USC-SIPI REPORT #425

Advanced Techniques for Human Action Classification and Text Localization

by

Harshad Kadu

December 2015

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Viterbi School of Engineering
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Suite 400
Los Angeles, CA 90089-2564 U.S.A.

ADVANCED TECHNIQUES FOR HUMAN ACTION CLASSIFICATION AND TEXT LOCALIZATION

by

Harshad Kadu

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

December 2015

Table of Contents

List of Tables

List of Figures

Abstract

This thesis contains two main research topics: 1) automatic human action classification with mocap data and 2) text localization in natural scene images. The common theme of these two topics is the application of pattern recognition techniques to multimedia information processing as detailed below.

Automatic classification of human action in motion capture (mocap) data has many commercial, biomechanical and medical applications and is the principal focus of the thesis. First, we propose a multi-resolution string representation scheme based on the tree-structured vector quantization (TSVQ) to transform the time-series of human poses into codeword sequences. Then, we take the temporal variations of human poses into account via codeword sequence matching. Furthermore, we develop a family of pose-histogram-based classifiers to examine the spatial distribution of human poses. We analyze the performance of the temporal and spatial classifiers separately. To achieve a higher classification rate, we merge their decisions and soft scores using novel fusion methods. The proposed fusion solutions are tested on a wide variety of sequences from the CMU mocap database using 5-fold cross validation, and a correct classification rate of 99.6% is achieved.

Searching for text regions in natural images is a challenging task for many computer vision applications. In the second part of our research, we propose a novel text localization scheme based on multi-stage incremental region classification technique. The stable extremal region detector investigates peculiar characteristics of text to discover regions with possible textual content across different channels. The coarse decision stump classifiers designed on geometric features and context-based text grouping stages efficiently remove false positives and outline the regions of interest. An ensemble of trained decision tree classifiers categorizes the remaining regions into text or non-text using the gradient profile features. Finally, the ROIs from different views and channels are fused together to procure a consolidated list of text regions. As per the experimental results, our suggested technique is among the top algorithms reported in the literature.

Chapter 1

Introduction

This thesis contains two main research topics: 1) automatic human action classification with mocap data and 2) text localization in natural scene images. The common theme of these two topics is the application of pattern recognition techniques to multimedia information processing. Research significance, related previous work, contributions and content organization of these two topics are presented below.

1.1 Significance of the Research

1.1.1 Human Action Classification

The increasing demand for rendering smooth and plausible 3D motion is fueling the development of motion capture (mocap) systems. This new format of high quality 3D motion data has paved its way into animation movies, high-end computer games [CCY11], biomechanics, robotics, gait analysis and rehabilitation [SSL02] and machine translation of sign languages [LKL$^+$04, YS04]. The diverse applications of mocap data and the rapid development of mocap systems have resulted in a large corpus of motion capture data in recent years. However, the large amount of raw data files makes it difficult to organize. It is desirable that both file names and the associated metadata should be able to provide a high level description of the contents of mocap sequences. Since manual annotation of mocap sequences is labor intensive, it is essential to develop an automated technique that can segment mocap data into homogeneous intervals, classify each interval into a basic action type, and index it for future retrieval. With such annotated mocap databases in place, a proper motion editing and authoring tool can be used to synthesize realistic motion sequences.

To develop efficient indexing and retrieval techniques for mocap databases, the first step is to classify the data into subsets according to their similarities, which is known as the action classification problem. As far as the human full-body motion is concerned, the mocap system records the actions of human actors by placing several markers on their body. These markers and a grid of infrared cameras help determine the full-body movement of those actors in terms of joint orientations. This joint orientation information constitutes the mocap data used in our research. Human mocap data are essentially time series of human body poses (Fig. 3.1). Different human motions may be of different time duration, tempo and style. Motions from same action type, *e.g.* walking, can vary from person to person. Moreover, visually similar motions may start or end with different joint orientations or even with different human body postures. A robust classification algorithm has to compensate for these irregularities in the data.

1.1.2 Text Localization

In recent years, the task of locating text in natural scene images is attracting attention of researchers and industrial community alike, due to its inherent complexity and vast range of applications. Text region detection is a precursor to many computer vision tasks such as, optical character recognition, robotic navigation, compound video compression, scene understanding, text-based image indexing, search and retrieval etc. The aim of our text localization research is to detect text regions in the image and draw bounding boxes around each and every word. Contrary to the scanned documents, text in natural images have different sizes, fonts, orientations, illumination and colors. The cluttered background also poses a serious threat to the localization accuracy. This diversity and irregularity makes text localization in these circumstances quite difficult. The fact that, no algorithm has yet been able to achieve a considerable accuracy on the benchmark datasets, further illustrates the challenging nature of the problem.

1.2 Review of Related Previous Work

1.2.1 Human Action Classification

Different methods have been proposed to synthesize realistic human motions by reusing existing mocap data via data-driven motion editing and authoring, *e.g.*, Kovar and Gleicher [KG04], Safonova *et al.* [SHP04], Ren *et al.* [RSH$^+$05], Lee *et al.* [LCR$^+$02], Zordan *et al.* [ZMCF05], and Pullen and Bregler [PB02]. In order to reuse existing mocap data, efficient searching, indexing and browsing techniques are required. So the first step is to group motion clips into subsets based on their similarities, which is the focus of our current research.

Identifying similar action types in a mocap database has been considered a challenging problem. Most of the previous work on motion comparison adopted features that were close to the raw data. For example, Liu and Popović [LP02] included the principal component analysis (PCA) reduced raw data, 3D point clouds, joint angles, 3D positions, etc. Forbes and Fiume [FF05] used the weighted PCA for motion comparison. The performance was further enhanced by integrating the weighted PCA with other features such as characteristic points, seed points, etc. Kovar and Gleicher [KG04] proposed a numerical similarity technique to find similar motion clips from a large data set. Motion similarity is a semantic concept. The raw numerical data may differ a lot even for two visually similar motions. Hence these types of features do not work well in practice.

Liu *et al.* [LZWP03] used automatically extracted key frames and a hierarchical tree of clusters of motions to search similar actions in the database. Sakamoto *et al.* [SKK04] proposed a motion map method that trains the self-organizing-map (SOM) with motion data and indexes clips by SOM nodes. The motion map transforms the N-dimensional data to a 2D map and computes similarity using this map. However, this scheme lacks higher level data indexing and demands exact node-to-node matching in search.

Human motion is typically a long time series with no overt segmentation information so the dynamic time warping (DTW) based motion comparison techniques (*e.g.*, Chiu *et al.* [CCW$^+$04], Kovar and Gleicher [KG04], Wu *et al.* [WCYL03]) suffer from the complexity of finding the corresponding start/end points of motion data series for motion similarity comparison. Hsu *et al.* [HPP05] used an iterative motion warping (IMW) technique, which is an improvement over the traditional DTW technique, to find the correspondence by minimizing an objective function with dynamic programming. Generally speaking, these features work well only for certain applications.

Several algorithms were developed to identify locally similar segments in the motion dataset, for example, Arikan and Forsythe [AF02], Kim *et al.* [KPS03], Kovar and Gleicher [KG03], Lee *et al.* [LCR$^+$02], and Wang and Bodenheimer [WB03]. Arikan *et al.* [AFB03] proposed a semi-automatic annotation technique using SVM classifiers. Cardle *et al.* [CVB$^+$03] used the GEMINI framework to search mocap data sets. Yang and Shahabi [YS04] proposed a method called EROS to calculate similarity using PCA and eigenvalues. Li *et al.* [LZP07] used the kWAS method to measure similarity of motion clips via singular value decomposition.

Relational features [MRC05a,MRC05b,MR06,RKM08], built upon the Boolean relations between various body parts, are more suitable for generic applications. Common relational features include: the position of joints with respect to a certain plane, the angle between two limbs, the angular velocity of joint angles, the position of joints with respect to other joints, the direction of motion, touch, etc. Such features are more robust to pose distortion and other small aberrations in action sequences. Although relational features provide a good alternative, automatic selection of a good set of relational features for a wide range of motion types is not straightforward. This imposes a limitation on the utility of these features.

Wu *et al.* [WXWL09] presented a cluster-based scheme for mocap data indexing and retrieval, where temporal variations were accounted for while spatial variations were not captured effectively. Later, Wu *et al.* [WWX09] incorporated the spatial complexity of mocap data using hierarchical clustering. But this scheme cannot effectively differentiate between perceptually similar motions, such as *running* and *marching*. The different variations of the same action type (*e.g.* cartwheel) were also not clustered correctly. The SOM representation used in their scheme transforms the 62 dimensional mocap data into a crude 2D representation, which suffers from self occlusions. This significantly hinders its capability to categorize complex motions.

The main contribution of our work is automatic classification of perceptually similar mocap sequences based on the "multi-resolution string representation" that incorporates both spatial and temporal information effectively. It will be shown later in the experimental section that the proposed algorithm offers a correct classification rate of 99.6% for a database consisting of 30 different action types. To the best our knowledge, no other algorithm with better performance on such a diverse dataset has been reported in the literature.

Although our solution is promising, it has some limitations. For example, it might not work well on a clip containing actions from different categories, *e.g.* a clip with walking motion followed by running. However, with mocap data segmentation as the pre-processing step, this problem can be mitigated. The proposed solution was not thoroughly tested on categories with minor variations, such as, *get up from the floor without the hands touching the floor* and *get up from the floor by pushing hands against the floor*. As these categories are practically the same, we combined them into a single category in the experiment.

Researchers from the computer vision community have proposed a variety of methods for action recognition in unconstrained video databases such as UCF11 and UCF50. These methods used template matching, finite state models, bag of features, face detection, context, speech and text information etc. for action recognition [SC12, KS13]. But here, we work directly on the human skeleton posture information, which is available with the mocap data.

1.2.2 Text Localization

The techniques for text localization in scanned document can be broadly classified into three main categories: top-down, bottom-up and hybrid approaches [ODP99]. The top-down approach [LCC00], [KNSV93], [IKA87], [CYWM03] starts by detecting the page schematic and then successively splits up the cells to reach lower levels. For instance, Khedekar [KGS03] proposed a projection profile based algorithm using this top-down approach to separate images and text in the Devnagri documents. However, due to the dependence on page layout information, these methods are suitable only for the scanned document images. The bottom-up approach [Bix98], [Mak83] on the other hand, starts merging together the components at the lowest level and forms larger structures as it goes up. Though these methods are flexible, they suffer from accumulation errors [KGS03]. The hybrid approach based methods [JZ95], [CLKH96] and [JB92] try to allow some leeway in the above two approaches. As opposed to scanned document, the natural scene images do not have a fixed layout. Text can appear anywhere in the image and the orientation, font-size or color may vary from one region to the other. For these reasons, the techniques for scanned documents cannot be easily extended to natural images.

One group of text localization schemes for natural scene images use sliding window technique to search for relevant texture patterns. The sliding window methods [CY04], [LW02], [GEF04], [LDK00], [YHGZ05] employ local features within the windows of varying sizes and resolution, to classify windows as text or non-text. Apart from limiting text search to specific window sizes or orientations, these methods have high computational complexity due to different possible choices of image scales, window sizes and aspect ratios. Following this school of thought, Chen [CY04] devised a procedure of detecting regions of text in the city scenes. The approach is similar to the face detection framework proposed by Viola-Jones [VJ01a], [VJS05]. Chen used weak classifiers to build the strong text/non-text classifiers using AdaBoost machine learning algorithm [FS96]. An adaptive binarization and extension algorithm was then applied to the selected regions in order to mark out the text. This paper used the block features similar to the ones described in [VJ01b] and was designed specifically for reading the text on the signboards, billboards etc.

The other group of techniques detect regions of interest by merging similar pixels into connected components [NM13], [NM12], [LDL05], [JKJ04], [JY98], [Kim96], [YI06]. These methods are not constrained by orientation of text, font size or resolution. They are computationally tractable and the text segmentation is precise to the granularity of individual letters. This aids in subsequent character recognition using OCR techniques [BMP01], [GRL$^+$98], [RWH96]. Major disadvantages include false alarms, threshold selection, spurious connections, sensitivity to noise and background clutter. One such method is the stroke width transform (SWT) [EOW10], [HLYW13] that leverages the fact that letters in images have constant stroke width. The gradients on either side of the stroke are anti-parallel. Pairing those anti-parallel gradients help in finding the regions of interest. Neumann and Matas [NM12] proposed maximally stable extremal regions (MSER) based technique for detecting text regions. It relies on the fact that pixels within the text have similar intensities. Our proposed framework builds on the stable extremal region detector, but diverges significantly thereafter.

1.3 Contributions of the Research

1.3.1 Human Action Classification

In our tree-structured vector quantization (TSVQ) based scheme, the dynamics of human motion is represented by a sequence of multi-resolution codewords. Apparently, two motions are similar if their corresponding codeword sequences are similar. To leverage this similarity, we analyze the distribution of these codewords in the temporal and spatial domains separately. Then, to overcome the individual limitations of these two approaches, we fuse their outcomes and soft scores to make final decision. We examine the proposed solutions on a wide range of motions from the CMU mocap dataset [CMU07]. These motions are recorded in a controlled environment with only one performer per clip. Furthermore, the clips are assumed to be homogeneous, wherein every motion clip belongs to only one motion category.

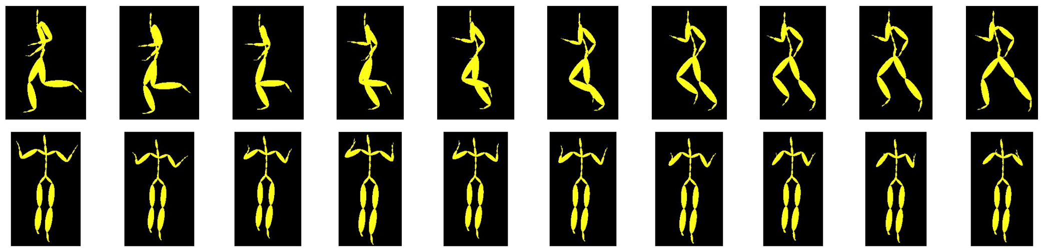


Figure 1.1: Illustration of ten consecutive poses of the running motion (top) and the teapot nursery rhyme rendition (bottom)

It is worthwhile to mention that a mocap data classification technique using hierarchical codewords was described in [KKK11], which is a 2-page summary of some intermediate results of our research. The algorithmic description was brief and experimental results were preliminary. Compared to [KKK11], the current work has a more complete treatment on this topic. Specifically, it has the following major contributions:

- Development of an advanced temporal domain approach to mocap data classification using codeword sequence matching;
- Development of a novel spatial domain approach for human mocap data classification with pose-histogram-based SVM classifiers;
- Performance evaluation on a larger dataset containing 278 human motion clips ($\sim$ 0.5 million frames) spanning 30 action categories;
- Achieving a correct classification rate of 99.6% for a data set comprising of simple, complex as well as perceptually similar action types.

The spatial domain approach takes the human skeleton structure into account so that it is restricted to human motions only. In contrast, the temporal domain approach does not assume any inherent structure in the data so it is more generic in this sense.

Moreover, other formats of marker-based mocap data can be handled with the proposed algorithms if they can provide the 3D pose information. We will describe these new approaches, their implementations, fusion and experimental results in greater detail to offer readers a complete picture of our solution and its superior performance.

1.3.2 Text Localization

Text regions have some peculiar properties which distinguish them from the non-text regions. For example, text has sharp edges, interior pixels have consistent color, related component have distinct spatial alignment etc. Based on these observations, we propose a multi-stage region classification model, that systematically detects individual words in the images. This multi-stage design for text localization enhances efficiency and reduces complexity. After detecting probable candidates in the first stage, the task of eliminating false alarms is distributed across subsequent stages of the framework. Each stage targets specific properties of text to come up with its own set of discriminative features. The commendable performance justifies the effectiveness of this incremental approach.

The stable extremal region (SER) detector [MCUP04], [CTS$^+$11] explores intra-component color similarity, to discover potential text candidates. But these detected regions may contain several false positives. Geometric filtering phase removes most of these false alarms. Later on, the context-based text grouping stage combines the remaining components into words, by exploiting the inter-component consistency and spatial alignment information. The text-like false positives are gradually phased out of the pipeline. An ensemble of decision tree classifiers, then decides the fate of these newly formed word candidates. These weak decision tree classifiers are built on gradient profile features and combined into an ensemble using the Adaboost framework. The whole process is repeated for different channels of the image and the detected words are aggregated. Experimental results on the ICDAR dataset [LPS$^+$03], [Luc05] show that our performance score is comparable to the state-of-the-art algorithms.

The major contributions of our research are listed below:

- Designed the stripe-scan features for removing text-like false positives
- Proposed the scan-line stroke width transform technique for efficient computation of stroke width values on binary components
- Developed a robust text grouping mechanism which is immune to missing components
- Built the novel gradient profile features that capture the peculiarities of text regions
- Demonstrated the effectiveness of the approach by achieving excellent results on publicly available datasets

1.4 Organization of the Dissertation

Rest of the thesis is organized as follows. Chapter 2 reviews some prominent tools in these fields. The advanced techniques for automatic human action classification are presented in Chapter 3. Chapter 4 entails detailed discussion on the proposed text localization framework. Finally, conclusions and future work items are listed in Chapter 5.

Chapter 2

Background Review

This chapter reviews some well-known tools used in our human action classification and text localization research. It also presents some relevant techniques that are widely used by other researchers in these fields. The chapter serves as a vignette for the methods briefly mentioned in the main chapters.

2.1 Human Action Classification With Mocap Data

The increasing demand for rendering smooth and plausible 3D motion is fueling the development of motion capture (mocap) systems. This new format of high quality 3D motion data has paved its way into animation movies, high-end computer games [CCY11], biomechanics, robotics, gait analysis and rehabilitation [SSL02] and machine translation of sign languages [LKL$^+$04, YS04]. The diverse applications of mocap data and the rapid development of mocap systems have resulted in a large corpus of motion capture data in recent years. It is essential to develop an automated technique that can segment mocap data into homogeneous intervals, classify each interval into a basic action type, and index it for future retrieval. With such annotated mocap databases in place, a proper motion editing and authoring tool can be used to synthesize realistic motion sequences.

To develop efficient indexing and retrieval techniques for mocap databases, the first step is to classify the data into subsets according to their similarities, which is known as the action classification problem. Human mocap data are essentially time series of human body poses. Different human motions may be of different time duration, tempo and style. Motions from same action type, *e.g.* walking, can vary from person to person. Moreover, visually similar motions may start or end with different joint orientations or even with different human body postures. A robust classification algorithm has to compensate for these irregularities in the data.

In our tree-structured vector quantization (Sec. 2.1.1) based classification scheme, the dynamics of human motion is represented by a sequence of multi-resolution codewords. Apparently, two motions are similar if their corresponding codeword sequences are similar. To leverage this similarity, we analyze the distribution of these codewords in the temporal and spatial domains separately. Suffix array technique (Sec. 2.1.2) serves the purpose by finding similar subsequences within longer codeword sequences. These similarities can later be utilized for action classification. A short summary of these methods is provided in the following sections.

2.1.1 Tree Structured Vector Quantization (TSVQ)

The human action classification algorithm categorizes actions in the mocap sequences. A frame in the mocap sequence is inherently a high dimensional vector of 3D rotation angles. For compact representation of such multi-dimensional vectors, they can be clustered together and represented by cluster centroids. Tree Structured Vector Quantization provides an elegant and fast way of clustering data while maintaining the codeword hierarchy.

TSVQ performs clustering of the data vectors by repeated application of the Generalized Lloyd Algorithm (GLA). Initially all the vectors are considered to be a part of a single cluster. Iteratively, each cluster is split into two using LBG splitting and fine tuned with the repeated application of the GLA to the perturbed centroids. After convergence, the parent codeword spawns two child codewords in the tree structure. Finally, a balanced binary tree-like structure is obtained.

For example, consider the 2D vectors represented by crosses in Fig. 2.1. Initially all the vectors are considered to be the part of a single cluster. The cluster centroid is represented by the level-0 node. The centroid is split into two points. The new child centroids thus obtained are fine tuned with repeated application of the GLA until convergence, to get two child codewords. Fig. 2.1 represents those child codewords with the two nodes at level-1 in the tree structure. Two green lines from the node at level-0 to nodes at level-1 emphasize the parent-child hierarchy. The points in the parent cluster are split among the two child centroids based on their distances from those new centroids. The split of the data points is represented by the green line. Similarly, the yellow, red and blue lines partition the points into smaller clusters. The stable centroids are depicted by nodes in the adjoining tree structure.

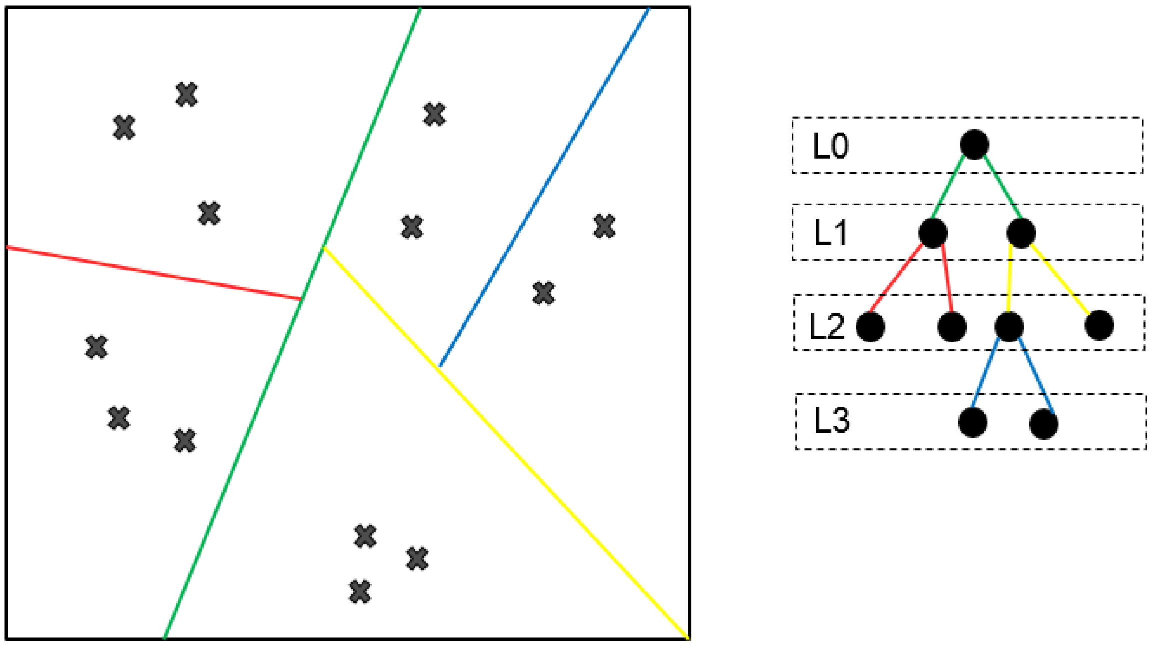


Figure 2.1: TSVQ data splitting and the corresponding codeword tree

The codewords or nodes at a certain level constitute the codebook at that specific level. For example, the four nodes at level-2 in the tree structure of Fig. 2.1 form level-2 codebook. To encode the data vectors, a level-n codebook is chosen from the tree structure and the vector is mapped to the nearest node or codeword.

2.1.2 Suffix Array Technique

In the human action classification research, the action sequences are represented using multi-resolution motion strings. Two actions are similar if their corresponding motion strings are similar. The problem of finding similarity between two motion strings can be reformulated to a string matching problem. The latter can be efficiently solved by using the suffix array technique [AKO04, KS03, KLA+01]. In this technique, all suffixes of a string, S, are sorted in an ascending order. The sorted suffixes for a sample motion string, $S = [25, 18, 87, 160, 98, 25, 18]$, are shown in the last column of Table 2.1. Note that the end of the string is denoted by a unique symbol '\$'. The '\$' symbol is assigned to have the highest value among all codeword indices in the motion string. The suffix notation S_q represents the sub-sequence of S starting from $S[q]$ till the end of the string, followed by \$. Only the starting point *i.e.* index 'q' is enough to represent this suffix S_q. The array of indices 'q' defines the "suffix array", denoted by *suftab* in Table 2.1. We have $suftab[\tilde{i}] = q$ for suffix S_q that is at the $\tilde{i}^{th}$ position in the sorted list starting with $\tilde{i} = 0$. To give an example, S_4 represents the sub-sequence starting from $S[4] = 98$, which is $[98, 25, 18, \$]$. The suffix S_4 is at the fifth location ($\tilde{i} = 5$) in the sorted list. Therefore, $suftab[5] = 4$ in Table 2.1.

$\tilde{i}$	$suftab[\tilde{i}]$	$lcptab[\tilde{i}]$	$bwttab[\tilde{i}]$	$S_{suftab[\tilde{i}]}$ *or* S_q
0	1	0	25	18 87 160 98 25 18 \$
1	6	1	25	18 \$
2	0	0	-	25 18 87 160 98 25 18 \$
3	5	2	98	25 18 \$
4	2	0	18	87 160 98 25 18 \$
5	4	0	160	98 25 18 \$
6	3	0	87	160 98 25 18 \$
7	7	0	18	\$

Table 2.1: The suffix array for motion string: [25, 18, 87, 160, 98, 25, 18, \$]

For a sequence of length p, the suffix array can be built in $\mathcal{O}(p \log p)$ time using an appropriate sorting algorithm. Besides, Kim *et al.* [KSPP05] proposed a linear time complexity algorithm for generating the suffix array. Other supplementary arrays are also constructed to enhance and optimize the string matching functionality. These arrays include:

1. The Burrows Wheeler Transform array, denoted by "bwttab", that stores the element before the first element of the corresponding suffix S_q. The $\tilde{i}^{th}$ element of bwttab is $bwttab[\tilde{i}] = S[suftab[\tilde{i}] - 1] = S[q-1]$. For example, $bwttab[5] = S[suftab[5]-1] = S[3] = 160$.

2. The longest common prefix array, denoted by "lcptab", that stores at the $\tilde{i}^{th}$ location, the length of the longest common prefix between $S_{suftab[\tilde{i}]}$ and $S_{suftab[\tilde{i}-1]}$. For example, the longest common prefix of $S_{suftab[3-1]}$ ($i.e.$ S_0) and $S_{suftab[3]}$ ($i.e.$ S_5) is $[25, 18]$. Therefore, $lcptab[3] = 2$.

The value stored in $lcptab[\tilde{i}]$ is the length of unique repeated sub-sequence when we have,

$$bwttab[\tilde{i}-1] \neq bwttab[\tilde{i}]. \tag{2.1}$$

Thus, there is only one unique repeated sub-sequence in this example, which is of length 2 and the sub-sequence is $[25, 18]$. In this manner, all unique matches within string S can be found using the suffix array technique.

