Improving Temporal Consistency in Aerial Based Crowd Monitoring Using Bayes Filters

Jan Calvin Kramer[1], Thomas Golda[2], Jonas Hansert[1], and Thomas Schlegel[1]

[1] Karlsruhe University of Applied Sciences, Institute of Ubiquitous Mobility Systems IUMS
{Jan_Calvin.Kramer, Jonas.Hansert, Thomas.Schlegel}@h-ka.de
[2] Fraunhofer Institute for Optronics, System Technologies and Image Exploitation IOSB
Thomas.Golda@iosb.fraunhofer.de

Abstract. In order to monitor mass events, crowd managers continuously require reliable measurements of the crowd count. For this purpose, a variety of deep learning algorithms has been developed. Most of these so-called crowd counting algorithms return good results for still imagery but return oscillating crowd counts for video data. This is because, most crowd counting algorithms evaluate video data frame by frame and ignore the temporal relation between adjacent frames. In this paper, a variety of Bayesian filters is presented that successfully smooth the oscillating counts which in turn can lead crowd managers to trust the system more. The proposed filters work on top of the crowd counting algorithms' estimates. Thus, they can be easily used with any existing crowd counting algorithm that outputs a density map for a given input image.

Keywords: crowd counting, crowd count, density map, crowd manager, mass panic, video data, bayesian filters, kalman filter, particle filter, aerial imagery

1 Introduction

Mass events take place every day all over the world. Despite the joy these events bring to many people, they always come with the threat of turning into a stampede. To prevent this, crowd managers must be aware of the current crowd count and density and take the right measures in time.
For this purpose, a variety of algorithms have been developed since 2006. These so-called crowd counting algorithms take an image as their input and estimate the amount of people in the picture. Recent crowd counting algorithms go even further. Using Convolutional Neural Networks, they estimate a whole density map for a given input image. The density map itself holds the information of the crowd count that can be obtained by adding up the density map's pixel values. Gao et al. give a good insight into current advances and the huge amount of different crowd counting algorithms that exist [1].
Despite the many advances, current crowd counting algorithms do not consider the temporal relation between adjacent frames. Most crowd counting algorithms evaluate video data frame by frame and ignore the temporal relation. This often leads to fluctuating counts that the crowd manager cannot rely on. Only a few isolated approaches, e.g., the Temporal Aware Network [2], exist, that try to incorporate the temporal relation of adjacent frames. In contrast to their approach of presenting a new architecture that incorporates the temporal relation, we developed a variety of Bayesian filters that work on top of the counts estimated by current crowd counting algorithms. The filters can be used with any modern crowd counting algorithm that outputs a density map for a given input image.

2 Related Work

The following section gives a brief overview of the existing technologies this work builds upon. Firstly, two crowd counting algorithms, i.e., the CSRNet and MRCNet, are introduced. The crowd counters are used throughout this work to estimate the crowd counts of different data sets and to reproduce the problem of having oscillating counts. Secondly, Bayesian filters on which the developed concepts are based are briefly discussed.

2.1 CSRNet

The Congested Scene Recognition Network (called CSRNet hereafter) is a crowd counting algorithm developed by Li et al. in 2018 [3]. The network is divided into two parts, i.e. the front- and back-end. Both parts exclusively rely on (dilated) convolutional and max pooling layers.
The CSRNet's front-end consists of the VGG-16's first 10 layers. It extracts several features from an input image. The features are stored in a so-called feature map whose resolution is 1/8 of the resolution of the input image. The feature map is further processed by the back-end of the network. The back-end uses several dilated convolutional layers. Thus, it is able to extract even deeper features without further shrinking the resolution of the feature map. In order to output a density map that has the same size as the input image, the CSRNet uses bilinear interpolation with a factor of 8. A more detailed description can be taken from [3].

