PETS-ECCV 2004 has a number of video clips recorded for the CAVIAR project, including people walking alone, meeting with others, window shopping, fighting and passing out, and leaving a package in a public place. The PETS 2006 dataset is the surveillance data of public spaces for detecting left luggage events. The PETS 2007 dataset considers both volume crime (theft) and a threat scenario (unattended luggage). The datasets for PETS 2009, PETS 2010 and PETS 2012 consider crowd image analysis and include crowd count and density estimation, tracking of individual(s) within a crowd and detection of separate flows and specific crowd events. Many of the datasets consists of three or four video scenarios categorized by the levels of object tracking difficulties.

The second dataset is maintained by Technical University of Darmstadt (TUD) in Germany. Video frames in TUD dataset are taken by a single camera, which include three video sequences of multiple pedestrians at different places. The splits and merges among objects often occur due to overlapping pedestrian images from a single camera view.

The third dataset is maintained by the Department of Electrical and Computer Engineering at ETH. There are two popular data in the ETH dataset. The first one is a video of walking pedestrians. The video is taken from the top of a hotel; therefore, it has a bird-eye-view. Similar to the dataset in the TUD, this dataset is taken with a single camera. Due to the limited view angle of the single camera, the appearance and disappearance of targets occur almost every frame in the da-

ta. The merging and splitting event is also differentiable in this data, and they appear less than the TUD dataset (about one per four frames). The second data from the ETH is the video of walking pedestrians recorded from the frontal viewpoint instead of the bird-eye perspective. This data is mainly used for detection purpose. However, it can be used for testing multi-target tracking algorithms as well. Similar to the first data, this data is also taken by a single camera; the birth and death of targets occur every frame, and the merging and splitting events do not occur at all. Comparing to both the PETS and TUD datasets, the ETH datasets have simpler split and merge patterns among targets. If the multi-target tracking algorithm focuses on handling the birth and death of targets only, the ETH dataset will be an ideal choice for testing the algorithm.

The last dataset is the Caviar dataset. It is maintained by the computer vision research group at University of Edinburgh. Comparing to previous datasets, this dataset gives a fewer number of targets, one person or two people per scene. Target merges, splits, births and deaths sequentially occurs. The Caviar datasets have lower level of difficulty than the aforementioned datasets, because there are only two targets per frame. With a known number of targets (*i.e.*, two people), and sequential events, the Caviar dataset is a good choice to test multi-target tracking algorithms at the first time before testing with PETS, TUD, or ETH datasets.

Although users can test and evaluate multitarget tracking algorithms with these datasets, they are unable to modify these datasets in order to produce the similar complexity of video scenarios that exist in the multitarget tracking application of their interest. The simulation model proposed in this paper can overcome this challenge. Our simulation model allows users to control the target's appearance, number of targets, and target's motion. In addition, users can generate different events, such as merging, splitting, birth and death as well as control the frequency of these events.

## 3. Discrete Event Simulator for Generating Multitarget Tracking Benchmark

As depicted in Figure 1, the simulator for generating benchmark data is a discrete event simulator which is described by a system state $\mathbf{S}_t$ at time $t$ and its state transition events with discrete time steps $t \in \{0,1,2,\cdots,T\}$. The system state $\mathbf{S}_t$ is a collection of the state vectors for each target $i$,

- $\left(x_{i,t}, y_{i,t}\right)$: a centroid coordinate of target $i$,
- $\mathbf{B}_i$: a vector of a finite number of $(x, y)$-coordinates that represents the outline of target $i$,
- $\theta_{i,t}$: a rotation angle of the appearance of target $i$,

and the system-level state variables of,

- $n_t$: the number of targets at time $t$, and
- $\mathbf{G}_t$: a symmetric matrix for merging states; $\left(\mathbf{G}_t\right)_{ij} = 1$ implies targets $i$ and $j$ are merged at time $t$ and $\left(\mathbf{G}_t\right)_{ij} = 0$ otherwise.

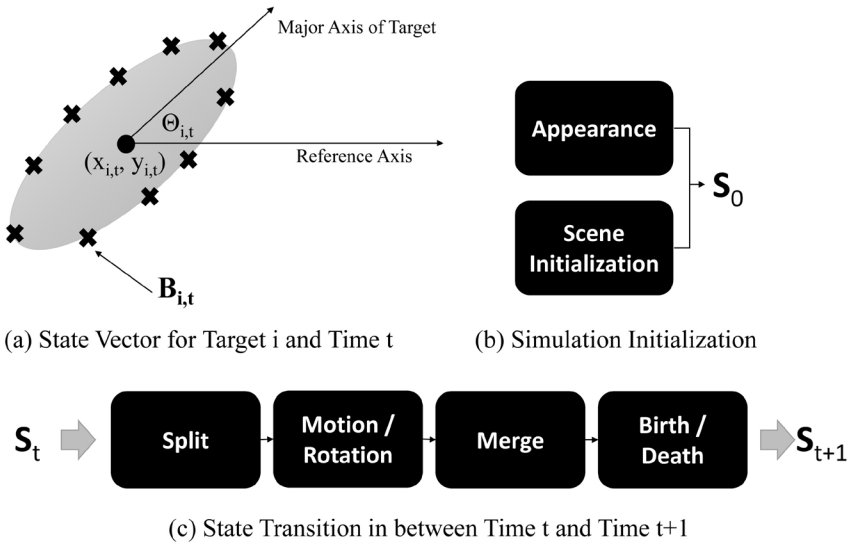



**Figure 1.** Discrete event simulation model for generating benchmark datasets.

For time $t$, a synthetic image $\mathbf{I}_t$ is generated from the system state $\mathbf{S}_t$ with different imaging conditions; see details in Section 3.7. The system state $\mathbf{S}_t$ is recorded and used to generate the groudtruth of the simulated image sequence.