To find similarity between two strings $S1$ and $S2$, they are concatenated into a new string S_c, denoted by $S_c = S1\#S2\$$, where symbol '#' is a unique logical separator between $S1$ and $S2$ but different from '\$'. Symbol '#' is assigned the second largest value in string S_c after '\$'. The same technique used to find the repeated sub-sequences for a single string is now applied to new string S_c with additional care. That is, one of the matching sub-sequence should begin before the '#' symbol in S_c and the other should begin after the '#'. It ensures that the match is not within $S1$ or $S2$ itself, but across strings $S1$ and $S2$. This additional constraint can be expressed in the following form:

$$\begin{aligned} &suftab[\tilde{i}-1] < |S1| < suftab[\tilde{i}],\ or \\ &suftab[\tilde{i}] < |S1| < suftab[\tilde{i}-1] \end{aligned} \tag{2.2}$$

where $|\cdot|$ is the cardinality of string $S1$.

2.1.3 Support Vector Machines (SVM)

To analyze the distribution of codewords in the spatial domain, encoded sequence are transformed into multi-resolution codeword histograms. These histograms are the features for training the SVM classifiers. Support vector machines [Wik04] are supervised learning models for regression analysis and binary classification. Assume the classification labels $y_j \in \{+1, -1\} \forall j$; the decision rule $\widetilde{C}(x_j) = sign(w^T \phi(x_j) + b)$; x_j to be the input feature vector for the j^{th} connected component, $\phi(\cdot)$ the mapping function, classifier $\widetilde{C}(\cdot)$ and w to be the parameter vector. Then the optimization algorithm trains the SVM classifier by minimizing the hinge loss function as shown.

$$\min_{w,b} \quad \sum_j max(0, 1 - y_j(w^T \phi(x_j) + b)) + \frac{\lambda}{2}\|w\|_2^2 \tag{2.3}$$

$$\min_{w,b} \quad C\sum_j max(0, 1 - y_j(w^T \phi(x_j) + b)) + \frac{1}{2}\|w\|_2^2 \tag{2.4}$$

$$\xi_n = max(0, 1 - y_j(w^T \phi(x_j) + b)) \quad \forall n \tag{2.5}$$

The last term in the above equation is the regularizer coefficient. SVM constructs a hyperplane with the largest separation between the two classes. Fig. 2.2 shows the case with linearly separable data. The vectors belonging to category-x and category-y are displayed in gray and yellow color respectively. The points lying on the dotted lines are the support vectors.

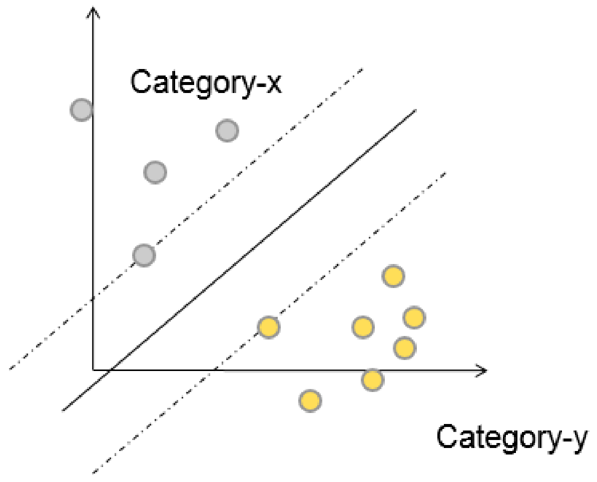


Figure 2.2: Support Vector Machines (SVM) binary classifier

The objective function can be rewritten in the well-known form as demonstrated in Eq. 2.4, where C parameter is equivalent to $\frac{1}{\lambda}$. The equation inside the summation sign can be replaced by ξ_n or the slack variable. This gives the primal formulation of the SVM introduced in Eq. 2.6 with the large margin constraints. In case of non-separable data, slack variables can still satisfy the large margin constraints.

$$\begin{aligned} \min_{w,b} \quad & C\sum_{j} \xi_n + \frac{1}{2}\|w\|_2^2 \\ such\ that, \quad & \xi_n \geq n \ \forall n \\ 1 - y_j(w^T \phi(x_j) + b) & \leq \xi_n \ \forall n \end{aligned} \tag{2.6}$$

2.2 Text Localization

In recent years, the task of locating text in natural scene images is attracting attention of researchers and industrial community alike, due to its inherent complexity and vast range of applications. Text region detection is a precursor to many computer vision tasks such as, optical character recognition, robotic navigation, compound video compression, scene understanding, text-based image indexing, search and retrieval etc. A well-known strategy is to narrow down the search space by finding candidate text regions, which requires efficient text segmentation techniques.

Text regions have some peculiar properties which distinguish them from the non-text regions. For example, text has sharp edges, interior pixels have consistent color, related component have distinct spatial alignment etc. Based on these observations, researchers have proposed several methods for text segmentation in natural images, such as, stable extremal regions detector (Sec. 2.2.1), stroke width transform (Sec. 2.2.2), morphological detectors etc. These detectors reduce the search space, but on the flip side, there are number of false positives to be dealt with. Gradient based features like gradient orientation histogram, gradient profiles and scale invariant feature transform codeword histogram can be used with the Adaboost enhanced decision tree classifiers to separate out the false positives. These features can be used for recognition task as well. The following sections provide a brief overview of some of these methods.

2.2.1 Stable Extremal Regions (SER) Detector

An efficient way of localizing text in natural scene images is to narrow down the search space by detecting Region Of Interest (ROI). Stable Extremal Region operator is perhaps the most widely used ROI detection tool in the text localization field. The pixels belonging to the same text region have similar intensities. SER operator tries to cluster those pixels together which constitute our ROIs. Suppose image I is a mapping $I : D \subset \mathbb{Z}^2 \rightarrow S$ for $S \in \{0, 1, \cdots, 255\}$. Then the region Q with boundary δQ is an extremal region, if it satisfies either the maximum or the minimum intensity region property:

$$\begin{aligned} &Q \subset D \quad such \quad that \quad \forall p \in Q, q \in \delta Q : I(p) > I(q) \qquad (maximum) \\ &Q \subset D \quad such \quad that \quad \forall p \in Q, q \in \delta Q : I(p) < I(q) \qquad (minimum) \end{aligned} \tag{2.7}$$

Let $Q_1 \cdots Q_{i-1}, Q_i \cdots$ be a sequence of nested extremal regions, *i.e.* $Q_i \subset Q_{i+1}$. $q(i)$ be mathematically represented by Eq. 2.8, where $\triangle \in S$ denotes the step size and $|\cdot|$ is the cardinality. The extremal region Q_{i*} is maximally/minimally stable if and only if, $q(i)$ has a local minimum at $i*$.

$$q(i) = |Q_{i+\triangle} \setminus Q_{i-\triangle}|/|Q_i| \tag{2.8}$$

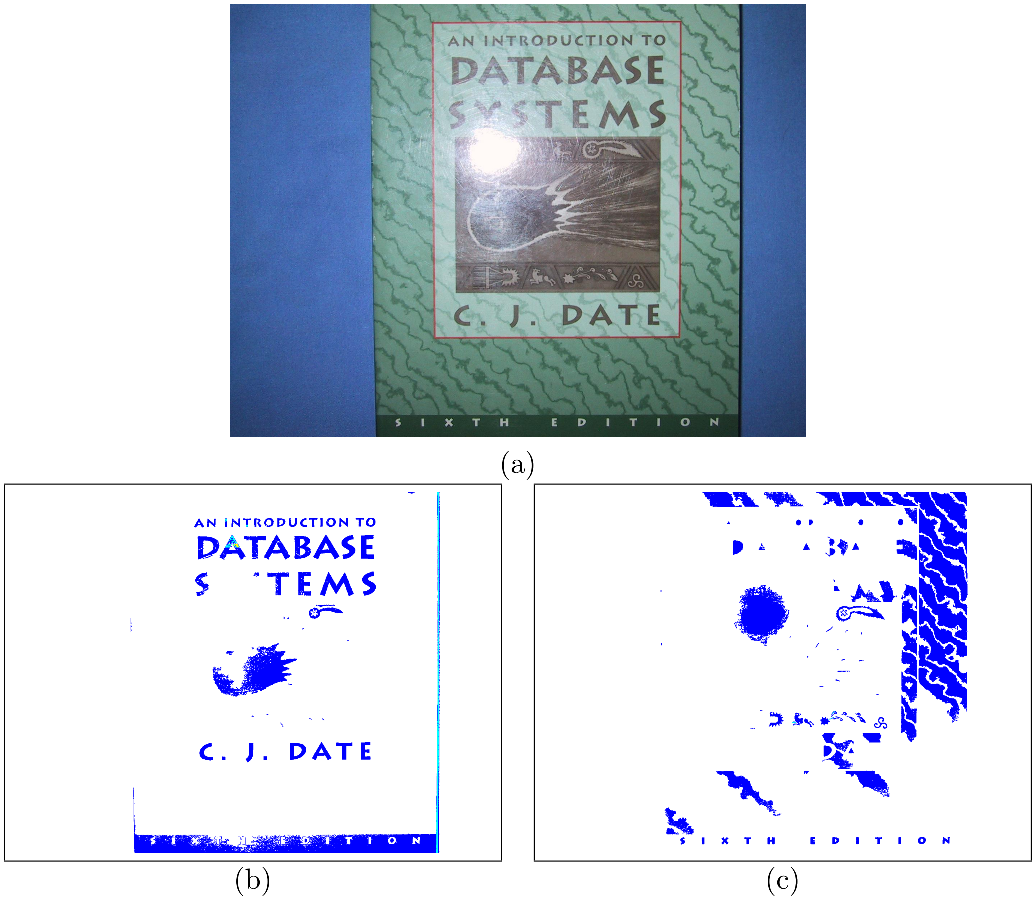


Figure 2.3: SER detector (a) Original image (b) Maximal regions in intensity image (c) Minimal regions in intensity image

For visual interpretation, consider an intensity image I and a variable threshold t. A binary image I_t is obtained by setting pixels p to black if $I(p) < t$ or white otherwise. Suppose t is varied from 0 to 255 and the resultant images I_0 to I_{255} are viewed as frames in a video, at first, only white pixels are seen. Subsequently, black spots will appear at the locations of local minima. These spots will grow and combine till the whole image becomes black. The connected components formed by those black pixels at various stages in the video are the extremal regions. Those extremal regions which do not grow rapidly over a wide range of thresholds are maximally stable. Repeating the process on an inverted image I_{inv} will fetch the minimally stable regions. For more details, refer to [MCUP04]. Text components generally have flat color tones and sharp contrast compared to the background. In other words, they are precisely the stable extremal regions, as they have flat color tones and sharp contrast. Fig. 2.3 shows the maximal extremal text regions like, "Database" and "C. J. Date" along with the minimal regions "Sixth" and "Edition".

2.2.2 Stroke Width Transform (SWT)

Stroke Width Transform or SWT is another widely used ROI detection technique in the text localization field. As the name suggests, stroke width transform [EOW10] computes the width of the stroke associated with each pixel. Stroke is the thickness of the brush used to draw the letters in the text. Letters within a word have similar thickness of the stroke. The pixels associated with these letters have the same stroke width values. These pixels can be grouped together to form ROIs.

Text components usually have sharp contrast with the background. So the gradient intensity values on the outline of the text regions are significantly higher than in the background or the interior regions. Fig. 2.4 depicts the gradient intensity at each pixel locations in the regions containing text. The length of the vector is proportional to the gradient intensity. It is evident from the figures that the pixels along the outline of the text have larger gradient values compared to the pixels in the background or the interior. Moreover, the gradients on the opposite edges of the stroke are anti-parallel. If it is a bright text component on a dark background as in Fig. 2.4(a), the gradients point towards each other. On the contrary, with dark text on bright background, the

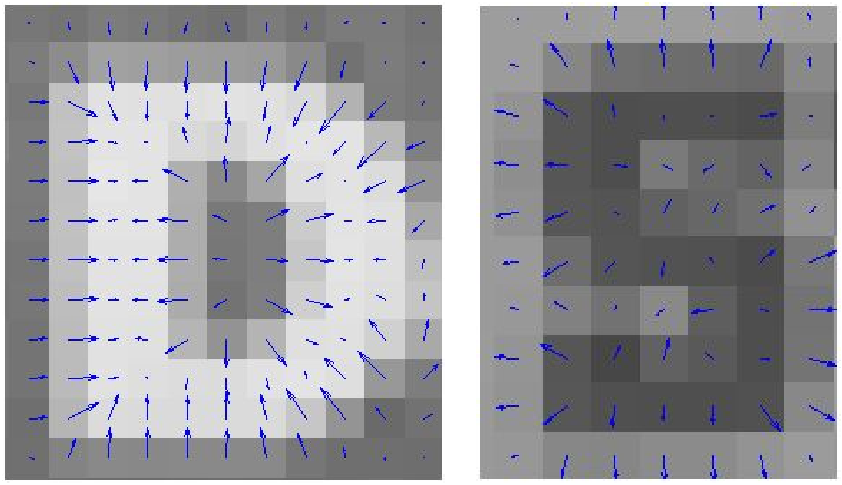

(a) (b)

Figure 2.4: Anti-parallel gradients (a) Gradient directions for bright text on dark background (b) Gradient directions for dark text on bright background

gradients are pointing away from each other. Refer to Fig. 2.4(b) for illustration. These key observations help us to compute the stroke width values at each pixel location.

The process of stroke width transform is depicted pictorially in Fig. 2.5. The top left section of the letter "D" (red box in Fig. 2.5(a)) is expanded in the subsequent images for demonstration. The Canny edge detector finds edge points in the image. Refer to Fig. 2.5(b). The gradient magnitudes and directions for those edge points are presented in Fig. 2.5(c). The stroke width transform matrix M_{in} has the same size as the input image. The initial values in the SWT matrix M_{in} are set to $+\infty$. For each edge pixel, we traverse along the direction of the gradient until an anti-parallel gradient is reached. The black pixels in Fig. 2.5(d) represent the pixels encountered during one such traversal. The process is repeated for other edge pixels in the image. The traversal paths for all the gradient pairs can be seen in different colors in Fig. 2.5(e). Finally all the pixels encountered during the traversal are replaced by the length of traversal in the SWT matrix M_{in}. This is the stroke width value for that pixel. If the traversal for a certain edge point does not lead to an anti-parallel edge, then that edge pixel is discarded and the corresponding stroke width values in the matrix are left unchanged.

The blank pixels of Fig. 2.4(f) have value of $+\infty$. The whole process is repeated again, but this time traversing in the direction opposite to the gradient. This gives another SWT matrix M_{opp} . The matrix obtained after taking the element-wise minimum of M_{in} and M_{opp} is the stroke width transform for that image. The pixels having similar stroke width values are grouped together to form ROIs.

2.2.3 Gradient Orientation Histogram

Gradient-based features are widely used in computer vision and pattern recognition for object classification. In the face recognition algorithms, gradients at the corresponding locations in different facial images are compared to detect facial similarities. But in text region classification, we are only interested in the histogram distribution of the gradient orientations and their magnitudes. The Gradient Orientation Pyramid (GOP) Magnitude Histogram feature serves this purpose.

Consider an image $\mathbf{I}(\ell)$, where ℓ represents the two-dimensional location of the pixel. The pyramid of the image $\mathbf{I}(\ell)$ is defined as $\wp(\mathbf{I}) = \mathbf{I}(\ell,\omega)^{p}_{\omega=0}$ at each scale ω [LSRJ10],

$$\mathbf{I}(\ell,0) = \mathbf{I}(\ell) \tag{2.9}$$

$$\mathbf{I}(\ell,\omega) = [\mathbf{I}(\ell,\omega-1) \otimes \Phi(\ell)]{\downarrow_2}, \omega = 1, 2 \cdots p \tag{2.10}$$

with $\otimes$ denoting the convolution operation and $\Phi(\ell)$ is the Gaussian kernel with the standard deviation of 0.5. $\downarrow_2$ denotes the down-sampling by a factor of 2 and p denotes the number of pyramid levels. For visual illustration refer to Fig. 2.2.3.

In the traditional GOP approach, the gradients for all the pixel position at all the scales of the image are normalized to a unit vector. This 2-dimensional normalized gradient image pyramid is stacked to create $d \times 2$ vector where d is the total number of pixels in the image at all scales. But in our approach, we compute the magnitude and the angle of the gradients of all the pixels in the image at all the scales. The angles are

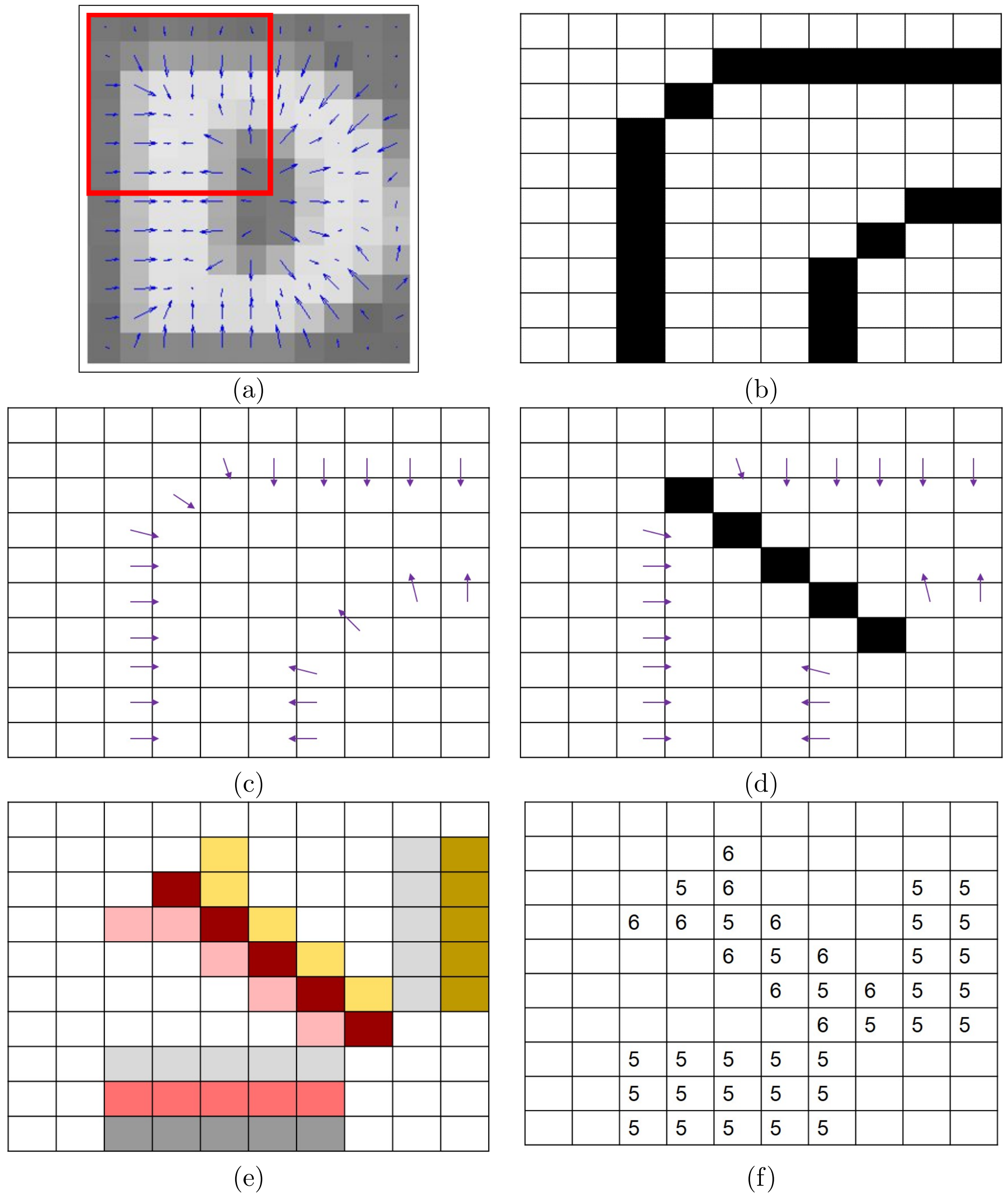


Figure 2.5: SWT or Stroke Width Transform (a) Section of image used for illustration of SWT (b) Edge pixels (c) Gradient magnitude and directions at the edge pixels (d) Traversal along the gradient direction to an anti-parallel edge (e) Different anti-parallel gradient pairs shown in different colors (f) Stroke width matrix M_{in}

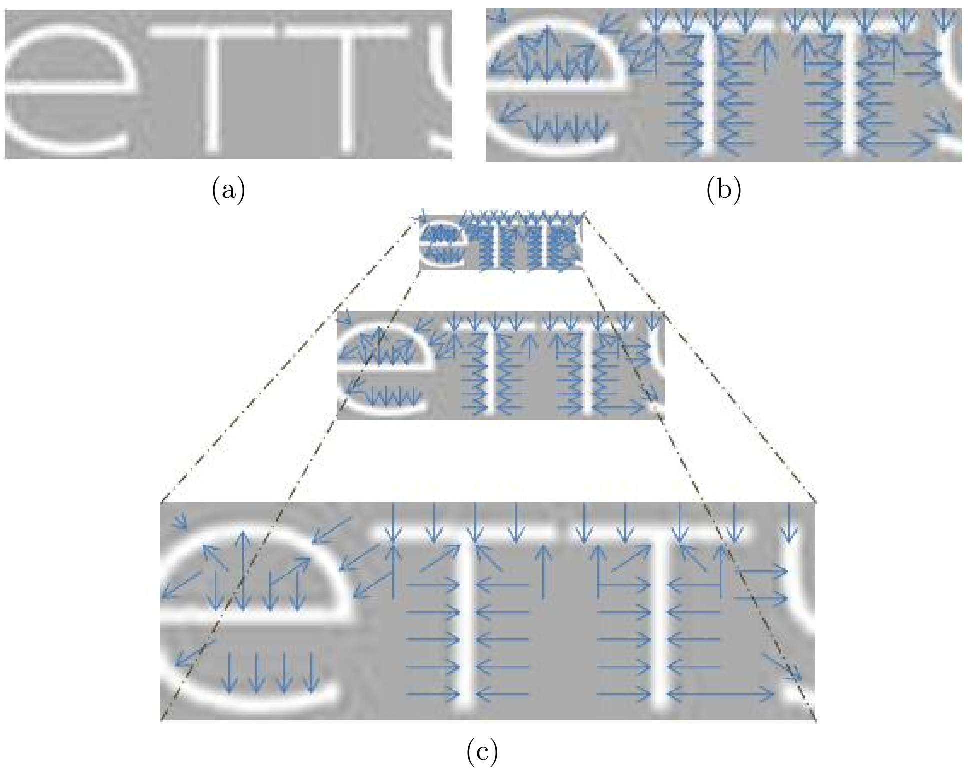


Figure 2.6: GOP or Gradient Orientation Pyramid features. (a)Training image $\mathbf{I}(\ell)$ (b)The Gradient Map $\mathbf{I}(\ell, 1)$ (c)Gradient Pyramid $\mathbf{I}(\ell, \omega), \omega = 1, 2, 3$

rounded off to the nearest integer between (0, 360°). The GOP Magnitude Histogram feature is then the histogram consisting 360 bins, one for every possible orientation angle, with the element in the bin weighted by the magnitude of the gradient.

2.2.4 Scale Invariant Feature Transform (SIFT) Codewords Histogram

Scale invariant feature transform (SIFT) is also a widely used gradient-based approach in computer vision for object and category classification [Low99], [Low04]. The SIFT extracts 128 dimensional local feature descriptors that are invariant to illumination, rotation and scaling. These features are especially useful in modeling the appearance of letters.

The images are convolved with Gaussian filters $\mathbf{G}(x, y, \sigma)$ of different scales as expressed mathematically in Eq. 2.11. $\otimes$ denotes the convolution operation. The difference between the adjacent Gaussian blurred images (Eq. 2.12) or Difference Of Gaussian (DoG) images *i.e.* $\mathbf{D}(x, y, \sigma)$ are searched for candidate points. The candidate points correspond to the extrema of DoG images across adjacent scales. The points with low contrast are eliminated. Each keypoint is assigned an orientation by computing a gradient orientation histogram in the neighborhood of that point. To provide rotation invariance, all the properties of the keypoint are measured with respect to its orientation.

$$\mathbf{L}(x, y, \sigma) = \mathbf{I}(x, y) \otimes \mathbf{G}(x, y, \sigma) \tag{2.11}$$

$$\mathbf{D}(x, y, \sigma) = \mathbf{L}(x, y, k\sigma) - \mathbf{L}(x, y, \sigma) \tag{2.12}$$

SIFT codeword histogram feature computation technique is described below. These features are used with the classifiers like Support Vector Machines, Decision Tree or Random Forests.

- The 128 dimensional SIFT key-point descriptors are computed for all the training images using the procedure outlined above.
- The keypoints from all the training images are analyzed to compute the principal component axes that capture maximum variance. The dimensionality of the keypoints is reduced by projecting those features to a lower dimensional sub-space spawned by the principal components.
- All the keypoints from all the training images are grouped together into distinct clusters using K-means clustering algorithm. Each cluster centroid is a codeword. For every image, each keypoint is matched to the nearest codeword. The histogram of these codewords is the SIFT codewords histogram for that image.

2.2.5 Adaboost

Adaboost is a meta-algorithm that combines many weak classifiers to arrive at complex decision boundaries. The algorithm can be used with any classifier. In the text localization framework, the decision tree classifiers are the weak classifiers used to build a strong ensemble classifier with Adaboost technique. Suppose N training pairs $\{\overline{x_n}, y_n\}$ are provided, where $\overline{x_n}$ denotes the feature vector and $y_n \in \{+1, -1\}$ be the classification label. Let the t^{th} weak classifier be depicted by $h_t(\overline{x})$ and all the training samples be initialized with equal weights *i.e.* $w_1(n) = \frac{1}{N} \quad \forall n$. During the t^{th} iteration, a weak classifier $h_t(\overline{x})$ is trained to minimize the classification error ϵ_t based on the current weights $w_t(n)$. The classification error ϵ_t is represented as,

$$\epsilon_t = \sum_n w_t(n) \mathbb{1}[y_n \neq h_t(\overline{x})] \tag{2.13}$$

$\mathbb{1}[\cdot]$ is the indicator function. The weight β_t for combining the weak classifier $h_t(\overline{x})$ is a function of the classification error as given by Eq. 2.14. Updates to the weights for next iteration are made according to the Eq. 2.15. The incorrectly classified samples get more weights in the next iteration. The new weights are normalized after every iteration so as to sum up to 1. Adaboost minimizes exponential loss function related to the classification error. The current classifier is constructed such that it minimizes the exponential loss.

$$\beta_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t} \tag{2.14}$$

$$w_{t+1}(n) \propto w_t(n) e^{-\beta_t y_n \overline{x_n}} \tag{2.15}$$

$$h[\overline{x}] = sign\big[\sum_{t=1}^{T} \beta_t h_t(\overline{x})\big] \tag{2.16}$$

The final output classifier (Eq. 2.16) is weighted combination of all the weak classifiers learned in the T iterations. The output classifier weighs in the predictions of the weak classifiers to make a final decision.

