Georg Krispel

Multiple Frame Integration for OCR on Mobile Devices

MASTER'S THESIS

to achieve the university degree of
Diplom-Ingenieur

Master's degree programme
Telematics

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Horst Bischof
Institute for Computer Graphics and Vision

Graz, Austria, Dez. 2016

Abstract

The problem of Optical Character Recognition (OCR) is one of the oldest tasks in computer vision and on a well segmented and orthogonally viewed document it is considered solved. However, text occurs in all types of man-made surroundings and is exposed to a tremendous variety. The processing of this *scene text* is still a popular topic of recent research. With the rise of smart mobile devices in the last decade and the steady availability of high resolution cameras, the amount of possible applications even increased. Nevertheless, state-of-the-art scene text detection and recognition methods are generally not real-time capable, especially not on the limited hardware of mobile devices. Furthermore, the recognition of text in various surroundings and under low lighting is often poor. Since, camera streams provide multiple frames showing the same text, they can be exploited to improve the overall results. Missing information in a single frame due to reflections or occlusions can be compensated. In this work we propose a text detection and tracking pipeline dedicated to integrate the redundant information available in multiple frames. Thereby, the recognition performance is significantly increased. Additionally, it is shown that the proposed approach is capable of running in real-time on mobile device hardware exploiting parallel computing threads.

Kurzfassung

Optische Texterkennung (*OCR*) ist eines der ältesten Probleme in der automatisierten Bildverarbeitung und angewendet auf gut segmentierte und orthogonal betrachtete Dokumente wird es als gelöst angesehen. Allerdings kommt Text in allen menschengemachten Umgebungen vor und unterliegt einer umfangreichen Variation. Die Verarbeitung von diesem sogenannten *Scenetext* ist noch immer ein populäres Thema in aktuellen Publikationen. Vor allem durch die Verbreitung von Smartphones und Tablets im letzten Jahrzehnt und der damit einhergehenden ständigen Verfügbarkeit von hochauflösenden Kameras vervielfältigte sich die Anzahl der Anwendungen. Nichts desto trotz, sind moderne Methoden zur Detektion und Erkennung von Text in der Regel auf mobilen Endgeräten nicht echtzeitfähig. Weiters ist die Erkennung von Text in verschiedensten Umgebungen mit teils dürftigen Lichtverhältnissen oft mangelhaft. Da ein Kamera-Stream jedoch mehrere Bilder die denselben Text zeigen enthalten, kann diese Redundanz genutzt werden um die Resultate zu verbessern. Fehlende Informationen verursacht durch Reflektionen oder Hindernisse können kompensiert werden. In dieser Arbeit präsentieren wir eine Text Detektions- und Tracking-Pipeline zur Integration dieser Information. Hiermit kann die Texterkennung signifikant verbessert werden. Zusätzlich, wird gezeigt, dass der vorgeschlagene Ansatz in der Lage ist in Echtzeit auf mobilen Endgeräten zu funktionieren. Hierfür werden Prozesse unter der Verwendung von mehreren CPU-Kernen parallelisiert.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.

The text document uploaded to TUGRAZonline is identical to the present master's thesis dissertation.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Ort	Datum	Unterschrift

Acknowledgments

First of all, I want to thank Prof. Horst Bischof for being my supervisor and for enabling the possibility to do this thesis at the Institute for Computer Graphics and Vision (ICG). Many thanks are owed to Clemens Arth, Andreas Hartl, Christian Pirchheim and Horst Possegger for their helpful advice and useful hints. Additionally, I want to thank the entire team of Anyline for providing the topic and supporting me throughout working on this thesis.

Attention will be drawn here especially to my study colleagues for their support and friendship. Particularly Christoph, David, Erich, Johannes, Jörg, Thomas and Severin for numerous exhausting but still enjoyable night shifts, nerdy discussions about Maxwell's equations and some of my most memorable experiences.

Furthermore, I wand to thank my family, especially my parents and my sister for their unconditional support, no matter what I do or where I am. Above all, I want to express my deepest gratitude to my beloved girlfriend Tina, for calming me down, cheering me up, motivating me and calming me down again.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Contribution & Outline . . . 3

2 Text Detection and Recognition in Still Imagery **5**
- 2.1 Plane Rectification . . . 5
  - 2.1.1 Estimation of the Rectification Homography . . . 8
    - 2.1.1.1 Detection of Rectangular Regions . . . 8
    - 2.1.1.2 Vanishing Points . . . 9
- 2.2 Text Detection . . . 12
  - 2.2.1 Sliding Window Methods . . . 13
  - 2.2.2 Connected Component Analysis . . . 13
    - 2.2.2.1 Stroke Width Transform . . . 13
    - 2.2.2.2 Maximally Stable Extremal Regions . . . 16
- 2.3 Text Recognition . . . 20
- 2.4 Summary . . . 21

3 Text Detection, Tracking and Recognition in Videos **23**
- 3.1 Text Tracking . . . 23
  - 3.1.1 Template Matching . . . 24
  - 3.1.2 Particle Filtering . . . 26
  - 3.1.3 Tracking-by-Detection . . . 26
- 3.2 Multiple Frame Integration . . . 28
  - 3.2.1 Recognition result fusion . . . 28
  - 3.2.2 Image Enhancement . . . 29
- 3.3 Summary . . . 30

4 A Pipeline for Scene Text Processing on Mobile Devices **31**
4.1 Pipeline . . . 31
4.1.1 Visual Tracking . . . 32
4.1.2 Rectification . . . 35
4.1.3 Scene Text Detection . . . 36
4.1.4 Multiple Frame Integration . . . 37
4.1.5 Text Recognition . . . 37
4.2 Requirements . . . 38
4.3 Summary . . . 38

5 Impact of MFI on overall Text Processing **39**
5.1 Configuration . . . 39
5.1.1 Datasets . . . 40
5.2 Detection and Tracking Accuracy . . . 40
5.2.1 The CLEAR-MOT Metrics . . . 41
5.2.2 Experiments . . . 42
5.3 Reading Accuracy . . . 44
5.4 Algorithm Runtime . . . 45
5.5 Summary . . . 47

6 Conclusion & Outlook **49**

A List of Acronyms **51**

Bibliography **53**

List of Figures

1.1 Mobile OCR Applications 2

2.1 Street View Text (SVT) Dataset Samples 6
2.2 Homography Geometry 6
2.3 Metric Plane Rectification 7
2.4 Angle based Error Definition for Vanishing Point Detection 11
2.5 Stroke Width Transform Example 14
2.6 Stroke Width Transform Algorithm 15
2.7 Ambiguity of first pass in Stroke Width Transform (SWT) 16
2.8 Extremal Region Component Tree 17

3.1 Scene Text Character by Rong et al. 24
3.2 SWT-SIFT 25
3.3 Particle Filtering Process 27
3.4 Multiple Frame Integration 28

4.1 Pipeline Processing 32
4.2 Pipeline Modules 33
4.3 Pipeline Processing Example 34
4.4 Plane Rectification Verification 35

5.1 Sample Frame of Evaluation Datasets 41
5.2 Multiframe Integration over Time 44
5.3 Degenerated Multiframe Integration over Time 46
5.4 Recognition Rates 46

List of Tables

5.1 Evaluation Dataset Specifications 40
5.2 Detection and Tracking Accuracy 43
5.3 Recognition Rates depending on different MFI Methods 45
5.4 Recognition Rates with ECC Registration 47
5.5 Average Time Performance Measurements 47

1 Introduction

Contents

1.1 Motivation 1
1.2 Contribution & Outline 3

1.1 Motivation

The task of Optical Character Recognition (OCR) is one of the oldest challenges in computer vision. Since, its beginnings in mid of the last century a vast amount of research has been published and outstanding progress has been made. Thus, text recognition given an image of a well segmented, orthogonally viewed document using standard fonts is considered to be solved.

Still, needless to say, text is not limited to this constraint setup and occurs in our everyday life in almost any surroundings and situations. Hence, the applications of *OCR* are almost unmanageable. Information retrieval from video imagery for indexing, camera based aid for visually impaired, traffic sign detection and recognition as part of Advanced Driver Assistance Systems (ADAS) are only a few examples. Detecting and recognizing text in such cases is still a topic of nowadays research.

The *ICDAR 2015 Robust Reading Competition* received a total of 44 newly introduced submissions [29]. The tremendous variety in color, font, size or stroke-width text in natural scenes can take on, is a difficult challenge for researchers. One possibility to improve the particular algorithms is to exploit the redundant information available in video data. Since they are often provided for free, multiple frames showing the same text can be utilized to enhance the recognition experience.

Text in videos is generally divided in two categories: *Caption* and *scene* text. *Caption text* is artificially overlaid on television or movies to provide additional information for the viewer e.g. the end credits or a label indicating the talking person's name during

an interview. This type of text most often is clearly legible, likely related the to video's content and thus, valuable for indexing the respective type of video data.

What remains is *scene text* occurring in the depicted scene itself e.g. on signs of all kinds, license plates or packaging just to name a few possibilities. Contrary to caption text it can be distorted depending on the relative viewing pose and as already mentioned can take on an enormous variety. Therefore, algorithms have to adapt to a vast amount of possibilities to robustly spot and recognize such texts. Additionally, the readability of *scene text* depends on many more factors. For example obstacles, reflections or specular highlights can obscure parts of the desired text and complicate a successful recognition.

(a) **(b)** **(c)**

Figure 1.1: Several applications of text recognition on mobile phones. In (a) a winning code on a can lid is scanned, image (b) illustrated the scan of an International Bank Account Number (IBAN) to speed up a transaction process and in (c) scene text is recognized, translated and overlaid in the display. Image taken from web source[1] and [16].

With the general availability of mobile devices including high resolution cameras within the last decade, the demand on such applications even increased. An enormous amount of

[1] https://www.anyline.io/

unique video data is produced every day. Text in such videos is an ideal source of information to subscript and categorize this huge volume of data. However, text recognition on mobile hardware itself even has a remarkable number of further applications. It is already used to overlay text with a respective translation[2] or scan in transfer forms, passports and meter readings[3] just to name a few examples.

A significant part of these applications requires a nearly failsafe text recognition. Nevertheless, especially using mobile hardware the setups can vastly differ. There is great variety in hardware and in environmental conditions. This fact makes the task of robust text recognition even more challenging. Nevertheless, the text information is almost always redundantly available respectively the text is visible in multiple frames of a camera stream. Integrating this information can avoid failures and substantially increase the recognition certainty. This process is often referred to as Multiple Frame Integration (MFI).

1.2 Contribution & Outline

In this thesis we will propose a modular text processing pipeline capable of running on state-of-the-art mobile devices in real-time. In a fully-automatic procedure text is detected, tracked and integrated. A modular design allows the exchange of these modules as well as to simply integrate a recognition engine.

In the following we are going to give an overview of nowadays research on video text detection, tracking and recognition. Thereby, thoughts relevant for this thesis are considered in detail. Chapter 2 is limited to considerations applicable to individual frames. We will review related work in text detection and recognition. These methods are often the basis of multi frame approaches.

Afterwards, Chapter 3 extends the consideration to essentially enhancements possible with the exploitation of multiple frames. As a consequence, research in the field of object tracking and *MFI*, i.e. the fusion of the available information in the context of text recognition will be described.

In Chaper 4 we are going to give an overview of our approach and subsequently describe the different parts of our pipeline in detail. Finally, we present an evaluation on an use-case dedicated for *MFI* (Chapter 5), before giving an outlook and conclude the achievements in Chapter 6.

[2] For example the mobile phone app WordLense (http://questvisual.com/) or in research [16, 67]
[3] https://www.anyline.io/

2 Text Detection and Recognition in Still Imagery

Contents

2.1 Plane Rectification 5
2.2 Text Detection 12
2.3 Text Recognition 20
2.4 Summary 21

Given an orthogonal view on a binarized, well formatted text document using standard printed text fonts, Optical Character Recognition (OCR) is considered solved. Even on handwritten text recent research achieved human-competitive performance on the widely used MNIST [11, 83] dataset.
However, so-called scene text in natural scenes can take on comprehensive variety. Figure 2.1 shows sample images as part of the Street View Text (SVT) dataset [84]. The task to detect and recognize text in this uncontrolled environment faces numerous challenges like different size, resolutions, colors and perspective distortions. Hereafter, we will survey methods approaching these problems.

2.1 Plane Rectification

Text in images of natural scenes often experiences strong distortion. A text detection and recognition module must adapt to that in order to function robustly. Contrary, there is the possibility to add image rectification as a preprocessing step i.e. providing an undistorted orthogonal view on the text as input for detection and recognition. Generally, this may be a hard task. Nevertheless, in most cases words are written on a nearly planar surfaces. Thus, it is reasonable to model the distortion with a projective transformation. As a consequence, a simple matrix transformation is sufficient to transform distorted pixel coordinates into their rectified pendants. Please note, that throughout this thesis the term rectification is used in this context and not as the stereo

Figure 2.1: Samples of the Street View Text (*SVT*) dataset [84] harvested from Google Street View. Scene text with high variability in different outdoor scenes.

vision problem *image rectification*.
In the following, we will review the underlying basics and summarize common approaches.

As well known, any two images of a world plane are related by a projective transformation[1] [21]. This means, there exists a 3×3 matrix $\mathbf{H}$, which maps the two dimensional

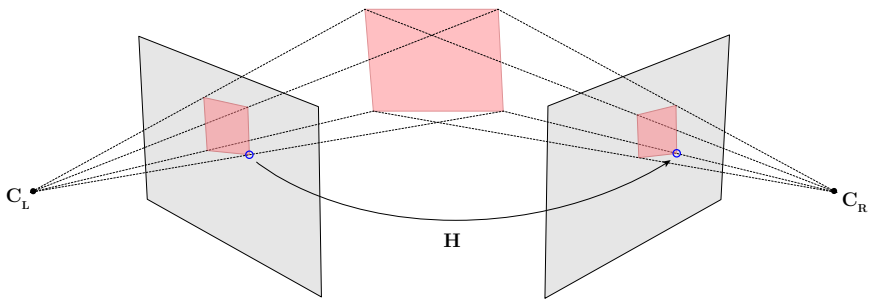


Figure 2.2: Corresponding points in different images representing the projected point on a world plane are related by the homography $\mathbf{H}$.

[1] Assuming a *pinhole camera model* [21].

point $\mathbf{x}$ in homogeneous coordinates in the first image to its corresponding point $\mathbf{x}'$ in the second image (see Figure 2.2) with

$$\mathbf{x}' = \mathbf{H}\mathbf{x}. \tag{2.1}$$

The projective transformation as well as the matrix are often referred to as *homography*, which has 9 entries, but only is defined up to scale. Hence, it has 8 Degrees of Freedom (DOF).
The target of plane rectification is now finding the transformation from a given image to a hypothetical one whose camera image plane is parallel to the world plane. The result would be an image showing a world plane without any distortion exceeding the class of similarity transforms, i.e. the world plane would be metrically rectified [41]. Figure 2.3 shows a possible input and result of a metric rectification.

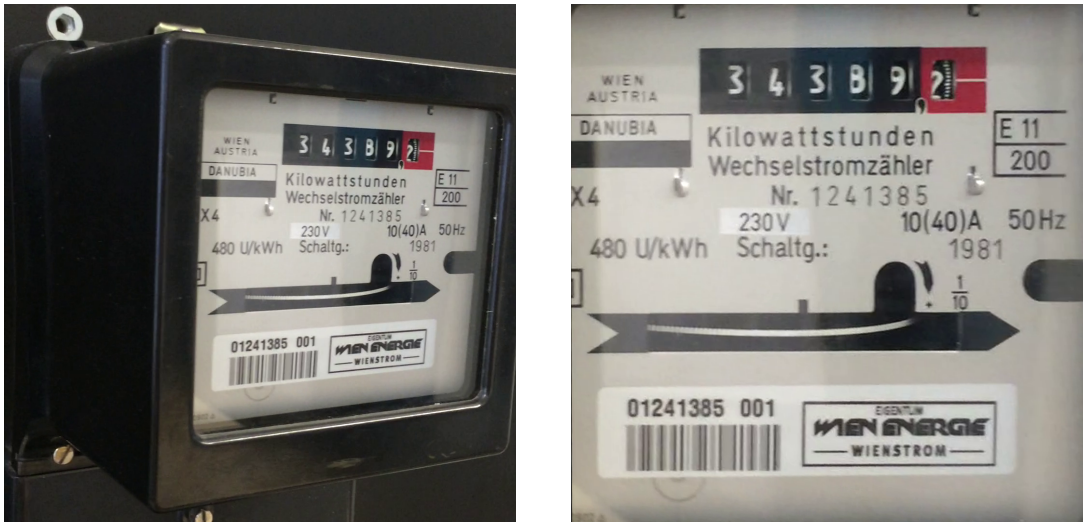


Figure 2.3: A perspectively distorted plane is metrically rectified.

