Feature mining for localized crowd counting

Seminar Computer Vision and Machine Learning

István Sárándi

Advisor: Wolfgang Mehner

RWTH Aachen University

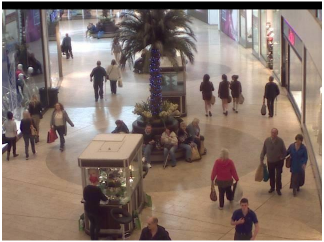
istvan.sarandi@rwth-aachen.de

July 29, 2013

Overview

1. Introduction

Introduction - CCTV surveillance



Introduction - CCTV surveillance

Ubiquity

Millions of cameras (UK: 1.8 million)

Applications

- Prevent crime
- Prevent dangerous crowd dynamics
- Create statistics
- Improve advertisement, etc.

How many people are there?

Introduction - Crowd counting

Functional requirements

- Locality (local-global)
- Directionality (coming-going)
- Quantitative (exact count) vs. qualitative (discrete crowdedness classes)

Introduction - Crowd counting

Functional requirements

- Locality (local-global)
- Directionality (coming-going)
- Quantitative (exact count) vs. qualitative (discrete crowdedness classes)

Non-functional requirements

- Robustness
  - Lighting conditions
  - Camera placement
- Low computational complexity
- Preserve privacy

Overview

1. Introduction
2. Related work
3. Methodology
4. Results
5. Summary

2. Related work

Related work

Two main approaches exist

Detection-based

- Detect each pedestrian
- Count them

Related work

Two main approaches exist

Detection-based

- Detect each pedestrian
- Count them

Regression-based

- Extract feature vector
- Learn a mapping from features to people count (machine learning)

Related work - Detection-based methods

Detection may be

Static

- Detection in each frame independently
- Sliding window + classification

Related work - Detection-based methods

Detection may be

Static

- Detection in each frame independently
- Sliding window + classification

Dynamic

- Detection based on multiple frames (high FPS needed)
- Detect pixel movements (optical flow)
- Cluster trajectories (pixels moving together)

Related work - Detection-based methods

Advantages

- Locality trivial
- Robust detectors exist
- Less manual work for ground truth

Related work - Detection-based methods

Advantages

- Locality trivial
- Robust detectors exist
- Less manual work for ground truth

Disadvantages

- Occlusion problems
- Computationally expensive
- Privacy concerns

Related work - Regression-based methods

Two steps:

Extract features

- Frame $\rightarrow$ feature vector
- E.g. based on edges

Related work - Regression-based methods

Two steps:

Extract features

- Frame $\rightarrow$ feature vector
- E.g. based on edges

Learn regression

- Supervised training
- Ground truth people count needed
- Many possible algorithms
  - Linear regression
  - SVM
  - Neural networks, etc.

Related work - Regression-based methods

Advantages

- Computationally efficient
- Privacy preserving

Related work - Regression-based methods

Advantages

- Computationally efficient
- Privacy preserving

Disadvantages

- Robustness problems
- Locality not automatic

Related work - Locality

- Grid subdivision
- Extract features from each cell
- Estimate head count in each cell

Related work - Locality

Same model used for all cells

- Train one regression model
- The single model has to estimate the head count in any cell

Related work - Locality

Same model used for all cells

- Train one regression model
- The single model has to estimate the head count in any cell

Separate model for each cell

- Train one regression model per cell
- Feature mining
  - Each feature can have different role/weight in different cells

Related work - Locality

Same model used for all cells

- Train one regression model
- The single model has to estimate the head count in any cell

Separate model for each cell

- Train one regression model per cell
- Feature mining
  - Each feature can have different role/weight in different cells

One multi-output model for the whole frame

- Regression input: feature vectors from cells concatenated
- Desired output: vector of head counts in each cell
- Information sharing between cells
  - Each cell's features contribute to every cell's head count estimation