At time 0, the system state variables are initialized with $n_0$ specified by a simulator user, $\mathbf{G}_0$ as a $n_0 \times n_0$ matrix of zeros, and the target-level state vectors initialized randomly by the following $q$ functions,

$$\left(x_{i,0}, y_{i,0}\right) = q_{x,y}\left(\right)$$
$$\theta_{i,0} = q_{\theta}\left(\right)$$
$$\mathbf{B}_i = q_{\mathbf{B}}\left(\right)$$

The details on the $q$ functions are described in Section 3.1 and Section 3.2.

The system state is changed from $\mathbf{S}_t$ to $\mathbf{S}_{t+1}$ at time $t+1$ by generating a series of the following events sequentially.

1) Consider a **Split** event for each of the merge events occurred at time $t$. When a split condition is satisfied for a merge case occurred at time $t$, split the merged target into target $i$ and target $j$, which resets $\left(\mathbf{G}_{t+1}\right)_{ij} = \left(\mathbf{G}_{t+1}\right)_{ji} = 0$. The split condition will be described in Section 3.5.

2) For each target $i$, **Motion** $f$ and **Rotation** $g$ make changes on

$$\left(x_{i,t+1}, y_{i,t+1}\right) = f\left(x_{i,t}, y_{i,t}\right)$$
$$\theta_{i,t+1} = g\left(\theta_{i,t}\right)$$

More details on $f$ and $g$ are described in Section 3.3.

3) **Merge** of target $i$ and target $j$ that exist at time $t$ occurs when the merging condition described in Section 3.4 is satisfied, and it sets $\left(\mathbf{G}_{t+1}\right)_{ij} = \left(\mathbf{G}_{t+1}\right)_{ji} = 1$.

4) **Birth** sets $n_{t+1} = n_t + b_t$ with the number of birth events $b_t$, which increases the number of system state variables by $n_t$. In the discrete event simulation, a birth process has been modeled by a Poisson arrival process with $\alpha$ as the mean birth rate per unit time interval, where the inter-birth time in

between two consecutive birth events follows an exponential distribution with mean $1/\alpha$ [46]. The rate parameter can be changing in time, which is denoted by $\alpha_t$. Following the approach, we sample $b_t$ from

$$b_t \sim \text{Poisson}(\alpha_t),$$

where $\alpha_t$ is the expected number of birth events occurred at time $t$ and it can be specified by simulator users to change the level of birth events. The state vector for each of the $b_t$ born targets is initialized by the $q$ functions,

$$\begin{aligned}(x_{i,t+1}, y_{i,t+1}) &= q_{x,y}(\,)\\ \theta_{i,t+1} &= q_\theta(\,)\\ \mathbf{B}_i &= q_{\mathbf{B}}(\,),\end{aligned}$$

and $\mathbf{G}_{t+1}$ is augmented by appending $b_t$ columns and $b_t$ rows of zero values to $\mathbf{G}_t$.

5) **Death** of target $i$ occurs when the new centroid $(x_{i,t+1}, y_{i,t+1})$ is out of a pre-specified image region $[0,m]\times[0,n]$. When it occurs, it would reduce $n_{t+1}$ by one and would make the removal of the corresponding target-level state vectors.

### 3.1. Appearance Function $q_{\mathbf{B}}$

The appearance function $q_{\mathbf{B}}$ is a random process that generates the image coordinates on the outline of a target. By default, our simulator generates the circular outline of a target with a random radius $r$,

$$\mathbf{B} = r(\cos(\mathbf{a}), \sin(\mathbf{a})),$$

where $r \sim \mathcal{N}(r_{mean}, r_{sig})$ and $\mathbf{a}$ is a vector of equally spaced numbers in $[0, 2\pi]$.

When users want to customize target boundaries, users can give a set of $N$ candidate target boundaries $\{\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \cdots, \mathbf{B}^{(N)}\}$. The appearance function $q_{\mathbf{B}}$ randomly chooses one of the boundaries for each target with the chosen boundary index sampled from $\text{Unif}(1,N)$, where $\text{Unif}(1,N)$ is the uniform distribution over integer numbers in 1, $N$.

### 3.2. Initial Scene Function $q_{x,y}$ and $q_\theta$

At time 0, the initial scene function determines the initial location and orientation of a target by

$$\begin{aligned}u &= \left\lceil \sqrt{n_0} - 1 \right\rceil\\ v &= i \bmod u\\ p &= \begin{cases} v, & \text{if } v \neq 0\\ u, & \text{otherwise}\end{cases}\\ (x_{i,0}, y_{i,0}) &= (-m+w, -n+w) + d\left(\left\lceil \frac{i}{u} - 1 \right\rceil, p-1\right)\\ \theta_{i,0} &\sim \text{Unif}(0, 2\pi),\end{aligned}$$

which evenly distributes $n_0$ targets over $[-m,m]\times[-n,n]$ on grid points with

distance $d$, and margin $w$. At time $t>0$, the initial scene function determines the initial location and orientation of a newly born target by

$$x_{i,t} \sim \text{Unif}(0,m),\ y_{i,t} \sim \text{Unif}(0,n) \text{ and } \theta_{i,0} \sim \text{Unif}(0,2\pi).$$

### 3.3. Motion Function *f* and Rotation *g*

The motion function $f$ determines each target's location at time step $t$ in the simulation. Users have two choices among the Brownian motion [47] [48] [49],

$$\left(x_{i,t+1}, y_{i,t+1}\right) \sim \mathcal{N}\left(\left(x_{i,t}, y_{i,t}\right), \begin{bmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{bmatrix}\right).$$

and the stochastic diffusion process [50],

$$\left(x_{i,t+1}, y_{i,t+1}\right) \sim \mathcal{N}\left(\left(x_{i,t}, y_{i,t}\right), \boldsymbol{\Sigma}\right),$$

where $\boldsymbol{\Sigma}$ is a two-by-two positive definite covariance matrix which determines the overall movement direction.

