Author Queries

JOB NUMBER: MS 213840

JOURNAL: GPAA

Q1 We have inserted a running head. Please approve or provide an alternative.

Q2 Please supply received, revised and accepted dates.

GPAA 213840—8/12/2006—SWAPNA—245462

The International Journal of Parallel, Emergent and Distributed Systems,
Vol. 00, No. 0, 2006, 1–17

Taylor & Francis
Taylor & Francis Group

Counting people using video cameras

DAMIAN ROQUEIRO†* and VALERY A. PETRUSHIN‡¶

†Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA
‡Accenture Technology Labs, Chicago, IL 60601, USA

Q2 *(Received ■; revised ■; in final form ■)*

The paper is devoted to the problem of estimating the number of people visible in a camera. It uses as features the ratio of foreground pixels in each cell of a rectangular grid. Using the above features and data mining techniques allowed reaching accuracy up to 85% for exact match and up to 95% for plus–minus one estimates for an indoor surveillance environment. Applying median filters to the sequence of estimation results increased the accuracy up to 91% for exact match. The architecture of a real-time people counting estimator is suggested. The results of analysis of experimental data are provided and discussed.

Keywords: Multi-camera surveillance; Video surveillance; Counting people; Machine learning algorithms; Number of people estimation

1. Introduction

Counting people is an important task in automatic surveillance systems. It can serve as a subtask in multiple stage processing or can be of primary interest. The robust estimate of people count could improve low level procedures, such as blob extraction or it can provide answers to questions such as, how many people were inside the room between time stamps t_1 and t_2? The related problem is to estimate the density of people in a crowd in places such as a subway platform or a street. Usually the output is a "fill rate" of the space expressed as a percentage.

To solve this problem, different types of location sensors could be used. We will not attempt to exhaustively enumerate all the types of location sensors (see Ref. [1] for a survey). However, we can mention some special sensors that are used mostly in indoor environments and for which there are many commercial solutions already in place: electro-optical, thermic and passive-optics directional sensors are just a few. These sensors are placed in the entrances of buildings, rooms or vehicles. They can detect the passage and direction of a person and report with high accuracy the count of people that entered or left the facility, but they come at

*Corresponding author. Email: droque1@uic.edu
¶Email: valery.a.petrushin@accenture.com

ISSN 1744-5760 print/ISSN 1744-5779 online © 2006 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/17445760601139096

the cost of having to install proprietary hardware (the sensors and the data collection devices) and software to analyse the data. In contrast to this, video-based sensors—or video cameras—have increasingly become a commercial off-the-shelf product for surveillance purposes. Additionally, existing video cameras used for surveillance can be leveraged to perform more tasks such as automatic tracking, behaviour recognition and crowd estimation, among others.

2. Related work

In the past years there has been a bulk of research work in the area of image processing with the objective of obtaining more accurate and reliable people-count estimations.

An intuitive solution to the problem of estimating the size of a crowd in an image will be, literally, to obtain a head count. While this would be a tedious, but yet feasible, task for a human it certainly is a difficult problem for an automatic system. That is exactly the problem tackled in Ref. [2], where wavelets are used to extract head-shaped features from the image. Further processing uses a support vector machine to correctly classify the feature as a "head" or "something else" and applies a perspective transform to account for the distance to the camera.

A similar idea is used in Ref. [3], where a face detection program is used to determine the person count. Unfortunately, as pointed out by its authors, this method is affected by the angle of view at which the faces are exposed to the camera. Additionally, images where a person's back is only visible will result in a poor estimation as well.

Another approach has been suggested in Ref. [4], it aims to obtain an estimation of the crowd density, not the exact number of people. It requires a reference image—where no people are present, in order to determine the foreground pixels in a new image. A single layer neural network is fed with the features extracted from the new image (edge count and densities of the background and crowd objects) and the hybrid global learning algorithm is used to obtain a refined estimation of the crowd density.

Our work compares different classification algorithms for estimating the number of people in an image obtained from a video surveillance camera. This approach differs from previous works in that we do not attempt to obtain and count specific features from the images (head-shaped objects in Ref. [2] or faces in Ref. [3]). We just exploit the correlation between the percentage of foreground pixels and the number of people in an image [5].

3. Datasets

In order to determine how different classification algorithms would perform on both an outdoor environment with high traffic and an indoor environment with moderate and low traffic, we selected the following two sources: (1) a publicly available webcam at Times Square; and (2) two webcams in the premises of the Accenture Technology Labs in Chicago.

The Times Square webcam, denoted herein as *camTS*, is located at the intersection of 46th Street and Broadway in New York City and it streams video images through a publicly available URL [6]. In contrast, the remaining two webcams can only be accessed from within the Accenture intranet and are part of a larger set of webcams of an experimental surveillance

GPAA 213840—8/12/2006—SWAPNA—245462

system already in place. The two cameras chosen for the experiments have opposite views of an elevator area on the 36th floor. This area is the gateway to the offices of the Accenture Technology Labs located on the same floor, and the cameras will be referred as *camEE* and *camEW*, covering the elevators from the East and West, respectively. The three cameras capture snapshot images in the JPEG format but with different resolutions: 353 × 239 for *camTS* and 640 × 480 for *camEE* and *camEW*.

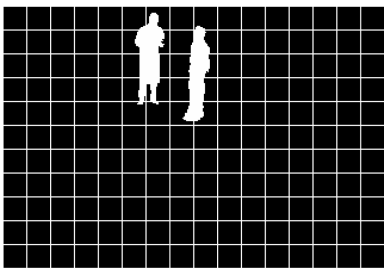
3.1 Image pre-processing

The features we extracted from the images were based on the density of foreground pixels. To extract the foreground pixels we used the well known median filter background modelling technique [7]. According to this technique each background pixel is modelled as median of a pool of images accumulated over some period of time and periodically updated by adding the current image and discarding the oldest one. This technique works well when each background pixel is occluded in less than 50% of images of the pool. To get the foreground pixels the background model is subtracted from the current image pixel by pixel, the absolute values of differences are summed and compared to a threshold. All pixels that have a difference above the threshold are marked as foreground. Then morphological operations are applied to smooth the result. Figure 1(b),(d) show the binary images with the results of foreground pixel extraction. The original images (figure 1(a),(b)) have an *ignore-zone* that

a) *camTS* captured image; ignore zone above $y = 60$

b) *camTS* binary image with foreground pixels; original grid of 83 cells; size of each cell is 25×30 pixels.

