Fakultät für Mathematik, Informatik und Naturwissenschaften
Lehrstuhl für Informatik VIII (Computergraphik, Computer Vision und Multimedia)
Computer Vision
Prof. Dr. Bastian Leibe

Seminar Report

Feature mining for localized crowd counting

István Sárándi
Matriculation Number: 328707

June 2013

Advisor: **Wolfgang Mehner**

Abstract

In this seminar report I analyze methods for counting people in images by means of computer vision and machine learning algorithms. The two main ways to solve this task are by detection of each individual person and by regression based on low-level image features. My focus is the feature-based regression approach with special attention to locality introduced by grid subdivision. I implement a system based on a recent paper by Chen et al., using foreground mask-based, edge-based and texture-based features to train multivariate ridge regression and multivariate kernel ridge regression models. Overall, my system achieves 26% lower mean absolute error than the one presented in said paper on the same dataset.

Contents

1 Introduction

Video surveillance systems are becoming ubiquitous (see Figure 1 for example CCTV images) and there is a considerably large demand for algorithms that can process their data automatically. These systems are used for a variety of applications, including the detection of suspicious behavior for security reasons, avoiding dangerous crowd dynamics in enclosed spaces and improving advertisement placement[CLG+12]. Traditionally, these tasks were solved by human labor, by having employees watch the video stream of surveillance cameras. With the advance of computer vision algorithms, it has become possible to automatize several aspects of these tasks and thus to reduce costs.

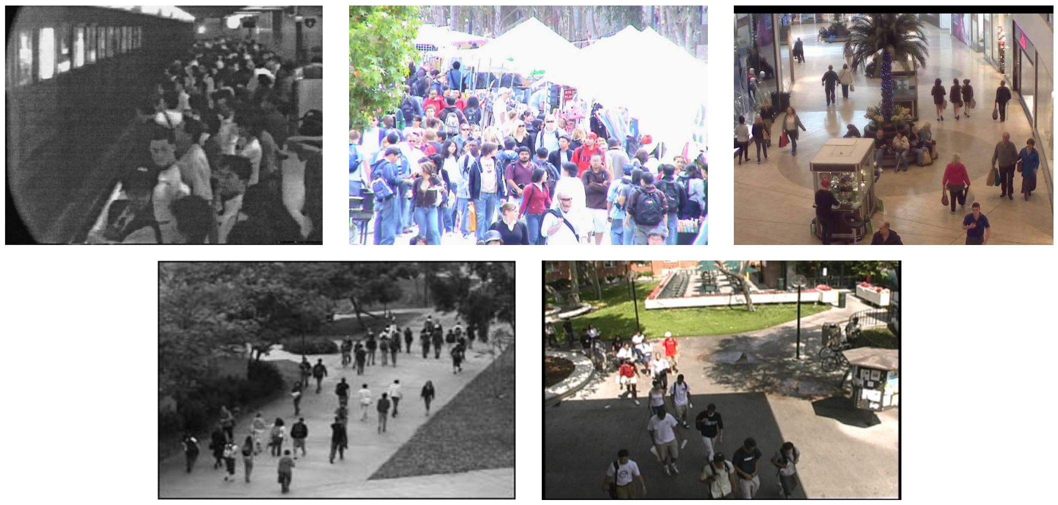

Figure 1: Examples of different scenes under video surveillance. Sources: [CCL99], [RB06], [CLG+12], [CLV08], [ZN03]

The topic of this seminar report is one of the basic questions one may ask when monitoring public spaces: How many people are there in the scene? This estimation of the number of people is commonly referred to as the crowd counting or the people counting problem. This topic is currently of great interest in the computer vision community and several different problem formulations and solution have been proposed for it. There are some other, closely related questions, for example, analyzing a video stream to count the number of people crossing a given line in the image[BIY+10]. This report, however, concentrates on counting the people in each individual frame (but not necessarily independently of each other).
Some applications have special requirements for locality or directionality, i.e. separately counting the people that are in different parts of the image or moving in different directions. In some cases special care must be taken to ensure privacy. Depending on the requirements, different approaches have been invented.
The two main types of solutions in the literature are detection-based and regression-based methods.
Detection-based methods find each person one-by-one and are typically more computationally complex. Additionally, according to some authors[CLV08], these methods are a concern for preserving privacy. Regression-based methods on the other hand are claimed to be privacy preserving. Furthermore, image processing and machine learning can be decoupled in the standard regression framework, so that the use of established machine learning algorithms independent of comuter vision becomes possible.
I researched the scientific literature about existing approaches and, in particular, examined the solution proposed by Chen et al. in their publication *Feature mining for localised crowd counting*[CLG+12]. An original contribution of this report is the use of multivariate kernel ridge regression for localized crowd counting, which improves the accuracy of the estimation.
The rest of the report is structured as follows. In Section 2, I review the approaches described in a wide range of publications. Section 3 contains the detailed explanation of the method that I implemented, based on the one used by Chen et al.[CLG+12]. In Section 4, I evaluate my implementation and analyze the results. Finally, Section 5 concludes the report by summarizing and discussing the presented solution.

2 Related work

Recently, there has been a great research interest in the crowd counting problem in the computer vision community. First, I examine the different requirements that may be posed on the solution in order to get a better understanding of the task. Then I review the algorithmic solutions that have been proposed.

2.1 Requirements

Choosing the right algorithm to solve the crowd counting problem depends on the requirements of the particular application. In the following, I describe the main aspects to consider.
The most important distinctions are between local and global estimation, quantitative and qualitative estimation and directionality-aware and non-directionality-aware estimation. Further requirements can be robustness against certain changes, constraints on computational complexity and preserving privacy.

2.1.1 Local vs. global

Local methods estimate the crowd size for different parts of the image, whereas global methods only estimate the overall crowd size for the whole image. Depending on the application, local or global counting may be more suitable. In a shopping mall, for example, one is not only interested in the overall people count in the scene, but also wants to discriminate between different locations of the scene[CLG$^+$12], thus for example measuring the popularity of different shops. Local counting is most easily formalized using a cell-based approach, in which the image is subdivided into a grid of equally sized rectangular cells and the task is to estimate the number of people in each cell. Since people may extend over multiple cells, a reference point has to be chosen on each person and the task is then refined to the estimation of reference points in each cell (see Figure 2). Usually the head is taken as a reference point, due to its good visibility on CCTV images and its prominent shape. A disadvantage of this formulation is that areas of interest (e.g. shops) and cells are not in a one-to-one correspondence in the image. One solution to this problem could be to create custom shaped regions in the image instead of the rectangular cell grid. Another way is to reduce the cell size, in the extreme case even to the one-cell-per-pixel scale as done in [LZ10]. With such estimation, counting is possible for any custom shaped area by summing up the counts in the cells/pixels contained in the area.

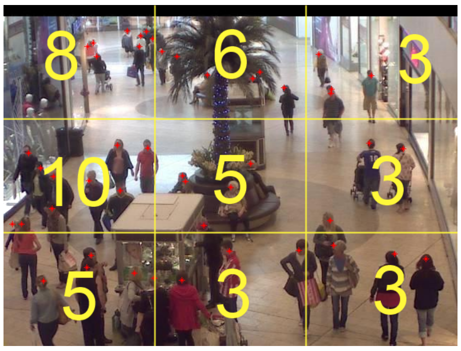

Figure 2: Locality can be introduced by grid cell-based subdivision. The number of reference points (heads) is displayed for each cell.

2.1.2 Directionality

There are applications in which not only the number of people is interesting, but also their movements. A basic piece of information about the movement of people is the direction in which they move. Counting the people going in different directions separately may be a requirement for the application or just a way to improve the overall performance of a solution. Such an application could be keeping track of the number of people in an enclosed space (e.g. a disco) by counting the entering and exiting people. Chan et al. use

a mixture of dynamic textures to segment the crowd into two groups[CLV08], one coming towards and one going away from the camera (see Figure 3). Since people look different when coming and going, the authors could make their system specifically learn the characteristics of each group, possibly increasing the performance of their solution. This segmentation based approach can introduce directionality to any counting method, by separately counting the people in each segment.
It must be noted, however, that movement-based approaches work only if the video frame rate is sufficiently high.

