Aligning Textual & Visual Data: Towards Scalable Multimedia Retrieval

Pramod Sankar Kompalli
200499002

Doctoral Dissertation
Submitted in partial fulfillment of requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

International Insititute of Information Technology, Hyderabad

May, 2015

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

This is to certify the thesis entitled "Aligning Textual & Visual Data: Towards Scalable Multimedia Retrieval" submitted by Pramod Sankar Kompalli, to the International Institute of Information Technology Hyderabad, India, for the award of the degree of Doctor of Philosophy in Computer Science and Engineering is a record of bonafide research work carried out by him under my supervision. The results embodied in this thesis have not been submitted for any other degree or diploma.

_____ _____

Date Adviser: Prof. C. V. Jawahar

Dedicated to those who made great sacrifices,
just so I could go off on an adventure

Mom, Dad and Appu

Acknowledgments

At the outset, I would like to sincerely thank my thesis advisor Prof. C. V. Jawahar for his constant support and guidance throughout this decade long journey. He has challenged me with hard problems and lofty goals, while maintaining a delicate balance between allowing me intellectual freedom and providing me with some much-needed help. He always had great insightful advice on navigating through the workings of the research world. Most importantly, he has been extremely patient in dealing with me, and my quirks. I'll always be grateful for all that he taught me explicitly and implicitly. I'd also like to thank Prof. P. J. Narayanan for being a great mentor. I am grateful to Dr. Rammurthy Garimella, who encouraged me to join IIIT-H and helped me secure a position at CVIT.

There a number of great peers at CVIT over the years. I must thank Balu, MNSSK and Sesh, who helped me get started on my work here. I would like to thank my collaborators: Pramod P, Saurabh, Anurag, Yashaswi, Sudha Praveen, Naveen and Udit. I've enjoyed great times, food and conversations with Rishi, Suman Karthik, Satya V., Praveen Dasigi, Sanjeev Kumar and Pawan Harish. I profusely thank Mr. Satya for all his help with administrative issues. I also thank the admins of DC++@IIIT and all the downloaders, who made research life so much fun.

I spent two lovely summers at Oxford where I enjoyed working with and learning from a great set of people: Josef, Mark, James, ME and Ondra. A special thanks to Mukta and Karteek for helping me out with new-place-issues. I will be indebted to Prof. Andrew Zisserman, for being patient and understanding in working with me, and for all the enthusiastic discussions we had during his visits to India. Dr. Manmatha has been a wonderful collaborator, whose support made a major impact on the outcomes of this thesis. He was particularly instrumental in bringing a sense of pride and purpose to my work, for which I can not thank him enough.

I've had the good fortune of some very great friends, that I might not get to thank elsewhere. Sai Chand's perspective on life has helped me think better, and his friendship is invaluable. Kranthi has always been a very dear friend who always had only good things

to say about me. Sravani has been a generous friend of several years, whose faith in me has kept me going during tough times.

My family, of course, played a major role in making this thesis possible, through their unwavering support and encouragement. My parents have let me pursue my passions, and persevered with me all these years. My Dad, was assertive that my research not be disturbed despite his poor health, my Mom only wanted to know what she could do to help. My darling sister has always been a pillar of support in all my endeavors. The love and understanding of my wife, Usha Sravanthi, is a source of my strength and peace. I can only hope to make up for their many sacrifices in some way.

Abstract

The search and retrieval of relevant images and videos from large repositories of multimedia, is acknowledged as one of the hard challenges of computer science. With existing pattern recognition solutions, one cannot obtain detailed, semantic description for a given multimedia document. Several limitations exist in feature extraction, classification schemes, along with the incompatibility of representations across domains. The situation will most likely remain so, for several years to come.

Towards addressing this challenge, we observe that several multimedia collections contain similar parallel information that are: i) semantic in nature, ii) weakly aligned with the multimedia and iii) available freely. For example, the content of a news broadcast is also available in the form of newspaper articles. If a correspondence could be obtained between the videos and such parallel information, one could access one medium using the other, which opens up immense possibilities for information extraction and retrieval. However, it is challenging to find the mapping between the two sources of data due to the unknown semantic hierarchy within each medium and the difficulty to match information across the different modalities. In this thesis, we propose novel algorithms that address these challenges.

Different $< Multimedia, Parallel\ Information >$ pairs, require different alignment techniques, depending on the granularity at which entities could be matched across them. We choose four pairs of multimedia, along with parallel information obtained in the *text* domain, such that the data is both challenging and available on a large scale. Specifically, our multimedia consists of movies, broadcast sports videos and document images, with the parallel text coming from scripts, commentaries and language resources. As we proceed from one pair to the next, we discover an increasing complexity of the problem, due to a relaxation of the temporal binding between the parallel information and the multimedia. By addressing this challenge, we build solutions that perform increasingly fine-grained alignment between multimedia and text data.

The framework that we propose begins with an assumption that we could segment the multimedia and the text into meaningful entities that could correspond to each other. The problem then, is to identify *features* and learn to match a text-entity to a multimedia-segment (and vice versa). Such a matching scheme could be refined using additional constraints, such as temporal ordering and occurrence statistics. We build algorithms that could align across i) movies and scripts, where sentences from the script are aligned to their respective *video-shots* and ii) document images with lexicon, where the words of the dictionary are mapped to clusters of word-images extracted from the scanned books.

Further, we relax the constraint in the above assumption, such that the segmentation of the multimedia is not available *apriori*. The problem now, is to perform a joint inference of segmentation and annotation. We address this problem by building an over-complete representation of the multimedia. A large number of putative segmentations are matched against the information extracted from the parallel text, with the joint inference achieved through dynamic programming. This approach was successfully demonstrated on i) Cricket videos, which were segmented and annotated with information from online commentaries and ii) word-images, where sub-words called Character N-Grams, are accurately segmented and labeled using the text-equivalent of the word.

As a consequence of the approaches proposed in this thesis, we were able to demonstrate text-based retrieval systems over large multimedia collections. The semantic level at which we can retrieve information was made possible by the annotation with parallel text information. Our work also results in a large set of labeled multimedia, which could be used by sophisticated machine learning algorithms for learning new concepts.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The success of search engines on the Web, such as Google [3], has tremendously transformed the way we access and use information every day. However, similar ability to search the content, seems to elude the massive multimedia collections available online. With the advent of economical digital cameras, camcorders and more recently, smartphones, all of us are creating vast amounts of image and video content, especially for others' consumption. However, the reach of the massive multimedia databases, including about 250 billion photos on Facebook [17] and 100 hours of video uploaded on YouTube [32] every minute, is limited mostly to close friends and family. The ability to search and retrieve from these collections would unlock their true information potential. In this thesis, we address this very problem and propose approaches, techniques and solutions to design retrieval schemes over a variety of real-world multimedia collections.

1.1 Motivation

Personal Use: We build our personal image and video content primarily to document our experiences. It should be convenient to find the relevant photos and videos when one wants to relive particular events, for example the video of one's child acting in a play.

Knowledge: Several multimedia collections store large amounts of valuable information, such as through photo-journalism, documentary film making, etc. Videos are being increasingly used as an effective medium to teach and to inform [25]. A retrieval system over such multimedia repositories would help in obtaining factual data as well as valuable knowledge.

Entertainment: Needless to say, movies, TV shows, live sports coverage, music videos, etc. form an important part of everyone's life. The volume of such content is much larger than what one could consume in a lifetime. Hence, a mechanism to find or recommend content based on user preference is necessary.

Security: Surveillance cameras have become an indispensable security tool. Much of the video feed is directly archived and accessed only in the case of an incident. It is necessary to build systems which can retrieve relevant video clips for a person or event of interest.

However, much of the semantic information criteria that is of interest in these domains, is extremely hard to define as well as understand using existing techniques.

1.2 Challenges with Multimedia Retrieval

Multimedia is a synchronized co-existence of several types of data such as visual, audio, text, etc. While processing and making sense of each modality individually is a challenge in itself, the joint inference is much harder. Further, such challenges have to be isolated from the user, who should be provided with the same kind of convenience expected from web search engines.

Challenges with Visual Data: Computer vision, the study towards the understanding of visual content, is acknowledged as one of the hardest recognition tasks. The challenge arises from the richness in the visual world and consequently the immense variety in the visual data. There have been some recent successes with respect to classifying the object

category for a given image (or window) [85], but semantic recognition and description still remains a challenge. In case of video, action recognition is further difficult, as it depends on unreliable human pose estimation at each frame. One of the core issues with building visual recognition systems is the limitation of existing classifiers: i) they need large amounts of training data, which is expensive to generate; ii) they are computationally expensive to train and evaluate and iii) they are not reliable across large number of visual categories.

Challenges with Auditory Data: Speech recognition has become mature enough to enable a voice-based assistant on a smartphone [31]. Despite this success, generic speech recognition is still an unsolved problem. Multiple speakers, background noise, varied accents, etc. pose serious challenges towards building a universal speech recognizer. Languages such as those from India, China, Africa, etc. do not yet have robust recognition engines, and might require much effort dedicated towards this. Apart from speech, it is also difficult to identify and recognize other kinds of sounds such as music, whistling, humming [100], etc.

Challenges in Information Retrieval: There is a "semantic gap" between how an information need is expressed and how multimedia is represented [207]. This problem appears to a lesser extent in text/web search engines [3], mostly because the users typically learn to pose better queries in text. It is rarely possible to search for a particular image with just a few keywords, while that is fairly easy in text search engines. With multimedia, the labels generated from recognizers or tags provided by humans are never thorough or enough to describe the content. In case of videos, the temporal annotation of long videos is quite complex also.

One of the key challenges with enabling retrieval is the lack of *apriori* understanding of the user's information need. In current retrieval solutions, one can only ask such questions that can be answered with the existing recognition technology. In contrast, if a user can

express the information need in a manner that could be modeled easily with visual data, retrieval would only be a matter of matching the information need with the data collection. In this context, the motivation for our work is the key observation that there exist several sources of similar or parallel textual information that is "consumable" by the users and that meet their immediate semantic expectations. If we could align this parallel information with visual data, the user's information need can be posed and processed in the easy-to-handle text domain.

1.3 Problem Setting

In this work, we address the challenges of multimedia retrieval by leveraging additional information available for several classes of visual data. Particularly, we are interested in relevant *textual* information that contains semantic descriptions of the multimedia content. We then apply state-of-the-art techniques for the respective multimedia modalities resulting in intermediary inferences, which are then fused to make a decision on the overall labeling problem. Specifically, we focus on four classes of multimedia-text pairs: i) movies & scripts, ii) sports videos & commentaries, iii) digitized books & dictionary of the language and iv) word-images & text labels.

Our approach is motivated from the ground-up by the need to create *text-based* retrieval systems, where the information need is expressed as a small set of keywords. Not only is the convenience of text-retrieval appreciated by the users, but several advances in text-IR community could be extended to multimedia data, if such the visual to text transformation were made feasible.

We limit the scope of the thesis to building prototype retrieval systems on multiple classes of large-scale real-world datasets. We do not aim to build commercial systems as part of the thesis, that might require better UIs, real-time data processing and significant resources for delivery of content and maintaining services.

1.4 Contributions

The major contributions of this thesis are:

1. **Annotation through Alignment:** Our first contribution is towards addressing the challenge of obtaining an alignment across visual data and parallel textual information. We present several novel *intermediate* representations of image/video and text, in order to match them across the modalities. We propose a dynamic programming based algorithm, that enforces temporal/sequential constraints to refine the matching. The proposed alignment scheme could also infer the segmentation of the visual data, when it has not been explicitly provided. We apply the proposed approach towards annotation of movies with scripts, Cricket videos with text-commentary and document images with a lexicon.

2. **Scalable Annotation:** Our second contribution is towards scalable annotation of multimedia. We propose an indexing based annotation approach, that re-uses classifier scores for similar data. Our work is significant because, we reduce the complexity of the test data, rather than that of the classification. As a consequence, an annotation scheme that would usually be of order $O(N \cdot M)$, (where N, M are test and training data sizes respectively), would only require $O(logN \cdot logM)$ operations. This translates to a speedup of 500$\times$, from 150 years to about 100 days while annotating 36 Million images. The presented approach is fairly agnostic to the underlying classification scheme, making it easily applicable to multiple recognition tasks. Further, our work enables scalable retrieval over large multimedia datasets, as the underlying representation that is indexed is in the text form. This allows us to build multimedia retrieval systems by using popular text search engines [12, 22].

3. **Significant Applications:** Our work has immediate practical applications towards building retrieval systems on Indian-language document image collections. Such

images cannot be made accessible with existing character recognition technology (OCR) due to challenges with Indic scripts. We address the problem by alleviating the challenge of character segmentation by focusing on higher-level primitives such as the *word*, and *Character NGrams*, which are sequences of character segments from a given word image. We demonstrate our approach over 1000 Telugu books, consisting of 120K images, the largest non-English document image collection that is searchable at the content level. As a result of our work, we were able to build document image retrieval systems that i) can be queried by text keywords, ii) can search from an unlimited vocabulary and iii) can retrieve at the sub-word level.

4. **Labeled Datasets:** The alignment schemes that we propose, are applicable to several situations where weak-supervision is available for multimedia datasets. We present a class of approaches that could turn *weak*-annotation into *strong*-annotation, at multiple semantic granularities. This allows for significant speed-up in annotation, requiring little human intervention for fine-grained labeling. The labeled data obtained in this fashion would be valuable towards learning features and classifiers for several new visual categories.

1.5 Thesis Outline

We begin with presenting a review of the literature in Chapter 2. Several previous approaches towards building multimedia retrieval solutions are described and their limitations and shortcomings are analyzed. The approaches proposed in this thesis are summarized in Figure 1.1, which provides details of the multimedia collections and the textual information used in the annotation process. The outcome of the work in each Chapter and the increasing scope of the solution across the Chapters should be evident.

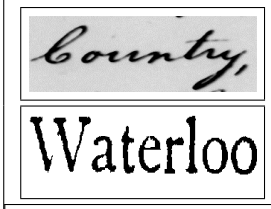
<table>
<tr><th></th><th>Multimedia</th><th>Parallel Text</th><th>Alignment</th></tr>
<tr><td rowspan="2">Chapter 3</td><td>Movies
  </td><td>Scripts
[Jerry's apartment]
(Jerry's explaining George's idea to Kramer)
JERRY: So, the show would be about my real life. And one of the characters would be based on you.
KRAMER: (Thinks) No, I don't think so.</td><td>Shots vs. Sentences</td></tr>
<tr><td colspan="3">Result: Video clips can be retrieved based on people, location, actions and dialogues
Limitation: Limited by availability of scripts & to video material trained for</td></tr>
<tr><td rowspan="2">Chapter 4</td><td>Cricket Videos
  </td><td>Commentary
16.1 Collingwood to Sehwag, SIX, oh, hello, Sehwag greets him with a slog over mid-off for six, its all going terribly wrong for England this morning! Sehwag just clears the leg and swings, fab shot
16.2 Collingwood to Sehwag, SIX, carnage! And thats the 100 stand, and Sehwag's fifty with six more, this one disdainfully slogged over midwicket, what a batsman he is</td><td>Scenes vs. Paragraphs</td></tr>
<tr><td colspan="3">Result: Video clips can be retrieved based on players, outcome and descriptions
Limitation: Limited to Cricket videos</td></tr>
<tr><td rowspan="2">Chapter 5</td><td>Digitized Books
  </td><td>Born-digital Content
  </td><td>Segmented Words vs. Dictionary</td></tr>
<tr><td colspan="3">Result: Relevant books & pages can be retrieved for Telugu word queries
Limitation: Limited to chosen vocabulary set of 100K</td></tr>
<tr><td rowspan="2">Chapter 6</td><td>Segmented Words
  </td><td>Word Labels
Country
Waterloo</td><td>Sub-Words vs. Character N-Grams</td></tr>
<tr><td colspan="3">Result: Words retrieved irrespective of prefix, suffix & sandhi/samaas
Limitation: Applicable to most document images</td></tr>
</table>

Figure 1.1 An overview of the different multimedia documents that we build retrieval systems for, with the approach of alignment between visual and textual content. The temporal binding between the multimedia and the text varies from rigidly tight to almost unrestricted depending on the granularity at which the text describes the visual content. The challenges, methods and results from each row of the figure are presented in a corresponding Chapter in the thesis.

We begin describing our work in Chapter 3 towards annotating movies and TV shows, with parallel information obtained from fan-generated *transcripts*. These scripts contain information about the characters that are present in the video, their actions, expressions, and what they are speaking. A movie of a few hours length, is typically described by a few pages of text. The video-script alignment problem is one of assigning a piece of text to the corresponding piece of video. In order to align the text to the video, we segment the video into shots and the text into sentences. The alignment is then posed as a multiple assignment problem that assigns multiple sentences of the script to a shot of the video. An intelligent assignment of sentences to shots requires a matching scheme between the two. This matching is based on three cues extracted from the sentences and shots: <Location, Face, Speech>. These properties are extracted from the shot using state-of-the-art methods for each cue. The alignment is obtained by optimizing a cost function defined using this matching scheme. The technique was applied on episodes from the sitcom Seinfeld, and also tested on Indian films and Charlie Chaplin's silent movies. The resulting alignment allows the user to retrieve all shots where for eg., Charlie Chaplin is picking something up, Elaine is answering the phone or Kramer enters Jerrys Apartment in his characteristic style. Through the method we propose here, we take the first step in aligning multimedia with text when the basic entities of both the video (i.e. shot) and text (i.e. sentence) are known beforehand.

In Chapter 4, we address the next class of videos that come from sports broadcasts of *Cricket*. The parallel information for these videos is obtained from online commentaries, written by journalists. Contrasting with movies, the meaningful segmentation of the Cricket videos is a sequence of related shots that form a *scene*. Also, the text consists of multiple sentences (a paragraph) that should always be assigned to the same scene. While the segmentation in the text space is straightforward, the challenge is in segmenting the video into meaningful scenes, and then labeling each scene with the appropriate

paragraph of text. This presents a chicken-and-egg problem, if the video is segmented it could be matched against descriptions, and vice-versa. We solve this problem by proposing a text-driven temporal segmentation of video in the case of broadcast Cricket videos. This is made feasible by learning models for scene-categories using a set of visual-temporal features. The commentary is used to build a *hypothetical video* using these scene models. This model is then matched and aligned with the visual-temporal features observed from the broadcast video, which in turn provides us the segmentation. Following this, the scene segments were annotated with descriptions from the commentary. Such an annotation allows for search and retrieval, personalized highlight generation, etc. For example, users could search for all instances where the batsman used a *square-cut*, or the bowler bowled a *full-toss*. These are examples of queries which are quite difficult to answer without using parallel-text. With this aspect of our work, we have performed a joint segmentation and annotation of videos, when the basic entities of the video are not provided *apriori*.

In Chapter 5, we turn our attention to visual data present in images, specifically those images that consist exclusively of *text* data. We are aware only of the language of the textual data present in the images, but not the actual content itself. The goal is to extract the textual content present in the form of pixels in the image. The parallel information for these document images, comes in the form of the language's lexicon. The semantic segment of both the visual and textual data is well-defined, which is the "word". The word-segments from the document images are compared with *exemplars* obtained for the dictionary words. However, unlike video data, there is no temporal information that could help with the alignment. Instead of performing explicit matching between all word-images and exemplars (which is inefficient), we propose an indexing scheme, where clusters of word-images are aligned with clusters of exemplars. The clustering itself is made efficient by using Hierarchical K-Means and KD-Trees. By this approach we were able to annotate a large collection of 1000 books of *Telugu* language, with a 500$\times$ speedup over traditional

word-recognition. The resulting search system searches from about 36 Million words in less than a second, without the use of an Optical Character Recognizer (OCR). Thus, in the case of document images, the image data and the text data are aligned across common entities that are completely distributed across the collection with no temporal ordering.