Overview

3. Methodology

Methodology

Based on *Feature mining for localised crowd counting* Chen et al. 2012

As a black box

- Quantitative
- Local
- Privacy preserving
- Computationally efficient

Methodology

Based on *Feature mining for localised crowd counting* Chen et al. 2012

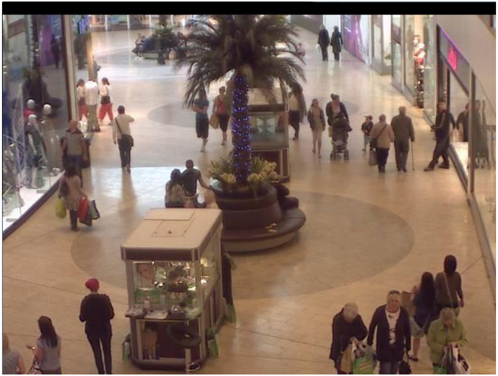
As a black box

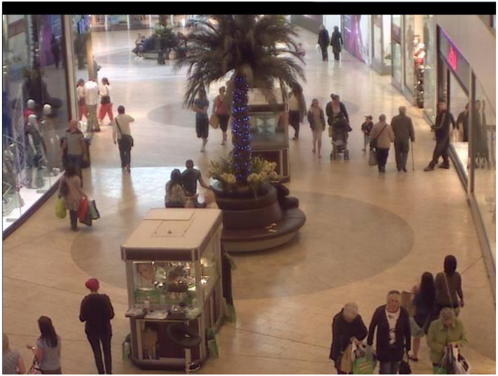
- Quantitative
- Local
- Privacy preserving
- Computationally efficient

As a transparent box

- Regression-based
- Locality with grid and information sharing
- Features based on foreground mask, foreground edges, texture
- Regression with (kernel) ridge regression

Methodology - Feature extraction

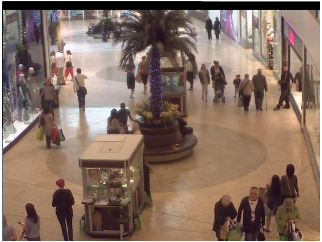




Foreground segmentation needed

Methodology - Feature extraction - Segmentation

Thresholded absolute difference from empty scene



Methodology - Feature extraction - Segmentation

Thresholded difference

Advantages

- Easy to implement
- Efficient to compute

Methodology - Feature extraction - Segmentation

Thresholded difference

Advantages

- Easy to implement
- Efficient to compute

Disadvantages

- Not robust to lighting change and camera repositioning

Methodology - Feature extraction - Segmentation

Thresholded difference

Advantages

- Easy to implement
- Efficient to compute

Disadvantages

- Not robust to lighting change and camera repositioning

Robust alternative: Background model

- Mixture-of-Gaussians color distribution for each pixel
- More weight to recent frames
- Foreground: observed color is improbable

Methodology - Feature extraction - Segmentation

Background model:

Advantages

- Adapts to new conditions

Methodology - Feature extraction - Segmentation

Background model:

Advantages

- Adapts to new conditions

Disadvantages

- Sequential model
- People standing/sitting for long are considered background

Methodology - Feature extraction - Segmentation

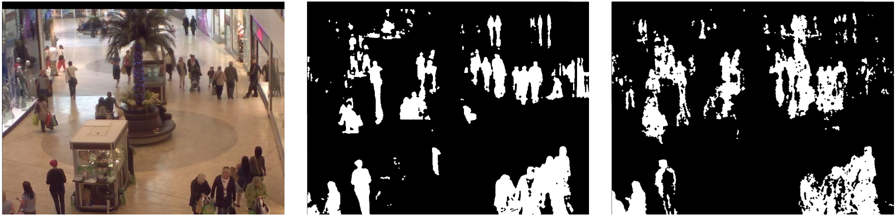
Background model:

