ArenaPeds: Pedestrian Flow in a highly crowded stadium

Jan Erik van Woerden

30 October 2020

Contents

Chapter 1

Introduction

A novel general purpose Line of Interest method, using a multi-headed network and a pixel-wise accumulator.

In a world with increasing accessibility to information, it is important to summarize all this information into useful information for the intended users. In this case we have access to a large amount of surveillance camera's which are used during busy events. To monitor all these camera's it requires a lot of manual watching what is happening on those camera's. There has a lot of research been done on reducing the amount of manual watching and translating the information into more actionable and measurable tasks.

In recent years several papers have been published about extraction pedestrian information using camera's under the subfield Crowd Counting [Chan and Vasconcelos, 2008, Zhao et al., 2016, Wang et al., 2020, Li et al., 2018, Fang et al., 2019, Liu et al., 2019b]. In this thesis we will focus on the Line of Interest problem. The goal is to count the amount of pedestrians that cross a specified line in a certain time frame. In area's with low density of pedestrians a generic object tracking solution would suffice. In high density crowds the results of the solutions will degrade quickly. (Because most object trackers can handle up to 50 people, YOLOv4 paper and another solution)

To handle high density crowds specialized solutions are presented over the years. Those methods typically combine two components, crowd density prediction (Region of Interest) and flow estimation. (Already citing papers, same as below?). Traditionally the density prediction is done using keypoint extraction and feeding those keypoints in a regression model (Cite some papers). Flow estimation is typically done with a Lucas-Kanade method (papers who implement this method).

More recently the rise of Convolutional Networks made it possible to improve both components. Both crowd counting (Papers) and flow estimation (Papers) research fields have state of the arts which heavily rely on convolutional networks. Several papers combine these CNN focused methods to a Line of Interest solution (Papers, Zhao et al. (2016) and more) which give good and promising results.

One of the major drawbacks of Neural Networks is the huge amounts of that that is required for training these networks (Probably should be papers, but this is like a very trivial thing in the AI field, so should it?). For the crowd density estimation several public datasets are available and labeling new images can be done easily. Labeling data for the flow estimator is much more difficult and time intensive. In earlier papers (Previous paragraph papers) this is done, but this is less feasible for implementation in actual applications.

Should I explain this more in detail. No real paper, but can show somewhere my calculation of time

Besides supervised learning of the flow estimation, more and more traction is gained for the unsupervised flow estimation research area. These methods provide a much more scalable solution for real world applications and come closer to supervised flow estimation performance. In this thesis we will focus on a solution which utilizes the unsupervised flow estimation. (See research question 1)

Besides the huge amount of data another problem with several convolutional neural networks is the running time. CNN's can take a long time to produce their predictions. To make methods applicable for real life applications quickly producing results with minimal performance degrading is crucial. This will be our second focus of this work. (See research question 2)

Lastly another major focus of the work is to make the proposed solution versatile and make it possible to easily use for new scenes. So we will focus on the minimization of extra required data when a the model is deployed on another scene. (See research question 3)

1.1 Contributions

In this thesis, The contribution is in threefold.

New, so maybe change the introduction based on this

1. A new challenging task is described around demands of real world use-cases.
2. To address the task a new real world dataset called AmsterdamPeds is published, additionally new labeled data of the Fudan ShanghaiTech dataset [Fang et al., 2019] is published for Line of Interest.
3. A state-of-the-art model is proposed on the described task as well as state-of-the-art on several earlier tasks.
4. Propose a model which is quick and robust enough to

Let's hope so on earlier tasks

1.2 Thesis outline

The rest of this thesis is divided into the next chapters:

- **Related work**, which explains more in depth related work and tries to give a good background to understand the proposed solution.
- **Method**, presents the method of the proposed solution. (Nothing fancy to tell about)
- **Implementation**, presents the hyperparameters, evaluation methods and the used datasets.
- **Results**, displays the results for the discussion.
- **Discussion**, discusses the research questions and try to answer it based on the results.
- **Conclusion**, wraps it up and summarizes what we can conclude

Chapter 2

Background

In this chapter a selection of terms is explained which gives a basis to understand the rest of this thesis. This background is created with the assumption that the reader has a basic background in Machine Learning and (Convolutional) Neural Networks.

2.1 Region of Interest

The Region of Interest problem is a widely studied problem in which the goal is to estimate the amount of pedestrians given a single image. Directly predicting the count given a Neural Network is a hard task, because of the lack of supervision, this would require a large amount of samples to accurately solve this task. All recent State-of-the-Art methods therefore use an intermediate representation to give the model enough supervision to perform Crowd Counting with a low amount of training samples.

In the early days of Crowd Counting several methods have been proposed which use *detection-based* methods to estimate the amount of pedestrians [Dalal and Triggs, 2005, Dollár et al., 2012] . Several papers were introduced which tried to detect only the head [Subburaman et al., 2012] and others tried to focus on general part detection [Wu and Nevatia, 2007, Lin and Davis, 2010]. These methods rely on individually detecting the pedestrians. This becomes much harder when occlusion of the pedestrians start to happen. This is why the performance of these methods start to degrade when the density of the pedestrians in an image start to increase.