Chapter 3

Automatic Human Action Classification With Mocap Data

Abstract

Automatic classification of human action in motion capture (mocap) data has many commercial, biomechanical and medical applications and is the principal focus of this chapter. First, we propose a multi-resolution string representation scheme based on the tree-structured vector quantization (TSVQ) to transform the time-series of human poses into codeword sequences. Then, we take the temporal variations of human poses into account via codeword sequence matching. Furthermore, we develop a family of pose-histogram-based classifiers to examine the spatial distribution of human poses. We analyze the performance of the temporal and spatial classifiers separately. To achieve a higher classification rate, we merge their decisions and soft scores using novel fusion methods. The proposed fusion solutions are tested on a wide variety of sequences from the CMU mocap database using 5-fold cross validation, and a correct classification rate of 99.6% is achieved.

3.1 Introduction

The increasing demand for rendering smooth and plausible 3D motion is fueling the development of motion capture (mocap) systems. This new format of high quality 3D motion data has paved its way into animation movies, high-end computer games [CCY11], biomechanics, robotics, gait analysis and rehabilitation [SSL02] and machine translation of sign languages [LKL+04,YS04]. The diverse applications of mocap data and the rapid development of mocap systems have resulted in a large corpus of motion capture data in recent years. However, the large amount of raw data files makes it difficult to organize. It is desirable that both file names and the associated metadata should be able to provide a high level description of the contents of mocap sequences. Since manual annotation of mocap sequences is labor intensive, it is essential to develop an automated technique that can segment mocap data into homogeneous intervals, classify each interval into a basic action type, and index it for future retrieval. With such annotated mocap databases in place, a proper motion editing and authoring tool can be used to synthesize realistic motion sequences.

To develop efficient indexing and retrieval techniques for mocap databases, the first step is to classify the data into subsets according to their similarities, which is known as the action classification problem. As far as the human full-body motion is concerned, the mocap system records the actions of human actors by placing several markers on their body. These markers and a grid of infrared cameras help determine the full-body movement of those actors in terms of joint orientations. This joint orientation information constitutes the mocap data used in our research. Human mocap data are essentially time series of human body poses (Fig. 3.1). Different human motions may be of different time duration, tempo and style. Motions from same action type, *e.g.* walking, can vary from person to person. Moreover, visually similar motions may start or end with different joint orientations or even with different human body postures. A robust classification algorithm has to compensate for these irregularities in the data.

In our tree-structured vector quantization (TSVQ) based scheme, the dynamics of human motion is represented by a sequence of multi-resolution codewords. Apparently, two motions are similar if their corresponding codeword sequences are similar. To leverage this similarity, we analyze the distribution of these codewords in the temporal and spatial domains separately. Then, to overcome the individual limitations of these two approaches, we fuse their outcomes and soft scores to make final decision.

We examine the proposed solutions on a wide range of motions from the CMU mocap dataset [CMU07]. These motions are recorded in a controlled environment with only one performer per clip. Furthermore, the clips are assumed to be homogeneous, wherein every motion clip belongs to only one action category.

It is worthwhile to mention that a mocap data classification technique using hierarchical codewords was described in [KKK11], which is a 2-page summary of some intermediate results of our research. The algorithmic description was brief and experimental results were preliminary. Compared to [KKK11], the current work has a more complete treatment on this topic. Specifically, it has the following major contributions:

- Development of an advanced temporal domain approach to mocap data classification using codeword sequence matching;
- Development of a novel spatial domain approach for human mocap data classification with pose-histogram-based SVM classifiers;
- Performance evaluation on a larger dataset containing 278 human motion clips ($\sim$ 0.5 million frames) spanning 30 action categories;
- Achieving a correct classification rate of 99.6% for a data set comprising of simple, complex as well as perceptually similar action types.

The spatial domain approach takes the human skeleton structure into account so that it is restricted to human motions only. In contrast, the temporal domain approach does not assume any inherent structure in the data so it is more generic in this sense. Moreover, other formats of marker-based mocap data can be handled with the proposed algorithms if they can provide the 3D pose information. We will describe these new approaches, their implementations, fusion and experimental results in greater detail to offer readers a complete picture of our solution and its superior performance.

The rest of this chapter is organized as follows. Related previous work is reviewed in Section 3.2. Our advanced action classification techniques are described in Section

3.3. Experimental results are presented and analyzed in Section 3.4. Finally, concluding remarks and possible future extensions are summarized in Section 3.5. The readers are hereby informed that the words action and motion, are used interchangeably throughout the chapter.

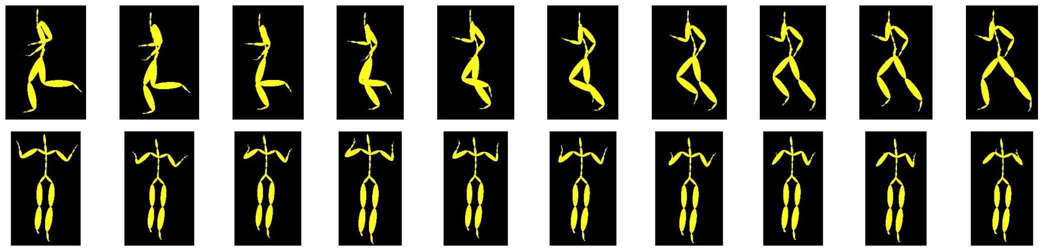


Figure 3.1: Illustration of ten consecutive poses of the running motion (top) and the teapot nursery rhyme rendition (bottom)

3.2 Review of Previous Work

Different methods have been proposed to synthesize realistic human motions by reusing existing mocap data via data-driven motion editing and authoring, *e.g.*, Kovar and Gleicher [KG04], Safonova *et al.* [SHP04], Ren *et al.* [RSH$^+$05], Lee *et al.* [LCR$^+$02], Zordan *et al.* [ZMCF05], and Pullen and Bregler [PB02]. In order to reuse existing mocap data, efficient searching, indexing and browsing techniques are required. So the first step is to group motion clips into subsets based on their similarities, which is the focus of our current research.

Identifying similar action types in a mocap database has been considered a challenging problem. Most of the previous work on motion comparison adopted features that were close to the raw data. For example, Liu and Popović [LP02] included the principal component analysis (PCA) reduced raw data, 3D point clouds, joint angles, 3D positions, etc. Forbes and Fiume [FF05] used the weighted PCA for motion comparison. The performance was further enhanced by integrating the weighted PCA with other features such as characteristic points, seed points, etc. Kovar and Gleicher [KG04] proposed a numerical similarity technique to find similar motion clips from a large data set. Motion similarity is a semantic concept. The raw numerical data may differ a lot even for two visually similar motions. Hence these types of features do not work well in practice.

Liu *et al.* [LZWP03] used automatically extracted key frames and a hierarchical tree of clusters of motions to search similar actions in the database. Sakamoto *et al.* [SKK04] proposed a motion map method that trains the self-organizing-map (SOM) with motion data and indexes clips by SOM nodes. The motion map transforms the N-dimensional data to a 2D map and computes similarity using this map. However, this scheme lacks higher level data indexing and demands exact node-to-node matching in search.

Human motion is typically a long time series with no overt segmentation information so the dynamic time warping (DTW) based motion comparison techniques (*e.g.*, Chiu *et al.* [CCW+04], Kovar and Gleicher [KG04], Wu *et al.* [WCYL03]) suffer from the complexity of finding the corresponding start/end points of motion data series for motion similarity comparison. Hsu *et al.* [HPP05] used an iterative motion warping (IMW) technique, which is an improvement over the traditional DTW technique, to find the correspondence by minimizing an objective function with dynamic programming. Generally speaking, these features work well only for certain applications.

Several algorithms were developed to identify locally similar segments in the motion dataset, for example, Arikan and Forsythe [AF02], Kim *et al.* [KPS03], Kovar and Gleicher [KG03], Lee *et al.* [LCR+02], and Wang and Bodenheimer [WB03]. Arikan *et al.* [AFB03] proposed a semi-automatic annotation technique using SVM classifiers. Cardle *et al.* [CVB+03] used the GEMINI framework to search mocap data sets. Yang and Shahabi [YS04] proposed a method called EROS to calculate similarity using PCA and eigenvalues. Li *et al.* [LZP07] used the kWAS method to measure similarity of motion clips via singular value decomposition.

Relational features [MRC05a,MRC05b,MR06,RKM08], built upon the Boolean relations between various body parts, are more suitable for generic applications. Common relational features include: the position of joints with respect to a certain plane, the angle between two limbs, the angular velocity of joint angles, the position of joints with respect to other joints, the direction of motion, touch, etc. Such features are more robust to pose distortion and other small aberrations in action sequences. Although relational features provide a good alternative, automatic selection of a good set of relational features for a wide range of motion types is not straightforward. This imposes a limitation on the utility of these features.

Wu *et al.* [WXWL09] presented a cluster-based scheme for mocap data indexing and retrieval, where temporal variations were accounted for while spatial variations were not captured effectively. Later, Wu *et al.* [WWX09] incorporated the spatial complexity of mocap data using hierarchical clustering. But this scheme cannot effectively differentiate between perceptually similar motions, such as *running* and *marching*. The different variations of the same action type (*e.g.* cartwheel) were also not clustered correctly. The SOM representation used in their scheme transforms the 62 dimensional mocap data into a crude 2D representation, which suffers from self occlusions. This significantly hinders its capability to categorize complex motions.

The main contribution of our work is automatic classification of perceptually similar mocap sequences based on the "multi-resolution string representation" that incorporates both spatial and temporal information effectively. It will be shown later in the experimental section that the proposed algorithm offers a correct classification rate of 99.6% for a database consisting of 30 different action types. To the best our knowledge, no other algorithm with better performance on such a diverse dataset has been reported in the literature.

Although our solution is promising, it has some limitations. For example, it might not work well on a clip containing actions from different categories, *e.g.* a clip with walking motion followed by running. However, with mocap data segmentation as the pre-processing step, this problem can be mitigated. The proposed solution was not thoroughly tested on categories with minor variations, such as, *get up from the floor without the hands touching the floor* and *get up from the floor by pushing hands against* *the floor*. As these categories are practically the same, we combined them into a single category in the experiment.

Researchers from the computer vision community have proposed a variety of methods for action recognition in unconstrained video databases such as UCF11 and UCF50. These methods used template matching, finite state models, bag of features, face detection, context, speech and text information etc. for action recognition [SC12, KS13]. But here, we work directly on the human skeleton posture information, which is available with the mocap data.

3.3 Proposed Classification Methods

We consider four mocap sequence classification methods.

- Method-A: Motion-string similarity

  The information about the temporal order of the poses is exploited in motion string similarity comparison using the suffix array technique. But due to the coarse quantization used, Method A is rigid and less robust. To overcome this problem, Method-B incorporated finer details.

- Method-B: Pose-histogram classifier

  TSVQ helps to compute the multi-resolution pose histogram for each motion clip. Multiple binary SVM classifiers, trained using these histograms, are used to generate soft scores for each test motion clip. One weakness of this method is that it does not use the temporal information at all. But Method-A compensates for that.

- Method-C: Two-Step Score Fusion

  The soft decision scores obtained from methods A and B are combined to make the final decision.

- Method-D: Two-Step SVM Fusion

  A more advanced SVM based fusion is used to combine the soft scores of methods A and B.

The details are given in the following subsections.

3.3.1 Method-A: Motion-String Similarity

Human mocap data is a high-dimensional time series where the data at each time instance (called a frame) represents the spatial location of markers in the 3D space [WWX09]. To eliminate the effect of bone sizes, this spatial data is converted to the rotational angles of joints. In simplified terms, the 3D trajectory of placed markers is stored as the trajectory of each degree of freedom (DOF) of the joint, over time [KCK10b, KCK10a]. Suppose that $r(t)$ is the 3D position of the root joint at time t, a_i is a scalar or vector representing the orientations of all the DOFs of joint i and N is the total number of joints in the human skeleton ($N = 29$). Then, a frame $f(t)$ corresponding to time t in the mocap sequence is a 62 dimensional vector which can be mathematically represented as

$$f(t) = \{r(t), a_1(t), a_2(t), \cdots, a_N(t)\}, t = 1, 2, \cdots \tag{3.1}$$

The first three dimensions (index range 1-3) representing the position of the root joint are neglected, as they are irrelevant to the 3D pose. The remaining 59 elements of vector $f(t)$ (index range 4-62) together form the "full-body" data vectors as shown in Table 3.1. We apply the Tree-Structured Vector Quantization (TSVQ) technique [Eff98, NK88] to the full-body data vectors in order to generate a full-body multi-resolution codebook as described below. The TSVQ performs clustering of full-body data vectors by repeated application of the Generalized Lloyd Algorithm (GLA). Iteratively, each cluster is split into two and fine tuned with another round of repeated application of the GLA to the perturbed centroids. After convergence, the parent codeword spawns two child codewords in the tree structure. Finally, a balanced binary tree-like structure is obtained. Fig. 3.2 shows the full-body level-n codebooks obtained using the TSVQ technique up to level $n = 3$. Owing to balanced binary tree structure, level n in the tree has 2^n codewords. All the full-body codewords at level-n together form the level-n codebook.

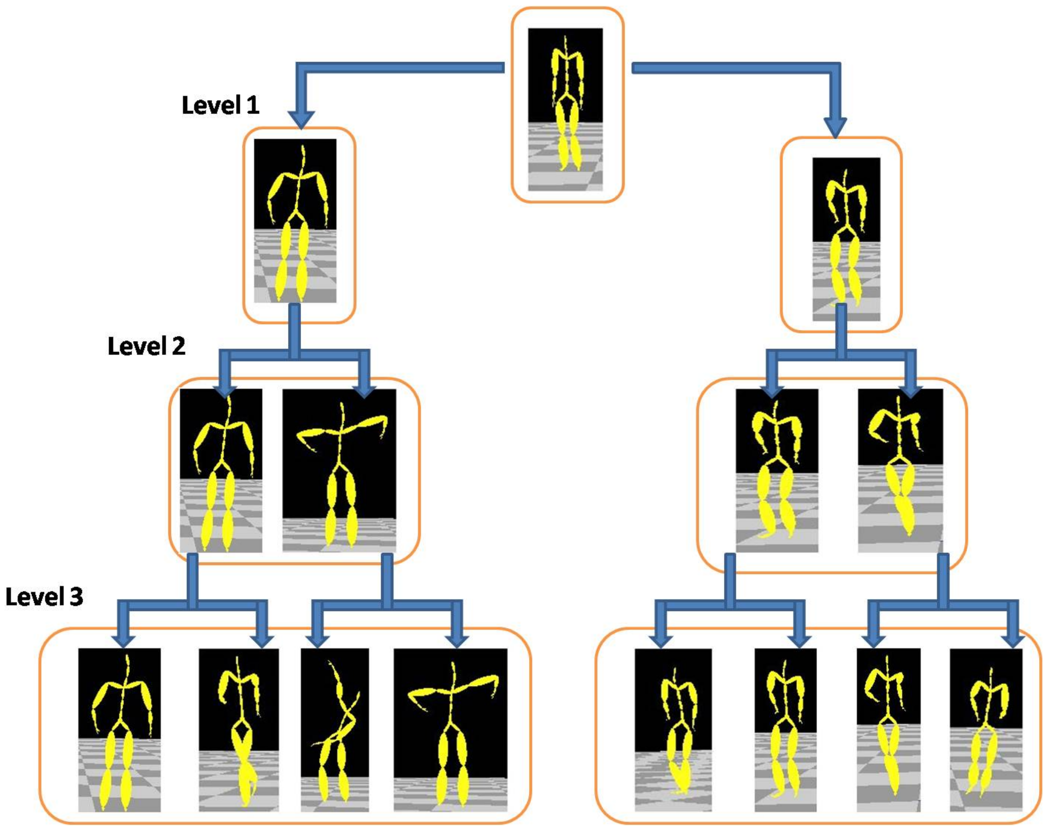


Figure 3.2: An illustration of full-body multi-resolution TSVQ codebooks.

Methods	Parts	No. of DOFs	Index Range
Method-A	Full-body	59	4-62
	Torso	21	4-24
	Right hand	12	25-36
Method-B	Left hand	12	37-48
	Right leg	7	49-55
	Left leg	7	56-62

Table 3.1: Partitioning of a human skeleton for classification.

This full-body level-n codebook encodes the mocap sequence by replacing each full-body pose vector with a codeword that is closest to it in terms of the Euclidean distance. For example, ten consecutive full-body poses of the running motion are shown in the top row of Fig. 3.1. If this partial sequence is encoded using a full-body level-9 codebook, each pose will be mapped to a codeword. Due to quantization, the mapped codewords can be the same for a set of consecutive frames. As a result, the mocap sequence can be compactly represented by a series of triplets (namely, a full-body codeword index number and the frame range). This series of codeword indices is called the "motion string". An illustration of the motion string representation is given in Fig. 3.3, which shows all frames in the aforementioned running motion sequence. The black horizontal segments indicate the group of consecutive frames that are mapped to the same full-body level-9 codeword. For the level-9 codebook, the codeword index ranges from 0 to 511 ($=2^9-1$).

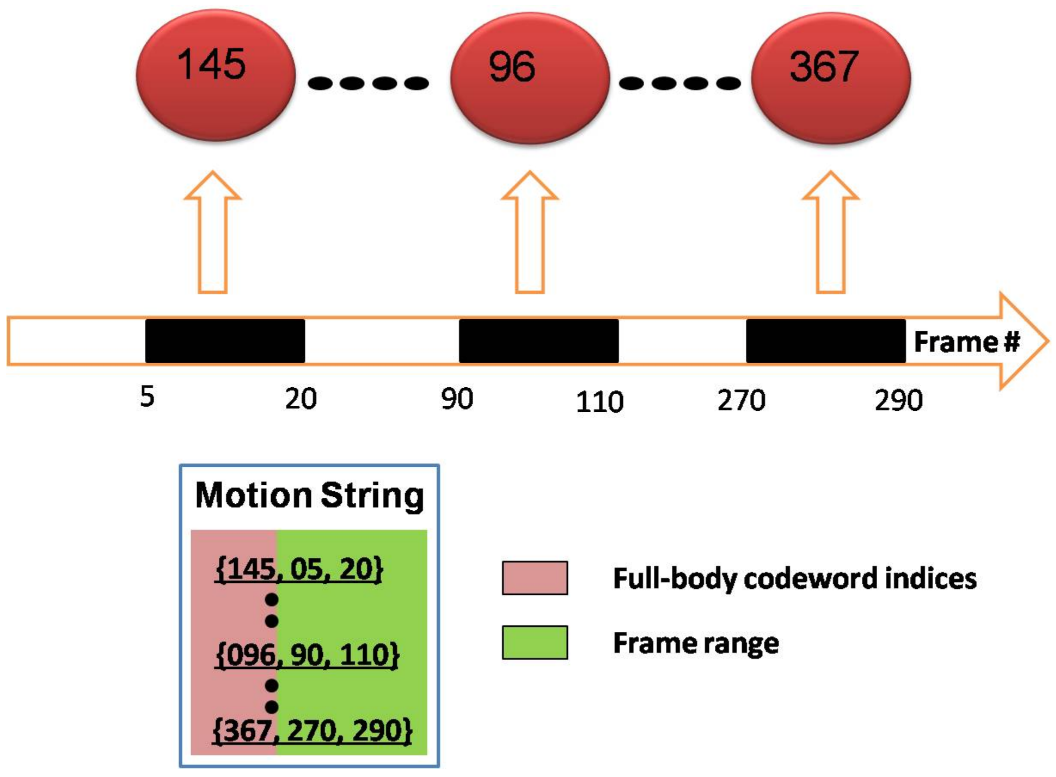


Figure 3.3: An illustration of the motion-string representation.

Apparently, two actions are similar if their corresponding motion strings are similar. The problem of finding similarity between two motion strings can be reformulated to a string matching problem. The latter can be efficiently solved by using the suffix array technique [AKO04, KS03, KLA+01]. In this technique, all suffixes of a string, S, are sorted in an ascending order. The sorted suffixes for a sample motion string, $S = [25, 18, 87, 160, 98, 25, 18]$, are shown in the last column of Table 3.2. Note that the end of the string is denoted by a unique symbol '\$'. The '\$' symbol is assigned to have the highest value among all codeword indices in the motion string. The suffix notation S_q represents the sub-sequence of S starting from $S[q]$ till the end of the string, followed by \$. Only the starting point *i.e.* index 'q' is enough to represent this suffix S_q. The array of indices 'q' defines the "suffix array", denoted by *suftab* in Table 3.2. We have $suftab[\tilde{i}] = q$ for suffix S_q that is at the $\tilde{i}^{th}$ position in the sorted list starting with $\tilde{i} = 0$. To give an example, S_4 represents the sub-sequence starting from $S[4] = 98$, which is $[98, 25, 18, \$]$. The suffix S_4 is at the fifth location ($\tilde{i} = 5$) in the sorted list. Therefore, $suftab[5] = 4$ in Table 3.2.

$\tilde{i}$	$suftab[\tilde{i}]$	$lcptab[\tilde{i}]$	$bwttab[\tilde{i}]$	$S_{suftab[\tilde{i}]}$ *or* S_q
0	1	0	25	18 87 160 98 25 18 \$
1	6	1	25	18 \$
2	0	0	-	25 18 87 160 98 25 18 \$
3	5	2	98	25 18 \$
4	2	0	18	87 160 98 25 18 \$
5	4	0	160	98 25 18 \$
6	3	0	87	160 98 25 18 \$
7	7	0	18	\$

Table 3.2: The suffix array for motion string: [25, 18, 87, 160, 98, 25, 18, \$]

For a sequence of length p, the suffix array can be built in $\mathcal{O}(p \log p)$ time using an appropriate sorting algorithm. Besides, Kim *et al.* [KSPP05] proposed a linear time complexity algorithm for generating the suffix array. Other supplementary arrays are also constructed to enhance and optimize the string matching functionality. These arrays include:

1. The Burrows Wheeler Transform array, denoted by "bwttab", that stores the element before the first element of the corresponding suffix S_q. The $\tilde{i}^{th}$ element of bwttab is $bwttab[\tilde{i}] = S[suftab[\tilde{i}] - 1] = S[q-1]$. For example, $bwttab[5] = S[suftab[5] - 1] = S[3] = 160$.

2. The longest common prefix array, denoted by "lcptab", that stores at the $\tilde{i}^{th}$ location, the length of the longest common prefix between $S_{suftab[\tilde{i}]}$ and $S_{suftab[\tilde{i}-1]}$. For example, the longest common prefix of $S_{suftab[3-1]}$ $(i.e.\ S_0)$ and $S_{suftab[3]}$ $(i.e.\ S_5)$ is $[25, 18]$. Therefore, $lcptab[3] = 2$.

The value stored in $lcptab[\tilde{i}]$ is the length of unique repeated sub-sequence when we have,

$$bwttab[\tilde{i} - 1] \neq bwttab[\tilde{i}]. \tag{3.2}$$

Thus, there is only one unique repeated sub-sequence in this example, which is of length 2 and the sub-sequence is $[25, 18]$. In this manner, all unique matches within string S can be found using the suffix array technique.

To find similarity between two strings $S1$ and $S2$, they are concatenated into a new string S_c, denoted by $S_c = S1\#S2\$$, where symbol '#' is a unique logical separator between $S1$ and $S2$ but different from '\$'. Symbol '#' is assigned the second largest value in string S_c after '\$'. The same technique used to find the repeated sub-sequences for a single string is now applied to new string S_c with additional care. That is, one of the matching sub-sequence should begin before the '#' symbol in S_c and the other should begin after the '#'. It ensures that the match is not within $S1$ or $S2$ itself, but across strings $S1$ and $S2$. This additional constraint can be expressed in the following form:

$$\begin{aligned} &suftab[\tilde{i}-1] < |S1| < suftab[\tilde{i}],\ or \\ &suftab[\tilde{i}] < |S1| < suftab[\tilde{i}-1] \end{aligned} \tag{3.3}$$

where $|\cdot|$ is the cardinality of string $S1$.

In our experiments, all training and testing action sequences are converted to their motion strings. A test motion string is compared with all training motion strings individually, using the suffix array technique discussed above and the following three parameters are computed for each testing-training string pair:

- **MLM**: the product of ratios of the **M**aximum **L**ength **M**atches to the total length, for the test and the training strings. The maximum length sequence 'l' is the maximum value in the lcptab array that satisfies Eqs. (3.2) and (3.3).

- **TEM**: the **T**otal number of **E**lements that **M**atch in the two strings. The total number of elements 't' is the sum of all the non-zero values in lcptab array that satisfy Eqs. (3.2) and (3.3).

- **SRP**: The **P**roduct of **S**imilarity **R**atios of the test string and the training string.

For example, the i^{th} test string has f codeword indices in its motion string while the j^{th} training string has g codeword indices. The t codeword indices in these two strings are identical and the largest common sequence is of length l. Then $TEM_{i,j} = t$ and

$$\begin{aligned} MLM_{i,j} &= (l/f \times 100)(l/g \times 100), \\ SRP_{i,j} &= (t/f \times 100)(t/g \times 100). \end{aligned} \tag{3.4}$$

As the calculation of $MLM_{i,j}$ and $SRP_{i,j}$ has its bottleneck in the suffix array formulation, their computational complexity is similar to that of the suffix array, with $p = f + g$. On the average, it takes around 0.5 ms to compare two strings using the suffix array technique on a 64-bit Windows Vista operating system with 4GB RAM and Intel Core 2 Duo CPU T6500 2.10 GHz.

For the i^{th} test string, the following two metrics are calculated with respect to a given action category k using the parameters in Eq. (3.4):

1. **max**-parameter: the average of MLMs of all training motions in category k;
2. **sim**-parameter: the average of the similarity product of all training motions in category k.

For the k^{th} category, we have

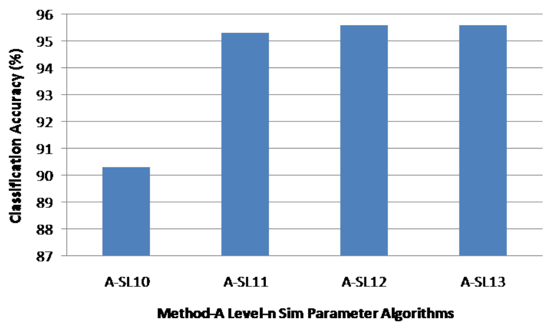
$$\begin{aligned} MAX_i^k &= \frac{\sum\limits_{j\in k} MLM_{i,j}}{\sum\limits_{j\in k} \mathbb{1}(j \in k)}, \\ SIM_i^k &= \frac{\sum\limits_{j\in k} SRP_{i,j}}{\sum\limits_{j\in k} \mathbb{1}(j \in k)}, \end{aligned} \tag{3.5}$$

where $\mathbb{1}(\cdot)$ is the indicator function.