To put it simply, the image shows the world plane with at most an isotropic scaling, rotation or translation to a certain extend. Most text detection and recognition methods can handle such an input. Some methods require text to be aligned horizontally or at least allow to speed up text detection under this assumption. Nevertheless, in practice proper rotation is generally achieved with no greater effort or given for free.

Assuming a *pinhole camera model* [21] a homogeneous world coordinate $\mathbf{X} = (X, Y, Z, 1)^T$ is projected to an image coordinate $\mathbf{x}$ with

$$\mathbf{x} = \mathbf{P}\mathbf{X}, \tag{2.2}$$

where $\mathbf{P}$ is the 3×4 projection matrix. For simplicity, we assume a world coordinate system such that every point on the world plane, which we want to rectify, has $Z = 0$.

Thus, we can reduce world coordinates to $\mathbf{X} = (X, Y, 0, 1)^T$ and leave out the third row of the projection leading to a 3×3 homography $\mathbf{M}$. Its inverse

$$\mathbf{H} = \mathbf{M}^{-1}, \tag{2.3}$$

is the desired rectification homography, since it maps arbitrary $\mathbf{x}$ to an image with a camera image plane identical to the world plane [35].

2.1.1 Estimation of the Rectification Homography

What remains is the task of finding the proper homography to metrically rectify the image. In the following, we will review two possibilities in the context of text detection.

2.1.1.1 Detection of Rectangular Regions

A common and well known way to estimate a homography between two images of the same planar surface is to establish point correspondences. Coordinates of at least 4 non-collinear points $\mathbf{x}'$ on the world plane and its projections $\mathbf{x}$ in the given image [21] have to be known. A homography $\mathbf{H}$ needs to satisfy 2.1 for all points. Since homogeneous coordinates are considered, the equality in Equation 2.1 is only defined up to scale. To take this into account commonly

$$\mathbf{x}' \times \mathbf{H}\mathbf{x} = \mathbf{0} \tag{2.4}$$

is used, where '$\times$' denotes the vector cross product. Given 4 point correspondences, there is an exact solution for $\mathbf{H}$. Having more than that leads to an over-determined system of equations. Most certainly there will only be an approximate solution due to measurement noise. There are several algorithms applicable to solve this type of linear problem, e.g. the Direct Linear Transformation (DLT) [21].

Many planar surfaces especially the ones containing text do have rectangular shape. Commonly, in images they are perspectively distorted and are depicted as general quadrangles. Suppose that the aspect ratio of the actual rectangle is known, metric plane rectification could be easily done by detecting the quadrangle.

The detection of rectangular regions has many applications as in object extraction especially in areal images, in augmented reality pipelines or license plate detection. Hence, there is a decent amount of research available. Lin et al. [44] used primitives likes edges and corners to form and verify rectangle hypotheses in building extraction. This procedure is often adapted in subsequent research [78]. Lagunovsky et al. [34] grouped line segements together, computed intersections to estimate quadrangles and approximated rectangles for object recognition.

A lot of research in this field is based on the Hough Transform (HT). Zhu et al. [97] introduces an rectangular *HT* to efficiently detect rectangles under the assumption that

the dimensions of the rectangular shaped objects are constant and known. Jung et al. [28] proposed a windowed *HT*. Thereby, line segments will be detected and grouped together, if certain geometric conditions are fulfilled.

Hartl et al. [20] proposed a rectangular region extraction approach capable of running efficiently on mobile devices. The method used a highly optimized and stack-based Canny edge detector and filtered text before line detection. The high frequent structures caused by text provoke a strong but false impact to the *HT*'s accumulator space. In order to roughly detect text, connected component are examined for certain criteria e.g. height, amount of pixel or aspect ratio relative to the particular bounding box.

After *HT*, the detected lines are grouped and filtered by their orientation to establish hypotheses for quadrangular structures. The resulting candidates are ranked. A dilated edge image is used to compute the support for each hypotheses. The best one is taken and is assumed to form the dominant rectangular region in the given image.

2.1.1.2 Vanishing Points

Manmade planar surfaces often possess texture containing parallel lines like on house fronts, documents or signs. Experiencing perspective distortion in images, these lines intersect in vanishing points, usually outside the image. Hereafter, we will consider a simple metric rectification using at least two vanishing points first introduced by Liebowitz et al. [41].

On a projective plane all vanishing points lie on the line at infinity $\mathbf{l}_\infty$. So given two vanishing points $\mathbf{u}$ and $\mathbf{v}$ it is easily computed by

$$\mathbf{l}_\infty = \begin{pmatrix} l_1 & l_2 & 1 \end{pmatrix}^T = \mathbf{u} \times \mathbf{v}. \tag{2.5}$$

In an image of a world plane experiencing perspective distortion vanishing points as well as the line at infinity are mapped to finite states. Thus, a transformation with the aim to undo this distortion must map all points $\mathbf{x}_\infty = (x, y, w)$ on this line back to infinity. For that coordinate w has to become zero, which leads to the matrix

$$\mathbf{H_P} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & 1 \end{pmatrix}. \tag{2.6}$$

After this transformation images are affinely rectified. The subsequent step is getting from affine to metric rectification. Therefore, the points

$$\mathbf{u_A} = \mathbf{H_P u}, \ \mathbf{v_A} = \mathbf{H_P v}, \tag{2.7}$$

which are the affine counterparts of the vanishing points $\mathbf{u}$ and $\mathbf{v}$, are computed. They lie

at infinity and represent directions. To remove skew and restore angles, the images have to be rotated such that $\mathbf{u_A}$ is aligned with the horizontal axis:

$$\mathbf{H_R} = \begin{pmatrix} \frac{u_{Ax}}{\|u_A\|} & \frac{u_{Ay}}{\|u_A\|} & 0 \\ -\frac{u_{Ay}}{\|u_A\|} & \frac{u_{Ax}}{\|u_A\|} & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.8}$$

Afterwards the transformation

$$\mathbf{H_A} = \begin{pmatrix} 1 & -cot(\theta) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.9}$$

is applied, whereas θ denotes the angle between the directions $\mathbf{u_A}$ and $\mathbf{v_A}$. There may still be a non-isometric scaling in one of the now orthogonal directions $\mathbf{u_A}$ and $\mathbf{v_A}$. If a length ratio of the world plane is known, it can easily be corrected with

$$\mathbf{H_S} = \begin{pmatrix} \mu & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.10}$$

while μ denotes the prevalent aspect ratio. The entire transformation $\mathbf{H}$ for metric rectification can be composed as

$$\mathbf{H} = \mathbf{H_S}\mathbf{H_A}\mathbf{H_R}\mathbf{H_P}. \tag{2.11}$$

Of course all considerations above rely on a robust vanishing point detection. Pilu et al. [69] exploited the structure of textural documents. Text is grouped using a proximity measurement to extract directional features and Random Sample Consesus (RANSAC) [15] to robustly estimate vanishing points. Miao and Peng [53] used morphological operations in order to obtain connected text and fit lines.

Yin et al. [92] proposed a fast vanishing point estimation dedicated for mobile devices. They perform a clustering and voting scheme on the Gaussian sphere in order to deal with problems based on viewpoint sensitivity and noise.

Hereafter, we will review a simple algorithm capable of finding multiple vanishing points first introduced by Nieto et al. [65]. It uses the M-Estimator Sample Consensus (MSAC) algorithm [80] in combination with an error function which is based on the angle between a proposed vanishing direction and a directional image feature.
Two different image features are used to get an idea of a vanishing direction, significant gradients and line segments. Let assume that a single feature sample $\mathbf{x}$ is parametrized by its homogeneous coordinates $\mathbf{r} = (x, y, 1)^T$, i.e. the gradients position respectively the midpoint of the line segment, furthermore the line $\mathbf{t} = (t_1, t_2, t_3)^T$ representing the proposed vanishing direction.

In an ideal situation, an actual vanishing point $\mathbf{v} = (v_1, v_2, v_3)^T$ lies on every feature's directional line $\mathbf{t}$ by means of

$$\mathbf{v} \cdot \mathbf{t}^T = 0. \tag{2.12}$$

Needless to say, due to outliers, noise and poor feature extraction the result will deviate from this expectation. A certain error will occur for all features. Thus, the aim is now to find a vanishing point $\tilde{\mathbf{v}}$ which robustly minimized this error.

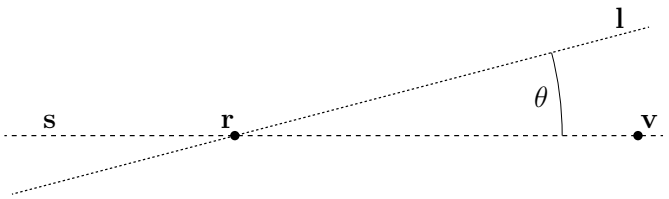


Figure 2.4: An image feature with position $\mathbf{r}$ and its proposed vanishing direction $\mathbf{l}$ deviate from a vanishing hypothesis $\mathbf{v}$. The error definition is based on the angle included by the lines $\mathbf{l}$ and $\mathbf{s} = \mathbf{v} \times \mathbf{r}$.

Nieto et al. propose a definition based on the angle θ between $\mathbf{t}$ and the line passing through the feature position and the vanishing point $\mathbf{s} = \mathbf{r} \times \tilde{\mathbf{v}}$ (see Figure 2.4):

$$d(\mathbf{x}, \tilde{\mathbf{v}}) = |sin(\theta)| = \left| \frac{-t_2 s_1 + t_1 s_2}{\sqrt{t_1^2 + t_2^2}\sqrt{s_1^2 + s_2^2}} \right|. \tag{2.13}$$

The sine is chosen since $sin(\theta) \simeq \theta$ for small θ and expensive trigonometric is avoided. Not all features are equally meaningful. Higher gradients and longer line segments make most likely a better statement. Hence, it is reasonable to add a weighting W

$$d'(\mathbf{x}, \tilde{\mathbf{v}}) = W \cdot d(\mathbf{x}, \tilde{\mathbf{v}}). \tag{2.14}$$

Assuming Gaussian noise the probability that a feature $\mathbf{x}$ points to the vanishing direction is

$$p(\mathbf{x}|\tilde{\mathbf{v}}) \propto exp\left(-\frac{1}{2\sigma^2} d^2(\mathbf{x}, \tilde{\mathbf{v}})\right). \tag{2.15}$$

Given a set of N features $\mathcal{X} = \{\mathbf{x_i}\}_{i=1}^N$, the likelihood that all features meet a single vanishing points is then

$$p(\mathcal{X}|\tilde{\mathbf{v}}) = \prod_{i=1}^{N} p(\mathbf{x_i}|\tilde{\mathbf{v}}). \tag{2.16}$$

Under the assumption that 2.15 is normal, i.e. $\sigma = 1$, the maximum likelihood is found by minimizing the sum of squared errors $E = \sum_{i=1}^N d^2(\mathbf{x_i}, \tilde{\mathbf{v}})$. Since, this a highly non-linear problem, the Levenberg-Marquardt [37] method is proposed for solving it.

Up to now only a single vanishing point and no outliers were considered. Hence,

MSAC is used. It generalizes *RANSAC* through providing a more descriptive definition of an hypothesis cost.
Both algorithms select a random minimum subset of samples to establish a hypothesis and evaluate the consensus. Here, two feature's line parameters are used to compute an estimated vanishing point $\mathbf{v}^* = \mathbf{l_i} \times \mathbf{l_j}$. A threshold T is defined for the error, differentiating between in- and outlier regarding the current hypothesis. *RANSAC* tries to maximize the cardinality of the consensus set

$$\mathcal{CS}(\mathbf{v}^*) = \left\{ \mathbf{x_i} \in \mathcal{X} : d^2(\mathbf{x_i}, \mathbf{v}^*) < T \right\}, \tag{2.17}$$

containing all features which match the hypothesis. Contrary, instead of simply counting the inliers, *MSAC* proposes to sum up the error itself truncated by the threshold to obtain the global loss

$$E(\mathcal{X}, \mathbf{v}^*) = \sum_{i=1}^{N} e(\mathbf{x_i}, \mathbf{v}^*), \tag{2.18}$$

whereas

$$e(\mathbf{x_i}, \mathbf{v}^*) = \begin{cases} d^2(\mathbf{x_i}, \mathbf{v}^*) & d^2(\mathbf{x_i}, \mathbf{v}^*) \leq T \\ T & \text{otherwise} \end{cases}. \tag{2.19}$$

Thereby, with no greater effort a more precise consensus evaluation is achieved. Nieto et al. also proposes Maximum Likelihood Estimation Sample Consensus (MLESAC) [80] as an even more accurate way to model inlier and outlier. It evaluates the hypothesis based on a probability distribution modeling inlier and outlier error. Assuming a Gaussian distribution for an inlier and a uniform distribution for an outlier this leads to

$$p(\mathbf{x_i}, \mathbf{v}^*) = \gamma e^{\left(-\frac{1}{2\sigma^2} d^2(\mathbf{x_i}, \tilde{\mathbf{v}})\right)} + (1 - \gamma) \frac{1}{d_{max}}, \tag{2.20}$$

whereas γ is the prior probability of a feature being an inlier and d_{max} is the maximal possible error. Since the prior probability is not known, an Expectation Maximization (EM) approach is used to estimate it.

2.2 Text Detection

The aim of text detection is to spot and localize text in an image. The output of this step is usually a bounding box or even a pixel-wise segmentation. The related methods are usually categorized into *Connected Component Analysis (CCA)* and *sliding window approaches*. In the following, we will summarize both methods and give a short overview of the related research.

2.2.1 Sliding Window Methods

These methods generally examine an image with a sliding window of multiple scales and utilize a subsequent classifier to spot text. Chen et al. [10] proposed an AdaBoost machine learning algorithm. They analyzed various image features on text and non-text to find good indicators. The resulting weak classifiers are used as input for AdaBoost to train a strong classifier.
Recent research used deep learning respectively Convolutional Neural Networks (CNNs). In [86] an entire end-to-end text recognition was build in this manner. During an unsupervised step low-level features are learned and fed into two *CNNs*, one for each, detection and recognition. Jaderberg et al. [26] used four different networks which share the same two input layers for low-feature extraction to learn a character/non-character classifier, case-insensitive character classifier, case-sensitive character classifier and a bigram classifier.

2.2.2 Connected Component Analysis

CCA tries to group characters together by exploiting properties, which the entire text has in common like color, size or stroke width. Thereby, text proposals are established. Generally, a verification step is appended to refine the result. Color bleeding, low resolution and perspective distortion can mitigate the results of this approaches. Contrary, the computation complexity usually does not depend on the text properties itself and most often a segmentation for recognition is provided with no additional effort. Distinguished outcomes can be achieved with this methods. Some of the most successful concepts in this category rely on the Stroke Width Transform (SWT) or Maximally Stable Extremal Region (MSER). In the following, we will outline the underlying principles of these approaches and review some applications.

2.2.2.1 Stroke Width Transform

The *SWT* operator first introduced by Epshtein et al. [13] exploits the constant stroke width of characters within text. It computes pixel-wise the width of the stroke it is most likely related to. Thus, the output is an image of same size as the input, with the width of the respective related strokes stored in each pixel. An example is illustrated in Figure 2.5.