Advantages

- Adapts to new conditions

Disadvantages

- Sequential model
- People standing/sitting for long are considered background



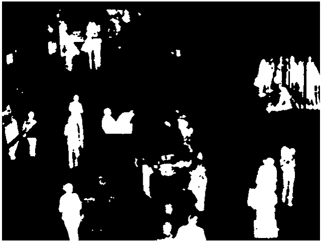
Left: thresholded difference; *right*: background model

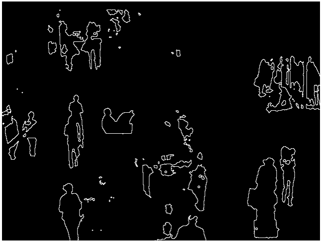
Methodology - Feature extraction

Low-level image features extracted (29 for each cell)

1. Foreground mask-based

- Foreground area
- Perimeter length: obtained by morphological operations
- Area/perimeter ratio: helps even if redundant
- Perimeter orientation histogram

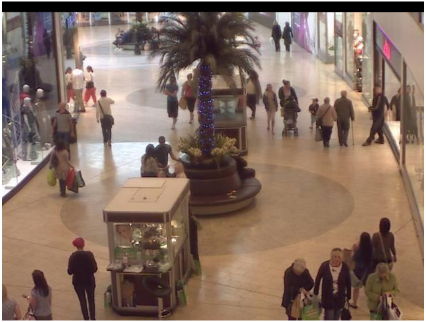




Methodology - Feature extraction

2. Edge-based (Canny)

- Number of edge pixels
- Edge orientation histogram
- Minkowski fractal dimension



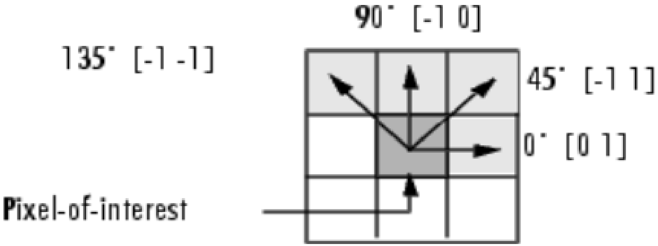
Methodology - Feature extraction

2. Edge-based (Canny)

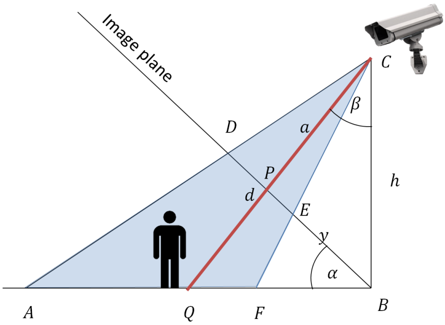
- Number of edge pixels
- Edge orientation histogram
- Minkowski fractal dimension