2.2 MRCNet

The Multi-Resolution Crowd Network (called MRCNet hereafter) is a crowd counting algorithm developed by Bahmanyar et al. from the German Aerospace Center [4]. The network relies on an encoder-decoder-structure. Analogous to the front-end of the CSRNet, the encoder of the MRCNet relies on the VGG-16. Yet, it does not only use the first ten layers but the first five CNN blocks of the VGG-16 that consist of 13 convolutional layers and five max-pooling layers in total. Since it uses more pooling layers, the size of the outputted feature map is 1/32 the size of the input image. Such a drastic reduction of the resolution can accidentally lead to people being removed. To prevent this, the feature maps at different stages of the encoder are added element-wise to the feature maps of the decoder.
The decoder of the MRCNet consists of five CNN blocks as well. However, instead of using pooling layers at the end of each CNN block, the decoder uses up-sampling layers. Each up-sampling layer increases the size of the feature map by a factor of two. Thus, at the end of the decoder, the MRCNet outputs a density map that has the same resolution as the input. Prior to outputting the density map, the MRCNet outputs a feature map whose resolution is 1/4 the resolution of the input image. This feature map is used to estimate the overall amount of people in an image. By estimating the amount of people at an early stage of the decoder, the remaining part of the decoder can be further used to output a full-resolution density map that has a higher localization precision. Further information about the MRCNet can be taken from [4].

2.3 Bayesian Filters

Bayesian filters have been around for quite a while. In short, they are a set of algorithms that iteratively estimate the hidden state of a system, e.g., the current crowd count, using

imprecise measurements, e.g., estimated density maps, and a model of the system state, e.g., a traffic flow model [5].
The Kalman and particle filter are a subset of the Bayesian filters. They differ in that the Kalman filter returns an optimal solution under certain restrictions whereas the particle filter returns a good approximation while being less restrictive. The Kalman filter assumes the underlying model to be linear and discrete in the time domain. Furthermore, the process and measurement noise are assumed to be Gaussian with a zero mean. If these restrictions are not fully complied with, the particle filter is likely to return even better results than the Kalman filter.
A thorough understanding of both filters is necessary to fully understand the concepts that follow. A good insight is given by [5], [6] and [7].

3 Concepts

The following chapter explains the developed concepts of this paper.

3.1 Kalman Filter

A complete concept of a Kalman filter requires the definition of the state vector, state transition matrix, measurement matrix, process noise variance and measurement noise variance.
To not further increase the computational costs that come with crowd counting, the state vector considers only the crowd count and not the density map as a whole. The crowd count is further expected not to change between two consecutive frames. This assumption leads to a linear state transition matrix that only consists of the value 1.
The accuracy of this assumption depends mainly on two variables, i.e., the area under consideration (denoted by A) and the frame rate of the video (denoted by f). For either A going towards zero or f going towards infinity, the assumption that the crowd count does not change becomes true. Thus, the concept is supposed to return better results for video data with higher frame rates and scenes with smaller areas. The scene of an image can be artificially reduced by applying the Kalman filter on grids rather than the whole image. Experiments testing this behavior are conducted in the next chapter.
Yet, in practice, neither A becomes zero nor f goes towards infinity. This makes the state transition matrix inaccurate. To model this error, one must define the process noise variance (matrix).

Data-Driven Process Noise Variance. To get an estimate for the process noise variance, the information given by the training data is used. Given the annotations of the training data, one can calculate the change of pedestrians between two consecutive frames by subtracting their annotated crowd counts. The calculated pedestrian changes can be fitted to a Gaussian curve in a next step. Assuming that the set of pedestrian changes is normally distributed with a zero mean, the variance of the Gaussian curve corresponds to the actual variance of the process noise when the Kalman filter was applied to the training data. To obtain an estimate that generalises better on data sets with different crowd counts, one must consider the percentage change of pedestrians rather than the absolute change. Let c_{k-1} and c_k be the crowd count of two consecutive frames, then the percentage change of pedestrians (Δc_{rel}) can be calculated as follows:

$$\Delta c_{rel} = \frac{c_k - c_{k-1}}{c_{k-1}} \tag{1}$$

Fitting the percentage changes to a Gaussian curve results in a relative process noise variance denoted by σ_{rel}. To retrieve an absolute estimate of the process noise variance, one must iteratively multiply the relative variance with the previous posterior estimate of the Kalman filter and the frame rates' ratio:

$$\sigma_k = \bar{c}_{k-1} \cdot \sigma_{rel} \cdot \frac{f_{training}}{f_{testing}} \tag{2}$$