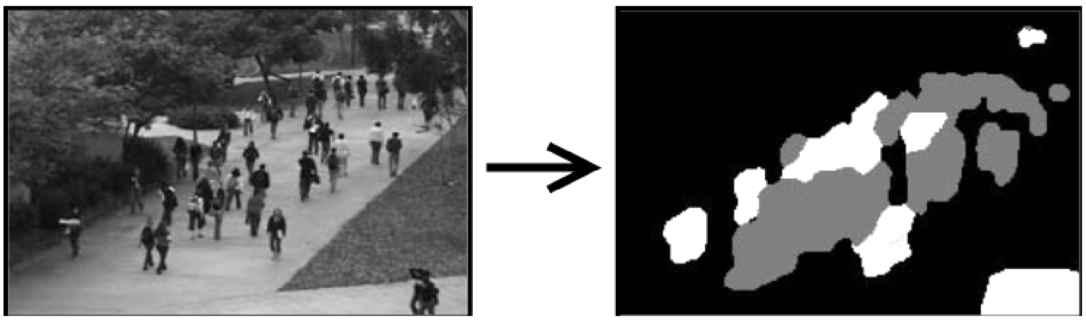

Figure 3: Crowd segmentation based on motion[CLV08].

2.1.3 Quantitative vs. qualitative

On a subway platform, a qualitative degree of crowdedness may be sufficient instead of the actual quantitative crowd size in order to ensure safety of the passengers. This qualitative crowdedness may be defined as the ratio of the current people count and the maximum possible people count in that area. The authors of [CCL99] use a global crowdedness formulation although a localized version is equally possible. Depending on the applied method, estimating the qualtitative crowdedness may be easier than estimating the count itself. Ma et al. use 5 discrete crowdedness classes and classify each image frame according to one of the classes[MHL10].

2.1.4 Robustness

It may be required that the estimation remain robust against changes in lighting conditions, movements of the camera or even to a full location change. In some applications (e.g. subway platform monitoring) lighting conditions and camera position remain approximately constant. In outdoor environments, day and night cause strong changes in lighting, which must be taken into consideration when choosing or customizing an algorithm.
It is also important to assess the amount of manual and computational work needed if the camera position has to be changed.

2.1.5 Computational complexity

An important aspect to consider when developing any algorithm is the computational complexity. Since real-time performance is crucial in security applications, the complexity of the algorithm is of high importance in many crowd counting scenarios. In other applications (e.g. analyzing people behavior in shopping malls to improve the location of advertisements) the real-time requirement may not apply. It is also worth noting that a method that is unsuitable for real-time applications today may be suitable in the future.

2.1.6 Privacy concerns

There is also a non-technical aspect to be considered, namely the protection of personal data, in other words, preserving the privacy of individuals. Several articles have been published about the public acceptance and legal regulations of CCTV surveillance and the issues of algorithmic analysis of these video streams. Davies and Velastin note[DV05] that in most countries, CCTV is not seen as an intrusion into the private sphere of the individual. However, Ferenbok et al. state[FCD$^+$11] that the transition towards intelligent processing is happening largely out of the public eye, implying that it is hard to determine what society would still accept as non-intrusive processing, since they are not even aware of the question. Oram quotes[Ora11] the guidance of the United Kingdom's Information Commissioner's Office stating that regarding privacy, "The simple rule of thumb is that you need to decide whether the image you have taken is aimed at learning about a particular person's activities". Applying this guideline, most crowd counting algorithms seem to be on the safe side, since their aim is to analyze the properties of crowds and not individual people. According to Chan et al.[CLV08], the algorithmic methodology also plays a role in determining whether a particular application violates privacy or not. I am not convinced by this reasoning, seeing that most laws and legal guidelines (similarly to the one mentioned above) are formulated in a general way that focuses on the purpose of the processing and mostly do not care about technical details of the method. If the system's overall task is to count people, I cannot to see the relevance of the intermediate steps (provided that the extraced data is not used for other purposes).

2.2 Algorithmic solutions

Most of the solutions proposed for the crowd counting problem can be categorized as either detection-based algorithms or feature-based regression algorithms. Detection-based algorithms can be further subdivided as static (based on individual images) and dynamic (based on motion).

2.2.1 Static detection-based methods

The most straightforward way to count people is to find them one-by-one in each individual the frame. One such approach is presented by Leibe et al.[LSS05], where video is used only in the training phase to avoid manual segmentation.

- *Locality:* Fulfilling the locality requirement is trivial with these methods. Since each pedestrian is localized in the image, counting them in any custom area is straightforward.
- *Robustness:* Robustness against lighting conditions depends on the robustness of the detector. Changing the camera position generally does not cause any problems, as a properly trained pedestrian detector will still detect the people.
- *Computational complexity:* Detection-based methods often involve exhaustive search over the image (e.g. with a multi-scale sliding window) and the classification of the respective image parts as either person or non-person. This exhaustive search may be too computationally expensive[CLG$^+$12], depending on the requirements.
- *Privacy concerns:* Chan et al. claim in [CLV08] that the detection step may be considered by some people as an "attempt to 'single out' the individual", and suggest to use only low-level features of the image to preserve privacy.

Another concern about detection-based methods is that they may perform poorly if large parts of the persons are occluded, which is common in crowd counting scenarios. This problem was addressed by Lin et al. by using a head detector instead of a full-body pedestrian detector[LCC01], since the head is usually not occluded due to the positioning of CCTV cameras well above head-height. The method described by Leibe et al. is claimed to be robust against occlusion by integrating local and global cues[LSS05].

2.2.2 Dynamic detection-based methods (trajectory clustering)

The second main type of approaches proposed in the literature for counting the number of people is based on keeping track of points moving together in the image sequence. Coherent motion of points suggests that they belong to the same person[BC06], and thus clustering these trajectories and counting the clusters can solve the crowd counting problem. Motion information can be extracted from the video by tracking keypoints (detected by methods such as the Harris corner detection algorithm[HS88]) and optical flow[KBN11][FHTN12] or a combination. These methods cannot be applied if the video frame rate is low.

- *Locality:* Since trajectories are inherently location-based, locality is trivially introduced in these approaches.
- *Robustness:* New camera positions do not pose a big problem, as trajectory clustering can be applied even within an unsupervised framework (as in [BC06]), meaning that the manual labor associated with a new setup is minimal.

2.2.3 Feature-based regression

The third main category of approaches is based on the standard regression framework. The image is first mapped to a list of real numbers (feature vector) by feature extraction, and a regression function is learned that maps from the feature vector to the target variable. The target variable may either be a qualitative density category ([MHL10] uses 5 levels) or directly the number of people. Approaches of this category differ in the choice of features and the learning algorithm for optimizing the regression function. The features might include motion features[KGT05] or only static information.
Since regression algorithms are supervised, ground-truth target values must be created manually.

- *Locality:* Locality can be introduced to regression-based methods by subdividing the image to multiple cells and learning a multi-output regression function where each output corresponds to the number of people in one cell. Approaches differ in whether they use information from other cells to predict the number of people in a given cell[CLG$^+$12] or only the features in the given cell are used[WLLX06]. The former way exploits information sharing between cells.
- *Robustness:* Robustness against lighting changes depends on the extracted features. Camera relocation may mean that new training data has to be acquired with new manual ground truth. In [KGT05], the authors use viewpoint invariant training to combat this problem.
- *Computational complexity:* Complexity can range from the very efficient (e.g. simple features and linear regression) to the very expensive (complex features and neural networks with several hidden layers).
- *Privacy concerns:* According to Chan et al.[CLV08], regression based on low-level features is preferable in privacy-sensitive cases, as they do not locate or track individuals.

3 Methodology

In this section I review and expand upon the approach presented in [CLG$^+$12], the paper of focus for this seminar topic. It is a feature-based regression method with rectangular grid-based locality estimating the people count in each grid cell. It makes use of feature mining and information sharing between the grid cells. Feature mining means that each feature can have different importance in different cells. Information sharing means that the features extracted from all the cells are used to predict the people count in each cell, not only the features that are extracted from that one cell.

3.1 Overview

There are two main steps to this method. First, a feature vector has to be assembled for each frame, and second, some learning algorithm must be applied to learn a regression from this feature vector to the multi-dimensional output (people count in each grid cell).Feature extraction starts with foreground segmentation. Then three types of features are extracted from each grid cell independently: foreground mask-based features (e.g. foreground area), edge-based features (e.g. edge orientation histogram) and texture-based features (using gray-level co-occurrence matrices). In order to compensate for perspective distortion, pixels are weighted according to scale when calculating the features. After extraction, the features of each grid cell are concatenated to create one long feature vector describing the whole frame. In the second main step, a regression algorithm is used to learn the functional dependency between image features and people counts in each cell. Chen et al. chose the multivariate ridge regression learning algorithm, as described in [Hai87]. I will also apply multivariate kernel ridge regression. For an illustration of the procedure, see Figure 4.