Later papers introduced a *regression-based* solution, which tries to predict the amount of pedestrians in crowd blobs [Chan and Vasconcelos, 2009, Idrees et al., 2013, Zheng et al., 2019]. Using SVM or other regressor methods and several features such as the amount of foregrounds pixels of the blob and detected key points the count inside crowd blobs were predicted. Regression based solutions were an improvement over the detection methods, but still lack the capabilities to estimate pedestrians counts in highly occluded areas.

2.1.1 Density Map

With the introduction of Convolutional Neural Networks in the field of Crowd Counting density maps were proposed as well to count pedestrians [Zhang et al., 2016, Liu et al., 2019b, Li et al., 2018]. A *density map* (Figure 2.1) used for Region of Interest is a map which represents the density of pedestrians of each pixel. The

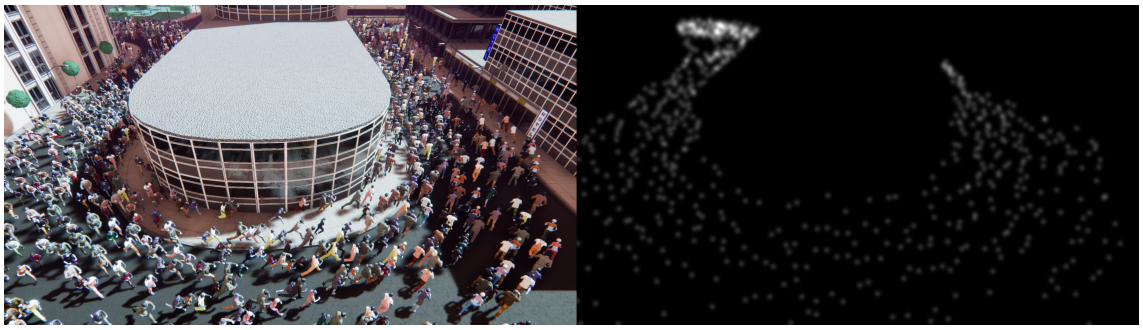

Figure 2.1: Example of generated density map on the right side, for the left image

density map is generated by taking the locations of each pedestrian ($p = \begin{bmatrix} x_p \\ y_p \end{bmatrix}$ in equation 2.1) and place those locations on the the density map.

Individual dots are very hard for a Neural Network to detect correctly and are proned to errors. To circumvent this a Gaussian shaped circle is created around this location, still with with a sum of 1. The amount of pedestrians in the frame can be extracted from the density map by taking the sum over all the pixels of the density map (Equation 2.2, where $D_t(p)$ is the density for location $p = \begin{bmatrix} x_p \\ y_p \end{bmatrix}$ for trainings frame t).

$$D_t(p) = \frac{1}{2\pi\sigma_p^2} \sum_{p \in P} e^{\frac{(x_p - x)^2 + (y_p - x)^2}{-2\sigma_p^2}} \tag{2.1}$$

$$C_t = \sum_{p \in P} D_t(p) \tag{2.2}$$

Several methods have been presented to optimize the generation of density maps [Zhang et al., 2016, Li et al., 2018, Wan and Chan, 2019]. For most medium dense frames the difference in methods is minimal. Often in benchmarks with medium dense frames a fixed sigma is used ($\sigma_p = \sigma_i$ in equation 2.1). For highly dense frames the use of different methods can have a difference, especially when the difference in size between close pedestrians and pedestrians in the background is large [Li et al., 2018].

A bit more papers of recent improvements

2.2 Flow Estimation

The research which is done on the Flow Estimation problem is widely used. Approaches on this topic can be used in a wide range of applications which makes it very interesting. Already in the early 1980's Horn and Schunck [Horn and Schunck, 1981] published the first paper which tried to predict flow. Since then lot's of different approaches have been published [Mémin and Pérez, 1998, Bruhn et al., 2005, Brox et al., 2014]. Long conventional mathematical approaches have ruled the flow estimation field. Later also learnable models were introduced [Pock et al., 2008, Wedel et al., 2009].

Recent papers however make use of Convolutional Neural Network based models [Dosovitskiy et al., 2015, Ilg et al., 2016, Sun et al., , Ranjan and Black, 2017,

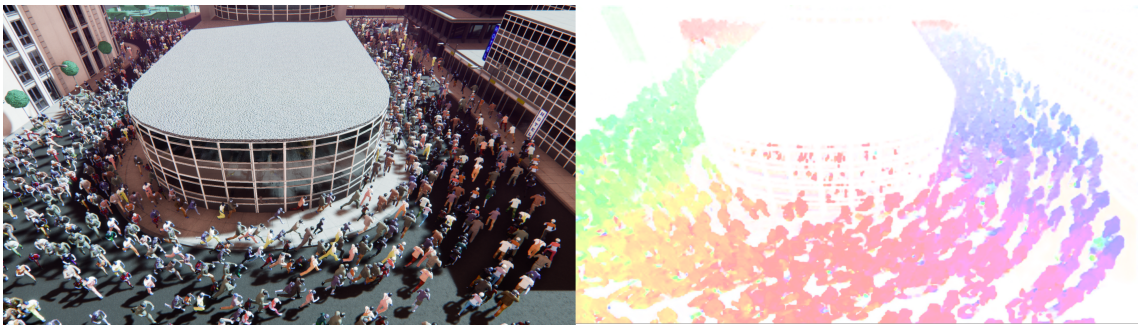

Figure 2.2: Example of generated velocity map on the right side, for the left image