The rotation function $g$ determines the random rotation angle of target $i$ at time $t+1$ by

$$\theta_{i,t+1} \sim \text{Unif}(0,2\pi).$$

Once $\left(x_{i,t+1}, y_{i,t+1}\right)$ and $\theta_{i,t+1}$ are sampled, the outline of target $i$ at time $t+1$ is determined by the rigid body transformation of $\mathbf{B}_i$,

$$\mathbf{B}_i \begin{bmatrix} \cos\left(\theta_{i,t+1}\right) & \sin\left(\theta_{i,t+1}\right) \\ -\sin\left(\theta_{i,t+1}\right) & \cos\left(\theta_{i,t+1}\right) \end{bmatrix} + \mathbf{1}\left(x_{i,t+1}, y_{i,t+1}\right),$$

where $\mathbf{1}$ is a column vector of ones which has the same size as the row size of $\mathbf{B}_i$.

### 3.4. Merging Condition

The merge event occurs at time $t$ in between target $i$ and $j$ if they spatially overlaps after the motion and the rotation applied. Since the target $i$'s outline would be

$$\hat{\mathbf{B}}_{i,t} = \begin{bmatrix} \cos\left(\theta_{i,t}\right) & \sin\left(\theta_{i,t}\right) \\ -\sin\left(\theta_{i,t}\right) & \cos\left(\theta_{i,t}\right) \end{bmatrix} \mathbf{B}_{i,t} + \begin{bmatrix} x_{i,t} \\ y_{i,t} \end{bmatrix},$$

the overlap of targets $i$ and $j$ would occur only if

$$d_H\left(\hat{\mathbf{B}}_{i,t}, \hat{\mathbf{B}}_{j,t}\right) = 0,$$

where $d_H(\mathbf{A},\mathbf{B})$ is the Hausdorff distance between $\mathbf{A}$ and $\mathbf{B}$. If the condition holds, the merge matrix $\mathbf{G}_t$ is updated by setting its ($i$, $j$) and ($j$, $i$) elements set to one. Once they merge, the simulator applies the same motion for the two targets so that they can move together before they split.

### 3.5. Split Condition

Each merged case is considered for being re-split into individual targets. The

probability of split is represented by a binomial distribution of the probability of split $p$.

### 3.6. Determination of the Number of Birth Events $b_t$

The number of birth events at time $t$ is determined by

$$b_t \sim \text{Poisson}(\alpha_t),$$

where $\alpha_t$ is the average number of event occurrings. The initial state of each new born target is randomly sampled from $q_{\mathbf{B}}$, $q_\theta$ and $q_{x,y}$.

### 3.7. Image and Noise Generation Function

For time $t$, a synthetic image $\mathbf{I}_t$ is generated from the system state $\mathbf{S}_t$. It is a $m \times n$ grayscale image where all targets outlined by $\mathbf{B}_{i,t}$ with centroid $(x_{i,t}, y_{i,t})$ and orientation $\theta_{i,t}$ are colored black (*i.e.* image intensity = 0), and the remaining background is colored white (*i.e.* image intensity = 255). We add the following noise components on the synthetic images to simulate different signal-to-noise ratios:

- Gaussian random noise: $\epsilon_{x,y} \sim \mathcal{N}(0, \sigma_\epsilon^2)$ for each image pixel at $(x, y)$.
- Non-uniform illumination: $H_{x,y} = f(x, y)$ for each image pixel $(x, y)$, and $f(x, y)$ is the function defined by users. The output image $\mathbf{L}_t$ at time $t$ follows [51]

$$\mathbf{L}_{x,y,t} = H_{x,y} * \mathbf{I}_{x,y,t}$$

for each image pixel at $(x, y)$.

- Both Gaussian random noise and non-uniform illumination: The output image $\mathbf{L}_t$ at time $t$ follows

$$\mathbf{L}_{x,y,t} = H_{x,y} * \mathbf{I}_{x,y,t} + \epsilon_{x,y}$$

for each image pixel $(x, y)$. $G_{x,y}$ and $\epsilon_{x,y}$ are the non-uniform illumination and Gaussian random noise generated above, respectively.

## 4. Performance Evaluation with the Simulator

The synthetic images generated by the simulator are used as inputs to a multi-target tracking algorithm to be tested, and the state vectors $\{\mathbf{S}_t; t = 0, 1, 2, \cdots, T\}$ are used as a ground-truth to compute the performance metrics of the algorithm.

If the algorithm being tested is a tracking-by-detection algorithm [14] [15] [22], target detections are first obtained for all image frames either by obtaining target boundaries with true system state $\mathbf{S}_t$ or by choosing and running an existing target detection algorithm on simulated images. A tracking-by-detection algorithm associates the detections to target identities via one-to-one, one-to-many or many-to-one associations, which will output the data association $\mathbf{J}_t$ for every time frame $t$, which is a $n_t \times 2$ matrix with the detection identifier in the first column and the corresponding target identifier in the second column for each row. The output association can be compared with the groud-truth association generated using $\mathbf{S}_t$; in particular, each target location $(x_{i,t}, y_{i,t})$ and the merge

state matrix $\mathbf{G}_t$ are used. The popular performance metric for this type of the algorithm is the accuracy of one-to-one, one-to-many (split) and many-to-one (merge) data associations in terms of the false positive (FPR) and false negative rates (FNR), which is defined by the following equations [52],

$$\text{FPR} = \frac{FP}{FP + TN} \tag{1}$$

$$\text{FNR} = \frac{FN}{FN + TP} \tag{2}$$

where $FP$ is the number of false positives, $FN$ is the number of false negatives, $TN$ is the number of true negatives, and $TP$ is the number of true positives. The false positive and false negative rates can be obtained by comparing $\mathbf{J}_t$ with the ground truth $\left\{\left(x_{i,t}, y_{i,t}\right), \mathbf{G}_t\right\}$. $\mathbf{G}_t$ contains all merge and split information and $\left(x_{i,t}, y_{i,t}\right)$ contains the trace of individual targets. Therefore, one can extract all one-to-one, one-to-many and many-to-one associations by tracking $\left\{\left(x_{i,t}, y_{i,t}\right), \mathbf{G}_t\right\}$, which can be directly compared with $\mathbf{J}_t$ for FPR and FNR.

