

Text Detection using Coarse detection and SVM Classification

Remco Gubbels
ICA-3820564
Universiteit Utrecht

Supervisor: Robby T. Tan, Remco Veltkamp & Zhouyu Fu

Abstract

In this paper a new approach to detect text in natural images is described using different detection methods. The end result will be that the text will be segmented from the image and can be used for different purposes. The approach is split up in two parts, a coarse detection step to extract patches from the image and a fine detection step that uses feature descriptors and a support vector machine in order to increase the precision of the coarse detection step. The methods used for the coarse detection are global threshold, mean threshold, Gaussian threshold, local binary pattern, maximum gradient difference filter and maximum difference filter. These methods are compared and the best results are used in combination with the fine detection. The feature descriptors used in the fine detection are Histogram of Oriented Gradients, Co-occurrence histogram of orientated gradients and local binary patterns. In order to increase the quality of the coarse detection a projection step is used. The approach performs on precision level worse than the current state-of-the-art methods, but has a better recall rate than most methods.

Contents

1	Introduction	3
1.1	Thesis Goals	3
1.2	Analysis	4
1.3	Outline	5
2	Related work	6
2.1	Outcome	9
3	Our approach	10
4	Coarse Detection	12
4.1	Local Binary Pattern	12
4.1.1	LBP-pixel	12
4.2	Pre-processing	13
4.2.1	Thresholding	13
4.2.2	Maximum gradient difference	13
4.2.3	Maximum difference	15
4.3	Bounding box	15
4.4	Projection	16
5	Classification using Support Vector Machine	18
5.1	Histogram of oriented gradients	18
5.2	Co-occurrence Histogram of Orientation Gradients	19
5.3	LBP-histogram	21
6	Experimental results and Evaluation	22
6.1	Experiments	22
6.1.1	Coarse Detection	24
6.1.2	Classification	24
6.1.3	Combination	24
6.2	Results of the experiments	25
7	Conclusion	32
7.1	Future work	33

Chapter 1

Introduction

The amount of images and videos that are currently on the world wide web and in databases is increasing on a daily basis. The amount of information that these images and videos harbour is astonishing, the main problem is how to find the information you are looking for. One of the main attributes of information for images and videos is the text that is located in it. For example the name of the person being interviewed, the score of a football match.

Detection of text in natural scenes has also received an increasing research attention in the last couple of years. This is due to the fact that it's crucial in the understanding of a scene. It can be used for multiple different applications like unmanned vehicle/robot navigation, living aids for visually impaired people etc. Optical character recognition on scanned images has been already solved to great success the recognition of text in natural images still has room for improvements as the recall rate and precision rate are still not good enough. The difference between optical character recognition and text detection is the fact that with text detection the target is to find the text in an image and it does not matter what it means. With optical character recognition the goal is to see the difference between each character and recognise if an A is an A and not a B for example. The differences in approach is that each character for each font has a certain style. This is not usefull for text detection as all the characters need to be recognised as text. That meas multiple shapes and angles can be text. The goal of text detection is to be able to segmentate the text in a natural image out of an image so it can be further used to retrieve information about the image.

1.1 Thesis Goals

The goal of the thesis is to get a text detection pipeline with a high precision and high recall, but will be used as target of the experiments.

Detecting text is a very delicate process as you have to consider a lot of different fonts, contrasts, font-sizes, colours and alignments. All these factors combined

make it very difficult to find one good solution that will fix it. Another problem is the fact that even we humans, we are considered the guideline to see if our algorithm performs the same way as our eyes and brain do, sometimes have problems detecting the text in images or videos.

In this thesis the problem is tried to be solved by using a few coarse detection approaches and a classifier for the fine detection. One of the methods used for the fine detection is an existing method which is originally used for character classification instead of text detection and for coarse detection methods are used that use a filtering tactic in order to only leave pixels that are part of a character.

Due to limited memory on the testing computer it is not possible to construct a feature descriptor that is insanely rich combined with a huge test data set. The training and test data set will be the set of ICDAR 2003 as this is used in most papers that are trying to tackle this problem.

1.2 Analysis

In order for humans to detect text there are some fundamental characteristics of texts, one of them is for example that if there is not a high contrast between the text and the background it will be hard to detect for a human.

Text has many sizes and font types, this makes it harder to detect as with my other detection problems a boundary can be set with certain techniques in order to filter out miss classifications. This is very difficult for a text detection algorithm as there are literally no boundaries.

Most words in signs and other places in the real world are written in an uniform colour as to make it easier to read for us. Some detection algorithms use this fact, however there is no rule that this is always the case making this not a usable way to detect everything.

In the English alphabet most characters have a stroke width that is almost the same across the entire character, however this is not the case in the Chinese alphabet where the stroke width is very different across one single character which makes it difficult to use stroke width as a universal solution to the problem.

In most cases a word is written a long a line and could be detected this way, however in images taken from nature we can not use this as easy as trying to find subtitles in a video clip for example. This is because a picture could be taken from a different angle or text could be written very playfully to attract customers for example. To not get false detection or miss characters it's easier to focus on single characters if you're not sure if the words and text in images will be lined up at all.

1.3 Outline

The remainder of this thesis will consist of the following few chapters, related work or background knowledge where some other methods are explained and why they don't succeed and Histogram of Oriented gradients and support vector machines will be explained as this will be used later on, theory about the used method explained in detail so that it can be implemented quickly when needed. Experimental results and the evaluation of this and a chapter with the conclusion and what can be done in the future.

Chapter 2

Related work

In the past several approaches in different researches have been used to solve the text detection problem on natural images. This chapter will discuss the strengths and weaknesses of four of these approaches.

All four approaches described below teach us something that will be used later in our own approach.

The first approach is a method where the stroke width for each pixel in an edge map is calculated. This approach is explained in the paper of Epshtein, B.[2]. The stroke width transform is a local image operator which computes per pixel the width of the most likely stroke containing this pixel. A stroke is defined as a continuous part of an image that forms a band of a nearly constant width.

The first step of the algorithm is to construct the edges using the Canny Edge detector. The second step is to use the gradient direction of each edge pixel p . The orientation of the gradient direction is followed until an edge point is detected. If the gradient of the other point is in the opposite direction of the gradient of p within margins then the new width, the distance between p and q , will be assigned to all pixels on the line between p and q that have a width larger than this new width (initially all points will have an infinite width. if a point has a width that is smaller than the calculated distance it means that it was set before. The smallest width of a point must be stored). If the direction is not close to the opposite direction it is ignored. This step results in an image where every pixel has a value assigned to it representing the width of the stroke it belongs to.

The next step is to group these pixels into chains representing character candidates. Two neighboring pixels are part of the same group if the value of the width assigned to them is relatively close. To guarantee both dark and bright characters are picked up, this method is executed twice: the first time with the positive gradient direction, the second time with a negative gradient direction.

The final step is to filter out the text areas with a few flexible rules:

The first rule is that a group that is a text area shouldn't have too much vari-

ance in stroke width inside the entire group.

The second rule is that the difference between the diameter of the connected components or group and its median stroke width is less than ten pixels.

The third rule is used to filter out edges surrounding letters, like the edge of a number plate surrounding the letter and numbers on it. It states that the bounding box of the connected components will include no more than two other connected components.

The fourth rule is that the height of a group must be between 10 and 300 pixels in order to be linked to different font sizes.

After these first set of rules another set of rules is applied on the remaining groups to find text lines instead of single pairs.

Two letters should have roughly the same stroke width and height ratio, and the distance between two letters should not exceed three times the width of the wider letter. If these rules do not apply to the two letters they are not part of the same text line.

Also a text line should consist of letters that are almost the same color as normally most texts found in images are the same color.

Pairs of letters can be connected into a chain if two pairs overlap and are in the same direction. Each chain should also at least contain three letters to be valid. The character candidates that pass these rules are considered text areas.

The above method is a very solid approach, however it can't be used on all different alphabets. The Chinese alphabet, for example, does not have the same consistency as the English alphabet when it comes to stroke width. Also, a lot of assumptions are made in order to filter the end results. The main strength of this approach is that it is shown that no use of feature descriptors is needed in order to get a good result.

Another approach of Muhammad Shehzad Hanif, the text hunter [14], uses two steps: text detection and fine localization of text regions. The text detector is a cascade of boosted classifiers trained using the AdaBoost algorithm. In contrary to current research in object detection, the method proposes to use heterogeneous (belonging to different families) weak classifiers in boosting paradigm. The weak classifiers used in this work belong to: generative and discriminant, linear and nonlinear, parametric and nonparametric families of classifiers. To encode textual information the method proposes two sets of features. One feature set is based on the contrast between foreground and background while the second feature set encodes shape and appearance of characters in a text region. These features are computed on an image (detection) window of small size which is further divided into 16 cells or blocks. Detection window size varies from 32x16 pixels to 288x144 pixels in 9 steps. Weak classifiers are trained in a feature space containing single and pair-wise features. The output of text detector is a set of detections at various scales together with confidence level of the detector. In the fine localization step, the detections in (quasi) horizontal direction are grouped and their confidence levels are added resulting in candidate text regions. Then connected components are extracted by applying

morphological operations on the canny edge map of each candidate text region. Next, connected components are validated using simple thresholds on features such as confidence level, edge density, height, width and aspect ratio. Finally, validated connected components are grouped to form text lines and/or words. The thresholds on features used for validation of connected components and grouping are learnt on training database using the genetic algorithm where the objective is to maximize the F-measure on the given training database. In case of scene text, features based on gradient information of connected components are also taken into consideration during validation.

This approach is similar to the first method in that it also uses assumptions in order to filter out results. The method shows us that feature descriptors can be used in order to diverge text from background.

The paper of Michael R. Lyu [4] explains a method that is designed to detect both low-contrast texts and high-contrast texts without being affected by different languages. Convert a video image to an edge map by using a color edge detector and continue with using a text area enhancement. This method highlights the text area by its density features by using two different operators. The first one that is used is an edge-strength-smoothing operator, this operator smooth out the edge-strength of the edge pixel by using values from the neighboring pixels as well. The second step consists of an edge-clustering-power operator reflecting the edge density around the smoothed pixel. These formulas are used to use these operators and end up with a good edge map for the next step.

The last step of the method is to define the text areas. This part consists of two phases: a coarse phase and a fine phase, both split up in a horizontally projection and vertically projection step. After these two steps the regions marked as text area are checked using simple rules for the average density and peak distribution to eliminate the last remaining non text-areas.