One of the limitations of the work in Chapter 5, is the restricted vocabulary that can be answered by the retrieval system; out-of-vocabulary (OOV) queries are simply rejected. In Chapter 6, we overcome this limitation by exploring the possibility of annotation at the sub-word level. We begin with a set of word-images that are provided along with their equivalent text label. We propose a novel segmentation of the word image into sequences of putative characters, called *Character N-Gram Images* (CNG-Img). Similarly, the text label is also segmented into text n-grams at character level (t-CNG). Due to over-segmentation of the word-image, there would typically be more CNG-Img than t-CNG. Each t-CNG would map to only one CNG-Img. The alignment problem is posed as a weakly-supervised segmentation, where models for t-CNG are "located" among the CNG-Img set, which results in an exact segmentation of the word into characters and t-CNGs. However, it is difficult to learn the models for the entire t-CNG, as that would require large amount of training data. We solve this issue by building models for the t-CNG by concatenating the individual character models. Following this process, we obtain CNG-level labeling of the word image in the given collection. With such a labeling, we were able to build a retrieval system that could retrieve similar words, irrespective of modifiers such as prefix-suffixes, and for Indian languages, across *sandhi-samaas* conjuncts. The labeled exemplars were then used to build a recognition system, that is robust to heavy degradations in document images. Through the CNG-Img work, we were able to apply an alignment between a bag-of-visual entities to a bag-of-text entities, where each bag contains objects at multiple granularities ranging from characters to words (and all the representations in-between).

While we deal with a variety of multimedia collections, the underlying solutions follow a common theme: i) an exploitation of domain-specific cues that are tailored for the given multimedia collection, ii) a fusion of information between the visual and textual modalities, that yields a solution that is otherwise not possible and, iii) use of spatial/temporal constraints that are enforced by the order of events in the two modalities or in the distribution of data in a high-dimensional space.

We conclude our work in Chapter 7 and present future directions of research that could be explored given the work carried out in this thesis.

1.6 Related Publications

Part of the work described in this Thesis has previously appeared as

- Pramod Sankar K., R. Manmatha and C. V. Jawahar, *Large Scale Document Image Retrieval by Automatic Word Annotation*, IJDAR 2014 [172]
- Udit Roy, Naveen Sankaran, Pramod Sankar K. and C. V. Jawahar, *Character N-Gram Spotting on Handwritten Documents using Weakly-Supervised Segmentation*, ICDAR 2013 [191]
- Shrey Dutta, Naveen Sankaran, Pramod Sankar K. and C. V. Jawahar, *Robust Recognition of Degraded Documents Using Character N-Grams*, DAS 2012 [80]
- Sudha Praveen M., Pramod Sankar K. and C. V. Jawahar, *Character n-Gram Spotting in Document Images*, ICDAR 2011 [212]
- Rahul Jain, Sudha Praveen M., Pramod Sankar K. and C. V. Jawahar, *An Indexing Approach for Speeding-Up Image Classification*, ICVGIP 2010 [117]
- Pramod Sankar K, C. V. Jawahar and R. Manmatha, *Nearest Neighbor based Collection OCR*, DAS 2010 [177].

- Pramod Sankar K., C. V. Jawahar and Andrew Zisserman, *Subtitle-Free Movie to Script Alignment*, BMVC 2009 [176].

- Pramod Sankar K. and C. V. Jawahar, *Probabilistic Reverse Annotation for Large Scale Image Retrieval*, CVPR 2007 [175].

- Pramod Sankar K., Saurabh Pandey and C. V. Jawahar, *Text Driven Temporal Segmentation of Cricket Videos*, ICVGIP 2006 [173].

- Pramod Sankar K. and C. V. Jawahar, *Enabling Search over Large Collections of Telugu Document Images-An Automatic Annotation Based Approach*, ICVGIP 2006 [174].

Chapter 2

Background

In this Chapter, we present the precursors to our work. We set the background for the work presented in the rest of this thesis. Literature review that is relevant to specific multimedia collections shall be covered in the respective Chapters addressing them.

2.1 Content Based Image/Video Retrieval

2.1.1 CBIR

The Content Based Image Retrieval (CBIR) approaches address the question: "given the image Q, find me similar images in the dataset W". This question can be asked of videos (CBVR) and multimedia (CBMR) as well. The CBxR approach has been actively pursued for several decades [36, 77, 139, 207]. Some of the key research works that have made the most impact in CBxR research have been pictured in Figure 2.1. These works have raised fundamental issues in content based access to image, video and document collections.

During the "early" years of CBIR, the focus has mostly been around the low-level feature extraction and matching. Several features such as color, texture and shape were proposed and evaluated along with various distance measures such as Euclidean, Earth-Movers, etc. The progress from this exploration resulted in early CBIR systems such as the QBIC [37, 94] (see Figure 2.2), Virage [41], WebSeer [98], etc.

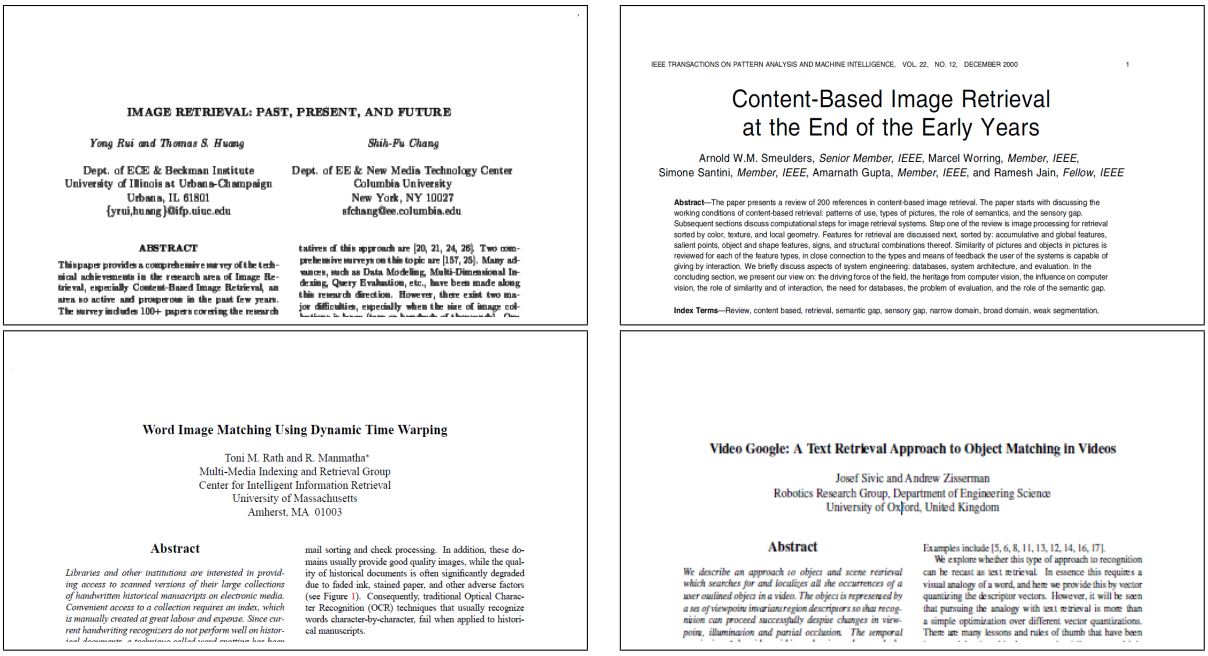


IMAGE RETRIEVAL: PAST, PRESENT, AND FUTURE

Yong Rui and Thomas S. Huang — Dept. of ECE & Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL 61801, {yrui,huang}@ifp.uiuc.edu

Shih-Fu Chang — Dept. of EE & New Media Technology Center, Columbia University, New York, NY 10027, sfchang@ee.columbia.edu

ABSTRACT

This paper provides a comprehensive survey of the technical achievements in the research area of Image Retrieval, especially Content-Based Image Retrieval, an area so active and prosperous in the past few years. The survey includes 100+ papers covering the research

tatives of this approach are [20, 21, 24, 26]. Two comprehensive surveys on this topic are [157, 25]. Many advances, such as Data Modeling, Multi-Dimensional Indexing, Query Evaluation, etc., have been made along this research direction. However, there exist two major difficulties, especially when the size of image col-

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 22, NO. 12, DECEMBER 2000 1

Content-Based Image Retrieval at the End of the Early Years

Arnold W.M. Smeulders, *Senior Member, IEEE*, Marcel Worring, *Member, IEEE*, Simone Santini, *Member, IEEE*, Amarnath Gupta, *Member, IEEE*, and Ramesh Jain, *Fellow, IEEE*

Abstract—The paper presents a review of 200 references in content-based image retrieval. The paper starts with discussing the working conditions of content-based retrieval: patterns of use, types of pictures, the role of semantics, and the sensory gap. Subsequent sections discuss computational steps for image retrieval systems. Step one of the review is image processing for retrieval sorted by color, texture, and local geometry. Features for retrieval are discussed next, sorted by: accumulative and global features, salient points, object and shape features, signs, and structural combinations thereof. Similarity of pictures and objects in pictures is reviewed for each of the feature types, in close connection to the types and means of feedback the user of the systems is capable of giving by interaction. We briefly discuss aspects of system engineering: databases, system architecture, and evaluation. In the concluding section, we present our view on: the driving force of the field, the heritage from computer vision, the influence on computer vision, the role of similarity and of interaction, the need for databases, the problem of evaluation, and the role of the semantic gap.

Index Terms—Review, content based, retrieval, semantic gap, sensory gap, narrow domain, broad domain, weak segmentation,

Word Image Matching Using Dynamic Time Warping

Toni M. Rath and R. Manmatha*
Multi-Media Indexing and Retrieval Group
Center for Intelligent Information Retrieval
University of Massachusetts
Amherst, MA 01003

Abstract

Libraries and other institutions are interested in providing access to scanned versions of their large collections of handwritten historical manuscripts on electronic media. Convenient access to a collection requires an index, which is manually created at great labour and expense. Since current handwriting recognizers do not perform well on histor-

mail sorting and check processing. In addition, these domains usually provide good quality images, while the quality of historical documents is often significantly degraded due to faded ink, stained paper, and other adverse factors (see Figure 1). Consequently, traditional Optical Character Recognition (OCR) techniques that usually recognize words character-by-character, fail when applied to historical manuscripts.

Video Google: A Text Retrieval Approach to Object Matching in Videos

Josef Sivic and Andrew Zisserman
Robotics Research Group, Department of Engineering Science
University of Oxford, United Kingdom

Abstract

We describe an approach to objects and scene retrieval which searches for and localizes all the occurrences of a user outlined object in a video. The object is represented by a set of viewpoint invariant region descriptors so that recognition can proceed successfully despite changes in viewpoint, illumination and partial occlusion. The temporal

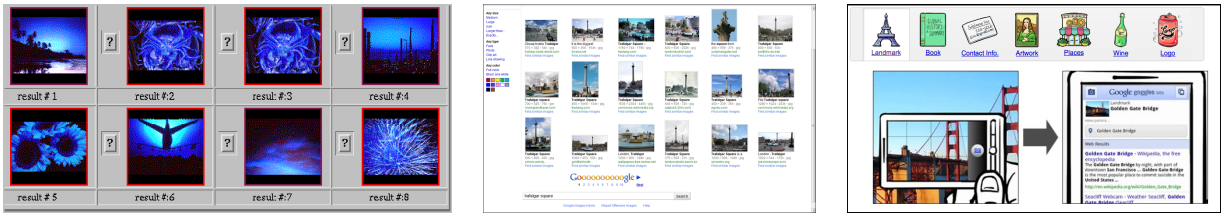
Examples include [5, 6, 8, 11, 13, 12, 14, 16, 17].

We explore whether this type of approach to recognition can be recast as text retrieval. In essence this requires a visual analogy of a word, and here we provide this by vector quantizing the descriptor vectors. However, it will be seen that pursuing the analogy with text retrieval is more than a simple optimization over different vector quantizations. There are many lessons and rules of thumb that have been

Figure 2.1 Some of the papers over the years that summarized and shaped the state-of-the-art of image and video retrieval. In order, the works presented are Rui *et al.* [193], Smeulders *et al.* [207], Rath & Manmatha [183] and Sivic & Zisserman [205].

One of the major limitation of early CBIR systems has been the difficulty to map user needs to low-level features. There are two aspects to this challenge [207]: i) sensory gap and ii) semantic gap. The loss of information of objects and scenes in the real world, when captured as an image, and consequently as features extracted from it, is called the "sensory gap". The "semantic gap", on the other hand, is the difficulty to extract information from visual data that is similar to the description of the user's need.

The features themselves had limited matching capability, for example, color features could identify the sky as being blue or grass as green, but would easily confuse an apple with a rose (an issue persistent in major search engines until very recently). The ability of low-level features to reliably represent content works well only for a *narrow domain* [207] of the images, where additional domain information could help identify or augment the features [82]. As a result, several CBIR systems are engineered to answer for a specific type of concept.



(a) (b) (c)

Figure 2.2 Image retrieval systems have come a long way from (a) depending on weak features such as color and texture [94, 145] to (b) practical large-scale systems using contextual text information [1] and (c) evolving into the more recent smartphone apps [19, 21] that provide on-the-go use of images as an entry point to information sources.

The second major drawback of CBIR systems is the enormous computation required to match the query with the database. This is typically $O(N \cdot d)$, N is the number of images in the database, d is the size of feature vector. The online computation of the query's nearest neighbors (NN) in the database, requires several minutes for a few thousand images. As the dataset size grows, CBIR systems take prohibitive amounts of time to answer each query. The computational complexity of exhaustive NN matching was overcome using efficient matching techniques such as B-Trees [48] (and other tree structures), Locality Sensitive Hashing [35], Hierarchical K-Means [163], KD-Tree [157] etc. The underlying principle in tree structured indexes is that, the feature or image space can be split into small groups of closely related points, in a hierarchical fashion. A function is generally evaluated at each hierarchy to identify the path to be taken for each data point. The height of these tree structures is typically $O(logN)$, which also gives the estimate for time in identifying the cluster for each point. By identifying the paths in the tree corresponding to the query, one could retrieve the relevant images quickly.

2.1.2 CBVR

A video can be treated as a sequence of images, allowing the use of any CBIR approach to build a retrieval system. However, it would be more efficient to identify *keyframes*,

and apply CBIR on this subset only. Thus, early work on CBVR revolved around temporal segmentation of the video into logical units such as shots and scenes [59, 108, 139]. A *shot-cut* is detected, whenever the camera shifts, or the scene being captured changes significantly. Shot-cut detection was performed using color histograms, motion vectors, etc. [127, 137]. A better temporal segmentation involved identifying clusters of shots that were semantically related, called the *scene*. Scene segments were obtained by clustering adjacent shots using visual features [216, 237] or dynamism [70, 181].

2.1.3 The Visual Words World

The global features used in classical CBIR did not work well in the presence of multiple objects or concepts within an image, and in presence of occlusions. Features extracted from regions obtained from image segmentation [221] or pixel-clustering [187, 202] were not reliable, as the segments did not directly correspond to semantic entities such as objects. To address this issue, several researchers have proposed the use of local features, where a feature is generated at several "interest points" in the image. The interest points are typically obtained by using a corner detector [198], maximally stable extremal regions [149], a difference of Gaussians [143], etc. Each point is described using features such as the patch of image around it, SIFT [143], histogram of oriented gradients [76, 143], and improvements thereof such as GLOH [151], SURF [47], DAISY [217], etc. With local features, each image typically generates several hundreds of features, resulting in an explosion in the number of features from a small image collection or from videos.

Towards efficiently matching large number of local features, the quantization of the feature space was proposed in [205]. Specifically, a large number of SIFT features are clustered using the K-means algorithm, each cluster centroid forming what is termed the *visual word*. Any given SIFT feature vector is assigned to its nearest cluster and given the name or number of this cluster. An image is then represented as a "bag-of-visual-

Figure 2.3 Example retrieval results from the Video Google system presented by Sivic & Zisserman [205], available at [15]. The query is given as an image region shown on the left. The SIFT features from this region are matched against those from the rest of the frames in the video to retrieve the best matches. This matching is tremendously sped-up using vector quantization of the features followed by a text-like indexing of the images.

words" (BoW), which is essentially a histogram of the visual word occurrences in the image. Such a representation enables an easy indexing of the visual data using popular text search engines, such as Lucene [12], Greenstone [22], etc. Given a query region in an image, the features in this region are identified and quantized to obtain the query-visual-words. Each query-visual-word is looked up in the index to identify the images that contain similar features. The image lists across the visual words are merged and ranked using the TF/IDF measure. Early results on the object retrieval task, as seen in Figure 2.3 showed tremendous promise over a challenging dataset of movie frames [203].

One of the shortcomings of the BoW approach is the loss of spatial information. Sivic & Zisserman [203] proposed the use of geometric verification, as a post-processing step to re-rank matches obtained from the visual word indexing. The spatial matching was further extended using a hierarchy of spatial segmentation of the retrieved regions [134]. Several improvements over the basic BoW approach have been proposed [168] to improve quantization quality [167, 169], speed [119], reducing memory load [99], re-ranking, etc. An alternative method of quantization, using a pyramid match kernel was proposed by Grauman [103], which was later extended to use a hashing based speedup. The bag-of-words approach has since, been a popular representation for object category recognition [165, 239].

2.1.4 CBxR: Summary

Inspite of major advances in CBxR approaches, the underlying mechanism assumes that query Q is of the same media as the collection W. This would imply the user needs to have atleast one instance of the target concept, which is often not the case; having the single instance might actually address the information need. More importantly, the user should be allowed flexibility to express the query, irrespective of the type of multimedia being indexed. For example, the user could ask "find me the music video for this audio clip", or "what is the news article corresponding to this picture", etc. With the advent of popular text search engines such as Google [3] and Bing [4], users are accustomed to posing queries in the form of *short text*.

Toward addressing textual queries for multimedia collections, the challenge is to generate a textual description for the multimedia. There are two broad categories of approaches that have been proposed to generate textual information for multimedia: i) Recognition-based and ii) Annotation-based.

2.2 Recognition Based Retrieval

A recognition-based solution can be understood as answering this question: "given a set of labels, which of these labels are active for the given visual data". A classical recognition problem is that of identifying the names of people in an image [241]. Given the wide applicability of face-recognition in security applications, this problem has received much attention, and reasonable progress has been made resulting in country-scale systems.