c) *camEE* captured image; ignore zone above $y = 40$



d) *camEE* binary image with foreground pixels; grid of 176 cells of 40×40 pixels

Figure 1. Captured images from the three webcams and their respective binary images with foreground pixels (the ignore zones and grids are shown for illustrational purposes only); (a) *camTS* captured image; ignore zone above $y = 60$; (b) *camTS* binary image with foreground pixels; original grid of 83 cells; size of each cell is 25 × 30 pixels; (c) *camEE* captured image; ignore zone above $y = 40$; (d) *camEE* binary image with foreground pixels; grid of 176 cells of 40 × 40 pixels.

represents an area of the image not worth using and from which no features are extracted. There was no ignore-zone defined for *camEW*.

3.2 Feature extraction

For each camera, we created three grids of different size and applied them to the binary images to obtain three datasets. A record of a dataset has the image ID and N real values in the range from 0 to 1, which correspond to the portion of foreground pixels in each cell (count of foreground pixels in the cell divided by its area). The size of the cells is arbitrary and for our experiments we considered grid sizes that differ by a factor of two: double (2x) or half the size (0.5x) of the original (1x). For *camTS* the original cell size was of 25×30 pixels whereas for *camEE* and *camEW* it was 40×40. We wanted to learn how the sizes of the cells influenced the accuracy of the classification algorithms. Figure 1 shows the sample layouts for two of the cameras that have been used in the experiments.

3.3 Labelling

The images from *camTS* were collected in the spring of 2005 during several days between 10:00 am and 3:00 pm. The images for the other two webcams were recorded on 6 different days, between 11:50 am and 2:00 pm, to benefit from the "rush-hour" activity at lunch time. They were captured at a frame rate that varied from 2 to 9 images/s and they were not synchronized (*camEE* could have captured one image in 1 s while *camEW* captured four in the same period). This raw set of images was analysed and several hundreds of them had to be discarded due to flickering in the cameras. Once the final set of images was clearly defined, we manually labelled them associating to each image the number of people visible in it. Even if part of a person was visible, it was still counted. The count was added as another attribute to the existing records in the datasets.

Figure 2(a)–(c) show the distribution of ground truth counts of people for the three original datasets. The original data contain 1452 images for *camTS*, 17,604 images for *camEE* and 26,718 images for *camEW*. As it can be seen from the two distributions in figure 2(b),(c), the datasets for the webcams covering the elevator area were biased toward low number of people. To improve the performance of the classification algorithms we decided to create two reduced datasets that have a more uniform distribution for low number of people. These datasets have a total of 3565 images for *camEE* and 4974 images for *camEW*, and their records were randomly selected from the original datasets. The distribution of labels in the reduced datasets is presented in figure 2(d),(e).

4. Classification algorithms

This section lists the algorithms used in our experiments and their parameters. For the sake of clarity, we would like to briefly explain how the algorithms were tested and what data they used. This digression will help the reader to understand how the datasets were actually processed. Each image that was captured and processed will have a matching record in a dataset with: a label and N values. The label is the count of persons in the image and the N values are the proportion of foreground pixels in each cell of the N-grid. For a given

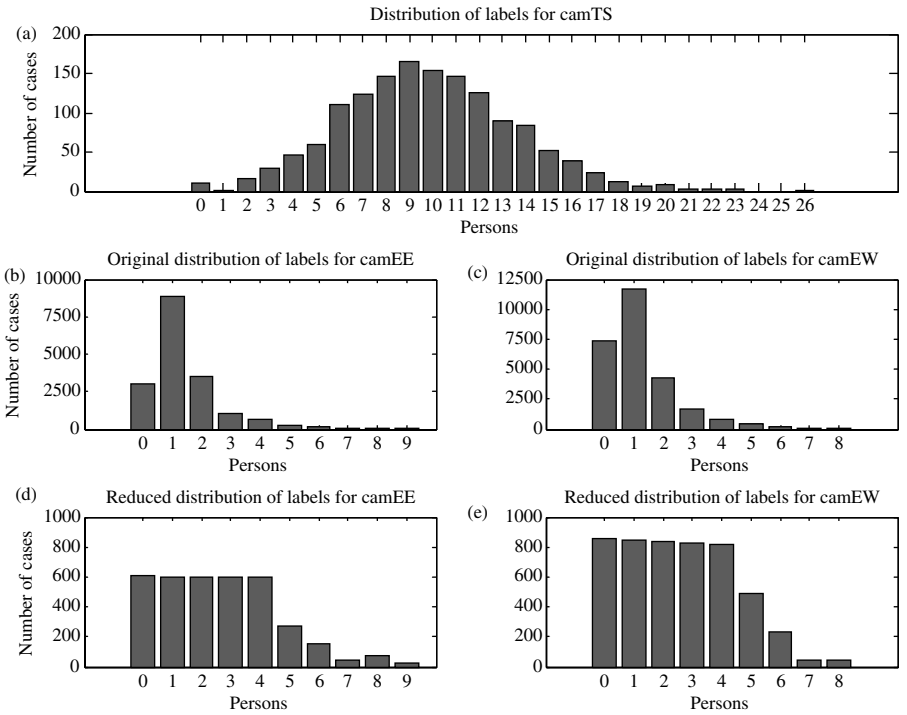


Figure 2. Distribution of labels for the original and reduced datasets.

classification method, some records from the dataset are used to train it. The rest of the records are used to test it and the labels are compared to the predicted values. This gives us a measure of the accuracy of the algorithm to estimate the count of persons in unseen images from the test set.