Hui et al., 2018]. These model predict pixel-precise velocity maps. The *velocity map* (Figure 2.2) is a map which predict per pixel of the frame the amount of movement to another location. In equation 2.3, $V_t(p)$ shows the velocity map as a difference between the location of the pixel in the current frame (p) and the location of this pixel in the next frame ($N_t(p)$).

$$V_t(p) = N_t(p) - \begin{bmatrix} x_p \\ y_p \end{bmatrix} \tag{2.3}$$

Creating a real world dataset that utilizes the power of pixel-wise flow estimation is very hard [Dosovitskiy et al., 2015]. There are no real world devices which could capture both video and create pixel perfect ground-truths to train the flow estimation models on. Most of the flow estimation benchmarks are therefore generated videos. Computer 3D-engines make it possible to generate pixel-perfect flow estimation based on the generated videos in the engine.

However the large gap in domain and scene between the generated datasets and real world applications [Liu et al., 2008]. Recent supervised papers [Dosovitskiy et al., 2015, Sun et al.,] tend to overfit on the datasets. Therefore perform rather poor on real world applications. One solution and promising direction is unsupervised learning [Yu et al., 2016, Janai et al., 2018, Liu et al., 2019a, Liu et al.,]. Early papers only predicted non-occluded pixels [Yu et al., 2016, Janai et al., 2018], but recent papers use methods to estimate occluded pixels as well [Liu et al., 2019a, Liu et al.,]. Further details about these methods in related work.

2.3 Line of Interest

Line of Interest is very similar to Region of Interest. Where Region of Interest is the interest of the amount of people inside the ROI, the Line of Interest is the focus on the amount of pedestrians that cross the specified line during a certain timeframe. This LOI is defined as a single line between two points p_1 and p_2.

Add an image with a line drawn inside the TUB dataset

With the Line of Interest problem the goal is to give the amount of pedestrians crossing of each side given a set of frames (a pre-captured video or video stream). The output of the prediction should give two numbers c_1 and c_2 which are the amount of pedestrians crossing from each side.

Only a handful of papers are published about Line of Interest. In the earlier papers [Ma and Chan, , Cao et al.,], slicing was a widely used approach to estimate the Line of Interest. With slicing a small area, called the LOI area, is taken around

the LOI. Over a set of consecutive frames each slice of the frame was taken and stitched together into a single image. On the images slow walking pedestrians appear rather wide and fast walking pedestrians shallow. By counting the amount of pedestrians present on the stitched image, the total amount of pedestrians crossing the line can be counted.

The area is defined by all the pixels that have a maximum distance to the LOI of d and can be projected on the LOI. When projected, the pixels fall between p_1 and p_2.

Recent papers discard this method [Zhao et al., 2016, Zheng et al., 2019], because it makes it hard to track pedestrian with different speeds and walking in different directions give artifacts which make it hard to track those pedestrians [Zhao et al., 2016]. The slicing method is replaced with an actual frame by frame prediction method. Using two consecutive frames the amount of pedestrians crossing the line is measured. These newer methods predict both location and direction of the pedestrian.

Based on these new papers, the problem of Line of Interest is divided into three separate problems. Locating the pedestrians (Region of Interest), estimate the direction (Flow Estimation) of the pedestrians and combining these two streams of information into the count for Line of Interest. Further details of this approach will be provided in related work.

Chapter 3

Related Work

In this chapter we try to explain a couple of key papers on which the proposed methods are build.

3.1 Flow Estimation

3.1.1 PWCNet

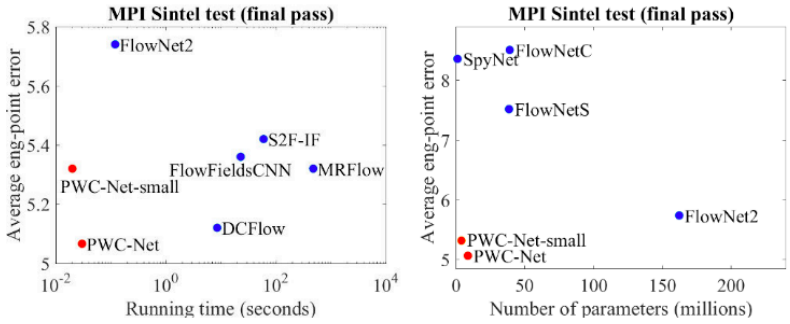

Figure 3.1: PWCNet compared to other existing flow estimation architectures

A popular Flow Estimation network is PWCNet [Sun et al.,]. It uses the original ideas of FlowNet, but it improves FlowNet in a lot of ways. FlowNet traditionally is used to fully predict the flow with a neural network. PWCNet massively reduces the number of weights (See figure 3.1), which results in faster training and much quicker prediction. Additionally the network shows a higher accuracy on several benchmarks.

PWCNet uses a pyramid shape architecture to predict the velocity map (See figure 3.2). The encoder encodes both input frames seperately into two feature volumes. During decoding several decoding steps are taken. After the initial velocity map estimation, each decoding step upscales the so-far estimated map and further refines the map. At the start of each decoding step the estimated velocity map is used to warp the feature volume of the second frame and correlate this warped volume with the volume of the first image to focus on area of refinement in the velocity map.