If the algorithm being tested is a filtering-based algorithm that estimates each target's location at each time step [27] [28], its estimates can be compared with the ground-truth $\left\{\left(x_{i,t}, y_{i,t}\right)\right\}$ in order to compute the popular CLEAR MOT metrics [53], the multiple object tracking precision (MOTP) and the multiple object tracking accuracy (MOTA). The MOTP is a measure of the overall target location estimation, which is computed by

$$\text{MOTP} = \frac{\sum_{i,t} d_{i,t}}{\sum_{t} c_t} \tag{3}$$

where $d_{i,t}$ is the Euclidean distance between the location estimate and the true target's location $\left(x_{i,t}, y_{i,t}\right)$ for target $i$ at time $t$, and $c_t$ is the number of the cases that the estimates are with a certain small range from the true target's location, at time $t$. The MOTA is

$$\text{MOTA} = 1 - \frac{\sum_{t} \left(FN_t + FP_t + MM_t\right)}{\sum_{t} g_t} \tag{4}$$

where $FN_t$, $FP_t$, and $MM_t$ are the number of false negatives, false positives, and missed matches respectively for $t$, and $g_t$ is the total number of ground-truth targets at time $t$.

## 5. Demonstration

In this section, we demonstrate the use of our simulator in generating benchmark data and evaluating multitarget tracking algorithms with the benchmark data.

### 5.1. Simulate Benchmark Datasets with Our Simulator

For the demonstration purpose, we simulated a benchmark dataset with the

following input parameters, total number of image frames: $T$; initial number of targets: $n_0$; target appearance: circle of a radius with mean $r_{mean}$ and standard deviation $r_{sig}$; spatial distance in between targets at the first frame: $d$; image size: $2m \times 2n$; temporal change in target locations: $\sigma_x$ along $x$-direction and $\sigma_y$ along $y$-direction; average number of birth events per time: $\alpha_t$; image noise: white noise with noise variance $\sigma_\epsilon^2$ and non-uniform illumination with $f(x,y)$; probability of split: $p$.

**Figure 2** shows example images generated with $T=10$, $n_0=5$, $r_{mean}=1$, $r_{sig}=0.1$, $d=6$, $m=13$, $n=13$, $\sigma_x=\sigma_y=0.5$, $\alpha_t=1$, $\sigma_\epsilon=0.09$,

$$f(x,y)=\frac{x^2+y^2}{1-x^2-y^2}, \text{ and } \quad p=0.8.$$

## 5.2. Testing and Evaluating Tracking-by-Detection Algorithms

We demonstrate the use of our simulator to evaluate the multiway data association [22] and the linear programming approach [21]. For the evaluation, we designed the experiment by a $2^3$ factorial design of three variables, the initial distance in between two closest targets ($d$), target movement speed ($\sigma_x$ and $\sigma_y$), and target birth rate ($\alpha_t$), as seen in **Table 1**. We fixed the other simulation inputs to $T=10$, $n_0=10$, $r_{mean}=1$, $r_{sig}=0.1$, $m=13$, $n=13$, $\sigma_\epsilon=0.09$, $f(x,y)=1$, and $p=0.8$.

For each design, we performed ten simulation runs with the same design variables for replicated experiments to reduce any random effects on the evaluation outcomes. For each simulation run, we recorded the numbers of merging events, splitting events, target births, and target deaths occurred during the simulation run. **Table 2** shows the average numbers for ten replicated experiments for each design. As observed in **Table 2**, the complexity of the simulated tracking scenarios change as we vary simulation inputs. For example, when we decrease the distance d and/or increase the temporal variation of target location, $\sigma_x$ and $\sigma_y$, more merging and splitting events occur. With the increased birth rate $\alpha_t$, more birth events occur.

For each replication, we input the images generated by our simulator ten

**Table 1.** Design of experiments for evaluating tracking-by-detection algorithms.

| | $d$ | $\sigma_x$, $\sigma_y$ | $\alpha_t$ |
|---|---|---|---|
| Design 1 | 2 | 0.15, 0.15 | 0.3 |
| Design 2 | 1.5 | 0.15, 0.15 | 0.3 |
| Design 3 | 2 | 0.25, 0.25 | 0.3 |
| Design 4 | 2 | 0.15, 0.15 | 0.5 |
| Design 5 | 1.5 | 0.15, 0.15 | 0.5 |
| Design 6 | 1.5 | 0.25, 0.25 | 0.3 |
| Design 7 | 2 | 0.25, 0.25 | 0.5 |
| Design 8 | 1.5 | 0.25, 0.25 | 0.5 |

**Table 2.** Average number of target split, merge, birth and death events occurred per an experimental run.

| Design | 1-2 split | 1-3 split | 2-1 merge | 3-1 merge | Birth | Death |
|---|---|---|---|---|---|---|
| 1 | 6 | 0 | 36 | 0 | 15 | 2 |
| 2 | 21 | 0 | 160 | 3 | 24 | 6 |
| 3 | 10 | 0 | 78 | 0 | 19 | 2 |
| 4 | 10 | 0 | 64 | 0 | 41 | 4 |
| 5 | 26 | 0 | 154 | 2 | 32 | 1 |
| 6 | 25 | 5 | 222 | 23 | 19 | 0 |
| 7 | 12 | 0 | 110 | 4 | 29 | 1 |
| 8 | 21 | 10 | 86 | 16 | 35 | 5 |