One of the strengths of this approach is that it is able to detect text of all languages. It is robust to different fonts, font sizes, background complexity and contrast levels. Also a coarse-to-fine detection schema is used in order to detect text. The major downside for the problem this paper tries to solve, is that it is focused on text detection in videos and uses frame to frame comparison. Also, non-horizontal texts cannot be detected.

The Neumann's Method [15] of Lukas Neumann and Jiri Matas based on an exhaustive effectively pruned search of the space of all character sequences, where individual characters are detected as Maximally Stable Extremal Regions (MSERs). The method exploits higher-order features and feedback loops to compensate errors in text detection. Text recognition is based on contour-based features and a multi-class SVM classifier, which is trained using synthetic data. The method takes into account multiple hypotheses in both text localization and recognition stages and selects the best hypothesis in the final stage using a simple language model.

This paper learns us that the support vector machine classifier is a good use for classification with the use of a blob detection algorithm.

2.1 Outcome

From these papers we can learn a few different things which are used as a basis for our approach, namely coarse-to-fine detection schema [4] can be used in order to find decent results, very cheap feature descriptors [14] can be used in order to do the fine detection and the different aspects of text which are described in section 1.2. Also it is shown that a filter based approach can be used for the coarse detection step as described in [4]. Support vector machine can be used as classification tool as used in [15]. Lastly it shows that the edges of the characters can be used in order to detect text in images as has been shown in [2].

Chapter 3

Our approach

In order to solve the problem described in the introduction a coarse-to-fine pipeline is introduced. The reason for using a coarse-to-fine pipeline is that by using a coarse step to start with a selection of interesting pixels can be found and easily segmented in order to create feature descriptors for the second part. The precision of the coarse detection will be low, hence why a second part is needed to increase this precision. In order to achieve better results, a pipeline is introduced which combines the coarse detection with the classification using support vector machines.

Support vector machines are being used in order it handles multidimensional feature descriptors very well and is a lot more precise compared to other methods like nearest neighbours. The first part of the pipeline consists of the pre-processing step and the coarse detection, this can be any of the methods in chapter 4 that will be explained next. All will be tested in order to see which one is the most suitable for this approach.

The coarse detection will deliver a list of bounding boxes, which represent a part of the image which should contain text. For each bounding box a feature descriptor is calculated using one of the methods mentioned in chapter 5.

Bounding boxes are used in order to segment the text parts instead of having to create a feature descriptor for every pixel which is way more expensive.

For the training phase a training set will be used and the model that results from using support vector machines will be saved and used for the test phase.

For the test phase each feature descriptor will be classified using the model of the training phase. All the bounding boxes that are classified as text will be transported back onto the image and the pixels that are part of these bounding boxes will be marked as text. In figure 3.1 the entire pipeline is visualized.

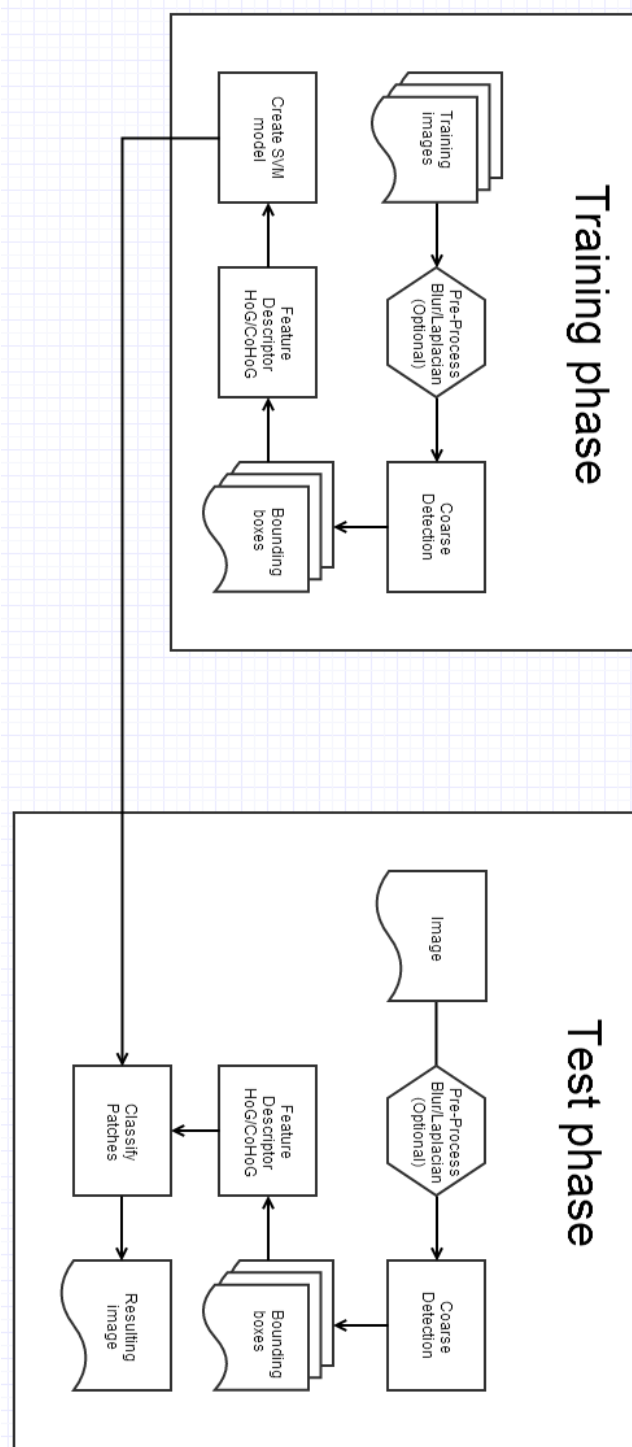


Figure 3.1: Pipeline

Chapter 4

Coarse Detection

As described in the introduction the first part of the thesis will be focused on coarse detection. A few methods will be tested in order to find the best one for our problem. Coarse detection will focus on a high recall but still improving the precision.

4.1 Local Binary Pattern

One of the aspects of text is that it has a unique pattern for each character for a certain font, one of the cheapest ways of constructing a feature descriptor for a window. For the text detection tool two different methods of local binary pattern will be used. Local binary pattern was first described in 1994 [18]. The reason why local binary pattern is tested is because it's a cheap and quick pixel based approach which can filter out unwanted pixels.

4.1.1 LBP-pixel

For each pixel in the image a pattern is calculated using a window of size $N \times M$. For each neighboring pixel it's checked if the gray-scale value is higher than the one that's currently focused, if it is it's marked as a one else a zero. This results in a binary number of size $N \times M - 1$ (Own pixel location is skipped). The binary number is then converted into a decimal number for easier use. This number represents this pixel. In the training phase each number is saved in a list for text-pixels and non-text pixels. When all pixels are done of the training set a trainings list is created by checking the amount of text-pixels representing a number and how many non-text pixels represent that number. In the test phase the resulting number is checked against this list in order to determine if it's a text or non-text pixel.

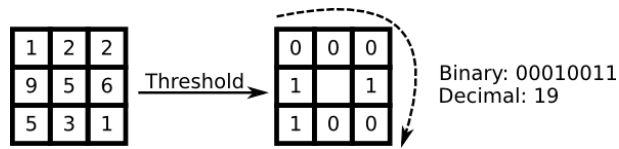


Figure 4.1: Local binary pattern

4.2 Pre-processing

4.2.1 Thresholding

Coarse detection can also be solved by using a form of thresholding on the image. The threshold will filter out unwanted noise and can solve the illumination problem by balancing out the contrast values. There are different methods to use a threshold, in this thesis there will be three different types of thresholding to test.

First the standard global thresholding is used, this can filter out unwanted noise, but doesn't help with the illumination problem, this is only used in order to show the difference. Global thresholding uses a set threshold value, if the gray-scale value is greater than the threshold it's kept, else it is removed from the image.

Secondly using a mean threshold, which uses a neighborhood block with size N , for every block with size N the mean is calculated of all the gray-scale values. The mean is then used as the threshold value for this pixel. This method also filters out unwanted noise, but also helps with the illumination problem.

As the last thresholding method, the Gaussian threshold is used. This method also uses a neighborhood block of N , but for every block with size N the threshold is calculated by using the weights of a Gaussian window.

In figure 4.2 it's easy to detect that the global threshold filters out some text pixels, while the mean and Gaussian thresholding enhances them.

4.2.2 Maximum gradient difference

One of the characteristics of text is that it has high contrast with the background, it doesn't flow over from the background, else it wouldn't be readable for humans. This results in a high difference of contrasts. This can be expressed in a high difference of gradients, as a gradient will be high if there is an edge and low if it's smooth. So basically we're expecting that a pixel that is part of a text has an edge nearby and a smooth area as most text also uses a very even value for the color of the text. By using a maximum gradient difference we can filter out a lot of background noise as background shouldn't have these characteristics.



Figure 4.2: Different thresholds applied to the same image

A sliding window is used to determine the maximum gradient difference of a pixel. This size of the window will be $1 \times N$ and N is determined by $\frac{\min(\text{height}, \text{width})}{M}$. Where M is a constant set by the user. For the entire image the gradient is calculated using a $[-101]$ mask this result in a matrix A , this than for every pixel this maximum gradient difference is calculated by obtained the maximum and minimum gradient in the window that is centered on the pixel which results in matrix B .

A pixel is classified as a text pixel if the gradient of that pixel is greater than the threshold T , T is calculated by using the following steps:

Calculate the average gradient of the image.

$$AGD = \frac{\sum_{i,j}^{h,w} A(i,j)}{h \times w}. \quad (4.1)$$

Where $h = \text{height}$ and $w = \text{width}$ of the image
Count how many pixels are greater than the average gradient.

$$NGD = \sum_{i,j}^{h,w} A(i,j) > AGD \quad (4.2)$$

The sum of the gradient differences of the windows.

$$SGD = \sum_{i,j}^{h,w} B(i,j) \quad (4.3)$$

The last step to construct the threshold is to take the sum of the gradient differences and divide it by the amount of pixels below the average gradient of the image.

$$T = \frac{SGD}{(h \times w) - NGD} \quad (4.4)$$

A pixel is classified as text as the gradient is greater than T . This is done for the X and Y direction of gradient and then combined to one result. Last step is to connect pixels that are close together a dilate mask is used in order to combine these pixels. This is to not filter out pixels that are a bit far from the edge or have some noise surrounding them

Two preprocessing steps have been used in order to see if this increases the quality of the result. The first one is a Gaussian blur in order to filter out noise in the image. Secondly a laplacian mask is used in order to get rid of the illumination problem that can occur.