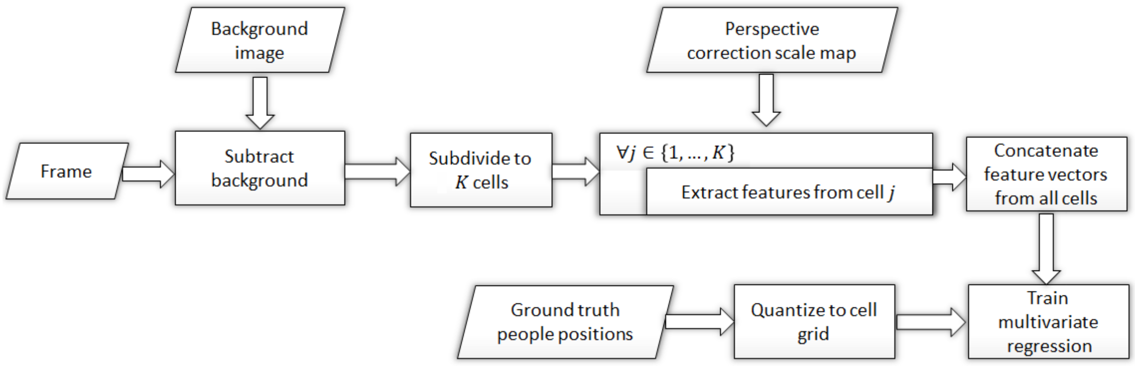
Background image
Perspective correction scale map
Frame
Subtract background
Subdivide to K cells
∀j ∈ {1, ..., K}
Extract features from cell j
Concatenate feature vectors from all cells
Ground truth people positions
Quantize to cell grid
Train multivariate regression

Figure 4: The overall procedure of the implemented people counting system.

3.2 Feature extraction

Following Chen et al. and Chan et al., I extract three types of features from each grid cell (29 features in total for each cell):

- *Foreground mask-based features:* foreground area, number of pixels on the foreground perimeter, perimeter-area ratio, perimeter orientation histogram. (9 features)
- *Edge-based features:* number of edge pixels, edge orientation histogram, Minkowski dimension (8 features)
- *Texture-based features:* homogeneity, energy and entropy, calculated from gray-level co-occurrence matrices (GLCM) (12 features)

3.2.1 Foreground mask-based features

Foreground mask-based features require binary segmentation of the frame to background and foreground. Chen et al. do not elaborate on the details of this step. Chan et al. perform segmentation based on motion with dynamic motion textures (see Figure 3). I found that if the lighting changes are negligible, a single photo of the scene without any people is sufficient as reference. Then the thresholded absolute difference of the empty scene and a given frame can be used as the foreground mask. To avoid detection of bright sunny spots as foreground, very bright areas are excluded from the mask. In this formulation the foreground mask M is generated in the following way, using the current frame I, the image of the empty scene BG and threshold parameters t_1 and t_2.

$$M(x,y) = \begin{cases} 1, & \text{if } |I(x,y) - BG(x,y)| > t_1 \text{ and } I(x,y) < t_2 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

A more robust method is to build a background model of the image and label parts of the image as foreground that are improbable to be explained by the background model. Such an approach based on Gaussian mixture models (GMM) for each pixel is described in [Ziv04]. It is an adaptive method in the sense that the background model is constantly updated with each new frame and recent frames are more important than older ones.
In order to remove noise, masks are post-processed by a closing and an opening morphological operation.
Comparing the two methods (see Figure 5), it can be noted that the first and simpler method is sufficient and also gives more visually appealing results than the more complex one, at least for the given scene. Unfortunately, there is no frame in the mall dataset that contains no people. Hence, I had to assemble an empty scene by putting together different empty parts of multiple frames. There is one person, who stays at the same place during the whole time (more than an hour), he was removed by manual image manipulation to create the empty image.
The features extracted from the foreground mask are:

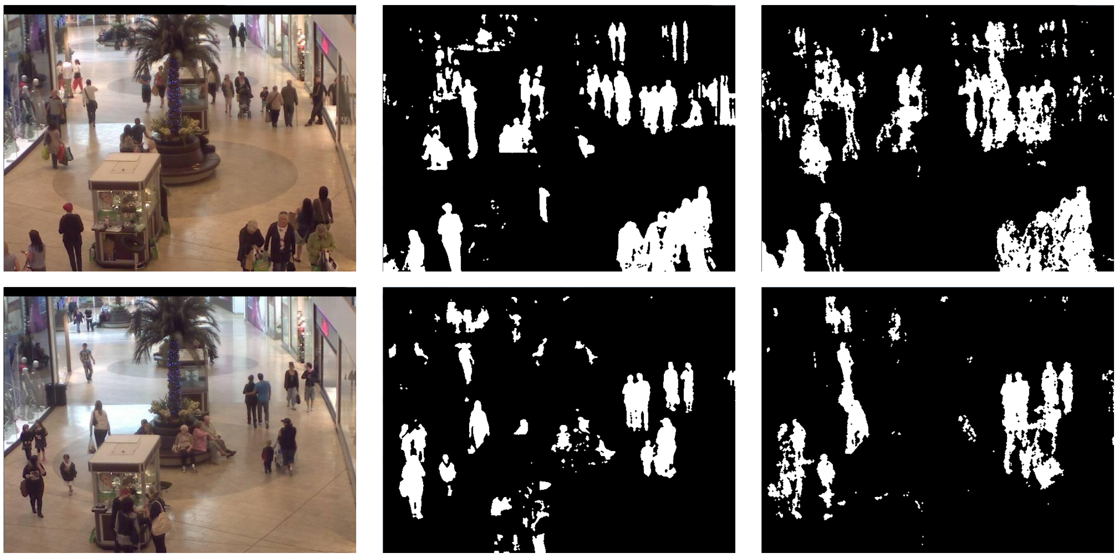
Figure 5: Results of foreground segmentation for frame #11 (*first row*) and frame #1811 (*second row*) of the mall dataset. *Left column:* original; *Middle column:* difference to a single background image (empty scene) using $t_1 = 40, t_2 = 200$; *Right column:* Mixture of Gaussians background modeling, using history length 10 and variance threshold 15.0. All masks have been post-processed by a closing and an opening operation (self-generated).

- *Foreground area:* The number of pixels in the foreground mask, obtained by background segmentation. The intuition is that, roughly speaking, the larger the foreground, the more people there are in the scene.
- *Perimeter length:* The number of pixels on the perimeter of the foreground mask. The perimeter image P can be obtained using morphological operations on the foreground mask M (see result in Figure 6):
$$P = M - (M \ominus S_{3\times3}) \tag{2}$$
where $\ominus$ denotes erosion and $S_{3\times3}$ is a 3x3 square structuring element.
- *Perimeter-area ratio:* The ratio between perimeter length and foreground area. If this number is large, it can be an indication that there are more people in the scene, since the head and limbs of people create bumpy structures in the foreground. The inclusion of this feature may seem redundant but it is still needed if the learning algorithm used in the next step is incapable of capturing non-linear relationships.
- *Perimeter orientation histogram:* The perimeter orientation histogram is obtained by first quantizing the orientation at each perimeter point to 6 bins and then counting the number of occurrences of each bin. Opposite directions are considered the same. The orientation at a given pixel can be calculated by finding the maximum response to a set of oriented ellipse-shaped Gaussian filters as in [CLV08]. An alternative choice is to determine and quantize the gradient direction by using the Sobel filter (discrete version of the derivative of Gaussian). More specifically I first apply the Sobel filter on the image in both x and y direction to obtain directional derivatives, and second, I use the $arctan$ function to find the gradient angle.

3.2.2 Edge-based features

The edge patterns inside the foreground regions give us additional cues about the number of people contained within the foreground blobs. In case of strong occlusions, segmentation-based features alone may not be able to accurately differentiate between one larger or two smaller occluding persons.

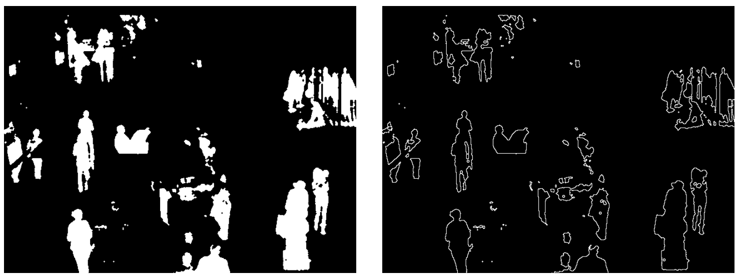
Figure 6: *Left:* foreground mask, *right:* perimeter extracted from the foreground mask.