Data-Driven Measurement Noise Variance. To model the measurement noise variance, a data driven approach similar to the one of the process noise variance is used. Yet, instead of computing pedestrian changes of the training data, the performance of the crowd counter on the validation data is used. For this purpose, the percentage differences between the crowd counter's estimate and the corresponding ground truth is calculated. The percentage differences are also fitted to a Gaussian curve. Again, the variance that is obtained using this approach is a relative value that must be iteratively multiplied with the current measurement of the crowd count.

Data-Driven Observation Matrix. The final parameter that must be modelled to obtain a complete concept of a Kalman filter is the observation matrix. The observation matrix says how the measurements must be processed before they are further used within the Kalman filter. Therefore, if the measurements that are passed to the Kalman filter do not have an error with a zero mean, the observation matrix can theoretically be used to process the measurements in such a way that their error has a zero mean afterwards. For this purpose, the Gaussian curve, that is obtained during the computation of the variance of the measurement noise, is used. The relative mean denoted by μ_{rel} expresses the average deviation of the ground truth from the estimates computed by the crowd counter on the validation set. The oversimplified assumption that the error of the crowd counter depends only on the crowd count, lets one use the simple term $1 - \mu_{rel}$ for the observation matrix.

3.2 Particle Filter

The developed particle filter only considers the crowd count as well. Yet, a more complex model for the state transition matrix is used. The model of choice is the macroscopic fundamental diagram of traffic flow. Let Q be the flow, Q^* be the flow density, v_0 be the velocity of the pedestrians, w be the width through a gateway in meters, ρ and ρ_{max} be the current density and maximum density of pedestrians respectively, then the flow of pedestrians can be calculated as follows:

$$Q^*(\rho) = \rho \cdot v_0 \cdot (1 - \frac{\rho}{\rho_{max}}) \, [\frac{pedestrians}{m \cdot s}] \tag{3}$$

$$Q = Q^* \cdot w \, [\frac{pedestrians}{s}] \tag{4}$$

Assuming that $\rho_{max} = 5 \frac{pedestrians}{m^2}$ and $v_0 = 1.4 \frac{m}{s}$, it only requires the current density and the width of the gateway to estimate the flow of pedestrians [8].
Unfortunately, the traffic flow model comes with a major restriction that it assumes the flow of pedestrians to be unidirectional. To loosen this restriction and obtain a more realistic model, the filter does not assume the overall flow of a scene to be unidirectional.

Flows of open borders that are located at the edge of the image are assumed to be independent from each other and unidirectional. Making this assumption, three questions arise that must be further clarified:

- How does the system detect open borders at the edge of the image?
- What area around an open border must be considered to estimate the density at the open border?
- Given the flow at an open border, how does one estimate the direction of flow?

3.3 Detection of Open Borders

The annotation of open borders is assumed to be manually done by the user at the beginning. The user is expected to create a mask for the first frame of a video. Areas where people can possibly walk are supposed to be colored in white, whereas areas where people cannot be, e.g., a frontage, must be colored in black. By reading in the mask and dividing the pixel values by 255, one obtains a 2 dimensional matrix of zeros and ones. The information where the open borders are located can then be easily obtained by looking for non-zero sequences in its outermost rows and columns.
It is further used as an alternative measurement matrix that is multiplied element-wise with incoming density maps. Subsequently, all elements of the resulting matrix are added up to obtain the crowd count that is further processed by the particle filter. It should be noted that this approach might delete some rightfully annotated persons. This is because the ground truth density maps that are used to train the crowd counting algorithms are created by blurring the given head annotations using a Gaussian kernel. Thus, persons standing nearby a frontage might overlay the frontage in a density map. Yet, this approach ensures that areas where pedestrians can impossibly be, e.g., a sea, are not wrongfully labeled by the crowd counter.