4.2.3 Maximum difference

Another approach that is similar to the approach of the maximum gradient difference is taking the maximum difference in values that are the result of a laplacian preprocessing step. This will also show the contrast of a sliding window For each window the maximum difference is calculated and saved in a new matrix. In order to filter out the unwanted pixels with a low difference a thresholding method is being used. In the paper of [19] the Otsu Method is used as a way to determine the optimal value for the threshold. The Otsu method is a thresholding method designed for defining a threshold between two classes [13]. After the optimal thresholding according to Otsu is calculated the image is filtered and the background pixels are removed. The same dilation step as from the previous method is used for the same reasons.

4.3 Bounding box

By using a filtering method a lot of the pixels that are part of a character will disappear, as mostly the pixels that are on edges will survive. This can be fixed by using a bounding box mechanism to find all the contours that are in the image. This however will also pick up some noise, but will have a high recall rate. Which can be used in order to filter out certain areas and have less pixels to deal with in order to get a better precision rate.

In figure 4.3 you can see an example of bounding boxes extracted from an image.



(a) Complete image



Figure 4.3: Image with some of the bounding boxes extracted

4.4 Projection

In order to combine the coarse detection with a SVM classification the bounding boxes are used and send to the SVM. This however gives some problems. The first problem that arises is the fact that the classification methods try to compare bounding boxes. An image with a single character will result in a different feature vector than an image with multiple characters. Another problem is that the feature descriptor standardises the image in order to construct a feature vector of the same size. The second problem isn't normally a problem but due to the fact that there are multiple characters in a bounding box they'll be squished and a lot of data will be lost.

Here is where the projection comes into place. For each bounding box an edge detection is used in order to find the edges of the characters that have been



Figure 4.4: Before Projection

found. Due to the nature of how words are built up is that there will be a little room between the letters in order to make them readable. All the pixels that are marked as edges will be counted and put in a histogram, a bin for each row or column of the image. If there are gaps between the amount of pixels

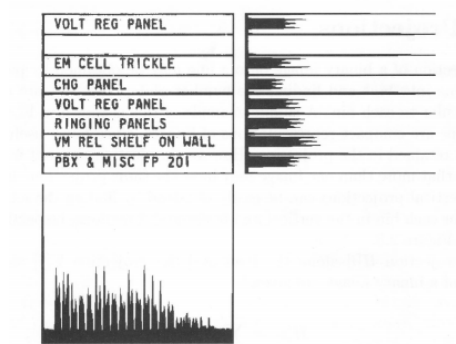


Figure 4.5: Horizontal and vertical projection of text

that are classified as edges the bounding box is split at that point. This will result in patches that only contain one character. Which we can pass on to the classification tools.



Figure 4.6: After Projection

Chapter 5

Classification using Support Vector Machine

5.1 Histogram of oriented gradients

There are a lot of different image descriptors out there that can be used as a classifier for certain objects in images, most commonly used one is scale-invariant feature transform (SIFT) [9] an alternative for SIFT is a Histogram of orientated gradients [16]. The main difference between SIFT and HOG is that SIFT is a local image descriptor and HOG a regional image descriptor. HOG has been introduced as a feature descriptor to track humans in images as described in [16].

The essential thought behind the Histogram of Oriented Gradient descriptors is that local object appearance and shape within an image can be described by the distribution of edge directions. The implementation of these descriptors can be achieved by compiling a histogram of gradient directions for each pixel based on a cell. For each pixel within this cell the gradient direction and magnitude is computed and added to a histogram which represents the pixel in question. For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block. This normalization results in better invariance to changes in illumination or shadowing.

The HOG descriptor has a few key advantages over other descriptor methods. Since the HOG descriptor operates on localized cells, the method upholds invariance to geometric and photometric transformations, except for object orientation. Such changes would only appear in larger spatial regions. Moreover, as Dalal and Triggs [16] discovered, coarse spatial sampling, fine orientation sampling, and strong local photometric normalization permits the individual body movement of pedestrians to be ignored so long as they maintain a roughly upright position.

5.2 Co-occurrence Histogram of Orientation Gradients

The theory of the main reference is based on one of the drawbacks of Histogram of Orientation Gradients, this drawback is the fact that there is no relation between the position of the pixels in the histogram. This can result in two histograms that are similar for two images that aren't similar apart from that they have the same amount of orientation and magnitude pixels as each other.

Co-occurrence histogram of oriented gradients tries to solve this problem, by introducing a co-occurrence matrix based on an offset. If an offset of (0, 0) is used it is a standard Co-occurrence histogram, this will be shown later. Co-HOG captures the spatial information by counting frequency of co-occurrence of oriented gradients between pixel pairs. The frequency of co-occurrence of oriented gradients is captured at each offset via co-occurrence matrix. The co-occurrence matrix at a given offset (x, y) is given by:

$$H_{x,y}(i, j) = \sum_{(p,q) \in B} \begin{cases} 1 & \text{if } O(p, q) = i \text{ \& } O(p+x, q+y) = j \\ 0 & \text{otherwise} \end{cases} \quad (5.1)$$

The amount of co-occurrence matrix depends on the set offset, by an offset of (2, 2) 24 co-occurrence matrix will be formed, for every offset combination possible from -2 to 2 in x and y direction. The (0, 0) co-occurrence matrix is normally ignored as it doesn't add enough information. The entire feature vector is constructed by vectoring and concatenation the co-occurrence matrix of all blocks of the image that is being studied.

This approach doesn't vector in the fact that some gradients are stronger than other gradients. A weighting mechanism based on the gradient magnitude where bi-linear interpolation is employed to vote between two neighboring orientation bins.

$$\begin{aligned} H(\theta_1, \theta_3) &= H(\theta_1, \theta_3) + M_1 \left(1 - \frac{\alpha - \theta_1}{\theta_2 - \theta_1}\right) + M_2 \left(1 - \frac{\beta - \theta_3}{\theta_4 - \theta_3}\right) \\ H(\theta_1, \theta_4) &= H(\theta_1, \theta_4) + M_1 \left(1 - \frac{\alpha - \theta_1}{\theta_2 - \theta_1}\right) + M_2 \left(\frac{\beta - \theta_3}{\theta_4 - \theta_3}\right) \\ H(\theta_2, \theta_3) &= H(\theta_2, \theta_3) + M_1 \left(\frac{\alpha - \theta_1}{\theta_2 - \theta_1}\right) + M_2 \left(1 - \frac{\beta - \theta_3}{\theta_4 - \theta_3}\right) \\ H(\theta_2, \theta_4) &= H(\theta_2, \theta_4) + M_1 \left(\frac{\alpha - \theta_1}{\theta_2 - \theta_1}\right) + M_2 \left(\frac{\beta - \theta_3}{\theta_4 - \theta_3}\right) \end{aligned} \quad (5.2)$$

Where α and β are the orientation of the two pixels that are being looked at and M_1 and M_2 are the magnitudes. θ_1 and θ_2 are the center of the bin to the left and right of the α where is θ_3 and θ_4 are these for β
Images etc

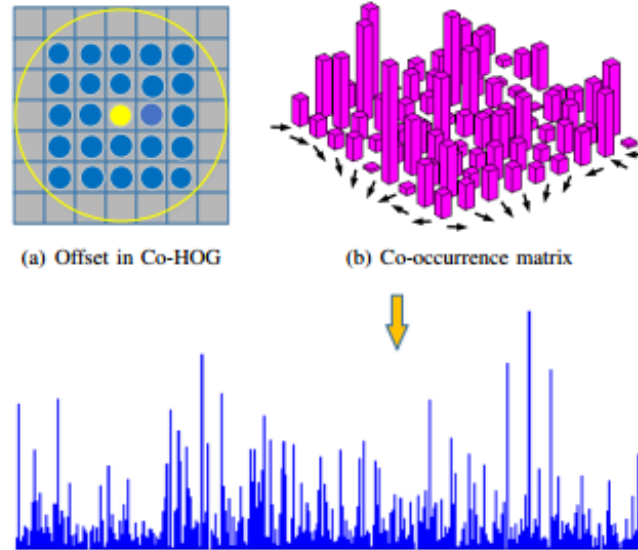


Figure 5.1: Vectorization

This could result in a pixel with a very small gradient magnitude value can have a fair large weight if it's pair pixel has a large gradient magnitude value. To avoid these kind of situations, pixel pairs where at least one of the gradient magnitudes is very small are filtered out and not used in the feature vector. This threshold that is used to filter out low pixel pairs has to be manually set and tested in order to find the right value.

Gradient computation One of the two main steps of computing the Histogram of Orientated Gradients is computing the orientation and magnitude of the each vector. Before this can be done the horizontal and vertical gradient components need to be computed using two easy vectors on the image. The

$$[-1, 0, 1] \quad [-1, 0, 1]^T \quad (5.3)$$

Figure 5.2: Where the left vector is the horizontal one and vertical on the right.

resulting maps will be used to compute the orientation and magnitude of each pixel in the image. To be able to compute the orientation the standard form is used, where θ is the orientation angle and x and y are respectively the horizontal and vertical gradient.

$$\theta = \left(2 \arctan \frac{y}{\sqrt{x^2 + y^2} + x} \right) * \left(\frac{180}{\pi} \right) \quad (5.4)$$

The magnitude can be used in order to display big changes in the gradient instead of small changes, the magnitude M ,

$$M = \sqrt{x^2 + y^2} \quad (5.5)$$

Binning

The resulting histogram consists of 8 bins which are based on 45 degree each ($[0 \rightarrow 44]$ $[45 \rightarrow 89]$... $[315 \rightarrow 359]$) and combine all the magnitudes of the vectors that have a related orientation as bin value. By combining all the histograms of the surrounding cells a final normalized histogram is constructed and can be used to compare different images.

5.3 LBP-histogram

Lastly a method based on local binary pattern is used to create a histogram for a particular window with size K . For each pixel inside this window the local binary pattern is calculated as described above, but this time a histogram is formed. This means that a histogram with bins for each value is created and if a decimal value is the result of the pattern it's added into the bin and for each window a normalized vector is created. For each window of K a histogram is formed. All the resulting histograms are concatenated into one feature histogram which can be used by a support vector machine to get classified.