3. Texture-based (gray-level co-occurrence)

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

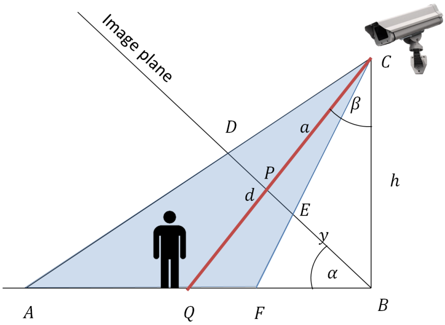


Methodology - Perspective correction



$$h = \overline{BC}, d = \overline{CQ}, y = \overline{BP} \quad (1)$$

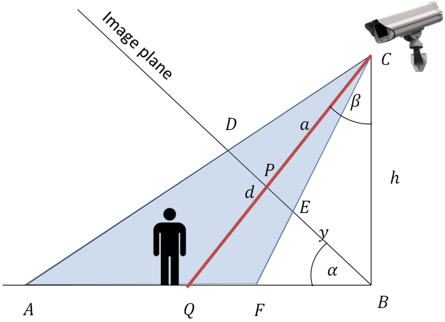
Methodology - Perspective correction



$$h = \overline{BC}, d = \overline{CQ}, y = \overline{BP} \tag{1}$$

$$d = \frac{h\sqrt{y^2 + h^2 - 2yh \sin \alpha}}{h - y \sin \alpha} \tag{2}$$

Methodology - Perspective correction

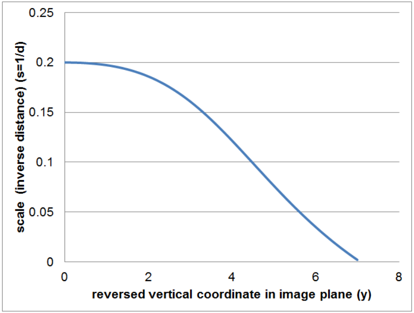


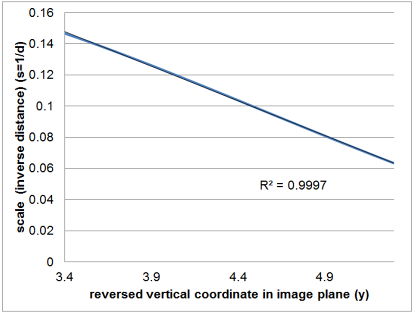
$$h = \overline{BC}, d = \overline{CQ}, y = \overline{BP} \tag{1}$$

$$d = \frac{h\sqrt{y^2 + h^2 - 2yh \sin \alpha}}{h - y \sin \alpha} \tag{2}$$

Too complicated, let's approximate!

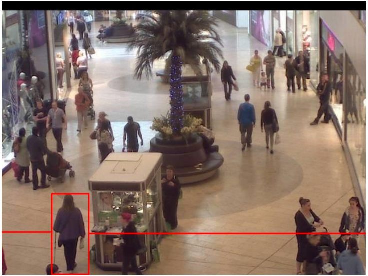
Plot of $\frac{1}{d}$





Methodology - Perspective correction

Inferring the linear approximation



Methodology - Perspective correction

Modify feature calculations with scale correction.

If the feature grows linearly with size

Example: perimeter

$$p = \sum_{(x,y):\ P(x,y)=1} \frac{1}{s(x,y)} \tag{3}$$

Methodology - Perspective correction

Modify feature calculations with scale correction.

If the feature grows **linearly** with size

Example: perimeter

$$p = \sum_{(x,y):\ P(x,y)=1} \frac{1}{s(x,y)} \tag{3}$$

If the feature grows **quadratically** with size

Example: foreground area

$$f = \sum_{(x,y):\ M(x,y)=1} \frac{1}{s^2(x,y)} \tag{4}$$

Methodology - Regression

Goal

Given a training set of the form

$$\begin{aligned} \mathbf{x}_i &= \left[\mathbf{z}_i^1, \mathbf{z}_i^2, \ldots, \mathbf{z}_i^K\right] \in \mathbb{R}^D \\ \mathbf{y}_i &= \left[u_i^1, u_i^2, \ldots, u_i^K\right] \in \mathbb{N}^K \end{aligned} \tag{5}$$

Estimate the output $\mathbf{y}_{new}$ for a new input $\mathbf{x}_{new}$.

Methodology - Regression

Goal

Given a training set of the form

$$\mathbf{x}_i = \left[\mathbf{z}_i^1, \mathbf{z}_i^2, \ldots, \mathbf{z}_i^K\right] \in \mathbb{R}^D$$
$$\mathbf{y}_i = \left[u_i^1, u_i^2, \ldots, u_i^K\right] \in \mathbb{N}^K \tag{5}$$

Estimate the output $\mathbf{y}_{new}$ for a new input $\mathbf{x}_{new}$.