Szummer and Picard [213] and Serrano *et al.* [201] were among the early works toward classifying images into indoor/outdoor categories. Fleck *et al.* [93] and Wang *et al.* [226] have built systems to classify Web images as being objectionable or not, which were later extended in [95, 227] to other object categories. Features and classifiers have

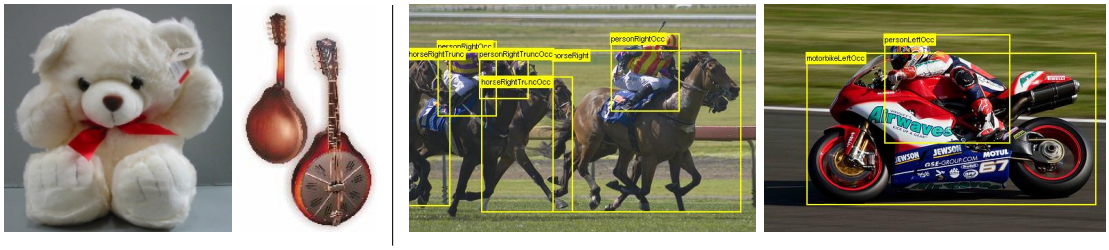


Figure 2.4 Benchmark datasets such as the (left) Caltech-256 [104] and (right) the PASCAL VOC [85] have been instrumental in concerted effort towards solving the visual recognition problem. The Caltech-256 dataset consisted mostly of images that contain only one kind of object from among 256 different classes, allowing for whole-image classification. The VOC dataset consisted of unconstrained images that contain multiple objects, requiring sliding-window like approaches. Though rapid progress has been made in the last decade, the results are far from user-acceptable accuracy or human-understandable semantics.

been studied extensively for scene recognition [164]. The BoW-like representation with strong discriminative classifiers such as Support Vector Machines (SVM) have been quite successful towards object detection and recognition [85, 171] over benchmark datasets such as those shown in Figure 2.4. The most successful object detection solution thus far, has been the use of deformable template models learnt with a latent-SVM [89].

More recently, Deep Belief Networks (DBN) [111], have shown promise by automatically learning the appearance of a concept using a stack of neural networks that progressively "compress" the visual data. The applicability and improved performance of DBNs on a variety of problems is yet to be evaluated.

2.2.1 Video Recognition

Recognition in videos could be treated as an application of image-level recognition over *keyframes* extracted from the video. This is the popular approach to labeling large video datasets such as the TRECVID challenge [206], example results are shown in Figure 2.5. However, treating videos as a stack of images, does not utilize additional sources of information present in the video such as, i) temporal and ii) auditory.

Figure 2.5 Video shot classification results over the TRECVID dataset [206], for the concepts (above) *crowd* and (below) *mountain*. Video classification involves classifying the *keyframes* for each video shot, in this case using a random forest classifier on an ensemble of features as reported in [170]. The TRECVID challenge is important as it i) raises the semantic level from objects to scenes and ii) the scale of video data is significantly larger than previous benchmark datasets.

The temporal information is typically used to recognize the actions and activities of people [34, 69], with a focus on security applications. One of the core issues in video recognition is to *track* people [96] and objects across the frames in the video. Some of the popular approaches to tracking include optical-flow estimation [44, 52, 234], Kalman filtering, successive detection [178, 179], kinematic models [57, 58], etc.

Activity recognition is typically built on top of spatio-temporal segments from videos [114], mostly by learning a Hidden Markov Model (HMMs) [91, 231]. Sign language recognition is an interesting subset of activity recognition, which was addressed using HMMs in [211] and modeled on Markov chains of discriminative visual features in [55].

Auditory information present in videos was used for scene segmentation and highlight generation [70, 79, 192]. Speech recognition has been used to annotate videos with text in the form of closed-captions [32]. Speech recognition, however, is not a solved problem especially in the presence of multiple or unknown speakers and background noise.

Another interesting generative approach for videos is based on the authoring paradigm. Snoek [209] proposed a framework, where multimedia documents are considered the result of a set of specific (mostly unknown) authoring steps. The authoring process is reversed by analyzing the multimedia and combining with common style conventions in film making. With this process, multimedia could be indexed at a semantic level. However, this is restricted to production quality videos and is thus not directly applicable to user generated content.

2.2.2 Recognition: Summary

Though tremendous progress has been made in image/video recognition, detailed semantic labeling is still a distant dream for automated solutions. Recognition systems are expensive to train as they require large amounts of training data and expensive to deploy due to the computational cost of complex classifier solutions. This makes recognition systems scale very slowly with new concepts and new datasets.

2.3 Image/Video Annotation

An alternative to recognition is the process of annotation, which can be loosely defined as: "Given the visual data V, what pieces of (textual) information is relevant for it?". The difference between recognition and annotation is: i) the set of labels is not fixed or known beforehand in annotation, and are generated on-the-fly, depending on the visual data, ii) the structure of the labeling is unrestricted in annotation, while recognition results in a binary or real-valued score for each label. Annotation is not expected to be thorough, in the sense, when visual data is not annotated with a label, it *does not* mean that it is absent.

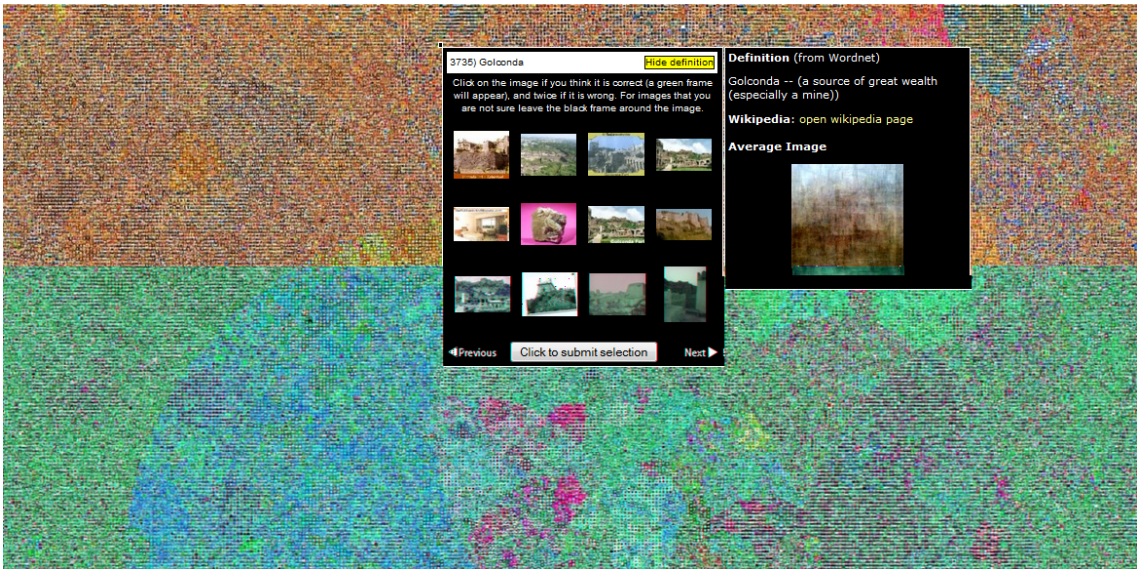


Figure 2.6 An aggregation of 80 Million images crawled against 75K keywords [218] from the WordNet hierarchy using popular search engines such as Google, Flickr, etc. Example images are shown for the category *Golconda*, a popular tourist site in Hyderabad, India. The results from a web search engine typically consist of images that have either high similarity with the query or are almost irrelevant. A few specific keywords exhibit a strong structure among the relevant images, so much so that the average image is actually recognizable.

2.3.1 Manual Annotation

A classical annotation system, that continues to be used today is that of manual annotation of stock photographs such as Getty Images [20], by a set of trained personnel. The labels are placed in a hierarchy, which allows for easy browsing and searching, but labels can be added and deleted without loss of performance or applicability. Manual annotation is the foundation of search in photography communities such as Flickr [18], 500px [11], etc., where the labels are quite uncontrolled, sometimes descriptive, and depend heavily on the personal context of the photographer. Large video datasets such as movies and TV shows on Netflix [28], Hulu [23], etc. are also labeled manually. YouTube [32] search depends on titles and tags assigned to videos by the uploader.

Another interesting annotation scheme is the use of image search engines such as Google [3] to collect images for a given keyword. Using this method, 80 million images [218] were gathered *pre-annotated* against 75K keywords from the WordNet [153] hierarchy. Sample images collected this way for an example category are shown in Figure 2.6. Image search engines inevitably result in images that are not particularly relevant for the query. Filtering schemes have been proposed using a mixture of textual and image features [49, 199, 225] to yield better datasets from such noisy retrieval results.

There is a more recent trend to label images and videos by crowdworkers [210, 224, 230], yielding some of the commonly used benchmark datasets such as LabelMe [195], PASCAL VOC [85], ImageNet [78], etc. The difficulty of using crowds for large scale annotation is due to the unreliability of crowdworker performance, the subjectivity of individuals and the high cost of labeling. Consequently, automated solutions are still preferred over manual annotation.

2.3.2 Syntactic Annotation

One of the popular annotation schemes for images and videos is to identify the *text* present in the data. Overlaid text recognition has been successfully used for scene detection and annotation of news videos [197], sports videos [40], etc. Scene text recognition in images is an active research area, with several recent works attempting to address this problem [154, 228, 229].

Annotation of videos based on the people in the video has also been quite popular. For example, Sivic *et al.* [204] index video shots from a feature film, using the facial appearance of the characters in the scene. The same group has annotated sitcom episodes by the names of the characters, using the script and subtitles of the series [83] (without any facial matching). Similarly, faces in news photographs were labeled using a named entity

recognizer over associated captions [50]. Yang and Hauptmann [236] have presented an annotation scheme for news stories based on the locations of events.

2.3.3 Annotation Propagation

Several works fall under the category of annotation propagation: "given a set of images with labels, find the suitable labels for new images". This approach is different from the "recognition" paradigm, as the labels are not explicitly mapped to the concept in the visual data. The annotation of the image is typically modeled as learning the "co-occurrence" between the keywords and images or the keywords and image regions. Barnard and Forsyth [46] present a word-image mapping using a hierarchical structure to cluster features and keywords at various semantic levels.

The region-word co-occurrence model was first presented in Duygulu *et al.* [81], where the annotation problem is posed as one of translation between discrete features and the keywords. The translational model learnt from the data is used to estimate the probability of a keyword given the observed image feature, thereby annotating each image region. A more complex generative model conditions the co-occurrence of features and keywords is conditioned on a set of hidden parameters. The model parameters for each class are learnt from the data, using the EM algorithm. New images are annotated by estimating the maximum likelihood of the class parameters that would have generated the image features. Blei and Jordan [53] proposed a generative method for word and image features using the latent Dirichlet allocation model. Similar latent space models were also used by Monay and Gatica-Perez [156]. Other generative models such as cross media relevance model [120], continuous relevance model [132], multiple Bernoulli relevance model [90], etc., have been used to model the joint probability distributions of image features and keywords. Jeon *et al.* [121] have addressed annotation in the presence of noisy image descriptions. This work is also notable for having annotated more than 25K news images with over 4000

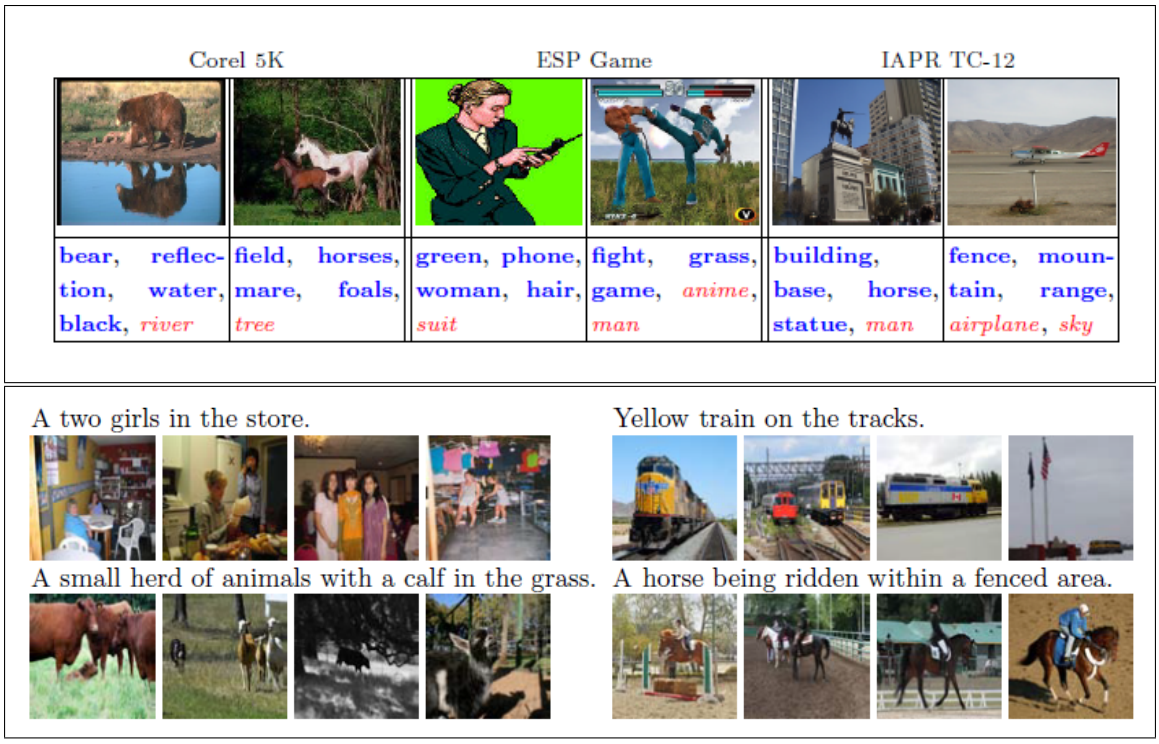


Figure 2.7 (Above) Example results from an annotation propagation framework presented in [219]. Results over multiple datasets such as Corel, ESP and TC-12 are shown, the labels for the images are fairly accurate. The blue colored labels are the ones present in the groundtruth, while the red colored labels are additional relevant keywords for the images. Most annotation propagation approaches still work at the *keyword* level rather than at semantic sentence level. (Below) Example images that match a given sentence in the "meaning" space of keywords [87]. Sentences were *aligned* with the images by identifying the keywords from both and matching them.

keywords. The annotations performance in generative modeling approaches is affected by the quality of training data and the modeling assumptions.

Recently, discriminative models have been proposed towards image annotation. Nearest Neighbor (NN) techniques [105, 146, 219] have surprisingly good performance in label propagation, perhaps because of the availability and computability of large amounts of labeled data. The nearest neighbor (NN) classifier with a learnt distance metric, was used to propagate tags in [105]. Another method that used NN classifiers involves a two step classification method that matches images to keywords in one step, and then images with

other similar images, called 2PKNN in [219]. Support Vector Machines (SVMs) have shown improved performance, especially over confusing labels [220].

Further, the advantages of discriminative and generative classifiers could be combined with hybrid approaches such as Li *et al.* [140] and Carneiro *et al.* [64]. Li *et al.* [140] annotates images using the two techniques separately in a two stage process, while Carneiro *et al.* [64] propose a supervised multi-class labeling formulation that combines the advantages of discriminative and generative techniques. Most annotation approaches, however, require large amount of training data to satisfactorily learn the keyword models. Another limitation of annotation propagation approaches is that the labeling granularity is still at the *keyword* level.

2.3.4 Annotation by Alignment

The category of *Annotation Alignment* addresses the goal of "given visual data and corresponding text, identify the appropriate mapping between the two". This is similar to works like Berg *et al.* [50] that labels people in the photographs with names identified from textual captions. Another interesting annotation alignment was proposed to label images with sentences [87]. Given a set of sentences, the *meaning* of the sentences is identified by a triplet of keywords. Similar keywords are generated from the images too. For each image, the sentences with the closest meaning could be identified and vice versa. Example images mapped to their sentences are shown in Figure 2.7.

Annotation of videos by alignment of closed-captions with movie scripts was first shown in [83], and later applied to a larger collection by [130]. The resulting annotations are themselves useful for retrieval [131], the labels from which were then used to train a face/activity recognizer. As an extension, solutions have been proposed around aligning closed captions with sign language in news broadcasts [60, 61, 86], such as the example shown in Figure 2.8. Further, the causal relationships across individual actions were learnt

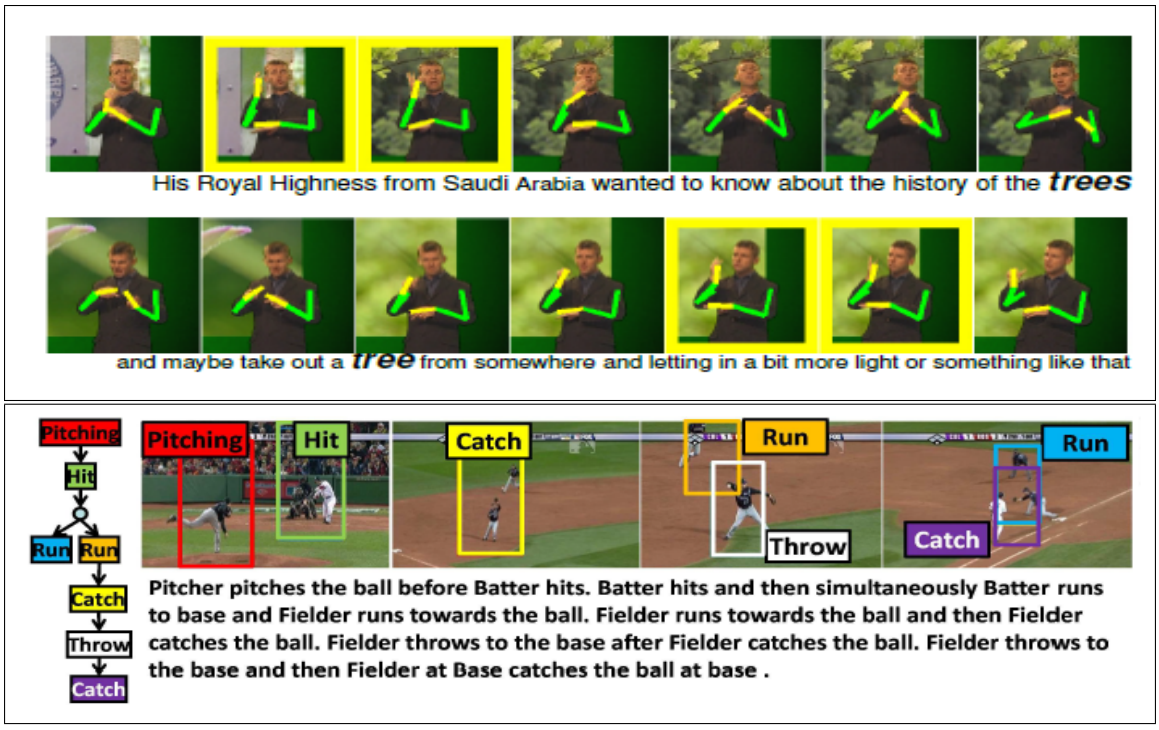


Figure 2.8 (Above) Closed captions were used as a source of weak supervision to learn the sign language vocabulary [60,61]. In this case, the British sign language symbol for "trees" is shown. (Below) Annotation of a baseball video by learning an action model as well as a storyline model, from a training data of videos with captions [106]. The video is analyzed to generate the graph shown on the right based on activity recognition algorithms. The descriptive text is then generated using this graph.

to generate a storyline for baseball videos [106, 107], such as the description shown in Figure 2.8.

