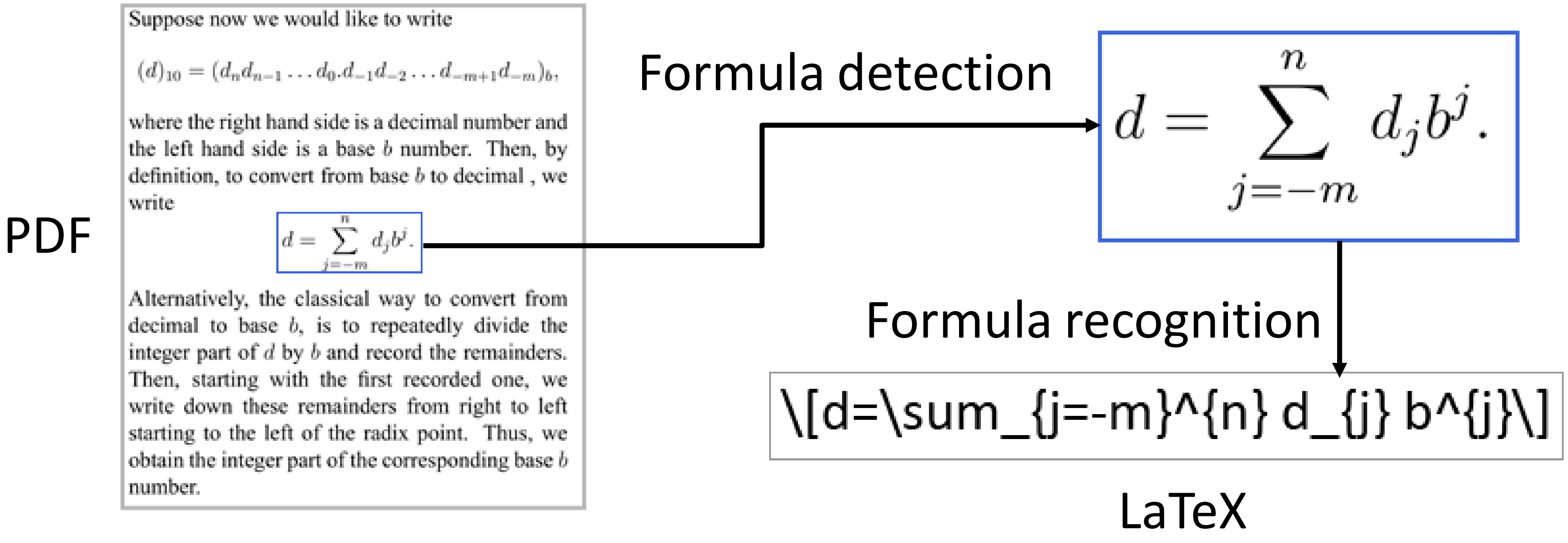
Machine Learning-based Location Detection of Mathematical Expressions in PDF

Example



- Goals:
  - Reuse in own LaTeX documents
  - Make accessible to screen readers
  - Search for mathematical content

Displayed vs. inline

b_i picks k random polynomials of the form:

$$f_{b_i,v_j}(x) = s + \alpha_1 x + \ldots + \alpha_{a_t} x^{a_t} \pmod p$$

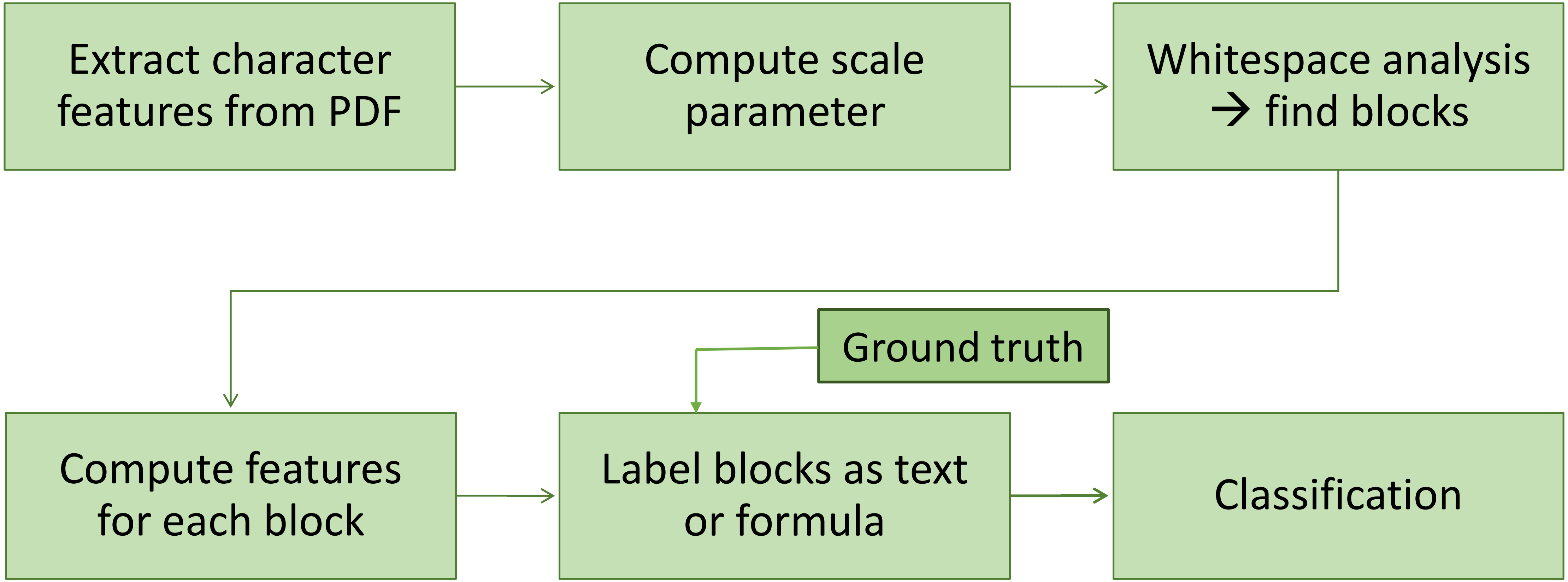
Where the coefficients are uniformly randomly chosen for each polynomial. If b_i is willing to bid at price v_j, then s is set to be ID_{b_i,v_j}, (i.e., b_i's ID for price v_j). Otherwise s is set to zero. b_i sends $f_{b_i,v_j}(\alpha_i)$ to a_i for all j, $1 \leq j \leq k$, and all i, $1 \leq i \leq \ell$.