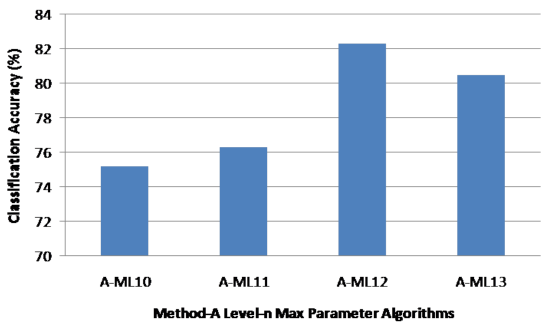
In standalone Method-A, the category with the highest parameter value wins. Alternatively, for each test motion, all categories are pitted against each other, one-on-one, and the category with the higher value for the parameter wins the vote. All category-specific votes are aggregated together and, then, normalized to get the soft scores. That is, we have

$$\begin{aligned} Vote_{i,MAX}^k &= \sum_{\forall m \neq k} \mathbb{1}(MAX_i^k > MAX_i^m), \\ Vote_{i,SIM}^k &= \sum_{\forall m \neq k} \mathbb{1}(SIM_i^k > SIM_i^m), \end{aligned} \tag{3.6}$$

for the k^{th} category with $m \in \{1, 2 \cdots 30\}$. The standalone Method-A classification results are presented in Fig. 3.4. The full-body level-n codebooks that offer the best classification results in standalone Method-A are short-listed for fusion to yield the final decision.



(a)



(b)

Figure 3.4: Standalone action classification performance of the motion string similarity comparison method: (a) overall performance using the sim parameter, (b) overall performance using the max parameter

3.3.2 Method-B: Pose-Histogram Classifier

Dynamic human behaviors are intrinsically low dimensional since legs and hands work in a coordinated fashion [SHP04]. According to biomechanics, a human skeleton can be decomposed into five functionally independent body-parts: right hand, left hand, right leg, left leg and torso. The mocap data inherently provides a clear demarcation between these body-parts. In other words, different portions of the mocap data vectors $f(t)$ can be associated with different body-parts. The first 3 elements give the root joint location. The next 21 elements (i.e. with index number 4-24) provide the information about the angular orientations of joints in the torso, the next 12 elements (i.e. with index number 25-36) for right hand and so on. The number of DOFs for each body-part and their associated index range in the vector $f(t)$, are shown in Table 3.1.

To leverage the spatial symmetry of human bodies, we propose a scheme that partitions a human skeleton into five body-parts using the information from Table 3.1. Then, TSVQ is applied to the DOF vectors of each body-part separately to generate five balanced tree structured codebooks, one for each limb (*i.e.* right hand, left hand, right leg and left leg) and one for the torso. These five body-part codebooks provide a multi-resolution description of the pose for each body-part. As opposed to the previous method which analyzes the entire body together using the "full-body" codebooks, this method analyzes "body-parts" separately using separate codebooks.

For each frame of a mocap sequence, every body-part is encoded using the corresponding level-n codebook, to get a string of five codewords. Because of the quantization effect, these strings do not change for a chunk of consecutive frames. This chunk can be represented by a set of five body-part codeword indexes as well as the frame range through which this chunk exists. For instance, Fig. 3.5 shows the frames in the *bend down and pick up a box* motion sequence encoded using level-5 body-part codebooks. The black portions in the time-line indicate the frames that have the same set of five body-part poses. Three frames selected from each of the chunks are shown above the time-line, and the level-5 codeword indices are shown below, where RH, LH, RL, LL and TR represent the Right Hand, Left Hand, Right Leg, Left Leg and Torso codewords, respectively. The codeword index numbers are in the range, 0 to 31 *i.e.* $(2^5 - 1)$ for each of the level-5 codebooks. All the frames in each chunk are represented collectively using a set of five body-part codeword indexes and the corresponding frame range. This compact notation is called the "pose string representation".

The pose string representation is used to obtain the body-part codeword histograms. The level-n histograms obtained for each body part (namely, right hand, left hand, right leg, left leg and the torso) for the j^{th} motion clip are concatenated to get the level-n pose histogram vector, denoted by $\overrightarrow{x_j}$. If $\overrightarrow{h_j}^{TR}$ is the level-n torso histogram for the j^{th} clip, $\overrightarrow{h_j}^{RH}$ is the level-n right hand histogram, and so on, then

$$\overrightarrow{x_j} = \{\overrightarrow{h_j}^{TR}, \overrightarrow{h_j}^{RH}, \overrightarrow{h_j}^{LH}, \overrightarrow{h_j}^{RL}, \overrightarrow{h_j}^{LL}\}. \tag{3.7}$$

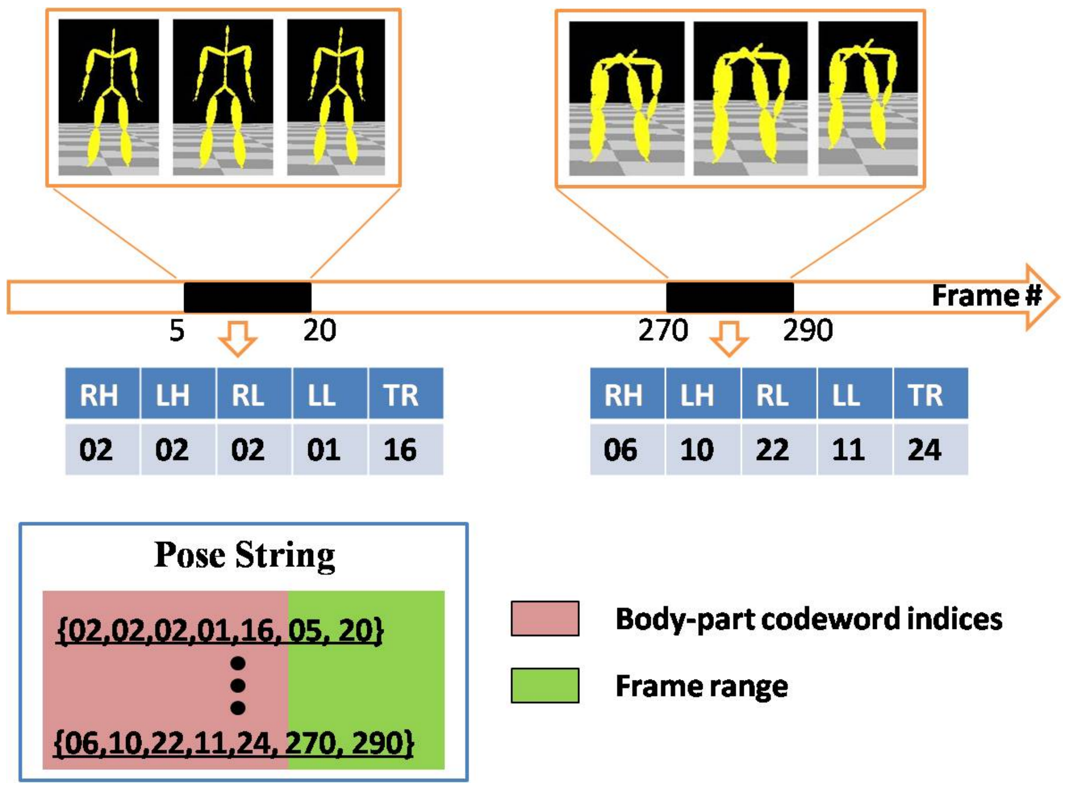


Figure 3.5: A pose string representation of a mocap sequence based on body-part codewords.

The pose histograms of motions belonging to the same action category are expected to be similar. The level-n pose histogram vector offers the information about the frequency of occurrence of level-n codewords for four limbs and the torso in the pose string representation of a given motion clip. Higher TSVQ levels in these histograms correspond to a higher resolution in the spatial domain. Motions belonging to the same type show similarity up to higher levels while motions of different types begin to differ at lower levels of the histogram. The number of levels can be gradually increased to get a better match between motions. However, a compromise has to be made, since similar motions will show differences if the level becomes too high. In other words, we start to see the characteristics of an individual motion clip rather than that of a action category.

Support vector machines [Wik04] are supervised learning models for regression analysis and binary classification. Assume the classification labels $y_j \in \{+1, -1\} \forall j$; the decision rule $C(x_j) = sign(w^T \phi(x_j) + b)$; x_j to be the input feature vector for the j^{th} motion, $\phi(\cdot)$ the mapping function, classifier $C(\cdot)$ and w to be the parameter vector. Then the optimization algorithm trains the SVM classifier by minimizing the hinge loss function in form of

$$\min_{w,b} \sum_j max(0, 1 - y_j(w^T \phi(x_j) + b)) + \frac{\lambda}{2} \|w\|_2^2, \tag{3.8}$$

where the last term λ is the regularizer coefficient. SVM constructs a hyperplane with the largest separation between the two classes.

For multi-class classification with $\tilde{m}$ classes, $\tilde{m}(\tilde{m} - 1)/2$ binary SVM classifiers are built. Every binary SVM classifier C_{ab} distinguishes between a pair of classes (a, b) where $a, b \in \{1, 2 \cdots \tilde{m}\}$ and $a \neq b$.

$$C_{ab}(x_i) = \begin{cases} +1 & \text{if } x_i \in a \text{ or} \\ -1 & \text{if } x_i \in b \end{cases} \tag{3.9}$$

In our case, $\tilde{m} = 30$. LIBSVM [CL11] trains these multiple binary SVM classifiers using the level-n pose histogram feature vectors and Gaussian RBF kernel. Refer to Eq. (3.7) for the interpretation of pose histogram vectors. The hyper-parameters of SVM classifiers are carefully tuned using cross validation on the training set. Regularization, careful tuning of hyper-parameters and cross validation help avoid overfitting. Every classifier then categorizes the test motion into one of two classes. The winning class gets that classifier's vote. For the k^{th} category and the i^{th} test motion, the total votes are calculated as

$$Vote_i^k = \sum_{b>k} \max(0, C_{kb}(x_i)) + \sum_{a<k} 1 - \max(0, C_{ak}(x_i)). \tag{3.10}$$

In standalone Method-B, the class that gets the maximum number of votes wins. The standalone Method-B classification results are depicted in Fig. 3.6. Alternatively, the votes of all classes are normalized to get $\tilde{m}$ soft decision scores (one for each category) for that test motion.

The performance of standalone Method-B which uses the spatial domain information is presented in Section 3.4. The classification results at various levels are analyzed to choose the best performing levels in the multi-resolution tree histograms. These shortlisted levels are then used in the decision fusion step.

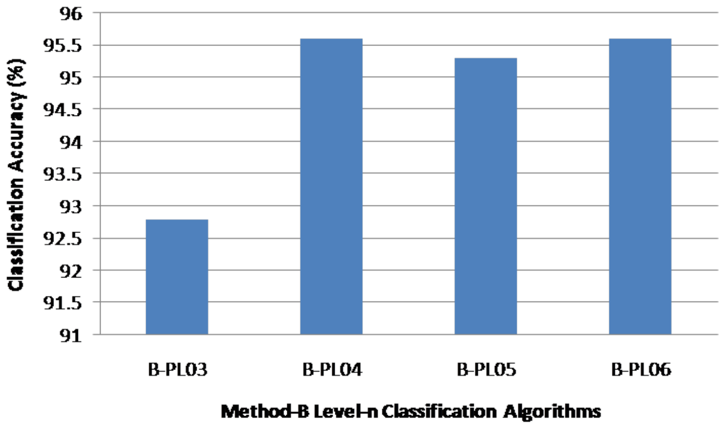


Figure 3.6: Standalone action classification performance of the pose histogram classifier method with level-n codewords, where $n = 3, 4, 5, 6$.

The motion string similarity (Method-A) inspects the order of the full-body poses while the pose histogram classifier (Method-B) examines the frequency of body-part poses. The two methods use completely different information to analyze the mocap data. These diverse perspectives complement each other in the fusion step to enhance the overall performance. It is worthwhile to point out that the codebooks for Method-A demand deeper levels ($n = [10, 11, 12, 13]$) in TSVQ as compared to that for Method-B ($n = [3, 4, 5, 6]$). This is because Method-A considers the full-body codebook and its level has to be deep enough to offer sufficient discriminant power. In contrast, Method-B uses specialized body-part codebooks that can reach sufficient discriminating capability with shallower levels in TSVQ.

3.3.3 Method-C: Two-Step Score Fusion

For simplicity, we use the following nomenclature to represent the selected algorithm and its parameter setting:

- **A** (or **B**): Method-**A** (or Method-**B**) classification algorithm;
- **MLn**: **M**ax-parameter with **L**evel-**n** TSVQ full-body codebooks (associated with Method-A only);
- **SLn**: **S**im-parameter with **L**evel-**n** TSVQ full-body codebooks (associated with Method-A only);
- **PLn**: **P**ose histograms with **L**evel-**n** TSVQ body-parts codebooks (associated with Method-B only).

The top two algorithms in Method-A (A-SL12, A-SL13) and Method-B (B-PL04, B-PL06), are shortlisted for fusion, in Method-C. Refer to the Tables 3.3, 3.4 and 3.5 for details. Given the individual outcomes of these four algorithms and their soft scores, Method-C makes the final decision on the action category using a two-step procedure as outlined below.

Step 1: Hard Decision Fusion The individual outcomes of the above-mentioned four algorithms (for a certain test motion) are taken together. If these outcomes are biased towards a specific category, then the test motion is classified into that category. Otherwise, we move on to the second step. For example, if algorithms A-SL12, B-PL05 and B-PL06 classify the motion into *category-M* while A-SL13 classifies the motion into *category-N*. Then, Method-C classifies the motion into *category-M* since this category has the maximum number of votes. If there is any tie of some sorts, we proceed to the second step to break the tie.

Step 2: Soft Score Fusion If there is a tie, soft scores of all these four algorithms are merged together as

$$\begin{aligned}
(A)^k_{i,Soft} &= \sum_{n\in\{12,13\}} (A\text{-}SLn)^k_{i,Soft}, \\
(B)^k_{i,Soft} &= \sum_{n\in\{4,6\}} (B\text{-}PLn)^k_{i,Soft},
\end{aligned} \tag{3.11}$$

where $k \in \{1, 2 \cdots \tilde{m}\}$ and i denote an action category and a test motion sequence, respectively, and subscript *Soft* indicates soft scores. The category getting the maximum score is assigned to that of test motion i; namely,

$$C_i = \max_{k\in\{1,2\cdots\tilde{m}\}} \{(A)^k_{i,Soft} + w * (B)^k_{i,Soft}\}. \tag{3.12}$$

where weight w can be determined heuristically by analyzing the variance of the soft scores and the standalone classification outcomes. For simplicity, w is set to 1 in our experiments. The performance results of Method-C are given in Table 3.6.

3.3.4 Method-D: Two-Step SVM Fusion

To improve the performance further, we developed another two step fusion approach based on SVM classifiers, *a.k.a.* Method-D. The details of this two step approach are given below.

Step 1: Hard Decision Fusion Similar to Method-C, the individual outcomes of four algorithms A-SL12, A-SL13, B-PL04 and B-PL06 (for a certain test motion) are taken together. If these outcomes are biased towards a specific category, the test motion is classified into that category. If there is a tie, we proceed to the second step.

Step 2: SVM Fusion For Method-D, we adopt a new mechanism to compute soft scores of Method-A and Method-B. For Method-A, we use the normalized SIM parameter values as the soft scores. For Method-B, we use the 'one-versus-rest' SVM classifier

$$C_k\{x_i\} = w^T\phi(x_i) + b, \tag{3.13}$$

where $\phi(\cdot)$ is the mapping function, b is a constant and $k \in \{1, 2 \cdots \tilde{m}\}$, to yield the soft score for feature vector x_i with respect to category k. A higher absolute positive score implies a higher possibility of test motion i belonging to class k while a higher absolute negative scores imply otherwise.

The soft scores, computed in this fashion for the four aforementioned algorithms *i.e.* A-SL12, A-SL13, B-PL04 and B-PL06, are combined to form the following four dimensional feature vectors:

$$\begin{aligned} \overrightarrow{p_i^k} = \{&(A\text{-}SL12)^k_{i,Soft}, (A\text{-}SL13)^k_{i,Soft},\\ &(B\text{-}PL04)^k_{i,Soft}, (B\text{-}PL06)^k_{i,Soft}\} \end{aligned} \tag{3.14}$$

Suppose j is the training motion. In the training phase, feature vectors p_j^k belonging to class k are used to build new 'one-versus-rest' SVM classifiers N_k for class k. In the testing phase, the four dimensional feature vectors p_i^k and the corresponding classifiers N_k are used to find the SVM scores $N_k\{p_i^k\}$ for i and its corresponding class k. Then, in Method-D, the category having the maximum SVM score, *i.e.*,

$$D_i = \max_{k \in \{1,2 \cdots \tilde{m}\}} \{N_k\{x_i\}\}, \tag{3.15}$$

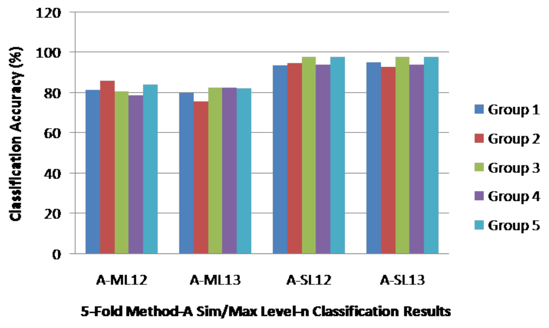
is the category of test motion i. The performance results of Method-D are presented in Table 3.6.

3.4 Experimental Results

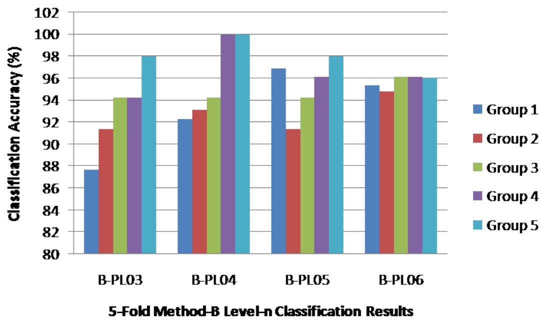
3.4.1 Dataset Description

Our experimental dataset contains a total of 278 high quality labeled motion clips (~ 0.5 million frames) belonging to 30 distinct action categories taken from the CMU mocap database [CMU07]. 33 different subjects were involved in recording these motions. The captured data has 62 dimensions of freedom (DOFs). The first three dimensions give the 3D position of the centroid of the human skeleton while the remaining 59 DOFs describe the angular positions of the joints as shown in Table 3.1. These 59 DOFs together define a full-body pose in the 3D space. The pre-processing step irons out the kinks in the full-body raw data file and also separates it into five body-part files for Method-B. The generated TSVQ codebooks are used to classify the motions from the dataset using n-fold cross validation process.

The n-fold cross validation method is adopted to evaluate the performance of the proposed classification algorithms. In this procedure, all motion clips of the same category are divided into n subsets. We choose an arbitrary subset as the test data and the other $n - 1$ subsets as the training data, and conduct the experiment to get the classification performance. Following the same procedure, we can perform n such tests using each subset as a test subset once. Finally, the classification results for all n tests are aggregated together. We set $n = 5$ in our experiment. Fig. 3.7 shows the cross validation results for each of the five groups.



(a)



(b)

Figure 3.7: (a) Group-wise performance using the sim and max parameters and (b) group-wise classification results for pose histogram classifiers.

3.4.2 Classification Performance and Discussion

We compared the performance of the four classification methods: 1) Standalone Method-A, 2) Standalone Method-B, 3) Method-C and 4) Method-D. The overall correct classification results for 278 test motions using the 5-fold cross validation with different level TSVQ codebooks and parameter settings are summarized in Table 3.3. More detailed classification results are listed in Tables 3.4, 3.5 and 3.6.

Stage	*Features*	*Accuracy*
Method-A only	A-ML12	82.3%
	A-ML13	80.5%
	A-SL12	95.6%
	A-SL13	95.6%
Method-B only	B-PL03	92.8%
	B-PL04	95.6%
	B-PL05	95.3%
	B-PL06	95.6%
Method-C	Score Fusion	98.2%
Method-D	SVM Fusion	**99.6%**

Table 3.3: Comparison of classification results.

In contrast with the pose histogram classification approach (Method-B), the motion string similarity comparison approach (Method-A) examines the transition of "coarsely quantized poses" in the temporal domain. Though there is an approximation, the temporal information helps in classifying some of the motions correctly, which the pose histograms missed. For instance, the misclassified *Kickball* motions are all correctly categorized by the temporal domain approach, indicating the fact that the sequence of poses do matter. Please check A-SL12 and B-PL06 columns in Tables 3.4 and 3.5, respectively. In addition, the temporal domain approach reinforces the confidence in the correctly classified motions. On the flip side, due to the coarse quantization used in the temporal domain approach, a lot of motions from the *Cock Robin* category are misclassified as shown in column A-SL13, Table 3.4. However, the finer resolution of the pose histogram classification algorithms give us correct results for those motions as shown in

Sr.No.	**Category(#Motions)**	**A – ML12**	**A – ML13**	**A – SL12**	**A – SL13**
1	Run(27)	26(96%)	26(96%)	27(100%)	26(96%)
2	Walk(47)	40(85%)	40(85%)	46(97%)	47(100%)
3	Forward Jump(9)	8(88%)	8(88%)	8(88%)	9(100%)
4	Forward Dribble(5)	5(100%)	5(100%)	5(100%)	5(100%)
5	Cartwheel(5)	5(100%)	5(100%)	5(100%)	5(100%)
6	Kickball(6)	6(100%)	6(100%)	6(100%)	6(100%)
7	Boxing(7)	0(0%)	0(0%)	6(85%)	7(100%)
8	Mickey Walk(7)	7(100%)	7(100%)	7(100%)	7(100%)
9	Sit and Stand(5)	4(80%)	4(80%)	5(100%)	5(100%)
10	Laugh(6)	4(66%)	6(100%)	6(100%)	6(100%)
11	Sweep Floor(5)	2(40%)	2(40%)	5(100%)	5(100%)
12	Wash Windows(5)	3(60%)	3(60%)	5(100%)	5(100%)
13	Climb Ladder(5)	5(100%)	5(100%)	5(100%)	5(100%)
14	Steps(7)	4(57%)	6(85%)	7(100%)	7(100%)
15	Eating(5)	5(100%)	5(100%)	5(100%)	5(100%)
16	Tiptoe(5)	5(100%)	3(60%)	5(100%)	5(100%)
17	Pick Box/Bend Waist(6)	6(100%)	6(100%)	5(83%)	6(100%)
18	Limp(5)	5(100%)	4(80%)	5(100%)	5(100%)
19	Balancing Walk(12)	10(83%)	9(75%)	12(100%)	12(100%)
20	Get Up From Chair(5)	4(80%)	4(80%)	5(100%)	4(80%)
21	Breast Stroke(6)	3(50%)	1(16%)	5(83%)	5(83%)
22	Hop on Left Foot(6)	4(66%)	6(100%)	6(100%)	6(100%)
23	Bouncy Walk(6)	4(66%)	4(66%)	6(100%)	6(100%)
24	Marching(10)	10(100%)	10(100%)	10(100%)	9(90%)
25	Rhyme Tea Pot(16)	13(81%)	13(81%)	12(75%)	12(75%)
26	Rhyme Cock Robin(15)	9(60%)	6(40%)	13(86%)	13(86%)
27	Swing(10)	10(100%)	10(100%)	9(90%)	10(100%)
28	Placing Tee(5)	5(100%)	5(100%)	5(100%)	4(80%)
29	Salsa Dance(15)	13(86%)	11(73%)	15(100%)	14(95%)
30	Get Up From Floor(5)	4(80%)	4(80%)	5(100%)	5(100%)
	Total(278)	**229(82.3%)**	**224(80.5%)**	**266(95.6%)**	**266(95.6%)**

Table 3.4: The standalone motion string similarity comparison results

column B-PL06 of Table 3.5. In this manner, these two diverse approaches complement each other.

Both the temporal and the spatial domain approaches have their own strengths and shortcomings. Due to this diversity, it is desired to combine them so as to improve the classification accuracy even further. Both the fusion approaches are able to do so. For example, the *Run* and the *Walk* motions which were not classified by either Method-A or Method-B are correctly classified by both the fusion approaches. This demonstrates the significance of proper fusion. Within these two fusion approaches, the SVM fusion

Sr.No.	**Category(#Motions)**	**B – PL03**	**B – PL04**	**B – PL05**	**B – PL06**
1	Run(27)	26(96%)	27(100%)	26(96%)	26(96%)
2	Walk(47)	46(97%)	46(97%)	46(97%)	46(97%)
3	Forward Jump(9)	8(88%)	8(88%)	7(77%)	9(100%)
4	Forward Dribble(5)	5(100%)	5(100%)	5(100%)	5(100%)
5	Cartwheel(5)	5(100%)	5(100%)	5(100%)	5(100%)
6	Kickball(6)	5(83%)	5(83%)	5(83%)	5(83%)
7	Boxing(7)	7(100%)	7(100%)	7(100%)	7(100%)
8	Mickey Walk(7)	6(85%)	7(100%)	7(100%)	7(100%)
9	Sit and Stand(5)	5(100%)	5(100%)	5(100%)	4(80%)
10	Laugh(6)	6(100%)	6(100%)	6(100%)	6(100%)
11	Sweep Floor(5)	4(80%)	5(100%)	5(100%)	5(100%)
12	Wash Windows(5)	5(100%)	5(100%)	5(100%)	5(100%)
13	Climb Ladder(5)	5(100%)	5(100%)	5(100%)	5(100%)
14	Steps(7)	7(100%)	7(100%)	7(100%)	7(100%)
15	Eating(5)	5(100%)	5(100%)	5(100%)	5(100%)
16	Tiptoe(5)	5(100%)	5(100%)	5(100%)	5(100%)
17	Pick Box/Bend Waist(6)	5(83%)	5(83%)	6(100%)	6(100%)
18	Limp(5)	5(100%)	5(100%)	5(100%)	5(100%)
19	Balancing Walk(12)	10(83%)	12(100%)	11(91%)	11(91%)
20	Get Up From Chair(5)	4(80%)	5(100%)	5(100%)	3(60%)
21	Breast Stroke(6)	6(100%)	6(100%)	5(83%)	5(83%)
22	Hop on Left Foot(6)	6(100%)	5(83%)	6(100%)	6(100%)
23	Bouncy Walk(6)	3(50%)	4(66%)	5(83%)	6(100%)
24	Marching(10)	10(100%)	10(100%)	10(100%)	10(100%)
25	Rhyme Tea Pot(16)	12(75%)	14(87%)	14(87%)	14(87%)
26	Rhyme Cock Robin(15)	14(93%)	13(86%)	12(80%)	15(100%)
27	Swing(10)	10(100%)	10(100%)	10(100%)	10(100%)
28	Placing Tee(5)	4(80%)	4(80%)	5(100%)	4(80%)
29	Salsa Dance(15)	15(100%)	15(100%)	15(100%)	15(100%)
30	Get Up From Floor(5)	4(80%)	5(100%)	5(100%)	4(80%)
	Total(278)	**258(92.8%)**	**266(95.6%)**	**265(95.3%)**	**266(95.6%)**

Table 3.5: The standalone pose histogram classification results

(Method-D) outperforms the score fusion (Method-C). This is mainly due to the fact that Method-C weighs all the features equally while Method-D is able to put more emphasis on the relevant features. This effect is more pronounced for the *Tea Pot* nursery rhyme category as shown in Table 3.6.