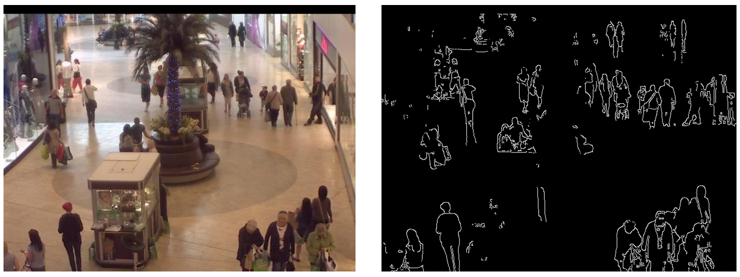
Figure 7: The edges of the foreground, as detected by the Canny algorithm.

Edges are extracted using the well-known hysteresis approach introduced by Canny[Can86], resulting in a binary image, in which edges are 1 pixel thick. The edge map is then masked to the foreground region in order to avoid irrelevant background edges (see Figure 7). Using this masked edge image, the following features are extracted:

- *Edge pixel count:* The number of edge pixels in the edge map. More edges may indicate a cluttered region with more people
- *Edge orientation histogram:* Analogous to the perimeter orientation histogram described above.
- *Minkowski fractal dimension:* The Minkowski fractal dimension is used to estimate the degree of space filling of a curve, in this case the foreground edges. As described in [MdFCLV99], the Minkowski dimension can be approximated using the following algorithm. The morphological operation of dilation is applied to the image with successively larger and larger disk-shaped structuring elements, so that the edges become thicker (becoming so called Minkowski sausages). For each of these dilated images we compute the size of the filled area. This area will obviously increase with growing disk size until it reaches the area of the whole image. The rate of growth of the filled area A for growing disk diameter d is used to estimate the Minkowski dimension. First we apply log-log transformation to the samples:

  $$(d,A) \mapsto (\log(d), \log(A)) \tag{3}$$

  Then least-squares linear regression is used to obtain the slope of this log-log plot resulting in the approximation

  $$\begin{aligned} \log(A) &= w_1 \cdot \log(d) + w_0 \\ A &= \exp(w_0) \cdot d^{w_1} \end{aligned} \tag{4}$$

  Finally, the Minkowski dimension m is approximated as

  $$m = 2 - w_1. \tag{5}$$

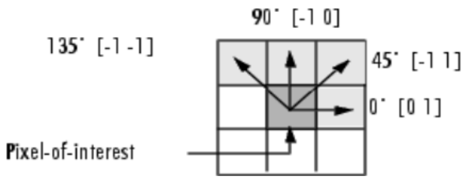

Figure 8: Principle of the gray-level co-occurrence matrix with $d = 1$ and $\theta \in \{0°, 45°, 90°, 135°\}$ (source: [JD97]).

3.2.3 Texture-based features

As examined by Marana et al. in [MCLV98], texture features can be useful to estimate crowd density. Both Chan et al. and Chen et al. use texture features based on gray-level co-occurrence matrices (GLCM) to estimate the number of people[CLV08][CLG+12]. Given a distance d and an angle θ, the gray-level co-occurrence matrix of an image I (quantized to q gray levels) is defined as

$$G(d,\theta) \in \mathbb{N}^{q\times q} \tag{6}$$

$$G(d,\theta)_{ij} = \left| \left\{ \big((x_1,y_1),(x_2,y_2)\big) \;\middle|\; dist\big((x_1,y_1),(x_2,y_2)\big) = d \text{ and } \operatorname{atan2}(y_2-y_1, x_2-x_1) = \theta \right.\right. \\ \left.\left. \text{and } I(x_1,y_1) = i \text{ and } I(x_2,y_2) = j \right\} \right| \tag{7}$$

Informally, it describes how many times two pixels that are at the given distance and angle to each other have the gray level i and j, respectively. The GLCM is computed for $d = 1$ and $\theta \in \{0°, 45°, 90°, 135°\}$ (see Figure 8), using only the foreground region, i.e. I only take those pairs into consideration that are both in the foreground. The GLCM is then normalized, i.e. it is divided element-wise by the number of pixel-pairs that were used to compute it.
Haralick et al.[HSD73] define 14 features that can be extracted from the normalized GLCM (N). Following Chan et al., I use the following three of them.

- *Homogeneity:* $H = \sum_{i,j} \frac{N_{ij}}{1+(i-j)^2}$
- *Energy:* $E = \sum_{i,j} N_{ij}^2$
- *Entropy:* $S = -\sum_{i,j} N_{ij} \log N_{ij}$

3.2.4 Perspective compensation

People at a larger distance appear smaller, which should be taken into consideration when calculating the features. [KGT05], [CLG+12] and [CLV08] all use the same principle to combat perspective distortion. Although the presentation in the respective papers is different, the method can be summarized as follows. First, an approximate relative distance map $d(x,y)$ or scale map $s(x,y) = \frac{1}{d(x,y)}$ is calculated for the image, and second, this distance is later used to weight the contribution of individual pixels to the features. Kong et al. estimate a homography between the ground plane and it's projection in the image[KGT05], but as it turns out, the linear approximation method explained below is also very accurate while being considerably simpler.
Below, I provide the derivation of the functional dependency between the vertical position in the image and the camera-to-scene distance. Assume a pinhole camera with no tilt (i.e. that vertical poles in the scene are mapped to vertical lines in the image). Consider Figure 9. P is the point in the image plane that we are interested in. The corresponding point in the scene is Q. Thus we need to calculate the distance d knowing h, y and α. Using the law of cosines in triangle BCP for the angle at B:

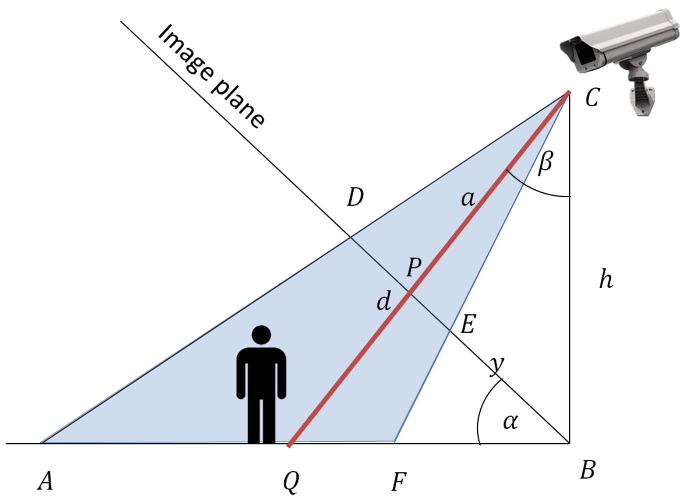

Figure 9: Sketch for calculating the distance at a given image point. $h = \overline{BC}$, $a = \overline{CP}$, $d = \overline{CQ}$, $y = \overline{BP}$

$$\begin{aligned} a^2 &= y^2 + h^2 - 2yh\cos\left(\frac{\pi}{2} - \alpha\right) \\ a &= \sqrt{y^2 + h^2 - 2yh\sin\alpha} \end{aligned} \tag{8}$$

The law of cosines for the angle at C yields

$$y^2 = a^2 + h^2 - 2ah\cos\beta \tag{9}$$

By definition of the cosine, $\cos\beta = \frac{h}{d}$. Substituting this into Equation 9, we can obtain d as follow:

$$\begin{aligned} y^2 &= a^2 + h^2 - 2ah\frac{h}{d} \\ d &= \frac{2ah^2}{a^2 + h^2 - y^2} \\ &= \frac{2h^2\sqrt{y^2 + h^2 - 2yh\sin\alpha}}{y^2 + h^2 - 2yh\sin\alpha + h^2 - y^2} \\ &= \frac{2h\sqrt{y^2 + h^2 - 2yh\sin\alpha}}{2h - 2y\sin\alpha} \\ &= \frac{h\sqrt{y^2 + h^2 - 2yh\sin\alpha}}{h - y\sin\alpha} \end{aligned} \tag{10}$$