First the output image is initialized with ∞. To obtain the stroke's boundaries, Canny edge detection [8] is performed on the input. Two parallel running boundaries with nearly constant distance most likely form a stroke. To compute its width, each boundary pixel $\mathbf{p}$ is considered. Searching along its gradient direction $\mathbf{d_p}$ results in finding pixel $\mathbf{q}$ on opposite site of the stroke. If gradient direction $\mathbf{d_q}$ of pixel $\mathbf{q}$ approximately points in opposite direction as $\mathbf{d_p}$ each pixel along the segment $[\mathbf{p}, \mathbf{q}]$ is

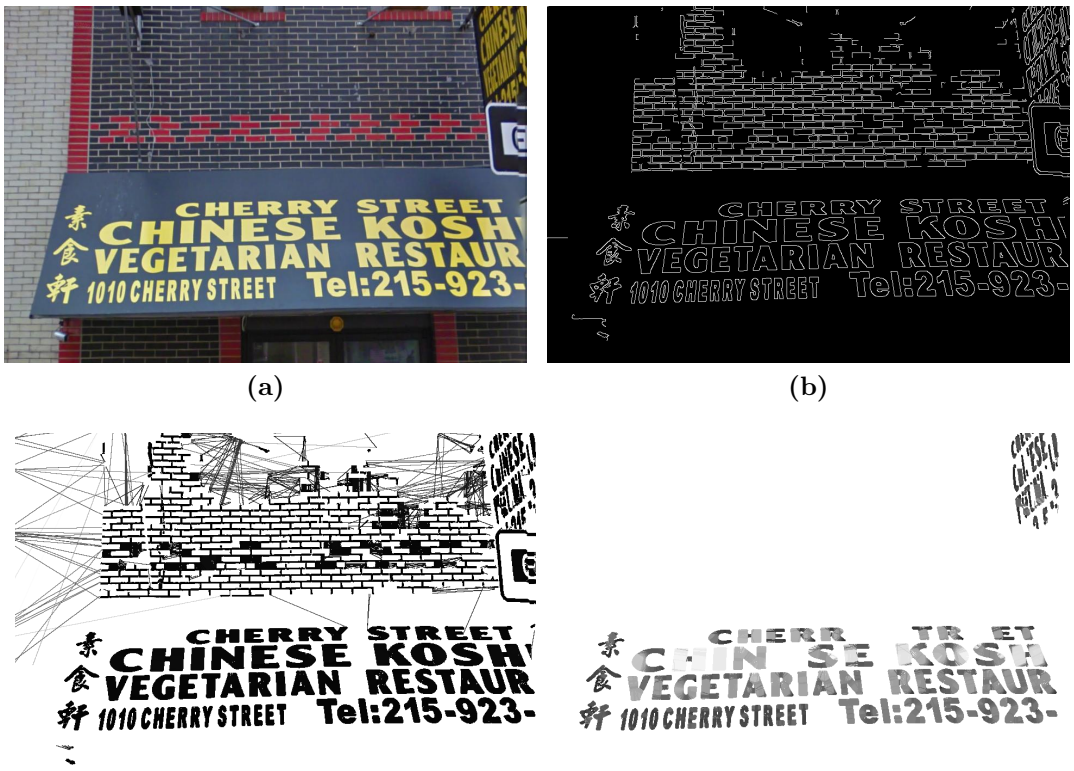
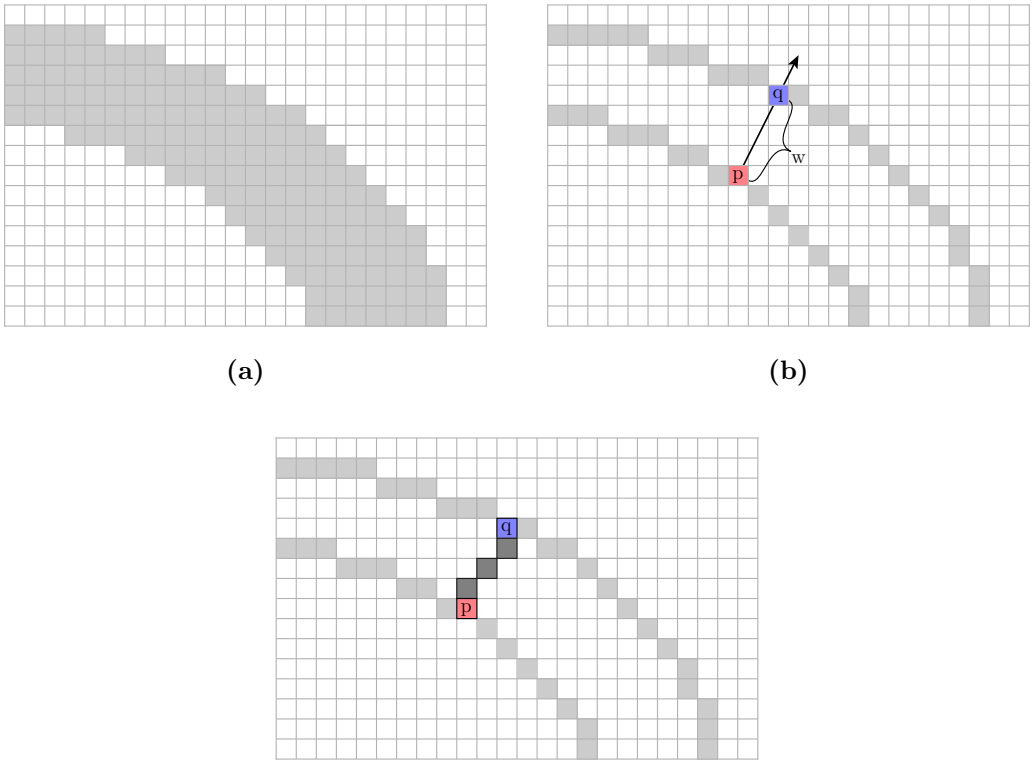


Figure 2.5: Stroke Width Transform (*SWT*) applied to an image of *SVT* dataset. (a) shows the input image. Canny edges detection is applied (b) and the *SWT* assigns the estimated width of the particular strokes to each pixel (c). In (d) non-text regions are filtered. Not all characters were detected.

assigned to $||\mathbf{p} - \mathbf{q}||$, besides it already has a lower value. If no opposite pixel $\mathbf{q}$ is found, or a smaller value is already assigned, no values are changed. Figure 2.6 depicts the process for a single edge pixel.

After this first step especially in corners, pixel do not contain the actual stroke width as depicted in Figure 2.7. Hence, each non-discarded search direction is passed again and all pixel values are replaced by their former median. This ensures that each pixel value along a search direction contains the same and most probable stroke width.
Given this output neighboring pixels are grouped together using the classical *CCA* approach [22]. As association characteristic the *SWT* values are used and their ratio is spatially limited to a certain value. This permits slightly varying stroke widths given more exceptional fonts and perspective distortions. To detect dark text on bhright background and vice versa, the steps above are executed with both, positive and negative gradient direction.



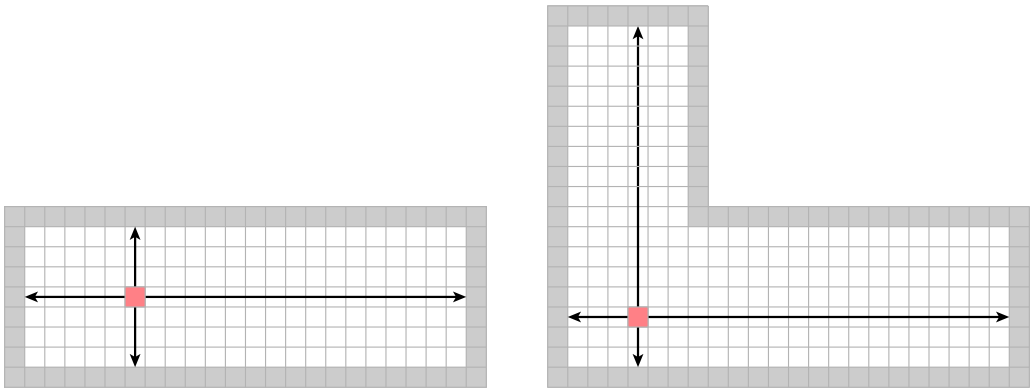
(a) **(b)** **(c)**

Figure 2.6: Stroke Width Transform (*SWT*) algorithm. (a) shows a typical stroke. (b) illustrates the distance computation. After Canny edge detection at each border pixel it is searched in gradient direction to recover the opposite border pixel. In (c) each pixel in search direction is assigned to the distance w.

To identify actual letters from these components several learned parameter are proposed. Among others the variance and aspect ratios of the stroke widths, the ratio between the components diameter to the median stroke width and the dimensions of the components are restricted. Finally, the often linear form of the text is used to group the letters into text lines. Hence, similar properties like size, stroke width, average color and spacing is exploited.

Due to its effectiveness, several researchers adopted the *SWT*. Yao et al.[90] enhanced the component filtering and verification step: They proposed a two-layer method to discard non-text regions, which exploits fast geometric heuristics and a more sophisticated trained classifier. Later on, the classification was improved and dictionary based error correction was introduced to enhance the recognition [89].

Huang et al. [23] introduced the Stroke Feature Transform (SFT), an extension of *SWT* which additionally considers the color information. During the search for opposite pixel, it



(a) **(b)**

Figure 2.7: During first iteration of *SWT* the minimum distance is assigned to the respective pixels (a). Especially in corners often not the actual width is assigned (b). Thus, a second pass is needed to review all valid search directions and assign the median of widths to each pixel.

examines if there is no sudden color change along the gradient direction. Further, a second output image is returned with each pixel containing the mean color of the respective stroke. Both, the stroke color map and the stroke width map are used to improve the grouping procedure.

2.2.2.2 Maximally Stable Extremal Regions

MSERs were originally introduced by Matas et al. [49] as an affine covariant and stable blob detector. They were used to establish correspondences between images showing the same scene under vastly different viewpoints and improve object detection.
Nevertheless, they experience a high acceptance as basis for text detection. In fact, almost all submissions at the *ICDAR 2015 Competition on Robust Reading* contest make use of a segmentation based on the *MSER* algorithm [29]. Hereinafter, we will outline the underlying principles.
We will assume a gray valued image I_{in} with intensity values $S = \{0, 1, ..., 255\}$[2]. Considering a thresholding at every intensity value t results in the binary images

$$I_b^t = \begin{cases} 1 & \text{if } I_{in} \geq t \\ 0 & \text{otherwise} \end{cases} . \tag{2.21}$$

We will refer to pixels with values zero and one as black and white respectively in the following. Let threshold t grow in the interval $[min(I_{in}), max(I_{in})]$. The higher it gets, the more pixel are set to zero. Beginning with an entire white image, black regions will appear, grow and merge until the white area finally disappears. In each of these binary

[2] *MSER* can be defined on any image as long there is a total ordering.

images Extremal Regions (ERs) can be recovered. An *ER* $\mathcal{Q}$ is defined by

$$\forall p \in \mathcal{Q}, \forall q \in boundary(\mathcal{Q}) \longrightarrow I_b(p) \leq I_b(q), \tag{2.22}$$

i.e. a contiguous region which outer boundary pixels have strictly higher values than the region itself. With the thresholded binary images in mind they are black contiguous regions surrounded by white pixels[3]. This concepts relies on a certain definition of adjacency relation or contiguity. Matas et al.propose a 4-neighborhood where every pixel with the coordinates

$$(x \pm 1, y) \text{ or } (x, y \pm 1) \tag{2.23}$$

is connected to the pixel at (x, y).
Starting with the *ER* at $t = max(I_{in})$ as root, a *component tree* can be introduced. Each node represents an *ER* and the tree depth is related to the decreasing threshold t. If an *ER* is split up due to a lower threshold, the particular tree node will have a child for each part. Thus, the tree's edges represent an inclusion relation, i.e. if a region $\mathcal{R}$ is the child of a region $\mathcal{Q}$ this means that

$$\forall p \in \mathcal{Q} \longrightarrow p \in \mathcal{R}. \tag{2.24}$$

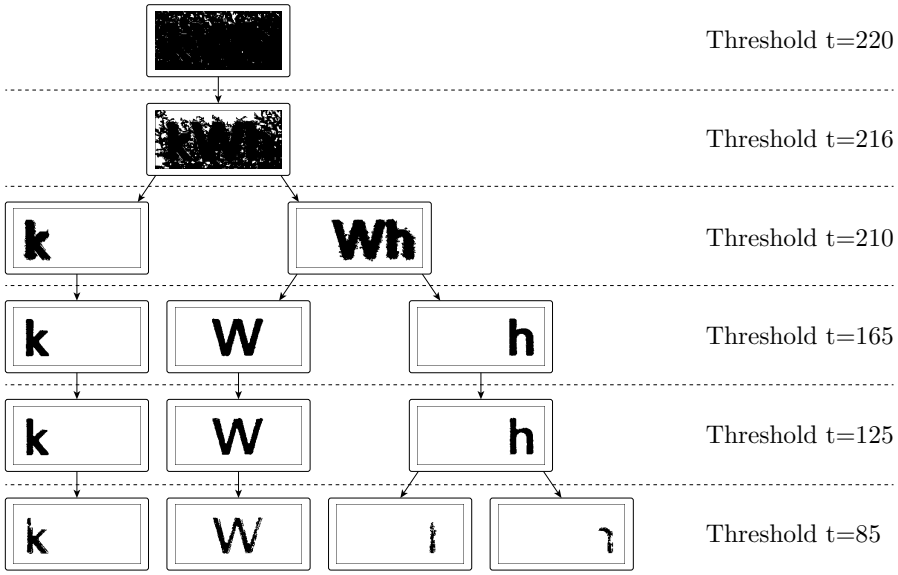


Figure 2.8: Extremal Region (*ER*) component tree showing the different thresholded binary images and their inclusion relation. The letter k stays nearly unchanged over a wide range of different thresholds and hence is most likely an Maximally Stable Extremal Region (*MSER*).

Figure 2.8 depicts a simplified *ER* component tree sampling certain thresholds. As

[3] In accordance to [49] they are referred to as *minimum intensity regions*.

can be seen, the letter k stays nearly the same respectively is *stable* over a wide range of thresholds t. Connected components which maximize this stability property are the wanted *MSERs*. What remains is to give a precise definition of this stability criteria.
Let $\mathcal{Q}_1, ..., \mathcal{Q}_i, \mathcal{Q}_{i+1}, ...$ be a set of nested *ERs*, i.e. they are nodes on a path of the component tree and $\mathcal{Q}_i \subset \mathcal{Q}_{i+1}$. The variation or instability Ψ of a region $\mathcal{Q}_i$ is given by

$$\Psi(\mathcal{Q}_i) = |\mathcal{Q}_{i+\Delta} \setminus \mathcal{Q}_{i-\Delta}| \; / \; |\mathcal{Q}_i|, \tag{2.25}$$

whereas $|.|$ denotes the cardinality. Δ is a parameter of method and influences the range over witch the instability is considered. The region $\mathcal{Q}_i^*$ which minimizes Ψ along a path to the root is considered a *MSER*.

A lot of research used the *MSER* algorithm to establish text candidates. Needless to say, a lot of non-text regions are *MSERs* as well. Thus, a subsequent verification step is commonly used to discard non-text regions.

Chen et al. [9] enhanced the mask of an *MSER* with the help of Canny edges from the original gray-scale image: To remove over-segmented pixels in blurry images, foreground pixels are pruned along the gradient direction of the edges. Subsequently, an adapted distance transform estimated the stroke width of each pixel. Connected components with a large variation in stroke width were discarded. Contrary, they were grouped into text lines by similar stroke width and proximity.

In [74] Shi et al. formulated the categorization into text/non-text as minimal cut problem of an undirected graph. Each *MSER* denotes a node and is connected to similar and nearby pendants as well as to two terminals, a foreground and a background node. The weighted edges were cut in a minimal sense leaving labeled regions, which are grouped into words and text lines by exploiting several similarity and proximity heuristics.

Koo et al. [31] clustered *MSERs* by pair-wisely examining adjacency relationships with an AdaBoost classifier. The candidates are geometrically normalized, binarized and fed into a multilayer perceptron classifier to filter non-text regions.

Depending on the parametrization, simple *MSER* extraction can lead to multiple hierarchically redundant and overlapping text candidates. Yin et al. [93] filtered *MSERs* by additionally (see Equation 2.25) penalizing a degenerated aspect ratio of the region. Thereby, characters are most likely preferred over child/parent regions. Text is grouped using a single-link clustering approach [27] whilst exploiting distance measurements computed by a self-training learning algorithm.

The adaptation of *ER* by Neumann and Matas [60–64] is one of the most successful end-to-end scene text localization an recognition approaches. The pipeline is referred to as *TextSpotter 2013*. They examined the set of *ERs* in order to efficiently select character proposals. During an exhaustive search [61] they are grouped into words.
They exploited the inclusion relation denoted in Equation 2.24 to efficiently compute

descriptors which should indicate whether an *ER* is a character or not: A region $\mathcal{Q}^t$ at threshold t contains the union of some (or none) *ERs* at $(t-1)$ and the adjacent pixels with value t. Thus, they chose descriptors which deliberate this property i.e. can be computed using the including *ER*'s descriptors and the additional pixels.

More specifically let assume that the regions $\{\mathcal{Q}^{t-1}\}$ and the pixels $\{p : I_{in}(p) = t\}$ form the *ER* $\mathcal{Q}^t$. Then the descriptor $\phi(\mathcal{Q}^t)$ is computed by

$$\phi(\mathcal{Q}^t) = \left(\oplus\, \phi(\mathcal{Q}^{t-1})\right) \oplus \left(\oplus\, \psi(p)\right), \tag{2.26}$$

whereas $\oplus$ denotes an operation which combines the descriptors of regions respectively pixels and $\psi(p)$ is a function that initially computes the descriptor for the additional pixels.
Assuming this property, the descriptors for all *ER* can be computed by successively increasing the threshold t and using the respective preceding descriptors. Only the ones from the last step have to be stored, which decreases the storage footprint.

Neumann and Matas proposed the following descriptors:

Area a. The cardinality of pixel in an *ER*. Thus, $\psi(p) = 1$ for all p and $\oplus$ denotes a simple addition.

Bounding box $(x_{min}, y_{min}, x_{max}, y_{max})$. The initial function $\psi(p)$ for pixel with coordinates (x, y) returns the quadruple $(x, y, x+1, y+1)$ and the combination function $\oplus$ denotes a min/max operation respectively. From this descriptor width w and height h are derived.

Perimeter p. The boundary length of an *ER*. Depending on the adjacency of a newly added pixel $\psi(p)$ is computed by

$$\psi(p) = 4 - 2n_a, \tag{2.27}$$

whereas n_a is the number of adjacent pixels having less or equal value i.e. belonging to the region itself. The operation $\oplus$ again is an addition.