- Displayed – isolated from other text
- Inline / embedded – inside a paragraph

Background

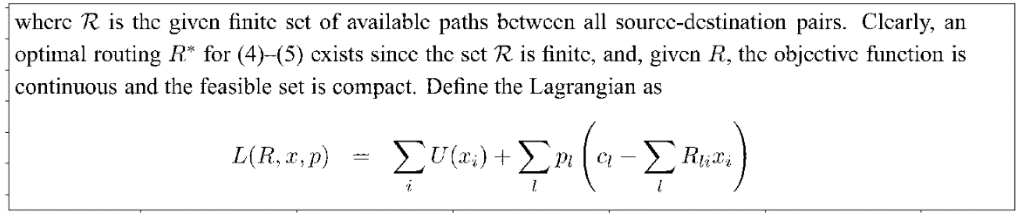
- Beginning: Optical Character Recognition (OCR)
- State of the art: OCR + PDF Parsers
- Formula image to LaTeX conversion → mostly solved
- Comparison between state of the art methods difficult

Method Overview

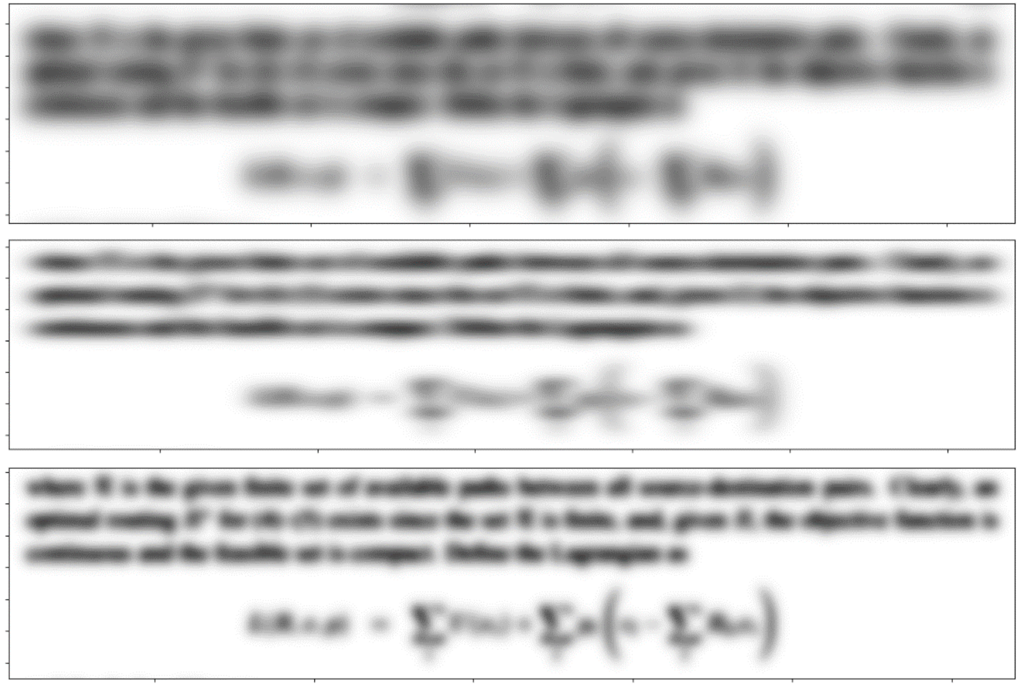


Whitespace Analysis – Gaussian Filter

scale = median of character sizes


where $\mathcal{R}$ is the given finite set of available paths between all source-destination pairs. Clearly, an optimal routing R^* for (4)–(5) exists since the set $\mathcal{R}$ is finite, and, given R, the objective function is continuous and the feasible set is compact. Define the Lagrangian as

$$L(R, x, p) = \sum_i U(x_i) + \sum_l p_l \left(c_l - \sum_i R_{li} x_i \right)$$