The interesting range of y (those values that correspond to points within the image boundaries) can be determined by specifying the camera height h and the angles $\measuredangle BCE$ and $\measuredangle BCD$. Then in triangle BCE, one side (h) and the two angles adjacent to this side are known. From that, any other side can be calculated. Similarly for BCD. It can be shown that

$$\begin{aligned} \overline{BE} &= \frac{h\tan\measuredangle BCE}{\cos\alpha + \sin\alpha\tan\measuredangle BCE} \\ \overline{BD} &= \frac{h\tan\measuredangle BCD}{\cos\alpha + \sin\alpha\tan\measuredangle BCD} \end{aligned} \tag{11}$$

The parameters $h = 5$, $\alpha = 60°$, $\measuredangle BCE = 40°$ and $\measuredangle BCD = 80°$ seem reasonable. For these values $\overline{BE} \approx 3.42$ and $\overline{BD} \approx 5.24$. With these parameter settings, Figure 11 shows how d changes with y according to

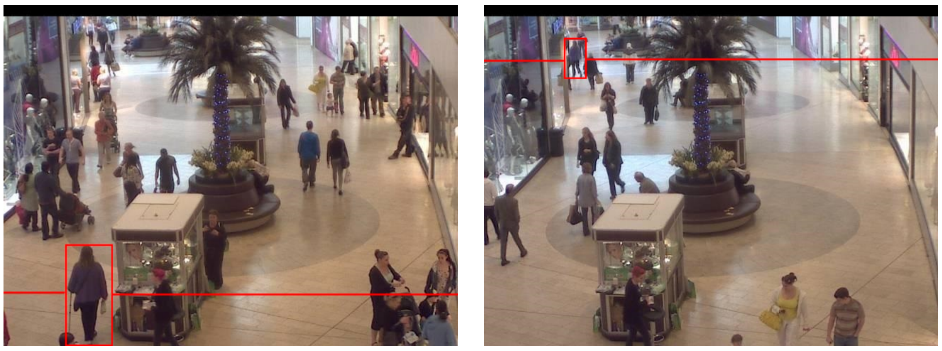

Figure 10: Inferring the perspective scale map. The height of the same person is measured when she is near and far from the camera. The scale function is then a linear interpolation using these values.

Equation 10. The reciprocal of the distance function ($s = 1/d$), despite being non-linear on the whole, can be well approximated by a linear function in the interesting range (with a coefficient of determination $R^2 = 1.000$). Thus, I conclude that the linear approximation used by Chen et al. is theoretically well-founded, and I will evaluate how it empirically affects the quality of the estimation.

As suggested by Chan et al., the approximate scale map can be acquired for a concrete application by measuring the same person's height when she is at the bottom of the image and again when she is at the top of the image. The values in between are linearly interpolated to get a scale map $s(x,y)$ (see Figure 10).

All features, except for the Minkowski fractal dimension and the texture-based features, can then easily be corrected by weighting each pixel's contribution with the inverse of the linearly interpolated scale map. For example the perimeter length feature p is then not the actual number of pixels of the perimeter image P, but a weighted version of it:

$$p = \sum_{(x,y):\ P(x,y)=1} \frac{1}{s(x,y)} \tag{12}$$

Area-based features grow quadratically with scale (e.g. the foreground area f of the foreground mask M) and have to be weighted quadratically:

$$f = \sum_{(x,y):\ M(x,y)=1} \frac{1}{s^2(x,y)} \tag{13}$$

Histogram features can be weighted by accumulating the $\frac{1}{s(x,y)}$ weights in each bin instead of simply counting the pixels.

3.3 Regression

The second step of feature-based regression methods is to learn the functional relationship between image features and ground truth people counts using a regression algorithm. Each of the N frames are subdivided to K cells. As in the paper by Chen et al., frame #i is represented by a long vector $\mathbf{x}_i$, composed by concatenating the feature vectors $\mathbf{z}_i^j$ of each cell $1 \leq j \leq K$ in the frame. Let the full length of these concatenated feature vectors be D.

The people counts u_i^j (frame i, cell j) are put together to get the desired vectorial output $\mathbf{y}_i \in \mathbb{N}^K$ of the system, containing the people counts for each cell of the frame.

Vectors are considered to be row vectors throughout the whole report.

$$\begin{aligned} \mathbf{x}_i &= \left[\mathbf{z}_i^1, \mathbf{z}_i^2, \ldots, \mathbf{z}_i^K\right] \in \mathbb{R}^D \\ \mathbf{y}_i &= \left[u_i^1, u_i^2, \ldots, u_i^K\right] \in \mathbb{N}^K \end{aligned} \tag{14}$$

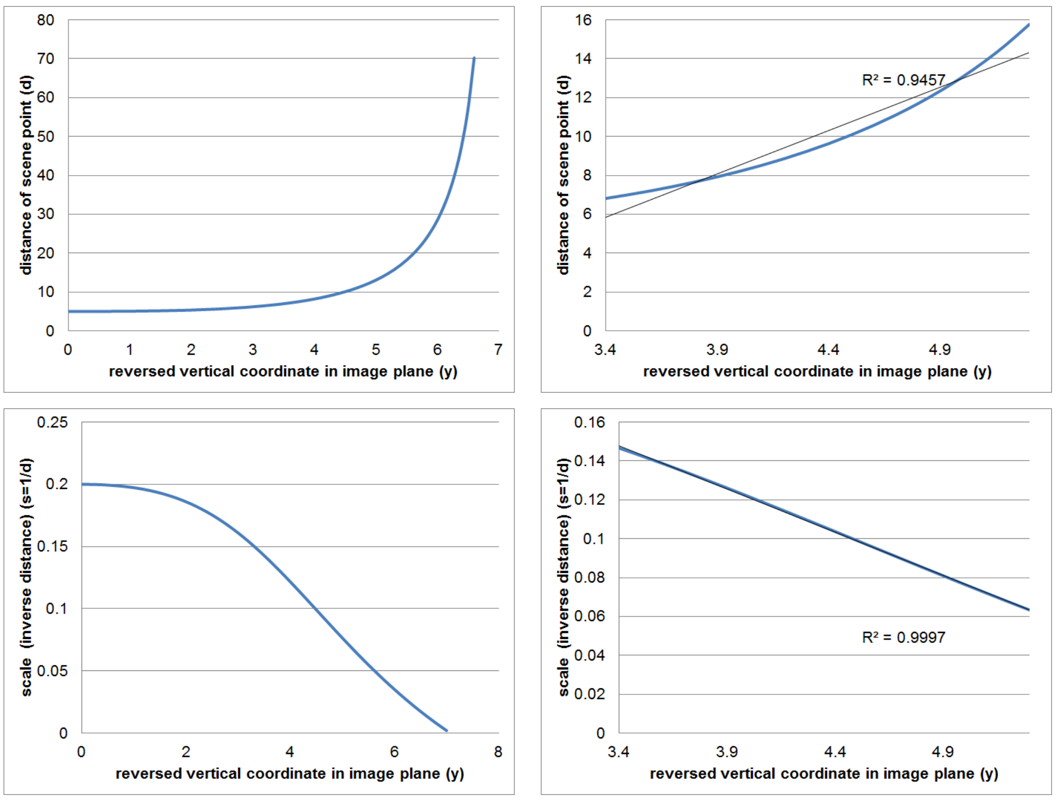

Figure 11: *Top left:* Plot of the exact distance of the scene point (d), when varying the vertical position on the image plane (y). *Top right:* The same plot, showing only the range for y within the image boundaries. *Bottom row:* plots of the reciprocal function $s = 1/d$

3.3.1 Multivariate ridge regression

Ridge regression is the name of least-squares linear regression with L2-regularization. In multivariate ridge regression the target variable is a vector and not a scalar. Since this is a linear model, we are looking for a weight matrix W and a bias vector $\mathbf{b}$, such that the function

$$\hat{\mathbf{y}}(\mathbf{x}) = \mathbf{x}W + \mathbf{b} \tag{15}$$

is a good approximation for the training set. In ridge regression, the following criterion is minimized

$$J(W, \mathbf{b}) = \frac{1}{2}||W||_F^2 + C\sum_{i=1}^{N}||\mathbf{y}_i - (\mathbf{x}_i W + \mathbf{b})||_F^2, \tag{16}$$

where $||\cdot||_F$ denotes the Frobenius norm (for which $||A||_F^2 = \sum_{i,j} A_{ij}^2$) and C controls the strength of regularization (the smaller C is, the stronger the regularization).