2.3.5 Annotation: Summary

Image and video annotation have expanded the boundaries of labeling multimedia documents from a restricted set of classes (as in recognition) to a more generic label-set. Semantic information is preserved and propagated across labeled and unlabeled datasets, through an effective use of co-occurrence statistics across visual and textual domains. However, there is much scope to improve i) the granularity of the visual data at which

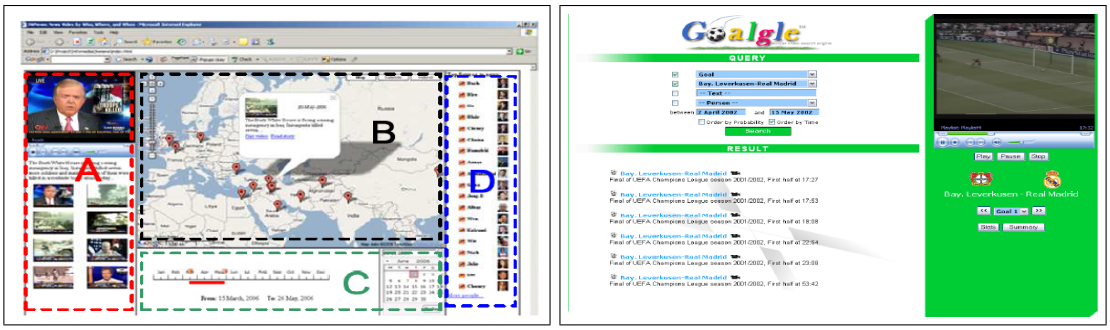


Figure 2.9 Prototype retrieval systems developed for specific datasets. (Left) The News retrieval system built by Yang & Hauptmann [236], the different panes of the interface are marked as: A) Story pane, B) Map interface for different news stories, C) Timeline and D) Frequent person viewer. (Right) The Goalgle system for soccer video retrieval developed by Snoek [209].

annotation is performed and ii) the semantic depth of labels and descriptions. Further, annotation approaches too require sufficient amount of labeled training data.

2.4 Multimedia Retrieval Systems

Unlike several domains where solutions are typically limited to laboratory settings, multimedia retrieval systems need to satisfy varied demands from end-users. There is considerable focus on easy-to-use interfaces, an interesting example being the 3WNews system from CMU, shown in Figure 2.9, which allows news video browsing based on geographical locations [235]. This system was built by performing named-entity recognition over the speech transcripts to identify locations (and people) discussed in a story, such as the one shown in Figure 2.9. Similarly, the retrieval system built by [209] (see Figure 2.9) over soccer videos allows the user to search for events such as goals, cards, substitutions, etc.

The mode of querying the retrieval systems is an important design aspect. Typically, queries are posed as *short text* that includes a few keywords, similar to text search engines. An example retrieval system over handwritten letters by George Washington [184] (shown in Figure 2.10), can retrieve lines containing the given query keyword. More-

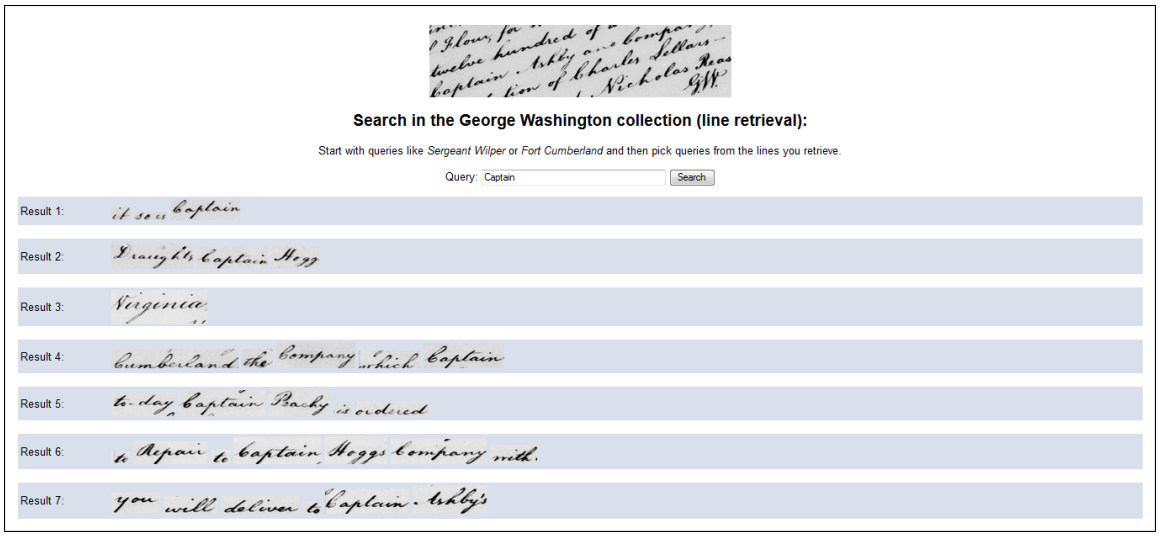


Figure 2.10 Line retrieval demo [185] over images of handwritten letters by George Washington. The underlying retrieval system is based on the concept of *Word Spotting* [147], where matching occurs in the image space. Text retrieval is enabled by manual labeling over clusters instead of individual word-images. This retrieval system was the first available solution to search across unconstrained offline handwritten images.

over, interesting applications could be built if different modalities could be handled for a given document collection. In case of image databases, novel queries such as by sketching have been proposed [200]. Another interesting query-modality is the use of the stickman model [56] of a person to query for human pose [118].

A challenging query modality is a stream of images from a webcam, such as the document retrieval system shown in Figure 2.11. As each image in the stream is considered a query, the retrieval systems needs to be optimized to process the query and match it across the database in real-time. This is achieved using a technique called locally likely arrangement hashing (LLAH), over the locations of the words in the document. Applications of such a retrieval system include seamless access of digital documents through printed documents [124, 141, 142], tracking user reading behavior [129], etc.

Similar query-by-example systems have found their way into smartphone applications, such as Google Goggles [21], Amazon Flow [19], etc. The Google Goggles system is shown

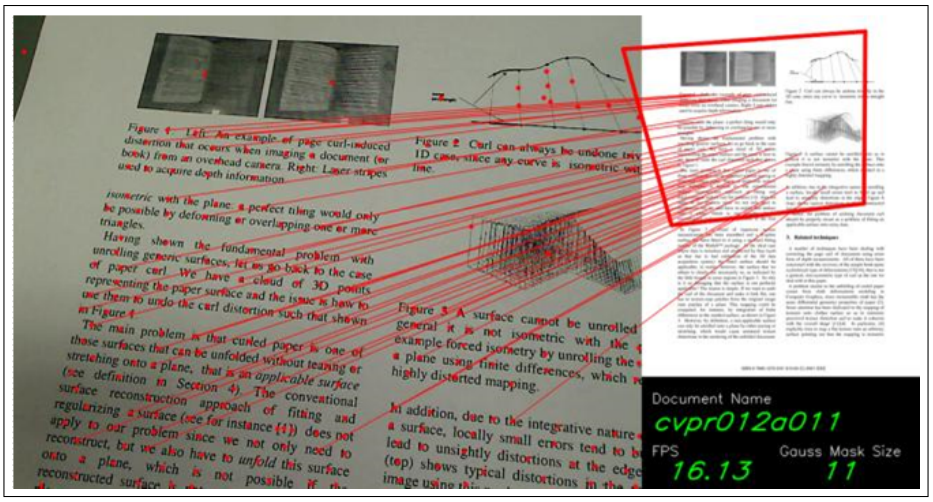


Figure 2.11 Document retrieval at page-level [159], where an image from a webcam is used as a query to find the relevant document from the database, in real time. Document images are matched robustly against geometric distortions based on the arrangement of words in the page. The matching is sped-up using an approach called locally likely arrangement hashing. The method has been recently [215] scaled up to retrieve from a large 10 Million page dataset.

in Figure 2.2 (c) , where the smartphone camera is used to capture a picture of a limited set of objects such as a landmark, a painting, etc. The image is analyzed on the device and queried against a cloud-based index of annotated images. The matching image is used to extract additional information regarding the object of interest.

Towards addressing semantic-queries, query expansion [72] is an interesting way to generate multiple relevant queries that could cover the semantic meaning of the query, and obtain multiple retrieval results from the document collection. Another mechanism to disambiguate the query semantics is through relevance feedback [194, 196]. Personalization has also become an important aspect of multimedia systems. Zhang *et al.* [240], for example, have applied relevance feedback mechanisms to video retrieval systems to refine search results in annotated sports videos.

Another challenge with IR systems is a standard procedure to evaluate various retrieval approaches. Towards this end, controlled datasets, queries and query modalities have

been put forth with competitions such as TREC and its video counterpart, TRECVID [206]. The evaluations rely on manual groundtruth, which is feasible for these relatively small datasets.

2.4.1 Systems: Summary

Several multimedia retrieval prototypes demonstrate the progress of the field towards "usable" systems that could address real-world needs. However, these systems tend to be restricted in the domain of the multimedia and in the scale of the data that is indexed. Unlike text search engines (e.g. Lucene [12]), there are few instances where MR systems are open-sourced to the community, towards building customized retrieval systems over novel multimedia collections. A single retrieval system that seamlessly integrates across several multimedia domains, remains a distant goal.

Chapter 3

ScriptAlign: Alignment of Movies with Scripts

3.1 Introduction

Movies and Television shows constitute a major portion of entertainment for the masses. Statistics show that in a 65-year life span, as much as 9 years are spent watching Movies or TV. However, each person can only watch a small fraction of the entire video material available. More video material is being generated by TV channels and Movie studios, than can be consumed by people. This calls for an intelligent browsing, search and retrieval mechanism for movies, such that users can watch only the video material (and parts of) which are interesting to them. This can be addressed to certain extent by using metadata information such as genre, plot keywords, theme, actors, director etc. Online databases such as IMDB [24], provide detailed metadata for movies and TV shows. However, metadata alone is not sufficient for all user preferences, such as to retrieve specific actions, expressions, favorite scenes, memorable action sequences, etc.

The major technical challenge in enabling such retrieval, is the labeling of videos with semantic information. Given a video, it is hard for computers to automatically infer meaning from the data. This would involve comprehensively solving the computer vision problems. In such a situation, we need to look for clever alternatives to bootstrap the recogni-

tion process. Consequently, we infer that the problem can be sufficiently addressed if we could obtain certain level of meaningful annotations of the video.

The ideal scenario is the availability of manual annotations for the multimedia. While manual annotation is a costly procedure, there are many domains for which manual annotations are readily available. For example, movies are described by their scripts, news videos are associated with online news articles and sports broadcasts are accompanied by webcast commentaries. This parallel information is in the *textual* form, and at a semantic level similar to user queries or requirements. Scripts are written with sufficient detail regarding the location, setting, mood, actions, etc. Such information is valuable in annotation of videos for retrieval. In this Chapter, we concentrate on labeling movie videos with detailed textual information obtained at no additional cost. Scripts for a large number of movies and TV shows are available at IMSDb [26], Script-o-Rama [27], TwizTV [29], etc.

The two sources of information, namely the video and the text of the script, are depicted in Figure 3.1. Given the corresponding script for a movie, the problem is to infer a synchronization between the text of the script and the visuals of the movie. The typical method of aligning a script (or transcript) with TV or movie videos is dynamic time warping with the subtitles/closed-captions, as introduced by Everingham *et al*. [83]. Scripts aligned with videos offer a number of possibilities: they can provide supervisory information for identifying characters [83] or learning actions [130]; they enable a scene level organization of the video material [73]; and they enable text-based retrieval and search [173].

Previous work such as [83, 130] that exploits closed-captions and subtitles are limited to videos which have such information available. Hence they are not applicable to videos with little or no spoken dialogue, and to situations where subtitles are unavailable, which is common in many non-European language films (especially Indian movies/TV) or silent films. We address this problem in this Chapter, where our objective is the visual alignment between TV/movie videos and their scripts, *without* using subtitles. Achieving such an

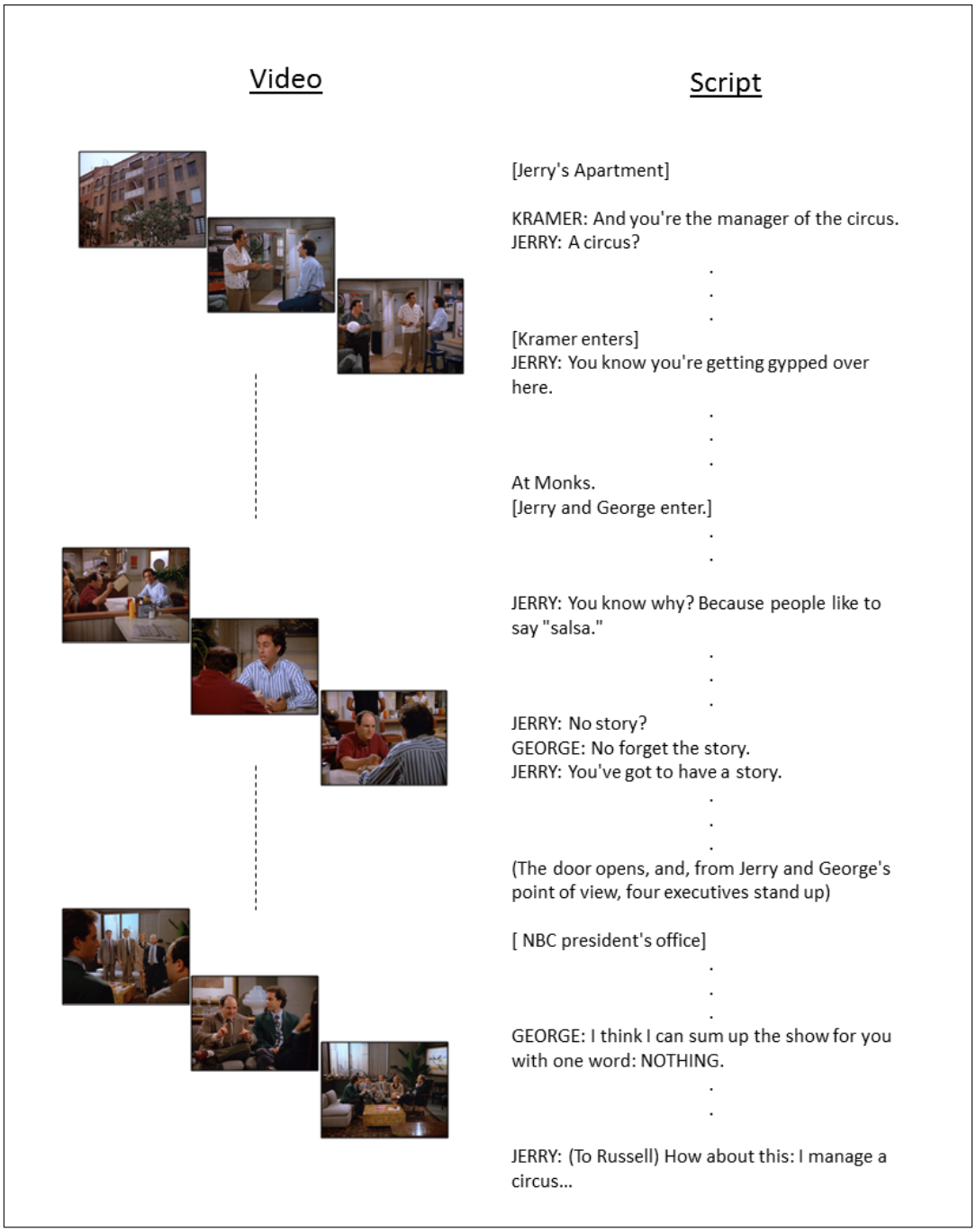


Figure 3.1 In this Chapter, we propose to annotate videos using the information from its *script*. In this example, we see a few scenes from the episode *The Pitch* of the TV show *Seinfeld*, with the relevant script [30] alongside. The multimedia annotation problem is now posed as finding the correspondences between the sentences of the script to the shot of the video. As a result, we could annotate videos with semantic information from the script, without explicitly recognizing these concepts from visual data alone.

alignment increases the scope and applicability of script-based approaches. Therefore, it considerably increases the extent of video material that is available for training visual classifiers, and is suitable for text based search.

The challenge, however, is the comparison and matching of visual and audio information with the script descriptions of the video. The description may be quite generic (a street, an office) or quite specific (Jerry's apartment), and may involve objects and actions for which visual recognition is not yet mature. Our approach is to combine several cues, both visual and audio, which by themselves are not quite reliable, but when combined provide sufficiently strong constraints for a full alignment.

We pose the problem as one of multi-state labeling of a sequence of shots, where the states for each shot correspond to the sentences of the script. In other words, we would like to find the appropriate assignment of each *sentence* of the script to its corresponding *shot* in the video. This would result in sentence-level alignment between the script and the video. The assumption here is that the sentences are *atomic*, i.e., they belong entirely to one shot and not spread across multiple shots. This assumption mostly holds true as a sentence of the dialogue is generally spoken within one shot. The descriptions of actors' action/expression are typically localised to a shot as well.

In order to reduce the ambiguity we explore segmenting the video into scenes associated with locations.

3.2 Data and Performance Measure

We use episodes from the popular TV situation comedy *Seinfeld*, to evaluate our proposed approach. Our dataset consists of episodes from Season 4 of Seinfeld: *The Pitch and The Ticket* (training data), *The Contest* and *The Pick* (test data). We also apply similar techniques on a smaller collection of Charlie Chaplin and Indian movies. The Charlie Chaplin videos are excerpts from *The Gold Rush* and *City Lights*, and the Indian movie in

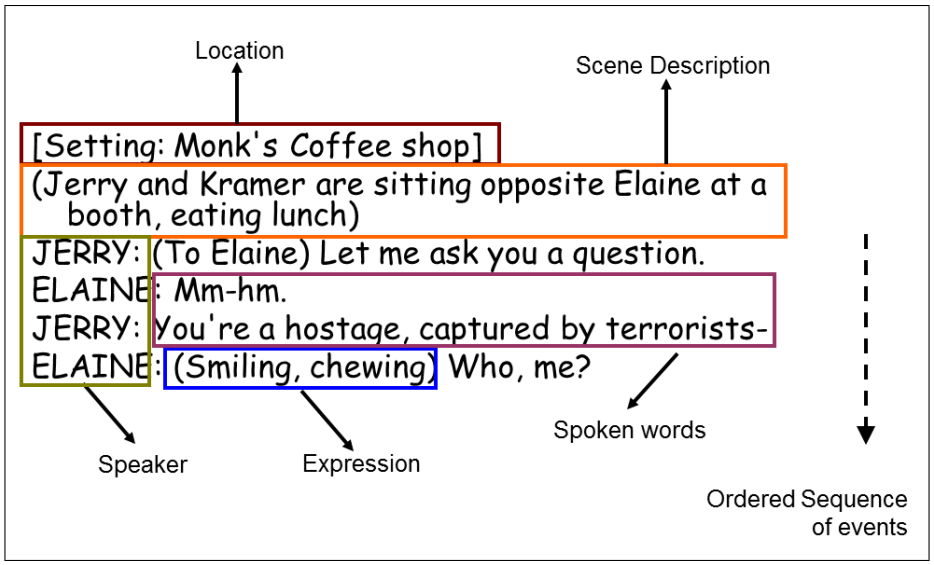


Figure 3.2 An excerpt from a script for an example Seinfeld episode. The various types of information present in the script, that would enable better understanding of the events in the video are highlighted.

consideration is the Bengali film *Agantuk*. The videos are divided into shots by computing the difference between colour histograms of consecutive frames. Whenever this difference is greater than a certain threshold, a shot-cut is detected. A typical 20 minute episode of Seinfeld has around 310 shots.

Scripts for the Charlie Chaplin movies were obtained from [13], for Agantuk from [186] and for the Seinfeld shows from [30]. A portion of the script from the Seinfeld episode *The Contest* is shown in Figure 3.2, along with the various aspects of information available in it that could be exploited in the annotation process. A typical script specifies the location of the scene (given as "Setting"), along with a brief description of the scene. The rest of the script has two distinct elements. The first is the detail about who is speaking and what, the other is a description of one or more of action and expressions of the characters.