4.1 Algorithms implemented

We implemented the following algorithms: (a) *multiple linear regression*; (b) *k-nearest neighbour* with Euclidean distance; (c) a two-layer resilient *backpropagation neural network* (BPNN) with a log-sigmoid transfer function for the first layer and ten neurons [8,9]. The second layer has a linear transfer function and its output was rounded to the closest integer number; (d) a two-layer *probabilistic neural network* (PNN) [10]. The first layer is a radial basis layer that computes the distance (probability) of a new vector to the training input vectors and then a competitive layer finds the maximum of these probabilities; and (e) *support vector machines* (SVM)

5. Results

The methodology implemented to compare the above mentioned classifiers follows the traditional sequence of steps used in creating and evaluating supervised learning algorithms. Each classifier was trained and tested with different datasets and, unless otherwise noted, all the experiments detailed in this section were repeated 100 times to attain statistical

significance. At each iteration, 70% of the records of the dataset were randomly selected to train the classifier and the remaining 30% was used to test it. Then the labels in the test set were compared to the output generated by the classifier and the accuracy of the classifier was estimated.

The records in the training and test set were randomly selected using a uniform distribution. Before an experiment started, all the records in the dataset were shuffled to avoid any side effects due to the chronological order of the images§. The experiments were run using Matlab® 7.1 on a Pentium 4, 2.8 GHz CPU with 512 Mb of RAM. Two-third party toolboxes were used for SVM [11] and *k*-nearest neighbour [12].

Before presenting the results of our experiments, we would like to revisit the goal of this work. Our original goal of *finding a method to estimate the count of people in an image* can be refined further as two sub-goals:

(1) Find a classifier to determine with certain reliability if there are zero-persons in an image.
(2) Find a classifier to estimate the count of people in an image that has one or more people.

In some real settings it makes sense first to filter out frames when no people are present in front of a camera. This two-step process can both expedite the recognition process and improve the accuracy of the person count estimation. An architecture that will implement this idea is depicted in figure 3. First, an image is captured and pre-processed to obtain the foreground pixels. Based on a given grid, the percentage of foreground pixels in each cell of the grid is determined. These values are then used by an already trained classification algorithm that determines if the number of people in the image is zero (zero-person detector). If the image is classified as having zero people, it is discarded. Otherwise another classifier is used to estimate the people count and the whole process is repeated for a new image.

This section describes our experience for finding the best classifier for each of the above mentioned sub-goals. For zero-person detection we tested *k*-nearest neighbour, the PNN and SVM with the 1x grid. For counting people we tested all the classifiers, except for SVM, with the three grid sizes (1x, 2x and 0.5x).

5.1 Zero-person detection

Here we tackle the problem of creating a recogniser that classifies images into two classes, when (a) no person; and (b) one or more persons are visible in the image. Taking into account that a simple threshold applied to the percentage of foreground pixels does not work well (for example, when the elevator doors opened or closed and no people were visible), we decided to use machine learning techniques to solve the problem. We evaluated three different classifiers for this task: (1) SVM, (2) *k*-nearest neighbour with $k = 1$, and (3) a PNN. The classifiers were tested using the original datasets described in Section 3.3, where all labels with values 1 or more were substituted with value 1.

§As it will become clear in Section 6, training the classifiers with shuffled data provides us with a baseline for accuracy that leaves room for improvement, e.g. if the classifier can take advantage of the chronological order of the images.

GPAA 213840—8/12/2006—SWAPNA—245462

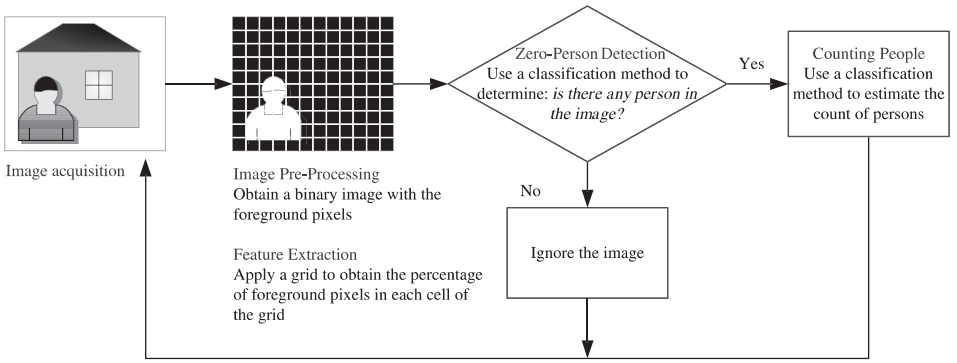


Figure 3. Architecture of a real-time person counter.

Our results show that SVM was slightly more accurate than the other classifiers in two datasets (*camTS* and *camEE*) while having a comparable accuracy to *k*-nearest neighbour for *camEW*. In general, SVM and *k*-nearest neighbour were significantly more accurate than the probabilistic neural network. Table 1 shows the confusion matrices obtained for *camEE*.

To obtain the execution times, we had to perform two different time measurements: one at the group level—measure the time to train 70% of the dataset and to classify the remaining 30% and one at the individual record level. Although high accuracy is very desirable, it cannot come at the price of a slow performance. In that aspect, table 2 shows how *k*-nearest neighbour outperforms SVM by a factor of 10.

Clearly, from the values in table 2, we cannot take the time required by a method to classify 1070 records and divide it by 1070 to obtain the time to classify one record. That is why the average execution time of classifying only one record had to be computed separately. From an implementation perspective these functions have an overhead that cannot be ignored, especially since we target to do classification in real-time.

5.2 Counting people

As it was mentioned before, if an image is classified as having one or more persons in it we proceed to run a classifier that will estimate the people count in the image. This means that, when comparing the classification algorithms, we can assume that the datasets have no records with labels equal to zero. To do so, we used the reduced datasets described in Section 3.3 and deleted the records with images that were labelled as having zero-persons.

Table 1. Confusion matrices with results of classification between 0 or 1 + persons (one or more) for *camEE* and grid size 1x.

		SVM		*1-nearest neighbour*		*PNN*	
		Predicted		*Predicted*		*Predicted*	
		0	*1 +*	*0*	*1 +*	*0*	*1 +*
Actual	0	96.49	3.51	97.56	2.44	100.00	0.00
	1 +	0.60	99.40	1.14	98.86	4.09	95.91
Total Accuracy			98.86%		98.64%		96.61%

Table 2. Comparison of execution times between classifiers for *camEE*.