To make the notation more compact, let $\tilde{\mathbf{x}} = \begin{bmatrix}\mathbf{x} & 1\end{bmatrix}$ and $\tilde{W} = \begin{bmatrix} W \\ \mathbf{b} \end{bmatrix} \in \mathbb{R}^{(D+1)\times K}$. Let $\tilde{X} \in \mathbb{R}^{N\times D}$ denote the matrix obtained by stacking the row vectors $\tilde{\mathbf{x}}_i$ on each other for all training samples $1 \le i \le N$. Similarly let $Y \in \mathbb{Z}^{N\times K}$ denote the same for the $\mathbf{y}_i$ vectors. It can be shown[Hai87] that the optimal solution for $\tilde{W}$ is

$$\tilde{W} = \left(\tilde{X}^\top \tilde{X} + R\right)^{-1} \tilde{X}^\top Y, \tag{17}$$

where R is the diagonal matrix responsible for regularization of the weights. The bias components are usually not regularized, hence the last component is 0.

$$R = \frac{1}{2C}\operatorname{diag}\left(\begin{bmatrix}1 & 1 & \dots & 1 & 0\end{bmatrix}\right) \in \mathbb{R}^{(D+1)\times(D+1)} \tag{18}$$

The main claim of Chen et al. is that crowd counting can benefit from feature mining and information sharing between cells. The multivariate ridge regression has the capability to make use of both. Feature mining is the phenomenon that the same feature (i.e. foreground segment area) can have different importance in different cells. This is indeed possible with multivariate ridge regression, since each output variable has its own set of weights in the weight matrix. Information sharing becomes also possible as all output variables are calculated by a weighted sum of the features from all cells. By contrast, if a separate regression model were learned for each cell, this would not be possible.

3.3.2 Multivariate kernel ridge regression (Gaussian process regression)

Ridge regression, as presented above, can only fit a linear model to the data. This is a considerable restriction and it is worth examining if allowing the nonlinear models can improve the performance of people counting. One way to introduce nonlinear capabilities over a linear learning algorithm is to map the input vectors to a (typically higher dimensional) feature space and perform the linear fitting in that feature space. This results in the following optimization criterion in case of multivariate ridge regression:

$$J(W) = \frac{1}{2}||W||_F^2 + C\sum_{i=1}^{N}||\mathbf{y}_i - \phi(\mathbf{x}_i)W||_F^2, \tag{19}$$

where $\phi : \mathbb{R}^D \to \mathbb{R}^H$ is the mapping to the higher, H-dimensional feature space. Note that there is no bias term in this formulation. If needed, the $\phi(\cdot)$ mapping can emulate it by mapping to vectors that have a constant 1 component. The equation can be transformed by introducing

$$\begin{aligned} &A := -2C \cdot (\Phi W - Y), \text{ for which} \\ &W = \Phi^\top A \end{aligned} \tag{20}$$

The equivalent task is then to minimize

$$J_{\text{dual}}(A) = \frac{1}{2}\text{tr}\left(A^\top \Phi\Phi^\top A\right) + C\cdot \text{tr}\left(A^\top \Phi\Phi^\top\Phi\Phi^\top A - 2Y^\top \Phi\Phi^\top A + Y^\top Y\right), \tag{21}$$

where $\text{tr}(\cdot)$ denotes the trace of a matrix, i.e. the sum of the elements on the main diagonal.
In this formulation, all $\phi(\mathbf{x}_i)$ vectors appear only in dot products with other $\phi(\mathbf{x}_j)$ vectors. This allows for a shortcut, the so called kernel trick. The kernel function is defined as $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \phi(\mathbf{x}_j)$. It is possible to construct and calculate the kernel function, without explicitly defining or calculating the corresponding $\phi(\cdot)$ function, resulting in the possibility of using $H \gg D$, even $H = \infty$. Chan et al. used this kernel ridge regression algorithm in their global counting approach. I apply it now for the local approach defined by Chen et al.
In kernel ridge regression, the task becomes to minimize the so called dual criterion

$$J_{\text{dual}}(A) = \frac{1}{2}\text{tr}\left(A^\top KA\right) + C\cdot \text{tr}\left(A^\top KKA - 2Y^\top KA + Y^\top Y\right), \text{ where} \tag{22}$$

$$K \in \mathbb{R}^{N\times N} \text{ and } K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) \tag{23}$$

for which the optimal solution is

$$A^* = \left(K + \frac{1}{2C}I_{N\times N}\right)^{-1} Y. \tag{24}$$

Prediction for new test input vectors requires storing all training input vectors. The prediction is done using the following formula:

$$\hat{\mathbf{y}}(\mathbf{x}) = \mathbf{k}(\mathbf{x})A, \text{ where} \tag{25}$$

$$\mathbf{k}(\mathbf{x}) \in \mathbb{R}^N \text{ and } \mathbf{k}(\mathbf{x})_i = k(\mathbf{x}_i^{training}, \mathbf{x}). \tag{26}$$

For the sake of simplicity, I use the Gaussian radial basis function (also known as exponential kernel) as the kernel function.

$$k_{RBF}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \cdot ||\mathbf{x}_i - \mathbf{x}_j||^2\right) \tag{27}$$

This kernel has only one hyperparameter (γ), so it takes less time to perform search in the hyperparameter space than with the kernel of Chan et al. which has 4 hyperparameters. The hyperparameter $\gamma = \frac{1}{2\sigma}$ controls the variance (width) of the radial basis functions. In the Gaussian process interpretation of kernel ridge regression, this corresponds to enforcing more or less covariance between the outputs that belong to faraway input points. A small γ value results in a smoother approximation, since even inputs that are far away have highly correlated outputs.

3.3.3 Data normalization

Both the linear ridge regression and the kernel ridge regression algorithm treats all of the input dimensions equally. In linear ridge regression there is a single C value controlling the regularization strength, even though weights acting on different features would need different treatment (in Bayesian terms: we should have different priors on different weights). In kernel ridge regression, the RBF kernel also treats all dimensions the same way. This only makes sense if the dimensions have the same scale. The scale (range) of an input dimension can be formalized as the sample variance of that dimension's values in the training set. Therefore, I normalize the input data to zero mean and unit variance in each dimension. The mean and variance of each dimension is stored, and the same normalization is also applied to each input in the validation/test set (using the mean and variance values calculated from the training set). The subtraction of the mean has actually no effect in this case, since the bias is not regularized in the linear model, and in the kernel version the RBF kernel only uses differences.

3.4 Implementation

I implemented the solution described above using Microsoft's .NET 4.0 platform and the C# programming language. For image processing I use EmguCV, a wrapper library around OpenCV version 2.4.9. For linear algebra calculations the Math.NET Numerics library is used. I implemented all feature extraction myself.

4 Results

I evaluated my models using the mall dataset[Che13] provided by Chen et al. of the Queen Mary University of London. Some properties of this dataset:

- *Number of frames:* 2000
- *Resolution:* 320x240 [1]
- *Frames per second:* <2
- *Number of people per frame:* 13-53
- *Number of people instances in total:* 62325

4.1 Separation of data to training, validation and test set

In order to have comparable results to the ones in the publication of Chen et al., I will apply the same test set, that is frames #801-2000. In order to keep the process free of bias, I do not use the test set to tune any hyperparameter, in fact it will only be used for the final testing. For this reason, I divide the frames 1-800 into a training set (80%) and a validation set (20%). The validation set is used to evaluate models trained on the training set with different hyperparameter settings.

- *Training set:* Frames 1-640
- *Validation set:* Frames 641-800

[1]Although the authors give 320x240 in their paper[CLG+12], the data on their website contains 640x480 resolution images. I used the 320x240 scaled down versions in my results.

- *Test set:* Frames 801-2000

Cross-validation, a popular method for tuning hyperparameters, is not appropriate here. In cross-validation the training set is subdivided into n parts. For each hyperparameter configuration n experiments are perfomed, in each case one part is used for testing and the others together for training. Then the hyperparameter combination that attains the lowest average error is chosen. In the case of people counting, however, the data are obtained from a video stream and have an inherent ordering. If we used cross-validation, the test samples would be in most cases between the frames that are used for training. This would lead to interpolation rather than extrapolation, which is undesirable[CLV08], so the validation is done using a fixed part of the dataset that comes after all the training frames.

4.2 Effect of hyperparameters

I examined the effect of changing various hyperparameters on the validation set (test set was kept separate for final testing).