Performance measure: The problem of alignment is now defined as, *given a video and the script for the events occurring in the video, assign each segment of text to the appropriate segment of the video*. The segment of the text is chosen to be a sentence, and that of the

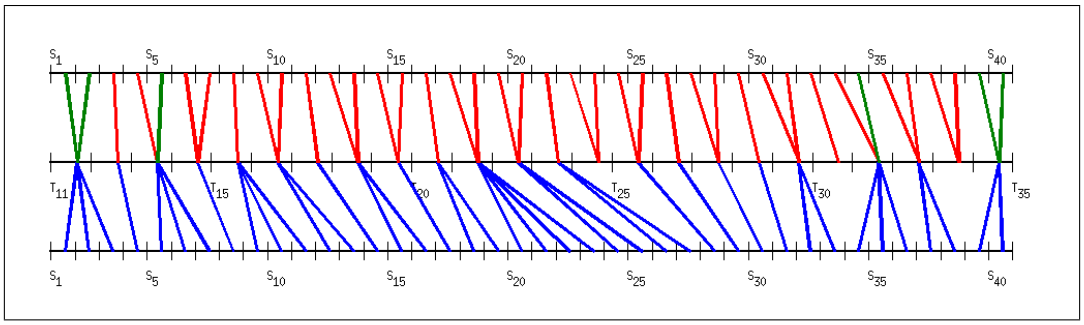


Figure 3.3 Example of a naive alignment scheme, shown over the first 40 sentences from the episode *The Contest*. S_s corresponds to sentences and T_t to shots. The blue lines correspond to the groundtruth alignment. The green and red lines indicate the correct/incorrect correspondences from naive alignment. It can be seen that errors at the beginning drastically affect the rest of the alignment.

video to be a shot. We shall call the spoken sentences $S_s, s \in [1, N_S]$, and localised descriptions as $D_d, d \in [1, N_D]$, where N_S, N_D are respectively the number of such sentences. Thus, for each sentence in $S_s \cup D_d$, the alignment tries to identify the right shot $T_t, t \in [1, N_T]$. The performance of the alignment is evaluated against manually ground truthed sentence-shot correspondences (Figure 3.3), the accuracy given as the number of sentences from S $\cup$ D assigned to the right shot. We shall denote by S^t, D^t, the sentences aligned with the shot t. The groundtruth is represented as S^t_G, D^t_G. The performance measure is defined as

$$\rho = \frac{\sum_t |S^t \cap S^t_G| + \sum_t |D^t \cap D^t_G|}{N_S + N_D} \tag{3.1}$$

If every sentence is aligned correctly, the intersection between the alignment and the groundtruth, summed over all shots yields a numerator equal to the total number of sentences. The value of ρ in this case would be one. Whenever the sentences are assigned to the wrong shot, the numerator decreases, and so does the ρ. We also define ρ_k, as the performance measure that allows errors in assignment upto k shots per sentence.

Figure 3.4 Example shots from the location categories of Jerry's Apartment (above) and Monk's Cafe (below). The shot on the left is called a "stock shot", which is a leading shot to the scene at a given location. The same location could have multiple distinct "views" which makes it harder to build a location recognizer.

3.3 Recognizing Visual-Audio Aspects

For a correct alignment, each sentence should compete for the right shot to fall into. The voting of a shot should depend on common features that can be extracted from both the sentence and the shot. For example, let us suppose that we know from the script that a sentence was spoken by Kramer while at the Monk's Cafe. Such a sentence would more likely occur at a shot known to belong to that location, in which Kramer can be visually recognised. Additional clues from the speech domain,would provide further evidence in support of such an assignment. Towards this end, we extract three clues from each shot/sentence: $< Location, Face, Speech >$. We apply state-of-the-art techniques for each of these modules, and the results on their performance are reported in the following Section.

3.3.1 Location Recognition

Television series are characterised by repetitive locations and recurring characters. In the case of Seinfeld, "Jerry's Apartment" and "Monk's Cafe" are the locations for a large number of the scenes. The setting of these locations remains relatively similar throughout the series, making them good candidates for scene recognition. In the case of sitcoms, each scene in the video is preceded by a *stock-shot* of the location from an exterior view. Example stock shots are shown in Figure 3.4. The recognition of this stock-shot reliably identifies the starting shot for that location. The stock-shot varies in viewpoint, illumination and scale, across different occurrences. SIFT [143] features have been proven to handle these variations robustly. We approach this problem as a near-duplicate image matching, given a set of stock-shot exemplars. Exemplars for stock-shot are identified from the training data.

The SIFT features are vector quantized to K clusters [205], and each feature point is represented by its closest cluster center. A shot is then represented as a histogram of the clusters corresponding to each feature point in it. Since the spatial layout of the ROIs is ignored, the representation is said to be a "bag" of visual words (BoW). The BoW for each shot is compared by the L_1-Norm with the BoW representation of the exemplars for the stock-shots. If the closest exemplar is less than a small threshold, the shot is classified to the particular scene category. By this method, we are able to reliably identify the beginning of scenes whose location is either Jerry's apartment or Monk's Cafe. As depicted in the second row of Figure 3.5, the stock shots have been correctly identified for the episode *The Contest*.

Given the beginning of these scenes, the next step is to determine their extent. The goal here is to classify the subsequent shots as belonging to that location or not. This is a multi-class problem: the shot can belong to Jerry's apartment or Monk's cafe or 'other' (which covers all other locations). The classification proceeds in two steps: first individual

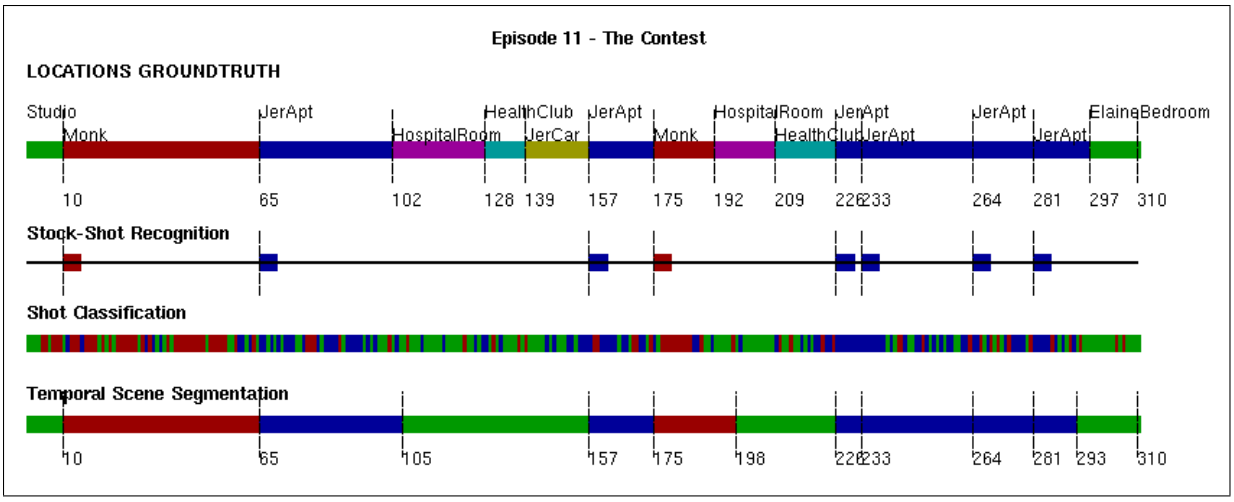


Figure 3.5 Location recognition pipeline. Stock shots (row 2) are recognised to identify the beginning of Jerry's Apartment and Monk's Cafe. Shots are then classified using an SVM (row 3) into one of Jerry's Apartment (blue), Monk's Cafe (red) and Other (green). Temporal scene segmentation is performed by combining the stock-shot and classifier information with temporal smoothing resulting in the labeling shown in the bottom row. By comparison with the groundtruth shown above, one can notice the high accuracy of the location recognition method.

shots are classified as belonging to one of the location classes; then a sliding window (on shots) is used to determine the scene boundary.

In detail, I-frames are extracted from each shot, over which Hessian-Affine [152] and SIFT [143] interest regions and are obtained. These interest regions are represented using the SIFT [143] feature vector. Two classifiers are learnt for location recognition using these features. The first classifier is based on exemplar-matching. A set of exemplars are provided for each location from the training data. The exemplars are chosen to represent different viewpoints of the given location. A given shot is classified by matching its i-frames with the given exemplars. The similarity score is computed as the number of matching keypoints between the two frames, according to one of the following matching techniques:

1. M1: Nearest-Neighbour(NN) keypoint match, refined with a threshold on the nearest-neighbor's distance.

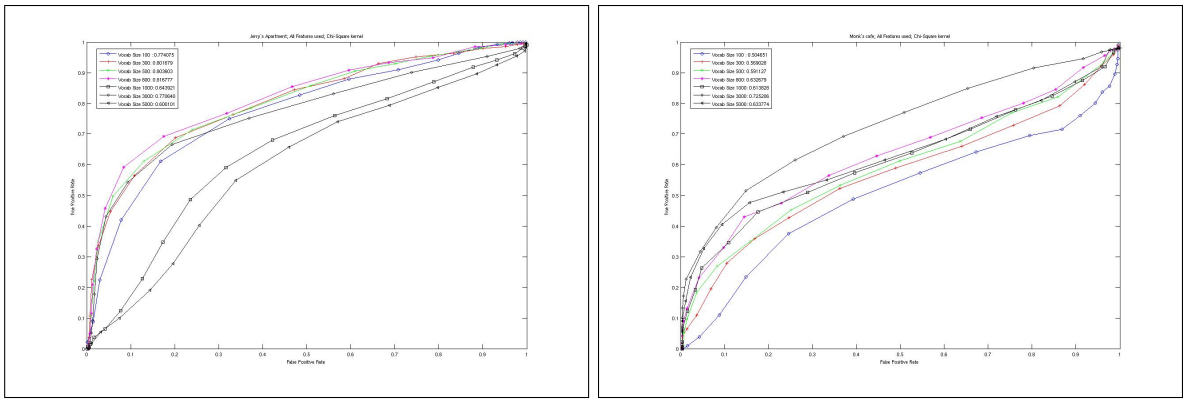


Figure 3.6 Precision-Recall curve of location recognition of (left) Jerry's Apartment and right) Monk's Cafe across various vocabulary sizes.

2. M2: Thresholded NN dist + Second NN test [143]. The second NN test rejects a keypoint if the ratio of second NN distance over first NN distance is less than a certain threshold. This would mean the first NN match is not a sufficiently strong match.

3. M3: Thresholded NN dist + Second NN test + Spatial consistency.

The "Second NN" test uses the ratio of distance between the first-NN and the second-NN. Spurious NN matches typically have a small Second-NN ratio, which can be filtered with a suitable threshold. Following this, spatial consistency [205] evaluates the similarity in the surrounding region of the matched feature points. Assuming affine transformations, the features present within a small neighbourhood will remain neighbours in the target image. The spatial consistency is measured by computing the overlap in the NN-sets of the matching feature points in the two images. Matching features that do not have the "support" of the neighbourhood are discarded.

Further, we consider a second classifier, which is an SVM learnt over Bag-of-Words features. A Kernel-SVM classifier is trained for each class, with the χ^2 kernel distance between

(a) (b) (c)

Figure 3.7 A depiction of masking the information about the people present in the video, in order to extract features from the location alone. (a) Original image, (b) Image with upperbody masked and (c) Image with the entire person masked. The location of a person is obtained using the Upper-Body detector of Ferrari *et al*. [92]. ROIs from the image are filtered using these masks to obtain the feature representation for location recognition.

two shots p, q given as

$$K(p,q) = e^{-\alpha\chi^2(p,q)} \qquad where, \quad \chi^2(p,q) = \sum_{i=1}^{N} \frac{(p_i - q_i)^2}{p_i + q_i} \tag{3.2}$$

Here, the parameter α is set to be the average χ^2 distance between training shots. The SVM-light package was used for building the classifier.

It was observed that many of the regions-of-interest (ROI) fall on people present in the frame. Since we are interested in learning the location (background), we would like to remove features over people (foreground). A HoG feature based upperbody human detector, provided by Ferrari *et al*. [92], was used to detect humans in the videos. Based on these detections an upperbody mask was placed on the image, as shown in Figure 3.7 (b), and all ROIs falling within the mask were filtered out. This feature set is referred to as *UbRem*. A full-body mask is constructed by extending the upperbody mask down to the height of the image (Figure 3.7 (c)), assuming people are always shown upright. Features falling within this fullbody mask were filtered, giving a set of features referred to as *FbRem*. The full set of features is referred to as *FullFeat*.

Method	FullFeat	UbRem	FbRem
M1	47%	48%	48%
M2	53%	55%	54%
M3	58%	54%	49%
Vocab. Size	FullFeat	UbRem	FbRem
100	59%	47%	47%
300	62%	54%	54%
500	66%	56%	55%
800	67%	50%	49%
1000	52%	53%	49%
3000	55%	53%	51%
5000	49%	51%	48%

Table 3.1 Location recognition performance of (above) NN-classifier across methods and (below) Kernel-SVM classifier, across various vocabulary sizes and feature sets.

In both cases of classifiers, the class with the greater score is assigned to the shot, provided the score is greater than a reasonable value (chosen empirically). The evaluation of the visual vocabulary sizes is shown in Figure 3.6. The classification results are provided in Table 3.1. The best performance for the two classes was about 67% using the Kernel-SVM over BoW. The shot classification results for an example video are shown in the third row of Figure 3.5. The lack of success of excising features on people, is possibly due to the camera focusing on the people, thereby defocusing the background. Also, removing features over people results in the loss of some features from the background.

Temporal recognition refinement. The location recognition accuracy is improved by using the property that the location does not change with every shot, but only across scenes. Errors in shot classifiers can be corrected with temporal filtering. The beginning of Jerry's Apartment and Monk's Cafe are obtained from the stock-shot recognition. We now need to identify where such scenes end and the ones from *Other* category begin. To identify the *scene-boundaries* between location L_l and L_{l+1}, a sliding window of W shots is

moved across the video. At each shot s, an objective function is computed as

$$E_s = \alpha \cdot \Big\{ \sum_{i=s-W/2}^{s} (1 - P(L_l|i)) + \sum_{i=s}^{s+W/2} (1 - P(L_{l+1}|i)) \Big\} + (1-\alpha) \cdot \{Dist(s, s+1)\} \quad (3.3)$$

The first term in E_s evaluates the cost of assigning the shots $[s - W/2, s]$ to the location L_1, and those from $[s, W]$ to L_2. $P(L_l|i)$ is the score obtained from the shot classifier. The second term penalizes scene-boundaries at similar looking adjacent shots, thereby enforcing a piece-wise smoothness constraint on scene segmentation. The $Dist(s, s+1)$ is obtained as the inverse of the L1-Norm difference between the shot's BoWs. The scene-boundary B_l is obtained as the shot-boundary at which the objective function is minimum. From the training data, we infer the best performing α_1 and α_2 to be 0.3 and 0.7 respectively, using a window size of eight shots.

The scene boundary between adjacent scenes, both belonging to *Other* is estimated from the number of sentences spoken in the given location. It is assumed that the scene duration is proportional to the length of the speech. This estimate is refined by assigning the scene-boundary to the closest shot with no faces or speech detected (see Sections 3.3.2 and 3.3.3). Such a shot would typically represent a change in the scene. The final location recognition accuracy is measured as the number of shots assigned to the correct location. The accuracy for the training and test data is 96%. An example result for scene segmentation is shown in Figure 3.5.

3.3.2 Face Recognition

Seinfeld consists of four main characters, namely Jerry, George, Elaine and Kramer. By recognizing these characters, a large percentage of faces can be labeled in the video. We use the face detection/tracking/recognition pipeline of Everingham *et al*. [83], which is depicted in Figure 3.8. Faces are first detected in each frame using the Viola-Jones detector [222]. While this detector works robustly, it does output a large number of false

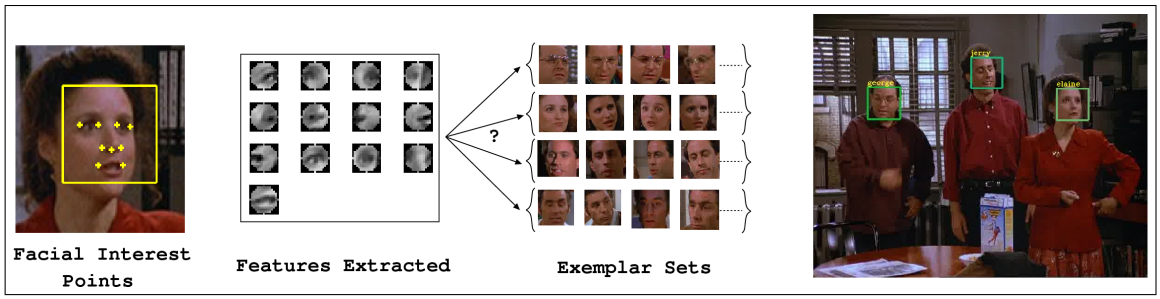


Figure 3.8 The face recognition pipeline. SIFT-like features are extracted from the corners of the eyes, lips and nose. These features are matched against those from an exemplar set for each character. The matching scores are accumulated over the tracked faces, and each track is classified as a whole. Sample recognition result is shown on the right.

Figure 3.9 Examples of errors in the face detection module. TV logos, plants, clothing and other textures are typically wrongly called faces by the Viola-Jones detector. A large percentage of these false detections are removed by performing skin-detection.

positives, especially on clothing and similar textured regions, some examples are shown in Figure 3.9. To remove such false detections, we add a step of skin pixel detection by learning the color distribution for the faces that are typically found in the considered video material. The threshold for skin/non-skin classification is chosen such that 75% of the false detections are removed, while retaining about 95% of true detections. This removes typical false detections that occur over windows, TV channel logos, clothing etc. Skin-filtered faces are tracked through a shot based on spatial location and scale. These tracked faces form a *face-track*.

Next in the pipeline, facial feature points are extracted from the corners of the eyes, nose and mouth using the code provided by [83]. SIFT-like features are computed for each of 13 points, which are concatenated to form a single vector for each face image. Faces are classified against a set of hand picked exemplars for each character. As in location recognition, we use a K-NN classifier and Kernel-SVM for face recognition. For both sets of

Neigbourhood Size (K-NN)	Accuracy
10	53%
15	58%
20	51%
25	49%

Kernel	Kernel Definition	Accuracy
Linear	$K_L(x,y) = (x^T y + c)$	43%
Polynomial	$K_P(x,y) = (x^T y + c)^d$	45%
RBF	$K_{RBF}(x,y) = exp(-\frac{\|\|x-y\|\|_2^2}{2\sigma^2})$	52%
Min-Min	$K_{MinMin}(x,y) = \max_{p_i \in x, p_j \in y} K_{RBF}(p_i, p_j)$	80%

Table 3.2 Face recognition accuracy for the four main characters in Seinfeld. The table on the left gives results from a K-NN classifier, the one on right for a Kernel-SVM classifier.

classifiers, the label with the maximum score wins, only if it the margin of victory is atleast $K/3$. This technique refuses to label close to 78% of the other characters on the show, while retaining a similar percentage of main characters. Classification accuracy across various classifier settings are given in Table 3.2.

The best face recognition results were obtained using a Kernel-SVM with what is called the min-min kernel [84]. Here, the kernel is defined as the min-min distance, or the maximum similarity (as given by a Kernel), between a given face track and exemplar face tracks:

$$K_{MinMin}(F_i, F_j) = \max_{p_i \in F_i, p_j \in F_j} s(p_i, p_j) \tag{3.4}$$

where $s(p_i, p_j)$ is computed as a RBF Kernel on the distance between the facial feature vectors of p_i and p_j.