Used algorithms

- Multivariate ridge regression
- Multivariate kernel ridge regression (a.k.a. Gaussian process regression)

Methodology - Regression

Goal

Given a training set of the form

$$\mathbf{x}_i = \left[\mathbf{z}_i^1, \mathbf{z}_i^2, \ldots, \mathbf{z}_i^K\right] \in \mathbb{R}^D$$
$$\mathbf{y}_i = \left[u_i^1, u_i^2, \ldots, u_i^K\right] \in \mathbb{N}^K \tag{5}$$

Estimate the output $\mathbf{y}_{new}$ for a new input $\mathbf{x}_{new}$.

Used algorithms

- Multivariate ridge regression
- Multivariate kernel ridge regression (a.k.a. Gaussian process regression)

Note: Vectors will be row vectors

Methodology - Regression

Assumption: noisy linear function

$$\mathbf{y} = \mathbf{x}\mathbf{W} + \mathbf{b} + \epsilon_{noise} \tag{6}$$

Multivariate ridge regression

$$\min_{\mathbf{W},\mathbf{b}} \left\{ \frac{1}{2} ||\mathbf{W}||_F^2 + C \sum_{i=1}^{N} ||\mathbf{y}_i - (\mathbf{x}_i \mathbf{W} + \mathbf{b})||_F^2 \right\} \tag{7}$$

Interpretations

Methodology - Regression

Assumption: noisy linear function

$$\mathbf{y} = \mathbf{x}\mathbf{W} + \mathbf{b} + \epsilon_{noise} \tag{6}$$

Multivariate ridge regression

$$\min_{\mathbf{W},\mathbf{b}} \left\{ \frac{1}{2} ||\mathbf{W}||_F^2 + C \sum_{i=1}^{N} ||\mathbf{y}_i - (\mathbf{x}_i\mathbf{W} + \mathbf{b})||_F^2 \right\} \tag{7}$$

Interpretations

- Punish large weights to avoid overfitting

Methodology - Regression

Assumption: noisy linear function

$$\mathbf{y} = \mathbf{x}\mathbf{W} + \mathbf{b} + \epsilon_{noise} \tag{6}$$

Multivariate ridge regression

$$\min_{\mathbf{W},\mathbf{b}} \left\{ \frac{1}{2} ||\mathbf{W}||_F^2 + C \sum_{i=1}^{N} ||\mathbf{y}_i - (\mathbf{x}_i\mathbf{W} + \mathbf{b})||_F^2 \right\} \tag{7}$$

Interpretations

- Punish large weights to avoid overfitting
- Maximum-a-posteriori (priors: regularization, noise: least-squares)

Methodology - Regression

Assumption: noisy linear function

$$\mathbf{y} = \mathbf{x}\mathbf{W} + \mathbf{b} + \epsilon_{noise} \tag{6}$$

Multivariate ridge regression

$$\min_{\mathbf{W},\mathbf{b}} \left\{ \frac{1}{2} ||\mathbf{W}||_F^2 + C \sum_{i=1}^{N} ||\mathbf{y}_i - (\mathbf{x}_i\mathbf{W} + \mathbf{b})||_F^2 \right\} \tag{7}$$

Interpretations

- Punish large weights to avoid overfitting
- Maximum-a-posteriori (priors: regularization, noise: least-squares)
- Minimize L_2 loss considering all weight possibilities together (Gaussians behave nicely)

Assumption: noisy linear function in a *transformed space*

$$\mathbf{y} = \phi(\mathbf{x})\mathbf{W} + \epsilon_{noise} \tag{8}$$

Multivariate ridge regression with basis functions

$$\min_{\mathbf{W}} \left\{ \frac{1}{2} ||\mathbf{W}||_F^2 + C \sum_{i=1}^{N} ||\mathbf{y}_i - \phi(\mathbf{x}_i)\mathbf{W}||_F^2 \right\} \tag{9}$$

$$\min_{\mathbf{A}} \left\{ \frac{1}{2}\mathrm{tr}\left(\mathbf{A}^\top \Phi\Phi^\top \mathbf{A}\right) + C \cdot \mathrm{tr}\left(\mathbf{A}^\top \Phi\Phi^\top \Phi\Phi^\top \mathbf{A} - 2\mathbf{Y}^\top \Phi\Phi^\top \mathbf{A} + \mathbf{Y}^\top \mathbf{Y}\right) \right\} \tag{10}$$