Euler number η. The Euler number (genus) of an binary image is the difference between the number connected components and the number of holes. A simple algorithm [70] counts specific 2×2 pixel patterns often referred to as *quads*:

$$\mathbf{P_1} = \left\{ \begin{matrix} 1 & 0 \\ 0 & 0 \end{matrix}, \begin{matrix} 0 & 1 \\ 0 & 0 \end{matrix}, \begin{matrix} 0 & 0 \\ 0 & 1 \end{matrix}, \begin{matrix} 0 & 0 \\ 1 & 0 \end{matrix} \right\} \tag{2.28}$$

$$\mathbf{P_2} = \left\{ \begin{matrix} 0 & 1 \\ 1 & 1 \end{matrix}, \begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix}, \begin{matrix} 1 & 1 \\ 1 & 0 \end{matrix}, \begin{matrix} 1 & 1 \\ 0 & 1 \end{matrix} \right\} \tag{2.29}$$

$$\mathbf{P_3} = \left\{ \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}, \begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \right\}. \tag{2.30}$$

If n_1, n_2, n_3 are the number of the occurrences belonging to the particular quad group P_1, P_2, P_3 respectively, then the Euler number is computed by

$$\eta = \frac{1}{4}(n_1 - n_2 + 2n_3). \tag{2.31}$$

Therefore, the initialization function $\psi(p)$ simply needs to count the number of changing quads $\Delta n_1, \Delta n_2, \Delta n_3$

$$\psi(p) = \frac{1}{4}(\Delta n_1 - \Delta n_2 + 2\Delta n_3) \tag{2.32}$$

and the combining function $\oplus$ again is reduced to a simple addition.

Horizontal crossings $\mathbf{c_i}$. The horizontal crossing $\mathbf{c_i}$ is obtained by counting the transitions between region and non-region along a horizontal line at row i of the region. The function $\psi(p)$ just needs to examine the two horizontal adjacent pixels and $\oplus$ is a element-wise vectorial addition.

Given this feature descriptors, a Real AdaBoost [73] classifier with decision trees is employed to estimate the probability $p(character|\mathcal{Q})$ of region $\mathcal{Q}$ being a character. Incrementally the threshold is increased, descriptors are computed and only *ERs* corresponding to a local maximum of $p(character|\mathcal{Q})$ are passed to a second classification step. More expensive but more distinctive features in combination with a Support Vector Machine (SVM) classifier are used to finally determine whether a region is a character or not. Supplementary to the features mentioned above the following ones were proposed:

Hole area ratio a_h/a. a_h denotes the number of pixels belonging to pixel holes.

Convex hole ratio a_c/a. a_c denotes the number of pixels within the convey hull of the area.

Number of outer boundary inflexion points κ. The number of transitions between convex and concave angles of the boundary pixels of a region. Compared to other regions this number is usually quite small considering characters.

By now, only dark text on white background was considered. In order to detect the opposite too, the process is simply repeated for an inverted input image. Further, the proposed features are scale-invariant but not all are rotation invariant. To detect multiple oriented text as well, it is proposed to rotate the image step wise.

As already mentioned, in a final exhaustive search [61] step the *ERs* still classified as characters are grouped into words[4].

2.3 Text Recognition

Text recognition is the task of converting an image representation of text into a string. In research (scene) text recognition is often done employing existing and commercial OCR

[4] *ER*/*MSER*-based text *tracking* is described in Section 3.1.1

solutions [10, 32, 33]. Recently, words usually are the central focus of most recognition approaches. Due to their compactness they are easy to segment and group. Further, high level priors can be introduced by statistically analyzing the morphological and lexical structure of speech [55, 75, 87].

Nevertheless, text recognition relies on a proper segmentation. If not already provided by a prior text detection, a particular dedicated step has to be appended e.g. Saidane and Garcia [72] introduced a *CNN*-based binarization to robustly segment text from complex background.

Contrary to these segmentation-based word recognition methods, the field of *word spotting* tries to spot a word out of a certain set without any character segmentation [48, 84].

Recently, some research focused on designing and learning *CNNs* capable of processing words as a whole [24, 25]. In [3] *CNNs* are used to project text images as well as embedded labels into a common subspace where they are comparable.

2.4 Summary

In this chapter we reviewed text detection and recognition approaches with the focus on scene text in single images or individual frames. We summarized the occurring problems, which are associated with the great variety of text in natural scenes as well as approaches to solve them.

Possibilities to rectify perspectively distorted text were stated, either based on vanishing point or rectangular region detection. Further, we resumed the two main categories regarding text detection: *CCA* methods, which try to associate and segment possible text regions exploiting their common properties e.g. color/intensity or stroke-width and *sliding window* methods, which examine the texture within a search window to determine whether text is present or not. Finally, we summed up text recognition methods especially designed for scene text.

3 Text Detection, Tracking and Recognition in Videos

Contents

3.1 Text Tracking 23

3.2 Multiple Frame Integration 28

3.3 Summary 30

Hereinafter, we will extend the considerations of Chapter 2 to video footage. Obviously, a lot concepts apply to multiple frames as well. Nevertheless, the temporal redundancy in videos opens possibilities to more enhanced methods which are more robust and may process text even faster, especially under challenging circumstances. Typically text is detected and tracked over time. As a result multiple images of the same text are available. They can be used to create a single more legible image of the text or to provide multiple recognition results. Either way, there is the opportunity to decrease the number of false recognitions.

Additionally to scene text, videos often contain caption text, i.e. artificially overlaid text especially occurring in television and movies. In general such texts are more legible and easier to track due to the restricted motion. In the following, the possible approaches in accordance to nowadays research are reviewed.

3.1 Text Tracking

The aim of text tracking is to follow the position of text in consecutive frames within video streams. Needless to say, standard video tracking approaches are often applicable to text as well. Thus, their categorization can be done similar. Usually methods are divided into tracking with *template matching*, *particle filtering* or *tracking-by-detection*. In the following, we will give a short overview of possible approaches in the context of video text.

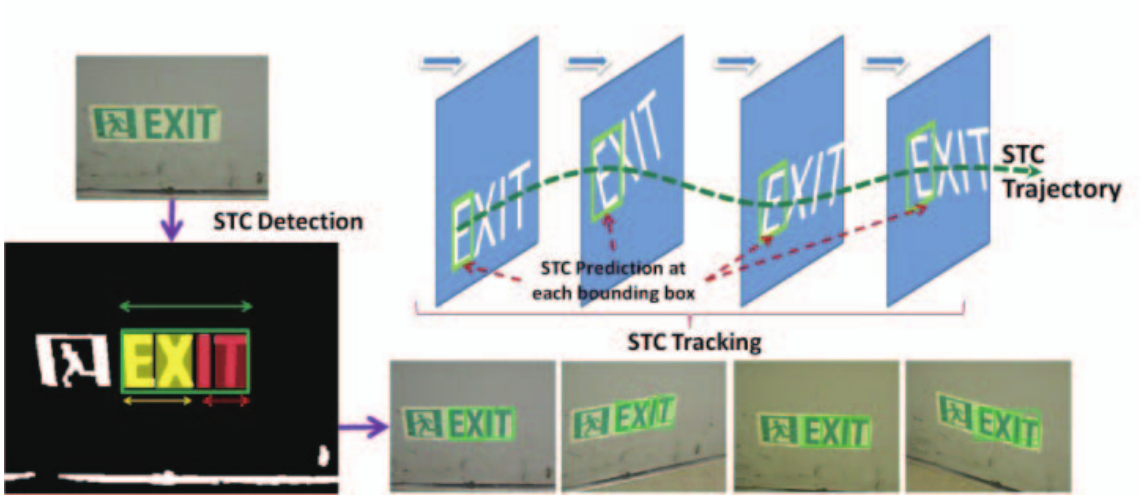


Figure 3.1: Schematic illustration of the scene text detection and tracking framework by Rong et al. [71]: First text regions are extracted by grouping together Maximally Stable Extremal Regions (MSERs) due to size and alignment. Then each Scene Text Character (STC) is tracked independently. The estimated trajectories are used to improve text recognition. Image taken from [71].

3.1.1 Template Matching

Template matching basically tries to find the same feature under a different view in following frames. It searches for the most similar region in a frame given an image patch. Therefore, a feature representation and a similarity measurement is essential.
The feature extractor and descriptor are the most important parts of trackers [85]. Since they have plenty of applications, various well researched and widely used feature representations are available nowadays. Nevertheless, in the following we will examine only those applied in the context of text tracking. For a general summary we would like to refer to a more detailed survey e.g. [81].

Early work often focused just on caption text to categorize and semantically investigate video data. This overlaid text most likely is static or moves linearly e.g. the end credits of a movie. Even if it is static it is helpful to regard small translation for a few pixels. Lienhard and Stuber [43] matched the entire text region with its surrounding in subsequent frames using Mean Absolute Difference (MAD). Li et al. enhanced this approach by postulating the position of uniformly moving text to decrease the necessary search radius [38]. Later on [39] they achieved a speed up and sub-pixel accuracy by means of an image pyramid and added a contour based stabilization step to perform long-term tracking [40]. As a result of considering all sorts of scene text as well, more advanced motion have to be utilized. Text most often is written on nearly planar surfaces and hence, depending on the camera movement can experience various perspective distortions.

Na and Wen [59] used Scale-invariant Feature Transform (SIFT) [46] to extract features in an region around the detected text. They perform Nearest Neighbor (NN) matching to establish correspondences and handled motion estimations up to similarity transforms.

To eliminate outliers they analyzed the relative distances between feature points in the reference frame and assume that the corresponding distances should scale uniformly in the target frame. Phan et al. [68] introduced a scheme called *SWT-SIFT*: Given a bounding box around detected text they performed a Stroke Width Transform (SWT) [13] to obtain the text's edges and computed *SIFT* feature descriptors at fixed scale and each of the resulting pixels. Thereby, the number of extracted features was increased on low resolution video text. The homography estimated via Random Sample Consesus (RANSAC) was refined by sliding the *SWT* mask over the result.

Figure 3.2: SWT-SIFT scheme by Phan et al. [68]: They compute *SIFT* [46] descriptors at each pixel of the text's edges obtained by *SWT*. At the top the comparison of the amount of extracted features using native *SIFT* (a) and SWT-SIFT (b) is illustrated. At the bottom the *NN* matching of SIFT (c) and SWT-SIFT (d) is compared. Images taken from [68].

Fragoso et al. [16] assumed text on nearly planar surfaces and performed short-term tracking on a mobile phone using Efficient Second Order Minimization (ESM) [6]. *ESM* iteratively minimizes the difference between a reference and a target frame assuming a projective transformation. Since, it is costly for a large intra-frame movement, they assume a sufficiently smooth motion of the camera. Yusufu et al. [94] used Speeded Up Robust Features (SURF) [5] and Fast Library for Approximate Nearest Neighbors (FLANN) [57] to extract and match features within caption text regions efficiently. Instead of *RANSAC* an efficient histogram based algorithm was introduced to remove false matches.

Hartl et al. [20] assumed text on a planar rectangular region and computed an initial pose using BRISK features [36]. This information is passed to a highly optimized patch-based tracker [82]. The proposed method was able to run in real-time on State-of-the-Art mobile phones.

Since *MSER* experience great acceptance as basis for text detection (see Section 2.2.2.2) Gómez and Karatzas [17] adapted the efficient *MSER* tracking algorithm by Donoser and Bischof [12]. Compared to full feature matching, the speedup is achieved by searching only in a small window and looking only into a subset of levels in the component tree. However, two changes to the original tracking module are proposed to exploit text specific properties and thus enhance the tracking performance: Invariant moments are used as features to increase the robustness compared to the originally proposed simple but efficient features

e.g. gray value or region size. Additionally, instead of single *MSER*s entire text lines are considered. Hence mismatches, which are not compliant with a respective dominant line model can be rejected. The proposed approach is suitable for running on restricted mobile hardware.

3.1.2 Particle Filtering

Particle filtering, also called *Sequential Monte Carlo method*, is a filtering technique estimating a dynamic process's hidden state based on available observations. It solves the problem of handling non-linear process models or not normally distributed parasitic errors. Thus, it is popular for vision based object tracking. With particle filters the conditional state density $p(\mathbf{x_t}|\mathbf{y_t})$ is represented by a set of N weighted particles $\left\{(\mathbf{x_t}^{(1)}, w_t^{(1)}), ..., (\mathbf{x_t}^{(N)}, w_t^{(N)})\right\}$. Considering a single particle $\mathbf{x_t}^{(n)}$ at time t, its weight $w_t^{(n)}$ is adapted proportionally to the probability of the observation $p(\mathbf{y_t}|\mathbf{x_t}^{(n)})$. To put is simply, the iterative process of particle filtering for text behaves as follows: The amount and the weights of the particles represent the probability of a certain state i.e. text position. In each step the particles are updated due to a certain dynamic model and weighted based on an observation. Finally, particles are re-sampled, i.e. it is ensured that $\sum_{i=0}^{N} w_t^{(i)} = 1$ and strong particles are replaced by multiple equally weighted ones. Figure 3.3 schematically illustrates this process.

As for every tracker particle filtering, a sufficient robust feature representation is necessary. Regarding text there has been research using projection profiles [50, 51], cumulative histograms [18, 77] and Histogram of oriented gradients (HOG) [54].

Further, particle filtering requires a dynamic model, i.e. the definition how particles' current states are predicting depending on their previous state. In [18, 77] just the velocity was taken into account and the particle were scattered around the predicted center to bear the uncertainty in mind. Merino et al. [50, 51] stated that the text movements are too unpredictable and hence, utilized a constant position prediction model in combination with an uniform and Gaussian random walk strategy.

Finally, it has to be defined how the particles weights are influenced by observations. Tanaka and Goto [18, 77] weighted a particle with a measurement denoting the similarity of the previous and current text block, provided the particle remains onto the text block, otherwise it is discarded. Merino et al. [50, 51] compared the projected features' position with the actual text components found in the image. Minetto et al. [54] exploited the Battacharyya's similarity coefficient of the respective *HOG* descriptors.

3.1.3 Tracking-by-Detection

Tracking-by-detection methods detect text in each single frame separately and match the detections. Therefore, different type of text features are used like color, position or gradient

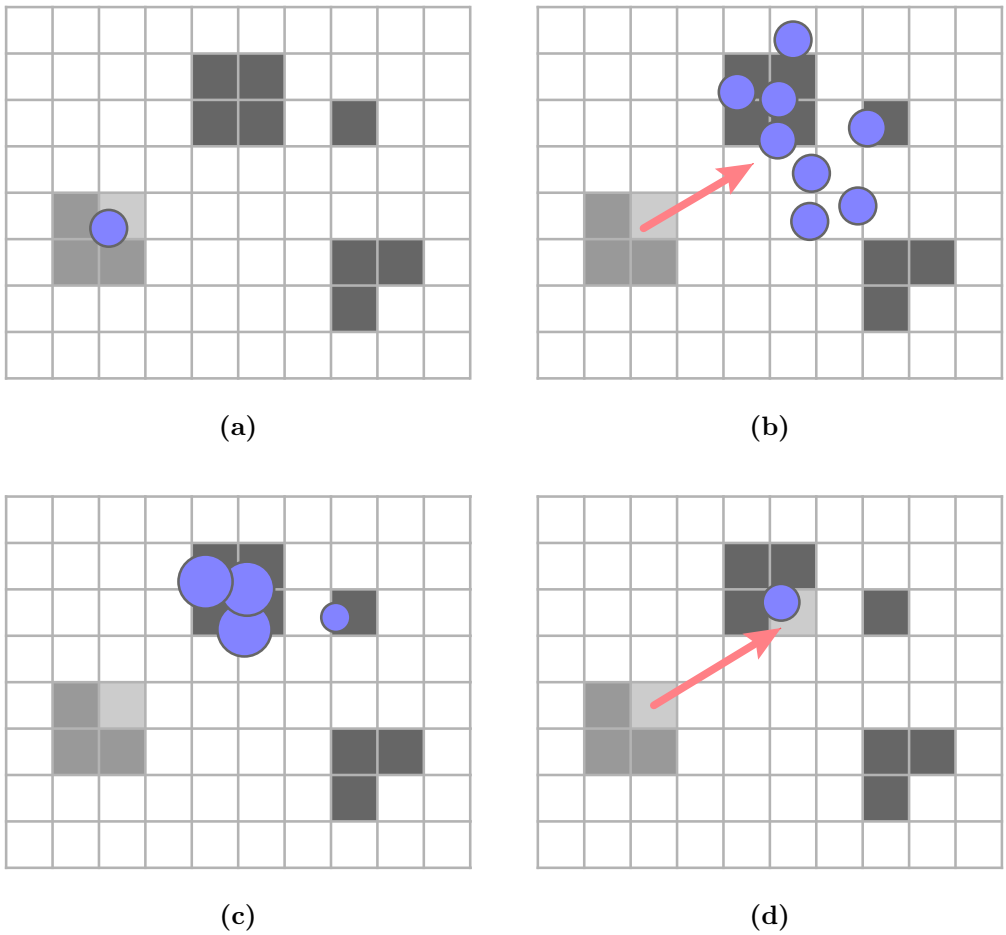


Figure 3.3: Tracking using particle filtering is illustrated: In (a) a particle lies on a text block pixel set from the previous frame (light gray). The particle is scattered due to a dynamic model (b), weighted on the basis of an observation (c) and resampled (d). Thereby, an association from previous to current text block pixel can be established.

histogram. Since text detection tends to be computationally more expensive these methods are usually not real-time capable. However, *Tracking-by-detection* approaches solve the re-initialization problem after losing track of an object. Lienhard and Effelsberg [42] matched characters detected in different frame with similar features, moving speed and time span during they appear. Wolf et al. [88] performed a simple method. They matched bounding boxes of the detected text by their overlapping ratio over consecutive frames.