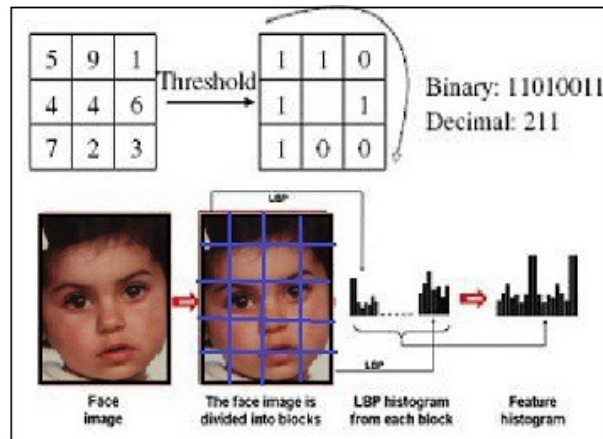


Figure 5.3: Local binary pattern histogram

Chapter 6

Experimental results and Evaluation

6.1 Experiments

In order to test the theories described in the previous chapters two test sets and two training sets are needed. For all the three approaches the ICDAR 2003 dataset is used as this set is mostly used in other papers related to this research. This will make it easier to compare the results between papers. This data set consists of random images in nature that contain some text. The font-types and sizes differ a lot making it a well rounded set. It consists of 249 training images and 250 test images.

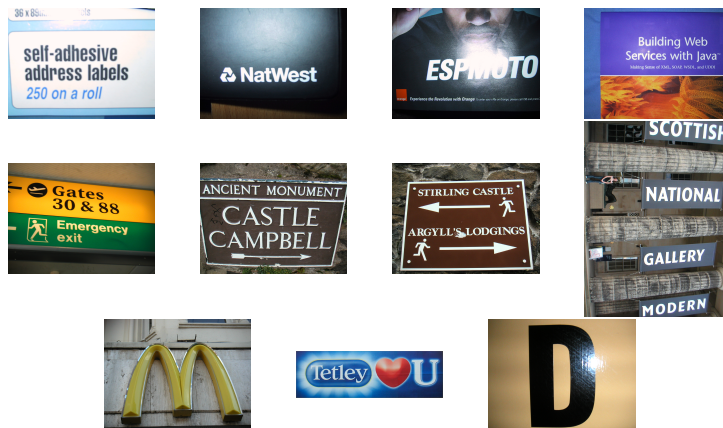


Figure 6.1: Examples from the data set

Analysis of the dataset used

The data set might be very broad with lots of different typesets and sizes, however some of the files are missing ground truth information, mixed interpretation of punctuation and special characters as part of words and the bounding boxes around words are not tight. This will all influence the results with more false positives that are actually not part of a character and some false negatives of pixels that are actually part of a character but not marked as one by the data set.

In order to see if there is any hope to classify text from non-text patches with the use of Histogram of oriented gradients, Co-occurrence histograms of oriented gradients and Local binary pattern a small training and test set is formed by using 200 character images from ICDAR 2003 and 200 random non-text images. The test set consist of 50 characters images and 50 non-text images in order to see if this approach will work on a larger scale.



Figure 6.2: Examples from the data set of characters used



Figure 6.3: Examples from the data set of non text images used

In order to test the text detection for all approaches a precision and recall formula is used. Two different formulas will be used. This is as there will be two different experiments for the coarse detection. Firstly precision and recall are defined as:

$$\begin{aligned}
 Recall &= \frac{\text{Num}(\text{Correct pixels})}{\text{Num}(\text{All text pixels})} \\
 Precision &= \frac{\text{Num}(\text{Correct pixels})}{\text{Num}(\text{Pixels marked as text})} .
 \end{aligned}
 \tag{6.1}$$

This means the the precision represents the ratio of area of the true positive extracted text regions to area of the detected regions and recall represents the ratio of area of the true positive extracted text regions to area of the ground truth regions.

Also the f-measure is defined as the combination of the precision and the recall by the harmonic mean.

6.1.1 Coarse Detection

Coarse detection is done using the approaches described above and result in a binary image with text pixels and non-text pixels. For each approach different parameters can be set and these will be tested with different values in order to find the best one. For the thresholding this will be tested by using different window sizes, with LBP the size of N and M will be adjusted to find a better result and for the maximum difference and maximum gradient difference a different value for M is being used in order to get different window sizes. For the coarse detection the hypotheses is that this will give a high recall but not a low precision rate as also a lot of other edge pixels will be picked up seeing as it is a coarse filtering.

6.1.2 Classification

For the classification two different experiments will be conducted. The first one is by using the small test set in order to compare images of a single character with non-text images in order to see if the accuracy of this approach is any decent and hopeful to use in the combination experiment with the coarse detection. Secondly a sliding window will be used in order to construct feature descriptors for each window and if 90% of the window is part of text it's marked as a text window. This will be tested with different sizes of images of the same set as used for the coarse detection. The hypotheses is that the order of accuracy will be co-occurrence histogram of oriented gradients followed by normal histogram of oriented gradients and as last local binary pattern as the richness of this feature descriptor is the lowest.

6.1.3 Combination

For the complete pipeline only the big set will be used, but also with different sizes of images. Also the use of vertical and horizontal in order to split up the bounding boxes found after the coarse detection will be experimented with in order to see if this improves the results.

6.2 Results of the experiments

As described in the last section for the coarse detection multiple parameters were used. For the full results check appendix A.

Table 6.1: Best result of each different coarse detection

Method	Precision	Recall	F
Baseline	13%	100%	23.0%
None	22.39%	70.53%	33.99%
Global	17.41%	94.07%	29.38%
Mean	19.31%	87.03%	31.60%
Gaussian	20.11%	86.98%	32.66%
LBP	12%	27%	16.61%

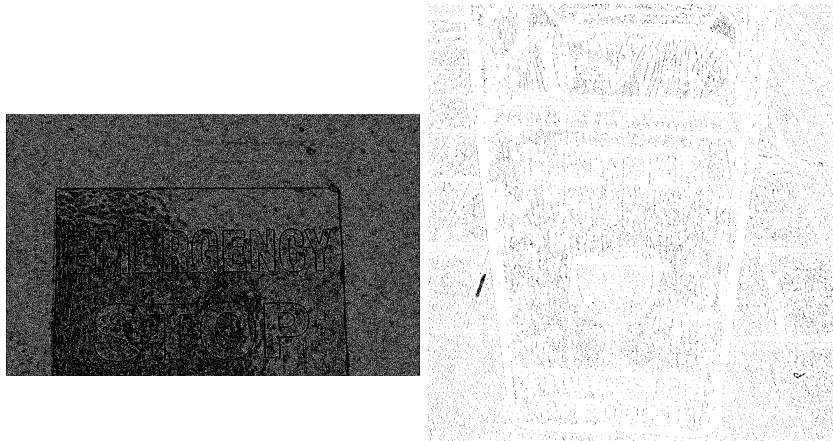
From these results it is visible that the local binary pattern has a very low f-measure. This is due to the fact that the pattern covers a very small area of the image which will not be able to get a lot of differences in a pixel part of text and a pixel not part of a text area. This can be seen in figure 6.4

Due to the fact that the results are not good the local binary pattern coarse detection will not be used in the following combination tests.

In figure 6.5 it is clear that there is a difference between all the different coarse detection approaches on a clear image. Global shows that it can be very precise on clear images, but in figure 6.6 it is clear that it is very difficult for the global threshold to be precise if the image has more noise on it. The mean and gaussian threshold result in smaller patches, but also pick up a lot more noise. The maximum gradient difference also picks up smaller patches and some noise. The maximum difference is very precise, but this also results in too very small patches. These patches reduce the amount of recall due to the fact of how the precision/recall is determined.

There are also plenty of images that have a patch covering the entire picture, due to the fact that there are no restrictions on patches in an image these can not be filtered. The image could be just of a single character. This however increases the recall by a lot and also reduces the precision rate with a lot. This happens mostly by the global threshold due to the fact that a lot of pixels at the edges for example can be above the globally defined threshold.

Before the combination is being used a small test is done with 200 images with characters and 200 images with no text in it too see if the feature descriptors are promising. All the three feature descriptors that have been described in chapter 5 will be used in order to see if there is any use to use them in the overall pipeline. This has been done because of the fact that the generated patches from coarse detection will be of lesser quality compared to the training set here. Also the parameters for HoG and LBP need to be tested. This will be



(a) Local Binary Pattern

Figure 6.4: Local binary pattern

done at the same time.

Method	Accuracy
HoG	91%
CoHoG	89%
LBP	61%

Table 6.2: Comparing text with non-text feature descriptors using a 200/200 training set

From the results we can see that the Histogram of Oriented Gradient and Co-occurrence Histogram of Orientation Gradients give decent enough results to continue research. The same feature descriptor has been used on a smaller training set and it showed it has less accuracy. This can be explained due to the fact that the feature descriptors require a training set of decent size. The training set used in the overall pipeline will be bigger than the training set here. Due to the low results of the local binary pattern feature descriptor this will not be used in the overall pipeline as this will not result in any decent precision.

For detailed results look at table A.9 and table A.10 in the appendix. If

SVM	Coarse Detection	Projection	Precision	Recall	F
HoG	Global	No	33.36%	62.36%	43.46%
CoHoG	Mean	Yes	35.98%	57.94%	44.39%

Table 6.3: Best results of using different coarse detection with SVM based on f-measure

compared with the results of the maximum difference coarse detection it's visible that the precision went up and the recall down. As expected. However it was expected before hand that the projection would increase the precision, this however has been the case for most coarse detections, but not for all. However the recall rate also dropped by a lot. Especially on the maximum difference coarse detection where there was an increase of an average 6% for both feature descriptors, but also a massive drop towards the 14% recall. This shows that the projection method is very poorly in combination with the feature descriptors. Making it easier to detect the correct patches, but had filtered out too many pixels in order to create decent results.

The average accuracy of HoG was 63% and for CoHoG the average accuracy was 61% without using the projection step on the classification of the patches. With the projection step the average accuracy of HoG was 71% and for CoHoG 69%. This results in the poor precision of the overall pipeline. This does give a better result precision wise than purely the coarse detection.