3.4 Determining the Density at an Open Border

To estimate the density at an open border j at the previous point in time $k-1$, a rectangular cutout of the density map at $k-1$ is used. One side of the rectangle is the open border itself, whereas the length of the other side is determined by the maximum distance pedestrians are assumed to walk between two consecutive frames (called step size hereafter). Let v_0 be the pedestrian's estimated velocity and Δt be the time between two consecutive frames, then the step size can be calculated as follows:

$$\Delta s_m = \Delta t \cdot v_0 \; [m] \tag{5}$$

$$\Delta s_{pixel} = \Delta t \cdot v_0 \cdot 1/g \; [pixel] \tag{6}$$

where g corresponds to the ground sampling distance in meter per pixel.
Given a cutout of a density map, you can easily calculate the estimated crowd count within the cutout (denoted by $c_{j,k-1}$) by adding up the pixel values of the cutout. Let further a_j and b_j be the lengths of the rectangle in pixels, then the density of the rectangle can be calculated as follows:

$$\rho_{j,k-1} = \frac{c_{j,k-1}}{a_j \cdot b_j \cdot g^2} \; [\frac{pedestrians}{m^2}] \tag{7}$$

where

- $\rho_{j,k-1}$ $[pedestrians/m^2]$ is the density of the j'th rectangle at k-1,
- g $[m/pixel]$ is the ground sampling distance in meter per pixel.

Given $\rho_{j,k-1}$, the estimated flow at the rectangle can be calculated using the equations 3 and 4. It should be noted that w corresponds to the length of the open border in meters.

3.5 Determining the Flow of Direction at an Open Border

It is further important to know whether pedestrians are either leaving or entering the scene. To estimate the flow of direction, Gunnar Farneback's algorithm is used. For each open border, the algorithm is given an enlarged rectangular cutout of the previous and current frame (not the density map!). The algorithm estimates the movement of the pixels in the x- and y-direction within the enlarged rectangle [9]. Although the algorithm calculates the movement in both directions, only one direction is relevant. For open borders located on the left or right of the image, the x-direction is of interest, whereas for open borders located at the top or bottom of the image the movement in the y-direction is of interest. By adding up the magnitudes by which the pixels are estimated to move along the relevant direction, you can obtain the magnitude of the pixels' overall movement along the direction. This value is denoted by $m_{j,k-1}$ $[pixel]$ in the following. Whereas negative values of $m_{j,k-1}$ measured on the left side or at the bottom of an image indicate an outflow, they indicate an inflow when they are measured at the top or on the right side of an image.
If you would use a rectangle with the same dimensions as the rectangle used to determine the density, you would probably run into the problem that the algorithm would detect no movement when all pedestrians in the rectangle were to leave the rectangle. Therefore, an enlarged rectangle is used that holds pedestrians that are unable to leave the rectangle between two consecutive frames. One side of the enlarged rectangle is the open border itself. The other side is three times the step size.
Given this information, one can calculate the maximum magnitude the algorithm can detect. Let w_j be the length of the open border in pixel, then the maximum magnitude can be calculated as follows:

$$m_{j,max} = 2 \cdot \Delta s^2_{pixel} \cdot w_j \cdot \frac{1}{pixel^2} \; [pixel] \tag{8}$$

Whereas Q is an estimate of the pedestrian's flow obtained by the fundamental traffic flow model, the ratio of $m_{j,k-1}/m_{j,max}$ is an actual measurement. To get an estimate of the pedestrian change, both values are combined:

$$\Delta c_{j,k-1} = Q(\rho_{j,k-1}) \cdot \Delta t \cdot \frac{|m_{j,k-1}|}{m_{j,max}} \; [pedestrians] \tag{9}$$

Depending on whether $m_{j,k-1}$ indicates an outflow, $\Delta c_{j,k-1}$ must be further multiplied by -1:

$$\Delta c_{j,k-1} = \begin{cases} +\Delta c_{j,k-1} & inflow \\ -\Delta c_{j,k-1} & outflow \end{cases} \tag{10}$$