	Number of records	*Mean time (in seconds)*		
		SVM	*1-nearest neighbour*	*PNN*
Train	2495 (70%)	68.94	N/A	0.20
Classify	1070 (30%)	7.09	25.46	32.72
	One record	0.26	0.02	0.05

These "reduced-no-zeros" datasets have 1442, 2960 and 4121 records for *camTS*, *camEE* and *camEW*, respectively.

The following classification methods were evaluated:

(1) Multiple Linear Regression; (2) Regression Tree; (3) *k*-nearest neighbour; (4) Backpropagation neural network (BPNN); and (5) Probabilistic neural network (PNN). Once the results were obtained, the methods were compared according to three variables of interest: (a) accuracy; (b) training time; and (c) execution time.

Accuracy.The accuracy of a method was categorized in four levels: the first, and most desirable one, is exact match. This level has the percentage of accurate estimations, i.e. when the classifier—for a given image—reported a person count that was equal to the label assigned to that image. The other three levels account for the estimations that had a difference of 1 person ($+1$ or -1), 2 ($+2$, -2) and 3 ($+3$, -3) persons, respectively. The accuracy is reported in a cumulative fashion, for example, the percentage of estimations with a difference of 2 includes the exact matches and the estimations that were off by 1.

Training time. This time has different implications according to the classification method analysed. For Multiple Linear Regression, the training time is the time required to solve the system of equations. In the case of a Regression Tree, it is not only the time necessary to create the tree but also to prune it. *k*-nearest neighbour does not have a training phase and for both neural networks, the training time has the correct interpretation. In all cases, this time is expressed in seconds and involved the processing of 70% of the dataset in question.

Execution time. It is the time required for a classifier to classify only one record. It will also be expressed in seconds and, in the case of very small numbers, in scientific notation.

Table 3 shows the accuracy of the classifiers for exact match. At first sight, Linear Regression and Regression Tree do not seem to be good classifiers. If instead of focusing on exact matches we consider a difference of 3 ($+3$ or -3) the mean accuracy for this two classifiers can in some cases be above 95% and as close as to 100%. Of course, this relaxation in our prediction can be acceptable in an environment similar to the one covered by the Times Square webcam. In this case, when having to determine the count of people in a crowd of 20

Table 3. Mean accuracy (in percentage) for exact match.

Dataset method	*camTS*			*camEE*			*camEW*		
	1x	*2x*	*0.5x*	*1x*	*2x*	*0.5x*	*1x*	*2x*	*0.5x*
Multiple linear regression	21.7	20.0	22.5	47.9	48.6	44.6	56.9	50.5	58.4
Regression tree	22.7	21.3	22.2	69.6	69.7	68.3	71.9	72.9	70.4
1-nearest neighbour	36.5	37.0	36.9	83.3	79.6	85.0	86.6	85.5	86.9
BPNN	27.3	21.5	31.2	65.7	59.7	67.0	69.9	65.0	69.5
PNN	37.0	37.0	30.5	83.0	72.9	80.7	86.2	83.5	79.8

GPAA 213840—8/12/2006—SWAPNA—245462

persons, an estimation of 23 may be acceptable. On the other hand, in a confined and controlled area as the one covered by *camEE* and *camEW*, we would expect a higher accuracy for exact matches and will probably consider a difference of up to 1. For this, 1-nearest neighbour, the BPNN and the PNN provide a better estimation. The *k*-nearest neighbour classifier was evaluated for different values of $k = 1, 3, 5$ and 7 and the best results were obtained when $k = 1$, with the mean accuracy degrading for higher values of *k*. For example, for *camEE* (dataset 1x) and $k = 3$, 5 and 7 the mean accuracy for exact match was 80, 77.6 and 76.3%, respectively.

It is worth mentioning that the density of the grid has an effect on *k*-nearest neighbour. For the webcams in the elevator area we had a higher accuracy when the density of the grid increased (the size of the cells decreased). As an example, for *camEE* and grids of 1x, 2x and 0.5x the accuracies of up to a difference of 1 were 95.9, 94.5 and 96.4%. It is just a slight increase in accuracy but, as we could have expected, the classifier takes advantage of the extra number of features to find a better match.

The BPNN does not provide a high accuracy for exact matches but improves dramatically when considering a difference of 1. In particular, for *camEE* and *camEW* the precision increases at least to 95%. The PNN outperforms the BPNN in exact matches for all datasets and grid sizes—except for *camTS* and grid 0.5x—but has a comparable performance with BPNN when considering a difference of 1. Also of interest, is the fact that the number of cells in the grid does not improve the classification for PNN. Considering the mean accuracy of these methods, 1-nearest neighbour and PNN provide equivalent results. Figure 4 compares the mean accuracy for exact matches in the original grid (1x) for all three webcams.

Figures 5 and 6 illustrate the mean accuracy for the different labels. In the case of *camTS* (figure 5), 1-nearest neighbour gives slightly better estimations for exact matches. For the other webcams (figure 6(a),(b)), as it was mentioned before, the mean accuracy of the PNN is comparable to that of the BPNN when considering a difference of 1.

The training and execution time were also calculated for each classifier. In practice, the training time is not important in determining the most appropriate classifier because training is conducted off-line. Therefore, we put more stress on comparing the methods based on their execution time, i.e. the time it takes the classifier to estimate the number of people in one image. Table 4 lists the execution times of the classifiers for the different datasets.

Due to their architecture the PNNs have shorter training time and larger execution time in comparison to the BPNN, which in turn is computationally more expensive to train, but faster to execute. The execution of multiple linear regression is just a dot product of two vectors,

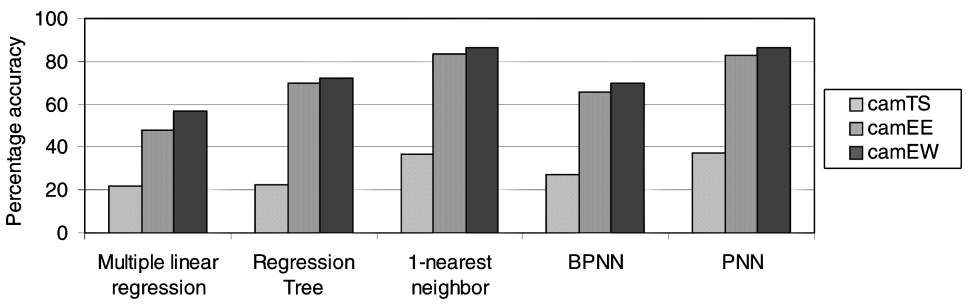


Figure 4. Comparison of mean accuracy (exact match) for three cameras.