In figure 6.7 it is shown that the HoG feature descriptor works very well with big characters, but also classifies noise as text. The Co-HoG feature descriptor classifies very precisely, but also filters away bigger characters in images and just correctly identifies small parts of these characters.

It can also be seen that the projection step filters out some noise patches, but also filter out some characters.

Compared to some state-of-the-art methods this approach does give a decent result. The data set has not been optimised compared to the results of the last ICDAR 2013 test results [17]. From these results it is shown that the other

Method	Recall	Precision	F
USTB TexStar	66.45%	88.47%	75.89%
Text Spotter	64.84%	87.51%	74.49%
CASIA NLPR	68.24%	78.89%	73.18%
Text Detector CASIA	62.85%	84.70%	72.16%
I2R NUS FAR	69.00%	75.08%	71.91%
I2R NUS	66.17%	72.54%	69.21%
TH-TextLoc	65.19%	69.96%	67.49%
Text Detection	53.42%	74.15%	62.10%
Our approach	57.94%	35.98%	44.39%
Baseline	34.74%	60.76%	44.21%
Inkam	35.27%	31.20%	33.11%

Table 6.4: Caption

approaches have a better score on precision and on recall rate. However if not focused on F-measure a lot of our results as been shown in Appendix A reflect that a better recall rate is generated, but at the cost of precision. This can be

explained due to the fact that our approach uses patches and if the patch is classified as text, it will be completely marked as text. Most other methods use pixel based classification, which will be more precise, but also filter out wanted pixels. If compared to other methods in this table almost all of these methods have a very strong coarse detection method, which extracts almost all of the text parts. This is not the case for our approach, which does have a high recall it still excludes some parts. Also no extra step in between the coarse detection and fine detection is used as many other methods use an elimination step in between.

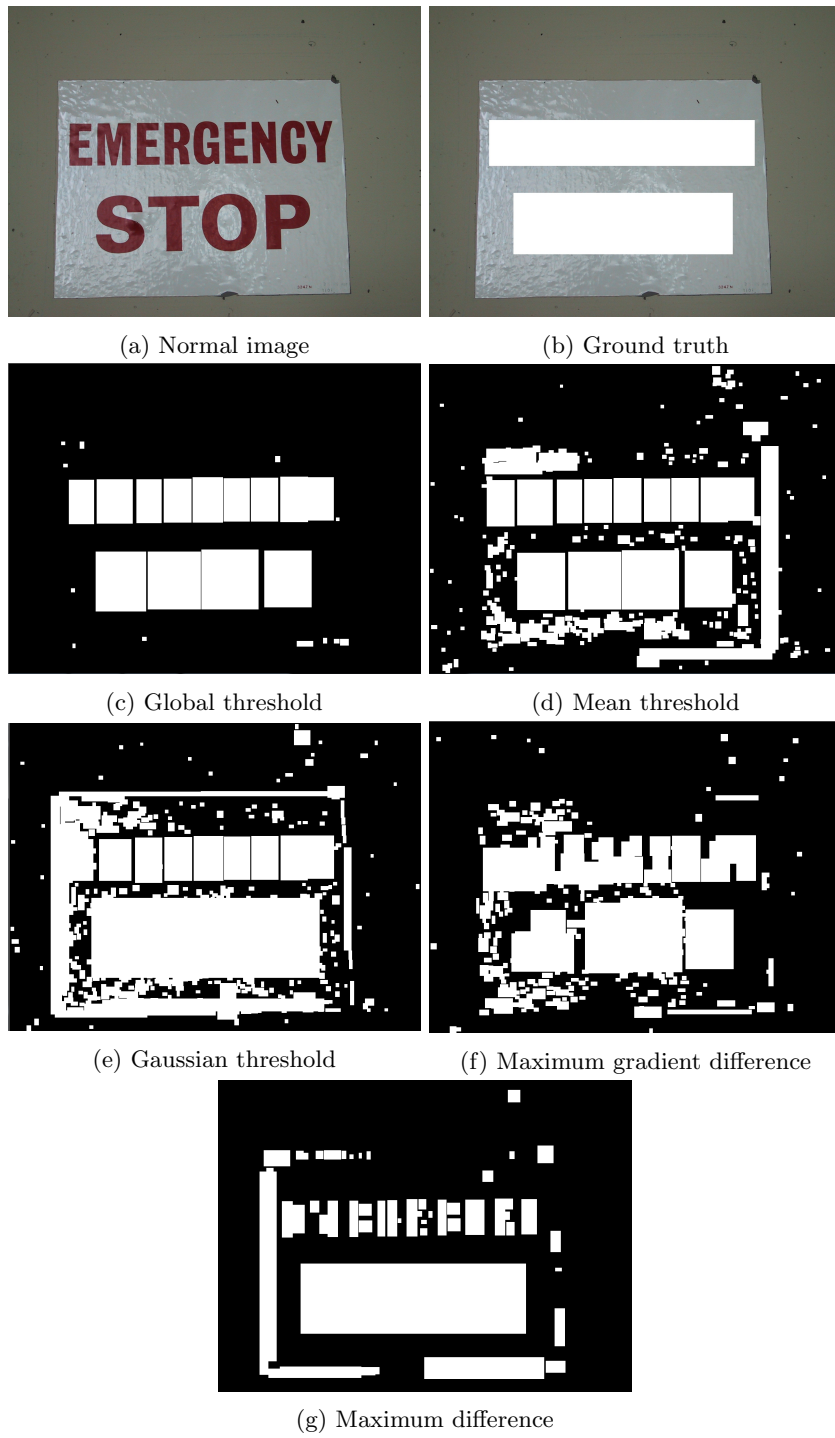


Figure 6.5: Results of coarse detection

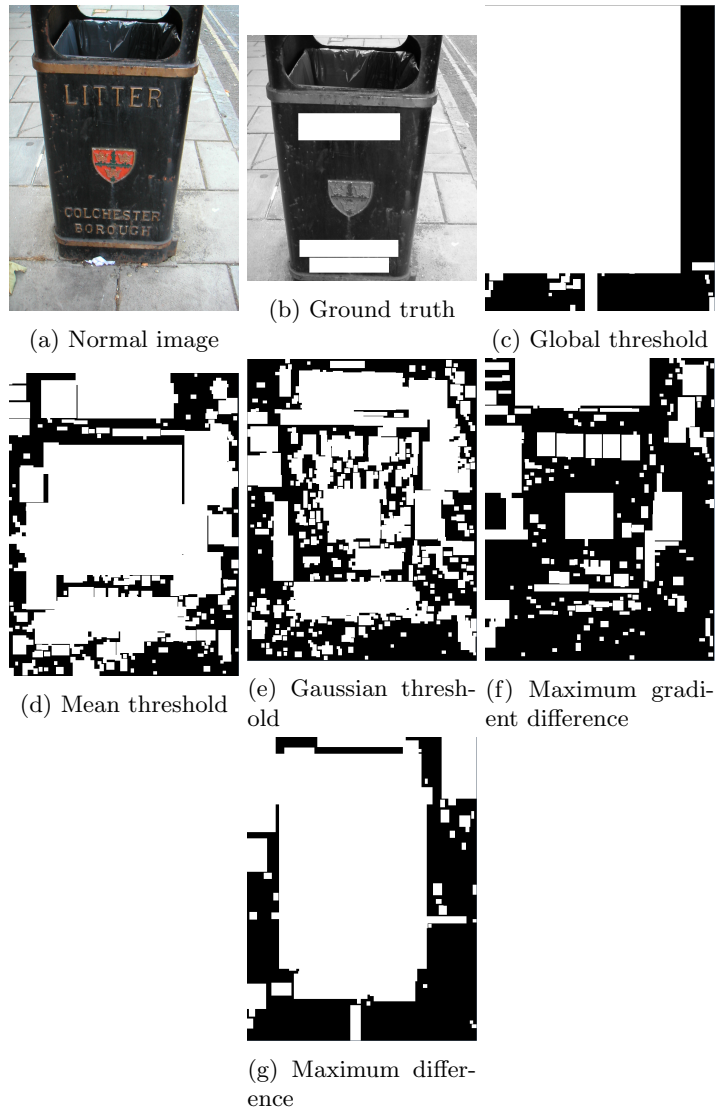


Figure 6.6: Results of coarse detection

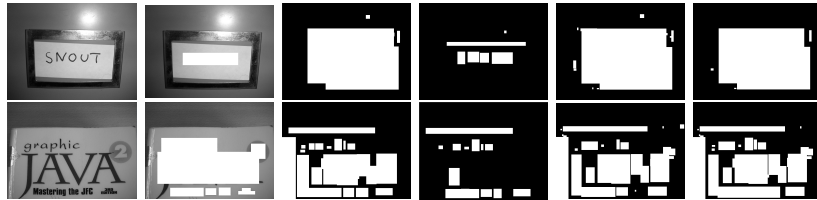


Figure 6.7: Difference between HoG and CoHoG using the max diff detection: from left to right, normal image, ground truth, HoG with projection, CoHoG with projection, HoG without projection, CoHoG without projection

Chapter 7

Conclusion

In this project an attempt has been made to find a new approach to text detection in natural images. This approach has to make sure that there are no constraints on size and amount of text in an image. A few different methods have been tested in order to find the best solution for this problem. For the first detection a few coarse detection methods have been used, namely global threshold, Gaussian threshold, mean threshold, maximum difference filter, maximum gradient difference filter. These approaches worked as expected, namely given a low precision, but maintaining a very high recall. From all these approaches the maximum difference filter performed best, having the highest precision but also having a decent recall rate. For coarse detection also local binary pattern was used, but this approach resulting in very bad results, worse than expected.

In order to combine the coarse detection with the fine detection that improves the precision of the results patches of the image are extracted. These patches have been marked by the coarse detection as text pixels. For each of these patches a projection step was used in order to split up the patches into single characters instead of full words. This filters out a lot of false detections from the coarse detection, but also some of the correctly marked text areas. The projection is based on vertical and horizontal lines with no text pixels, but not all words are split vertically or horizontally. Some fonts have an overlap in those directions. This meant that sometimes full words were not split up into single characters and were given the fine detection as a complete word. A better projection mechanism can increase the final results by a lot. Another problem with the projection step is that it sometimes removed patches that contained text, but were too close to the edge or partly out of the image.