a) Original

b) σ = *scale* — too much

c) σ = (*scale*, 0.5 * *scale*) — ok

d) σ = 0.5 * *scale* — ok

Whitespace Analysis – Dark Pixel Sum

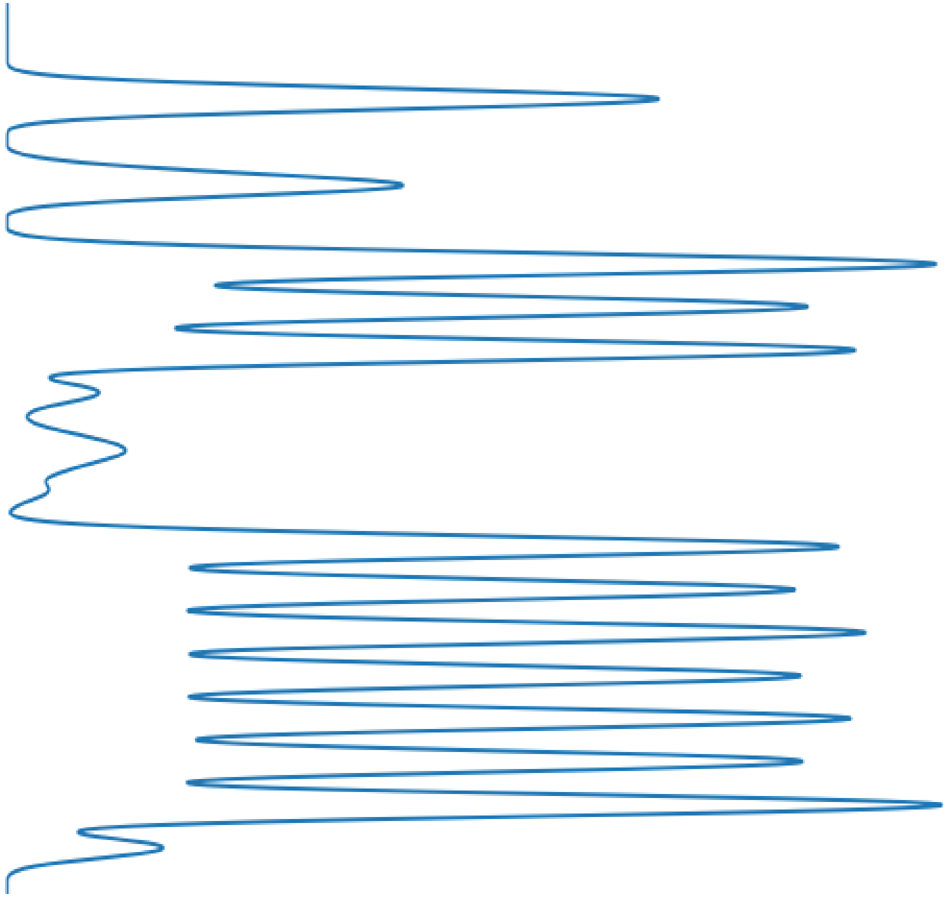
Suppose now we would like to write

$$(d)_{10} = (d_n d_{n-1} \dots d_0.d_{-1} d_{-2} \dots d_{-m+1} d_{-m})_b,$$

where the right hand side is a decimal number and the left hand side is a base b number. Then, by definition, to convert from base b to decimal , we write

$$d = \sum_{j=-m}^{n} d_j b^j.$$

Alternatively, the classical way to convert from decimal to base b, is to repeatedly divide the integer part of d by b and record the remainders. Then, starting with the first recorded one, we write down these remainders from right to left starting to the left of the radix point. Thus, we obtain the integer part of the corresponding base b number.

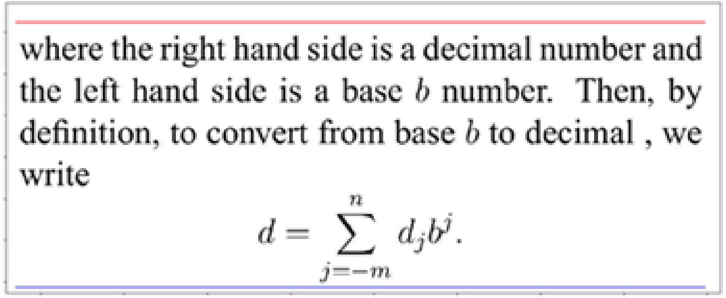


Whitespace Analysis –Threshold Problems

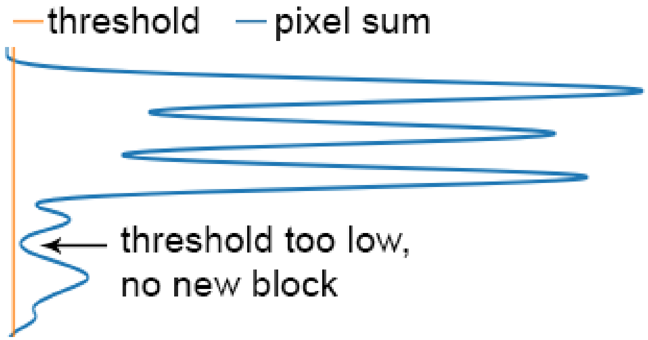
Block start

Block end

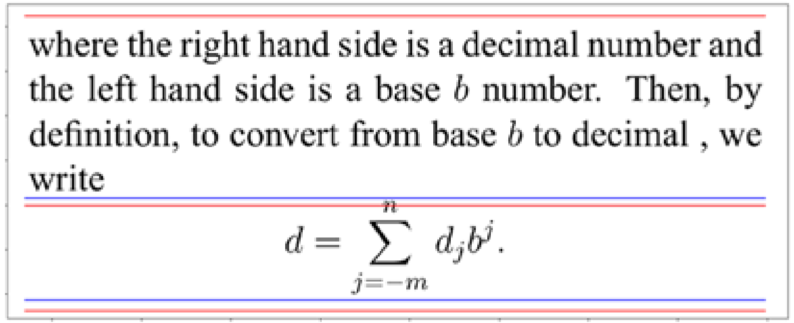
Problem:


where the right hand side is a decimal number and the left hand side is a base b number. Then, by definition, to convert from base b to decimal , we write