To get the total change of pedestrians that is expected between the consecutive frames, one must add up all Δc:

$$\Delta c_{k-1} = \sum_{j=1}^{k=n} \Delta c_{j,k-1} \tag{11}$$

Adding this result to the samples of the posterior distribution at $k-1$ returns the samples of the priori distribution at k. Let $\bar{c}_{k-1}^{i}$ be the i'th sample from the posterior distribution at $k-1$, then the priori estimate of the i'th sample can be calculated as follows:

$$\bar{c}_{k|k-1}^{i} = \bar{c}_{k-1}^{i} + \Delta c_{k-1} \tag{12}$$

4 Experiments

To train and test the crowd counting algorithms and filters, different data sets are used. Table 1 shows the fundamental differences of the data sets. Tests in the aerial domain are conducted as follows: Firstly, the crowd counters are pre-trained using the DLR-ACD data set [4] to overcome the lack of video data in that domain. Then, the crowd counting algorithms are fine-tuned on the VisDrone-CC2020 data set [10]. Subsequently, tests on the VisDrone-CC2020 and AgoraSet [11] are conducted.
Finally, tests on the WorldExpo'10 data set [12] are conducted. Due to the different domain, the crowd counting algorithms must be trained from scratch prior to testing.

Table 1. Comparison of the data sets. N is the number of annotated frames; FPS is the number of annotated frames per second; GSD is the ground sampling distance in cm/pixel; A_R is the average resolution and I_P is the interval in which the annotated crowd counts lay. The GSDs of the AgoraSet and VisDrone-CC2020 are estimated values. All specified values of the AgoraSet refer to the the frames 418-1936 of its first sequence

	N	**FPS**	**Aerial**	**Artificial**	**GSD**	$\mathbf{A_R}$	$\mathbf{I_P}$
DLR-ACD	33	image	true	false	4.5 - 15	3619×5226	285-24368
VisDrone-CC2020	3360	1	true	false	0.118 - 0.706	1920×1080	25-421
AgoraSet	1519	25	true	true	4.5	640×480	1-180
WorldExpo'10	3980	1/30	false	false	perspective	576×720	1-253

To determine the performance of the filters, the Mean Absolute Error of the estimated crowd counts and the Mean Absolute Error of the estimated crowd counts' slope is calculated. Let c be the crowd count and $\bar{c}$ be the estimated crowd count, then the MAE of the estimated crowd count and its slope can be calculated as follows:

$$MAE_{crowd\,count} = \frac{\sum_{k=1}^{k=n} |c_k - \bar{c}_k|}{n} \tag{13}$$

$$MAE_{slope} = \frac{\sum_{k=2}^{k=n} |c_k - c_{k-1} - (\bar{c}_k - \bar{c}_{k-1})|}{n-1} \tag{14}$$

Both MAEs indicate better performance for values closer to zero and worse performance for higher values. They differ in that the MAE_{slope} shows how good the filters smooth the data and the $MAE_{crowd\,count}$ shows how close the counts are to the actual crowd count. Since the focus of this paper lays on smoothing the counts, it is the overriding goal to reduce the MAE_{slope} without increasing the $MAE_{crowd\,count}$.

4.1 VisDrone-CC2020

Tests on the VisDrone data set are conducted to test the behavior of the filters on a data set from the aerial domain the crowd counters are fine-tuned on.
Table 2 shows the results. In general, the filters smooth the temporal courses of the estimated crowd counts. This is shown by a decrease in the $\text{MAE}_{\text{slope}}$. The values of the raw crowd counters are reduced by 56% to 65%.
The Kalman filter applied to the whole frame smooths the data the most. This contradicts the initial assumption that the Kalman filter works better on smaller grids. The problem here is that the pixel values of the estimated density maps are not consistently positive. Therefore, if the density map is divided into smaller grids, some of the grids contain negative crowd counts. If one inputs these negative values into a Kalman filter, the filter outputs arbitrarily high or low numbers over time. To solve this problem, the count is set to zero, the process noise variance is set to 1, and the measurement noise variance is set to 1000 when a negative crowd count occurs.
Although this approach makes the grid-based Kalman filter work, it does not enable the filter to develop its full potential. Yet, it comes with a more than welcome side effect. By setting the negative crowd counts of single grids to zero, the grid-based Kalman filter returns the best results for the $\text{MAE}_{\text{crowd count}}$.
The displayed results of the grid-based Kalman filter are obtained using 1x1 meter grids. Tests with 3x3 and 10x10 meter grids were conducted. Yet, it turned out that on the VisDrone as well as on the other data sets, 1x1 meter grids return the best results.