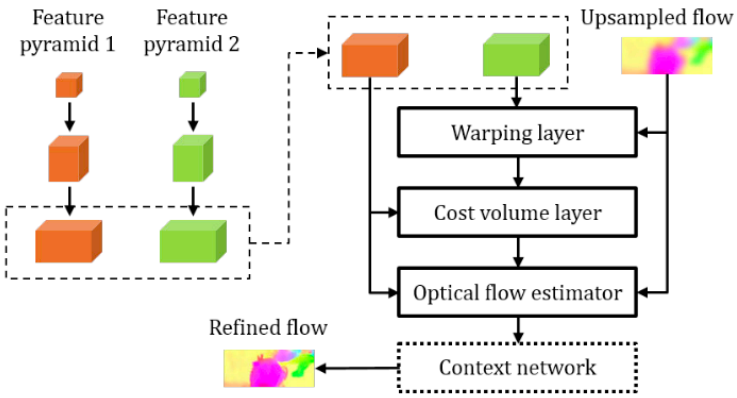

Figure 3.2: PWCNet architecture

3.1.2 DDFlow

In DDFlow [Liu et al., 2019a] a general method is proposed to estimate a velocity map in an unsupervised matter. Earlier methods already proposed several methods to optimize using a photometric loss [Yu et al., 2016] and ignore occluded-pixels during loss calculation [Janai et al., 2018]. However all these earlier methods used handcrafted methods to estimate the occluded-pixels.

The paper [Liu et al., 2019a] proposes a method which learns occluded-pixels using a distillation from unlabeled data without supervision of humans to label the ground truth. Because of the generality of the method, the method can be wrapped around all existing supervised flow estimation architectures.

The method uses a teacher and a student approach to train the occluded pixels. The teacher network is trained on all the non-occluded pixels, with exclusion of the occluded pixels using only the photometric loss with occlusion-awareness. After training the teacher network, the velocity maps of the teacher network are used as ground-truth for the student network.

The student network is then trained on patches of the original predicted velocity map. On the borders of the patches, occluded pixels will appear, because moving pixels from inside the patched frame will move to outside the patched frame. These border pixels will be marked as occluded pixel by the student network, but will be non-occluded pixels according to the ground-truth of the teacher network.

$$\begin{aligned} L_p = &\sum \psi \left(I_1 - I_2^w\right) \odot \left(1 - O_f\right) / \sum \left(1 - O_f\right) \\ &+ \sum \psi \left(I_2 - I_1^w\right) \odot \left(1 - O_b\right) / \sum \left(1 - O_b\right) \end{aligned} \tag{3.1}$$

The photometric loss [Yu et al., 2016] with occlusion-awareness [Janai et al., 2018] proposed in the earlier papers is defined in equation 3.1. Where I_1 and I_2 are the full input frames and I_i^w the backwarped image based on the predicted velocity map. Additionally O_b and O_f are masks for the occluded pixels. These maps are calculated by checking if the mismatch between forward flow and backward flow is too large.

For the student model the photometric loss is used together with the loss for occlusion (equation 3.2). The loss of the student model then just $L_p + L_o$. In equation 3.2, $\widetilde{w}_f$ and $\widetilde{w}_b$ are the predicted velocity map forward and backward

Maybe to background? Additionally add unflow to the related work, Meister2018 for the optimized census loss

using the student model. w_b^p and w_f^p are the cropped patches from the ground-truth velocity map predicted by the teacher model. M_f and M_b are just defined as the difference of occluded pixels between the ground-truth of the parent and the teacher ($M_f = \text{clip}\left(\widetilde{O}_f - O_f^p, 0, 1\right)$. These pixels are not occluded in the full frame and occluded in the patch. This means that the ground-truth using distillation is available.

$$\begin{aligned} L_o = &\sum \psi\left(\mathrm{w}_f^p - \widetilde{\mathrm{w}}_f\right) \odot M_f / \sum M_f \\ &+ \sum \psi\left(\mathrm{w}_b^p - \widetilde{\mathrm{w}}_b\right) \odot M_b / \sum M_b \end{aligned} \tag{3.2}$$

3.2 Line of Interest

3.2.1 Instant LOI counting

Zhao et al. [Zhao et al., 2016] introduces a new approach to calculate the amount of pedestrians crossing the Line of Interest. Instead of slicing a range of consecutive frames around the Line of Interest and stitching them together, they present a method that directly predicts the amount crossing pedestrians using two consecutive frames.

They use both a density map and velocity map and finally merge them together to obtain the LOI counts. Both the maps are trained fully supervised. Because annotating raw velocity maps is a very hard task, they simplify the velocity map in disk-shaped area's around the pedestrian locations. Annotating each frame is still very demanding, but lowers the amount of labeling significantly in comparing to full velocity maps.