$$d = \sum_{j=-m}^{n} d_j b^j .$$



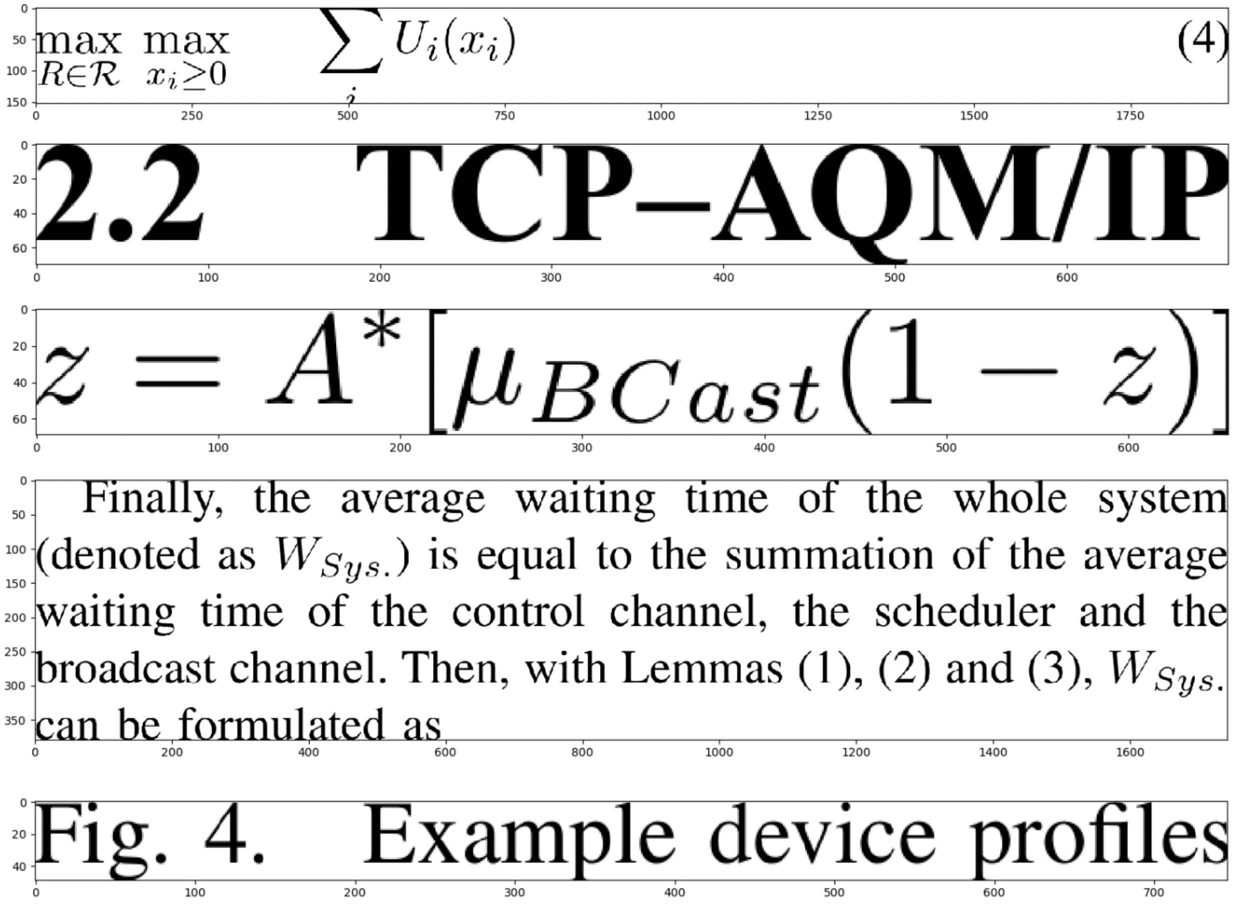
Solved:


where the right hand side is a decimal number and the left hand side is a base b number. Then, by definition, to convert from base b to decimal , we write

$$d = \sum_{j=-m}^{n} d_j b^j .$$

new block started because area above formula is empty

Features - Sparsity



a) 96.5%

b) 75.8%

c) 91.0%

d) 88.5%

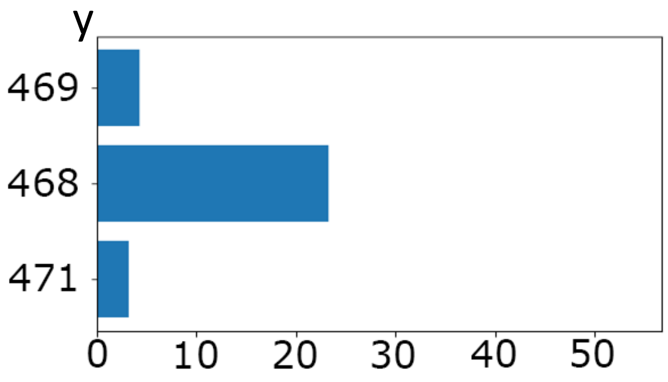
e) 96.3%

$$\text{sparsity} = \frac{\#\,white\ pixels}{total\ \#\,pixels}$$

Features – Horizontal Glyph Densities

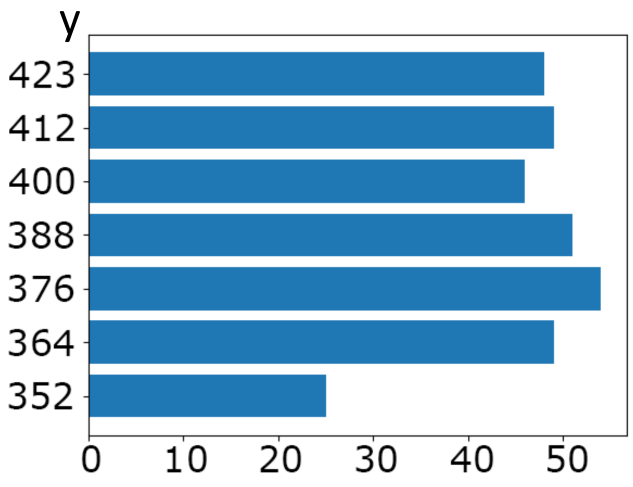
formula example

$$W_{Sys.} \quad = \quad W_{Ctrl.} + W_{Sche.} + W_{BCast} \qquad (1)$$



text example

This section shows the design of the proposed scheme ODB-QoS (standing for On-demand Data Broadcasting with QoS). An overview of scheme ODB-QoS is given in Section IV-A. The determination of the system state is given in Section IV-B. Finally, the proposed version decision policy and admission control scheme of scheme ODB-QoS are described in Section IV-C and IV-D, respectively.