The Histogram of Oriented gradients and co-occurrence histogram of orientated gradient descriptors performed decent on small set around the 90% accuracy, but the local binary pattern descriptor did not perform up to the expected standard and was not used in the combination step. The final accuracy of the HoG and CoHoG descriptors on the patches extracted from the coarse detection

resulted at around 63% and around the 70% if no projection step was included. Due to the results of the small set it should be possible to increase the accuracy of these results by a factor of 20% if the projection step improves. The final results of the overall pipeline did improve by using the feature descriptors, it improved the precision by 5%, but also reduced the recall by around the same amount. Compared to state-of-the-art methods this approach did not perform well. The precision of many state-of-the-art technology is a lot higher, but the recall rate is usually lower than the one found in this approach.

The main contribution of this paper is a coarse-to-fine detection schema that can be used as a base for future work that is cheap. Also it is shown that the CoHoG feature descriptor can be used for different purposes. A strength of the coarse-to-fine detection approach is that the problem of detecting text can be split up in smaller problems, each focusing on their own focus area instead of combining this into one big feature descriptor for example.

7.1 Future work

In order to improve the results of this approach a few techniques could be used.

- In order to improve the projection step instead of using horizontal and vertical projection a method from Cihan Topal and Cuneyt Akinlar called Edge Drawing can be used. This is an improvement on the canny edge detection which follows the edge. This method can be tuned in order to only keep edges that are connected. Each connection could be used in order to create a patch.
- Secondly different descriptors can be used in order to increase the accuracy. The descriptor of Kwang In Kim et al [5] can be experimented with as well. This feature descriptor has been used without any coarse detection step with decent results on precision level.

The coarse-to-fine detection has shown that it works pretty well, however an extra elimination step could be used in order to filter out even more unwanted patches. The coarse-to-fine detection schema can also be applied to existing methods that only use a fine detection for example the Stroke width approach from Ephstein [2].

Bibliography

- [1] Min Cai , A new approach for video text detection (2002), Image Processing.
- [2] Epshtein B., Detecting text in natural scenes with stroke width transform, 2010, Computer Vision and Pattern Recognition.
- [3] Datong Chen, Jean-Marc Odobez, Herve Bourlard, Text detection and recognition in images and video frames, Elsevier, 2003.
- [4] Michael R. Lyu, Jiqiang Song, Min Cai, A Comprehensive Method for Multilingual Video Text Detection, Localization, and Extraction, IEEE Transactions on Circuits and Systems for Video Technology, 2005.
- [5] Kwang In Kim , Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm, 2003, Pattern Analysis and Machine Intelligence.
- [6] Serge Belongie , Jitendra Malik , Jan Puzicha. Shape Context: A new descriptor for shape matching and object recognition (2000). In NIPS.
- [7] Gunnar Farneback, Fast and Accurate Motion Estimation using Orientation Tensors and Parametric Motion Models, Pattern Recognition, 2000.
- [8] Dalal, Nadal and Triggs, Bill (2005). Histograms of oriented gradients for human detection. Proc. Computer Vision and Pattern Recognition (CVPR'05) (San Diego, CA): I:886-893.
- [9] Lowe, David G. (2004). Distinctive image features from scale-invariant key points. International Journal of Computer Vision 60(2): 91-110.
- [10] Datong Chen, Jean-Marc Odobez, Herve Bourlard, 2003. Text detection and recognition in images and video frames.
- [11] Huizhong Chen, Sam S. Tsai, Georg Schroth, David M. Chen, Radek Grzeszczuk and Bernd Girod, 2011. Robust text detection in natural images with edge-enhanced maximally stable extremal regions (Stanford).
- [12] Marios Anthimopoulos, Text Detection in Images and Videos.

- [13] Nobuyuki Otsu, A Threshold Selection Method from Gray-Level Histograms, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-9, No. 1, January 1979
- [14] Shehzad Muhammad Hanif, Lionel Prevost, Text Detection and Localization in Complex Scene Images using Constrained AdaBoost Algorithm, International Conference on Document Analysis and Recognition, 2009
- [15] Lukas Neumann and Jiri Matas, A method for text localization and recognition in real-world images, Center for Machine Perception, Czech Technical University in Prague, Czech Republic, 2010
- [16] Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, Computer Vision and Pattern Recognition, 2005.
- [17] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. Gomez, S. Robles, J. Mas, D. Fernandez, J. Almazan, L.P. de las Heras, ICDAR 2013 Robust Reading Competition, 2013
- [18] T. Ojala, M. Pietikäinen, and D. Harwood, "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions", Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR 1994)
- [19] WEI, Baogang, et al. "A Novel Approach to Text Detection and Extraction from Videos by Discriminative Features and Density." Chinese Journal of Electronics 23.2 (2014).

Appendix A

Threshold full results

Table A.1: Results of using no thresholds and different bounding boxes

Direction	DilateX	DilateY	Precision	Recall	F
X	-	-	29.02%	48.65%	36.35%
X	3x5	-	22.12%	67.54%	33.3%
X	3x6	-	21.54%	69.98%	32.94%
X	3x7	-	22.39%	70.53%	33.99%
X	3x8	-	21.87%	73.23%	33.68%
X	3x9	-	21.43%	74.19%	33.25%
X	3x10	-	20.28%	73.76%	31.8%
X	4x9	-	21.12%	75.45%	33.00%
X	5x9	-	19.56%	77.28%	31.21%
X	6x9	-	19.43%	76.49%	30.99%
Y	-	-	23.34%	44.43%	30.6%
Y	-	3x1	21.16%	52.65%	30.18%
Y	-	3x2	21.46%	55.89%	31.01
Y	-	5x2	19.12%	61.23%	29.14%
Y	-	7x1	20.36%	62.38%	30.69%
Y	-	7x2	19.87%	65.49%	30.49%
Y	-	9x2	19.42%	68.38%	30.79%
Y	-	10x2	19.39%	66.86%	30.06%
Y	-	9x3	18.98%	69.98%	29.86%
Y	-	9x4	18.38%	69.38%	29.06%
X+Y	3x3	3x3	21.41%	78.02%	32.59%
X+Y	5x5	5x5	20.39%	80.78%	32.56%
X+Y	6x6	6x6	18.87%	83.34%	30.77%
X+Y	5x6	5x6	20.38%	82.29%	32.67%
X+Y	5x7	5x7	20.29%	83.36%	32.64%
X+Y	7x7	7x7	18.68%	84.74%	30.61%
X+Y	8x8	8x8	19.40%	86.49%	31.69%
X+Y	9x9	9x9	19.17%	86.85%	31.41%
X+Y	10x10	10x10	18.37%	87.12%	30.34%

Table A.2: Results of using gaussian thresholds and different bounding boxes

Direction	DilateX	DilateY	Precision	Recall	F	Window
X	-	-	22.05%	40.23%	28.48%	91
X	3x5	-	18.37%	79.74%	29.87%	91
X	3x6	-	18.48%	82.23%	30.17%	91
X	3x7	-	18.23%	84.43%	29.98%	91
X	3x8	-	19.22%	85.17%	31.36%	91
X	3x9	-	18.97%	85.49%	31.05%	91
X	3x10	-	17.76%	84.22%	29.33%	91
X	4x9	-	19.45%	83.77%	31.56%	91
X	2x9	-	19.28%	85.75%	31.48%	91
X	1x9	-	18.76%	85.45%	30.76%	91
X	2x8	-	19.45%	85.12%	31.66%	91
Y	-	-	23.37%	44.32%	30.60%	91
Y	-	3x1	17.18%	65.17%	27.19%	91
Y	-	3x2	17.48%	69.18%	27.90%	91
Y	-	5x1	17.27%	74.75%	28.05%	91
X	-	5x2	17.84%	75.43%	30.14%	91
Y	-	7x1	19.65%	62.24%	29.86%	91
Y	-	7x2	18.27%	78.86%	29.66%	91
Y	-	8x2	18.49%	80.02%	30.03%	91
Y	-	9x2	18.17%	81.17%	29.69%	91
Y	-	10x2	18.75%	82.45%	30.55%	91
Y	-	11x2	17.65%	80.97%	28.98%	91
Y	-	10x3	19.27%	83.14%	31.28%	91
Y	-	10x4	18.48%	82.11%	30.16%	91
X Y	-	-	18.16%	88.32%	30.12%	91
X & Y	3x6	5x1	16.87%	65.41%	26.82%	91
X & Y	2x9	10x3	17.90%	75.17%	28.91%	91
X Y	2x9	10x3	20.11%	86.98%	32.66%	91
X Y	3x6	5x1	19.30%	90.23%	31.79%	91
X Y	3x8	5x2	18.71%	88.63%	30.89%	91
X Y	3x6	5x1	17.15%	90.17%	28.81%	101
X Y	3x6	5x1	18.17%	89.19%	30.18%	71
X Y	3x6	5x1	18.40%	89.06%	30.49%	81

Table A.3: Results of using mean thresholds and different bounding boxes

Direction	DilateX	DilateY	Precision	Recall	F	Window
X	-	-	15.05%	40.32%	21.91%	91
X	3x5	-	15.42%	80.19%	25.86%	91
X	3x6	-	16.39%	81.81%	27.30%	91
X	3x7	-	16.07%	83.38%	26.94%	91
X	3x8	-	16.98%	84.34%	28.26%	91
X	3x9	-	17.08%	84.71%	28.42%	91
X	3x10	-	17.17%	84.42%	28.53%	91
Y	-	-	11.06%	31.01%	16.30%	91
Y	-	3x1	12.43%	64.67%	20.85%	91
Y	-	3x2	13.29%	67.82%	22.22%	91
Y	-	5x1	12.67%	74.05%	21.63%	91
X	-	5x2	13.89%	75.41%	23.45%	91
Y	-	7x1	14.06%	78.36%	23.84%	91
Y	-	7x2	15.41%	77.91%	25.73%	91
Y	-	8x2	15.31%	79.04%	25.65%	91
Y	-	9x2	15.08%	80.75%	25.41%	91
Y	-	10x2	15.72%	81.65%	26.36%	91
Y	-	11x2	16.30%	81.04%	27.14%	91
Y	-	10x3	15.64%	80.38%	26.18%	91
Y	-	10x1	15.36%	81.27%	25.83%	91
X Y	-	-	15.43%	90.26%	26.35%	91
X & Y	3x6	5x1	11.76%	63.41%	19.84%	91
X & Y	2x9	10x3	14.18%	73.67%	23.78%	91
X Y	2x9	10x3	19.31%	87.03%	31.60%	91
X Y	3x8	5x2	17.81%	90.44%	29.75%	91
X Y	3x6	5x1	16.94%	90.39%	28.53%	101
X Y	3x6	5x1	18.19%	88.72%	30.19%	71
X Y	3x6	5x1	18.30%	90.84%	30.46%	91
X Y	3x6	5x1	18.28%	92.10%	30.50%	81
X Y	3x6	5x1	17.91%	91.36%	29.94%	85