The performance results of the state-of-the-art mocap classification algorithms, tested on similar datasets, are presented in Table 3.7. Our classification results are better than the other. However, the direct comparison of all these algorithms with ours is difficult due to the following reasons.

Sr.No.	**Category(#Motions)**	**Method – C**	**Method – D**
1	Run(27)	27(100%)	27(100%)
2	Walk(47)	47(100%)	47(100%)
3	Forward Jump(9)	9(100%)	9(100%)
4	Forward Dribble(5)	5(100%)	5(100%)
5	Cartwheel(5)	5(100%)	5(100%)
6	Kickball(6)	6(100%)	6(100%)
7	Boxing(7)	7(100%)	7(100%)
8	Mickey Walk(7)	7(100%)	7(100%)
9	Sit and Stand(5)	5(100%)	5(100%)
10	Laugh(6)	6(100%)	6(100%)
11	Sweep Floor(5)	5(100%)	5(100%)
12	Wash Windows(5)	5(100%)	5(100%)
13	Climb Ladder(5)	5(100%)	5(100%)
14	Steps(7)	7(100%)	7(100%)
15	Eating(5)	5(100%)	5(100%)
16	Tiptoe(5)	5(100%)	5(100%)
17	Pick Box/Bend Waist(6)	6(100%)	6(100%)
18	Limp(5)	5(100%)	5(100%)
19	Balancing Walk(12)	12(100%)	12(100%)
20	Get Up From Chair(5)	5(100%)	5(100%)
21	Breast Stroke(6)	5(83%)	6(100%)
22	Hop on Left Foot(6)	6(100%)	6(100%)
23	Bouncy Walk(6)	6(100%)	6(100%)
24	Marching(10)	10(100%)	10(100%)
25	Rhyme Tea Pot(16)	12(75%)	15(93%)
26	Rhyme Cock Robin(15)	15(100%)	15(100%)
27	Swing(10)	10(100%)	10(100%)
28	Placing Tee(5)	5(100%)	5(100%)
29	Salsa Dance(15)	15(100%)	15(100%)
30	Get Up From Floor(5)	5(100%)	5(100%)
	Total(278)	**273(98.2%)**	**277(99.6%)**

Table 3.6: Decision fusion results

- Most of state-of-the-art algorithms tested their approach on their own datasets (or a mixture of CMU and self-generated datasets). We have no access to their datasets. Our approach uses the motions from the freely available online CMU mocap database [CMU07] only as listed in the Appendix.

- We adopted a stricter classification performance evaluation metric. For example, Wu *et al.* [WWX09] clustered the *running* and *marching* motions into one group as these motions are similar. In contrast, our evaluation metric requires that, if the labels of motions are different, they should be categorized into separate groups. Apart from that, some motions belonging to the same category are clustered into separate sub-groups in [WWX09]. As long as samples in each sub-group are homogeneous, it is treated as correct clustering. However, they are treated as misclassification in our performance metric.

- We have more action categories as compared to others. For example, our dataset contains both simple and complex motions from 30 categories while Wu's dataset had only simple motions from 14 categories.

Although we do not have a direct performance comparison because of different datasets, evaluation metrics and action categories used. The results in Table 3.7 still offer a rough idea of the performance of prior art.

Algorithms	*Accuracy*	*Dataset*
D-SBGPLVM [NPP$^+$13]	72.9%	CMU dataset
EROS [YS04]	87.5% [1]	Self generated
kWAS [LZP07]	90.3% [1]	Self generated
Wu *et al.* [WXWL09]	97.0% [1]	CMU + Self generated
Kadu *et al.* [KKK11]	97.0% [2]	CMU dataset
Wu *et al.* [WWX09]	98.1% [1]	CMU + Self generated
Proposed Method	**99.6%**	CMU dataset

Table 3.7: Performance comparison of human mocap data classification algorithms.

3.5 Conclusion and Future Work

A technique for automated mocap data classification was presented in this work. The TSVQ method was adopted to approximate static human poses with codewords while a

[1]Refer to Wu *et al.* [WWX09] for clustering accuracy.

[2]Refer to Kadu *et al.* [KKK11].

dynamic human motion was represented by a sequence of codewords. Fusion approaches were proposed to classify mocap data into different categories. We tested the proposed algorithms on the CMU mocap database using the 5-fold cross validation procedure and obtained a correct classification rate of 99.6%.

The proposed algorithm can be extended to various tasks required by mocap database management such as segmentation, indexing and retrieval, without much effort. We only sketch the basic ideas below.

- **Segmentation**

  If a complex motion clip contains multiple basic motions, then the sequence of these basic motions will be present in the sequence of that complex motion. By using string matching, we can find out the location of sequences of all basic motions in the complex motion. This corresponds to the segmentation of a complex action sequence into multiple basic action sequences.

- **Indexing**

  After running the classification algorithm, unknown motion clips can be classified into one of the known categories and indexed accordingly. Consider a mixed motion that has running, jumping and bending actions one after the other. Using the string matching and the database of sequences, the location of all these basic motions in that mixed motion can be determined and these basic actions can be indexed accordingly.

- **Retrieval**

  The retrieval problem can be solved using suffix array technique. To be more specific, for a given test motion, we would like to extract motion clips or parts of motion clips from the database that are similar to the query. To perform this retrieval task, we convert the test motion into a sequence and use the string matching algorithm to search the motion clips or portions of the motion clips that have the same sequence. Those clips can then be retrieved.

There are several algorithms for extracting human skeleton from depth images produced by Microsoft Kinect [SSK$^+$13, AD13]. The data generated by these algorithms are similar to the mocap data [CK13]. It is interesting to explore whether the proposed methodology can be extended to the Kinect sensor data as well. High quality 3D rendering can be achieved by blending stored mocap data with the pose information from the Kinect sensor inputs.

Chapter 4

Text Localization In Natural Scene Images

Abstract

Searching for text regions in natural images is a challenging task for many computer vision applications. In this research, we propose a novel text localization scheme based on multi-stage incremental region classification technique. The stable extremal region detector investigates peculiar characteristics of text to discover regions with possible textual content across different channels. The coarse decision stump classifiers designed on geometric features and context-based text grouping stages efficiently remove false positives and outline the regions of interest. An ensemble of trained decision tree classifiers categorizes the remaining regions into text or non-text using the gradient profile features. Finally, the ROIs from different views and channels are fused together to procure a consolidated list of text regions. As per the experimental results, our suggested technique is among the top algorithms reported in the literature.

4.1 Introduction

In recent years, the task of locating text in natural scene images is attracting attention of researchers and industrial community alike, due to its inherent complexity and vast range of applications. Text region detection is a precursor to many computer vision tasks such as, optical character recognition, robotic navigation, compound video compression, scene understanding, text-based image indexing, search and retrieval etc. The aim of our text localization research is to detect text regions in the image and draw bounding boxes around each and every word. Contrary to the scanned documents, text in natural images have different sizes, fonts, orientations, illumination and colors. The cluttered background also poses a serious threat to the localization accuracy. This diversity and irregularity makes text localization in these circumstances quite difficult. The fact that, no algorithm has yet been able to achieve a considerable accuracy on the benchmark datasets, further illustrates the challenging nature of the problem.

Text regions have some peculiar properties which distinguish them from the non-text regions. For example, text has sharp edges, interior pixels have consistent color, related component have distinct spatial alignment etc. Based on these observations, we propose a multi-stage region classification model, that systematically detects individual words in the images. This multi-stage design for text localization enhances efficiency and reduces complexity. After detecting probable candidates in the first stage, the task of eliminating false alarms is distributed across subsequent stages of the framework. Each stage targets specific properties of text to come up with its own set of discriminative features. The commendable performance justifies the effectiveness of this incremental approach.

The stable extremal region (SER) detector [MCUP04], [CTS+11] explores intra-component color similarity, to discover potential text candidates. But these detected regions may contain several false positives. Geometric filtering phase removes most of these false alarms. Later on, the context-based text grouping stage combines the remaining components into words, by exploiting the inter-component consistency and spatial alignment information. The text-like false positives are gradually phased out of the pipeline. An ensemble of decision tree classifiers, then decides the fate of these newly formed word candidates. These weak decision tree classifiers are built on gradient profile features and combined into an ensemble using the Adaboost framework. The whole process is repeated for different channels of the image and the detected words are aggregated. Experimental results on the ICDAR dataset [LPS+03, Luc05, SSD11] show that our performance score is comparable to the state-of-the-art algorithms.

The major contributions of our research are listed below:

- Designed the stripe-scan features for removing text-like false positives
- Proposed the scan-line stroke width transform technique for efficient computation of stroke width values on binary components
- Developed a robust text grouping mechanism which is immune to missing components
- Built the novel gradient profile features that capture the peculiarities of text regions
- Demonstrated the effectiveness of the approach by achieving excellent results on publicly available datasets

The rest of this chapter is organized as follows. Related previous work is summarized in Sec. 4.2. Sec. 4.3 presents the intricate details of our proposed text localization framework. Experimental results are provided in Sec. 4.4 along with the performance analysis. Finally, Sec. 4.6 has the concluding remarks and future research directions.

4.2 Review of Previous Work

The techniques for text localization in scanned document can be broadly classified into three main categories: top-down, bottom-up and hybrid approaches [ODP99]. The top-down approach [LCC00], [KNSV93], [IKA87], [CYWM03] starts by detecting the page schematic and then successively splits up the cells to reach lower levels. For instance, Khedekar [KGS03] proposed a projection profile based algorithm using this top-down approach to separate images and text in the Devnagri documents. However, due to the dependence on page layout information, these methods are suitable only for the scanned document images. The bottom-up approach [Bix98], [Mak83] on the other hand, starts merging together the components at the lowest level and forms larger structures as it goes up. Though these methods are flexible, they suffer from accumulation errors [KGS03].

The hybrid approach based methods [JZ95], [CLKH96] and [JB92] try to allow some leeway in the above two approaches. As opposed to scanned document, the natural scene images do not have a fixed layout. Text can appear anywhere in the image and the orientation, font-size or color may vary from one region to the other. For these reasons, the techniques for scanned documents cannot be easily extended to natural images.

One group of text localization schemes for natural scene images use sliding window technique to search for relevant texture patterns. The sliding window methods [CY04], [LW02], [GEF04], [LDK00], [YHGZ05] employ local features within the windows of varying sizes and resolution, to classify windows as text or non-text. Apart from limiting text search to specific window sizes or orientations, these methods have high computational complexity due to different possible choices of image scales, window sizes and aspect ratios. Following this school of thought, Chen [CY04] devised a procedure of detecting regions of text in the city scenes. The approach is similar to the face detection framework proposed by Viola-Jones [VJ01a], [VJS05]. Chen used weak classifiers to build the strong text/non-text classifiers using AdaBoost machine learning algorithm [FS96]. An adaptive binarization and extension algorithm was then applied to the selected regions in order to mark out the text. This paper used the block features similar to the ones described in [VJ01b] and was designed specifically for reading the text on the signboards, billboards etc.

The other group of techniques detect regions of interest by merging similar pixels into connected components [NM13], [NM12], [LDL05], [JKJ04], [JY98], [Kim96], [YI06]. These methods are not constrained by orientation of text, font size or resolution. They are computationally tractable and the text segmentation is precise to the granularity of individual letters. This aids in subsequent character recognition using OCR techniques [BMP01], [GRL$^+$98], [RWH96]. Major disadvantages include false alarms, threshold selection, spurious connections, sensitivity to noise and background clutter. One such method is the stroke width transform (SWT) [EOW10], [HLYW13] that leverages the fact that letters in images have constant stroke width. The gradients on either side of the stroke are anti-parallel. Pairing those anti-parallel gradients help in finding the regions of interest. Neumann and Matas [NM12] proposed maximally stable extremal regions (MSER) based technique for detecting text regions. It relies on the fact that pixels within the text have similar intensities. Our proposed framework builds on the stable extremal region detector, described next, but diverges significantly thereafter. Refer to Fig. 4.1 for the schematic representation, where R_e stands for estimated rectangles while T and NT represent text and non-text components respectively.

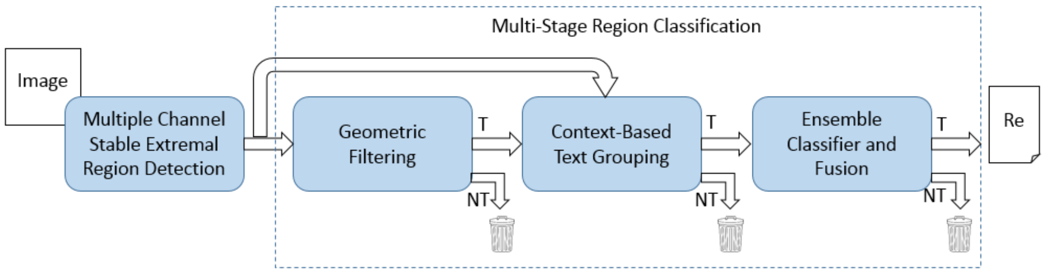

Multi-Stage Region Classification
Image
Multiple Channel Stable Extremal Region Detection
Geometric Filtering
T
NT
Context-Based Text Grouping
T
NT
Ensemble Classifier and Fusion
T
NT
Re

Figure 4.1: Schematic diagram of the proposed text detection algorithm

4.3 Proposed Algorithm

The four main stages of the proposed text localization algorithm are outlined below. The detailed description is included in the subsections that follow.

1. *Multiple Channel Stable Extremal Region (SER) Detector:* In natural scene images, the pixels belonging to a letter or word component have consistent gray scale intensity or color tone. These regions are effectively detected by the stable extremal region operator. Multiple color channels improve detection results, but the number of false alarms also increase.

2. *Geometric Filtering:* Second stage examines the structural characteristics of text to get rid of false detections. Bulk of the false positives are efficiently removed in this stage without significant computational overhead.

3. *Context-Based Text Grouping:* The spatial distribution of text blobs in an image is constrained by the syntax of the script. In general cases, text is laid out horizontally. Components belonging to a given line of text have similar size, font, color and stroke widths. This alignment and similarity information helps to discover text lines in the image. On the other hand, the non-text blobs neither have a specific spatial arrangement nor the neighborhood similarity. Such scattered components can be regarded as non-text and removed.

4. *Ensemble Classifier and Fusion:* To prune the false positives further, an ensemble of decision tree classifiers is trained on samples from the training dataset using the gradient profile features. Finally the detections from multiple views and channels are concatenated together to get the aggregated list of bounding boxes.

In the algorithm description that follows, the terms components, blobs and objects are used interchangeably.

4.3.1 Multiple Channel Stable Extremal Region (SER) Detector

Suppose image I is a mapping $I : D \subset \mathbb{Z}^2 \rightarrow S$ for $S \in \{0, 1, \cdots, 255\}$. Then the region Q with boundary δQ is an extremal region, if it satisfies either the maximum or the minimum intensity region property:

$$\begin{aligned} &Q \subset D \quad such \quad that \quad \forall p \in Q, q \in \delta Q : I(p) > I(q) \qquad (maximum) \\ &Q \subset D \quad such \quad that \quad \forall p \in Q, q \in \delta Q : I(p) < I(q) \qquad (minimum) \end{aligned} \tag{4.1}$$

Let $Q_1 \cdots Q_{i-1}, Q_i \cdots$ be a sequence of nested extremal regions, *i.e.* $Q_i \subset Q_{i+1}$. $q(i)$ be mathematically represented by Eq. 4.2, where $\triangle \in S$ denotes the step size and $|\cdot|$ is the cardinality. The extremal region Q_{i*} is maximally/minimally stable if and only if, $q(i)$ has a local minimum at $i*$.

$$q(i) = |Q_{i+\triangle} \setminus Q_{i-\triangle}| / |Q_i| \tag{4.2}$$

For visual interpretation, consider an intensity image I and a variable threshold t. A binary image I_t is obtained by setting pixels p to black if $I(p) < t$ or white otherwise. Suppose t is varied from 0 to 255 and the resultant images I_0 to I_{255} are viewed as frames in a video. Then at first, only white pixels are seen. Subsequently, black spots will appear at the locations of local minima. These spots will grow and combine till the whole image becomes black. The connected components formed by those black pixels at various stages in the video are the extremal regions. Those extremal regions which do not grow rapidly over a wide range of thresholds are maximally stable. Repeating the process on an inverted image I_{inv} will fetch the minimally stable regions. For more details, refer to [MCUP04]. Text components are generally the stable extremal regions, as they have flat color tones and sharp contrast. Fig. 4.2 shows the maximal extremal text regions like, "Database" and "C. J. Date" along with the minimal regions "Sixth" and "Edition".

In order to incorporate color information, the $\mathbf{A}^*$ and $\mathbf{B}^*$ channels of the CIELAB color space are examined along with the gray scale intensity image I. The $\mathbf{A}^*$ and $\mathbf{B}^*$ channels have a different range from the gray scale images *i.e.* $S = \{-128, \cdots, +127\}$ but the procedure is same as before. Apart from being device independent, the CIELAB is the most complete space which describes all the colors perceptible to the human eye. The non-linear mapping and wider gamut allows better differentiation between basic colors, which facilitates text detection even under adverse illumination conditions. The detection results from three channels, each with two views *i.e.* $\mathbf{I}$, $\mathbf{I_{inv}}$, $\mathbf{A}^*$, $\mathbf{A}^*_{\mathbf{inv}}$, $\mathbf{B}^*$ and $\mathbf{B}^*_{\mathbf{inv}}$ are considered in our approach. Fig. 4.3 demonstrates the significance of multiple channels. Even as the intensity image fails, $\mathbf{A}^*$ and $\mathbf{B}^*$ channels are able to detect the text "Royal London" in the image. But on the contrary, multiple channels tend to increase the number of false positives as well.

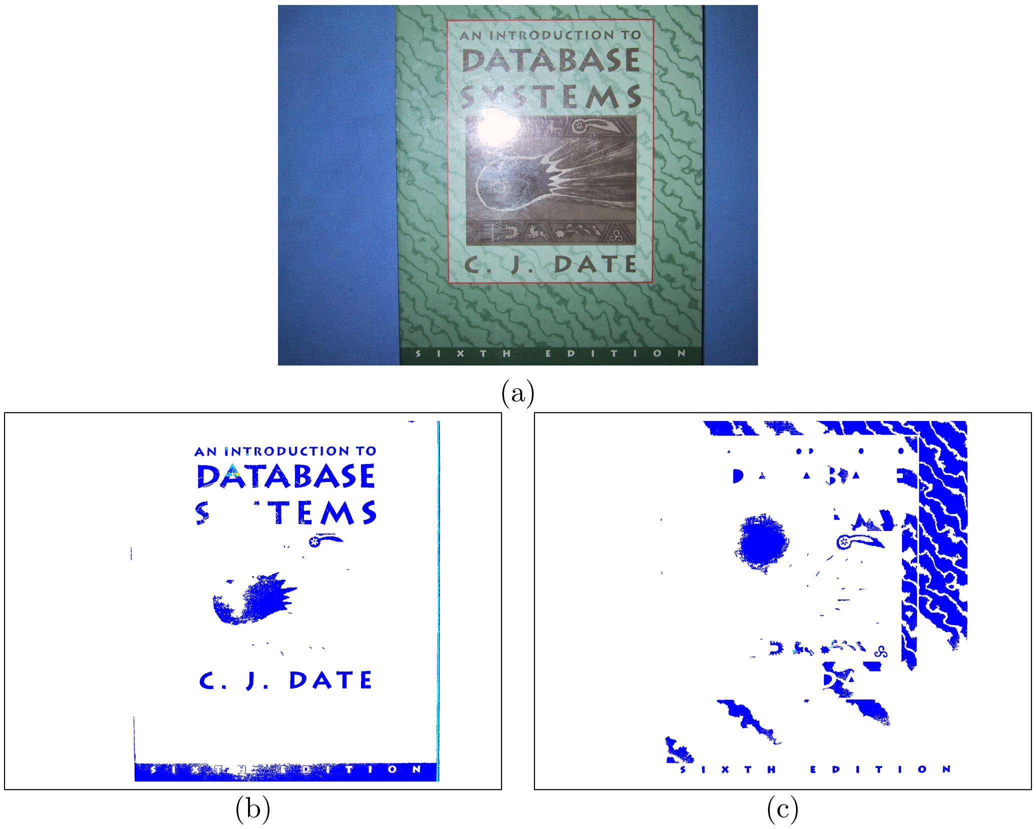


Figure 4.2: SER detector (a) Original image (b) Maximal regions in intensity image (c) Minimal regions in intensity image

4.3.2 Geometric Filtering

The text components exhibit a peculiar structure, continuity and compactness that is seldom perceptible in the non-text components. The geometric filtering stage leverages this fact to remove a majority of false positives. Each binary component is cropped and examined individually to compute the following analytical features. Owing to the large number of false positives, there features are purposely designed for fast processing.

4.3.2.1 Stripe Scan Feature

The binary component is broken down into non-overlapping vertical and horizontal rectangular sections *a.k.a.* stripes. For instance, the component $\widehat{C}$ is segmented into $\tilde{v}$

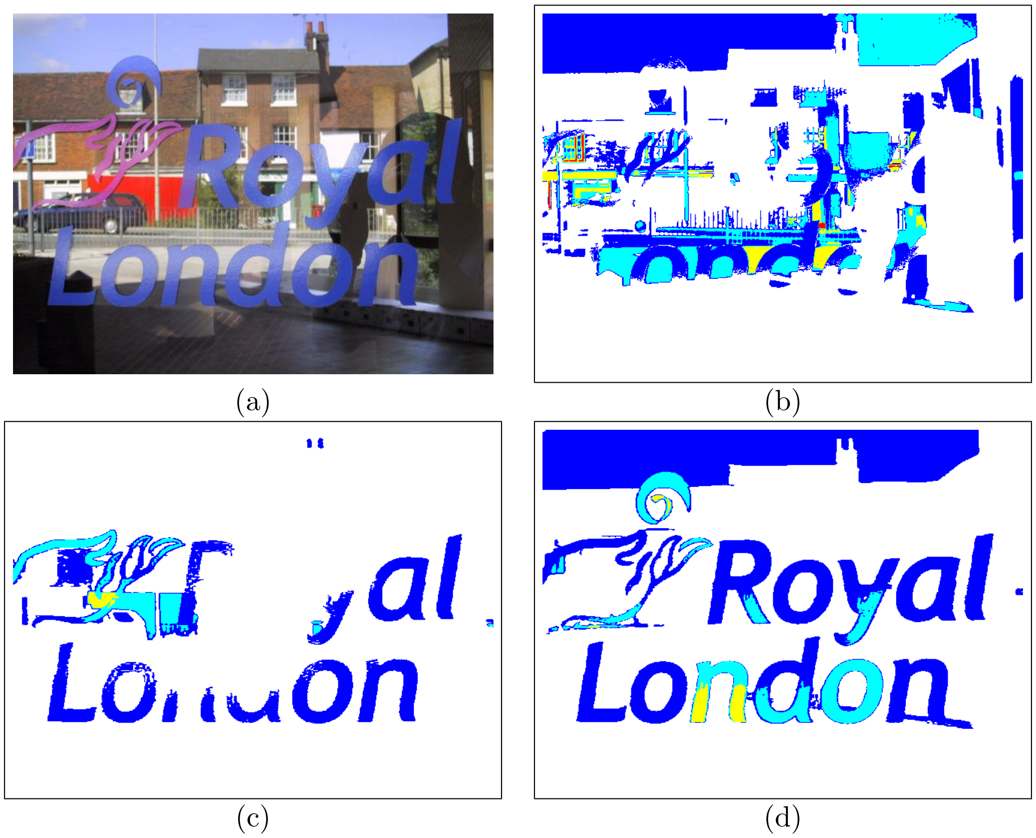


Figure 4.3: Multi-channel SER detection (a) Original image (b) Intensity Image (c) A* channel (d) B* channel. Overlapping regions are represented by different colors.

vertical stripes S_v, where $v \in \{0, \cdots, \tilde{v}-1\}$. These stripes are disjoint *i.e.* $S_i \cap S_j = \phi$ for all distinct $i, j \in \{0, \cdots, \tilde{v}-1\}$ and cover the entire component $\bigcup_{v=0}^{\tilde{v}-1} S_v = \widehat{C}$. Thereafter the number of disjoint fragments within each stripe is computed and stored at the corresponding index of a $\tilde{v}$ dimensional vector F_v. The $L^1 - norm$ of the first order difference of elements in F_v (Eq. 4.3) gives the Vertical Stripe Scan feature or VSS.

$$VSS = \sum_{n=0}^{\tilde{v}-2} |F_v(n) - F_v(n+1)| \tag{4.3}$$

For text components, the numbers in the sequence F_v vary in a gradual and predictable manner as depicted in Fig. 4.4. The fragments within the stripes are highlighted using yellow circles. The text has a low score for the VSS feature. On the contrary, the variation is random and unpredictable for the non-text components as shown in Fig. 4.5. That randomness yields a high score for the VSS feature. The steps are repeated for $\tilde{h}$ horizontal stripes S_h, where $h \in \{0, \cdots, \tilde{h} - 1\}$ to evaluate the Horizontal Stripe Scan or HSS feature. In our experiments, both $\tilde{v}$ and $\tilde{h}$ are set to 10.