- # characters on same y-coordinate

- Maximum
- Mean
- Standard deviation

Features – Permutation Entropy

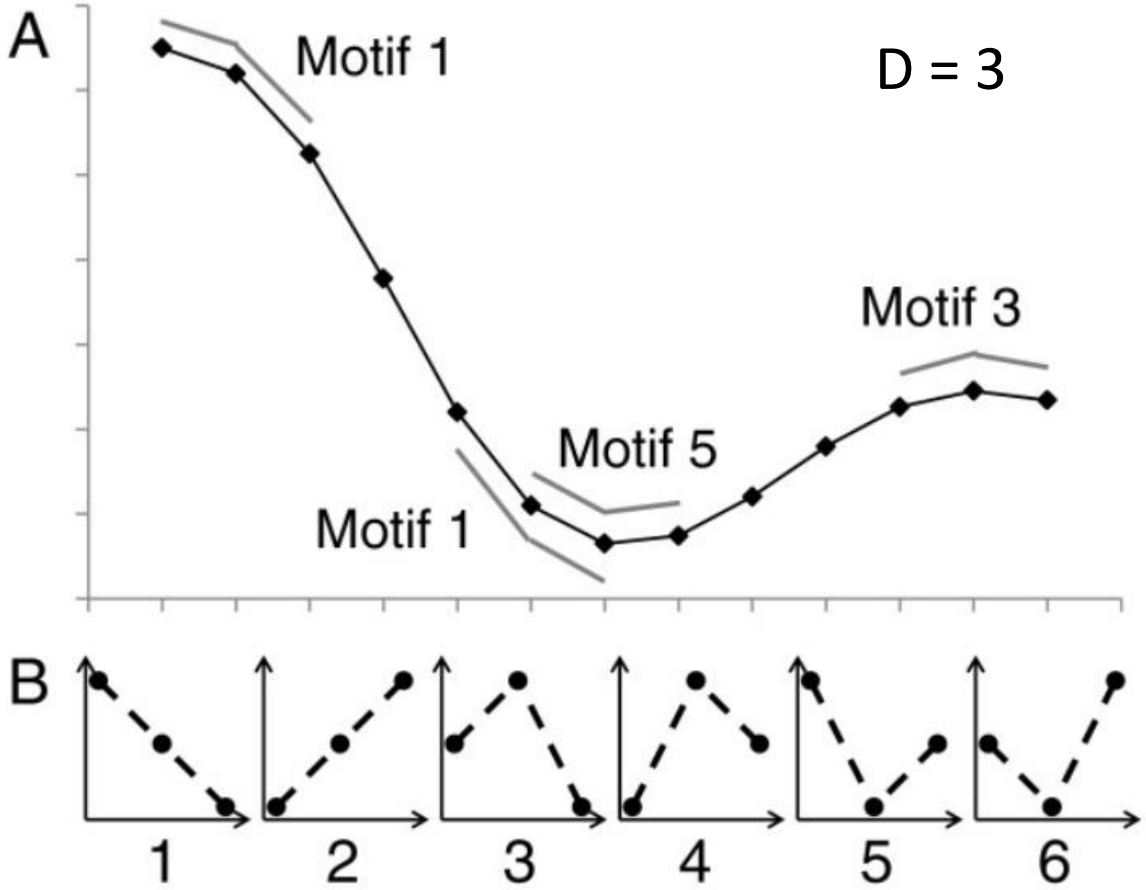


Image source: Hillen, Brian & Yamaguchi, Gary & Abbas, James & Jung, Ranu. (2013) 10.1186/1743-0003-10-97

- Estimate probability for ordinal patterns
- Compute entropy
- Parameters
  - step size τ
  - window size D
- $PE_{D,norm} = -\frac{1}{log_2 D!} \sum_{i=1}^{D!} p_i log_2 p_i$

Features – Font Information

- Idea: text blocks use standard font
  formula blocks use math font

1. Calculate relative frequencies for all fonts in document
2. For each block
   – Calculate most used font
   – Choose corresponding relative frequency from 1.

Features -Example

where $\mathcal{R}$ is the given finite set of available paths between all source-destination pairs. Clearly, an optimal routing R^* for (4)–(5) exists since the set $\mathcal{R}$ is finite, and, given R, the objective function is continuous and the feasible set is compact. Define the Lagrangian as

$$L(R, x, p) \quad = \quad \sum_i U(x_i) + \sum_l p_l \left(c_l - \sum_l R_{li} x_i \right)$$

Sparsity	Max	Mean	StD	Perm.Entropy	Font	Label
0.87	83	37.67	37.68	0.81	0.90	0 (text)
0.95	18	5.50	5.80	0.74	0.03	1 (formula)

Classification

But which method?

k-Nearest Neighbor

Boosting

Support Vector Machines

Bagging

Random Forest

Mulinomial Naive Bayes

Logistic Regression

LSTM

Naive Bayes

Gaussian Naive Bayes

Artificial Neural Networks

Recurrent NN

Convolutional NN

Classification

But which method?

k-Nearest Neighbor

Boosting

Support Vector Machines

Bagging

Random Forest

Mulinomial Naive Bayes

Logistic Regression

Naive Bayes

LSTM

Gaussian Naive Bayes

Artificial Neural Networks

Recurrent NN

Convolutional NN

Evaluation

- Data
  - Marmot Dataset: 400 PDF pages and corresponding image files
  - XML Ground truth files
- Test SVM, Random Forest, Naïve Bayes, Neural Network
- 3 selected approaches for comparison
- Measures
  - Precision, Recall, F1 score and MCC

Results

	Method	Precision	Recall	F1
My method	SVM	86.1	90.5	88.2
	Random Forest	91.6	76.0	83.0
	Naïve Bayes	58.9	91.6	71.7
	Neural Network	74.6	87.3	80.5
Literature methods	Deep Neural Network	-	-	93.4
	Font Setting Bayesian (FSB)	**99.4**	88.9	93.9
	Unsupervised Font Modeling	93.6	**99.4**	**96.4**

Inconsistent Results

	Precision	Recall	F1
FSB in FSB paper	99.4	88.9	93.9
FSB in Unsupervised Font Modelling paper	80.3	90.3	85.0
My method	86.1	90.5	88.2

- Documentation error?
- Limitation of method?