Table 2. Results on the VisDrone-CC2020 data set. The MAE_{slope} is reduced by 56% to 65%. All filters successfully smooth the estimated crowd counts

	$\textbf{MAE}_{\textbf{crowd count}}$		$\textbf{MAE}_{\textbf{slope}}$	
	CSRNet	MRCNet	CSRNet	MRCNet
Unfiltered	29.30	21.08	4.23	3.98
Kalman	30.89 (+5%)	18.31 (-13%)	**1.47 (-65%)**	**1.44 (-64%)**
$\text{Kalman}_{\text{grid}}$	**25.78 (-12%)**	**16.52 (-22%)**	1.60 (-62%)	1.75 (-56%)
Particle	28.63 (-2%)	17.00 (-19%)	1.59 (-62%)	1.62 (-59%)

4.2 AgoraSet

In order to see how the filters work on data with more realistic frame rates (25 fps), tests on the AgoraSet are conducted. Although its frame rate covers a more realistic use case, its data is artificial. To the best of our knowledge, non-artificial alternatives are not publicly available due to the high effort that comes with annotating such data sets.
When speaking of the AgoraSet only the first sequence of the AgoraSet is meant. This is because, all sequences cover a pretty similar scenario from a crowd counting perspective. The backgrounds are monotonous and do not significantly differ. In addition to that, pretty much all of the sequences' temporal courses of the crowd count resemble a parabola with a downward opening.
The tests were directly conducted after the tests on the VisDrone-CC2020 data set without fine-tuning the crowd counters on the AgoraSet or reconfiguring the Kalman filter's

parameters. Nevertheless, the filters are able to smooth the temporal courses of the crowd count even more. The MAE_{slope} is reduced by 68% to 88%. Again, the Kalman filter applied to the whole image smooths the data the best. Yet, the particle and grid-based Kalman filter return good results as well. If only the $MAE_{crowd\ count}$ is considered, the particle filter returns the best results.
The results show that the filters smooth the data even better on data sets with higher frame rates. In addition to that, the tests indicate that the initialization of the Kalman filter's parameters on a previous data set does not impair its performance on another data set.

Table 3. Results on the first sequence of the AgoraSet. Although the crowd counting algorithms and filters have not seen data from the AgoraSet before, the filters are able to smooth the estimated crowd counts even more

	$\mathbf{MAE_{crowd\ count}}$		$\mathbf{MAE_{slope}}$	
	CSRNet	MRCNet	CSRNet	MRCNet
Unfiltered	26.06	43.72	1.5	2.01
Kalman	27.55 (+6%)	44.45 (+2%)	**0.27 (-82%)**	**0.25 (-88%)**
$Kalman_{grid}$	25.49 (-2%)	43.98 (+1%)	0.45 (-70%)	0.65 (-68%)
Particle	**25.40 (-3%)**	**43.84 (±0%)**	0.39 (-74%)	0.42 (-79%)