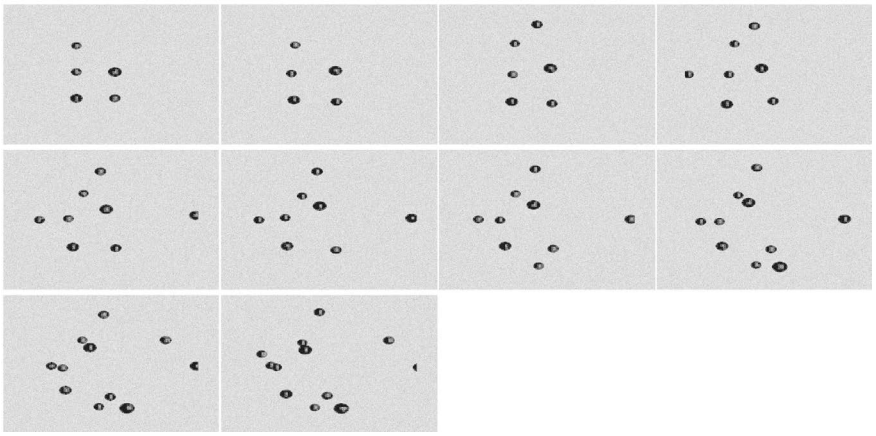


**Figure 2.** Simulated video frame.

simulations to each of the multiway data association [22], and the linear programming approach [21]. The outcomes of the two methods were compared with the ground-truth given by our simulator. The popular performance metric for a tracking-by-detection algorithm is the accuracy of one-to-one, one-to-many (split) and many-to-one (merge) data associations in terms of the false positive (FPR) and false negative rates (FNR) as we summarized in Section 4. The comparison results were summarized by the average false positive (FPR) and false negative rates (FNR) over ten simulation runs for each design. **Table 3** and **Table 4** show comparative results. The multiway data association [22] performed better than the linear programming approach [21] in handling birth events for all experiments. In terms of the capability of handling death events, the multiway data association [22] had a better performance than the linear programming approach [21] with the exception of Design 4 and 6. The multiway data association [22] detected better 2-1 merging and 1-2 splitting events than the linear programming approach [21] for most of the designs except Design 1. The linear programming approach failed to track 3-1 merging and 1-3 splitting events, while the multiway data association [22] was able to track 3-1 merging and1-3

**Table 3.** The average false positive rate (FPR) and false negative rate (FNR) for the multiway data association [22] and the linear programming approach [21] over ten replicated simulations for Designs 1 through 4.

| | Multiway data association | | Linear programming approach | |
|---|---|---|---|---|
| Design 1 | FPR | FNR | FPR | FNR |
| 1-to-1 | 0.0202 | 0.0041 | 0.0202 | 0.0041 |
| 1-to-2 | 0.1111 | 0.0000 | 0.1111 | 0.0000 |
| 2-to-1 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 1-to-3 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 3-to-1 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| Birth | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| Death | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| Design 2 | FPR | FNR | FPR | FNR |
| 1-to-1 | 0.0263 | 0.0032 | 0.0305 | 0.0033 |
| 1-to-2 | 0.0000 | 0.1000 | 0.0689 | 0.1333 |
| 2-to-1 | 0.0000 | 0.0222 | 0.0000 | 0.0227 |
| 1-to-3 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 3-to-1 | 0.0000 | 0.0000 | 0.0000 | 1.0000 |
| Birth | 0.1363 | 0.0952 | 0.2400 | 0.1000 |
| Death | 0.5000 | 1.0000 | 0.6667 | 1.0000 |
| Design 3 | FPR | FNR | FPR | FNR |
| 1-to-1 | 0.0184 | 0.0011 | 0.0184 | 0.0011 |
| 1-to-2 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 2-to-1 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 1-to-3 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 3-to-1 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| Birth | 0.0769 | 0.0000 | 0.07969 | 0.0000 |
| Death | 0.5000 | 0.0000 | 0.5000 | 0.0000 |
| Design 4 | FPR | FNR | FPR | FNR |
| 1-to-1 | 0.0370 | 0.0119 | 0.0370 | 0.0119 |
| 1-to-2 | 0.3636 | 0.1764 | 0.3636 | 0.1764 |
| 2-to-1 | 0.0555 | 0.1500 | 0.0555 | 0.1500 |
| 1-to-3 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 3-to-1 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| Birth | 0.0869 | 0.2857 | 0.0869 | 0.2857 |
| Death | 0.5000 | 0.0000 | 0.5000 | 0.0000 |

**Table 4.** The average false positive rate (FPR) and false negative rate (FNR) for the multiway data association [22] and the linear programming approach [21] over ten replicated simulations for Designs 5 through 8.

| | Multiway data association | | Linear programming approach | |
|---|---|---|---|---|
| Design 5 | FPR | FNR | FPR | FNR |
| 1-to-1 | 0.0332 | 0.0021 | 0.0343 | 0.0021 |
| 1-to-2 | 0.0550 | 0.0801 | 0.0550 | 0.0801 |