Conclusion

- My method
  - Best results: SVM, RBF kernel
- Literature methods better
- But: inconsistencies in results

- Conclusion: in-depth analysis & benchmark dataset needed

Thank you for your attention

Detailed Results SVM

SVM Kernel	Weights	Precision	Recall	F1
Linear	1:1	80.35	79.40	79.87
Linear	1:3	77.66	90.99	83.79
Linear	1:4	75.7	92.60	83.30
Linear	1:5	74.74	**93.56**	83.09
RBF	1:1	**92.22**	82.73	87.22
RBF	1:3	87.45	88.95	88.19
RBF	1:4	86.11	90.45	**88.23**
RBF	1:5	85.24	91.09	88.07
Polynomial	1:1	89.52	81.55	85.35
Polynomial	1:3	80.70	89.27	84.77
Polynomial	1:4	76.65	89.81	82.71
Polynomial	1:5	75.29	90.24	82.09
Sigmoid	1:1	75.82	84.44	79.9
Sigmoid	1:3	72.51	92.27	81.21
Sigmoid	1:4	73.28	90.34	80.92
Sigmoid	1:5	71.43	91.20	80.11

Detailed Results

- Random Forest (100 trees)

Depht limit	Precision	Recall	F1
10	**91.59**	**75.97**	**83.05**
15	90.01	73.50	80.92
None	89.38	72.21	79.88

- Naïve Bayes

Type	Precision	Recall	F1
Gaussian	**58.87**	91.64	**71.69**
Multinomial	44.51	**94.33**	60.48

Detailed Results Neural Network

Neurons	Precision	Recall	F1
6-6	73.93	87.03	79.93
18	**74.59**	87.33	**80.46**
24	74.30	**87.36**	80.30

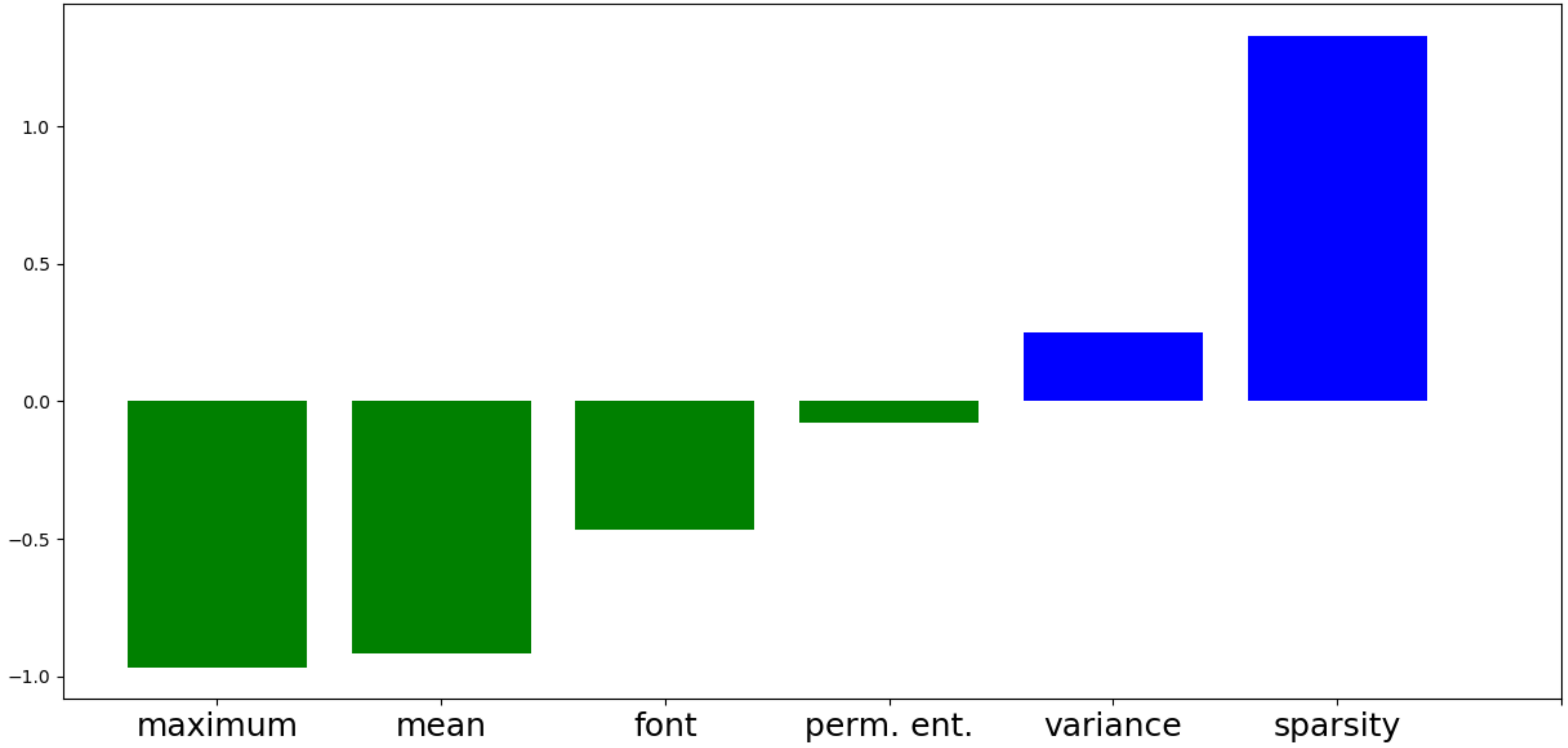
- Weight init. → random normal, mean 0, variance 1
- Hidden layers: ReLU   Output layer: sigmoid
- Training with Adam optimizer & binary cross-entropy loss
- Training for 100 epochs

Deep Learning Formula Detection

- Region proposal for formula candidates
- Convolutional Neural Network & Recurrent Neural Network
- Image features & text features
- Joint layer connects features