4.2.1 Regularization coefficient and RBF kernel length scale

Multivariate ridge regression has one parameter, the one that balances the trade-off between regularization and fitting the data (C). Figure 12 (*left*) shows that the optimum is around $C = 0.001$. For smaller values of C overfitting occurs, for larger ones underfitting. Multivariate kernel ridge regression with RBF kernel has two parameters: C, and the parameter controlling the length scale of the kernel (γ). I used grid search to optimize these hyperparameters (see Figure 12, *right*).

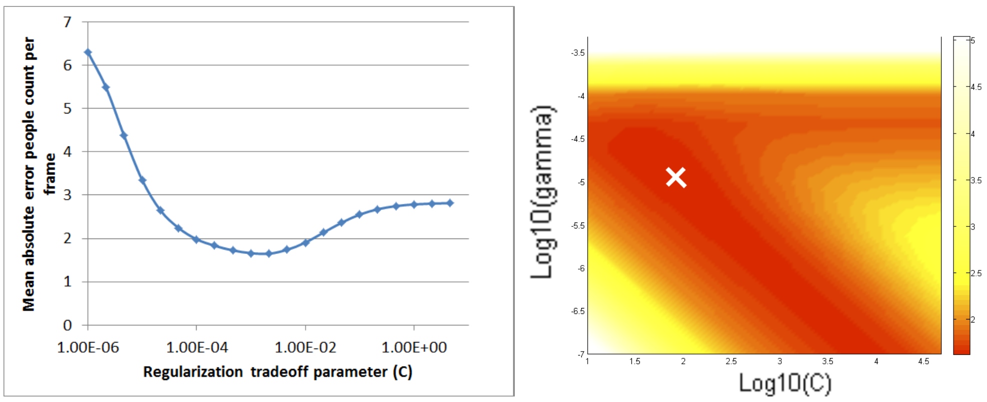

Figure 12: Effect of hyperparameters on the mean absolute error, calculated on the validation set with linear ridge (*left*) and kernel ridge (*right*) regression. The white cross marks the chosen hyperparameter configuration.

4.2.2 Grid resolution and perspective normalization

Grid subdivision	Without perspective correction	With perspective correction
1×1	1.88	1.88
2×2	1.62	1.58
4×4	1.46	1.47
6×6	1.70	1.71

Figure 13: Mean absolute validation error depending on the grid resolution and perspective correction.

This table shows how the mean absolute error varies with the grid resolution and whether linear perspective correction is applied. Surprisingly, perspective correction has almost no effect on the results. It can also be seen that the 4×4 grid size gives the best estimates.

Having determined the best settings and hyperparameters on the validation set, we can now turn to the test set to see how it performs on previously unseen data.

4.3 Global counting evaluation

In the following, the global (per frame) people count estimation is evaluated on the mall dataset. Even though the evaluation is done globally, the 8x8 grid is still used in the method, and the global count is obtained by summing up the predictions for the individual cells (without any rounding to integers).
Following Chen et al., three evaluation criteria are used: mean squared error, mean absolute error and mean relative deviation error. Let M denote the number of test frames, K the number of grid cells $\hat{Y} \in \mathbb{R}^{M \times K}$ the output predicted by the model and $Y \in \mathbb{R}^{M \times K}$ the ground truth people count values.

$$\begin{aligned} E_{sq} &= \frac{1}{M}\sum_{i=1}^{M}\left(\sum_{j=1}^{K}\hat{Y}_{ij} - \sum_{j=1}^{K}Y_{ij}\right)^2 \\ E_{abs} &= \frac{1}{M}\sum_{i=1}^{M}\left|\sum_{j=1}^{K}\hat{Y}_{ij} - \sum_{j=1}^{K}Y_{ij}\right| \\ E_{rel} &= \frac{1}{M}\sum_{i=1}^{M}\frac{\left|\sum_{j=1}^{K}\hat{Y}_{ij} - \sum_{j=1}^{K}Y_{ij}\right|}{\sum_{j=1}^{K}Y_{ij}} \end{aligned} \tag{28}$$

Table 4.3 show the performance comparison of different methods. My linear ridge regression uses $C = 2 \times 10^{-3}$, the kernel version uses $C = 100, \gamma = 10^{-5}$, values obtained by logarithmic grid search over the hyperparameter space.

Learning algorithm	E_{sq}	E_{abs}	E_{rel}
Linear ridge regression 8×8 (own)	8.72	2.38	0.0768
Kernel ridge regression 8×8 (own)	8.43	2.34	0.0756
Kernel ridge regression 4×4 (own)	7.94	2.24	0.0706
Linear ridge regression 8×8 (Chen et al.)	15.7	3.15	0.0986

Figure 14: Comparison of different methods, using a 320×240 px resolution version of the Mall dataset (trained on frames 1-800, evaluated on frames 801-2000).

The kernelized algorithm slightly outperforms the linear one and both of my models outperform the one presented by Chen et al.

4.4 Cell-based inspections

Following the global evaluation, let us now analyze how the grid-based local estimation makes use of information sharing and feature mining.

4.4.1 Information sharing

The weight matrix W obtained by linear ridge regression can be used to quantify how much the features from a given cell contribute to the estimation of the people count in another (or the same) cell. A feature is important if the absolute value of the corresponding weight is large. Let $p \in \{1, ..., K\}$ denote the index of the source cell and $q \in \{1, ..., K\}$ that of the target cell. We then compute the sum of the absolute values of the weights in W that correspond to features extracted from cell p and are used for estimating the count in cell q. This value is then normalized by dividing with the sum of absolute values of all weights that are used in the estimation for cell q. Thus, the formula for measuring information sharing from cell p to q becomes:

$$S_{pq} = \frac{\sum_{i \in \{\text{indices of features extracted from cell } p\}} |W_{iq}|}{\sum_{i=1}^{D} |W_{iq}|} \tag{29}$$

Figure 15 shows the result for each pair of cells as a color coded matrix where a brighter color means higher information sharing.

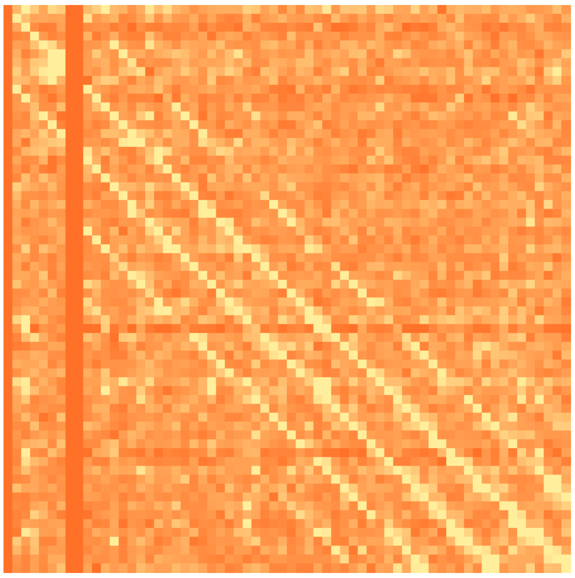

Figure 15: Illustration of how the features extracted from a given cell influence the prediction of the people count in other cells. This matrix has $K = 64$ rows and column. One row of it corresponds to one source cell, one column to one target cell. Brighter color means more influence.

It can be readily seen that there are diagonal structures in the matrix shifted by 8 (the number of cells in a row of the grid). This shows that the greatest influence comes from the target cell's own features, and the ones from above and below it. There are two diagonals to the to the bottom direction, whereas only one to the top direction. This means that even the features in a cell that is by two cells lower can be important for the estimation. Looking at the dataset, it is clear that this is caused by vertical shapes of the bodies and also reflections and shadows on the floor, as these appear much lower than the reference points (head) of the people.

4.4.2 Kernel weight matrix

The weight matrix $A \in \mathbb{R}^{N\times K}$ obtained by kernel ridge regression determines the importance of each frame for the estimation of the people count in each grid cell. Figure 16 shows the matrix as follows:

$$Z_{pq} = \frac{|A_{pq}|}{\sum_{i=1}^{N} |A_{iq}|} \tag{30}$$

The green marking shows that certain grid cells' people counts (corresponding to one column here) are best estimated by comparing the features with those extracted from a certain time interval (a certain range of frames). Vertical stripes of distance 8 are also visible here. This is because cells that are above/under each other have relevant action happening around the same time in the video.

4.5 Feature importance

In Figure 17 the importance of features is illustrated. It is measured by the sum of the absolute values of weights associated with each type of feature (irrespective of the cell from which it is extracted). It can be seen that the values for the same kind of features (such as GLCM entropy) have very similar values.