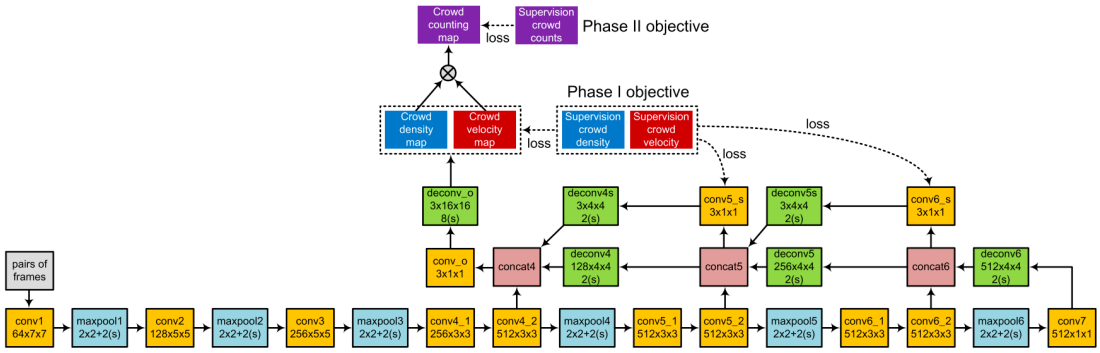

Figure 3.3: Pipeline of Zhao et al. with FlownetSimple as architecture, where in the last layer both the density map as the velocity map are predicted

They are as well the first who predict both the density map and the velocity map in a single Convolutional Neural Network. A network they use a simplified model of FlowNet. Because the velocity map and density map are so similar in their location, they predict both the velocity map and the density map in the final layer (See figure 3.3).

To finally merge both the density map and velocity map together they use a pixel-wise approach. Where they turn te density map and velocity map in a directional counting map $C_t = D_t \otimes V_t$. According to the paper the directional counting map

gives the amount of pedestrians crossing that pixel between the time of the two frames.

$$\begin{aligned} c_{1,t} &= \sum_{\{p|\cos(\theta_p)\geq 0\}} \sqrt{C_{t,x}(p)^2 + C_{t,y}(p)^2} \cdot \cos\left(\theta_p\right) \\ c_{2,t} &= \sum_{\{p|\cos(\theta_p)<0\}} \sqrt{C_{t,x}(p)^2 + C_{t,y}(p)^2} \cdot \left(-\cos\left(\theta_p\right)\right), \end{aligned} \tag{3.3}$$

$$c_1 = \sum_{\{t|t\in T\}} c_{1,t}, \quad c_2 = \sum_{\{t|t\in T\}} c_{2,t} \tag{3.4}$$

To calculate per frame pair the amount of pedestrians crossed the line, a set of locations p around the LOI is taken. Then the directional counting map is normalized and summed together in equation 3.3, where θ_p is the angle between $V_t(p)$ and the LOI.

To calculate the total line crosses over a certain timeframe, the results of equation 3.3 can be summed as in equation 3.4.

3.2.2 Region-level LOI counting

Zheng et al. [Zheng et al., 2019] provides a fast method of predicting the Line of Interest. The paper outperforms [Zhao et al., 2016] in the UCSD benchmark. Zheng et al. discusses the problems with Neural Networks and the complexity the models, which makes it hard to run the models real time. It therefore introduces a non-CNN based method based on a SVM, linear regression and the Lucas-Kanade optical flow tracker.

It uses the idea of [Zhao et al., 2016] to discard the slicing method and uses pairwise prediction and uses the same method to merge density and velocity. Because the methods used in [Zheng et al., 2019] are not on a pixel-level, a method is proposed to bin on a region-level. In equation 3.5 a single velocity $v_{t,r}$ is calculated with a weighted average over all moving keypoints inside the region.

$$v_{t,r} = \frac{\sum_p w_r(p) \cdot v^{(t)}_{towards}(p)}{\sum_p w_r(p)} \tag{3.5}$$

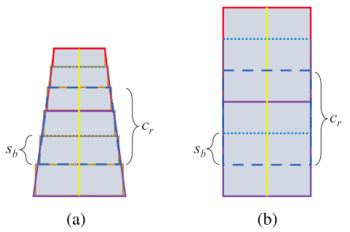

Figure 3.4: Regions around the LOI with skewness and normalized to a straight LOI

Additional to the region-level LOI counting, they introduce the method to skew the region to take into account the perspective of the camera view (See figure 3.4). This helps the SVM and linear regressor to more accurately predict the amount of pedestrians inside the region.

Chapter 4

Method

4.1 Instant LOI counting

In our thesis two method are use to come up with instant LOI counts. Both methods use the same approach, but the region-wise uses the pixel-wise approach on a region-level to improve smoothing of the velocity and density map. This will help when the density map and velocity map don't align correctly.

4.1.1 Pixel-level counting

The proposed method by [Zhao et al., 2016] in related work section 3.2.1 uses some simplifications. By reframing the pixel-level counting in the following way. The approach is much more theoretical correct.

We define v_{perp} as the normalized directional vector perpendicular to the LOI (Two solutions are perpendicular on the LOI and this defines sides 1 and 2 of the LOI counting). Then we define the collection of the pixels on the left side of the LOI and inside the LOI area as M_1 (side 1) and the pixels on the right side (side 1 and inside the LOI area) as M_2.