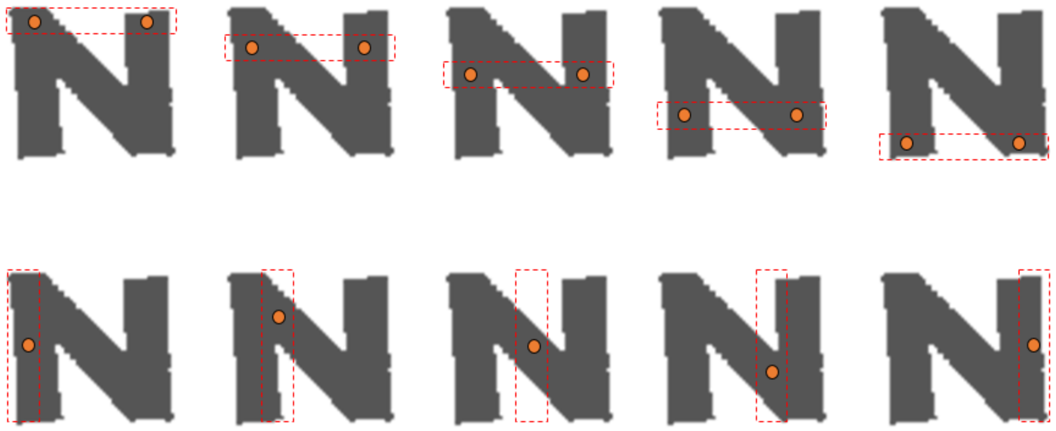


Figure 4.4: Stripe scan feature for text component using horizontal stripes (top row) and vertical stripes (bottom row). The example is for explanation purpose only. $\tilde{v} = \tilde{h} = 5$, HSS = 0 and VSS = 0

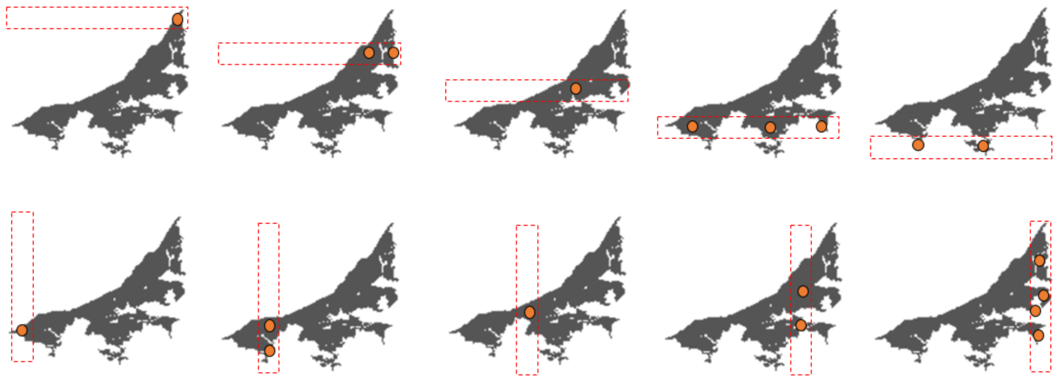


Figure 4.5: Stripe scan feature for non-text component using horizontal stripes (top row) and vertical stripes (bottom row). The example is for explanation purpose only. $\tilde{v} = \tilde{h} = 5$, HSS = 5 and VSS = 5

4.3.2.2 Scan-Line Stroke Width Transform

The stroke of a pen or a brush on a piece of paper or canvas has uniform thickness called stroke width. Text can be viewed as a set of ordered and connected strokes having constant width. Following this intuition, Epshtein *et. al.* [EOW10] proposed the popular stroke width transform method for text detection. The technique can be greatly simplified for a given binary connected component $\widehat{C}$. The proposed customized version for binary component is called the scan-line stroke width transform, abbreviated as SL-SWT.

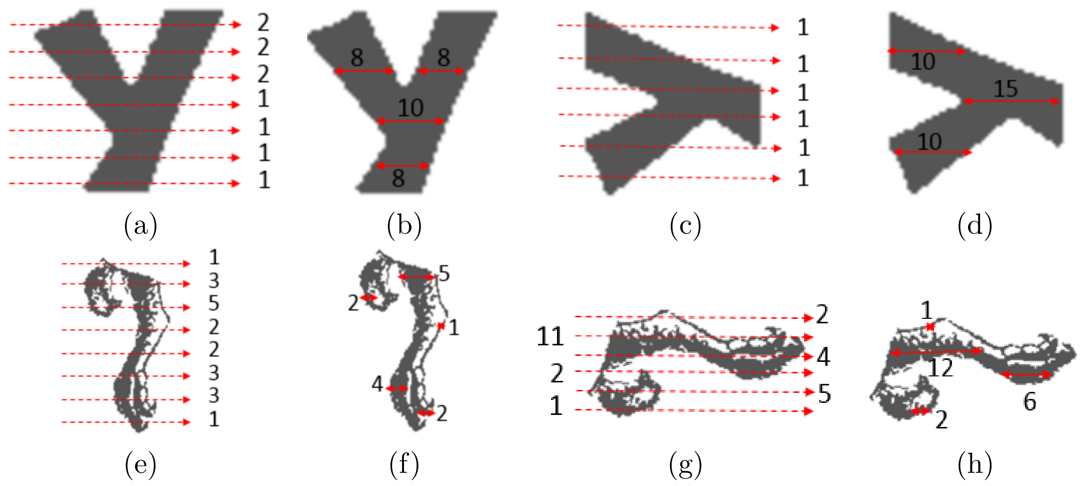


Figure 4.6: Scan-Line SWT (a), (c), (e) and (g) show the ray intersection process for the original and transposed components. The numbers at the end of the dotted arrows, represent the elements of array Θ_r or Θ_c. (b), (d), (f) and (h) depict the SL-SWT values for components and its transpose. The scan-line stroke width values for the pixels spanned by the solid red arrows are provided alongside. Only a selected scan-lines are shown for illustration purpose.

Consider component $\widehat{C}$ to be enclosed within a rectangular bounding box having $\tilde{r}$ pixel rows and $\tilde{c}$ pixel columns. The proposed method shoots a ray along every row of $\widehat{C}$ in a scan-line fashion. For each of these rays, it computes the number of intersections with the component and stores it in the corresponding index of an $\tilde{r}$ dimensional array, Θ_r. A ray entering the component and leaving it is counted as one intersection. Review Figs. 4.6 (a) and (e) for pictorial representation. The parameter VRI or Vertical Ray

Intersections is the $L^1 - norm$ of the first order difference of elements in the array, formulated mathematically in Eq. 4.4. As illustrated in the Figs. 4.6(c) and (g), Horizontal Ray Intersection or HRI parameter is similarly calculated on transpose of $\widehat{C}$.

$$\begin{aligned} VRI &= \sum_{n=0}^{n=\tilde{r}-2} |\Theta_r(n) - \Theta_r(n+1)| \\ HRI &= \sum_{m=0}^{m=\tilde{c}-2} |\Theta_c(m) - \Theta_c(m+1)| \end{aligned} \tag{4.4}$$

Text blobs have low values for ray intersection parameters, while the non-text blobs exhibit higher values. For example, VRI and HRI parameters for the text component in Fig. 4.6 (top row) are 1 and 0 respectively. On the contrary, the non-text component (bottom row) has higher values *i.e.* 10 and 25. Note that, only a subset of the scan-lines are shown in Fig. 4.6. For explanation purpose, the previously stated values of ray intersection parameters are computed on those selected rays only. These values are normalized with respect to the dimensions of the component for a fair comparison across components of varying sizes.

The ray intersection parameters VRI and HRI can be viewed as fine grained version of the stripe scan features, but both of them are equally important. Ray intersection features give wrong classification for text with noisy background. While stripe scan features might miss some false positives due to its coarseness. Taking into account all these features together helps in avoiding such mistakes. Besides, the ray intersections is used to compute more relevant features as described next.

The component $\widehat{C}$ can be a letter or word fragment as demonstrated in Fig. 4.7. We follow a conservative assumption, that a letter component cannot have aspect ratio more than unity. Based on this hypothesis, the aspect ratio gives a lower bound on the number of letters contained in the word fragment. For example, the text component with an aspect ratio of 5.0 is expected to have at least five letters. Most of the rays shot along the rows of such a component should intersect at least five times. In other words,

$median(\Theta_r)$ should be greater than or equal to five, *i.e.* the aspect ratio. If this value is less than the aspect ratio, then that component is too smooth to be text and should be discarded.

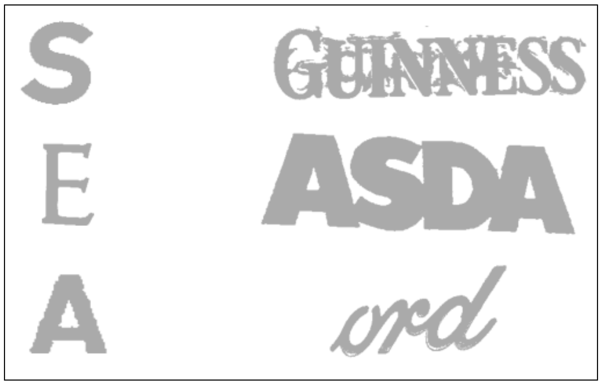
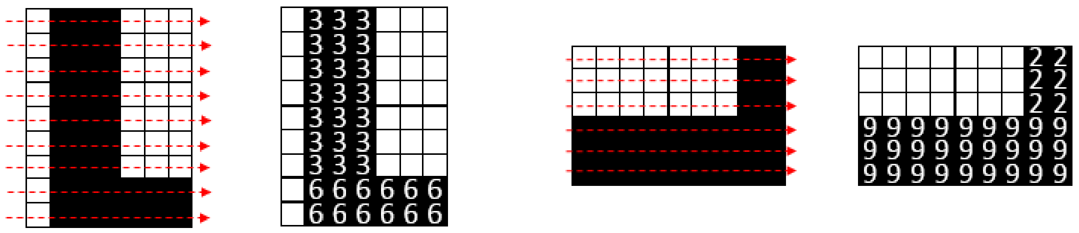


Figure 4.7: Connected components $\widehat{C}$ representing letters (left) and those representing word fragments (right)

The scan-line stroke width value matrix $\mathbf{M}$, corresponding to $\widehat{C}$ is generated as follows. Suppose the component is tightly enclosed within a rectangle having $\tilde{r}$ rows and $\tilde{c}$ columns and $\mathbf{M}_r$, $\mathbf{M}_c$ be two matrices with all elements initialized to $+\infty$. To compute matrix $\mathbf{M}_r$, rays are shot along the rows of $\widehat{C}$. The number of pixels a ray encounters during an intersection with $\widehat{C}$ is determined. As illustrated in Fig. 4.8, all the corresponding elements in the matrix $\mathbf{M}_r$ are replaced by the number of intersecting pixels. The process is repeated for $\widehat{C}^T$ and the values are stored in $\mathbf{M}_c$. Superscript T denotes the transpose operation. The SL-SWT matrix for the component $\widehat{C}$ is then the element-wise minimum of those two matrices, *i.e.* $\mathbf{M} = min(\mathbf{M}_r, \mathbf{M}_c^T)$. The most frequently occurring value in $\mathbf{M}$ is the stroke width value of $\widehat{C}$. For instance, the value of stroke width for the letter "L" in Fig. 4.8 is 3.

The components having consistent stroke width values resemble text, while the objects having non-uniform widths are false positives. The blobs having relatively small stroke width value, compared to the dimensions of the circumscribing rectangle, are also discarded. Figs. 4.6 (b), (d), (f) and (h) are some examples. Apart from geometric filtering, the matrix $\mathbf{M}$ is extensively used in Sec. 4.3.3 for grouping the components.



(a) Matrix $\mathbf{M}_r$ (b) Matrix $\mathbf{M}_c$

Figure 4.8: Stroke width values for connected component representing letter L. The squares with no numerical value are equal to $+\infty$. (a) Rays shot along the rows and the $\mathbf{M}_r$ matrix (b) Rays shot along the rows of $\widehat{C}^T$ and the $\mathbf{M}_c$ matrix

4.3.2.3 Plausibility Constraints

The stable extremal regions that do not satisfy the following plausibility constraints are cleaned up.

Low Occupancy Ratio: Text has high compactness, as it is closely packed in the circumscribing bounding box. Consider a text component enclosed in a rectangular bounding box and also circumscribed by a convex contour. The ratio of area enclosed within the contour to the area of the bounding box is called the occupancy ratio. This ratio is close to unity for text. All the components with this ratio less than half can be safely removed.

Long Vertical Blobs: The text is usually laid out horizontally. Hence, the long vertical components that mostly resemble poles, iron bars, printed designs etc. can be removed.

Massive Blobs: It is highly unlikely for the natural scene images to have text covering more than 95% of a certain image dimension. These extremely large components are discarded.

Tiny Blobs: The blobs with $\tilde{r} \leq 10$ pixels can be ignored since they are practically imperceptible in a normal resolution image.

For all the features described above, decision stumps classifiers are chosen for text/non-text categorization. Thresholds are learned heuristically from the examples in the training dataset. These thresholds are generous enough to retain most of the text components, even if that leads to reduction in false positive removal rate. Collectively, these features overcome the weakness of generous thresholds. Fig. 4.9 justifies this hypothesis. Most of the false positives in Fig. 4.9 (b) are removed after geometric filtering due to multitude of features, while all the true positives are retained thanks to the generous thresholds.

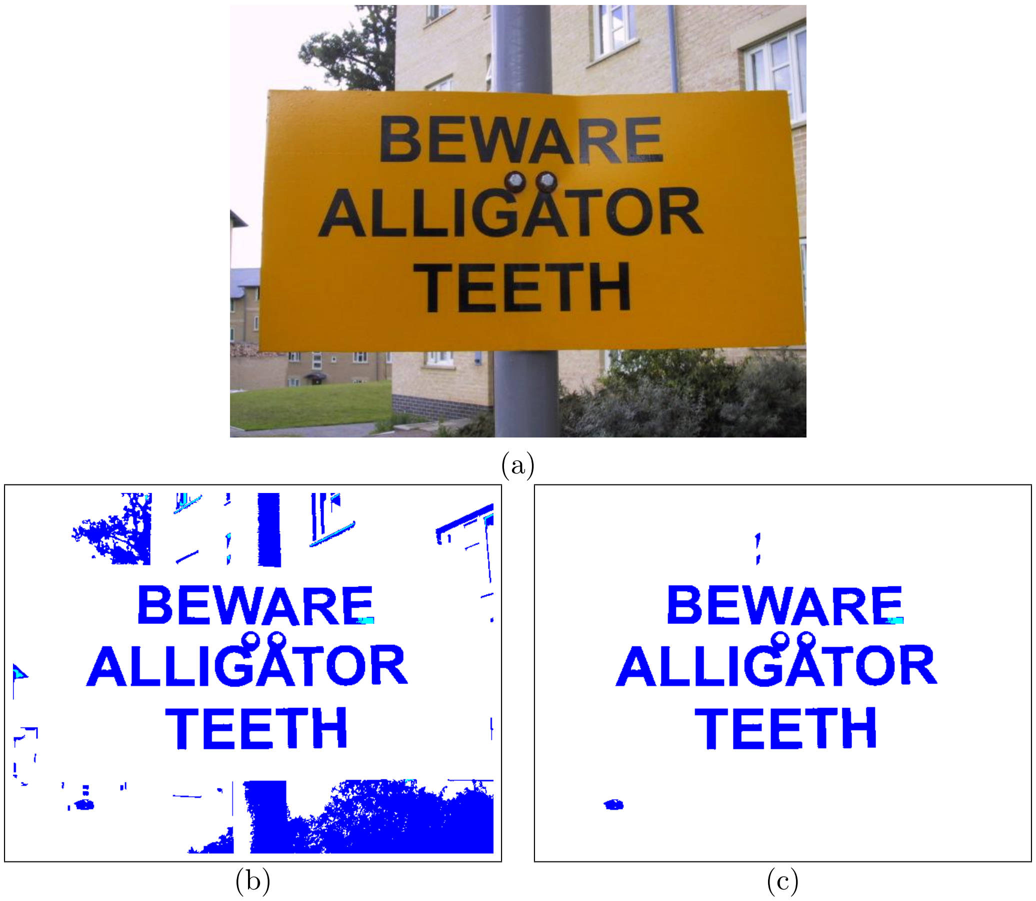


Figure 4.9: Geometric filtering (a) Sample image (b) Detected stable extremal regions (c) Output after geometric filtering

4.3.3 Context-Based Text Grouping

The previous stage eliminates a considerable number of non-text regions without any extensive processing. Then the context-based text grouping stage clusters the remaining components into words. First, the components are linked together into text lines and then those text lines are broken down into words. As a by-product, this stage can also prune some text-like false positives, which are otherwise difficult to remove. It can also recover some letters or words which were wrongfully rejected by the earlier stages. These wrongfully rejected components are synonymously referred to as missing components. These grouping, pruning and self-correcting mechanisms are explained here in detail.

Text is laid out horizontally, as if, it is resting on an invisible baseline. Moreover, the letters or words belonging to the same line of text have similar spatial properties. Especially, their heights, stroke widths and color palettes are similar. The components that are spatially consistent to their horizontally aligned neighbors, can be paired. Most probably these component pairs represent text. On the contrary, non-text components pop up randomly in the image. The probability of finding a similar blob in the neighborhood of a non-text region is considerably low. Even if there happens to be a matching blob, it might not be aligned properly. In this way, these spatial and alignment cues serve a dual-purpose. They help to pair the related text regions and detect the isolated components as non-text.

4.3.3.1 Agglomerative Clustering

Let N_l and N_r be the left and the right neighborhood of a component $\widehat{C}$. Suppose h represents the height of that component, while (c_x, c_y) are the coordinates of the centroid in the image space and Δ is the extent of the region. Then N_l and N_r can be formulated as,

$$\begin{aligned} N_l &= \{\forall (x,y) : x \in (c_x - \Delta, c_x) \quad \text{and} \quad y \in (c_y - h/2, c_y + h/2)\} \\ N_r &= \{\forall (x,y) : x \in (c_x, c_x + \Delta) \quad \text{and} \quad y \in (c_y - h/2, c_y + h/2)\} \end{aligned} \tag{4.5}$$

In the above equation, (x, y) denotes a 2D point in the image space. The closest blob in each of these neighborhoods N_l and N_r, that matches the component $\widehat{C}$ is linked to it. In this manner, $\widehat{C}$ can have a maximum of two links; one on the left and other on the right hand side. Please refer to Fig. 4.10(a) for illustration. The blue box is the left neighborhood of $\mathbf{p}$ and the green box is the right neighborhood. The matching process is described next.

4.3.3.2 Matching Criteria

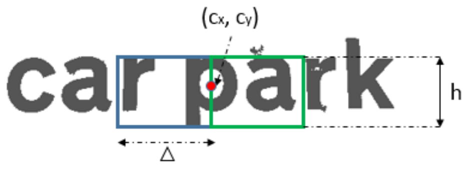
The aforementioned spatial features of the component $\widehat{C}$, namely, height h, stroke width s, color c and horizontal baseline b, where $b = c_y + h/2$, can be stored in the form of a vector $\overline{S} = \{b, c, h, s\}$. If these vectors are viewed as points in a higher dimensional space, then components that match with $\widehat{C}$ would lie in a hyper-rectangle centered at point $\overline{S}$. The dimensions of the hyper-rectangle along b and h axes are proportional to height of the component. This allows invariance to the size of the component. In c and s directions, the dimensions are fixed. This constraints the color and stroke width of the matching components to lie within a specific range. Fig. 4.10(b) shows some matching components in the neighborhood. The blob $\mathbf{p}$ links to $\mathbf{r}$ and $\mathbf{a}$. In turn, component $\mathbf{r}$ matches with $\mathbf{p}$ and $\mathbf{a}$, while $\mathbf{a}$ pairs with $\mathbf{p}$ and $\mathbf{r}$.

4.3.3.3 Merging Pairs Into Lines

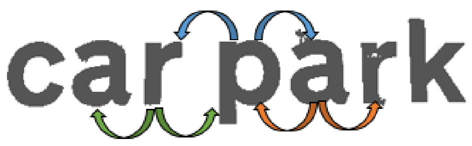
Consider a graph, $G = (V, E)$ with every component in the image represented by a vertex and the links corresponding to the edges in G. In this representation, the components belonging to the same line of text have edges among them. While the components belonging to different lines are not connected. Each sub-graph in such a disconnected graph, represents a text line. A sub-graph is connected section in the graph that has no edges coming out or going into it. Depth-first or breadth-first search can separate out those sub-graphs to get text lines. The sub-graphs containing less than three nodes are discarded. The matching pairs in Fig. 4.10(b) are linked together to form the text line "car park" (Fig. 4.10(c)).

4.3.3.4 Decimation Of Lines Into Words

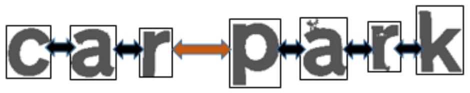
Every component in the text line has two links; one to the component in the left neighborhood and other to the right. Exceptions being the first and the last components in the line. As far as components within a word are concerned, they can be categorized into three types, namely, (i) starting component (ii) intermediate components and (iii) ending component. Consider L_{left} and L_{right} to be the lengths of the left and the right links respectively. The disparity of these lengths is large for the starting ($L_{left} \gg L_{right}$) and the ending components ($L_{left} \ll L_{right}$) of a word. On the contrary, the ratio of these lengths is close to unity for intermediate components. Put differently, intra-word gaps are smaller compared to inter-word gaps. This property helps to segment out the words from those text lines. In Fig. 4.10(c), the brown link that describes the inter-word gap has to be broken down. To mitigate the effect of missing components, the output of the stable extremal regions stage is checked to verify that the inter-word gap is indeed empty.



(a)



(b)



(c)

Figure 4.10: Context-based text grouping (a)Left and right neighborhood (b)Links between adjacent and similar components (c)Sub-graph with vertexes and internal edges that together represent a text line. The orange link has to be broken to form individual words.

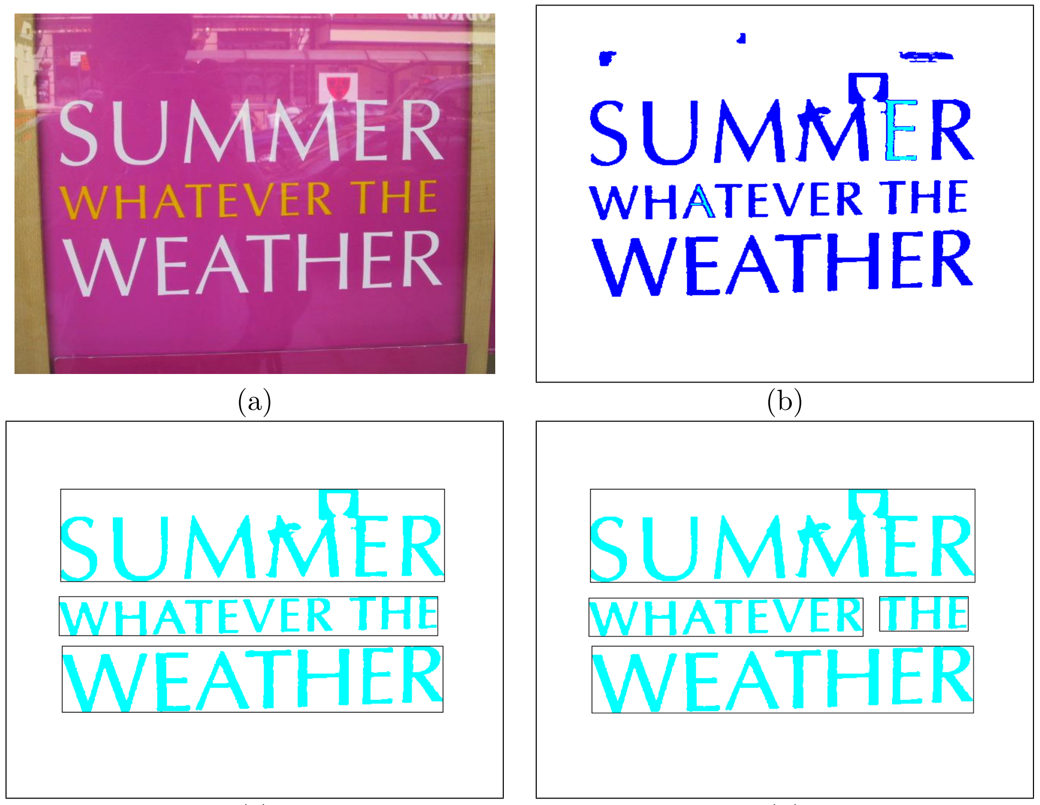


Figure 4.11: Context-based text grouping (a) Raw image (b) Blobs retained after geometric filtering (c) Bounding boxes generated after merging pairs into lines (d) Decimation of lines into words

Fig. 4.11 demonstrates the grouping procedure for a sample image. It must be noted that the isolated false positives that survived after geometric filtering in Fig. 4.11 (b) are all removed in the grouping stage, *i.e.* Fig. 4.11 (c). After drawing bounding boxes around the tentative words in the image, the next stage inspects their contents to classify the boxes as text or non-text. The text boxes are retained, while the others are removed.

4.3.4 Ensemble Classifier and Fusion

The inherent properties of text are markedly different from the non-text regions. The features which exploit these properties are better suited for region classification. Such features include, Gradient Orientation Histogram, Scale Invariant Feature Transform

Codewords Histogram etc. Refer to Sec. 2.2.3 and 2.2.4 for details. Taking these things into account, we developed novel gradient profile features to train an ensemble of decision tree classifiers using the Adaboost framework. After region classification, the bounding boxes from different views, namely, $\mathbf{I}$, $\mathbf{I_{inv}}$, $\mathbf{A}^*$, $\mathbf{A}^*_{\mathbf{inv}}$, $\mathbf{B}^*$ and $\mathbf{B}^*_{\mathbf{inv}}$ were fused together to get the consolidated list of bounding boxes.

4.3.4.1 Gradient Profile Features

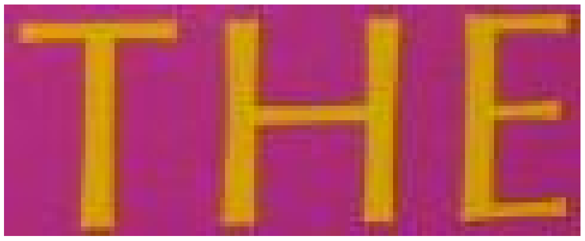
The text regions have sharp edges with consistent blank spaces between consecutive letters or words. On the contrary, non-text regions seldom have sharp edges and certainly, there is no consistent spacing throughout. Gradient profiles of these regions, help to exploit these peculiar properties of text for classification purpose.

The regions within the bounding boxes are resize to fixed size of 96×32, followed by gradient computations along the X and Y direction, *i.e.* G_x and G_y respectively. The mean vector along the rows of the matrix, G_x is termed as the horizontal gradient profile of G_x. Similarly, the mean vector along the columns of G_x is the vertical gradient profile of G_x. Then the gradient profile feature vector $\overline{GP}$ is the concatenation of horizontal and the vertical gradient profile vectors of G_x and G_y as shown in Eq. 4.6. The gradient profile features for a sample text and non-text region are shown in Figs. 4.12 and 4.13 respectively. The peaks in the profiles of text regions are interleaved with valleys. The spacing between those valleys is fairly consistent. On the contrary, the profiles for non-text regions do not usually have such consistent spacing.

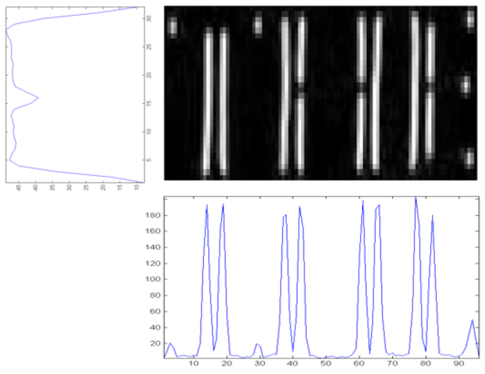
$$\overline{GP} = \{(\sum_i G_x(i,j))^T, \sum_j G_x(i,j), (\sum_i G_y(i,j))^T, \sum_j G_y(i,j)\} \tag{4.6}$$

4.3.4.2 Ensemble of Decision Tree Classifiers

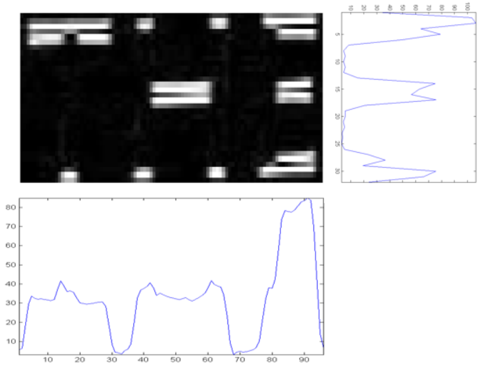
Decision tree classifiers are used as weak classifiers in the Adaboost framework. An ensemble of these weak classifiers is trained on the normalized gradient profile features.