The goal is to obtain high precision at acceptable recall rate. We use a refuse-to-predict scheme of [83] where labels are given to face tracks only if we are confident about such an assignment. The precision-recall curve for the classifier is given in Figure 3.10. Precision is the fraction of correctly labeled face tracks, and recall is fraction of the tracks whose label is predicted. Our classifier achieves a precision of 80% at 65% recall.

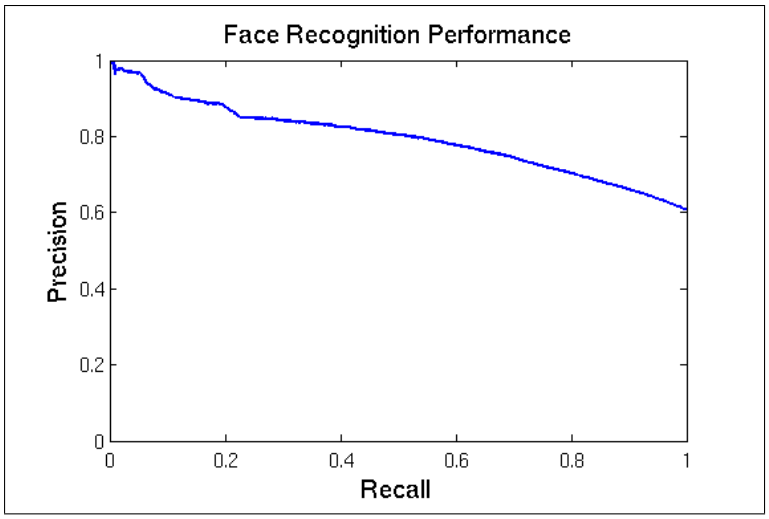


Figure 3.10 Precision-Recall curve of face recognition. We pick the operating point where the precision is 0.8.

3.3.3 Speech Recognition

The audio track provides useful clues for aligning the spoken sentences. We explore the use of speech recognition for our alignment procedure. With ideal speech recognition, we would easily avoid the need for subtitles, since the speech would be sufficient to align sentences to shots. However, as we shall see, this turns out to not be the case for our data.

The audio from each shot is isolated and provided as input to standard speech recognition packages, namely CMU-Sphinx [14] and Dragon Naturally Speaking (DNS) [16]. We do not perform either speaker or sentence segmentation of the audio speech. The in-built speech/speaker models were directly used, since training the speech models would require substantial training data for each speaker.

The recognition output for an example shot is given in Figure 3.11. From this audio, the word *psychiatrist* was correctly recognised once in both systems, even though it occurs twice in the conversation. Other recognised words were *see a*, *going* (DNS), *now* (Sphinx). Matches over stopwords such as {*a, the, and, I, it, ...*} are not considered. The recognition performance of speech recognition was understandably poor [71, 113], owing to the fact

Actual speech:
No, ma, I'm not gonna see a psychiatrist. N- I don't care if you do pay for it! No! Discussion over. Yeah, alright, I'll see you later. Yes, of course I'm gonna come by. Alright. My mother wants me to see a psychiatrist now.
Recognised speech with CMU Sphinx:
ooohh contest you psychiatrist now how difficult re horrible now shuttle door s elaine guess what sound that and i a hair and the walls visiting just now
Recognised speech with Dragon Naturally Speaking:
home is an interest rate for no destruction of the IIRC it would -- of course I'm going to combine for a little as we see a psychiatrist

Figure 3.11 Speech recognition output over the audio of an example shot (no. 66) , from *The Contest* episode of Seinfeld. The speech of the 18 second clip, is given in the top row. The recognition output from two speech recognition engines, namely CMU Sphinx [14] and Dragon Naturally Speaking [16], is shown in the other two rows. One can notice the difficulty faced by modern speech recognition engines to recognize the unconstrained speech from TV show videos.

that we provide the software with "wild" data: the audio files contain multiple speakers in a single shot, laughter of the audience, varying speed of speech delivery, background music etc., which are not trained for in generic speech recognition systems. We get a word level recognition accuracy of 10%. The number of sentences in the training episodes with at least one word recognised correctly by DNS was 21%. The same for the test episodes was 23%.

3.4 Aligning Videos with Scripts

As was seen in the previous Section, the visual-audio recognition modules are not accurate enough to align independently. However, additional characteristics of the problem can

be exploited for aligning the script to the video. We formulate the problem as one of multi-state labeling of the shots, with the states of each shot corresponding to the sentences of the script. We will illustrate the formulation using the spoken sentences S, though a similar development can be given for the descriptions D. Let $S_i \Leftrightarrow T_j$ indicate that the i^{th} sentence is assigned to the j^{th} shot. We shall denote by $d(i,j)$, the local cost of the assignment $S_i \Leftrightarrow T_j$, and $\mathscr{D}(i,j)$ the global cost of assignment. We have the following constraints in our formulation:

- **Uniqueness constraint:** Each sentence can be assigned to only one shot.
- **Ordering constraint:** The sentences and shots are ordered lists, hence implying a sequence constraint to the sentences and shots. Therefore, if $S_i \Leftrightarrow T_j$, then $\forall i' < i$ and $S_{i'} \Leftrightarrow T_{j'} \Rightarrow j' \leq j$.
- **Null-assignment:** It is possible that certain shots do not have any sentences associated with them. This could be because no character is speaking in the shot, or if it is a stock-shot. Hence, the predecessor of $S_i \Leftrightarrow T_j$ in the alignment could be s.t. $S_{i-1} \Leftrightarrow T_{j-k}, k \in [1, j-1]$. A penalty term is associated with each jump over a shot. There are no null-assignments over sentences, i.e. every sentence is always assigned to a shot.
- **Multiple assignment:** Multiple (contiguous) sentences can be assigned to a single shot. However, their combined word count should fit during the shot duration. The local cost function is modified as $d'(i,j) = d(i,j) + \gamma \cdot dist_{Length}(i, i-1, \ldots, i-k, j)$. We estimate the average number of words that could be spoken in a shot, based on its length. The $dist_{Length}$ is the difference between the estimated word count and the number of words in the sentences $[i-k, i]$ assigned to the shot j.

Speech recognition costs. Speech recognition results are filtered by word length, only words longer than four characters are considered. The similarity score is based upon

the number of words overlapping between the speech recognition output and the given sentence. The maximum overlap was observed to be two. The speech based distance measure $Cost_{Speech}$ is set to be 0 for two matching words, 0.5 for one match and 1 for no matches.

Face recognition costs. Three statistics over the training data are used in constructing the face recognition costs: (i) the probability of the speaker being visible in the shot is 0.94; (ii) the probability that a non-speaker is present in the shot is 0.36; and (iii) the probability that the speaker is visible, but not detected in the shot is 0.07. Accordingly, the $Cost_{Face}$ is defined as $(1 - 0.94) \cdot (1 - classifier\_score)$, if the speaker of a sentence is visible in the given shot. In case the speaker is not visible, $Cost_{Face} = (1 - 0.07) \cdot avg_C(classifier\_score)$, C is the character in consideration. For each non-speaker recognised in the shot, the face cost is *incremented* by 0.36 times the average classifier scores for those face tracks.

Location recognition costs. The location cost depends on the location for the sentence. Since the accuracy of scene segmentation of Jerry's Apartment or Monk's Cafe is 96%, the cost of a sentence being assigned to a shot recognised to be in these locations is set to be 0.04, and 0.96 otherwise.

Occlusion Cost. The cost of occluding a shot could depend on the shot being occluded. A longer shot is less likely to have no sentences associated with it, than a shorter shot. Further, a shot with multiple faces detected, and speech words recognized is less likely to have no sentence associated with it. Thus, the penalty term is defined as:

$$OcCost(j) = 0.2 \cdot shot\_length + 0.4 \cdot N_{faces\_detected} + 0.4 \cdot N_{words\_recognized} \tag{3.5}$$

The sentence occlusion cost, where multiple sentences are assigned to the same shot depends on the sentence-shot lengths. Multiple sentences can be assigned to a shot, only if all these sentences could have been spoken during the shot duration. We estimate the average number of words that could be spoken for each shot, basing on the length of the shot. The $dist_{Length}$ is the difference between the estimated word count and the number

of words in the given sentence. This value is normalized by $\max dist_{Length}$ for the given video.

The local cost $d(i,j)$ is computed from a weighted combination of the location, faces and speech distances:

$$d(i,j) = \alpha_1 \cdot Cost_{Location}(i,j) + \alpha_2 \cdot Cost_{Face}(i,j) + \alpha_3 \cdot Cost_{Speech}(i,j), \alpha_1 + \alpha_2 + \alpha_3 = 1 \quad (3.6)$$

The recursive definition of $D(i,j)$ is given as

$$D(i,j) = \begin{cases} (j-1) \cdot OcCost + d(i,j) & \text{i = 1} \\ \inf & \text{i > 1, j = 1} \\ \min \begin{cases} \min_{k \in [1,j-1]} (D(i-1,k) + (j-k+1) \cdot OcCost + d(i,j)) \\ D(i-1,j) + d'(i,j) \end{cases} & \text{elsewhere} \end{cases} \quad (3.7)$$

This formulation lends itself to be solved using dynamic programming. Apart from the global cost array $\mathscr{D}$, an indicator array I is maintained, where $I(i,j)$ points to the table entry corresponding to the optimum subproblem solution of $\mathscr{D}(i,j)$. By backtracking I, we recover the alignment between the sentences and shots. The complexity of our algorithm is of the order $O((N_S + N_D) \cdot N_T)$. The value of the optimisation function along with the inferred alignment between the shots and sentences is shown in Figure 3.12. It can be observed that the sentence-shot correspondences obtained by our approach are very close to the groundtruth. We shall now quantitatively evaluate the performance of our algorithm.

3.5 Results

Baseline alignment. A blind alignment scheme would uniformly spread the sentences across the shots. Given $N_S + N_D$ sentences and N_T shots, such a scheme would allot $\lfloor (N_S + N_D)/N_T \rfloor$ sentences to each shot, sequentially. This alignment performs poorly,

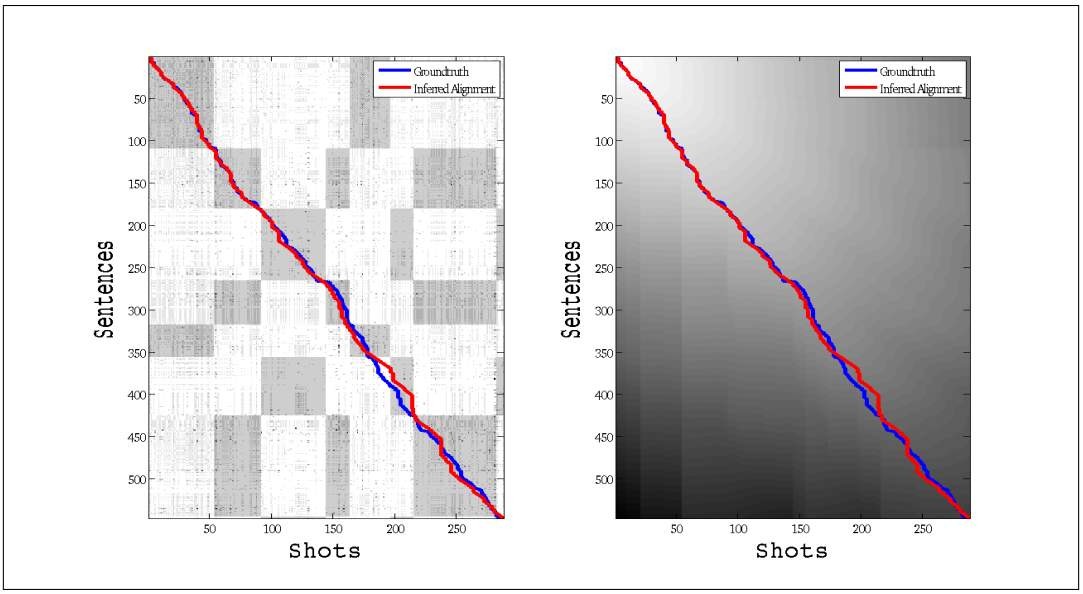


Figure 3.12 (Left) The pairwise matching costs between sentences on the y-axis and the shots on the x-axis. The blocks in the pairwise matching correspond to scenes occurring at a particular location. (Right) The optimization function that is minimized by the Dynamic Programming. The backtracked path is shown overlaid in red, and the groundtruth is shown in blue. It can be seen that the inferred alignment is quite close to the actual groundtruth. The performance of the alignment is particularly significant because the alignment is obtained using the audio-visual information rather than by using the timing information in subtitles.

since errors once committed are hard to recover from, and such errors drastically effect the subsequent alignment. The performance of this method, ρ, is only about 4%. From the groundtruth sentence-shot correspondences (shown as blue lines), it can be seen that the uniform speech assumption is invalid. A stronger hypothesis is required to assign a particular sentence to a given shot.

Subtitle alignment. Subtitle based alignment uses the timing information in subtitles to assign the sentences to shots. However, the subtitles are designed such that they cover multiple sentences, displayed over multiple shots. There are many instances where a single shot would mark the end of a subtitle and the begin of the next. Such sentences are spread across multiple shots using the naive alignment scheme. The ρ of this scheme was 91%.

Figure 3.13 Example shots from the sitcom Seinfeld which are correctly annotated by their descriptions. The first two shots are from the episode *The Contest*, the other two from *The Pick*. The annotations from scripts of Seinfeld episodes are aligned with the video, without using the timing information from subtitles, but by using clues from recognition alone.

Global alignment. Aligning the script with video can be performed using different combinations of the modalities. In the first instance of using speech alone, the sparse sentence-shot correspondences obtained from speech are used to drive the dynamic programming. With ideal speech recognition, this method would replicate subtitle-based alignment. However, given the insufficient speech matches for a bulk of shots in some of the scenes, the ρ of this modality is about 47% on the training data. On the other hand, using the face recognition results alone gives a ρ of 33%. Since we only recognise the faces (we do not perform speaker detection), the sentences are matched across shots where the character is present but is not speaking. Further, the errors in face recognition deteriorate the alignment when the speaking character is not correctly recognised in the shot. These errors confuse the alignment by providing false matches between sentence and shots.

The weights α for each modality are learnt using the training data. The accuracy of alignment for different modalities, and the top five combinations overall are given in Table 3.3. The weights for each modality guide the alignment in cases where the clues do not agree with each other. If for example, the location and speech clues conflict, the respective α determines which modality takes precedence. Thus, with a higher weight for speech, errors due to location could be overcome and vice-versa. The best performing parameters

Combination	Weight Set	Accuracy	Combination	Weight Set	Accuracy
Location alone	{1, 0, 0}	30%	Loc + Face	{0.2, 0.2, 0.6 }	67%
Face alone	{0, 1, 0}	34%	Speech	{0.2, 0.1, 0.7}	63%
Speech alone	{0, 0, 1}	46%		{0.3, 0.2, 0.5}	59%
Loc + Face	{0.5, 0.5, 0}	42%		{0.2, 0.5, 0.3}	54%
Loc + Speech	{0.5, 0, 0.5}	56%		{0.2, 0.7, 0.1}	49%
Face + Speech	{0, 0.5, 0.5}	51%			

Table 3.3 Alignment performance using different combinations of clues/features and their weights.

were found to be {0.2, 0.2, 0.6} for location, face and speech respectively. At this setting, the ρ over training data was 71%, and over the test episodes was 67%.

Scene level alignment. Using the location recognition of Section 3.3.1, we restrict assigning the sentences within a scene to the shots of a location video segment. The alignment within scenes is carried out using the face and speech information. This procedure essentially anchors the sentences and shots known to belong together, and identifies an alignment between such anchors. With this procedure, we are able to improve results and we obtain a ρ of 74%. Figure 3.13 shows example shots that were correctly annotated through the alignment process with this setting.

We have analysed the errors in the episode *The Contest*, some of the errors are shown in Figure 3.14. The two main reasons for the alignment errors are mistakes in the scene segmentation, and the lack of sufficient clues from speech to correct the location based errors. Of the 15 scenes in the episode, 5 scenes have poor temporal segmentation and sparse speech-based matches. These scenes account for about 71 erroneous sentence assignments. The remaining 68 errors are distributed across the other 10 scenes.

In most error cases, the sentence is assigned within a few shots of the actual correspondence. Over our test data, the maximum distance of an erroneous assignment was of five shots, hence ρ_5 = 100%. This can be seen from the graph in Figure 3.15. In a video re-

Shot	Annotation	Shot Error	Reason for Error
	(Jerry and Elaine look at each other - enjoying the story) My mother had a Glamour magazine, I started leafing through it.	-1	Sentence spread across two shots
	(Snapping) Is that important, really? What is this, a police investigation?	+2	Dialogues spoken faster than usual; insufficient clues from speech
	(Jerry looks back at Kramer in envy) It's hot in there.	-2	Four shots at the end of the scene have no dialogues, upsetting the alignment

Figure 3.14 Examples of erroneous alignment over Seinfeld. Much of the other errors occur due to insufficient/erroneous clues obtained from visual-audio recognition.

trieval scenario, for a given textual query, we could provide a video segment consisting of multiple shots. By returning video segments consisting of about 11 shots, the precision of retrieval would be 1, even though the precise alignment might be less accurate. Example retrieval results from Seinfeld for an example query is shown in Figure 3.18.

3.6 Aligning Scripts of Silent Movies and Indian Films

We apply the various cues discussed above, to align the silent movies of Charlie Chaplin with their scripts obtained from [13]. We apply our techniques to two scenes, one from

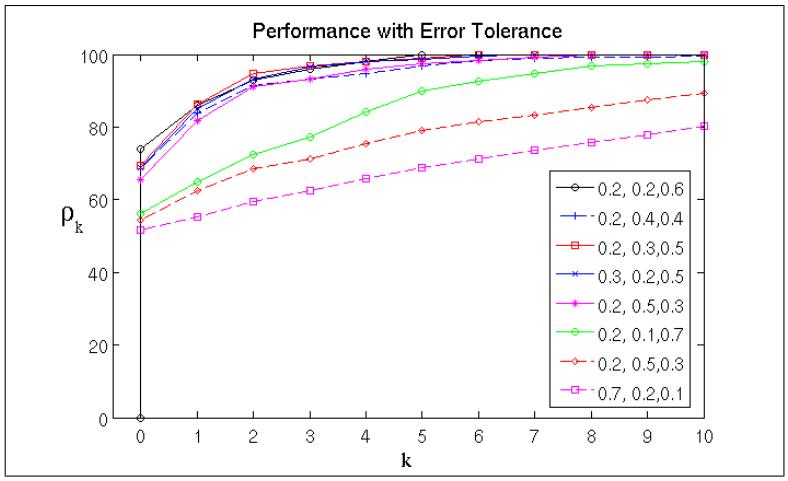


Figure 3.15 Performance metric ρ_k across various values of k, as evaluated over the test episodes; the weights for the {location, face, speech} costs are specified in the legend.

The Gold Rush and the other from *City Lights*. In Seinfeld videos, we have used the stock-shot information for scene segmentation. In the case of Chaplin movies, scene changes are observed as a visual fading to a blank screen. By detecting the blank screen, which can be reliably performed, we find the scene boundaries. The scene segment of the text is identified from the mention of *Curtain*. For example, the scene changes whenever the script indicates similar to "Curtain lowered to denote lapse of time"

For the video clip from Gold Rush, the script only describes scenes occurring in the setting of *The Cabin*. To detect this scene, we use a Kernel-SVM classifier over BoW representation, similar to the one used in location recognition of Seinfeld. Shots are classified as belonging to the Cabin, the scene with the most non-cabin shots was classified as the un-scripted scene. Following this, the scenes in the text are correctly aligned with the corresponding video segment. Within a scene segment, the alignment of descriptions is carried out using face recognition, with the method of Section 3.3.2 to label the characters Lonly, Old Timer, Black Larsen. Sample annotations after alignment of the script with the video, are shown in Figure 3.16 (left). The clip from City Lights has *Intertitles*, where the dialogue is written on the video frame itself, as shown in Figure 3.16. These intertitles

	Lonly, is seen at stove stirring boiling pot. Old Timer is leaning over table groaning.		Tramp stands up and accepts a glass from Millionaire
	Lonly pokes fork in pot as if testing tenderness of what he is cooking		*Here's to our friendship —
	Old Timer grimaces as he bites into piece of upper.		He pours glasses full. They both hold glasses above heads in toast fashion.