Table A.4: Best result of each different threshold

Method	Precision	Recall	F
Baseline	13%	100%	23.0%
None	22.39%	70.53%	33.99%
Global	17.41%	94.07%	29.38%
Mean	19.31%	87.03%	31.60%
Gaussian	20.11%	86.98%	32.66%

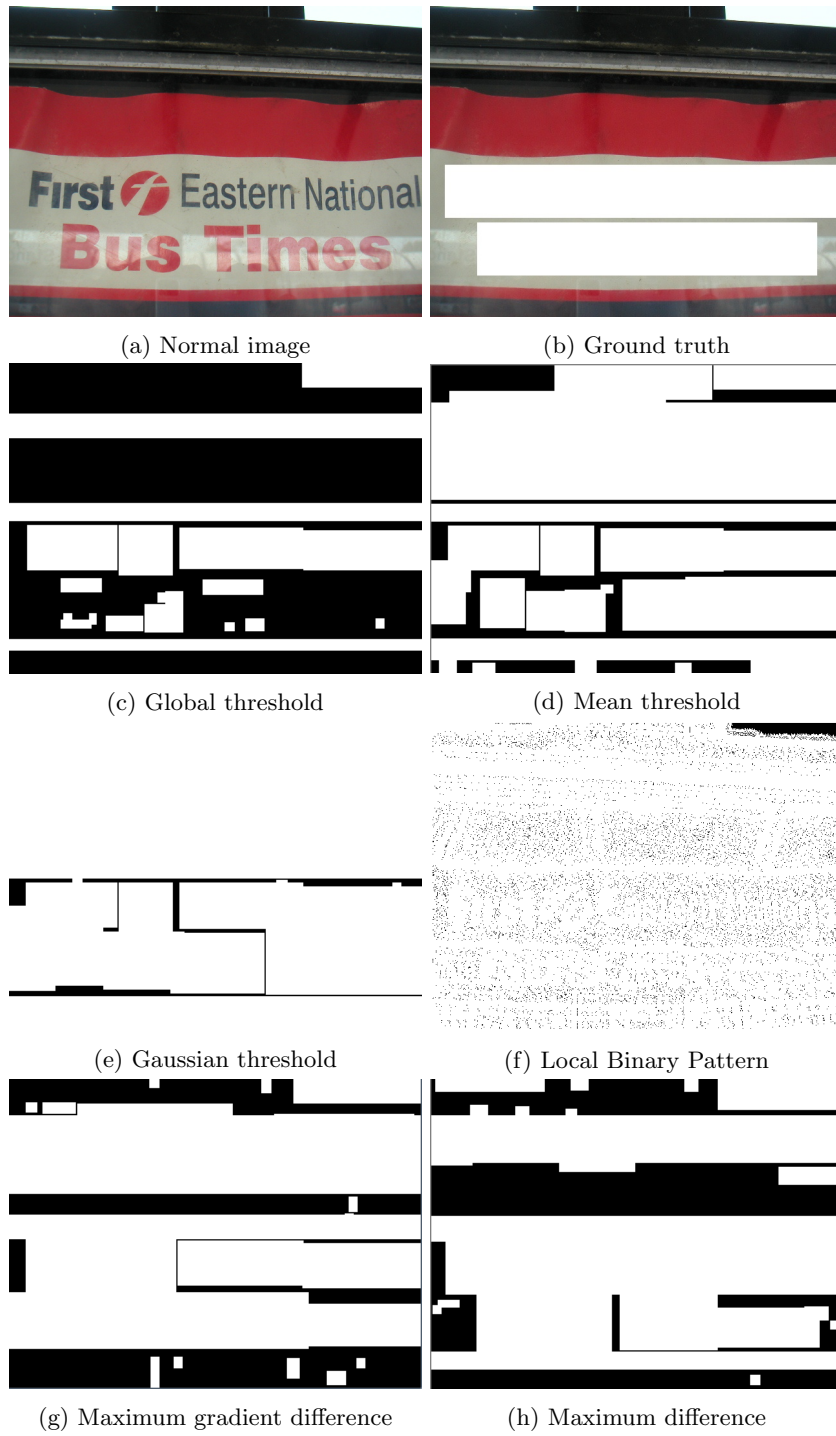


Figure A.1: Results of coarse detection

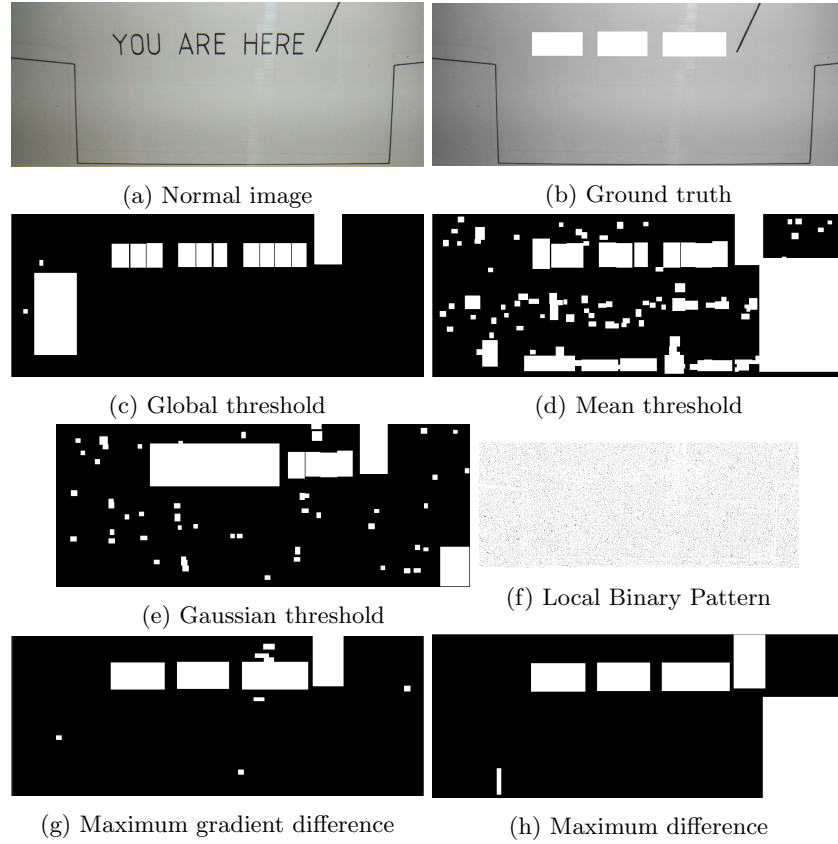


Figure A.2: Results of coarse detection 2

Table A.5: Local binary pattern using different window sizes

Window	Precision	Recall	F
5x5	12%	11%	11.47%
3x3	11%	12%	11.47%
50x50	12%	27%	16.61%
40x40(Star)	12%	23%	15.77%
20x20(Star)	11%	26%	15.45%
15x25(Star)	11%	17%	13.35%