Mi et al. [52] assumed non-moving caption text i.e. subtitles and thus, actually did not perform tracking in the proper sense. Nevertheless, they detect whether the text has changed by exploiting the region and the edge overlap.

Rong et al. [71] tracked each character independently by searching for the best matching *MSER*. An motion trajectory is estimated and utilized to lead the text detection

module in subsequent frames and mitigate the effects of motion blur. This process is illustrated in Figure 3.1.

3.2 Multiple Frame Integration

Scene text in video footage often experiences low resolution, color bleeding or light reflections. However, due to poor video quality, even the usually better legible caption text is sometimes hard to read. Common text recognition under this circumstances is error-prone. Nevertheless, the text information in videos is highly redundant. Thus, multiple frames can be exploited to integrate the required information and increase the probability of getting a correct result. Basically a distinction can be made between *recognition result fusion* and *image enhancement* methods. Figure 3.4 illustrated the difference between these two categories.

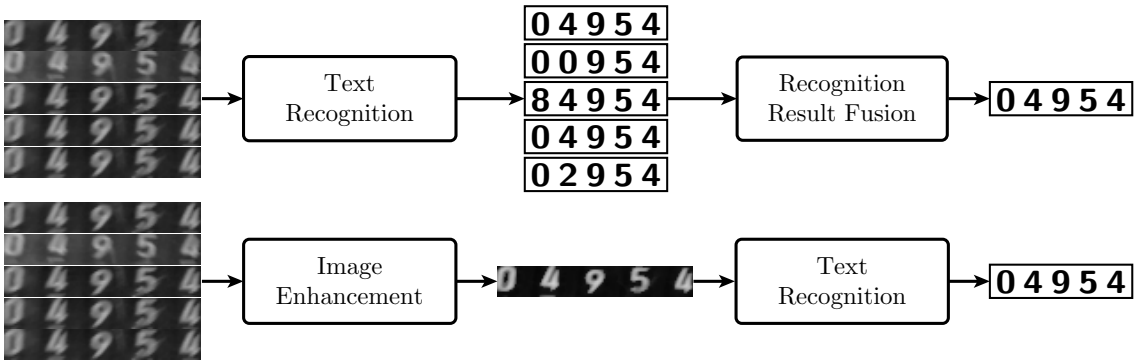


Figure 3.4: The two categories of Multiple Frame Integration (*MFI*): *recognition result fusion* methods (top) perform text recognition on each frame separately and fuse the resulting text; *image enhancement* methods (bottom) merge the frames itself to get a single more legible image to perform text recognition on.

3.2.1 Recognition result fusion

Recognition result fusion methods try to recognize the text in each frame separately and fuse/compare the gathered text to reject false recognitions and get a final text string. In [43] simply the most frequent recognition result was taken.

Mita et al.[56] grouped redundant recognized characters by position and size and introduced a voting system based on the recognition certainty of the OCR engine to obtain a likely result. It considered that multiple characters can be spuriously recognized as a single one or allegedly characters are added.

Additionally to majority voting procedure regarding each character, Rong et al. [71] proposed a second fusion method. They employed a Conditional Random Fields (CRF) model to constraint the intermediate result under lexical aspects.

Greenhalgh et al. [19] recognized text-based traffic signs and reviewed the ten most recent recognition results. Entire words are weighted by their recognition certainty and grouped together if recognized equally. The resulting histogram is used to obtain the most probable text.

3.2.2 Image Enhancement

Approaches in the first category merge the frames itself to get a single more suited image to perform text recognition on. Depending on the situation this could mean that the result should be without any reflections or obstacles and better readable. A simple technique is averaging and binarizing the multiple text masks to get more reliable one [39]. Wolf et al. [88] scaled up each related text image using a modified bilinear-interpolation before averaging and binarizing them to obtain an integrated output. An additional weighting factor which depends on the temporal mean and standard deviation of a pixel increased the robustness against outliers during the interpolation.

Zhen et al. [95] split up the text box into smaller boxes and averaged only the clearer ones. Therefore, they detected and counted corners as contrast measurement. Instead of averaging, Zhou et al.[96] chose the minimum/maximum intensity value of all corresponding pixels, depending on whether the text is light on dark background or vice versa. Unfortunately, this method is vulnerable to noise.

Mi et al.[52] mitigated this problem by adding an edge based integration step: Canny edge detection is performed on each temporal instance of a text region. All edge images are averaged, thresholded and morphological operations are used to improve the output. Each edge in the result is associated with the corresponding region in the averaged text mask. The number of edge pixels relatively to the number of mask pixels is used to reject small non-text regions produced by noise. However, since the Canny algorithm returns edges which do not exceed the width of one pixel, this approach obviously assumes strictly static video text.

Yi et al.[91] filtered frames with blurred text before integration. They divided the pixels p of the output image into text t_{text} and background t_{back} by locally thresholding the average of all frames using the Otsu method [66]. Afterwards they are integrated separately. While considering bright text on a dark background, the text/foreground is averaged and a minimum operator is applied to the background:

$$t_{int}(p) = \begin{cases} min\{t_i(p)\} & p \in t_{back} \\ \frac{1}{M}\sum t_i(p) & p \in t_{text} \end{cases}, \tag{3.1}$$

$$t_{text} = \{p|t_{avg}(p) > H_{otsu}\},\ t_{back} = \{p|t_{avg}(p) \leq H_{otsu}\}. \tag{3.2}$$

$t_1, t_2, ..., t_m$ are the already filtered frames, whereas M denotes their total number. H_{otsu} is the local Otsu threshold calculated in t_{avg}.

In [45] and [94] a simple AND was applied to consecutive masks showing the same

caption text. It filtered contradictory false positive pixels and retained stable ones.

3.3 Summary

In this chapter the concepts from scene text detection in still imagery were extended to videos. The enhancements essential while considering video text were discussed. To exploit the redundant information and improve robustness of text recognition multiple instances of the same text have to associated. Therefor, different text tracking approaches were summed up. Further, the information fusion process, often referred to as *Multiple Frame Integration (MFI)*, was treated.

4 A Pipeline for Scene Text Processing on Mobile Devices

Contents

4.1 Pipeline 31
4.2 Requirements 38
4.3 Summary 38

In this chapter we are going to explain our proposed pipeline in order to detect, track, integrate and recognize text on mobile phone hardware. It exploits multiple threads in order to outsource the time consuming text detection module while tracking and recognizing already detected text. The modular concept allows to exchange most of the parts with little effort.
Further, our method was designed to operate without any user intention, by means of tap gestures or manual settings. In other words it is capable of processing text in the described manner just by aiming the camera into the right direction. In the following we will first give an overview of our pipeline and afterwards describe each part in detail.

4.1 Pipeline

Without constraining practical use cases we make a few assumptions: (1) We postulate that the text is written on a nearly planar surface (2) which is sufficiently textured for feature-based tracking. Due to the text, this is almost always the case. (3) While the short processing phase, we assume an adequately smooth motion of the camera i.e. a user who does not shake the device.

Since nowadays mobile phones almost all own at least dual core processors, we use two threads to process scene text. The main thread is responsible for tracking, Multiple Frame Integration (MFI) and the actual recognition. The second thread performs text detection, hence hereinafter referred to as *text detection thread*.
As soon as the camera is focused and aimed to a structured surface, tracking is triggered

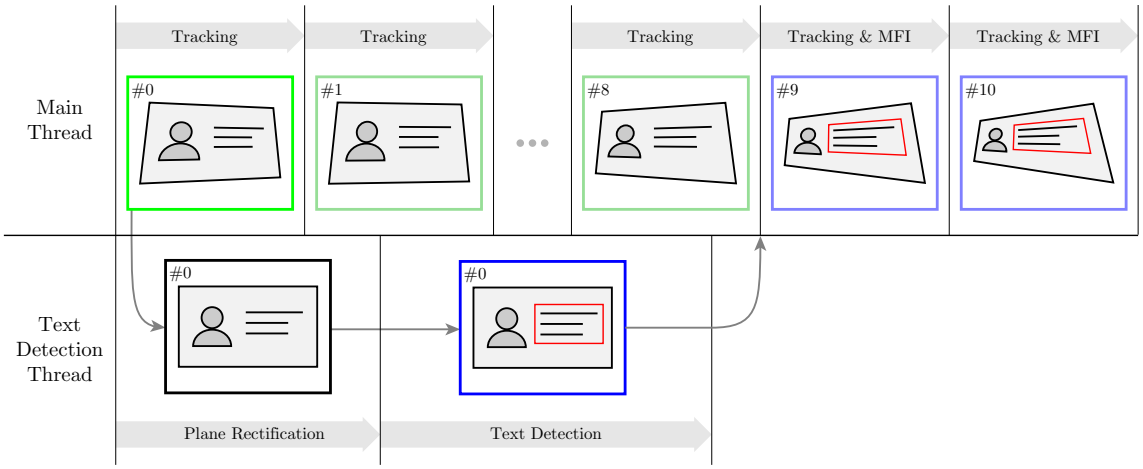


Figure 4.1: A schematic illustration of the pipeline process during initialization. Whereas in the main thread the planar target is tracked in each frame, the rectification and the text detection are processed in a parallel text detection thread. Before each tracking cycle, it is examined whether the text detection is already done. If so, the information telling where the text is located can be exploited to perform Multiple Frame Integration (*MFI*) and text recognition. The color coding indicates the the respective keyframe for tracking.

and estimates the motion of the text plane in each frame. The first frame is considered as keyframe and each frame is registered to it by computing the associating projective transformation.

Simultaneously, the same frame is passed to the text detection thread. Parallel to tracking the keyframe is warped such that the visible underlaying plane is rectified and subsequently examined for text. Between the tracking cycles, it is checked whether the text detection has finished. As soon as it returns, the now rectified frame is considered as the new keyframe. Due to the known homography between the different views on the plane, we now can deduce the text position in each frame and start the *MFI* as well as the text recognition. Figure 4.1 schematically illustrates the entire initialization process and Figure 4.3 shows an actual example.

If the tracking is lost, we reset the pipeline and repeat the described initialization phase. Even without losing track, we frequently perform text detection to introduce more current keyframes.

In the following, the individual modules of our pipeline are described in detail. An overview is given in Figure 4.2.

4.1.1 Visual Tracking

Our pipeline does not track the text itself, rather the planar surface it is written on. Thus, the motion is modeled with a homography. This means that the tracker returns a homography describing the transformation from a current frame to a keyframe. In other words we *register* the frames without actually transforming them.

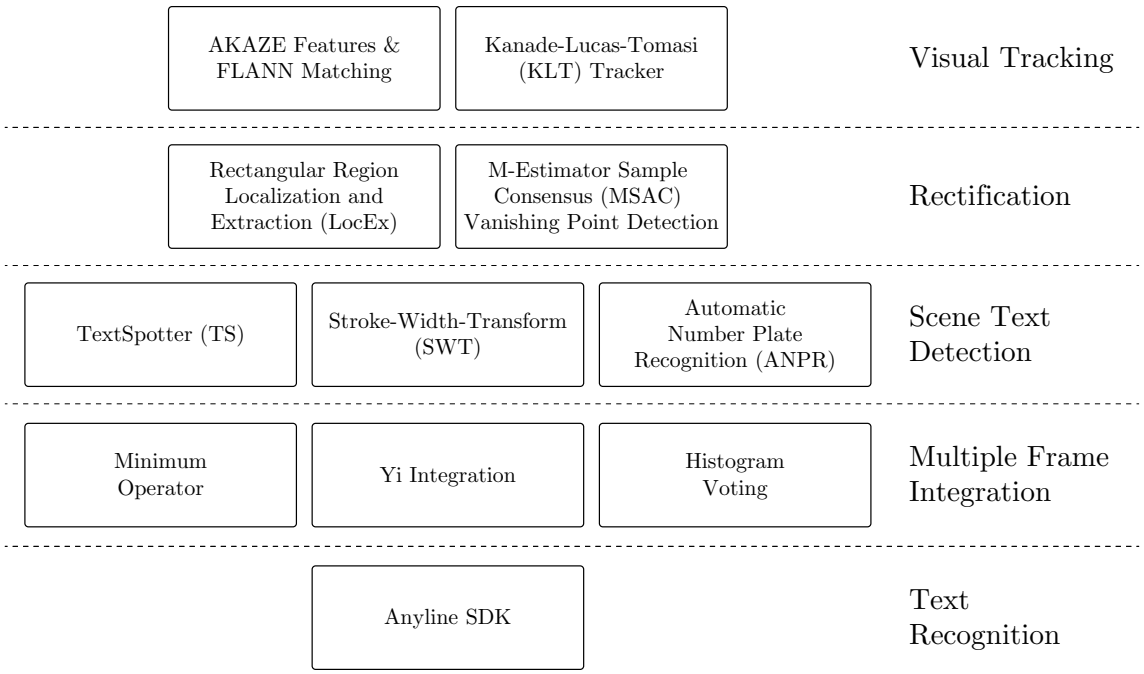


Figure 4.2: The different modules of our pipeline utilized to perform rectification, text detection, tracking and multiple frame integration.

In order to not mitigate the user experience we tried to minimize the initialization phase of the tracker. In our tests we utilized solely feature based tracking methods. Therefore, this phase consists only of a quick feature extraction. More specifically, we integrated a pyramidal implementation of the well known Kanade-Lucas-Tomasi (KLT) feature tracker [47, 76, 79]. It tracks a set of keypoints over time, which we exploit to estimate the homography to a certain keyframe. Additionally, we evaluated a tracking approach based on AKAZE features [1, 2]. In each frame features are extracted and matched with the ones from the keyframe using the Fast Library for Approximate Nearest Neighbors (FLANN) by Muja et al. [57].

To work without any user intention, we need an indicator which triggers tracking. Compared to other approaches [17], we do not wait until the text detection module finishes. This has the advantage that at the time of a successful text detection we already have processed and registered several frames and can use them to recognize the showed text. Thus, we need to know when the camera has finished focusing and is held still or just moved smoothly. A common focus measurement applied to a short window in center of the current frame covers both. We used the *Tenengrad* metric, which simply averages the intensity gradient within the region of interest.