Methodology - Regression

Kernel trick

Avoid defining $\phi(\cdot)$, define directly $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})\phi(\mathbf{x}')^\top$

$$k_{RBF}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \cdot ||\mathbf{x}_i - \mathbf{x}_j||^2\right) \tag{11}$$

Multivariate kernel ridge regression

$$\min_{\mathbf{A}} \left\{ \frac{1}{2}\mathrm{tr}\left(\mathbf{A}^\top \mathbf{K}\mathbf{A}\right) + C \cdot \mathrm{tr}\left(\mathbf{A}^\top \mathbf{K}\mathbf{K}\mathbf{A} - 2\mathbf{Y}^\top \mathbf{K}\mathbf{A} + \mathbf{Y}^\top \mathbf{Y}\right) \right\} \tag{12}$$

$$\mathbf{A}^* = \left(\mathbf{K} + \frac{1}{2C}\mathbf{I}_{N\times N}\right)^{-1} \mathbf{Y} \tag{13}$$

$$\hat{\mathbf{y}}(\mathbf{x}) = \mathbf{k}(\mathbf{x})\mathbf{A} \tag{14}$$

Alternative interpretation: Gaussian process

Methodology - Regression - Data normalization

Until now all input dimensions (image features) are treated the same.

Methodology - Regression - Data normalization

Until now all input dimensions (image features) are treated the same.

- Linear ridge: common C value regularizes the weights acting on all dimensions

Until now all input dimensions (image features) are treated the same.

- Linear ridge: common C value regularizes the weights acting on all dimensions
- Kernel ridge: the RBF kernel treats all dimensions the same

Methodology - Regression - Data normalization

Until now all input dimensions (image features) are treated the same.

- Linear ridge: common C value regularizes the weights acting on all dimensions
- Kernel ridge: the RBF kernel treats all dimensions the same

Only makes sense if the input dimensions have the same scale:
Normalization!

Methodology - Regression - Data normalization

Until now all input dimensions (image features) are treated the same.

- Linear ridge: common C value regularizes the weights acting on all dimensions
- Kernel ridge: the RBF kernel treats all dimensions the same

Only makes sense if the input dimensions have the same scale:
Normalization!

- Calculate sample mean and variance for each dimension from the training set
- Transform the training set to have zero mean and unit variance
- Use the same transformation on each test input

Methodology - Implementation

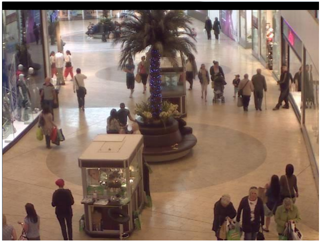
- C# on .NET 4.0
- EmguCV (OpenCV 2.4.9)
- Math.NET Numerics 2.5

Overview

4. Results

Results - Dataset

Mall dataset



- 2000 frames
- 640x480
- 2 FPS
- 13-53 people per frame

Results

- Training set: Frames 1-640 (22 minutes)
- Validation set: Frames 641-800 (5 minutes)
- Test set: Frames 801-2000 (40 minutes)

Results

- Training set: Frames 1-640 (22 minutes)
- Validation set: Frames 641-800 (5 minutes)
- Test set: Frames 801-2000 (40 minutes)

Avoiding bias:

- Experiment with settings and tune hyperparameters without touching the test set
- Train the selected method on the training+validation set
- Evaluate on the test set with no feedback