**Continued**

| | | | | |
|---|---|---|---|---|
| 2-to-1 | 0.0000 | 0.0416 | 0.0212 | 0.0416 |
| 1-to-3 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 3-to-1 | 0.0000 | 0.0000 | 0.0000 | 1.0000 |
| Birth | 0.0000 | 0.0416 | 0.0000 | 0.0416 |
| Death | 0.0000 | 0.0000 | 0.5000 | 0.0000 |
| Design 6 | FPR | FNR | FPR | FNR |
| 1-to-1 | 0.0567 | 0.0286 | 0.1048 | 0.0279 |
| 1-to-2 | 0.2264 | 0.3050 | 0.3114 | 0.2881 |
| 2-to-1 | 0.1212 | 0.2739 | 0.2428 | 0.3424 |
| 1-to-3 | 0.5000 | 0.5000 | 0.0000 | 1.0000 |
| 3-to-1 | 0.0000 | 0.0000 | 0.0000 | 1.0000 |
| Birth | 0.1428 | 0.2352 | 0.7000 | 0.2352 |
| Death | 1.0000 | 0.0000 | 1.0000 | 0.0000 |
| Design 7 | FPR | FNR | FPR | FNR |
| 1-to-1 | 0.0332 | 0.0082 | 0.0422 | 0.0073 |
| 1-to-2 | 0.1290 | 0.0689 | 0.1379 | 0.1379 |
| 2-to-1 | 0.0000 | 0.0882 | 0.7638 | 0.0882 |
| 1-to-3 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 3-to-1 | 0.2500 | 0.0000 | 0.0000 | 1.0000 |
| Birth | 0.0000 | 0.2000 | 0.2105 | 0.2000 |
| Death | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| Design 8 | FPR | FNR | FPR | FNR |
| 1-to-1 | 0.0910 | 0.0389 | 0.1125 | 0.0366 |
| 1-to-2 | 0.2250 | 0.4833 | 0.2682 | 0.5000 |
| 2-to-1 | 0.1428 | 0.3731 | 0.2678 | 0.3880 |
| 1-to-3 | 0.5000 | 0.7143 | 0.0000 | 1.0000 |
| 3-to-1 | 0.3333 | 0.5454 | 0.0000 | 1.0000 |
| Birth | 0.2083 | 0.3200 | 0.5000 | 0.3200 |
| Death | 0.8000 | 0.6667 | 0.9444 | 0.6667 |

splitting evets. Overall, the multiway data association [22] performed better than the linear programming approach [21] in detecting all merging and splitting events.

### Evaluating Filtering-Based Algorithms

In this section, we use our simulator to evaluate a simple Kalman filtering-based algorithm [1] [54] [55] to see how it performs with different levels of image noises and different numbers of targets. We performed a $2^2$ factorial design of two factors $n_0$ and $\sigma_\epsilon$ as described in **Table 5**, while controlling the other factors to $T=10$, $r_{mean}=1$, $r_{sig}=0.1$, $m=13$, $n=13$, $\sigma_x=0.15$, $\sigma_y=0.15$, $d=5$, $f(x,y)=1$, $\alpha_t=0.3$ and $p=0.8$.

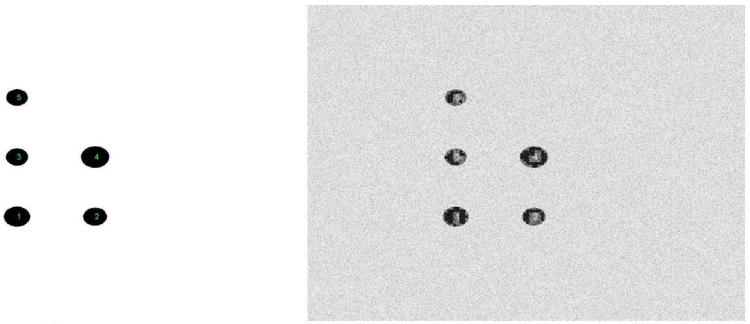
For each design, we performed ten simulation runs with the same factor levels. **Figure 3** shows some of the generated images under Design 1 and Design 2.

Table 5. Design of experiments for evaluating a Kalman filtering-based algorithm.

| | $n_0$ | $\sigma_\epsilon$ |
|---|---|---|
| Design 1 | 5 | 0 |
| Design 2 | 5 | 0.5 |
| Design 3 | 20 | 0 |
| Design 4 | 20 | 0.5 |

Table 6. Average MOTA and MOTP metrics.

| | MOTA | MOTP |
|---|---|---|
| Design 1 | 0.3406 | 154 |
| Design 2 | 0.2350 | 135 |
| Design 3 | 0.3010 | 138 |
| Design 4 | 0.1717 | 102 |



(a) (b)

Figure 3. Simulated video frame. (a) Design 1; (b) Design 2.

For each experiment run, the Kalman-filtering-based algorithm first read in simulated images, detects each target in the scene and finally estimates each target's position with the Kalman filter. We computed the MOTP and MOTA metrics in order to evaluate the accuracy of the estimation. Table 6 summarizes the average MOTP and MOTA metrics over ten simulation runs for each design, where high values implies higher accuracy for both of the metrics [53]. The performance degradation of the algorithm was significant with the increased level of noises, but the effect of the number of targets on the performance was not significant.

## 6. Conclusion

We presented a novel simulation model that simulates video frames containing the movements and interactions of multiple targets with fully controlled complexities of target motion, target appearance, image noise levels, target birth and death rates, and target merge and split rates. The simulator also generates the groundtruth locations of the targets for the simulated video frames, which makes the simulation model an ideal tool for generating benchmark datasets to

evaluate multitarget tracking algorithms. We demonstrated the use of the proposed simulation model to evaluate two tracking-by-detection algorithms and one filtering-based algorithm. In addition, the simulator can generate new public benchmark datasets with high-degree of interactions and complexity with ease from the user's inputs.

## Acknowledgements

This project was partially supported by the National Science Foundation with grant no NSF-CMMI-1334012, the Air Force Office of Scientific Research with grant no FA9550-13-1-0075, and the Florida State University Council on Research and Creativity Planning Grant 036656.

## Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

## References

[1] Popoli, R. and Blackman, S.S. (1999) Design and Analysis of Modern Tracking Systems. Artech House, Norwood, MA.

[2] Yilmaz, A., Javed, O. and Shah, M. (2006) Object Tracking: A Survey. *ACM Computing Survey*, **38**, 1-45. https://doi.org/10.1145/1177352.1177355

[3] Skolnik, M.I. (1990) Radar Handbook. McGraw-Hill, New York City, NY.

[4] Stone, L.D., Streit, R.L., Corwin, T.L. and Bell, K.L. (2013) Bayesian Multiple Target Tracking. Artech House, Norwood, MA.

[5] Bar-Shalom, Y. (1987) Tracking and Data Association. Academic Press Professional, Inc.

[6] Blackman, S.S. (1986) Multiple-Target Tracking with Radar Applications. Artech House, Inc., Dedham, MA.

[7] Mahler, R.P.S. (2007) Statistical Multisource-Multitarget Information Fusion. Artech House, Norwood, MA.

[8] Lane, D.M., Chantler, M.J. and Dai, D.Y. (1998) Robust Tracking of Multiple Objects in Sector-Scan Sonar Image Sequences Using Optical Flow Motion Estimation. *IEEE Journal Oceanic Engineering*, **23**, 31-46. https://doi.org/10.1109/48.659448

[9] Kocak, D.M., da Vitoria Lobo, N. and Widder, E. (1999) A Computer Vision Techniques for Quantifying, Tracking, and Identifying Bioluminescent Plankton. *IEEE Journal Oceanic Engineering*, **24**, 81-95. https://doi.org/10.1109/48.740157

[10] Durrant-Whyte, H. and Bailey, T. (2006) Simultaneous Localization and Mapping: Part I. *IEEE Robotics and Automation Magazine*, **13**, 99-110. https://doi.org/10.1109/MRA.2006.1638022

[11] Bailey, T. and Durrant-Whyte, H. (2006) Simultaneous Localization and Mapping (SLAM): Part II. *IEEE Robotics and Automation Magazine*, **13**, 108-117. https://doi.org/10.1109/MRA.2006.1678144

[12] Spagnolini, U. and Rampa, V. (1999) Multitarget Detection/Tracking for Monostatic Ground Penetrating Radar: Application to Pavement Profiling. *IEEE Transac-*

*tions on Geoscience and Remote Sensing*, **37**, 383-394. https://doi.org/10.1109/36.739074

[13] Sethi, I.K. and Jain, R. (1987) Finding Trajectories of Feature Points in a Monocular Image Sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **9**, 56-73.

[14] Veenman, C.J., Reinders, M.J.T. and Backer, E. (2001) Resolving Motion Correspondence for Densely Moving Points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **23**, 54-72. https://doi.org/10.1109/34.899946

[15] Pirsiavash, H., Ramanan, D. and Fowlkes, C.C. (2011) Globally-Optimal Greedy Algorithms for Tracking a Variable Number of Objects. *IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, 20-25 June 2011, 1201-1208.

[16] Jiang, H., Fels, S. and Little, J.J. (2007) A Linear Programming Approach for Multiple Object Tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, 17-22 June 2007, 1-8.

[17] Zhang, L., Li, Y. and Nevatia, R. (2008) Global Data Association for Multi-Object Tracking Using Network Flows. *IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, 23-28 June 2008, 1-8.

[18] Jaqaman, K., Loerke, D., Mettlen, M., Kuwata, H., Grinstein, S., Schmid, S.L. and Danuser, G.R. (2008) Single-Particle Tracking in Live-Cell Time-Lapse Sequences. *Nature Methods*, **5**, 695-702. https://doi.org/10.1038/nmeth.1237

[19] Serge, A., Bertaux, N., Rigneault, H. and Marguet, D. (2008) Dynamic Multiple-Target Tracing to Probe Spatiotemporal Cartography of Cell Membranes. *Nature Methods*, **5**, 687-694. https://doi.org/10.1038/nmeth.1233

[20] Choi, W. and Savarese, S. (2012) A Unified Framework for Multi-Target Tracking and Collective Activity Recognition. *European Conference on Computer Vision*, Florence, 7-13 October 2012, 215-230.

[21] Henriques, J.F., Caseiro, R. and Batista, J. (2011) Globally Optimal Solution to Multi-Object Tracking with Merged Measurements. *IEEE International Conference on Computer Vision*, Barcelona, 6-13 November 2011, 2470-2477.

[22] Park, C., Woehl, T.J., Evans, J.E. and Browning, N.D. (2014) Minimum Cost Multi-Way Data Association for Optimizing Multitarget Tracking of Interacting Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **37**, 611-624. https://doi.org/10.1109/TPAMI.2014.2346202

[23] Broida, T.J. and Chellappa, R. (1986) Estimation of Object Motion Parameters from Noisy Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **8**, 90-99.

[24] Beymer, D. and Konolige, K. (1999) Real-Time Tracking of Multiple People Using Continuous Detection.

[25] Rosales, R. and Sclaroff, S. (1999) 3D Trajectory Recovery for Tracking Multiple Objects and Trajectory Guided Recognition of Actions. *IEEE Conference on Computer Vision and Pattern Recognition*, Colorado, 23 June 1999, 117-123.

[26] Tanizaki, H. (1996) Nonlinear Filters: Estimation and Applications. Springer, New York City, 400.

[27] Khan, Z., Balch, T. and Dellaert, F. (2004) An MCMC-Based Particle Filter for Tracking Multiple Interacting Targets. *European Conference on Computer Vision*, Prague, 11-14 May 2004, 279-290.