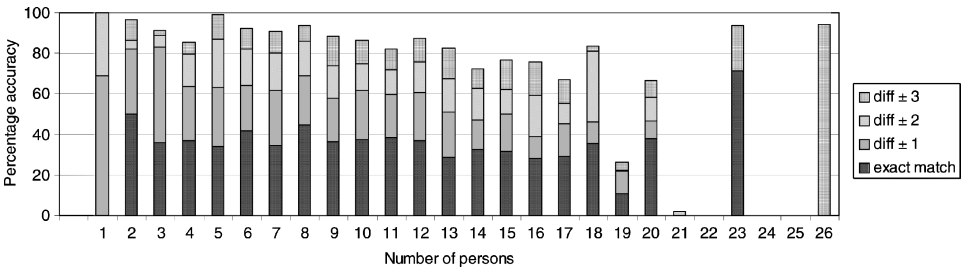


Figure 5. Results for 1-nearest neighbor (*camTS* 1x).

thus it has extremely fast execution time. The execution time of a nearest neighbour approach depends on the number of cells and size of the dataset.

6. Extensions

The results presented in the previous section provide a foundation for creating a real-time person counter as the one depicted in figure 3 with two main modules: (a) a zero-person classifier; and (b) a counting-people estimator. We can pick up a pair of classifiers that gives the best accuracy, but the total accuracy depends on how well the counting-people estimator treats the situation when it is fed with wrong output from the zero-person classifier. In general we have three sources of error, namely:

(1) An image with no persons is classified as $1+$ by the zero-person classifier.
(2) An image with $1+$ persons is classified as 0 by the zero-person classifier.
(3) An image with $1+$ persons is classified as $1+$ by the zero-person classifier but the final count is incorrectly estimated by the counting-people classifier.

When the percentage of errors from 1 to 2 are sufficiently close to zero, the accuracy of the whole system—for exact match—will be close to what was shown in table 3 (note that the third source of error listed before is the complement of the figures in table 3). We know from table 1 that, for *camEE*, we can expect SVM and 1-nearest neighbour to produce 1.14 and 1.36% of errors 1 and 2 combined, respectively. This section addresses these problems and attempts to

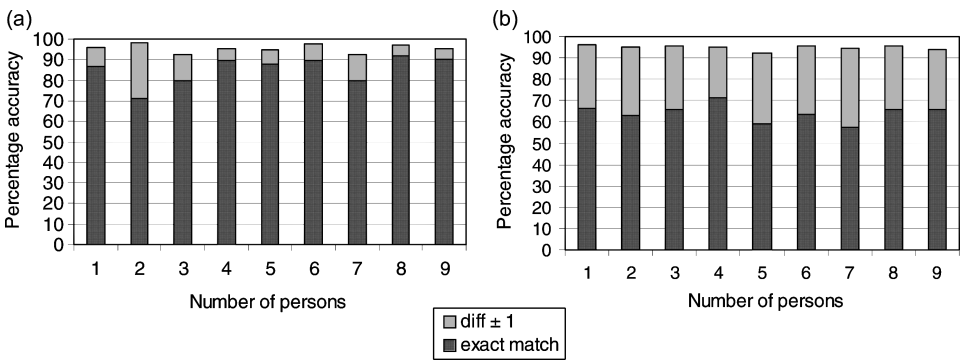


Figure 6. Mean accuracy for *camEE* (1x); using (a) PNN and for (b) BPNN.

Table 4. Average execution time in seconds.

Dataset method	*camTS*			*CamEE*			*camEW*		
	1x	*2x*	*0.5x*	*1x*	*2x*	*0.5x*	*1x*	*2x*	*0.5x*
Multiple linear regression	1.e-5	1.e-5	1.e-5	2.e-5	1.e-5	2.e-5	2.e-5	1.e-5	4.e-5
Regression tree	8.e-4	7.e-4	8.e-4	0.002	0.001	0.001	0.001	9.e-4	0.006
1-nearest neighbour	0.005	0.001	0.010	0.020	0.006	0.084	0.031	0.009	0.122
BPNN	0.005	0.005	0.005	0.006	0.005	0.006	0.005	0.005	0.005
PNN	0.015	0.007	0.023	0.045	0.017	0.160	0.061	0.021	0.224

simulate (off-line) a real-time person counter. Our goal is to extend our analysis and to explore techniques that could help the real-time person counter to increase its accuracy and minimize the effect of the errors in the predictions of 0 or 1 + persons for the elevator cameras *camEE* and *camEW*. The strategies we will explore include: (1) utilization of median filters; (2) creation of an assorted (heterogeneous) grid; and (3) variable (feature) selection.

6.1 Median filters

A median filter with window of size k (where k is an odd number) scans a sequence of numbers and replaces a value in the middle position with the median value of the window. The advantage of the median filters is that they can smooth the result without averaging the values. Applying median filters to a sequence of chronologically ordered images that were captured within a certain time interval, we can correct the output of our classifiers. Let us assume that a counting-people classifier predicted the sequence of counts depicted in figure 7(a). If the time interval between the images is not large, we would expect a smooth transition of counters. As a result of this, we can notice that number 8 in the 5th position is an incorrect prediction due to, probably, flickering of the camera, and the value 1 in the 15th position is likely the result of occlusion of a person by another one. Using a median filter with a window of size 3 corrects the errors (figure 7(b)).