Figure 3.16 Examples of annotated scenes from (left) *Gold Rush*, (right) *City Lights*. The second shot in the City Lights example is the Intertitle; the OCR output for this is shown alongside.

were detected and fed to a commercial OCR [33]. The recognised text provides us with additional constraints for the alignment. Resulting annotations are shown in Figure 3.16 (right).

We further test the applicability of our approach in aligning scripts of films with no subtitles, a common situation for Indian films. For this example, we choose a segment from the movie *Agantuk*, the script for which was available in published form [186]. We use the face and speech modalities for the matching sentences and shots. Though much of the dialogue is in the language of Bengali, it is interspersed with English words. By applying speech recognition using the DNS engine, we could obtain about nine strong matches between the recognised speech and the script. Using the speech and face clues, we achieved satisfactory alignment results for this data. Sample annotations are shown in

	  	 
He springs up, snapping his fingers, and breaks into song.	Prithwish loses his patience. He expresses his irritation freely, sweeping out his pack of cigarettes and matchbox from his pocket, and banking them down on the table	The domestic help comes in and clears the table, taking away the tea tray.

Figure 3.17 Examples of annotated scenes from the Bengali movie *Agantuk*. The movie DVD does not provide subtitles, in spite of which we were able to align the video with the dialogues and descriptions provided in the script.

Figure 3.17. Retrieval results from both Charlie Chaplin movies and Agantuk are shown in Figure 3.18.

We were able to demonstrate the possibility of using scripts to annotate movies which have no dialogue, and those for which the subtitles are not available. The audio-visual clues that we propose can be easily applied to new video material, allowing the alignment framework to be applied broadly. However, though our results look promising, we are limited by the availability of scripts for much of this video material. One could make further progress in this direction, if the paper-based scripts available in reputed libraries are made available for all, over the web.

3.7 Summary

In this Chapter, we presented an approach to annotate segments of a multimedia document with segment of a parallel text, given such segmentation of the two domains is straightforward. We propose the use of pattern recognition tools to identify clues towards matching between the visual and textual information. Temporal constraints are used to refine the matching to providing a high accuracy annotation system. The resulting annotation is at a higher semantic level that what can be achieved using current state-of-the-art

Figure 3.18 Example retrieval results for the query of the action *Stand*. Due to the presence of aligned text, these actions are easy to retrieve, rather than through recognition. While the precision of retrieval results is quite high, the recall is limited by the detail in the descriptive text, the script in our case.

recognition technology. The annotated data could be used in future work to train newer classifiers, which in turn could help in improving the alignment/annotation scheme.

The solutions presented in this Chapter vary across the video material as we exploit the domain specific cues in each set of videos to build a stronger solution. The limitation of this, is the solutions are not generalizable across a larger variety of movies and TV shows. A truly generalizable solution is still far from the reach of current visual and learning solutions.

Further, Natural Language Processing (NLP) techniques could be used to obtain semantics from the script, which could provide better cues for alignment. However, the limitation is the lack of robust visual features and recognition modules that can detect "semantics" of the visual data rather than merely the "syntactic" objects/locations. We hope that with newer machine learning tools, these limitations could be overcome in the future.

In the next Chapter, we shall explore the possibility of performing annotation of videos where the segmentation of the multimedia is not given *apriori*.

Chapter 4

Automatic Temporal Segmentation and Annotation of Cricket Videos

4.1 Introduction

In the previous Chapter, the alignment between script and video involved well defined text and video segments. Shots of video are easy to obtain, rendering the segmentation of the video into logical segment was straightforward. In this Chapter, we shall address the problem of jointly inferring the temporal segmentation and alignment of text to video. The particular case in consideration is that of broadcast Cricket videos. In Cricket, the meaningful entity of the video is a scene called the *ball*. Each *ball* consists of multiple shots. The segmentation of the video into scenes is unknown, which needs to be inferred along with the alignment with the parallel text. Based on the results from such a process, we have built a *text-based* search and retrieval system for Cricket videos. The problem setting of this Chapter is depicted in Figure 4.1.

Our approach results in the following key achievements:

1. A mechanism to utilize parallel textual information for segmenting large videos in to semantically meaningful entities.

2. A modeling approach that can match multi-modal information in an intermediary visual-temporal space.

3. Annotation of events Cricket matches are annotated with thorough semantic textual descriptions.

4. The Cricket video collection is searchable with textual queries; accurate results could be retrieved in less than a second. Users can build personalized summaries of videos.

To build a successful retrieval system using unaligned parallel text, three major aspects need to be addressed:

Temporal Segmentation: Textual descriptions correspond to discrete entities commonly referred to as *scenes*. Unlike shots, which are low level visual entities, we are interested in scenes, which are a sequence of multiple shots that form a meaningful entity. Temporal segmentation is the process of identifying video clips pertaining to individual scenes.

Automatic Annotation: To be able to annotate videos with unaligned text, it is necessary to identify the correspondence across different media of information, i.e. between the scenes and their descriptions. Given scene segments from a video, annotation would identify the right description for the scene.

Search, Retrieval and Personalization: Annotated multimedia should be indexed to efficiently answer user queries. The search system should accept semantic queries and retrieve a ranked list of relevant scenes from the database. Search and summarization should be personalized, by letting the user control the content to suit his/her taste.

The parallel text regarding the match videos was obtained from commentaries available at Cricinfo.com [5]. Cricinfo is a webcast that provides textual commentaries of the action on the field. As it is designed to suit users that cannot view the match live, it is very detailed

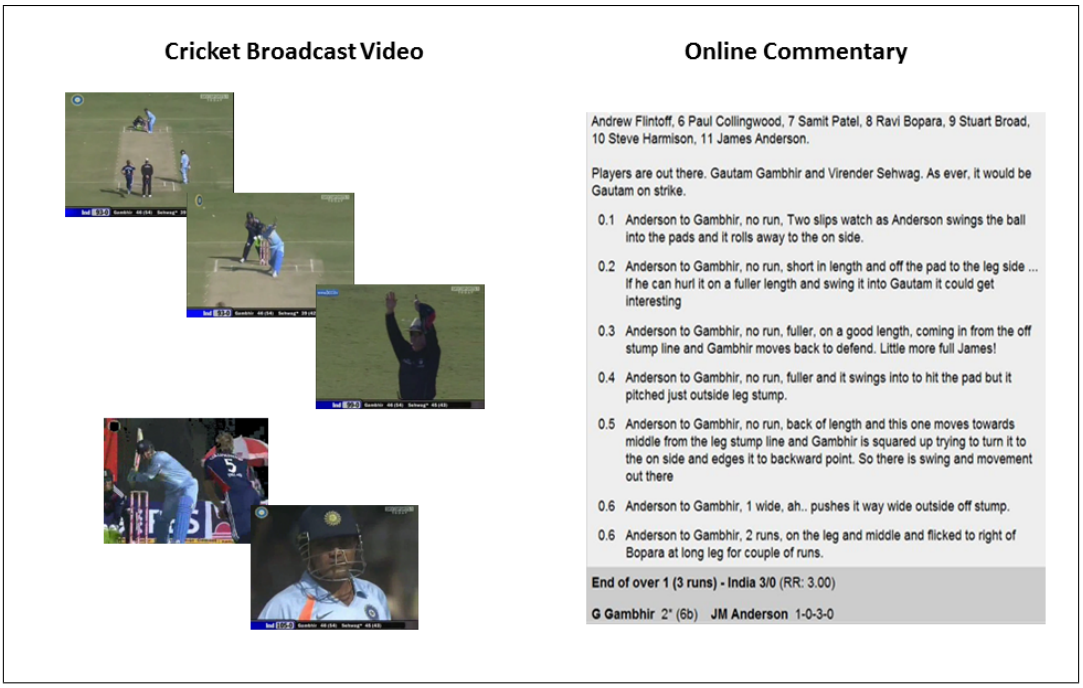


Figure 4.1 In this Chapter, we address the challenge of aligning broadcast Cricket videos with the corresponding online commentary. We build a solution towards matching paragraphs of the text to *scenes* of the video. Further, the segmentation of these scenes is obtained jointly by building a model for the video using information in the parallel text.

and descriptive. Such webcasts were recently used to detect events in soccer videos, and index them for efficient retrieval [233, 240].

In general, sports video processing papers tend to heavily use domain knowledge regarding the sport. Hence, the techniques used for one sport are not applicable to another. While there is ample work on soccer videos, there are few works concerning the sport of cricket. Chambers *et al.* [66] have used gesture recognition to annotate scenes in cricket videos. Kolekar and Sengupta [125] have used HMM based techniques to classify cricket video scenes. However, there is no work that provides a comprehensive retrieval system over cricket videos.

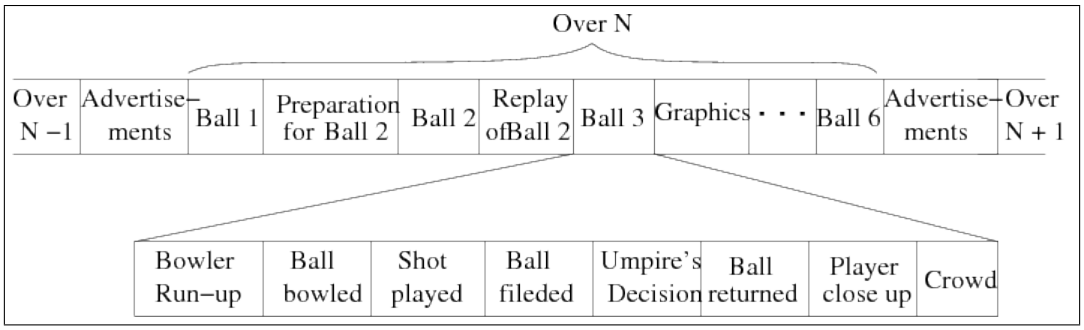


Figure 4.2 Depiction of a generic cricket video. Each *over* has 6 (or more) *balls*, each scene consisting of the ball being delivered, played, fielded and returned. In a broadcast, replays and graphics are shown between the scenes and advertisements between the overs.

4.2 Data and Problem Setting

Cricket is the second most popular sport in the world, with an estimated 3.3 Billion viewers. However, while an average match lasts about 7 hours, much of the interesting action occurs in only about a third of this time. It thus provides ample scope to build intelligent match browsing and summarization tools. Cricket is categorized under the so-called "action-stop" sports. Though the match lasts for several hours, the real action constitutes only a small fraction of the entire length. These action sequences, or *balls* in cricket, occur periodically at semi-regular intervals of time. When the ball is being played there is intense/interesting activity, while there is hardly any action happening between the balls. This provides a clear definition of the scene in the video.

Cricket is very similar in form to baseball. A schematic of a Cricket match is shown in Figure 4.2. The primary action involves a player from Team A called the *bowler* throwing the ball, (called bowling, similar to pitching in baseball) and a player from team B called the *batsman*, hitting the ball. After hitting the ball, the batsman runs across the playing area called the *pitch* to score a *run*. The batsman can score as many runs as possible before the ball is stopped by the bowler's team mates and returned to the bowler. If the ball crosses the field without being stopped, the batsman scores four or six runs without having to run across the pitch. The batsman could get *out* in many ways, for eg., if the

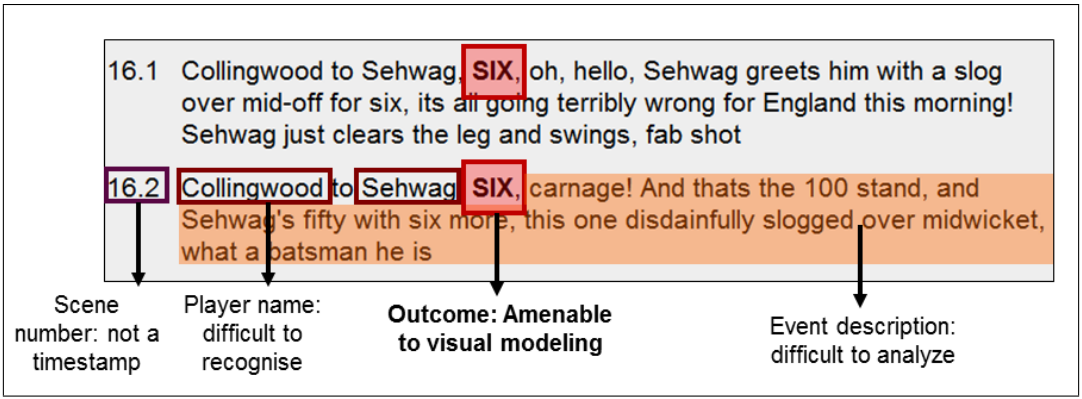


Figure 4.3 Anatomy of the commentary obtained from Cricinfo.com. Each event is described by: i) the scene number, which is not an explicit timestamp, ii) the names of the key players in the event, who are difficult to recognize, iii) the description of the happenings in the event which is difficult to automatically analyze. The information that we find most useful is the *outcome* of the event, which shows a possibility to model with visual analysis.

hit ball was caught before it touched the ground. The entire sequence of action including the ball being thrown, played, stopped and returned is called one *ball*. The same bowler bowls six consecutive balls, called an *over*. Each team is given one innings of 20 or 50 overs to score their runs. The team with more runs at the end of their innings is declared the winner.

In the broadcast video, the lull between the scenes is sometimes replaced by replays, graphical slides, etc. The short break between consecutive overs is typically filled with advertisements. For our purposes, the "scene" starts with the bowler running to deliver the ball, and ends at the beginning of either i) the next ball or ii) an advertisement. Example scenes are shown as a sequence of keyframes in Figure 4.4.

Commentaries for cricket matches are obtained from Cricinfo.com [5]. The text is well structured with each ball being described in fair detail, including the bowler-batsman involved, outcome and a brief description of the action. Unlike webcasts of soccer, there are no timestamps in the text for direct synchronization with the video. An example piece of commentary from Cricinfo is shown in Figure 4.3.

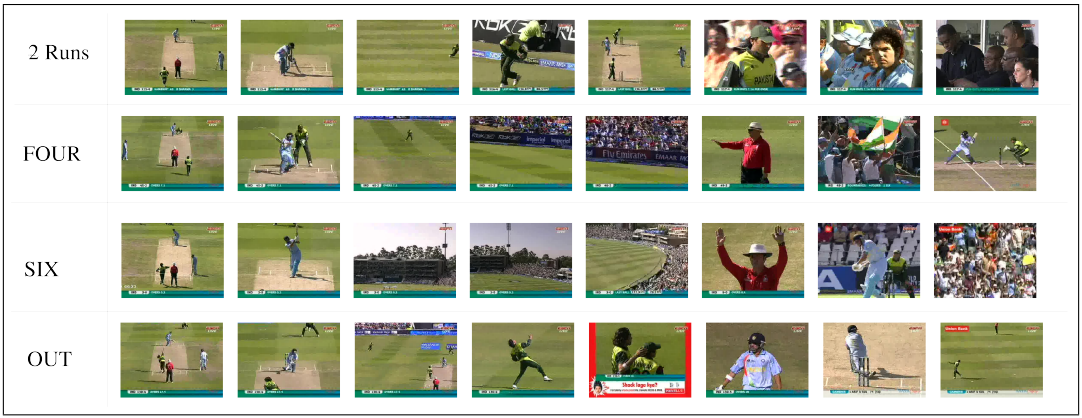


Figure 4.4 Examples of scene segments with different scene categories. Each row is an example for the scene category mentioned on extreme left. The keyframes in each column are extracted at approximately the same relative position from the first keyframe of the scene. In the first two frames, the ball is bowled and played. The next three frames show the action after the ball was hit. The next frame shows player or umpire reaction. The last two frames depict either replays or random visuals used to fill the gap between the action.

4.3 Our Approach

Let us begin with the assumption that we are given segmented scenes from a video (we later relax this assumption in the next Section). Classifying these scenes would now correspond to assigning the right label to each scene. The label-set for the scenes are defined by the domain of the videos. For our case of cricket videos, we first need to identify the suitable labels to classify scenes against. We observe that the *outcome* of each *ball* is a suitable scene category.

Consider the example scene for the outcome FOUR in Figure 4.4 (second row). The scene begins with the bowler running toward the pitch, to bowl the ball. In the first keyframe, we see the pitch just before he throws the ball. The thrown ball is then hit by the batsman in the second frame. The ball then travels across the ground in the next two frames, and crosses the boundary in the fifth frame. The umpire then signals “four” in frame 6. Soon after, a replay of this action is shown, as shown in the last two frames of the example. This sequence of visuals are typical for the outcome FOUR. On the other hand, the sequence of events is considerably different from that of other scene categories, as can

be seen from the 3rd, 4th and 5th keyframes of those categories in Figure 4.4. Further, the average duration of the different outcomes is not the same, which is also an important clue in distinguishing between scene categories.

These properties allows us to categorize the scenes based on the *outcome* of the ball. Our scene labels is the set of all outcomes: {*no-run, one-run, two-runs, three-runs, four, six, out*}. Following the above observations, each scene category needs to be modeled based on visual *change* across the scene. Though FSMs and HMMs [125] are popular candidates to model such changes, they do not provide control over the actual duration of scenes. Instead, we build a simple scene model, as an ordered sequence of the constituent frames' representation, as we shall describe below.

4.3.1 Visual Representation of Frames

The basic visuals in a frame of Cricket videos belong to one of C = {*pitch, ground, team A player close-up, team B player close-up, sky, replay, advertisement*}. More than one of these categories could be present in a single frame. Each of these categories was modeled using color histograms in the RGB space. The players are modeled using the color of their team's clothing. Representative features were learned from a training set of marked regions corresponding to the ground, pitch, sky and player jerseys. Once the histograms are learned for each category, each pixel in a new frame is matched against each class-histogram. The class with the most match with the given pixel color is assigned to the pixel. The frame is then represented as the percentage of pixels in the frame belonging to the each visual category. A few results from this classification are shown in Figure 4.5. The ground, pitch and sky are colored with magenta, cyan and yellow respectively. It can be observed that the pixels are accurately classified, making this a strong representation for the frames.

On the other hand, advertisements and replays cannot be easily modeled using visual features, since there is no control on the visual content in this category. It was noted that in case of these two categories, the scorecard at the bottom of the frame is removed by the broadcast production (as can be seen in Figure 4.5). By modeling the scoreline's color distribution we could detect the scoreline, to distinguish between match play and ad/replays. In Figure 4.5, at the third row is a frame from a replay. As can be seen, the scorecard is missing at the bottom of the frame, hence there is no scorecard detected for this frame.

4.4 Scene Model Learning & Matching

Each scene would consist of a set of frames whose visual content changes in a particular manner. A set of suitable features are extracted from each frame, and the scene is modeled using these frame descriptors across the scene. From a set of training examples, the scene model is learnt by averaging the scene models extracted from each example. It is sometimes useful to have a set of categories for the frames, as well. This enables one to model the scene descriptor, as a set of probabilities for each frame in the scene, to belong to each of the frame categories.