[28] Hue, C., Le Cadre, J.-P. and Perez, P. (2002) Tracking Multiple Objects with Particle

Filtering. *IEEE Transactions on Aerospace Electronics System*, **38**, 791-812. https://doi.org/10.1109/TAES.2002.1039400

[29] Genovesio, A. and Olivo-Marin, J.-C. (2004) Split and Merge Data Association Filter for Dense Multi-Target Tracking. *International Conference on Pattern Recognition*, **4**, 677-680.

[30] Khan, Z., Balch, T. and Dellaert, F. (2005) Multi-Target Tracking with Split and Merged Measurements. *IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, 20-26 June 2005, Vol. 1, 605-610.

[31] Khan, Z., Balch, T. and Dellaert, F. (2006) MCMC Data Association and Sparse Factorization Updating for Real Time Multi-Target Tracking with Merged and Multiple Measurements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**, 1960-1972. https://doi.org/10.1109/TPAMI.2006.247

[32] Storlie, C.B., Lee, T.C.M., Hannig, J. and Nychka, D. (2009) Tracking of Multiple Merging and Splitting Targets: A Statistical Perspective. *Statistica Sinica*, **19**, 1-52.

[33] Yang, B. and Nevatia, R. (2012) Multi-Target Tracking by Online Learning of Non-Linear Motion Patterns and Robust Appearance Models. *IEEE Conference on Computer Vision and Pattern Recognition*, Providence, 16-21 June 2012, 1918-1925.

[34] Alahi, A., Jacques, L., Boursier, Y. and Vandergheynst, P. (2011) Sparsity Driven People Localization with a Heterogeneous Network of Cameras. *Journal of Mathematical Imaging and Vision*, **41**, 39-58. https://doi.org/10.1007/s10851-010-0258-7

[35] BIWI Walking Pedestrians Dataset Computer Vision Laboratory-ETH, 2009.

[36] Pellegrini, S., Ess, A., Schindler, K. and Van Gool, L. (2009) You'll Never Walk Alone: Modeling Social Behavior for Multi-Target Tracking. *International Conference on Computer Vision*, Kyoto, 27 September-4 October 2009, 261-268.

[37] Andriluka, M., Roth, S. and Schiele, B. (2010) Monocular 3D Pose Estimation and Tracking by Detection. *IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, 13-18 June 2010, 623-630.

[38] Yang, B. and Nevatia, R. (2012) An Online Learned CRF Model for Multi-Target Tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, Providence, 16-21 June 2012, 2034-2041.

[39] Milan, A., Roth, S. and Schindler, K. (2014) Continuous Energy Minimization for Multitarget Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **36**, 58-72. https://doi.org/10.1109/TPAMI.2013.103

[40] Milan, A., Schindler, K. and Roth, S. (2013) Challenges of Ground Truth Evaluation of Multi-Target Tracking. *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Portland, 23-28 June 2013, 735-742.

[41] Pedestrian Mobile Scene Analysis Computer Vision Laboratory-ETH, 2009.

[42] Ess, A., Leibe, B. and Van Gool, L. (2007) Depth and Appearance for Mobile Scene Analysis. *International Conference on Computer Vision*, Rio de Janeiro, 14-20 October 2007, 1-8.

[43] Benfold, B. and Reid, I. (2011) Stable Multi-Target Tracking in Real-Time Surveillance Video. *IEEE Conference on Computer Vision and Pattern Recognition*, Colorado, 20-25 June 2011, 3457-3464.

[44] Nannuru, S., Coates, M. and Mahler, R. (2013) Computationally-Tractable Approximate Probability Hypothesis Density and Cardinalized Probability Hypothesis Density Filters for Superpositional Sensors. *IEEE Journal Selective Topics in Signal Processing*, **7**, 410-420.

[45] Hoseinnezhad, R., Vo, B.-N. and Vo, B.-T. (2013) Visual Tracking in Background Subtracted Image Sequences via Multi-Bernoulli Filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **61**, 392-397.

[46] Pinsky, M. and Karlin, S. (2010) An Introduction to Stochastic Modeling. Academic Press, Waltham.

[47] Hida, T. (1980) Brownian Motion. Springer, New York.

[48] Karatzas, I. (1991) Brownian Motion and Stochastic Calculus. Springer, New York.

[49] Revuz, D. and Yor, M. (1999) Continuous Martingales and Brownian Motion. Springer, New York.

[50] Bibbona, E., Panfilo, G. and Tavella, P. (2008) The Ornstein-Uhlenbeck Process as a Model of a Low Pass Filtered White Noise. *Metrologia*, **45**, 117. https://doi.org/10.1088/0026-1394/45/6/S17

[51] Matsushita, Y., Nishino, K., Ikeuchi, K. and Sakauchi, M. (2004) Illumination Normalization with Time-Dependent Intrinsic Images for Video Surveillance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**, 1336-1347. https://doi.org/10.1109/TPAMI.2004.86

[52] Sheskin, D.J. (2003) Handbook of Parametric and Nonparametric Statistical Procedures. CRC Press, Boca Raton.

[53] Keni, B. and Rainer, S. (2008) Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP Journal in Image and Video Processing*, **2008**, Article ID: 246309.

[54] Brown, R.G., Hwang, P.Y.C., *et al.* (1992) Introduction to Random Signals and Applied Kalman Filtering. Wiley, Hoboken.

[55] Kim, P. and Huh, L. (2011) Kalman Filter for Beginners: With MATLAB Examples. CreateSpace, Seattle.

## Nomenclature

ETH: Eidgenssische Technische Hochschule

FN: False Negative

FNR: False Negative Rate

FP: False Positive

FPR: False Positive Rate

MCMC: Markov Chain Monte Carlo

MOT: Multiple Object Tracking

MOTA: Multiple Object Tracking Accuracy

MOTP: Multiple Object Tracking Precision

PETS: Performance Evaluation of Tracking and Surveillance

TN: True Negative

TP: True Positive

TUD: Technical University of Darmstadt