We store the obtained metrics for the last few frames and analyze slope and curvature of the resulting discrete function to determine whether the focus is on a local maximum. Ad-

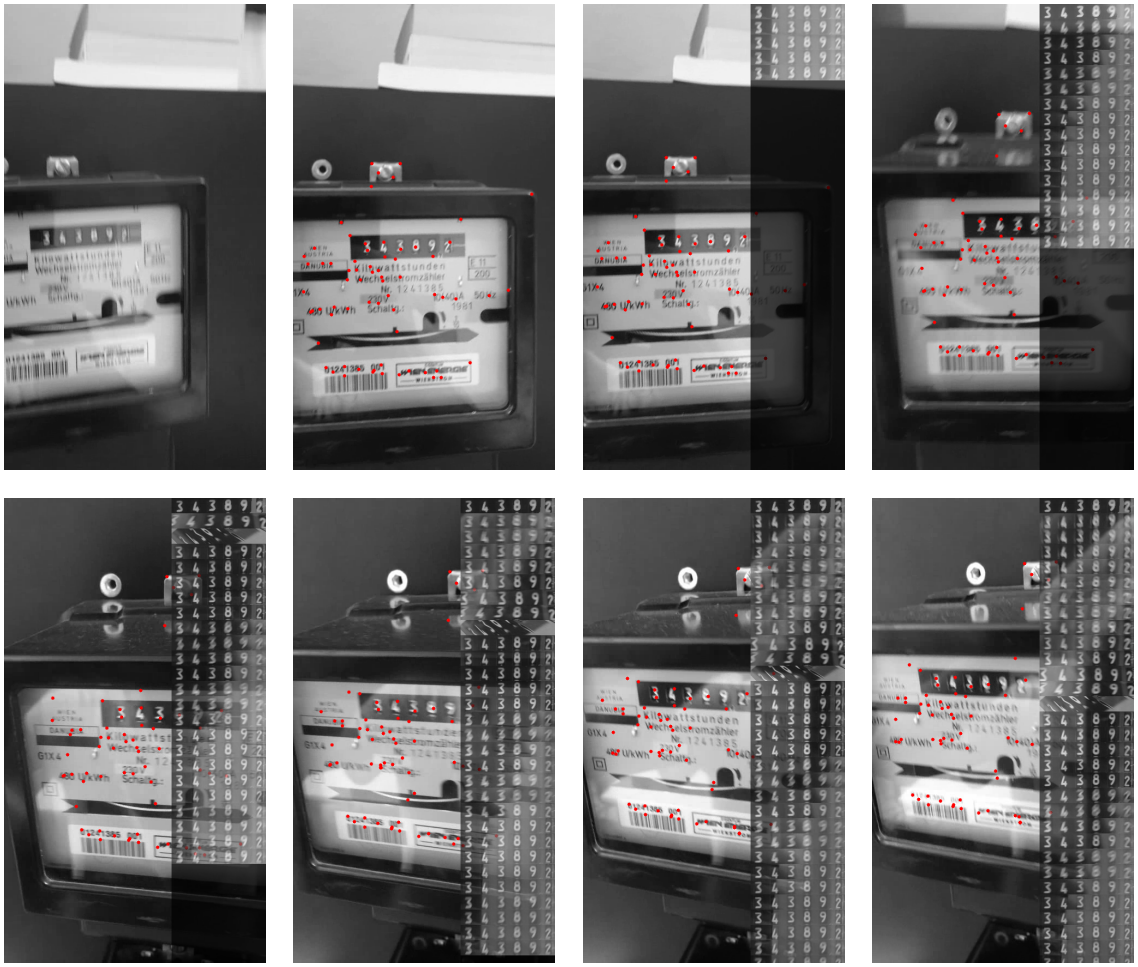


Figure 4.3: Example of the processing of the pipeline: As soon as the camera focuses, tracking and text detection is triggered. When text detection is finished the extracted frames are propagated to *MFI*. The debug output shows the integrated text patch in the first row and the respective single extracted patches underneath. Single outliers due to imperfect tracking do not disturb the process.

ditionally we exploit an absolute threshold, which avoids tracking on continuously blurred frames or almost non-textured scenes. The resulting algorithm behaves intuitively and coincides well with user intentions. As soon as the camera focus is adjusted tracking is triggered and the first frame is considered as keyframe. This frame is also passed to the text detection thread. Figure 4.1 illustrates this process.

Tracking can obviously fail, when the target moves out of the camera view or the motion becomes to strong. It is essential to robustly detect when this happens in order to pause tracking or trigger reinitialization. Regarding feature based tracking one obvious measurement is the amount of features still considered as tracked. Nevertheless, they can drift away from their dedicated position and a proper functioning tracking is the basis for the subsequent *MFI*. Depending on the used approach, the final result can be rendered

unusable due to a single outlier. Further, valuable processing time can be saved, if outliers are detected immediately. Therefore, we warp a rectangle, which has the same dimensions as a video frame and evaluate the resulting quadrilateral.

First of all, we test whether the quadrilateral is still convex by trying to find the diagonals intersection. If the two line segments still intersect, the homography preserves the convexity of the rectangle's hull and therefore, is said to be *quasi-affine* with respect to the corners of the rectangle. Thus, the homography does not map any of the points lying on the border of the rectangle to the plane at infinity. This would lead to a degenerated image that is unlike any possible view seen by a camera [21].

(a) **(b)** **(c)** **(d)**

Figure 4.4: To ensure that the tracking and the image rectification work properly, we evaluated the particular resulting homographies using an relative area measurement: In the event of success the area of a rectangle covering the entire frame (a) does not change dramatically due to the transformation (b). If the homography is becoming degenerated, the influence on the area will be more severe (c). In order to avoid transformations like shown in (d), we additionally checked whether the resulting quadrilateral is still convex.

Second, we compute the area of the resulting quadrilateral. If the relative alteration is to high, we will consider the homography as unusable as well. Empirical tests showed us that this approach leads to the clearest distinction. Additionally, the accompanying slight restriction of scaling, in other words the motion away from and towards to the target is in the sense of the application: Pausing tracking when the camera is intentionally moved away from the target is reasonable. Figure 4.4 shows the rectification in case of success and failure.

4.1.2 Rectification

The rectification (see Section 2.1) can be considered as preprocessing step for the text detection and is processed in the same thread as illustrated in Figure 4.1. It transforms the

current frame such that the dominant planar surface is metrically rectified (see Figure 2.3). In most cases this means that text is horizontally aligned within the warped image. We integrated two selectable methods described in Section 2.1.1 for estimating the essential homography.

First, we utilized the rectangular region extraction approach from Hartl et al. [20]. It assumes a dominant distinctive rectangular region within the camera view. Although, it operates very well under this assumption, it is not useful, if text is to small in relation to the rectangular region. Intuitively, the camera is held too close to the target, the borders of the rectangle are not visible and the Hough Transform (HT) based approach fails.

Therefore, we exploit a second approach based on the vanishing point detection by Nieto et al. [65]. The methods detects directional image features and robustly estimates the two required vanishing points. With this, it is easily possible to compute the required homography [41].

In exactly the same manner as described for the tracking in Section 4.1.1, the viability of the resulting homography matrix is estimated. Since text detection is one of the most time consuming parts of our pipeline it is reasonable to avoid performing it on a frame warped by a degenerated homography.

4.1.3 Scene Text Detection

The rectification step returns a transformed frame such that the text is horizontally aligned. This assumption allows to speed up most of the text detection methods. They achieve rotation invariance with great effort respectively processing time, e.g. by searching for text in several rotated versions of the input image. Through the rectification we can omit this additional work.

Several different text detection modules were utilized in our pipeline: First of all the Stroke Width Transform (SWT) operator[1] by Epshtein et al. [13] described in Section 2.2.2.1. Second, we integrated the approach of Neumann et al. [62] often referred to as TextSpotter 2013 (TS), based on the classification and grouping of Extremal Region (ER) described in Section 2.2.2.2. Finally, we added a simple but effective method based on the Automatic Number Plate Recognition (ANPR) described in [4], which detects text by finding close edges with the help of morphological operations.

The output of the text detection module is at least a bounding box surrounding the text. If available we also store the pixel wise segmentation, which dramatically can speed up and/or improve text recognition. It occurs that some detection methods return overlapping bounding boxes indicating parts of the same text. We merged such bounding boxes depending the overlap metric

$$overlap(\mathcal{X}, \mathcal{Y}) = \frac{|\mathcal{X} \cap \mathcal{Y}|}{min(|\mathcal{X}|, |\mathcal{Y}|)}. \tag{4.1}$$

[1] Implementation from https://github.com/aperrau/DetectText

Further, predefined use cases often allow to introduce some restrictions for the text, e.g. if it is required to detect just the meter reading in Figure 2.3. Therefore, we implemented several heuristics to filter non-required text based on dimension, area, aspect ratio and color histogram of the bounding box respectively the pixels within.

4.1.4 Multiple Frame Integration

The previous parts of our pipeline provide multiple cropped frames, showing an orthogonal view of the same text (see Figure 3.4). Obviously, several different text parts can be detected, *MFI* is applied on each.

We built in three different *MFI* methods. First of all, we employed the simple result fusion approach also used by Greenhalgh et al. [19]. It groups equally recognized results and weights them by their recognition certainty. The emerging histogram is used to determine the most likely text.

Second, we applied a simple minimum operator to all extracted and registered image patches simply returning the lowest valued pixel at the respective positions.

Finally, we exploited the method first introduced by Yi et al.[91]: It performs a minimum and an average operator each on all frames. The averaged image is thresholded using Otsu's method [66] in order to distinguish between text and background and the latter is replaced by its minimized pendant. Thereby, the contrast is increased (see Section 3.2.2 for more details).

Several experiments with common Super Resolution approaches failed, since the underlying optimization techniques retained the major complicating interferences like reflections.

Like most *image enhancemening MFI* methods, they are vulnerable to jitter respectively slight movement due to imperfect tracking. Therefore, we added the possibility to enable a registration refinement step based on the research of Evangelidis and Psarakis [14]. They proposed an iterative l_2-based algorithm for parametric image alignment. They maximize the so-called Enhanced Correlation Coefficient (ECC), a measurement which is robust against geometric and photometric distortions. To speed up this process, we pass the homography provided by the tracking as initialization and limit the number of iterations. If the *ECC* value is still too low after this optimization step, we have the possibility to omit the frame during the actual *MFI*.

4.1.5 Text Recognition

For our evaluation we used a text recognition module based on the Anyline SDK[2]. Nevertheless, the available debug version is not dedicated for real-time applications and therefore, only used to provide a meaningful evaluation of the remaining modules. The commercial OCR engine already embeds several preprocessing steps assuring proper text segmentation customizable to the required use case. Thus, it properly processes any cropped

[2]https://www.anyline.io/

input image without any prior segmentation. The SDK's core is written in C++ to provide support on all major platforms.

4.2 Requirements

Our pipeline is a CMake[3] project written in C++11 and is based on *OpenCV 3.1* and its *Contrib* Modules. We ported the core to Android NDK utilizing the Toolchain provided by OpenCV[4]. Other external code was used for the *M-Estimator Sample Consensus (MSAC) based vanishing point detection* [65], the *SWT operator*[5] and the *rectangular region localization and extraction* module by Andreas Hartl [20].

4.3 Summary

In this chapter, we described our proposed pipeline to efficiently process text on mobile phone hardware. Due to the usage of multiple threads, we achieve a speed up which is necessary for real-time capable applications. We depicted each part/step of the pipeline and the thread communication in detail. Furthermore, we summarized approaches which make it possible to get along without any configuration or user interaction beside aiming the camera into the right direction.

[3] https://cmake.org/
[4] http://opencv.org/platforms/android.html
[5] https://github.com/aperrau/DetectText

5 Impact of MFI on overall Text Processing

Contents

5.1 Configuration 39
5.2 Detection and Tracking Accuracy 40
5.3 Reading Accuracy 44
5.4 Algorithm Runtime 45
5.5 Summary 47

In this chapter we are going to evaluate our pipeline and its impact on the overall text recognition results. Therefore, we assumed the use case of recognizing energy meter readings. The usually present glass plate in front of the meter reading is prone to reflections and usually mitigates text recognition results.

Furthermore, it is a reasonable application for scene text detection on mobile hardware. Power companies can save a considerable amount of money by speeding up the reading process of analogous energy meters. Even a bigger benefit is gained in customer support. Human mistakes can be minimized and by storing the respective frames a verifiable evidence of the meter reading is available for free[1].

5.1 Configuration

We tailored our pipeline to solely detect and recognize energy meter readings and omit the remaining text on the meter. Thus, we introduced additional constraints on the extracted text patches respectively their bounding boxes to reject non-relevant text. First, we limit ourselves to meter readings with white text on black background. Therefore, we adjusted the text detection accordingly (see Section 2.2.2). The digits of the meter readings are

[1] https://www.anyline.io/energy-anyline-io-en/

almost the only bright text on dark background. Second, we set upper and lower limits of the aspect ratio and the area[2].

Further, we manually extracted multiple meter readings of different meter types and computed an average intensity histogram H_R. We correlate it with each text patch's histogram H_P:

$$d(H_P, H_R) = \frac{\sum_I (H_P(I) - \bar{H_P})(H_R(I) - \bar{H_R})}{\sqrt{\sum_I (H_P(I) - \bar{H_P})(H_R(I) - \bar{H_R})}}, \; \bar{H_k} = \frac{1}{N} \sum_J H_k(J). \tag{5.1}$$

N is the number os histogram bins and during the evaluation set to 16. If the correlation $d(H_P, H_R)$ is less than 0.2, we will reject the respective text patch.

5.1.1 Datasets

We recorded several videos with usual mobile phone cameras and resized them in order to obtain different resolutions. We put a focus on varying lighting conditions. Still, they represent reasonable use cases for OCR applications on mobile phones. Especially, while trying to recognize energy meter readings illumination could be poor and the mobile phones flash LED as additional light source may be required. Table 5.1 provides an overview of the different evaluation videos and their specifications.

Video ID	Light source	Camera type	max. Resolution	No. of Frames	Duration
1	Tungsten	iPhone 5S	768x1366	228	00:07
2	Daylight	iPhone 5S	768x1366	209	00:07
3	Flash	iPhone 5S	768x1366	581	00:19
4	Daylight	XPeria Z2	768x1366	1280	00:42
5	Tungsten	iPhone 5S	768x1366	507	00:16
6	Tungsten	iPhone 5S	768x1366	234	00:07

Table 5.1: Evaluation dataset specifications, including the dominant light source.

Figure 5.1 shows representative frames of the evaluation dataset.
We annotated the videos manually with quadrangles surrounding the digits of the meter readings and added its current value. Therewith, we were able to establish a ground truth data for a tracking, detection and recognition evaluation.

5.2 Detection and Tracking Accuracy

We tested the influence of the different pipeline detection and tracking modules on the overall performance. Therefore, we utilized the CLEAR-MOT evaluation framework [7]

[2]The text patch's area is considered relatively to the frame area.

Figure 5.1: Exemplary frames of the evaluation datasets showing different types of energy meters and ground truth annotation. The digits of the meter readings are almost the only bright text on dark background. Reflections due to the front glass and the shiny surfaces, especially in combination with a camera phones flash light can mitigate the Optical Character Recognition (OCR) results.

dedicated to deliver intuitive and meaningful measurements describing the tracking performance of a multi-object tracker.

In the following the framework is briefly described. Subsequently, the experiment results are presented.

5.2.1 The CLEAR-MOT Metrics

The CLEAR-MOT metrics [7] constitute a widely used evaluation framework for multiple object tracking among others for text in videos [17, 29, 30]. They provide a reasonable and intuitive statement on the text detection and tracking performance. In the following, we will briefly describe the two main metrics.

Let o_i^t be an object according to the ground truth and h_j^t be a hypothesis established by the evaluated system each in a single frame at time t. In our case objects and hypotheses are both represented by surrounding bounding boxes and typically are visible over multiple frames. The metrics now rely on a set of mappings (o_i^t, h_j^t), which correspond ground truth objects with detected and tracked hypotheses. In each frame these correspondences are established by maximizing the sum of overlaps

$$overlap(o_i^t, h_j^t) = \frac{a(o_i^t \cap h_j^t)}{a(o_i^t \cup h_j^t)}, \tag{5.2}$$

whereas $a(\cdot)$ denotes a function computing the area of its input polygon. This is a maximum weight assignment problem, which is solved by the Munkres' algorithm [58]. In accordance to [17, 29, 30], a mapping is considered valid only if $overlap(o_i^t, h_j^t) > 0.5$.

Given the overlap measurements of all valid correspondences $\{v_i^t\}_{t,i}$ the Multiple Object Tracking Precision (MOTP) describes the precision of the detection and tracking

$$MOTP = \frac{\sum_{i,t} v_i^t}{\sum_t c_t}, \tag{5.3}$$

whereas c_t is the amount of valid correspondences in frame t.

The Multiple Object Tracking Accuracy (MOTA) is calculated as:

$$MOTA = 1 - \frac{\sum_t (fn_t + fp_t + mm_t)}{\sum_t g_t}. \tag{5.4}$$

Thereby, fn_t, fp_t, mm_t, g_t denote respectively the amount of false negatives, false positives, mismatches and ground truth objects at frame t. A mismatch is counted once if an id switch occurs, in other words a ground truth object's hypothesis is swapped.

5.2.2 Experiments

We compared the varying detection and tracking modules with a full tracking-by-detection approach using the respective text detection. Thereby, text is detected in each frame separately and the subsequently occurring bounding boxes are associated by their overlap. Again we maximized the overall sum of overlaps with help of Munkres' algorithm [58].

Since we exploit multiple threads to detect and track text, the performance of our pipeline relies on the temporal interaction of the two threads. Thus, to establish real world conditions, we limited the frame rate with which the single frames are provided to 30 frames per second (FPS).

Table 5.2 shows the comparison of the methods with respect to varying video resolutions. The metrics represent the entire data set by means of all videos. The methods denoted as native are without any additional modifications beyond respective parametrization. The remaining methods exploit the basic filtering techniques described in Section 5.1.

Due to the overall results, it can be seen that our evaluation dataset is quite challenging. However, the relative low metrics do not contradict a feasible usage of the evaluated methods. Text still can be robustly detected and recognized during its appearance. Especially the misses stem from the often substantial motion blur whilst moving the mobile device towards the energy meter.

The available implementation of the Stroke Width Transform (SWT) as well as the Automatic Number Plate Recognition (ANPR) often failed to group the detected digits together and due to the small overlap, the resulting bounding boxes were hardly matched to ground truth data. Due to these high miss rates, we favored the Extremal Region (ER) classifier TextSpotter 2013 (TS) by Neumann et al. [62] in the following experiments.