Draw picture with LOI area, sets of pixels and vperp

The velocity towards the LOI is then defined as the dot-product of V_t and v_{perp} (Equation 4.1).

$$Q_t(p) = V_t(p) \cdot v_{perp} \tag{4.1}$$

$$\begin{aligned} c_{1,t} &= \sum_{\{p \in M_1 | Q_t(p) > 0\}} C_t(p) \cdot \frac{Q_t(p)}{d} \\ c_{2,t} &= \sum_{\{p \in M_2 | Q_t(p) < 0\}} C_t(p) \cdot \frac{-Q_t(p)}{d} \end{aligned} \tag{4.2}$$

Then then the LOI count on timestep t is then defined in equation 4.2. Where $\frac{Q_t(p)}{d}$ defines the percentage that the density on the specific pixel, has crossed the LOI area. Lastly we can sum the count over a timespan into a single count for each side as in equation 3.4.

4.1.2 Region-level counting

To smooth the velocity around the LOI a new method is proposed. Instead of using the individual pixel velocities for the density map, binning is applied. Several regions have been defined to smooth out the velocity and density. Each region is defined $M_{1,r}$, which is a subset of M.

$$
\begin{aligned}
c_{1,t} &= \sum_{r}^{R} \frac{\sum_{\{p \in M_{1,r} | Q_t(p) > 0\}} Q_t(p)}{d \cdot \sum_{\{p \in M_{1,r} | Q_t(p) > 0\}} 1} \cdot \sum_{\{p \in M_{1,r} | Q_t(p) > 0\}} C_t(p) \\
c_{2,t} &= \sum_{r}^{R} \frac{\sum_{\{p \in M_{2,r} | Q_t(p) > 0\}} -Q_t(p)}{d \cdot \sum_{\{p \in M_{2,r} | Q_t(p) > 0\}} 1} \cdot \sum_{\{p \in M_{2,r} | Q_t(p) > 0\}} C_t(p)
\end{aligned}
\tag{4.3}
$$

The same setup is applied as in section 4.1.1, but equation 4.2 is replaced for equation 4.3. Where for each region the average velocity is taken (towards the LOI), which is then multiplied with the sum of the density (towards the LOI) inside the region.

Display a with regions defined LOI

4.1.3 Adaptive LOI area

How to adaptively estimate the LOI area.

More thorough seeking in this

4.2 Network

For our system we split up the program into 3 parts. The Crowd Counting models, the Flow Estimation models and the Line of Interest. The methods which combine the Crowd Counting and the Flow Estimation in a single model are called Crowd Flow models (Useful as directory in the final code)

train.py # Train the models (All models should be trainable in this file, because they all share the same kind of dataloader)

test.py # Test the LOI methods using a trained network from train.py (Based on the datasets provided)

run.py # Run from a video stream live. (Would be super cool and very useful to actual demo this thing)

loi_models/ (Line of Interest models) - Pixelswise/Regionwise

fe_models/ (Flow Estimation models) - PWCNet cc_models/ (Crowd Counting models) - CSRNet cf_models/ (Crowd Flow models) - New Network

Chapter 5

Implementation

5.1 Models

5.1.1 Baseline

As baseline we use

5.1.2 Shared encoder

5.1.3 Flow enhancing

5.2 Environment

5.2.1 Loss function

$$L_t = L_v + \lambda \cdot L_c \tag{5.1}$$

The loss function for the final training is given in equation 5.1. Where L_v is the loss function for the velocity map. We finally applied the occlussion photometric loss on all results (Equation 3.1).

L_c is used for the density map. Where the L2 loss (Mean squared error) is applied for comparing the ground truth density map with the predicted map.

For λ a default of 5 is applied when not mentioned else.

5.2.2 Optimizer

For all the experiments the Adam optimizer is used with a learning-rate of $5e^{-5}$. No regularization is applied.

5.2.3 Augmentation

To augment the dataset several augmentations will be applied on the trainingssamples. First a crop of the image is made. Ranging from 1/3 and 1/6 of the total image size. Then the cropped image is resized to a size of 1/4 of the original image (So 1/2 the width and 1/2 the height). Lastly the cropped image is half of the time flipped horizontally.

5.2.4 Metrics

For both the LOI and the ROI the same metrics are used. Both are measured with the Mean Average Error and the Mean Squared Error.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|C_i - P_c^{(i)}| \tag{5.2}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(C_i - P_c^{(i)})^2 \tag{5.3}$$

For the ROI the MAE and the MSE are defined as equation 1 and 2. Where $P_c^{(i)}$ is the predicted density map for the given frame.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|G_l^{(i)} - P_l^{(i)}| + |G_r^{(i)} - P_r^{(i)}| \tag{5.4}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(G_l^{(i)} - P_l^{(i)})^2 + (G_r^{(i)} - P_r^{(i)})^2 \tag{5.5}$$

For the LOI the MAE and MSE are defined as quation 1 and 2. Where $G_l^{(i)}$ is the ground truth for sample i for side left-to-right. And $P_r^{(i)}$ is the predicted value for right-to-left.

5.3 Datasets

In this thesis we make use of four different datasets. Three of them are already public datasets and one dataset is created using City of Amsterdam footage.

5.3.1 UCSD

The UCSD Pedestrian dataset [Chan and Vasconcelos, 2008] is a public dataset created in 2008. The dataset is in black and white and has only a resolution of 234x158. It has 6 scenes with each around 30 videos which each has around 10 seconds at 10fps of footage. Only a small amount of videos has precise labeled data for Crowd Counting. Most of the other video's have small parts of information about the pedestrians crossing.