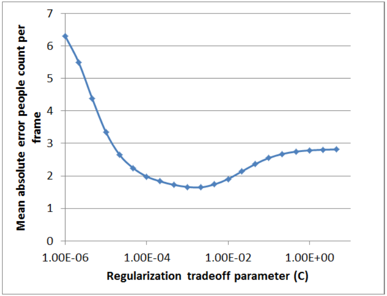
Results

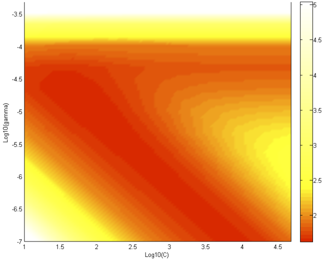
Evaluation metrics:

$$
\begin{aligned}
E_{sq} &= \frac{1}{M}\sum_{i=1}^{M}\left(\sum_{j=1}^{K}\hat{Y}_{ij} - \sum_{j=1}^{K}Y_{ij}\right)^2 \\
E_{abs} &= \frac{1}{M}\sum_{i=1}^{M}\left|\sum_{j=1}^{K}\hat{Y}_{ij} - \sum_{j=1}^{K}Y_{ij}\right| \\
E_{rel} &= \frac{1}{M}\sum_{i=1}^{M}\frac{\left|\sum_{j=1}^{K}\hat{Y}_{ij} - \sum_{j=1}^{K}Y_{ij}\right|}{\sum_{j=1}^{K}Y_{ij}}
\end{aligned}
\tag{15}
$$

Results

Hyperparameter tuning:





Results

Observations:

- Perspective correction has no effect (mean abs. error about 1% worse)
- 4×4 is the best grid subdivision
- Kernel ridge is somewhat better than linear ridge
- Scaling down to 320×240 improves the estimation

Results

Most promising combination:

- No perspective correction
- 4×4 grid subdivision
- Kernel ridge
- 320×240

Now let's check the performace on the test set!

Results

Comparison to the result in the paper (8×8 grid, 320×240 px)

Learning algorithm	E_{sq}	E_{abs}	E_{rel}
Linear ridge regression 8×8 (Chen et al.)	15.7	3.15	0.0986

Results

Comparison to the result in the paper (8×8 grid, 320×240 px)

Learning algorithm	E_{sq}	E_{abs}	E_{rel}
Linear ridge regression 8×8 (Chen et al.)	15.7	3.15	0.0986
Linear ridge regression 8×8 (own)	8.72	2.38	0.0768

Results

Comparison to the result in the paper (8×8 grid, 320×240 px)

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

Results

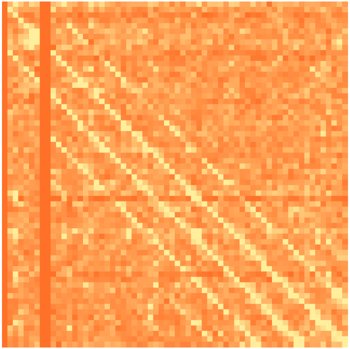
Comparison to the result in the paper (8×8 grid, 320×240 px)

Learning algorithm	E_{sq}	E_{abs}	E_{rel}
Linear ridge regression 8×8 (Chen et al.)	15.7	3.15	0.0986
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

Mean absolute error 29% smaller than in the paper.

Results

Information sharing between cells (8×8)



$$S_{pq} = \frac{\sum_{i \in \{\text{indices of features extracted from cell } p\}} |W_{iq}|}{\sum_{i=1}^{D} |W_{iq}|} \tag{16}$$

Overview

4. Summary

Summary

In short

- Reviewed requirements for the crowd counting problem
- Focused on the regression-based approach
- Used locality with grid and information sharing
- Extracted features (with background segmentation)
- Tuned hyperparameters of regression

Summary

In short

- Reviewed requirements for the crowd counting problem
- Focused on the regression-based approach
- Used locality with grid and information sharing
- Extracted features (with background segmentation)
- Tuned hyperparameters of regression

Room for improvement

- Use better features (tune parameters, add new features)
- Use better regression algorithm (neural networks, SVM, ...)
- Use features from previous frames

Thank you for your attention!