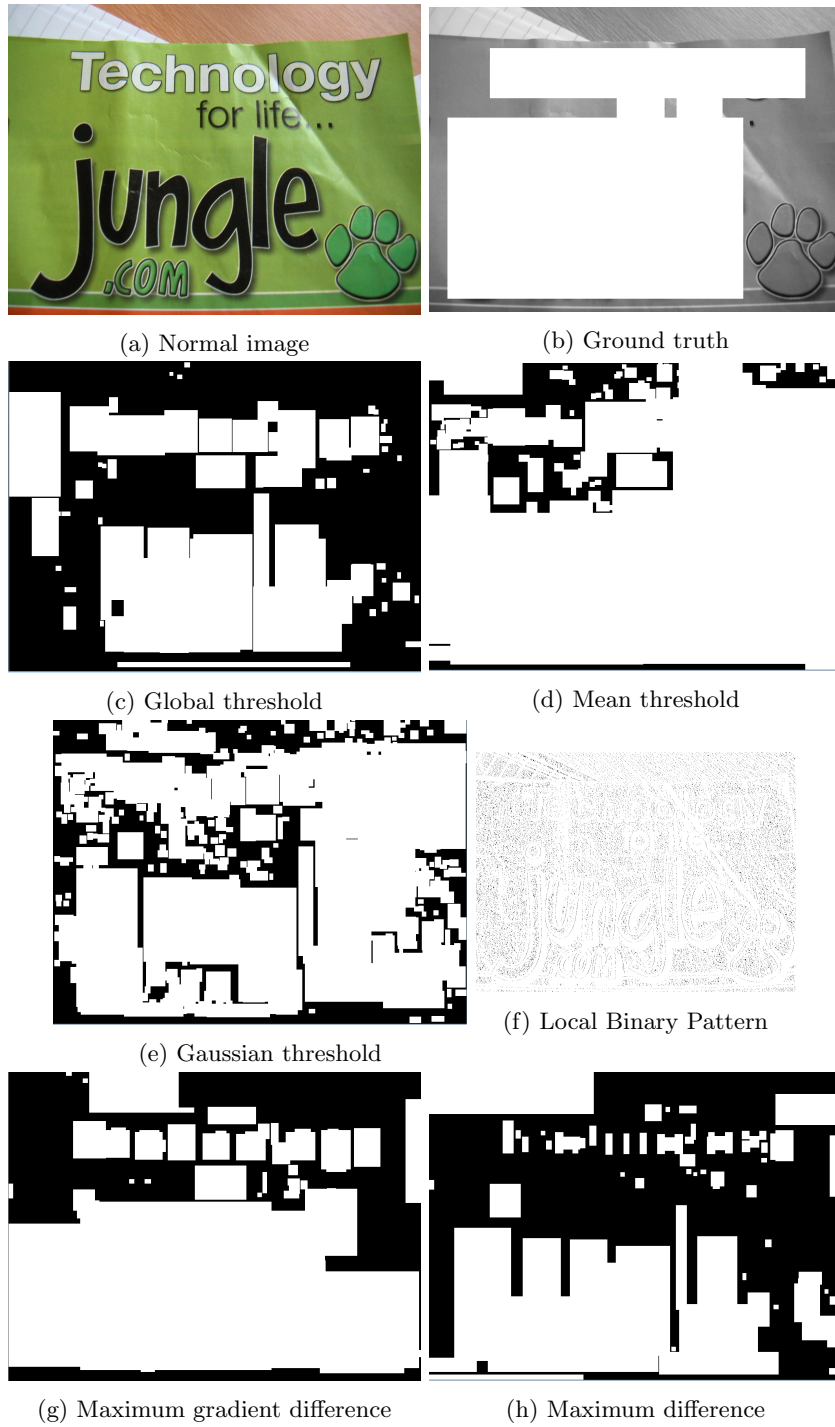


Figure A.3: Results of coarse detection 3

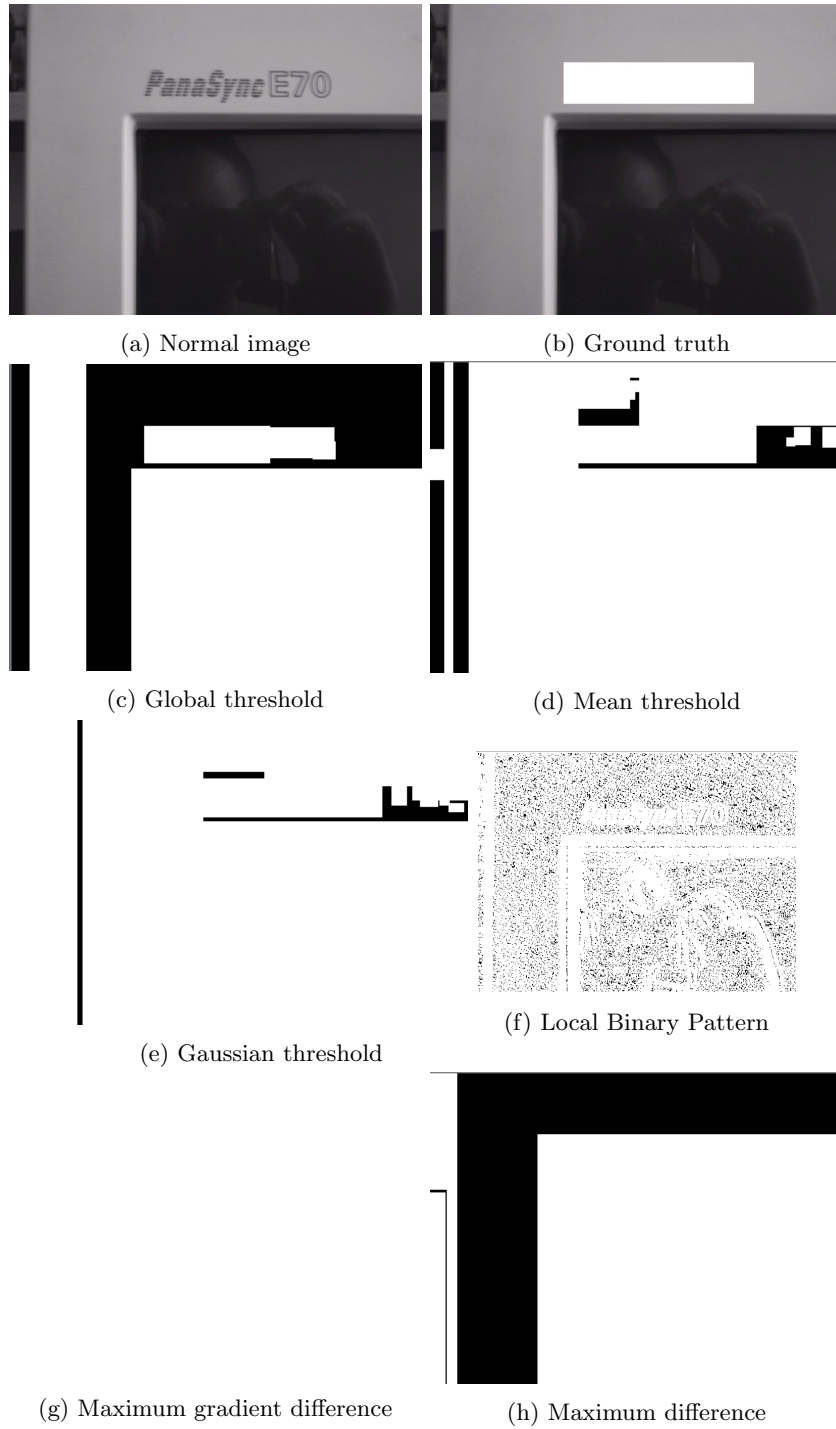


Figure A.4: Results of coarse detection 6

Table A.6: Results of using max gradient difference filter and different window sizes

Window	Preprocess	Combination	Precision	Recall	F
50	Laplacian	X+Y	22.06%	77.98%	34.39%
50	Laplacian	X	24.38%	64.61%	35.40%
50	Laplacian	Y	24.17%	61.47%	34.69%
50	None	Y	27.09%	48.06%	34.64%
50	None	X	26.81%	53.71%	35.76%
50	None	X+Y	22.72%	80.42%	35.43%
50	Gaussian Blur	Y	31.06%	50.38%	38.42%
50	Gaussian Blur	X	31.47%	55.92%	40.27%
50	Gaussian Blur	X+Y	24.03%	86.71%	37.63%
25	Laplacian	X+Y	26.38%	67.29%	37.90%
25	Laplacian	X	29.68%	50.81%	37.47%
25	Laplacian	Y	30.02%	46.08%	36.35%
25	None	Y	32.40%	36.73%	34.42%
25	None	X	32.58%	42.59%	36.91%
25	None	X+Y	28.34%	72.61%	40.76%
25	Gaussian Blur	Y	35.39%	38.31%	36.79%
25	Gaussian Blur	X	36.87%	46.28%	41.04%
25	Gaussian Blur	X+Y	29.06%	78.06%	42.35%
75	Laplacian	X+Y	19.45%	81.94%	31.43%
75	Laplacian	X	21.49%	73.72%	33.27%
75	Laplacian	Y	20.87%	69.03%	32.05%
75	None	Y	24.03%	57.49%	33.89%
75	None	X	24.42%	61.99%	35.03%
75	None	X+Y	20.87%	82.05%	33.27%
75	Gaussian Blur	Y	26.34%	53.71%	35.34%
75	Gaussian Blur	X	27.19%	56.34%	36.67%
75	Gaussian Blur	X+Y	22.30%	88.02%	35.23%

Table A.7: Results of using max difference filter and different window sizes

Window	Preprocess	Combination	Precision	Recall	F
50	Laplacian	X+Y	18.35%	74.56%	29.45%
50	Laplacian	X	21.02%	69%	32.22%
50	Laplacian	Y	21.26%	72.78%	32.90%
50	None	Y	24.40%	80.41%	37.43%
50	None	X	24.35%	82.74%	37.62%
50	None	X+Y	19.48%	77.06%	31.09%
50	Gaussian Blur	Y	26.58%	76.17%	39.40%
50	Gaussian Blur	X	28.03%	80.84%	41.62%
50	Gaussian Blur	X+Y	20.14%	77.03%	31.93%
25	Laplacian	X+Y	17.85%	67.17%	28.20%
25	Laplacian	X	20.68%	66.38%	31.53%
25	Laplacian	Y	20.26%	72.91%	31.70%
25	None	Y	21.72%	81.72%	34.31%
25	None	X	19.36%	66.71%	30.01%
25	None	X+Y	16.41%	57.08%	25.49%
25	Gaussian Blur	Y	24.31%	77.36%	36.99%
25	Gaussian Blur	X	21.38%	65.41%	32.22%
25	Gaussian Blur	X+Y	15.74%	58.29%	24.78%
75	Laplacian	X+Y	21.14%	73.75%	32.86%
75	Laplacian	X	21.84%	72.06%	33.52%
75	Laplacian	Y	21.23%	74.17%	33.01%
75	None	Y	25.49%	76.31%	38.21%
75	None	X	25.41%	82.47%	38.84%
75	None	X+Y	21.73%	86.91%	34.76%
75	Gaussian Blur	Y	27.64%	71.06%	39.79%
75	Gaussian Blur	X	30.26%	79.28%	43.80%
75	Gaussian Blur	X+Y	22.47%	86.94%	35.71%

Table A.8: Comparing text with non-text feature descriptors using a 200/200 training set

Method	Accuracy	Blocks	K
HoG	91%	4x4	-
HoG	88%	8x8	-
HoG	86%	16x16	-
CoHoG	89%	-	-
LBP	45%	-	8
LBP	51%	-	32
LBP	58%	-	64
LBP	61%	-	128
LBP	- %	-	256 (Too big for memory)

Table A.9: Results of using different coarse detection with HoG & SVM

Coarse Detection	Projection	Precision	Recall	F
Global	No	33.36%	62.36%	43.46%
Gaussian	No	27.18%	75.42%	39.95%
Mean	No	26.58%	76.97%	39.51%
MaxGrad	No	21.72%	55.03%	31.14%
MaxDiff	No	23.25%	80.68%	36.09%
Global	Yes	32.36%	61.01%	42.28%
Gaussian	Yes	30.47%	64.84%	41.45%
Mean	Yes	30.12%	65.08%	41.18%
MaxGrad	Yes	30.87%	14.03%	19.29%
MaxDiff	Yes	20.95%	73.04%	32.56%

Table A.10: Results of using different coarse detection with CoHoG & SVM

Coarse Detection	Projection	Precision	Recall	F
Global	No	25.36%	50.01%	34.52%
Gaussian	No	22.18%	67.06%	33.33%
Mean	No	19.09%	56.84%	28.58%
MaxGrad	No	21.75%	53.87%	30.98%
MaxDiff	No	26.84%	68.36%	38.54%
Global	Yes	35.06%	50.14%	41.26%
Gaussian	Yes	34.47%	51.84%	41.40%
Mean	Yes	35.98%	57.94%	44.39%
MaxGrad	Yes	19.04%	14.05%	16.16%
MaxDiff	Yes	32.35%	45.82%	37.92%

Table A.11: Best results of using different coarse detection with SVM based on F

SVM	Coarse Detection	Projection	Precision	Recall	F
HoG	Global	No	33.36%	62.36%	43.46%
CoHoG	Mean	Yes	35.98%	57.94%	44.39%