(a)



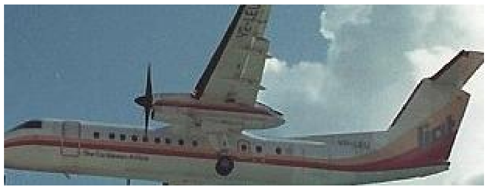
(b)



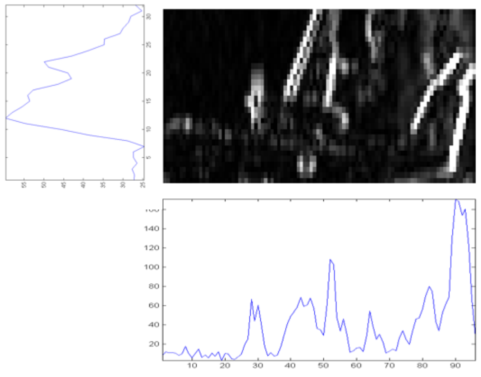
(c)

Figure 4.12: The gradient profile features (a) Sample text image (b) G_x and mean vectors along the rows and the columns (c) G_y and mean vectors along rows and columns

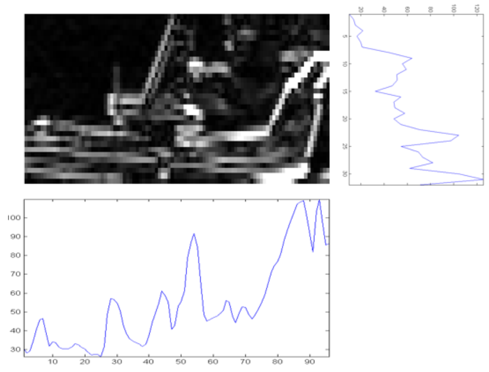
Training dataset for the ensemble classifier consists of 1391 positive and 1278 negative samples taken from the ICDAR [LPS+03,Luc05,SSD11,KSU+13] training dataset. Some images from the dataset are displayed in Fig. 4.14. The positive samples are cropped from the text regions using semi-automatic techniques, while the negative samples are randomly picked. The positive and negative samples are distributed into five equal sized sets. During cross validation, one set is used for training while the remaining four constitute the test set. The training data is purposely chosen to be smaller compared to the test data in order to simulate real word scenario. After five iterations, the average accuracy is 90.46%. The input to the ensemble classifier is the list of bounding boxes accrued after the grouping stage and the output is a probability value $\in [0, 1]$. The cutoff is the midway point and the rectangles with probability less than the threshold are filtered out.



(a)

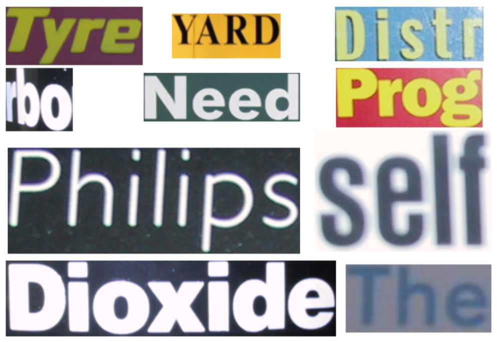


(b)



(c)

Figure 4.13: The gradient profile features (a) Sample non-text image (b) G_x and mean vectors along the rows and the columns (c) G_y and mean vectors along rows and columns



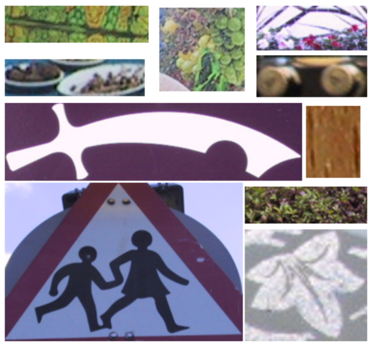


Figure 4.14: Positive samples for training ensemble classifier (left) and negative samples (right)

4.3.4.3 View and Channel Fusion

The whole localization process is repeated for both the views in each channel, namely, $\mathbf{I}$, $\mathbf{I_{inv}}$, $\mathbf{A}^*$, $\mathbf{A}^*_{\mathbf{inv}}$, $\mathbf{B}^*$ and $\mathbf{B}^*_{\mathbf{inv}}$; to obtain six different lists of bounding boxes. The objects detected in the complementary views, such as $\mathbf{I}$ and $\mathbf{I_{inv}}$, are mutually exclusive *i.e.* $\mathbf{I} \cap \mathbf{I_{inv}} = \phi$. So the corresponding lists from these complementary views are appended with minimal processing. Fig. 4.15 illustrates the concept of view fusion. In this case, no processing is required to aggregate the bounding boxes from $\mathbf{I}$ and $\mathbf{I_{inv}}$.

View fusion leaves us with three distinct lists, one for each channel. The bounding boxes from all these lists are categorized as overlapping or non-overlapping, based on the amount of intersection of those bounding boxes with the boxes from other channels. The rectangles from the non-overlapping pool are directly inducted into the final list of estimated rectangles $\mathbf{R_e}$. The overlapping boxes are either merged together or pruned. The decision to merge or prune an overlapping box is contingent on the amount of overlap, aspect ratio and number of components enclosed. This decision is delegated to an objective function that weighs in all those factors before making a decision. These merged rectangles are then appended to the list of estimated rectangles $\mathbf{R_e}$. Refer to Fig. 4.16 for an example of channel fusion. The bounding boxes from the channels $\mathbf{I}$, $\mathbf{A}^*$ and $\mathbf{B}^*$ are merged together to get a single rectangle enclosing the whole word.

4.4 Experimental Results

The ICDAR database is the most popular for text localization in natural scene images. This dataset was used in ICDAR 2003 [LPS+03] and ICDAR 2005 [Luc05] text localization competitions. The competing algorithms are evaluated on the basis of three parameters, namely, *precision*, *recall* and *f-parameter*. The list of ground truth rectangles is denoted by R_t and the competing algorithm is expected to output a similar list of estimated rectangles R_e. Consider the match $m_p(r, r')$ as the area of intersection of r and r' divided by the area of minimum rectangle circumscribing both these rectangles. Then the value $m(r_e, R_t) = max_{r' \in R_t} m_p(r, r')$ gives the matching score for $r_e \in R_e$ with the best fit rectangle in the set R_t. Based on these scores, the precision and the recall parameters are computed as shown in Eq. 4.7. With α set to 0.5 in Eq. 4.8, the *f-parameter* is a non-linear combination of *precision* and *recall.* A detailed description of the evaluation parameters can be found in [LPS+03].

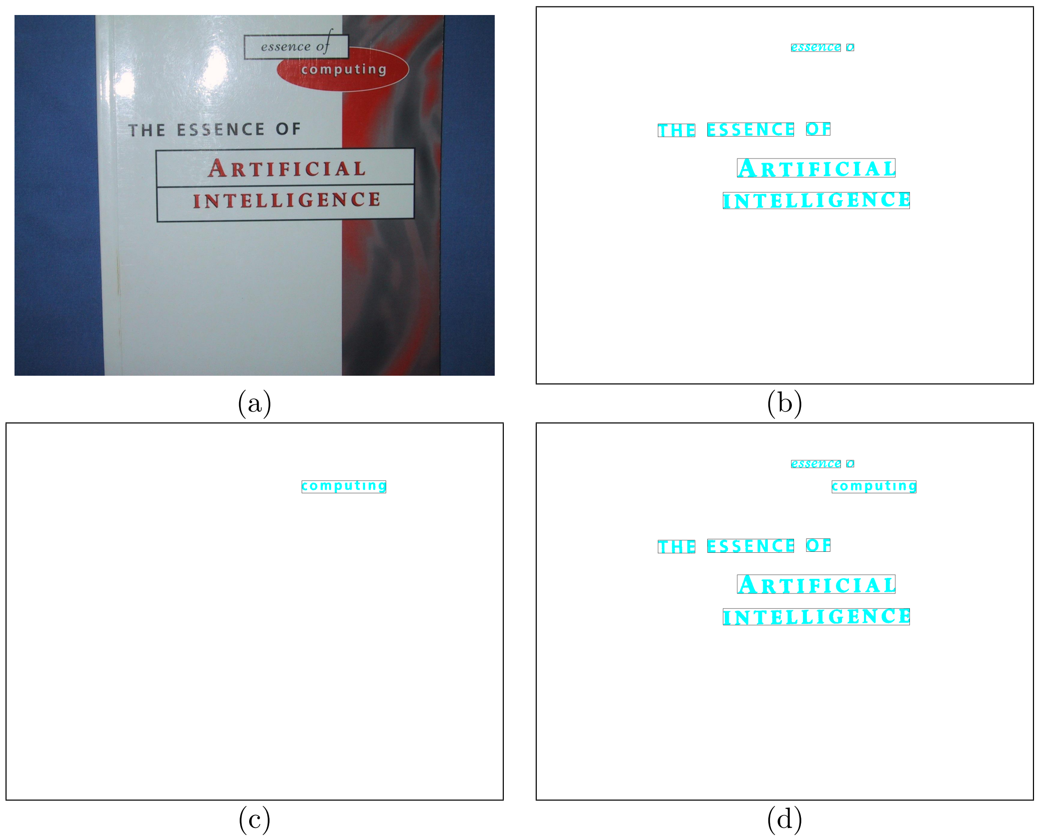


Figure 4.15: View fusion (a) Original image (b) Bounding boxes for **I** (c) Bounding boxes for $\mathbf{I_{inv}}$ (d) Fusion of **I** and $\mathbf{I_{inv}}$

$$precision = \frac{\sum_{r_e \in R_e} m(r_e, R_t)}{|R_e|}, \qquad recall = \frac{\sum_{r_t \in R_t} m(r_t, R_e)}{|R_t|}, \tag{4.7}$$

$$f = \frac{1}{\frac{\alpha}{precision} + \frac{1-\alpha}{recall}}, \tag{4.8}$$

Lucas *et. al.* [Luc05] reported the performance of the few top-ranked text localization algorithms from those competitions. The *precision*, *recall* and *f-parameter* scores were separately computed on individual images and then averaged over all the images in the dataset. Several other algorithms proposed thereafter propose a different method for calculating the overall performance score. They compute a single precision, recall and f-parameter value on the entire dataset. The results on ICDAR 2005 and 2011 datasets

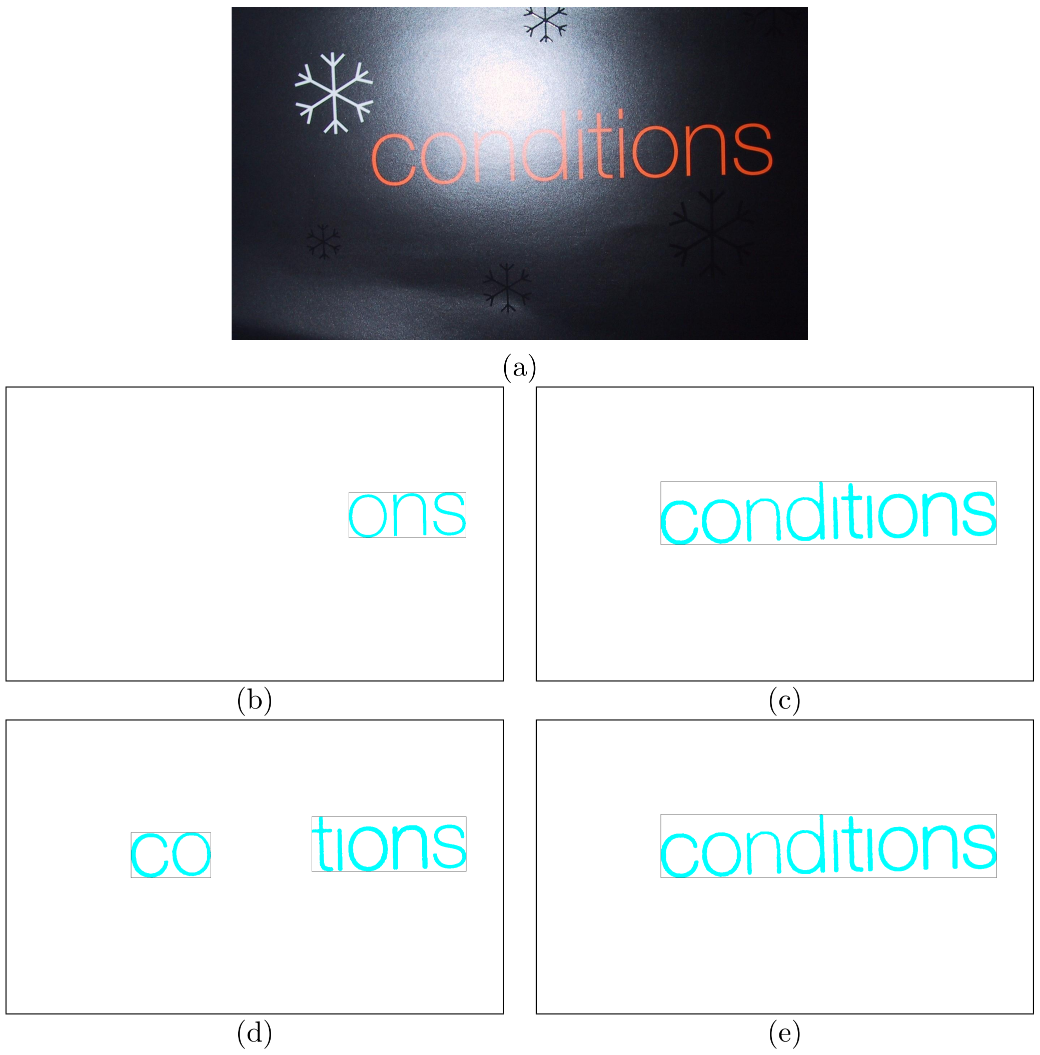


Figure 4.16: Channel fusion (a) Sample image (b) Bounding boxes for channel **I** (c) Bounding boxes for channel $\mathbf{A}^*$ (d) Bounding boxes for channel $\mathbf{B}^*$ (e) Fusion of **I**, $\mathbf{A}^*$ and $\mathbf{B}^*$

are tabulated in Tables 4.1, 4.2, 4.3 and 4.4. As evident from these tables, our proposed scheme is among the the state-of-the-art algorithms.

To demonstrate the effectiveness of individual stages of the proposed framework, a comparative analysis on a subset of the training data is performed. Table 4.5 lists the number of objects that are retained after every stage. These objects can be letters, words or false positives. The table also reports the percentage of components removed by the current stage, when compared to the output of the previous stage. For this analysis, the

Algorithms	Precision	Recall	f-parameter
Proposed Method	**0.70**	**0.62**	**0.64**
Neumann and Matas [NM11]	0.65	0.64	0.63
Hinnerk Becker	0.62	0.67	0.62
Alex Chen	0.60	0.60	0.58
Ashida	0.55	0.46	0.50
HWDavid	0.44	0.46	0.45
Wolf	0.30	0.44	0.35
Quiang Zhu	0.33	0.40	0.33

Table 4.1: Performance comparison of various text localization algorithms with the evaluation method from ICDAR 2005 competition

Algorithms	Precision	Recall	f-parameter
Proposed Method	**0.76**	**0.62**	**0.68**
Neumann and Matas [NM11]	0.72	0.62	0.67
Chen *et. al.* [CTS+11]	0.73	0.60	0.66
Epshtein *et. al.* [EOW10]	0.72	0.60	0.66
Yi and Tran [YT13]	0.71	0.62	0.63

Table 4.2: Performance comparison of various text localization algorithms with the new evaluation method on ICDAR 2005 dataset

Algorithms	Precision	Recall	f-parameter
Neumann and Matas [NM13]	0.79	0.66	0.72
Proposed Method	**0.76**	**0.61**	**0.68**
Yi and Tran [YT13]	0.76	0.68	0.67
Gonzalez *et. al.* [EOW10]	0.73	0.56	0.63
Yi and Tran [YT11]	0.67	0.58	0.62
Neumann and Matas [NM11]	0.69	0.53	0.60

Table 4.3: Performance comparison of various text localization algorithms on ICDAR 2011 dataset using [WJ06] evaluation criteria

Algorithms	Precision	Recall	f-parameter
Huang *et. al.* [HLYW13]	0.82	0.75	0.73
Proposed Method	**0.81**	**0.64**	**0.72**

Table 4.4: Performance comparison of various text localization algorithms on ICDAR 2011 dataset using Yao's evaluation method [YBL+12]

dataset is carefully chosen such that almost all of the true positives survive till the end. So the percentage of components removed is a reliable indicator of the false positive removal rate of that particular stage. The geometric filtering and context-based text grouping stages show a high removal percentage. On the contrary, region classifier stage has a low percentage because there are hardly any false positives left by the time this stage is reached.

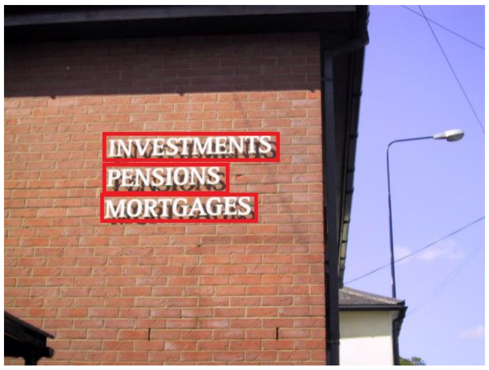
Stages	**Components Retained**	**Components Removed**
Stable Extremal Regions	1685	Not Applicable
Geometric Filtering	974	42.20%
Context-Based Text Grouping	387	60.27%
Ensemble Classifier and Fusion	374	3.36%

Table 4.5: Statistical performance analysis of each stage of the framework

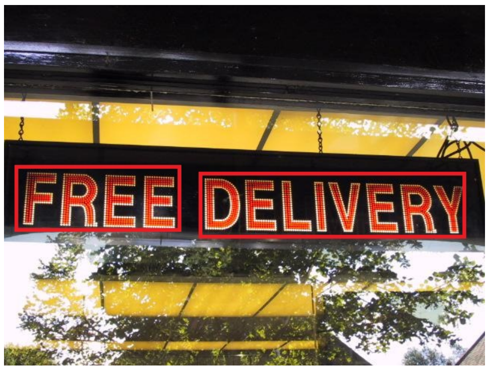
4.5 Error Analysis

The proposed text localization pipeline is able to detect words of different fonts, colors and sizes, even in the presence of reflection and background clutter. For instance, in Fig. 4.17(a) the textured brick pattern can be potentially misinterpreted as text due to its structural similarity with low resolution small font words. Our algorithm is able to correctly discarded those patterns with the help of geometric filtering. Fig. 4.17(b) shows that the reflection of vegetation on the transparent glass window can be effectively dealt with. The context-based text grouping stage is able to remove this vegetation as the spatial distribution of those blobs is drastically different from that of the text. Similar is the case with the background clutter in Fig. 4.17(c). The window patterns in Fig. 4.17(d) pass the grouping stage but they are cleaned up by the ensemble classifier.

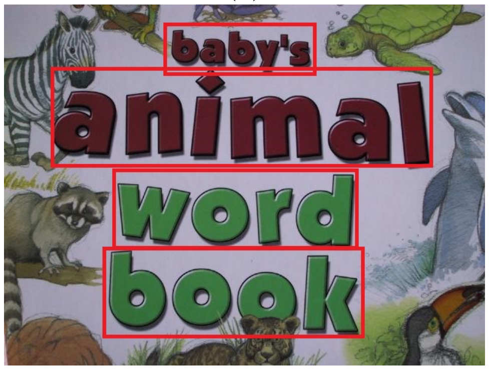
The algorithm is reliable, but under adverse circumstances it shows its weaknesses. For example, in Fig. 4.18 the Stable Extremal Region (SER) operator fails to detect some text in the image. The intensity values of the pixels within the text are similar to the values of the pixels in the background. Due to the background blending, those text



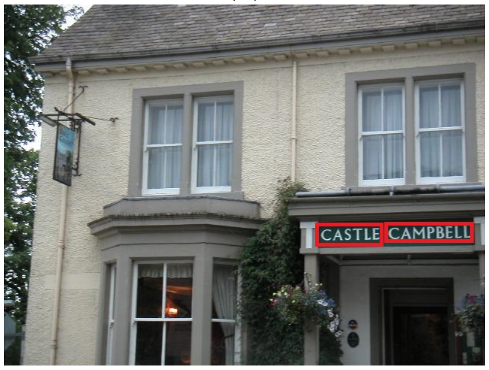
(a)



(b)



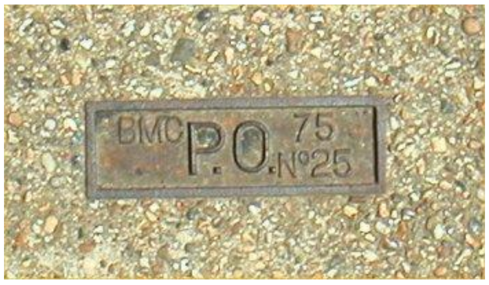
(c)



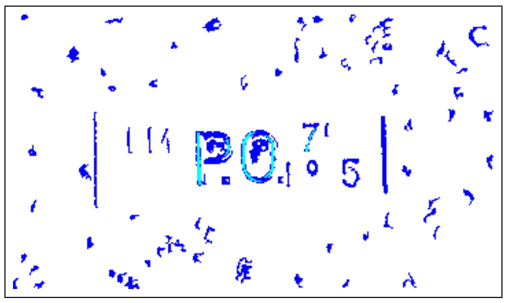
(d)

Figure 4.17: Correct detections

regions are not detected in the first stage itself. The regions lost at such an early stage in the algorithm cannot be recovered thereafter. Incorporating edge information in the SER operator functionality might alleviate the problem.



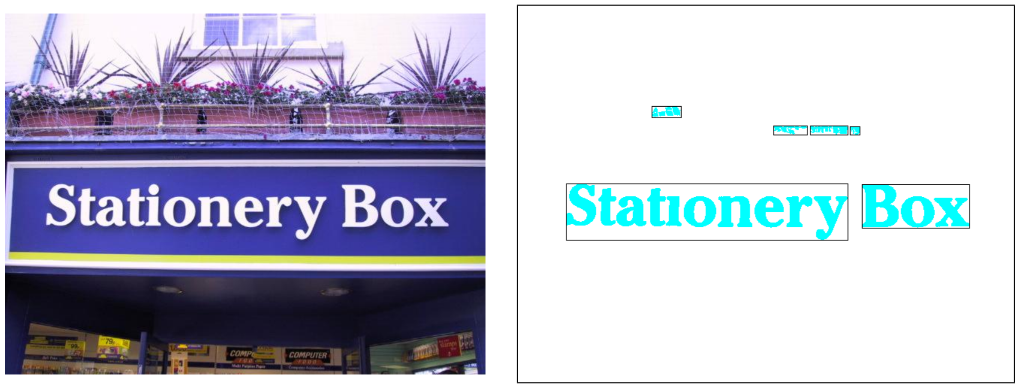
(a)



(b)

Figure 4.18: SER detection error (a) Sample image (b) Text blobs detected

In some cases, the repeated background patterns might be misinterpreted as text. As it can be seen in Fig. 4.19(b), the small non-text blobs at the top are mistaken as small font text. These blobs are passed on as text by the geometric filtering, context-based text grouping and ensemble classifier stages. A technique that can take care of such repeated patterns will be helpful in getting rid of these false positives.

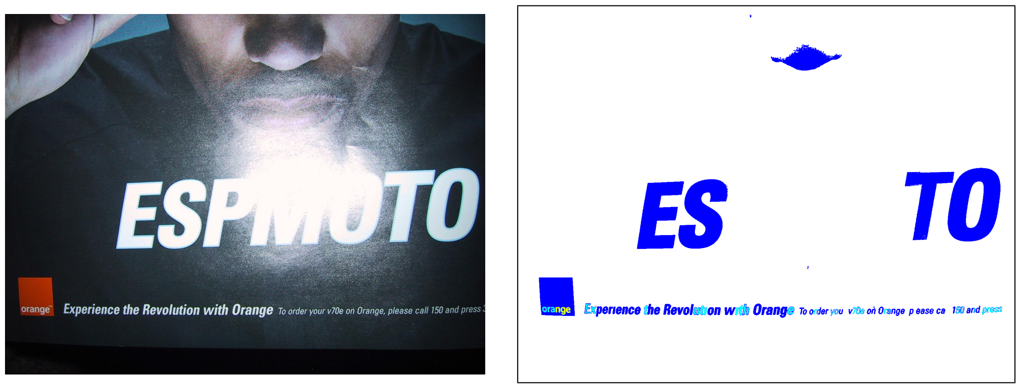


(a) (b)

Figure 4.19: Repeated patterns (a) Error image (b) Repeated patterns detected as words

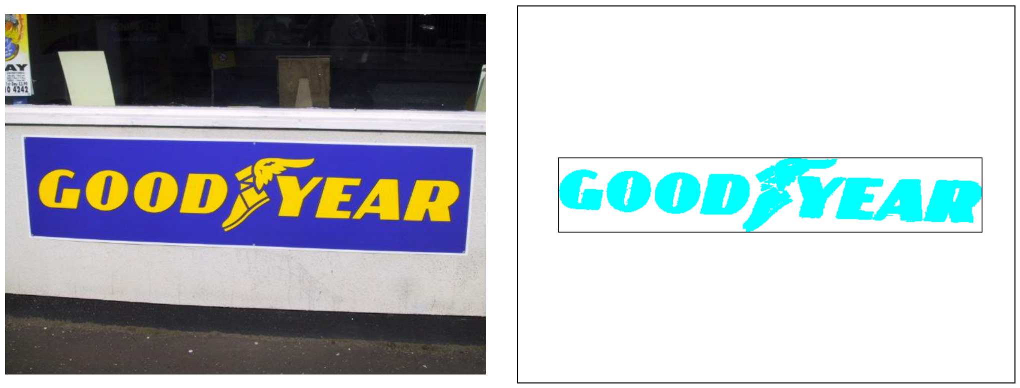
The text is difficult to recover in the images with partial occlusion. Fig. 4.20 has a strong camera flash that partially wiped out the underlying text. The SER operator coalesces the text and flash light together to form a big heterogeneous blob. The geometric filtering treats the whole blob as one entity and discards it as non-text. To avoid this problem, the camera flash should be removed in the pre-processing step.