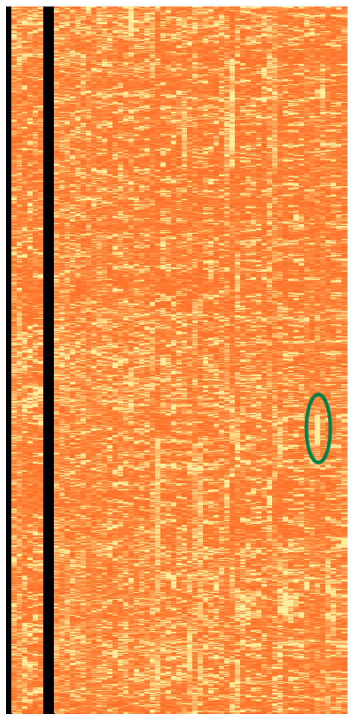
Figure 16: Illustration of the absolute values of the kernel weight matrix A after division by the sum of each column (same principle as with the weight matrix in Figure 15. The black stripes show that those cells are always empty according to the model.

5 Discussion and summary

In this seminar report I examined the task of counting people in crowded public places by means of computer vision algorithms applied to CCTV video frames. I reviewed previous works about the topic and elaborated on feature-based regression approaches. I described the different features that can be extracted from the image frames and gave the theoretical background for the linear approximation of perspective normalization.

I presented the results of my own implementation of a feature-based regression approach to localized people counting, based strongly on the work of Chen et al. My solution turned out to have better performance than the one described in the paper of Chen et al., reaching the performance of $\pm 7.5\%$ mean relative deviation error per frame. This may be due to differences in the details of feature extraction (e.g. different background segmentation), me using data normalization or different hyperparameter choices. I inspected the weight matrices that were obtained during the optimization of a multivariate ridge regression and a multivariate kernel ridge regression model.

One disadvantage of this approach is that with each new camera position a new labeled training set must be created and the regression algorithm must be tuned to new optimal hyperparameters by grid search. This may be solved by using many labeled datasets created at a wide range of different locations, as one could then create a system for learning the task in general and not only for one specific dataset.

In conclusion, it can be stated that the presented solution performs well in a controlled setting, but there is still room for improvement regarding the adaptation to real-life usage scenarios.

References

[BC06] Gabriel J. Brostow and Roberto Cipolla. Unsupervised detection of independent motion in crowds. In *IEEE Computer Vision and Pattern Recognition*, pages I: 594–601, 2006.

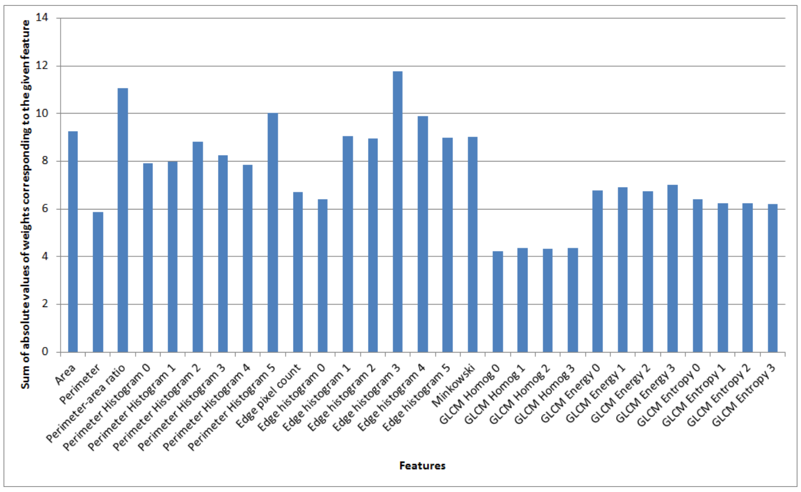

Figure 17: Feature importances in the linear ridge regression model based on the weight matrix W

[BIY+10] Yassine Benabbas, Nacim Ihaddadene, Tarek Yahiaoui, Thierry Urruty, and Chabane Djeraba. Spatio-temporal optical flow analysis for people counting. In *Proceedings of the 2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*, AVSS '10, pages 212–217, Washington, DC, USA, 2010. IEEE Computer Society.

[Can86] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[CCL99] Siu-Yeung Cho, T. W S Chow, and Chi-Tat Leung. A neural-based crowd estimation by hybrid global learning algorithm. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 29(4):535–541, 1999.

[Che13] K. Chen. Mall crowd counting dataset from the queen mary university of london, May 2013.

[CLG+12] Ke Chen, Chen Change Loy, Shaogang Gong, Tao Xiang, and Queen Mary. Feature mining for localised crowd counting. In *BMVC*, volume 1, page 3, 2012.

[CLV08] Antoni B Chan, Z-SJ Liang, and Nuno Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–7. IEEE, 2008.

[DV05] Anthony C Davies and Sergio A Velastin. A progress review of intelligent cctv surveillance systems. *Proc. IEEE IDAACS*, pages 417–423, 2005.

[FCD+11] Joseph Ferenbok, Andrew Clement, A Doyle, RK Lippert, and D Lyon. Hidden changes: From cctv to smart video surveillance. *Eyes Everywhere: The Global Growth of Camera Surveillance, A. Doyle, RK Lippert, and D. Lyon, Abingdon: Routledge*, pages 218–233, 2011.

[FHTN12] Shizuka Fujisawa, Go Hasegawa, Yoshiaki Taniguchi, and Hirotaka Nakano. Pedestrian counting in video sequences using optical flow clustering. In *Proceedings of the 11th international conference on Applications of Electrical and Computer Engineering*, ACA'12, pages 51–56, Stevens Point, Wisconsin, USA, 2012. World Scientific and Engineering Academy and Society (WSEAS).

[Hai87] Yoel Haitovsky. On multivariate ridge regression. *Biometrika*, 74(3):pp. 563–570, 1987.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[HSD73] Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, (6):610–621, 1973.

[JD97] Lee Jiann-Der. Object recognition using a neural network with optimal feature extraction. *Mathematical and Computer Modelling*, 25(12):105 – 117, 1997.

[KBN11] Vasileios Karavasilis, Konstantinos Blekas, and Christophoros Nikou. Motion segmentation by model-based clustering of incomplete trajectories. In *Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part II*, ECML PKDD'11, pages 146–161, Berlin, Heidelberg, 2011. Springer-Verlag.

[KGT05] D. Kong, Douglas Gray, and H. Tao. Counting pedestrians in crowds using viewpoint invariant training. In *Proc. British Machine Vision Conference (BMVC)*, 2005.

[LCC01] Sheng-Fuu Lin, Jaw-Yeh Chen, and Hung-Xin Chao. Estimation of number of people in crowded scenes using perspective transformation. *Trans. Sys. Man Cyber. Part A*, 31(6):645–654, November 2001.

[LSS05] Bastian Leibe, Edgar Seemann, and Bernt Schiele. Pedestrian detection in crowded scenes. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 878–885. IEEE, 2005.

[LZ10] Victor Lempitsky and Andrew Zisserman. Learning To Count Objects in Images. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1324–1332, 2010.

[MCLV98] A. Marana, L. da Costa, R. Lotufo, and S. Velastin. On the efficacy of texture analysis for crowd monitoring. In *Proceedings of the International Symposium on Computer Graphics, Image Processing, and Vision*, SIBGRAPHI '98, pages 354–, Washington, DC, USA, 1998. IEEE Computer Society.

[MdFCLV99] Aparecido Nilceu Marana, Luciano da Fontoura Costa, RA Lotufo, and Sergio A Velastin. Estimating crowd density with minkowski fractal dimension. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 6, pages 3521–3524. IEEE, 1999.

[MHL10] Wenhua Ma, Lei Huang, and Changping Liu. Crowd density analysis using co-occurrence texture features. In *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on*, pages 170–175. IEEE, 2010.

[Ora11] Jonathan Oram. Balancing surveillance between needs of privacy and security. 2011.

[RB06] Vincent Rabaud and Serge Belongie. Counting crowded moving objects. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 705–711. IEEE, 2006.

[WLLX06] Xinyu Wu, Guoyuan Liang, Ka Keung Lee, and Yangsheng Xu. Crowd density estimation using texture analysis and learning. In *Robotics and Biomimetics, 2006. ROBIO'06. IEEE International Conference on*, pages 214–219. IEEE, 2006.

[Ziv04] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31. IEEE, 2004.

[ZN03] Tao Zhao and Ram Nevatia. Bayesian human segmentation in crowded situations. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–459. IEEE, 2003.