The real-time person counter can use a median filter to correct the results of the zero-person classifier, the counting-people estimator or both. Using a median filter introduces a delay, which depends on the frame rate of the camera and the filter's window size. For majority of applications a delay of 1–3 s could be acceptable when it brings a benefit of increasing accuracy by 3–5%. We assume that the output of the zero-person classifier will be corrected with a median filter before invoking the counting-people classifier for those images classified as 1 + , and another median filter will be applied to the results to create the final estimations. At every point, if the interval between images is larger than a threshold then the

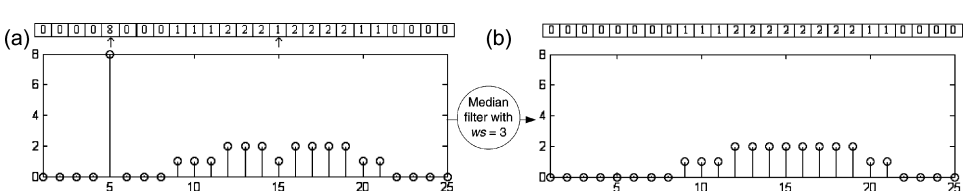


Figure 7. A sample sequence of counts with (a) two errors and (b) the corrected sequence after a median filter with window size $ws = 3$ was applied.

system outputs the results without filtering, cleans the window buffer and adds newly arrived images to the window buffer.

It is important to mention that the classifiers we used for prototyping the real-time person counter were trained using the reduced datasets described in Section 3.3. However, to test our prototype we used the original dataset that takes advantage of the sequential ordering of the images.

Figure 8 shows the accuracy levels attained by applying median filters of different window sizes to the output of the zero-person classifier. It is interesting to observe that for both SVM and 1-nearest neighbour in *camEE*, the larger the window size, the higher the accuracy. On the contrary, *camEW* degrades for larger window sizes of the median filter. This is due to the characteristics of the data we collected where *camEE* had a higher frame rate than *camEW*. The latter will have images separated by a longer period of time, therefore, allowing more fluctuation between consecutive frames. After a window size of 5 for SVM and of 7 for 1-nearest neighbour, the median filter for *camEW* changes some of the correct predictions and makes the classifiers perform worse.

Different window sizes of median filters were also applied to the output of the classifiers used for counting people. Figure 9 shows these results for the two neural networks (1-nearest neighbour had a comparable performance to that of the PNN). In this case, as opposed to what happened with zero-person detection, a large window size does not degrade the performance of the classifiers. This is simply because the median filter is applied to the results of the counting-people classifiers, which include all the zeros already predicted by the zero-person classifier.

Additionally, figure 10 shows how the accuracy per label changes with different window sizes of the median filter. The accuracy per label increases as the window size increases for labels 1–3. For other labels (e.g. 7) it fluctuates or increases slightly to decrease later on again (e.g. 4–6).

After analysing the accuracy gain by applying a median filter to the outputs of the zero-person detection and counting-people classifiers independently, figure 11 shows the final accuracy when both of them are combined according to the sequential steps shown below. Each step gets its input from the previous step:

(1) Run the zero-person detection classifier (SVM) and obtain an initial set of binary predictions.
(2) Apply a median filter with a window size of 5.

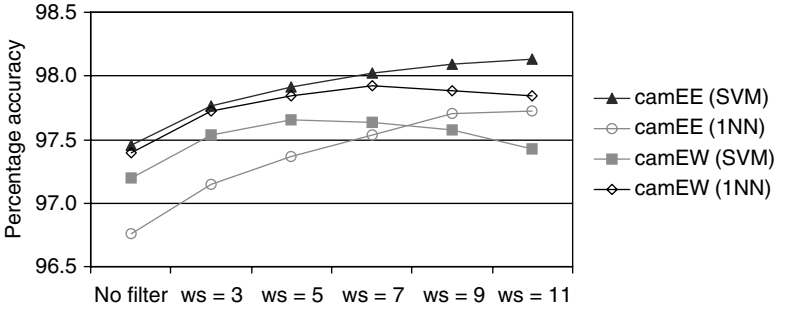


Figure 8. Accuracy of zero-person detection for the elevator cameras using median filters to correct the predictions of SVM and 1-nearest neighbor (1x).

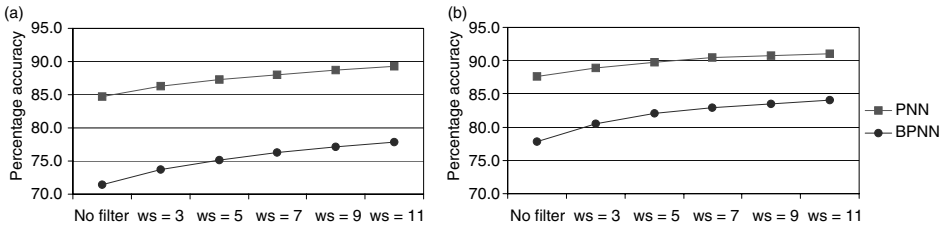


Figure 9. Accuracy of the counting-people classifiers for (a) *camEE* and (b) *camEW* using a median filter to correct the predictions of PNN and BPNN (with grid 1x).

(3) Run a counting-people classifier (1-nearest neighbour) and obtain an initial set of predictions for the records marked as having 1 + persons.
(4) Apply different window sizes of median filter.

6.2 Assorted grid

One of the premises of our work was to provide a way of extracting features from an image without having to know the gory details behind the camera calibration parameters. That is precisely why the datasets described in Section 3 were created with grids of equally-sized cells. We were able to show for 1-nearest neighbour (table 3) that the smaller the size of the cell, the higher the accuracy.

If we consider the near-far effect caused when a person, close to a camera, occupies more area than the same person far from it, we will have in a grid with cells of the same size, fewer number of cells with foreground pixels when the person is far from the camera as opposed to a much larger number when she is closer. In the rest of this section we will explore if a classifier that uses an assorted grid, i.e. a grid with smaller cells for the areas far from the camera and larger cells for the areas in its proximity, can perform better than a classifier that uses equally-sized cells. The simple heuristic we used to determine the size of the cells was to have a blob covered by no more than 4 rows. Figure 12 shows a sample assorted grid for *camEE*.