If the words are not separated out correctly from the text lines, then the performance score can get affected. In Fig. 4.21, the words "Good" and "Year" are detected correctly, but the icon in between the two words causes them to be wrongly classified as a single word. The root cause of the problem is that, the multi-stage incremental region classification technique was unable to remove the icon. But removing these blobs is inherently difficult for any region classification technique. If text recognition is incorporated in the framework then probably we might be able to correct such errors as well, using a feedback mechanism.



(a) (b)

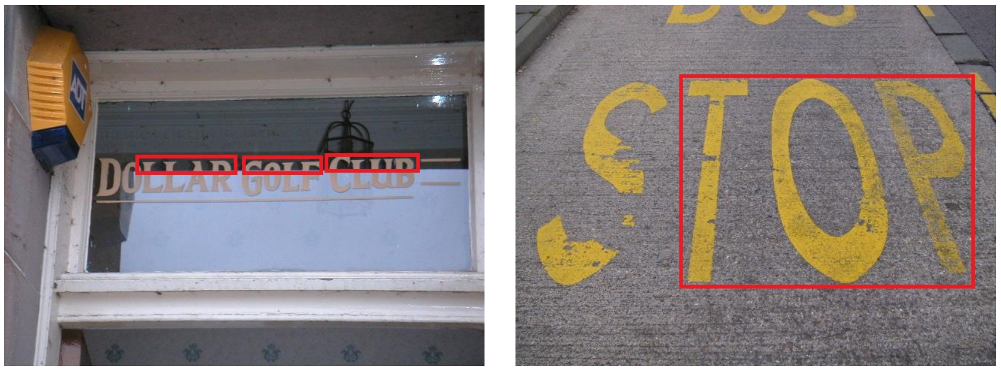
Figure 4.20: Partial occlusions (a) Flash light partially occludes the text (b) Text is wrongly discarded along with the flash light



(a) (b)

Figure 4.21: Grouping error (a) Sample image (b) Words wrongly grouped together

The other potential problems are shadow, reflection and orientation of the words. The characters "ADT" that are oriented at a specific angle to the camera are not detected in Fig. 4.22 (a). Shadow and reflection prevents the SER operator from detecting the words, "Dollar", "Golf" and "Club". Partial detections due to occlusions or incomplete detections may lead to missing text. In the example of Fig. 4.22(b), the letter "S" is broken down into two connected components and fails to form a coherent group with "TOP". The grouping stage considers it as an isolated blob and removes it.



(a) (b)

Figure 4.22: Incorrect detections (a) Effect of reflection and shadow (b) Broken text

4.6 Conclusion

In this research, we proposed a novel text localization framework based on multi-stage incremental region classification technique. The probable text regions detected by multi-channel stable extremal region detector are passed on to the geometric filters. Text context-based grouping stage combines the related text regions into words. A significant amount of false positives are removed in the process. We also developed a decision tree based ensemble classifier that categorizes the detected regions into text or non-text using the gradient profile feature information. Then the bounding boxes from different channels and views are merged together to get a consolidated list of estimated rectangles.

A region adaptive step selection can improve the accuracy of the stable extremal region detector. The properties of the area being processed can be used to determine the appropriate step-size for the detector. This would assist the subsequent stages by reducing the number of false alarms. Next step would be to recognize words in the bounding boxes with an integrated optical character recognition system. This OCR system can also be used as a feedback mechanism to improve the localization results further. No algorithm can detect text that is occluded by flash light illumination or the foreground objects. So we plan to incorporate an intelligent dictionary based system which might be able to reconstruct the whole word with this partial information at hand.

Chapter 5

Conclusion and Future Work

The conclusions of our research and future research prospects are summarized in the sections ahead.

5.1 Human Action Classification

A technique for automated mocap data classification was presented in this work. The TSVQ method was adopted to approximate static human poses with codewords while a dynamic human motion was represented by a sequence of codewords. Fusion approaches were proposed to classify mocap data into different categories. We tested the proposed algorithms on the CMU mocap database using the 5-fold cross validation procedure and obtained a correct classification rate of 99.6%.

The proposed algorithm can be extended to various tasks required by mocap database management such as segmentation, indexing and retrieval, without much effort. We only sketch the basic ideas below.

- **Segmentation**

  If a complex motion clip contains multiple basic motions, then the sequence of these basic motions will be present in the sequence of that complex motion. By using string matching, we can find out the location of sequences of all basic motions in the complex motion. This corresponds to the segmentation of a complex action sequence into multiple basic action sequences.

- **Indexing**

  After running the classification algorithm, unknown motion clips can be classified into one of the known categories and indexed accordingly. Consider a mixed motion that has running, jumping and bending actions one after the other. Using the string matching and the database of sequences, the location of all these basic motions in that mixed motion can be determined and these basic actions can be indexed accordingly.

- **Retrieval**

  The retrieval problem can be solved using suffix array technique. To be more specific, for a given test motion, we would like to extract motion clips or parts of motion clips from the database that are similar to the query. To perform this retrieval task, we convert the test motion into a sequence and use the string matching algorithm to search the motion clips or portions of the motion clips that have the same sequence. Those clips can then be retrieved.

There are several algorithms for extracting human skeleton from depth images produced by Microsoft Kinect [SSK$^+$13, AD13]. The data generated by these algorithms are similar to the mocap data [CK13]. It is interesting to explore whether the proposed methodology can be extended to the Kinect sensor data as well. High quality 3D rendering can be achieved by blending stored mocap data with the pose information from the Kinect sensor inputs.

5.2 Text Localization

In this research, we proposed a novel text localization framework based on multi-stage incremental region classification technique. The probable text regions detected by multi-channel stable extremal region detector are passed on to the geometric filters. Text context-based grouping stage combines the related text regions into words. A significant amount of false positives are removed in the process. We also developed a decision tree based ensemble classifier that categorizes the detected regions into text or non-text using the gradient profile feature information. Then the bounding boxes from different channels and views are merged together to get a consolidated list of estimated rectangles.

A region adaptive step selection can improve the accuracy of the stable extremal region detector. The properties of the area being processed can be used to determine the appropriate step-size for the detector. This would assist the subsequent stages by reducing the number of false alarms. Next step would be to recognize words in the bounding boxes with an integrated optical character recognition system. This OCR system can also be used as a feedback mechanism to improve the localization results further. No algorithm can detect text that is occluded by flash light illumination or the foreground objects. So we plan to incorporate an intelligent dictionary based system which might be able to reconstruct the whole word with this partial information at hand.

Bibliography

[AD13] Dimitrios S Alexiadis and Petros Daras. Quaternionic signal processing techniques for automatic evaluation of dance performances from mocap data. *IEEE Transactions on Multimedia*, 2013.

[AF02] O Arikan and D A Forsythe. Interactive motion generation from examples. *ACM Transactions on Graphics*, 21(3):483–490, 2002.

[AFB03] O Arikan, D A Forsythe, and O Brien. Motion synthesis from annotations. *ACM Transactions on Graphics*, 22(3):402–408, 2003.

[AKO04] M I Abouelhoda, S Kurtz, and E Ohlebusch. Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2(1):53–86, 2004.

[Bix98] J Bixler. Tracking text in mixed-mode document. In *Proc. Conference on Document Processing System*, pages 177–185. ACM, 1998.

[BMP01] S Belongie, J Malik, and J Puzicha. Matching shapes. In *In Proc. of the Intl. Conf. on Computer Vision*. IEEE, 2001.

[CCW$^+$04] C Chiu, S Chao, M Wu, S Yang, and H Lin. Content-based retrieval for human motion data. *Journal of Visual Communication and Image Representation*, 15(3):446–466, 2004.

[CCY11] Boon-Seng Chew, L-P Chau, and Kim-Hui Yap. A fuzzy clustering algorithm for virtual character animation representation. *IEEE Transactions on Multimedia*, 13(1):40–49, 2011.

[CK13] Xi Chen and Markus Koskela. Classification of rgb-d and motion capture sequences using extreme learning machine. In *Image Analysis*, pages 640–651. Springer, 2013.

[CL11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[CLKH96] D Chetverikov, J Liang, J Komuves, and R Haralick. Zone classification using texture features. *Proc. of Intl. Conf. on Pattern Recognition*, 3:676–680, 1996.

[CMU07] CMU. Carnegie-mellon mocap database, March 2007.

[CTS$^+$11] H Chen, S Tsai, G Schroth, D Chen, R Grzeszczuk, and B Girod. Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In *18th IEEE International Conference on Image Processing (ICIP), 2011*, pages 2609–2612. IEEE, 2011.

[CVB$^+$03] M Cardle, M Vlachos, S Brooks, E Keogh, and D Gunopulos. Fast motion capture matching with replicated motion editing. In *Proceedings of SIGGRAPH 2003 Technical Sketches and Applications*. ACM SIGGRAPH, 2003.

[CY04] X Chen and A Yuille. Detecting and reading text in natural scenes. In *Proceedings of Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2004.

[CYWM03] D Cai, S Yu, J Wen, and W Ma. Vips: a vision-based page segmentation algorithm. Technical report, MSR, 2003.

[Eff98] Michelle Effros. Practical multi-resolution source coding: Tsvq revisited. In *Proceedings of the Data Compression Conference*, 1998.

[EOW10] B Epshtein, E Ofek, and Y Wexler. Detecting text in natural scenes with stroke width transform. In *IEEE Conference on Computer Vision and Pattern Recognition, 2010*, pages 2963–2970. IEEE, 2010.

[FF05] K Forbes and E Fiume. An efficient search algorithm for motion data using weighted pca. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pages 67–76. ACM, July 2005.

[FS96] Y Freund and R Schapire. Experiments with a new boosting algorithm. In *Proc. of the Thirteeth Int. Conf. on Machine Learning*, pages 148–156. ICML, 1996.

[GEF04] J Gllavata, R Ewerth, and B Freisleben. Text detection in images based on unsupervised classification of high-frequency wavelet coefficients. In *Proceedings of the 17th International Conference on Pattern Recognition. ICPR 2004*, volume 1, pages 425–428. IEEE, 2004.

[GRL$^+$98] S Gold, A Rangarajan, C Lu, S Pappu, and E Mjolsness. New algorithms for 2d and 3d point matching:: pose estimation and correspondence. *Pattern Recognition*, 31(8):1019–1031, 1998.

[HLYW13] Weilin Huang, Zhe Lin, Jianchao Yang, and Jue Wang. Text localization in natural images using stroke feature transform and text covariance descriptors. In *2013 IEEE International Conference on Computer Vision (ICCV)*, pages 1241–1248. IEEE, 2013.

[HPP05] Eugene Hsu, Kari Pulli, and Jovan Popović. Style translation for human motion. *ACM Transactions on Graphics*, 24(3):1082–1089, July 2005.

[IKA87] O Iwaki, H Kida, and H Arakawa. A segmentation method based on office document hierarchical structure. In *Proc. of Intl. Conf. on Systems, Man and Cybernetics*, pages 759–763. IEEE, Oct 1987.

[JB92] A Jain and S Bhattacharjee. Text segmentation using gabor filters for automatic document processing. *Machine Vision and Applications*, 5(3):169–184, 1992.

[JKJ04] K Jung, K Kim, and A Jain. Text information extraction in images and video: a survey. *Pattern recognition*, 37(5):977–997, 2004.

[JY98] A Jain and B Yu. Automatic text location in images and video frames. *Pattern recognition*, 31(12):2055–2076, 1998.

[JZ95] A Jain and Y Zhong. Page segmentation using texture analysis. *The Journal of the Pattern Recognition Society*, 29:743–770, 1995.

[KCK10a] May Chen Kuo, Pei-Ying Chiang, and C.-C. Jay Kuo. Coding of motion capture data via temporal-domain sampling and spatial-domain vector quantization techniques. In *Proceedings of Pacific-Rim Conference on Multimedia, Shanghai, China*. PCM, September 2010.

[KCK10b] Maychen Kuo, Pei-Ying Chiang, and C.-C. Jay Kuo. Overview on mocap data compression. In *Proceedings of APSIPA Annual Summit and Conference 2010, Biopolis, Singapore*, December 2010.

[KG03] Lucas Kovar and Michael Gleicher. Flexible automatic motion blending with registration curves. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM SIGGRAPH/Eurographics, 2003.

[KG04] Lucas Kovar and Michael Gleicher. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics*, 23(3):559–568, August 2004.

[KGS03] S Khedekar, V Govindaraju, and S Setlur. Text - image separation in devanagari documents. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*. ICDAR, 2003.

[Kim96] H Kim. Efficient automatic text location method and content-based indexing and structuring of video database. *Journal of Visual Communication and Image Representation*, 7(4):336–344, 1996.

[KKK11] Harshad Kadu, Maychen Kuo, and C.-C. Jay Kuo. Human motion classification and management based on mocap data analysis. In *Proceedings of Joint ACM workshop on Human gesture and behavior understanding*. ACM Multimedia, December 2011.

[KLA$^+$01] T Kasai, G Lee, H Arimura, S Arikawa, and K Park. Linear-time longest-common-prefix computation in suffix arrays and its applications. In *Proceedings of 12th Annual Symposium on Combinatorial Pattern Matching. London, UK*. CPM, 2001.

[KNSV93] Y Krishnamoorthy, G Nagy, S Seth, and M Vishwanathan. Syntactic segmentation and labeling of digitized pages from technical journals. *IEEE Computer Vision, Graphics and Image Processing*, 47:327–352, 1993.

[KPS03] T Kim, S Park, and S Shin. Rhythmic-motion synthesis base on motionbeat analysis. *ACM Transactions on Graphics*, 22(3):392–401, 2003.

[KS03] J Karkkainen and P Sanders. Simple linear work suffix array construction. In *Proceedings of 13th International Conference on Automata, Languages and Programming*. Springer, 2003.

[KS13] Reddy Kishore and Mubarak Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications*, 24(5):971–981, 2013.

[KSPP05] Dong Kyue Kim, Jeong Seop Sim, Heejin Park, and Kunsoo Park. Constructing suffix arrays in linear time. *Journal of Discrete Algorithms*, 3(2):126–142, 2005.

[KSU$^+$13] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Mikio Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Jordi Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis-Pere de las Heras. Icdar 2013 robust reading competition. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 1484–1493. IEEE, 2013.

[LCC00] H Lee, Y Choy, and S Cho. Geometric structure analysis of document images: A knowledge-based approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22:1224–1240, Nov 2000.

[LCR$^+$02] J Lee, J Chai, P Reitdma, J Hodgins, and N Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, 21(3):491–500, 2002.

[LDK00] H Li, D Doermann, and O Kia. Automatic text detection and tracking in digital video. *IEEE Transactions on Image Processing*, 9(1):147–156, 2000.

[LDL05] J Liang, D Doermann, and H Li. Camera-based analysis of text and documents: a survey. *International Journal of Document Analysis and Recognition (IJDAR)*, 7(2-3):84–104, 2005.

[LKL$^+$04] C Li, P Kulkarni, L Liu, B Prabhakaran, and L Khan. Real-time classification of multivariate motion data using support vector machines (svm). In *Proceedings of the 5th Int'l Workshop on Multimedia Data Mining*, pages 1–7. MDM/KDD, 2004.

[Low99] D Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, pages 1150–1157. IEEE, 1999.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[LP02] Karen Liu and Zoran Popović. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics*, 21(3):408–416, July 2002.

[LPS$^+$03] S Lucas, A Panaretos, L Sosa, A Tang, S Wong, and R Young. Icdar 2003 robust reading competitions. In *ICDAR*, volume 2003, page 682, 2003.

[LSRJ10] H Ling, S Soatto, N Ramanathan, and D Jacobs. Face verification across age progression using discriminative methods. *IEEE Transactions on Information Forensics and Security*, 5:82–91, March 2010.

[Luc05] S Lucas. Icdar 2005 text locating competition results. In *Proceedings. Eighth International Conference on Document Analysis and Recognition*, pages 80–84. IEEE, 2005.

[LW02] R Lienhart and A Wernicke. Localizing and segmenting text in images and videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(4):256–268, 2002.

[LZP07] C Li, S Q Zheng, and B Prabhakaran. Segmentation and recognition of motion streams by similarity search. *ACM Transactions on Multimedia Computing, Communications and Applications*, 3(16), 2007.

[LZWP03] F Liu, Y Zhuan, F Wu, and Y Pan. 3d motion retrieval with motoin index tree. *Computer Vision and Image Understanding*, 92(2-3):265–284, 2003.

[Mak83] H Makino. Representation and segmentation of document images. In *Proc. of Computer Society Conference on Pattern Recognition and Image Processing*, pages 291–296. IEEE, 1983.

[MCUP04] J Matas, O Chum, M Urban, and T Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. 22(10):761–767, 2004.

[MR06] Müller and T Röder. Motion template for automatic classification and retrieval of motion capture data. In *Proceedings of 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2006)*. Eurographics Association, 2006.

[MRC05a] Müller, T Röder, and M Clausen. Efficient indexing and retrieval of motion capture data based on adaptive segmentation. In *Proceedings of Fourth International Workshop on Content-Based Multimedia Indexing (CBMI)*, 2005.

[MRC05b] M Müller, T Röder, and M Clausen. Efficient content-based retrieval of motion capture data. *ACM Trans. Graph.*, 24(3):677–685, 2005.

[NK88] N M Nasrabadi and R A King. Image coding using vector quantization: a review. In *Proceedings of IEEE Transactions on Communications*, August 1988.

[NM11] Lukáš Neumann and Jiří Matas. Text localization in real-world images using efficiently pruned exhaustive search. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 687–691. IEEE, 2011.

[NM12] Lukáš Neumann and Jiří Matas. Real-time scene text localization and recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3538–3545. IEEE, 2012.

[NM13] Luka Neumann and Jose Matas. Scene text localization and recognition with oriented stroke detection. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 97–104. IEEE, 2013.

[NPP$^+$13] Valsamis Ntouskos, Panagiotis Papadakis, Fiora Pirri, et al. Discriminative sequence back-constrained gp-lvm for mocap based action recognition. In *International Conference on Pattern Recognition Applications and Methods*, 2013.

[ODP99] Oleg Okun, David Dœrmann, and Matti Pietikainen. Page segmentation and zone classification: the state of the art. Technical report, DTIC Document, 1999.

[PB02] K Pullen and C Bregler. Motion capture assisted animation: texturing and synthesis. In *Proceedings of ACM SIGGRAPH, ACM, Computer Graphics Proceedings, Annual Conference Series.* ACM SIGGRAPH, 2002.

[RKM08] Bodo Rosenhahn, Reinhard Klette, and Dimitris Metaxas. *Human motion: Understanding, Modeling, Capture and Animation.* Springer, Dordrecht, Netherlands, 2008.

[RSH$^+$05] Liu Ren, Gregory Shakhnarovich, Jessica Hodgins, Hanspeter Pfister, and Paul Viola. Learning silhouette features for control of human motion. *ACM Transactions on Graphics*, 24(4):1303–1331, October 2005.

[RWH96] M Revow, C Williams, and G Hinton. Using generative models for handwritten digit recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):592–606, 1996.

[SC12] Sreemananth Sadanand and Jason Corso. Action bank: A high-level representation of activity in video. In *Computer Vision and Pattern Recognition*, pages 1234–1241. IEEE, 2012.

[SHP04] Alla Safonova, Jessica Hodgins, and Nancy Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics*, 23(3):514–521, August 2004.

[SKK04] Yasuhiko Sakamoto, Shigeru Kuriyama, and Toyohisa Ka. Motion map: Image-based retrieval and segmentation of motion data. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pages 259–266. The Eurographics Association, July 2004.

[SSD11] Asif Shahab, Faisal Shafait, and Andreas Dengel. Icdar 2011 robust reading competition challenge 2: Reading text in scene images. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1491–1496. IEEE, 2011.

[SSK$^+$13] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[SSL02] W Schollhorn, D Stefanyshyn, and W Liu. Identification of individual walking patterns using time discrete and time continuous data sets. *Gait and Posture*, 15:180–186, 2002.

[VJ01a] P Viola and M Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. *Advances in Neural Information Processing System*, 14, 2001.

[VJ01b] P Viola and M Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of Computer Society Conference on Computer Vision and Pattern Recognition, 2001*, volume 1. IEEE, 2001.

[VJS05] P Viola, M Jones, and D Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005.

[WB03] J Wang and B Bodenheimer. An evaluation of a cost metric for selecting transitions between motion segments. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2003*. ACM SIGGRAPH/Eurographics, 2003.

[WCYL03] M Y Wu, S Chao, S Yang, and H Lin. Content-based retrieval for human motion data. In *IPPR Conf. on Computer Vision, Graphics and Image Processing*. IPPR, 2003.

[Wik04] Wikipedia. Support vector machine, August 2004.

[WJ06] Christian Wolf and Jean-Michel Jolion. Object count/area graphs for the evaluation of object detection and segmentation algorithms. *International*

Journal of Document Analysis and Recognition (IJDAR), 8(4):280–296, 2006.

[WWX09] Shuangyuan Wu, Zhaoqi Wang, and Shihong Xia. Indexing and retrieval of human motion data by a hierarchical tree. In *Proceedings of VRST 2009*. ACM, November 2009.

[WXWL09] S Wu, S Xia, Z Wang, and C Li. Efficient motion data indexing and retrieval with local similarity measure of motion strings. *The Visual Computer*, 25(5-7):499–508, 2009.

[YBL$^+$12] Cong Yao, Xiang Bai, Wenyu Liu, Yi Ma, and Zhuowen Tu. Detecting texts of arbitrary orientations in natural images. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1083–1090. IEEE, 2012.

[YHGZ05] Q Ye, Q Huang, W Gao, and D Zhao. Fast and robust text detection in images and video frames. *Image and Vision Computing*, 23(6):565–576, 2005.

[YI06] L Yangxing and T Ikenaga. A contour-based robust algorithm for text detection in color images. *IEICE transactions on information and systems*, 89(3):1221–1230, 2006.

[YS04] K Yang and Shahabi. A pca-based similarity measure for multivariate time series. In *Proceedings of the 2nd ACM international workshop on Multimedia databases*, pages 65–74. ACM, 2004.

[YT11] Chucai Yi and YingLi Tian. Text string detection from natural scenes by structure-based partition and grouping. *Image Processing, IEEE Transactions on*, 20(9):2594–2605, 2011.

[YT13] Chucai Yi and Yingli Tian. Text extraction from scene images by character appearance and structure modeling. *Computer Vision and Image Understanding*, 117(2):182–194, 2013.

[ZMCF05] V B Zordan, A Majkowska, B Chiu, and M Fast. Dynamic response for motion capture animation. *ACM Transactions on Graphics*, 24(3):697–701, 2005.

Appendix

The CMU mocap database [CMU07] motion capture files used in our research, are listed below. All these files are *.amc* files containing the frame by frame values of 59 rotational angles of the 29 joints in the human skeleton, plus a triplet representing the 3D position of the root joint at that instance. The number before the '_' symbol is the subject identification number and the one after it is the motion serial number. *For e.g.* 09_01 is the 09_01*.amc* file which is the first motion of the subject '09'. The corresponding skeleton file having the details of the subject's skeleton structure is 09*.asf*.

1. *Run*: 09_01, 09_02, 09_03, 09_04, 09_05, 09_06, 09_07, 09_08, 09_09, 09_10, 09_11, 35_17, 35_18, 35_19, 35_20, 35_21, 35_22, 35_23, 35_24, 35_25, 35_26, 127_06, 127_07, 127_08, 141_01, 141_02, 16_55.

2. *Walk*: 35_01, 35_02, 35_03, 35_04, 35_05, 35_06, 35_07, 35_08, 35_09, 35_10, 35_11, 35_12, 35_13, 35_14, 35_15, 35_16, 35_28, 35_29, 35_30, 35_31, 02_01, 02_02, 07_01, 07_02, 07_03, 07_06, 07_07, 07_08, 07_09, 07_10, 07_11, 08_01, 08_02, 08_03, 08_06, 08_08, 08_09, 08_10, 12_01, 12_02, 12_03, 16_15, 16_16, 16_21, 16_22, 16_31, 16_32.

3. *Forward Jump*: 16_05, 16_06, 16_07, 16_09, 16_10, 13_11, 13_13, 13_19, 13_32.

4. *Forward Dribble*: 06_02, 06_03, 06_05.

5. *Cartwheel*: 49_06, 49_07, 49_08.

6. *Kickball*: 10_01, 10_02, 10_03, 10_05, 10_06, 11_01.

7. *Boxing*: 13_17, 13_18, 14_01, 14_02, 14_03, 15_13, 17_10.
8. *Balance*: 49_18, 49_19, 49_20.
9. *Sit and Stand up*: 13_01, 13_02, 13_03, 14_27, 14_28.
10. *Laugh*: 13_14, 13_15, 13_16.
11. *Sweep Floor*: 13_23, 13_24, 13_25.
12. *Wash Windows*: 13_20, 13_21, 13_22, 14_10, 14_11.
13. *Climb Ladder*: 13_33, 13_34, 14_33, 14_34, 14_35.
14. *Steps*: 13_35, 13_36, 13_37, 13_38, 14_21, 14_22, 14_23.
15. *Mop*: 14_13, 14_15.
16. *Tiptoe*: 13_10, 13_12, 14_07, 14_08, 14_09.
17. *Pick Box Bend Waist*: 115_01, 115_02, 115_03, 115_04, 115_05, 115_10.
18. *Pick Box Bend Knees*: 115_06, 115_07, 115_08, 115_09.
19. *Fly Stroke*: 126_06, 126_07, 126_08, 126_09.
20. *Free Style*: 126_10, 126_11, 126_12, 125_06.
21. *Breast Stroke*: 125_01, 125_02, 125_04, 126_03, 126_04, 126_05.
22. *Hop on Left Foot*: 132_23, 132_24, 132_25, 132_26, 132_27, 132_28.
23. *Bouncy Walk*: 132_29, 132_30, 132_31, 132_32, 132_33, 132_34.
24. *Marching*: 138_01, 138_02, 138_03, 138_04, 138_05, 138_06, 138_07, 138_08, 138_09, 138_10.
25. *Nursery Rhyme Tea Pot*: 24_01, 25_01, 26_03, 26_04, 27_03, 27_07, 28_02, 28_06, 29_03, 29_08, 30_02, 30_08, 31_02, 31_06, 32_04, 32_08.

26. *Nursery Rhyme Cock Robin*: 26_07, 26_08, 27_05, 27_10, 28_04, 28_08, 29_06, 29_10, 30_05, 30_06, 30_10, 31_04, 31_08, 32_06, 32_10.

27. *Swing*: 64_01, 64_02, 64_03, 64_04, 64_05, 64_06, 64_07, 64_08, 64_09, 64_10.

28. *Placing Tee*: 64_16, 64_17, 64_18, 64_19, 64_20.

29. *Salsa Dance*: 60_01, 60_02, 60_03, 60_04, 60_05, 60_06, 60_07, 60_08, 60_09, 60_10, 60_11, 60_12, 60_13, 60_14, 60_15.

30. *Get Up From Floor*: 111_06, 111_07, 111_08.