L. Gao, X. Yi, Y. Liao, Z. Jiang, Z. Yan, and Z. Tang. A deep learning-based formula detection method for pdf documents. 2017

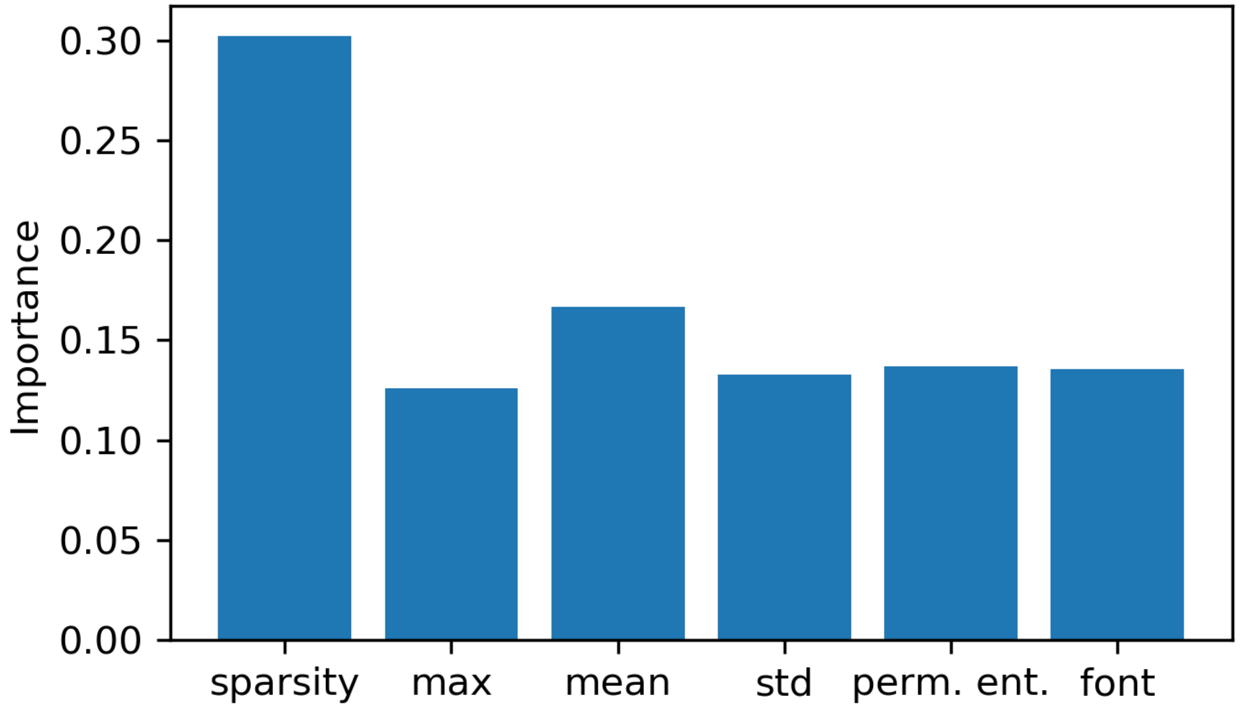
Feature Importance – Linear SVM



Text class

Formula class

Feature Importance – Random Forest



Valitation – Test Difference

SVM Kernel	1:1	1:3	1:4	1:5
Linear	+4.83%	+8.71%	+9.00%	+9.06%
RBF	+6.11%	+9.81%	+10.12%	+10.00%
Polynomial	+2.14%	+5.47%	+4.18%	+4.02%
sigmoid	+5.72%	+7.62%	+7.75%	+6.25%

Type	F1 change
Random Forest (10)	-1.26%
Random Forest(15)	-3.98%
Random Forest (None)	-4.58%
Gaussian NB	-7.45%
Multinomial NB	-1.37%
Neural Network (6-6)	+0.79%
Neural Network (18)	-0.11%
Neural Network (24)	+0.49%

Known Problems

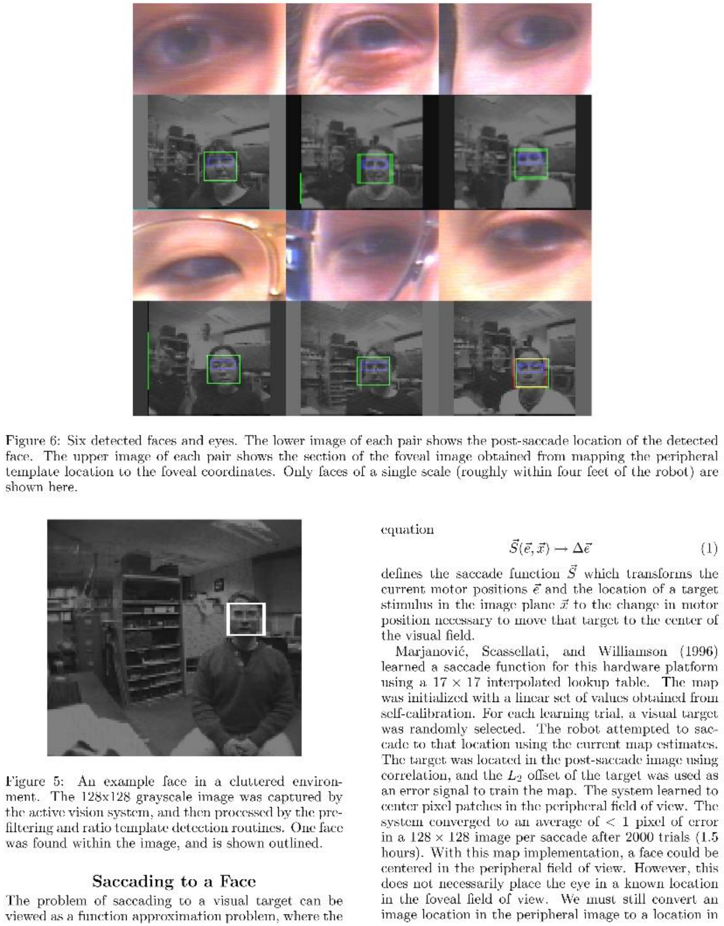


Figure 6: Six detected faces and eyes. The lower image of each pair shows the post-saccade location of the detected face. The upper image of each pair shows the section of the foveal image obtained from mapping the peripheral template location to the foveal coordinates. Only faces of a single scale (roughly within four feet of the robot) are shown here.