Table 5 (columns "%" and "grid") shows the best mean accuracy results presented in table 3 for *camEE* and *camEW*. These results are compared to the accuracy obtained by using the assorted grid (herein referred to as Ax). For example, *camEE* with a 1-nearest neighbour

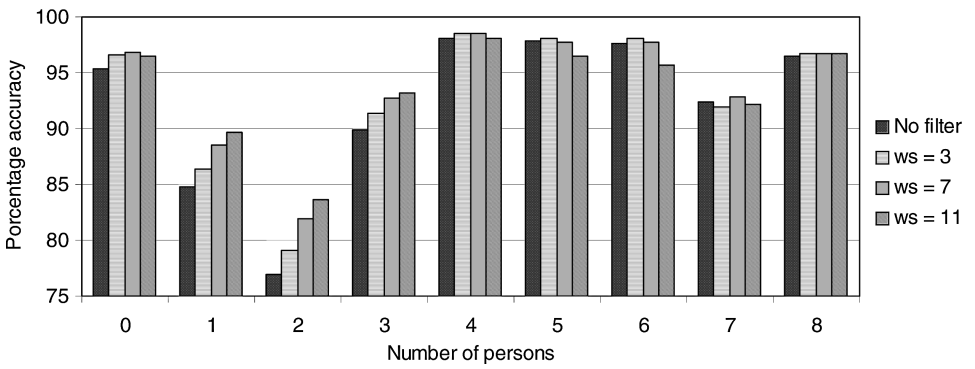


Figure 10. Accuracy per label using a median filter to correct the predictions of 1-nearest neighbor (for *camEW* and grid 1x).

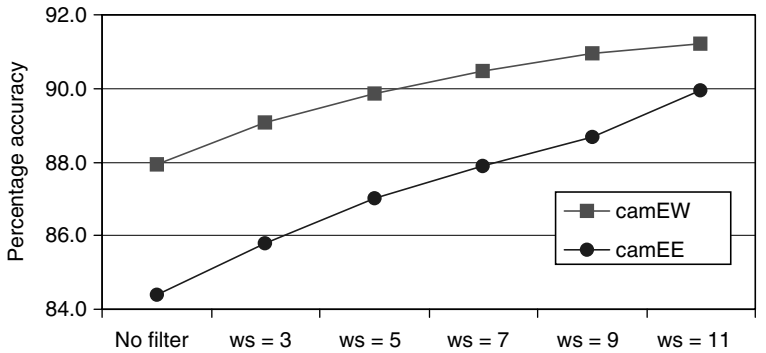


Figure 11. Accuracy of counting-people classifier (1-nearest neighbor 1x) for different window sizes of median filter. The input to the classifier was generated by zero-person detection (SVM 1x) with median filter $ws = 5$.

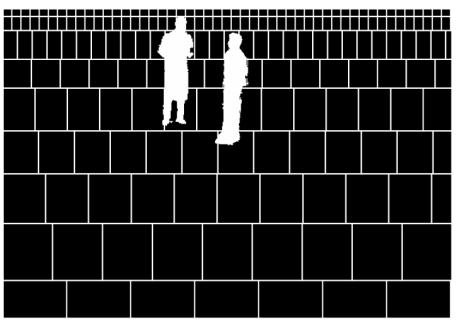
classifier using the grid 0.5x gave the highest accuracy in table 3. Additionally, when we compare this value to the one obtained when using Ax, the 0.5x grid outperforms Ax (85 vs. 84.7%, respectively). However, the highest accuracy obtained for *camEE* is now when the PNN uses the assorted grid (85.5%). In a similar way, *camEW* has the highest accuracy for PNN when using Ax. In general, using assorted grid does not improve the performance of people counting algorithms.

6.3 Variable selection

Variable and feature selection has become a topic of major relevance in machine learning. Although the previous two sections have described two important sources of improvement in the accuracy of our predictions, our analysis will not be complete unless we explore this topic and its benefits for our problem set. Among the many potential benefits of using the variable and feature selection methods described in Ref. [13] we will focus on determining what variables our classifiers can ignore to speed up the running time whilst maintaining a comparable accuracy to the one obtained with all the variables.

By doing a visual inspection of the images there are certain areas that intuitively can be marked as irrelevant or as sources of noise. As Ref. [13] suggests, having domain knowledge

a) Original image (with ignore zone)



b) Binary image with foreground pixels and assorted grid of 152 cells

Figure 12. Assorted grid applied to *camEE*. Our heuristic dictates that one blob cannot span through more than 4 rows. (a) Original image (with ignore zone); (b) Binary image with foreground pixels and assorted grid of 152 cells.

Table 5. Comparison of mean accuracy (in percentage) for exact match using the assorted grid (Ax).

Dataset Method	*camEE*			*camEW*		
	%	*Grid*	*Ax*	*%*	*Grid*	*Ax*
Multiple linear regression	48.6	2x	52.3	58.4	0.5x	55.2
Regression tree	69.7	2x	54.2	72.9	2x	64.4
1-nearest neighbour	85.0	0.5x	84.7	86.9	0.5x	87.0 >
BPNN	67.0	0.5x	66.3	69.9	1x	69.8
PNN	83.0	1x	85.5	86.2	1x	87.2 >

of the problem helps to create an "ad hoc" set of variables or features that can be better than the one generated by an automated process. In this section we will analyse the performance of our classification algorithms when using different sets of variables obtained by: (1) *ad hoc* method, and (2) the implementation of two different—yet simple—variable selection methods: *information gain* [14] and *relief* [15].

For these last two methods, we ranked the variables using 5-fold cross-validation and 1-nearest neighbour as classifier on the reduced datasets with grid 1x. We obtained a ranking of variables (cells in our case) and discarded those that were ranked low. The results of executing all the counting-people classifiers after applying variable selection can be seen in table 6. The column % has the reference accuracy obtained for that method and camera using the grid 1x (table 3).

For *camEE*, when doing variable selection, we discarded the bottom 43% of the variables based on the rankings obtained by information gain and relief. Each of these methods provided a different ranking, and as a result of this, the variables that were discarded varied from one method to the other. The subset chosen by information gain performed very well, even outperforming the reference accuracy. The best results for this camera were obtained with the *ad hoc* method that discarded 24% of "hand-picked" variables. See figure 13 to compare the cells (variables) discarded by information gain, relief and by the *ad hoc* method.

For *camEW*, we wanted to go somewhat farther in our desire of discarding variables. We decided to ignore the bottom 49% of them (according to the rankings). Unfortunately in this case, there was no significant improvement in accuracy but we were able to find a subset with almost half of the variables that was performing at a comparable level of accuracy (for the PNN and information gain, the subset of variables outperformed the complete set).

Table 6. Comparison of mean accuracy (in percentage) for exact match using variable selection.