Resolution	Method	MOTP	Misses	FP rate	Mismatches	MOTA
	NATIVE TS	0.74	0.27	0.98	0.08	-0.32
	NATIVE SWT	0.57	0.99	0.76	0.00	-0.75
	NATIVE ANPR	0.60	0.84	18.66	0.01	-18.52
768x1366	TS	0.75	0.54	0.13	0.02	0.31
	MSAC&TS	0.72	0.56	0.10	0.02	0.32
	LOCEX&TS	0.75	0.54	0.13	0.02	0.31
	KLT&MSAC&TS	0.70	0.65	0.20	0.00	0.15
	AKAZE&MSAC&TS	0.70	0.57	0.10	0.02	0.32
	NATIVE TS	0.74	0.29	1.52	0.14	-0.95
	NATIVE SWT	0.60	0.99	0.87	0.00	-0.86
	NATIVE ANPR	0.60	0.83	11.32	0.01	-11.16
480x854	TS	0.75	0.57	0.13	0.02	0.28
	MSAC&TS	0.73	0.59	0.10	0.02	0.29
	LOCEX&TS	0.75	0.57	0.13	0.02	0.28
	KLT&MSAC&TS	0.71	0.48	0.31	0.00	0.21
	AKAZE&MSAC&TS	0.70	0.52	0.11	0.02	0.36
	NATIVE TS	0.72	0.34	1.91	0.15	-1.40
	NATIVE SWT	0.60	0.98	0.73	0.00	-0.72
	NATIVE ANPR	0.60	0.82	6.94	0.01	-6.77
320x568	TS	0.74	0.62	0.14	0.02	0.22
	MSAC&TS	0.72	0.64	0.14	0.02	0.20
	LOCEX&TS	0.74	0.62	0.14	0.02	0.22
	KLT&MSAC&TS	0.67	0.53	0.41	0.00	0.06
	AKAZE&MSAC&TS	0.69	0.63	0.16	0.01	0.19
Hybrid	KLT&MSAC&TS	0.62	0.49	0.17	0.00	0.34

Table 5.2: Detection and tracking accuracy evaluated with the CLEAR-MOT evaluation framework [7]. Native full tracking-by-detection methods are compared with a use case tailored filtering enhancement and feature based detection and tracking methods.

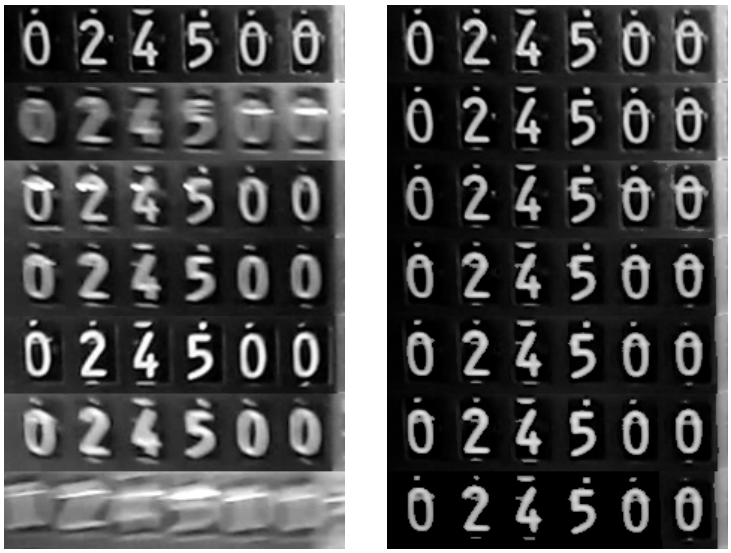
Both evaluated rectification steps led to neglectable improvement regarding the evaluation framework.

As observable especially the full-detection results do not vary significantly comparing the different resolutions. Therefore, we additionally evaluated an optimized hybrid solution, which scales the image down to speed up text detection and omit the expensive Enhanced Correlation Coefficient (ECC) image registration. The resulting pipeline was tested with a resolution of 480×854, respectively 240×427 for text detection. In this manner the overall process is speeded up and it takes less tracking cycles until the text detection module propagates the text position for the first time. As a result, text which only occurs for a short time is less likely missed. Whereas the tracking precision slightly decreased, an increase of accuracy can be observed. In Table 5.2 and hereinafter we refer

to this pipeline configuration as *Hybrid*.

5.3 Reading Accuracy

During this experiment, we evaluated the impact of Multiple Frame Integration (MFI) onto the reading accuracy. Therefore, we passed both to the text recognition, the single, rectified image patches showing the detected and tracked meter reading as well the integrated pendants, which were available by that time. Additionally, we fused the recognition results utilizing a simple histogram voting strategy (see Section 4.1.4). A reasonable integration method should lead to similar recognition results at the beginning of a tracked sequence and improve over time (see Figure 5.2). All detection and tracking approaches described in Section 5.2.2 were considered and used to obtain the text patches.



(a) **(b)**

Figure 5.2: Single extracted frames sampled during a sequence of 62 frames (a) and the integrated pendants in (b). The used integration method first published by Yi et al. [91] increases the contrast and mitigates reflections over time. It is not corrupted by individual frames with considerable motion blur. Nevertheless, it is prone to imperfect image registration.

The meter readings decimal places sometimes show ambiguous values. We omit them by truncating recognized text from right if it exceeds the respective length. The result is considered correct if it is identical to ground truth data.

As described in Section 4.1.4, we filtered frames with a too high *ECC* in a first experiment. Here, we only considered the text patches which passed this test. Further, our optimized *Hybrid* solution omitting the *ECC* refinement step as described in Section 5.2.2 was evaluated.

Resolution	Single frame	Minimum operator	Yi integration	Histogram voting
768x1366	0.45	0.44	0.55	0.63
480x854	0.38	0.50	0.50	0.62
320x568	0.36	0.48	0.43	0.61
Hybrid	0.33	0.29	0.27	0.61

Table 5.3: The relative recognition rates broken down to the image resolution. We passed all validly detected and tracked text patches as well as their integrated pendants to the text recognition module and counted the correct recognitions. Image enhancement methods even decrease the performance using the optimized *Hybrid* pipeline due to jitter and imperfect image registration. This configuration omits the computationally expensive *ECC* registration refinement.

We passed all extracted text patches as well as their integrated pendants to the recognition module and counted the correctly recognized values. We only considered text patches with a valid correspondence to a ground truth object (see Section 5.2.1).

Table 5.3 shows the relative reading accuracies using the different integration methods. In accordance with the experiment in Section 5.2.2, only the faster Kanade-Lucas-Tomasi (KLT) tracking was utilized in combination with the *Hybrid* pipeline. The remaining values combine the results of both tracking methods. As observable, the *image enhancement* approaches lead to a performance boost using the *ECC* refinement step. The reading accuracies even decrease without it. Jitter and motion blur lead to a smeared output and the digits disappear. Figure 5.3 illustrates this effects. However, the simple voting approach outperforms the *image enhancement* methods in all cases. Figure 5.4 summarizes these results.

Further, we evaluated the impact of *MFI* on various lightning conditions. Table 5.4 contains the recognition rates in dependence on the respective evaluation videos. It is noteworthy that especially the frontal light produced by the flash LED in Video 3 mitigates the results most. This illumination creates specular highlights interfering with the contours of the digits. However, even in this case the histogram voting can keep up with the image enhancement methods.

5.4 Algorithm Runtime

Finally, we reviewed the algorithm runtime on a standard Laptop as well as an Android device. More specifically, we used a Lenovo Thinkpad T440s including a Intel® Core™ i5-4200U processor and a Nvidia Shield tablet containing a Quad-Core-CPU with 2,2 GHz.

We compared the runtime of our optimized hybrid solution with the more accurate version using the registration refinement. Table 5.5 contains the respective results. It can bee seen, that the refinement step even with limited iterations is quite computationally expensive. Nevertheless, the performance of the optimized pipeline achieves practical

(a) (b)

Figure 5.3: Single extracted frames sampled during a sequence of 50 frames (a) and the integrated pendants in (b). A simple minimum operator was used to integrate the patches. As can be seen, the letter disappear due to motion blur and imperfect image registration.

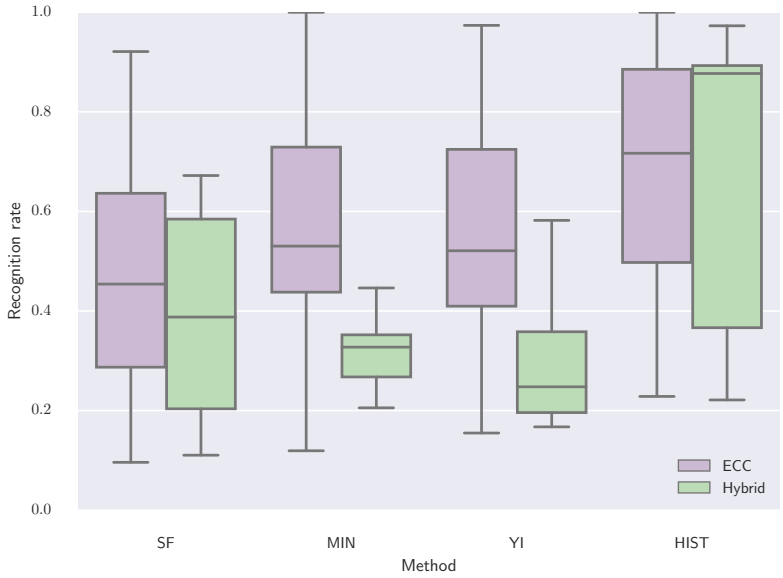


Figure 5.4: The recognition rates using the single extracted frames and the different *MFI* methods. As observable the image enhancement methods (minimum operator and YI integration) highly rely on a good image registration and do not perform very well using the optimized tracking. Nevertheless, the simple histogram voting outperforms the remaining methods in both cases.

results.

Video ID	Single frame	Minimum operator	Yi integration	Histogram voting
1	0.67	0.66	0.65	0.78
2	0.68	0.65	0.70	0.85
3	0.20	0.40	0.30	0.40
4	0.31	0.34	0.40	0.49
5	0.37	0.47	0.58	0.75
6	0.52	0.75	0.67	0.85

Table 5.4: Average recognition rates over all resolutions in combination with ECC registration and in dependence on the different videos. Especially Video 3 showing energy meter illuminated by flash light results in poor recognition rates.

Device	Resolution	Tracking Method	Rectification, Detection	Total (Tracking & MFI)
Laptop	480x854	AKAZE	318.2	144.4
	480x854	KLT	263.1	22.2
	Hybrid	KLT	73.1	5.0
Shield Tablet	480x720	KLT	2788.3	469.2
	Hybrid	KLT	519.2	84.9

Table 5.5: Average time performance measurements in milliseconds.

5.5 Summary

In this chapter, we examined the possibilities of a *MFI* approach on a defined *OCR* use case. We compared the tracking and detection performance of full tracking-by-detection approaches with the different variations of our proposed multi-thread solution and showed that the real-time capabilities are obtained with almost no loss of detection/tracking performance.

Furthermore, we analyzed the impact of several integration methods on the reading accuracy. Especially, frontal illumination due to e.g. a flash light mitigates the recognition accuracy in this use case and can be significantly improved by utilizing simple and fast *MFI* methods. Though, the analyzed *image enhancement* approaches are vulnerable to imperfect image registration. Thus, depending on the processing time of the used *OCR* engine, result fusion methods should be preferred. Finally, the runtime of our pipeline was illustrated. It was shown that real-time capabilities on a mobile device are achieved using an optimized version of our pipeline.

6 Conclusion & Outlook

The aim of this Master's thesis was to enhance the robustness of existing Optical Character Recognition (OCR) solutions running on mobile devices. Different illumination, reflections and specular highlights interfere with text and complicate successful recognition. However, the redundant information covered by a camera stream can be exploited to mitigate these effects. Therefore, the possibilities of Multiple Frame Integration (MFI) on such hardware were evaluated.

To overcome the problem of computationally expensive text detection methods, we outsourced this task to a parallel running thread while efficiently tracking the detection results in the remaining time. Thereby, we exploit multi-core architectures implemented in nowadays mobile devices.

During this thesis several different rectification, detection and tracking approaches and parametrization as part of our pipeline were examined. We compared them to the respective full tracking-by-detection approaches utilizing the CLEAR-MOT evaluation framework [7]. Therefore, a challenging dataset on the defined use case of energy meter readings was created. It contains digits interfered with significant reflections and highlights. We showed that our pipeline is able to keep up with full-detection approaches in terms of precision and accuracy and runs in feasible time for a practical usage on mobile device hardware.

TextSpotter 2013 (TS) by Neumann et al. [60–64] was the single reasonable text detection method in our evaluation. Even though the Stroke Width Transform (SWT) implementation just failed to group the digits together, its runtime does not allow a time efficient application. The Automatic Number Plate Recognition (ANPR) does detect all kinds of structured elements and the resulting high number of false positives was not defensible for our use case.

Since scene text processing is still far from being perfect in nowadays research [29], we propose to integrate use case specific adjustments for practical usage. We showed that our general scene text processing pipeline provides already practicable results with minor adjustments. The basic filtering techniques boosted the overall results tremendously. This

is because *false positives* are worse for the detection and tracking performance in this case than a few more misses. Even replacing the general scene text detection entirely by a more straightforward solution can be a reasonable step.

Furthermore, we evaluated the impact of different camera resolutions on the overall performance. We can conclude that the full resolution of cameras build-in in todays mobile phones and tablets is easily sufficient respectively not necessary to obtain satisfying results.

Our experiments on different *MFI* approaches showed that the image enhancement methods require a qualitative registration of the different image patches showing the same text. However, this kind of registration is likely to be computationally expensive and was not real-time capable on mobile phones using our configuration. More advanced plane tracking solutions may provide this accuracy and speed but mostly are commercial and not freely available.

Additionally, several simple optimization strategies can be exploited to decrease the runtime respectively enhance the user experience. For example one could filter more of the extracted text patches before *MFI* and outsource it to a parallel thread as well. Since reflections do not move much from one frame to another, it could be a good way to avoid expendable text patches and save even more runtime.

Though, if the text recognition is sufficiently fast, a *result fusion* approach can be even more performant. Although, we just examined a simple voting approach, it clearly outperformed the *image enhancement* methods. Thus, it is recommended to prefer this integration approaches. The Anyline energy module already filters recognized numbers outside the configured parameters. Therefore, only fully detected results containing a certain amount of digits were fused. A more sophisticated fusion e.g. considering individual letters and digits can lead to even more promising results.

Summing up, we showed that *MFI* in combination with state-of-the-art text detection and tracking is capable of running in real-time on mobile device hardware. The parallel processing pipeline constitutes a founded basis for practical applications. Additionally, this was verified by significantly improving *OCR* results. Nevertheless, there is still a notable margin for improvement regarding general state-of-the-art scene text processing. Thus, given a use case such as recording energy meter readings, introducing use case tailored adjustments is highly recommended for practical usage. We showed, that even basic ones can significantly improve the performance and lead to satisfying outcomes.