The UCSD dataset is in previous papers used as default benchmark. Although other datasets do represent the capabilities of the presented methods much better, this dataset is used to give some comparison with older methods.

Explain more in detail which labeled data is present and add other data papers

5.3.2 CrowdFlow

The CrowdFlow dataset [Schroder et al., 2019] is a public dataset generated at the TU Berlin in 2018. The dataset contains 10 sequences generated from 5 different scenes. Each scene is captured once with a drone view camera and once with a fixed view camera. Each scene is a virtual urban environment and it is generated in the Unreal Engine, a 3D-engine. Each sequence is roughly 10 seconds long with 25 fps with a resolution of 1280x720. The generated sequences are dense in pedestrians and

have up to 1451 pedestrians in a single frame. All the camera views are captured from a high surveillance style view.

5.3.3 Fudan-ShanghaiTech

The Fudan ShanghaiTech dataset [Fang et al., 2019] is a public dataset with 100 videos of 13 different scenes. Each video contains 6 seconds of footage at 25 fps and have a resolution of 1920x1080 or 1280x720. The scenes have between 20-100 pedestrians per frame. In each frame the pedestrians in the frame are labeled with a bounding-box and a center-point of the bounding-box.

5.3.4 ArenaPeds

For this thesis we have access to a dataset of the City of Amsterdam. It has xxx amount of footage of environments with crowds ranging from 10 pedestrians in the frame to 1000 pedestrians a few hours later. Only a tiny proportion is labeled with where each pedestrian is. So there is no labeled data on the Crowd Direction, only for the Crowd Counting. This is the reason why we want to try to give unsupervised learning a shot.

Chapter 6

Results

6.1 UCSD

- Table with Line of Interest

Method (ROI)	MAE	MSE
CSRNet	0.0	0.0
Baseline	0.0	0.0
Baseline + flow	0.0	0.0

Table 6.1: The ROI results for UCSD

Method (LOI)	MAE	MSE
CSRNet + PWCNet	0.0	0.0
Baseline	0.0	0.0
Baseline + flow	0.0	0.0

Table 6.2: The LOI results for UCSD

6.2 CrowdFlow

Method (ROI)	MAE	MSE
CSRNet	0.0	0.0
Baseline	0.0	0.0
Baseline + flow	0.0	0.0

Table 6.3: The ROI results for CrowdFlow

Method (LOI)	MAE	MSE
CSRNet + PWCNet	0.0	0.0
Baseline	0.0	0.0
Baseline + flow	0.0	0.0

Table 6.4: The LOI results for CrowdFlow

6.3 Fudan-ShanghaiTech

Method (ROI)	MAE	MSE
CSRNet	0.0	0.0
Baseline	0.0	0.0
Baseline + flow	0.0	0.0

Table 6.5: The ROI results for Fudan

Method (LOI)	MAE	MSE
CSRNet + PWCNet	0.0	0.0
Baseline	0.0	0.0
Baseline + flow	0.0	0.0

Table 6.6: The LOI results for Fudan

Method	FPS
CSRNet+PWC	0.0
CSRNet+PWC+optimized	0.0
Baseline	0.0
Baseline+flow	0.0
Baseline+flow+optimized	0.0

Table 6.7: Processing speed

FPS	MAE	MSE
25	0.0	0.0
12.5	0.0	0.0
8.3	0.0	0.0
5	0.0	0.0
2.5	0.0	0.0

Table 6.8: Optimal FPS

6.4 ArenaPeds

Method (ROI)	MAE	MSE
CSRNet	0.0	0.0
Baseline	0.0	0.0
Baseline + flow	0.0	0.0

Table 6.9: The ROI results for ArenaPeds

Method (LOI)	MAE	MSE
CSRNet + PWCNet	0.0	0.0
Baseline	0.0	0.0
Baseline + flow	0.0	0.0

Table 6.10: The LOI results for ArenaPeds

6.5 Flow estimation impact

Qualitative comparison

Method	FPS
CSRNet+PWC	0.0
CSRNet+PWC+optimized	0.0
Baseline	0.0
Baseline+flow	0.0
Baseline+flow+optimized	0.0

Table 6.11: Processing speed

FPS	MAE	MSE
25	0.0	0.0
12.5	0.0	0.0
8.3	0.0	0.0
5	0.0	0.0
2.5	0.0	0.0

Table 6.12: Optimal FPS

Chapter 7

Discussion

Bibliography

[Brox et al., 2014] Brox, T., Papenberg, N., and Weickert, J. (2014). High Accuracy Optical Flow Estimation based on warping - presentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3024(May):25–36.

[Bruhn et al., 2005] Bruhn, A., Weickert, J., and Schnörr, C. (2005). Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):1–21.

[Cao et al.,] Cao, L., Zhang, X., Ren, W., and Huang, K. Large scale crowd analysis based on convolutional neural network. 48(10):3016–3024.

[Chan and Vasconcelos, 2009] Chan, A. and Vasconcelos, N. (2009). Bayesian Poisson regression for crowd counting Cited by me. *Computer Vision, 2009 IEEE 12th International . . . .*

[Chan and Vasconcelos, 2008] Chan, A. B. and Vasconcelos, N. (2008). Modeling, clustering, and segmenting video with mixtures of dynamic textures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):909–926.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, I:886–893.