Dataset method	*camEE*				*CamEW*			
	%	*IG*	*Relief*	*ad hoc*	*%*	*IG*	*Relief*	*ad hoc*
Multiple linear regression	47.9	48.7	45.7	48.0	56.9	56.3	56.4	56.6
Regression tree	69.6	60.4	54.5	60.2	71.9	61.2	62.6	61.7
1-nearest neighbour	83.3	83.6	81.9	83.7	86.6	86.5	86.3	85.3
BPNN	65.7	65.2	60.1	64.8	69.9	68.4	69.4	68.0
PNN	83.0	83.7	80.9	83.9	86.2	86.3	86.1	84.9

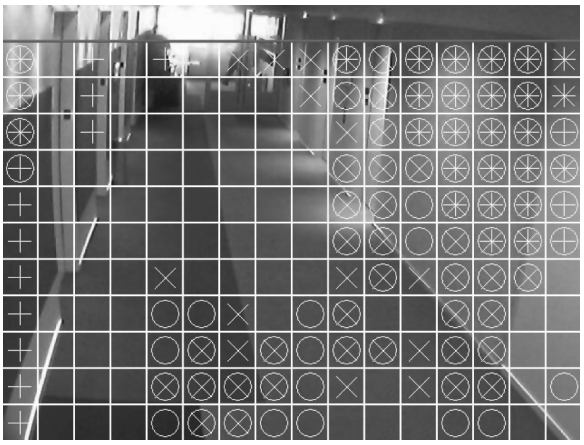


Figure 13. For *camEE*, cells marked were ignored during the training and test phase of the classifiers using the following approaches: *ad hoc* (+); information gain (○); relief (×).

Summarizing, we can tell that automatic feature selection approaches can give good results that are comparable or better than the *ad hoc* approach (at least for our problem domain).

7. Summary and future work

We described an approach to estimate the number of people in an image using rather simple features—a portion of foreground pixels in rectangular areas that cover the image. It turns out that these simple features fed into modern machine learning algorithms can produce very impressive results and can be used for creating a real-time people counting estimator for surveillance applications.

Our count estimator was divided into two tasks: (1) zero-person detection and (2) counting people in an image with one or more persons present in it. For each task we compared different classifiers based on their accuracy and execution times. The next step was to find methods to improve the baseline results. We suggested the use of median filters and assorted grids to increase the accuracy of the classifiers. Additionally, variable selection was examined with a hope to find a much smaller subset of the variables that would reduce the running time of the classifier without degrading the accuracy.

The next step will be to implement a prototype to count persons in real-time with an architecture similar to the one described in figure 3. Moreover, there are several directions of future research. First, all our recognizers do not take into account the interdependencies between frames. They assume that features are independent in consecutive frames. To conform to these requirements we shuffled the dataset before applying the classifiers. But as it was pointed out in Section 6.1, in an actual real-time implementation, our person counter will receive frames in chronological order and the system should be able to exploit this. In addition to the suggested extension of using median filters, another possibility could be to use recurrent neural networks such as Generalized Locally Recurrent Probabilistic Neural Networks (GLRPNNs) [16].

As to the extensions described in Section 6, the analysis we made was by no means exhaustive. Therefore, we can explore different heuristics to set the size of the assorted grids,

GPAA 213840—8/12/2006—SWAPNA—245462

or we can try more sophisticated algorithms for variable and feature extraction such as wrappers, filters or embedded methods.

References

[1] Hightower, J. and Borriello, G., 2001, Location systems for ubiquitous computing, *Computer*, August **34**(8), 57–66.

[2] Lin, S.-F., Chen, J.-Y. and Chao, H.-X., 2001, Estimation of number of people in crowded scenes using perspective transformation, *IEEE Transactions on Systems, Man and Cybernetics*, November, Part A **31**(6), 645–654.

[3] Sweeney, L. and Gross, R., 2005, Mining images in publicly-available cameras for homeland security. *AAAI Spring Symposium on AI Technologies for Homeland Security, Palo Alto*.

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

[5] Davies, A.C., Yin, J.H. and Velastin, S.A., 1995, Crowd monitoring using image processing, *Electronics & Communication Engineering Journal*, February **7**(1), 37–47.

[6] Times Square webcam URL:, http://www.earthcam.com/usa/newyork/timessquare/.

[7] Cheung, S.-C.S. and Kamath, C., 2004, Robust techniques for background subtraction in urban traffic video. In: S. Panchanathan and B. Vasudev (Eds.) *Proceedings of SPIE, Visual Communications and Image Processing 2004*, January Vol. 5308, , pp. 881–892.

[8] Boukadoum, M., Bensaoula, A. and Starikov, D., 2003, A neural network-based system for live bacteria detection, *Proceedings of (403) Artificial Intelligence and Applications*.

[9] Bates, R., Sun, M., Scheuer, M.L. and Sclabassi, R., 2003, Seizure detection by recurrent backpropagation neural network analysis. *4th International Symposium on Uncertainty Modelling and Analysis*, pp. 312–320.

[10] Specht, D.F., 1990, Probabilistic neural networks, *Neural Networks*, January **3**(1), 109–118.

[11] Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B. and Vandewalle, J., 2002, *Least Squares Support Vector Machines* (Singapore: World Scientific), (ISBN 981-238-151-1).

[12] Stork, D.G. and Yom-Tov, E., 2004, *Computer Manual in MATLAB to Accompany Pattern Classification*, 2nd ed., April, (ISBN: 0-471-42977-5).

[13] Guyon, I. and Elisseeff, A., 2003, An introduction to variable and feature selection, *Journal of Machine Learning Research*, **3**.

[14] Ayan, N.F., 1999, Using information gain as feature weight. *Proceedings of the 8th Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN'99)*, Istanbul, Turkey, June.

[15] Kira, K. and Rendell, L.A., 1992, A practical approach to feature selection. *Proceedings of the Ninth International Workshop on Machine Learning*, July, pp. 249–256.

[16] Ganchev, T., Fakotakis, N., Tasoulis, D.K. and Vrahatis, M.N., 2004, Generalized locally recurrent probabilistic neural networks for text-independent speaker verification. *IEEE International Conference on Acoustics, Speech, and Signal Processing*.