4.3 WorldExpo'10

Finally, tests are conducted on a perspective data set that has a significantly lower frame rate than the previous ones. Since the WorldExpo'10 data set does not contain aerial images, the particle filter as conceptualised in this paper cannot be applied. The grid-based Kalman filter also runs into the problem that its hard to divide a perspective image into same sized grids. To handle this problem, the face length of a person in the front and back of a random frame is measured. Assuming that the average face length of a person is 23 centimeters, one can calculate two GSDs - one for the front and one for the back of the image. The average of the two GSDs is used to determine the grids. It should be noted that this approach is only a hot-fix to make the filter work and that grids in the back still cover larger areas than grids in the front.
Table 4 shows the results. It can be seen that the MAE_{slope} does not significantly improve. However, this is much more due to the low frame rate and not the perspective of the data set. The time between two frames of the WorldExpo'10 data set is 30 seconds. Due to the large time interval between the frames, it can be said that consecutive frames do not hold any temporal information that could be incorporated by the filters. To sum up, the results stress out the importance of the frame rate when applying the filters. If the time interval between consecutive frames becomes too large, the filters do not improve the estimates of the crowd counters. Yet, since a frame rate of 1/30 fps does not capture a realistic scenario, the results of the tests on the WorldExpo'10 do not contradict with the applicability of the filters in a real-life situation.

Table 4. Results on the WorldExpo'10 data set. Due to the low frame rate, adjacent frames do not share temporal information that can be incorporated by the filters

	$\mathbf{MAE_{crowd\ count}}$		$\mathbf{MAE_{slope}}$	
	CSRNet	MRCNet	CSRNet	MRCNet
Unfiltered	17.15	18.12	8.8	8.6
Kalman	19.74 (+15%)	18.14 (±0%)	8.2 (-7%)	8.4 (-2%)
$\text{Kalman}_{\text{grid}}$	19.73 (+15%)	17.95 (-1%)	9.54 (+8%)	8.9 (+3%)

5 Conclusions

The paper shows that the oscillating counts estimated by current crowd counters for video data can be smoothed using Bayesian filters. As a measurement of the false oscillation the $\text{MAE}_{\text{slope}}$ is introduced. Provided that the frame rate of a data set is large enough that consecutive frames share temporal information, all of the three filters are able to smooth the temporal course of the crowd count without significantly increasing the $\text{MAE}_{\text{crowd count}}$. Future work may further address the maturation of the concepts. The development of Bayesian filters that smooth the estimated crowd counts is still in its infancy. A variety of traffic flow models exist that can be used to develop new concepts that may smooth the crowd counts even more.

References

1. Gao, G., Gao, J., Liu, Q., Wang, Q., Wang, Y.: Cnn-based density estimation and crowd counting: A survey (2020)
2. Wu, X., Xu, B., Zheng, Y., Ye, H., Yang, J., He, L.: Fast video crowd counting with a temporal aware network. Neurocomputing **403** (Aug 2020) 13–20
3. Li, Y., Zhang, X., Chen, D.: Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes (2018)
4. Bahmanyar, R., Vig, E., Reinartz, P.: Mrcnet: Crowd counting and density map estimation in aerial and ground imagery (2019)
5. Elmar, G.: Bayes–Filter zur Genauigkeitsverbesserung und Unsicherheitsermittlung von dynamischen Koordinatenmessungen. dissertation, Friedrich–Alexander–Universität (2014)
6. Rhudy, M.B., Salguero, R.A., Holappa, K.: A KALMAN FILTERING TUTORIAL FOR UNDERGRADUATE STUDENTS. (February 2017) Accessed: 2021-01-23.
7. Simon, Maskel an Neil, G.: A tutorial on particle filters for on-line nonlinear/non-gaussian Bayesian tracking - Target Tracking. (2001)
8. Treiber, M.: Skript zur vorlesung verkehrsdynamik und -simulation (2017)
9. Farnebaeck, G.: Two-frame motion estimation based on polynomial expansion. In: Scandinavian Conference on Image Analysis. Volume 2749. (2003) 363–370
10. Du, D., Wen, L., Zhu, P., Fan, H., Hu, Q., Ling, H., Shah, M., ..., Zhao, Z.: Visdrone-cc2020: The vision meets drone crowd counting challenge results (2021)
11. Allain, P., Courty, N., Corpetti, T.: Agoraset: a dataset for crowd video analysis. In: 1st ICPR International Workshop on Pattern Recognition and Crowd Analysis. (2012)
12. Zhang, C., Li, H., Wang, X., Yang, X.: Cross-scene crowd counting via deep convolutional neural networks. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2015) 833–841