[Dollár et al., 2012] Dollár, P., Wojek, C., Schiele, B., and Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761.

[Dosovitskiy et al., 2015] Dosovitskiy, A., Fischery, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Smagt, P. V. D., Cremers, D., and Brox, T. (2015). FlowNet: Learning optical flow with convolutional networks. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:2758–2766.

[Fang et al., 2019] Fang, Y., Zhan, B., Cai, W., Gao, S., and Hu, B. (2019). Locality-constrained spatial transformer network for video crowd counting. *Proceedings - IEEE International Conference on Multimedia and Expo*, 2019-July:814–819.

[Horn and Schunck, 1981] Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203.

[Hui et al., 2018] Hui, T. W., Tang, X., and Loy, C. C. (2018). LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8981–8989.

[Idrees et al., 2013] Idrees, H., Saleemi, I., Seibert, C., and Shah, M. (2013). Multi-source multi-scale counting in extremely dense crowd images. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2547–2554.

[Ilg et al., 2016] Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. (2016). FlowNet 2.0: Evolution of optical flow estimation with deep networks.

[Janai et al., 2018] Janai, J., Güney, F., Ranjan, A., Black, M., and Geiger, A. (2018). Unsupervised Learning of Multi-Frame Optical Flow with Occlusions. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11220 LNCS:713–731.

[Li et al., 2018] Li, Y., Zhang, X., and Chen, D. (2018). Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1091–1100.

[Lin and Davis, 2010] Lin, Z. and Davis, L. S. (2010). Shape-based human detection and segmentation via hierarchical part-template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):604–618.

[Liu et al., 2008] Liu, C., Freeman, W. T., Adelson, E. H., and Weiss, Y. (2008). Human-assisted motion annotation. *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

[Liu et al., 2019a] Liu, P., King, I., Lyu, M. R., and Xu, J. (2019a). DDFlow: Learning optical flow with unlabeled data distillation.

[Liu et al.,] Liu, P., Lyu, M., King, I., and Xu, J. SelFlow: Self-supervised learning of optical flow.

[Liu et al., 2019b] Liu, W., Salzmann, M., and Fua, P. (2019b). Context-aware crowd counting. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:5094–5103.

[Ma and Chan,] Ma, Z. and Chan, A. B. Counting people crossing a line using integer programming and local features. 26(10):1955–1969.

[Mémin and Pérez, 1998] Mémin, E. and Pérez, P. (1998). Dense estimation and object-based segmentation of the optical flow with robust techniques. *IEEE Transactions on Image Processing*, 7(5):703–719.

[Pock et al., 2008] Pock, T., Schoenemann, T., Graber, G., and Bischof, H. (2008). Learning optical flow. 5304(May 2014).

[Ranjan and Black, 2017] Ranjan, A. and Black, M. J. (2017). Optical flow estimation using a spatial pyramid network. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:2720–2729.

[Schroder et al., 2019] Schroder, G., Senst, T., Bochinski, E., and Sikora, T. (2019). Optical Flow Dataset and Benchmark for Visual Crowd Analysis. *Proceedings of AVSS 2018 - 2018 15th IEEE International Conference on Advanced Video and Signal-Based Surveillance*.

[Subburaman et al., 2012] Subburaman, V. B., Descamps, A., and Carincotte, C. (2012). Counting people in the crowd using a generic head detector. *Proceedings - 2012 IEEE 9th International Conference on Advanced Video and Signal-Based Surveillance, AVSS 2012*, pages 470–475.

[Sun et al.,] Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. PWC-net: CNNs for optical flow using pyramid, warping, and cost volume.

[Wan and Chan, 2019] Wan, J. and Chan, A. (2019). Adaptive density map generation for crowd counting. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:1130–1139.

[Wang et al., 2020] Wang, Q., Gao, J., Lin, W., and Li, X. (2020). Nwpu-crowd: A large-scale benchmark for crowd counting. *arXiv preprint arXiv:2001.03360*.

[Wedel et al., 2009] Wedel, A., Cremers, D., Pock, T., and Bischof, H. (2009). Structure- and motion-adaptive regularization for high accuracy optic flow. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1663–1668.

[Wu and Nevatia, 2007] Wu, B. and Nevatia, R. (2007). Detection and tracking of multiple, partially occluded humans by Bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266.

[Yu et al., 2016] Yu, J. J., Harley, A. W., and Derpanis, K. G. (2016). Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9915 LNCS:3–10.

[Zhang et al., 2016] Zhang, Y., Zhou, D., Chen, S., Gao, S., and Ma, Y. (2016). Single-image crowd counting via multi-column convolutional neural network. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:589–597.

[Zhao et al., 2016] Zhao, Z., Li, H., Zhao, R., and Wang, X. (2016). Crossing-line crowd counting with two-phase deep neural networks. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, volume 9912, pages 712–726. Springer International Publishing.

[Zheng et al., 2019] Zheng, H., Lin, Z., Cen, J., Wu, Z., and Zhao, Y. (2019). Cross-line pedestrian counting based on spatially-consistent two-stage local crowd density estimation and accumulation. 29(3):787–799.

Appendix A

Appendix