A

List of Acronyms

ADAS	Advanced Driver Assistance Systems
ANPR	Automatic Number Plate Recognition
CCA	Connected Component Analysis
CNN	Convolutional Neural Network
CRF	Conditional Random Fields
DLT	Direct Linear Transformation
DOF	Degrees of Freedom
ECC	Enhanced Correlation Coefficient
EM	Expectation Maximization
ER	Extremal Region
ESM	Efficient Second Order Minimization
FLANN	Fast Library for Approximate Nearest Neighbors
FPS	frames per second
HOG	Histogram of oriented gradients
HT	Hough Transform
IBAN	International Bank Account Number
KLT	Kanade-Lucas-Tomasi
MAD	Mean Absolute Difference
MFI	Multiple Frame Integration
MLESAC	Maximum Likelihood Estimation Sample Consensus
MOTA	Multiple Object Tracking Accuracy
MOTP	Multiple Object Tracking Precision
MSAC	M-Estimator Sample Consensus
MSER	Maximally Stable Extremal Region
NN	Nearest Neighbor

OCR	Optical Character Recognition
RANSAC	Random Sample Consesus
SFT	Stroke Feature Transform
SIFT	Scale-invariant Feature Transform
STC	Scene Text Character
SURF	Speeded Up Robust Features
SVM	Support Vector Machine
SVT	Street View Text
SWT	Stroke Width Transform
TS	TextSpotter 2013

Bibliography

[1] Alcantarilla, P. F., Bartoli, A., and Davison, A. J. (2012). KAZE features. In *European Conference on Computer Vision*. (page 33)

[2] Alcantarilla, P. F., Nuevo, J., and Bartoli, A. (2013). Fast explicit diffusion for accelerated features in nonlinear scale spaces. In *British Machine Vision Conference*. (page 33)

[3] Almazán, J., Gordo, A., Fornés, A., and Valveny, E. (2014). Word spotting and recognition with embedded attributes. *Transactions on Pattern Analysis and Machine Intelligence*, 36(12):2552–2566. (page 21)

[4] Baggio, D. L., Emami, S., Escriva, D. M., Ievgen, K., Mahmood, N., Saragih, J., and Shilkrot, R. (2012). *Mastering OpenCV with Practical Computer Vision Projects*. Packt Publishing, Limited. (page 36)

[5] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). SURF: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417. Springer. (page 25)

[6] Benhimane, S. and Malis, E. (2004). Real-time image-based tracking of planes using efficient second-order minimization. In *International Conference on Intelligent Robots and Systems*, volume 1, pages 943–948. IEEE. (page 25)

[7] Bernardin, K. and Stiefelhagen, R. (2008). Evaluating multiple object tracking performance: the CLEAR MOT metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):1–10. (page 40, 41, 43, 49)

[8] Canny, J. (1986). A computational approach to edge detection. *Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698. (page 13)

[9] Chen, H., Tsai, S. S., Schroth, G., Chen, D. M., Grzeszczuk, R., and Girod, B. (2011). Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In *International Conference on Image Processing*, pages 2609–2612. IEEE. (page 18)

[10] Chen, X. and Yuille, A. L. (2004). Detecting and reading text in natural scenes. In *Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–366. IEEE. (page 13, 21)

[11] Ciresan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *Conference on Computer Vision and Pattern Recognition*, pages 3642–3649. IEEE. (page 5)

[12] Donoser, M. and Bischof, H. (2006). Efficient maximally stable extremal region (MSER) tracking. In *Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 553–560. IEEE. (page 25)

[13] Epshtein, B., Ofek, E., and Wexler, Y. (2010). Detecting text in natural scenes with stroke width transform. In *Conference on Computer Vision and Pattern Recognition*, pages 2963–2970. IEEE. (page 13, 25, 36)

[14] Evangelidis, G. D. and Psarakis, E. Z. (2008). Parametric image alignment using enhanced correlation coefficient maximization. *Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1858–1865. (page 37)

[15] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395. (page 10)

[16] Fragoso, V., Gauglitz, S., Zamora, S., Kleban, J., and Turk, M. (2011). TranslatAR: A mobile augmented reality translator. In *Workshop on Applications of Computer Vision*, pages 497–502. IEEE. (page 2, 3, 25)

[17] Gomez, L. and Karatzas, D. (2014). MSER-based real-time text detection and tracking. In *International Conference on Pattern Recognition*, pages 3110–3115. IEEE. (page 25, 33, 41, 42)

[18] Goto, H. and Tanaka, M. (2009). Text-tracking wearable camera system for the blind. In *International Conference on Document Analysis and Recognition*, pages 141–145. IEEE. (page 26)

[19] Greenhalgh, J. and Mirmehdi, M. (2015). Recognizing text-based traffic signs. *Transactions on Intelligent Transportation Systems*, 16(3):1360–1369. (page 29, 37)

[20] Hartl, A. and Reitmayr, G. (2012). Rectangular target extraction for mobile augmented reality applications. In *International Conference on Pattern Recognition*, pages 81–84. IEEE. (page 9, 25, 36, 38)

[21] Hartley, R. and Zisserman, A. (2005). Multiple view geometry in computer vision. *Robotica*, 23(2):271–271. (page 6, 7, 8, 35)

[22] Horn, B. K. P. (1986). *Robot Vision*. MIT Press. (page 14)

[23] Huang, W., Lin, Z., Yang, J., and Wang, J. (2013). Text localization in natural images using stroke feature transform and text covariance descriptors. In *International Conference on Computer Vision*, pages 1241–1248. IEEE. (page 15)

[24] Jaderberg, M., Simonyan, K., Vedaldi, A., and Zisserman, A. (2014a). Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*. (page 21)

[25] Jaderberg, M., Simonyan, K., Vedaldi, A., and Zisserman, A. (2016). Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, 116(1):1–20. (page 21)

[26] Jaderberg, M., Vedaldi, A., and Zisserman, A. (2014b). Deep features for text spotting. In *European Conference on Computer Vision*, pages 512–528. Springer. (page 13)

[27] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323. (page 18)

[28] Jung, C. R. and Schramm, R. (2004). Rectangle detection based on a windowed hough transform. In *Symposium on Computer Graphics and Image Processing*, pages 113–120. IEEE. (page 9)

[29] Karatzas, D., Gomez-Bigorda, L., Nicolaou, A., Ghosh, S., Bagdanov, A., Iwamura, M., Matas, J., Neumann, L., Chandrasekhar, V. R., Lu, S., et al. (2015). ICDAR 2015 competition on robust reading. In *International Conference on Document Analysis and Recognition*, pages 1156–1160. IEEE. (page 1, 16, 41, 42, 49)

[30] Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., i Bigorda, L. G., Mestre, S. R., Mas, J., Mota, D. F., Almazan, J. A., and de las Heras, L. P. (2013). ICDAR 2013 robust reading competition. In *International Conference on Document Analysis and Recognition*, pages 1484–1493. IEEE. (page 41, 42)

[31] Koo, H. I. and Kim, D. H. (2013). Scene text detection via connected component clustering and nontext filtering. *Transactions on Image Processing*, 22(6):2296–2305. (page 18)

[32] Kumar, D., Prasad, M., and Ramakrishnan, A. (2012). MAPS: Midline analysis and propagation of segmentation. In *Indian Conference on Computer Vision, Graphics and Image Processing, Proceedings*, page 15. ACM. (page 21)

[33] Kumar, D., Prasad, M. A., and Ramakrishnan, A. (2013). NESP: Nonlinear enhancement and selection of plane for optimal segmentation and recognition of scene word images. In *IS&T/SPIE Electronic Imaging*, pages 865806–865806. International Society for Optics and Photonics. (page 21)

[34] Lagunovsky, D. and Ablameyko, S. (1999). Straight-line-based primitive extraction in grey-scale object recognition. *Pattern Recognition Letters*, 20(10):1005–1014. (page 8)

[35] Lefler, M., Hel-Or, H., and Hel-Or, Y. (2013). Metric plane rectification using symmetric vanishing points. In *International Conference on Image Processing*, pages 300–304. IEEE. (page 8)

[36] Leutenegger, S., Chli, M., and Siegwart, R. Y. (2011). BRISK: Binary robust invariant scalable keypoints. In *International Conference on Computer Vision*, pages 2548–2555. IEEE. (page 25)

[37] Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *The Quarterly of Applied Mathematics*, (2):164–168. (page 11)

[38] Li, H. and Doermann, D. (1998). Automatic text tracking in digital videos. In *Second Workshop on Multimedia Signal Processing*, pages 21–26. IEEE. (page 24)

[39] Li, H. and Doermann, D. (1999). Text enhancement in digital video using multiple frame integration. In *ACM International Conference on Multimedia*, pages 19–22. ACM. (page 24, 29)

[40] Li, H., Doermann, D., and Kia, O. (2000). Automatic text detection and tracking in digital video. *Transactions on Image Processing*, 9(1):147–156. (page 24)

[41] Liebowitz, D., Criminisi, A., and Zisserman, A. (1999). Creating architectural models from images. In *Computer Graphics Forum*, volume 18, pages 39–50. Wiley Online Library. (page 7, 9, 36)

[42] Lienhart, R. and Effelsberg, W. (2000). Automatic text segmentation and text recognition for video indexing. *Multimedia Systems*, 8(1):69–81. (page 27)

[43] Lienhart, R. W. and Stuber, F. (1996). Automatic text recognition in digital videos. In *Electronic Imaging: Science & Technology*, pages 180–188. International Society for Optics and Photonics. (page 24, 28)

[44] Lin, C., Huertas, A., and Nevatia, R. (1994). Detection of buildings using perceptual grouping and shadows. In *Computer Society Conference on Computer Vision and Pattern Recognition*, pages 62–69. IEEE. (page 8)

[45] Liu, X. and Wang, W. (2012). Robustly extracting captions in videos based on stroke-like edges and spatio-temporal analysis. *Transactions on Multimedia*, 14(2):482–489. (page 29)

[46] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110. (page 24, 25)

[47] Lucas, B. D., Kanade, T., et al. (1981). An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, volume 81, pages 674–679. (page 33)

[48] Manmatha, R., Han, C., and Riseman, E. M. (1996). Word spotting: A new approach to indexing handwriting. In *Computer Society Conference on Computer Vision and Pattern Recognition*, pages 631–637. IEEE. (page 21)

[49] Matas, J., Chum, O., Urban, M., and Pajdla, T. (2004). Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767. (page 16, 17)

[50] Merino, C. and Mirmehdi, M. (2007). A framework towards realtime detection and tracking of text. In *International Workshop on Camera-based Document Analysis and Recognition*, pages 10–17. (page 26)

[51] Merino-Gracia, C., Lenc, K., and Mirmehdi, M. (2011). A head-mounted device for recognizing text in natural scenes. In *International Workshop on Camera-Based Document Analysis and Recognition*, pages 29–41. Springer. (page 26)

[52] Mi, C., Xu, Y., Lu, H., and Xue, X. (2005). A novel video text extraction approach based on multiple frames. In *International Conference on Information Communications & Signal Processing*, pages 678–682. IEEE. (page 27, 29)

[53] Miao, L. and Peng, S. (2006). Perspective rectification of document images based on morphology. In *International Conference on Computational Intelligence and Security*, volume 2, pages 1805–1808. IEEE. (page 10)

[54] Minetto, R., Thome, N., Cord, M., Leite, N. J., and Stolfi, J. (2011). Snoopertrack: Text detection and tracking for outdoor videos. In *International Conference on Image Processing*, pages 505–508. IEEE. (page 26)

[55] Mishra, A., Alahari, K., and Jawahar, C. (2012). Top-down and bottom-up cues for scene text recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 2687–2694. IEEE. (page 21)

[56] Mita, T. and Hori, O. (2001). Improvement of video text recognition by character selection. In *International Conference on Document Analysis and Recognition, Proceedings Conference on*, pages 1089–1093. IEEE. (page 28)

[57] Muja, M. and Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. *International Conference on Computer Vision Theory and Applications*, 2(331-340):2. (page 25, 33)

[58] Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38. (page 42)

[59] Na, Y. and Wen, D. (2010). An effective video text tracking algorithm based on sift feature and geometric constraint. In *Pacific-Rim Conference on Multimedia*, pages 392–403. Springer. (page 24)

[60] Neumann, L. and Matas, J. (2010). A method for text localization and recognition in real-world images. In *Asian Conference on Computer Vision*, pages 770–783. Springer. (page 18, 49)

[61] Neumann, L. and Matas, J. (2011). Text localization in real-world images using efficiently pruned exhaustive search. In *International Conference on Document Analysis and Recognition*, pages 687–691. IEEE. (page 18, 20)

[62] Neumann, L. and Matas, J. (2012). Real-time scene text localization and recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 3538–3545. IEEE. (page 36, 42)

[63] Neumann, L. and Matas, J. (2013). On combining multiple segmentations in scene text recognition. In *International Conference on Document Analysis and Recognition*, pages 523–527. IEEE. (page)

[64] Neumann, L. and Matas, J. (2015). Real-time lexicon-free scene text localization and recognition. *Transactions on Pattern Analysis and Machine Intelligence*. (page 18, 49)

[65] Nieto, M. and Salgado, L. (2010). Real-time robust estimation of vanishing points through nonlinear optimization. In *SPIE Photonics Europe*, pages 772402–772402. International Society for Optics and Photonics. (page 10, 36, 38)

[66] Otsu, N. (1975). A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27. (page 29, 37)

[67] Petter, M., Fragoso, V., Turk, M., and Baur, C. (2011). Automatic text detection for mobile augmented reality translation. In *International Conference on Computer Vision Workshops*, pages 48–55. IEEE. (page 3)

[68] Phan, T. Q., Shivakumara, P., Lu, T., and Tan, C. L. (2013). Recognition of video text through temporal integration. In *International Conference on Document Analysis and Recognition*, pages 589–593. IEEE. (page 25)

[69] Pilu, M. (2001). Extraction of illusory linear clues in perspectively skewed documents. In *Computer Society Conference on Computer Vision and Pattern Recognition*, page 363. (page 10)

[70] Pratt, W. K. (2007). *Digital Image Processing: PIKS Scientific Inside*. John Wiley and Sons, Inc., New Jersey, USA. (page 19)

[71] Rong, X., Yi, C., Yang, X., and Tian, Y. (2014). Scene text recognition in multiple frames based on text tracking. In *International Conference on Multimedia and Expo*, pages 1–6. IEEE. (page 24, 27, 28)

[72] Saidane, Z. and Garcia, C. (2007). Robust binarization for video text recognition. In *International Conference on Document Analysis and Recognition*, volume 2, pages 874–879. IEEE. (page 21)

[73] Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336. (page 20)

[74] Shi, C., Wang, C., Xiao, B., Zhang, Y., and Gao, S. (2013a). Scene text detection using graph model built upon maximally stable extremal regions. *Pattern Recognition Letters*, 34(2):107–116. (page 18)

[75] Shi, C., Wang, C., Xiao, B., Zhang, Y., Gao, S., and Zhang, Z. (2013b). Scene text recognition using part-based tree-structured character detection. In *Conference on Computer Vision and Pattern Recognition*, pages 2961–2968. IEEE. (page 21)

[76] Shi, J. and Tomasi, C. (1994). Good features to track. In *Computer Society Conference on Computer Vision and Pattern Recognition*, pages 593–600. IEEE. (page 33)

[77] Tanaka, M. and Goto, H. (2008). Text-tracking wearable camera system for visually-impaired people. In *International Conference on Pattern Recognition*, pages 1–4. IEEE. (page 26)

[78] Tao, W.-b., Tian, J.-w., and Liu, J. (2002). A new approach to extract rectangular building from aerial urban images. In *International Conference on Signal Processing*, volume 1, pages 143–146. IEEE. (page 8)

[79] Tomasi, C. and Kanade, T. (1991). *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ. Pittsburgh. (page 33)

[80] Torr, P. H. and Zisserman, A. (2000). MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156. (page 10, 12)

[81] Tuytelaars, T. and Mikolajczyk, K. (2008). Local invariant feature detectors: a survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280. (page 24)

[82] Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., and Schmalstieg, D. (2010). Real-time detection and tracking for augmented reality on mobile phones. *Transactions on Visualization and Computer Graphics*, 16(3):355–368. (page 25)

[83] Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *International Conference on Machine Learning, Proceedings*, pages 1058–1066. (page 5)

[84] Wang, K. and Belongie, S. (2010). *Word spotting in the wild*. Springer. (page 5, 6, 21)

[85] Wang, N., Shi, J., Yeung, D.-Y., and Jia, J. (2015). Understanding and diagnosing visual tracking systems. In *International Conference on Computer Vision*, pages 3101–3109. IEEE. (page 24)

[86] Wang, T., Wu, D. J., Coates, A., and Ng, A. Y. (2012). End-to-end text recognition with convolutional neural networks. In *International Conference on Pattern Recognition*, pages 3304–3308. IEEE. (page 13)

[87] Weinman, J. J., Butler, Z., Knoll, D., and Feild, J. (2014). Toward integrated scene text reading. *Transactions on Pattern Analysis and Machine Intelligence*, 36(2):375–387. (page 21)

[88] Wolf, C., Jolion, J.-M., and Chassaing, F. (2002). Text localization, enhancement and binarization in multimedia documents. In *International Conference on Pattern Recognition*, volume 2, pages 1037–1040. IEEE. (page 27, 29)

[89] Yao, C., Bai, X., and Liu, W. (2014). A unified framework for multioriented text detection and recognition. *Transactions on Image Processing*, 23(11):4737–4749. (page 15)

[90] Yao, C., Bai, X., Liu, W., Ma, Y., and Tu, Z. (2012). Detecting texts of arbitrary orientations in natural images. In *Conference on Computer Vision and Pattern Recognition*, pages 1083–1090. IEEE. (page 15)

[91] Yi, J., Peng, Y., and Xiao, J. (2009). Using multiple frame integration for the text recognition of video. In *International Conference on Document Analysis and Recognition*, pages 71–75. IEEE. (page 29, 37, 44)

[92] Yin, X.-C., Hao, H.-W., Sun, J., and Naoi, S. (2011). Robust vanishing point detection for mobilecam-based documents. In *International Conference on Document Analysis and Recognition*, pages 136–140. IEEE. (page 10)

[93] Yin, X.-C., Yin, X., Huang, K., and Hao, H.-W. (2014). Robust text detection in natural scene images. *Transactions on Pattern Analysis and Machine Intelligence*, 36(5):970–983. (page 18)

[94] Yusufu, T., Wang, Y., and Fang, X. (2013). A video text detection and tracking system. In *International Symposium on Multimedia*, pages 522–529. IEEE. (page 25, 29)

[95] Zhen, W. and Zhiqiang, W. (2010). An efficient video text recognition system. In *International Conference on Intelligent Human-Machine Systems and Cybernetics*, volume 1, pages 174–177. IEEE. (page 29)

[96] Zhou, J., Xu, L., Xiao, B., Dai, R., et al. (2007). A robust system for text extraction in video. In *International Conference on Machine Vision*, pages 119–124. IEEE. (page 29)

[97] Zhu, Y., Carragher, B., Mouche, F., and Potter, C. S. (2003). Automatic particle detection through efficient hough transforms. *Transactions on Medical Imaging*, 22(9):1053–1062. (page 8)