Figure 5: An example face in a cluttered environment. The 128x128 grayscale image was captured by the active vision system, and then processed by the prefiltering and ratio template detection routines. One face was found within the image, and is shown outlined.

Saccading to a Face

The problem of saccading to a visual target can be viewed as a function approximation problem, where the equation

$$\vec{S}(\vec{e}, \vec{x}) \rightarrow \Delta\vec{e} \qquad (1)$$

defines the saccade function $\vec{S}$ which transforms the current motor positions $\vec{e}$ and the location of a target stimulus in the image plane $\vec{x}$ to the change in motor position necessary to move that target to the center of the visual field.

Marjanović, Scassellati, and Williamson (1996) learned a saccade function for this hardware platform using a 17×17 interpolated lookup table. The map was initialized with a linear set of values obtained from self-calibration. For each learning trial, a visual target was randomly selected. The robot attempted to saccade to that location using the current map estimates. The target was located in the post-saccade image using correlation, and the L_2 offset of the target was used as an error signal to train the map. The system learned to center pixel patches in the peripheral field of view. The system converged to an average of < 1 pixel of error in a 128×128 image per saccade after 2000 trials (1.5 hours). With this map implementation, a face could be centered in the peripheral field of view. However, this does not necessarily place the eye in a known location in the foveal field of view. We must still convert an image location in the peripheral image to a location in

- Large figure / table above 2 columns
- No extra space around display expressions
- 1 font for whole document

Known Problems

$$= \left(\frac{\partial}{\partial X_1}(F_n X_n)\ \frac{\partial}{\partial X_2}(F_n X_n) \cdots \frac{\partial}{\partial X_n}(F_n X_n) \right) \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \leftarrow j - th\ \mathrm{row} \\ \\ \\ \\ \end{matrix}$$

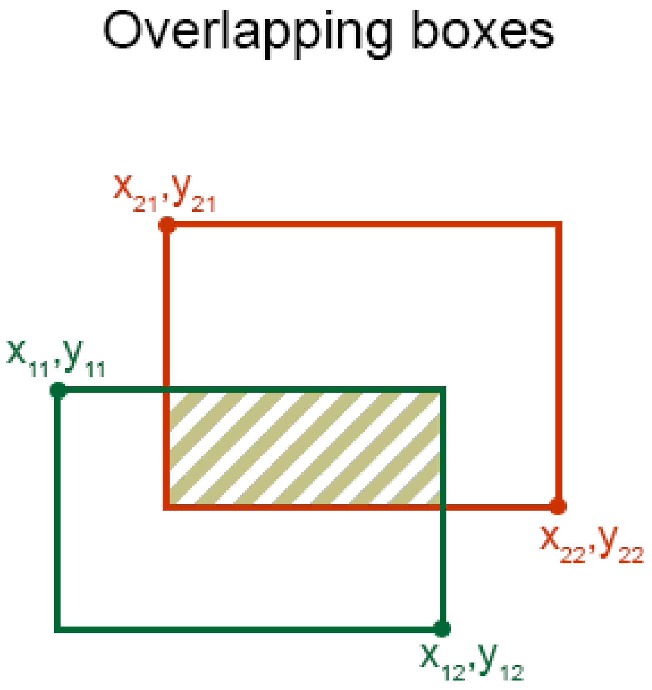
$$= \frac{\partial}{\partial \mathrm{X}_j}(X_n F_n) = X_n \frac{\partial F_n}{\partial X_j}$$

$$\uparrow \quad 0$$

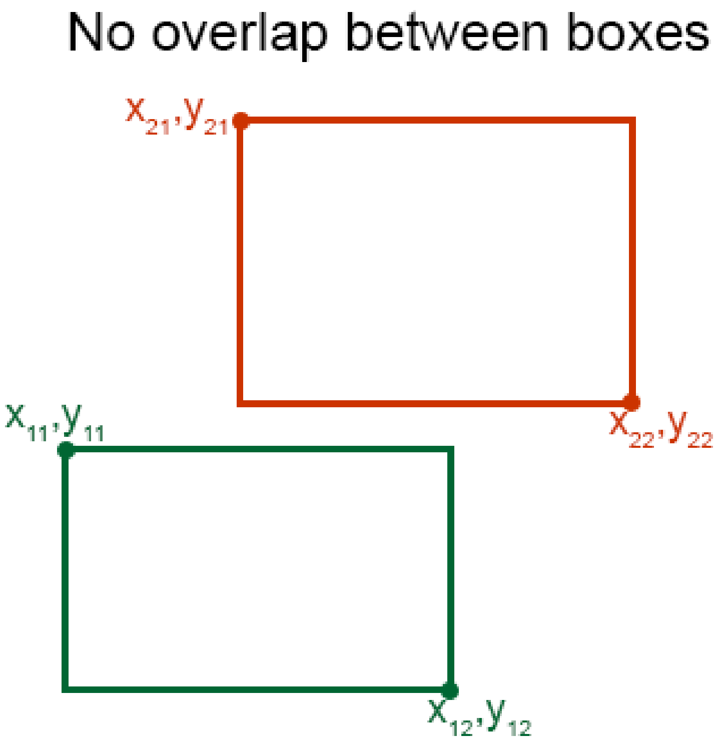
- Area above majority of formula empty

PDF Box vs. actual Character Size

Calculation of overlapping area



$y_{12} > y_{21}$ and $y_{22} > y_{11}$



$y_{12} > y_{21}$ but $y_{22} < y_{11}$