Algorithm 4.1 Train_Scene_Model(k', $V_{k'1}, V_{k'2}, ..., V_{k'N}$)

1: Find average length L of the training videos $V_{k'1}, V_{k'2}, ..., V_{k'N}$
2: **for** i = 1 to N training videos **do**
3: Obtain the scene representation P_i for $V_{k'i}$, as the probabilities of each frame to belong to the frame classes
4: Normalize P_i to length L by linear interpolation
5: **end for**
6: Obtain scene model $M_{k'} = \frac{\sum_{i=1,2,...,N} P_i}{N}$
7: Return $M_{k'}$

Given the above representation, we would now like to learn scene models for the scene categories. Scene models are learned from a set of training data, as given in Algorithm 4.1. The algorithm takes an input of training scenes for the category k'. These scenes are

Figure 4.5 (Above) Examples of pixel-wise classification for frame representation. For the original frame shown in the left most column, the pixels classified as ground, pitch and scorecard are colored with magenta, cyan and red respectively. Notice the absence of the scorecard in row 3, since the frame is from a replay shot. (Below) Frames and the pixels classified into the sky and player categories. Note that we learn different color models for players from different countries. It can be seen that the pixels are classified fairly accurately using our approach.

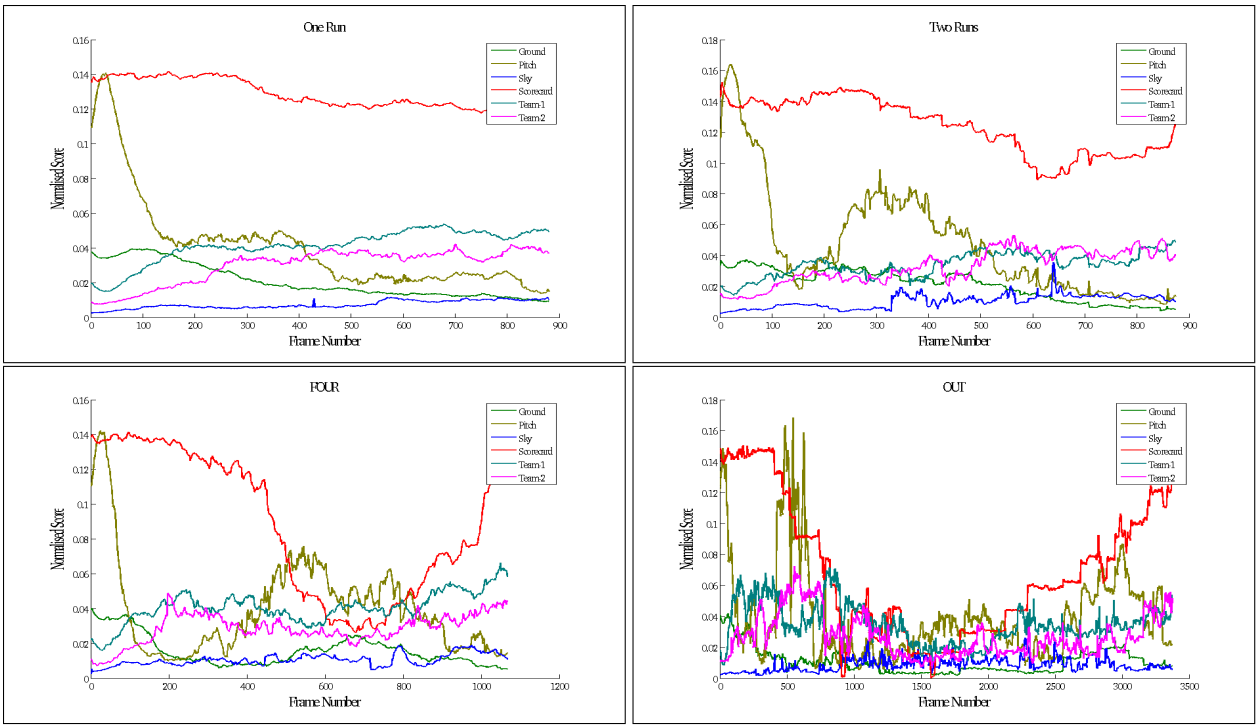


Figure 4.6 The models learnt for some of the scene categories. On the x-axis is the frame number, and the y-axis the score for each of the *frame categories*. The canonical length of the model varies across the different scenes. We can easily observe that different scene categories exhibit markedly different behaviour, even with a simple color based pixel classification scheme. This difference in models will help distinguish one scene category from another in the real video.

represented as the concatenation of the constituent frames, normalized to the average scene length. The learned model is given as $M_{k'}$. Example scene models for four categories are shown in Figure 4.6, where we can observe the marked differences across the outcomes using the features we represent them by.

Given learned scene models, the scenes are classified using a nearest neighbor classifier between the query scene and the models. Since the query and model descriptors would be unequal in length, the length of the query feature is normalized to that of the scene model. The two features are then compared using the L1-Norm to obtain the overlap between visual appearance of the scene and the model. The scene descriptor that we obtain from this procedure is the set of probabilities for a given frame to belong to each of the frame

classes, across the scene. The procedure is formally described in Algorithm 4.2. Let the given video be denoted as P_q, and the learned scene models be $M_{k'}, k' = 1, 2, ..., K'$.

4.5 Modeling Text

We have so far presented methods to model scenes and predict their category. This schemes assumes that the video-segment belonging to the scene is provided apriori. In our case, we do not have such a segmentation. We pose the temporal segmentation problem as *"given a set of scene models, find the temporal locations in the video, where there is a scene change"*. This is an ill-posed problem, since the categories of the scenes before and after the scene change are unknown. Moreover, the scene category can not be identified without proper scene segmentation. Simultaneous optimization of both the scene category and the scene segmentation would be difficult. The challenge can be alleviated by the use of additional information, available to us in the form of text commentaries of the events in the video. This information is very reliable in describing the content of the video, and constitutes a layer of top-down information above the scene models. If the scene categories could be identified from the textual description, the corresponding scene models could be used for segmentation.

Algorithm 4.2 Annotate_Video(P_q)

for k' = 1 to K' scene categories **do**
  Normalize P_q to the length($M_{k'}$), by linear interpolation
  $d_{k'} = 0$
  for i = 1 to length($M_{k'}$) **do**
    for k = 1 to K frame categories **do**
      $d_{k'} = d_{k'} + P_q(i,k).M_{k'}(i,k)$
    end for
  end for
end for
Find $category = \min_{k'} d_{k'}$
Return label corresponding to $category$

However, the text and video are not exactly synchronized. It is unknown as to which segment of text corresponds to that in video. There are two options to align text and video: 1) convert the video to text and align in text space 2) convert the text to a video, and align in visual space. Option 1 is the conventional approach and aligning in text domain is easy and efficient. However, robust activity recognition and scene analysis techniques are required to convert the video to text. A semantic description of the scenes (and video) would require the entire video understanding problem to be solved. Option 2 is equally difficult, since generating a real video from a scene description would require the complete computer graphics problem to be solved.

This is addressed by matching the text and video in an intermediary feature space. The video could be converted to such a space by extracting the features from the video. If we could convert text to this feature space, we could align, match and process the text and video on a common platform. This is the motivation for our generative video model. The generative video model uses the observations that a) it is known that the complete text, corresponds to the entire video, b) the video and text are constrained by temporal consistency across the visuals and descriptions. This allows us to build a *hypothetical video* from the textual descriptions.

The feature space chosen to model the video and text is the feature space of the frame descriptors. The scene categories have been modeled using these features for this precise purpose. The text provides the scene category of each scene in the video. At each slot of the scene, the corresponding scene model could be plugged-in. By concatenating the scene models according to the order of scenes given by text, we obtain the hypothetical video, H. The visual data is converted to this domain by classifying the frames in the video to obtain a feature representation of the real video, D.

4.6 Scene Segmentation by Visual Alignment

In a general model fitting problem, there are two unknowns: i) the model parameters and ii) the mapping of the the observed data, D to the assumed model for the data H. In case neither is known apriori, the unknowns are simultaneously estimated using an Expectation Maximization procedure. If the model parameters are known, the only unknown that remains, is the mapping of the observed data to the assumed model. This can be estimated using a Maximum Likelihood (ML) estimation.

The segmentation requires to identify the begin and end frames of the scenes, over the video. The real scene boundary Z_i is assumed to be fixed but unknown. The estimate z_i of the scene boundary, is assumed to be found near the real boundary Z_i with a probability distribution that follows a Gaussian. The Gaussian is centered around Z_i, with a variance σ. The estimate z_i is obtained from visual-temporal information. Let such an observation of the beginning and end of a scene S_i be z_{i_1} and z_{i_2} respectively. The likelihood that shot S_i bounded by z_{i_1} and z_{i_2} actually corresponds to a real scene X is given by $P(S_i|X) = P(z_{i_1}, z_{i_2}|X)$. This likelihood corresponds to a local cost of S_i corresponding to X. The global cost of matching scene estimate set γ with real scene boundaries is given by

$$L(\gamma) = p(Z_1, Z_2|\gamma) = \prod_{0<i<n} P(z_{i_1}, z_{i_2}|X) \tag{4.1}$$

where n is the number of shots in the video. The maximization of the global likelihood function corresponds to minimizing its negative logarithm. In cases where the scenes are not represented by a known model, the optimization of this function could be done using an Expectation Maximization approach, where both the segmentation and scene parameters are learnt simultaneously. However, using the textual information, the appropriate scene models could be plugged into the likelihood computation. The minimization in such a situation would correspond to a simple weighted matching or assignment problem, which could be solved in polynomial time using dynamic programming.

The video description, coupled with the scene models, provides an assumed model H for the video. We expect the real video D to closely resemble H, as seen by the frame descriptors. The ML estimation of scene segments can now be posed as a mapping of D to H, which can be computed using a Dynamic Programming (DP) technique [74]. Assuming that the distance array in the DP procedure is given as D, we use the DP cost computation:

$$D(i,j) = \min \begin{cases} D(i-1,j) + c(i,0) \\ D(i-1,j-1) + d(i,j) \\ D(i,j-1) + c(0,j) \end{cases} \tag{4.2}$$

where the local distance between two frames, i and j is given by

$$d(i,j) = \sum_{s \in shotclasses} P(i_s).P(j_s) \tag{4.3}$$

$P(i_s)$ being the probability that the ith frame belongs to the s shot class; and $c(i,0)$ is the cost of occlusion. The optimal path of the match is found by backtracking the DP matrix. With this match, the observed scenes from D are warped onto the generated model H. Such a warping specifies a correspondence between the scenes in H to those in D. Since, the scene boundaries in H are known apriori, the corresponding segment in D can be extracted from the optimal DP path.

The procedure is formally given in Algorithm 4.3. The algorithm takes as input the video to segment, V and the corresponding text commentary T. In the algorithm, Dynamic_Programming and Back_Track are standard dynamic programming and backtracking routines. The result of the Algorithm 4.3, is a segmentation of the given video into scenes. We also know which scene model was used to segment each scene. Thus, we have simultaneously segmented, as well as classified the scenes with their labels. The sequence of videos provides us with the scene ID for each segmented video.

Algorithm 4.3 Segment_Video(V, T)

Build D = video_representation(V)
/* Building Hypothetical Video */
Set H = NULL
for each scene i = 1 to n **do**
    Extract the scene category k' for scene i from text T
    Concatenate scene model $M_{k'}$ to H
end for
Compute DP = Dynamic_Programming(D, H)
Find optimal path using P = Back_Track(DP)
/*Segment Video*/
for each scene i = 1 to n **do**
    Find the scene segment S_i corresponding to i in generated video H
    Find correspondence of S_i in P as V_i'
    Output segmented video V_i'
end for

4.6.1 DP on Long Sequences

Classical Dynamic Programming is memory intensive, requiring $O(N^2)$ storage for the aggregated distance matrix D and the indicator matrix I. This places a serious restriction on the length N of the signals that can be aligned. On a typical computer with , we observed that the maximum length that can be aligned was only 20K frames, which would correspond to 20 scenes in the video. To address this issue, we propose a new space-efficient DP algorithm.

The key observation to make the DP space-efficient is that for calculating the cost $D(i, j)$, one only needs three values from the D array: $D(i-1, j-1), D(i, j-1), D(i-1, j)$. The values in the D sub-matrix from $[0, i-2][0, j-2]$ are not required to be stored, since their costs are already embedded in the cells $D(i-1, j-1), D(i, j-1), D(i-1, j)$. Thus, to compute the values for a single row i (or column j), it suffices to only know the values of the previous row $i-1$ (or column $j-1$), and the value for first value in row i, i.e., $D(i, 0)$ (or $D(0, j0)$). By keeping track of only one row or column, we could compute the entire D matrix without having to store all of it. Further, each cell of the indicator matrix I can be

stored as a pair of bits to indicate the direction of the lowest cost. With this improvements, we were able to improve the maximum length for alignment from 20K to 60K frames.

However, this limit corresponds to about 60 scenes of the video, while each video typically consists of 300 scenes. In this situation, instead of aligning the entire video in one instance, the input video and hypothetical video are segmented at five intervals corresponding to the 60 scene boundary. Anchors are manually provided at the location of the 60 scene intervals in the given video, while their location in the hypothetical video can be easily inferred in the generation process. Within each segment, the alignment then proceeds in the usual manner. Further improvements could be obtained by using an advanced computer with larger main memory; we observed that on a 16GB RAM memory, we could align close to 120K frames in once instance using our memory-efficient alignment technique.

4.6.2 Alignment Accuracy

In the process of DTW between the hypothetical video H and the real data D, there are seven parameters to learn for: i) the weights for each of the six features and ii) the occlusion cost. The parameters were learnt by doing a dense sampling of the parameter space, and picking the set of parameters that provide the highest alignment accuracy. The accuracy is measured as the percentage of frames that were assigned to the right scene in the video.

Baseline Alignment We compare our approach to baseline alignment techniques, one that is based on a heuristic segmentation of video and another where visual information is not considered. In $Baseline_1$, the video is segmented by detecting frames that contain the *pitch* visual category. This is inspired by the format of the video, where every scene always begins with a shot of the *cricket pitch*. In order for a sequence of *pitch* frames to correspond to the beginning of a scene, the frame needs to occur in a regular play, and

Method	Accuracy
Visual Alignment	**71.1%**
Duration Only	7.8%
Pitch Detection	16.2%

Process	Time per Video
Feature Extraction	90 minutes
Hypothetical Video Generation	0.35 seconds
Alignment Time	75 minutes

Table 4.1 (Left) Alignment accuracy across the proposed techniques and a couple of baseline algorithms based on heuristics. The best performing setting of weighted DTW alignment results in an accuracy of more than 70% frames in the correct scene. This is significantly higher than the baseline algorithms that do not perform explicit alignment between observed and estimated visual data. (Right) The time taken by the various steps involved in our solution pipeline, on a regular desktop machine. For a 3.5 hour video, we require about 2.75 hours of computation time.

not during a replay. By identifying such frames, one could obtain plausible beginnings for each scene, and hence the temporal segmentation.

The $Baseline_2$, uses information from the text commentary, but does not use visual information for the alignment. During building the hypothetical video H, visual features are discarded, but only the duration of scene is retained. Each frame of the hypothetical video only contains the scene-ID. Instead of warping H to D, we scale H to the same length as D, and read out the scene-IDS for each frame of D as given by the same frame in H.

We have manually generated groundtruth segmentations for 7 hours of video (1 match). The results from the baseline and those from DTW based alignment, over the groundtruth dataset, are presented in Table 4.1. The baselines perform quite poorly, which indicates that visual information is very important and useful in temporal scene segmentation. The evaluation results for the best alignment accuracy of 71% is achieved with the DTW based alignment approach. The time required for each step of the feature extraction and alignment process, over a commonplace desktop machine with about 4GB of memory, is given in Table 5.4. The total time required for 3.5 hour video is about 2.75 hours of computation.

4.7 Annotation of Segmented Videos

With our temporal segmentation procedure, we have simultaneously segmented and synchronized the video segments with their corresponding ball IDs. This allows us to annotate the segmented scenes with the information available in text commentaries. An example segment from the webcast commentary is shown in Figure 4.3. In these commentaries, every scene is recorded and reported. Since the webcast is generated by professionals, it is well-structured, which allows us to parse and extract information from the commentaries. The number on the extreme left is given as $< over\_number > . < ball\_number >$, which provides us with the ball ID. After the ball number, the name of the bowler is given, until the preposition "to", which is followed by the name of the batsman. The next phrase describes the outcome of the ball played. The rest of the sentences describe the action during the ball including description of the delivery, shot played, fielding action etc. Detailed comments regarding a ball generally implies that interesting activity has occurred during the particular scene. The webcast also includes scores at the end of each over and detailed descriptions at the beginning and ending of each innings. This additional information which is not relevant for scene annotation is parsed and removed.

Firstly, each scene is annotated with factual data obtained from the text. Each scene is annotated with the information about the ball number, bowler, batsman and outcome. The score at the end of each ball can be computed from the text commentary itself, using the outcome information. The score is given as $< number\_of\_runs >:< number\_of\_batsmen\_out >$. The number of runs are calculated at the end of each ball, by incrementing it with the number of runs in the outcome. If the outcome is "OUT", the number of batsmen out is incremented.

Apart from factual annotations, descriptive annotations are available from each ball from the comments. These comments are given to describe the action to audiences that cannot watch the match video. Hence the comments are fairly accurate and descriptive

in conveying the events. However, they might not be provided at the same level of detail for every scene. Finally, we store the duration of each ball in frames and seconds for summarization purposes.

The annotation is stored in an XML database, each inning's video described by one XML file. XML is the content description standard for video annotation, as specified by MPEG 7. The XML schema used in our system is given in Figure 4.7.

The XML schema describes the video and each scene in it. The video is annotated with meta information regarding the date of match, teams playing, innings number etc. Each scene or ball is then annotated with the information parsed from the commentaries. XML acts as a structured interface between the data and the retrieval & summarization system. It also helps in easy sharing and porting of annotated video databases.

4.8 Applications over Annotated Videos

We have built a system over the annotated video data with the following functionalities:

1. Video Browsing: Users can easily browse matches using an intuitive interface
2. Search and Retrieval: Users can query with semantic textual queries and retrieve results in ranked or chronological order
3. Summarization: Personalized summaries are generated based on user preference in summary content, summary duration and delivery mode

4.8.1 Video Browsing

The segmentation and annotation enables us to build an easy-to-use interface that allows the user to browse the match efficiently. The *Match Browser*, that we built is a table-of-contents like display that combines video clips with their annotations. The UI is provided

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="matchDate" type="xs:date"/>
<xs:element name="teamA" type="xs:string"/>
<xs:element name="teamB" type="xs:string"/>
<xs:element name="firstInnings" type="xs:string"/>
<xs:element name="inningsNumber" type="xs:integer"/>
<xs:element name="fileURI" type="xs:string"/>

<xs:element name="annotatedBalls">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="ballID" type="xs:integer"/>
   <xs:element name="startFrame" type="xs:integer"/>
   <xs:element name="endFrame" type="xs:integer"/>
   <xs:element name="ballURI" type="xs:string"/>
   <xs:element name="ballNumber" type="xs:string"/>
   <xs:element name="bowler" type="xs:string"/>
   <xs:element name="batsman" type="xs:string"/>
   <xs:element name="outcome" type="xs:string"/>
   <xs:element name="comments" type="xs:string"/>
   <xs:element name="scoreRuns" type="xs:integer"/>
   <xs:element name="scoreWickets" type="xs:integer"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

Figure 4.7 The XML schema used to store annotated Cricket video segments.

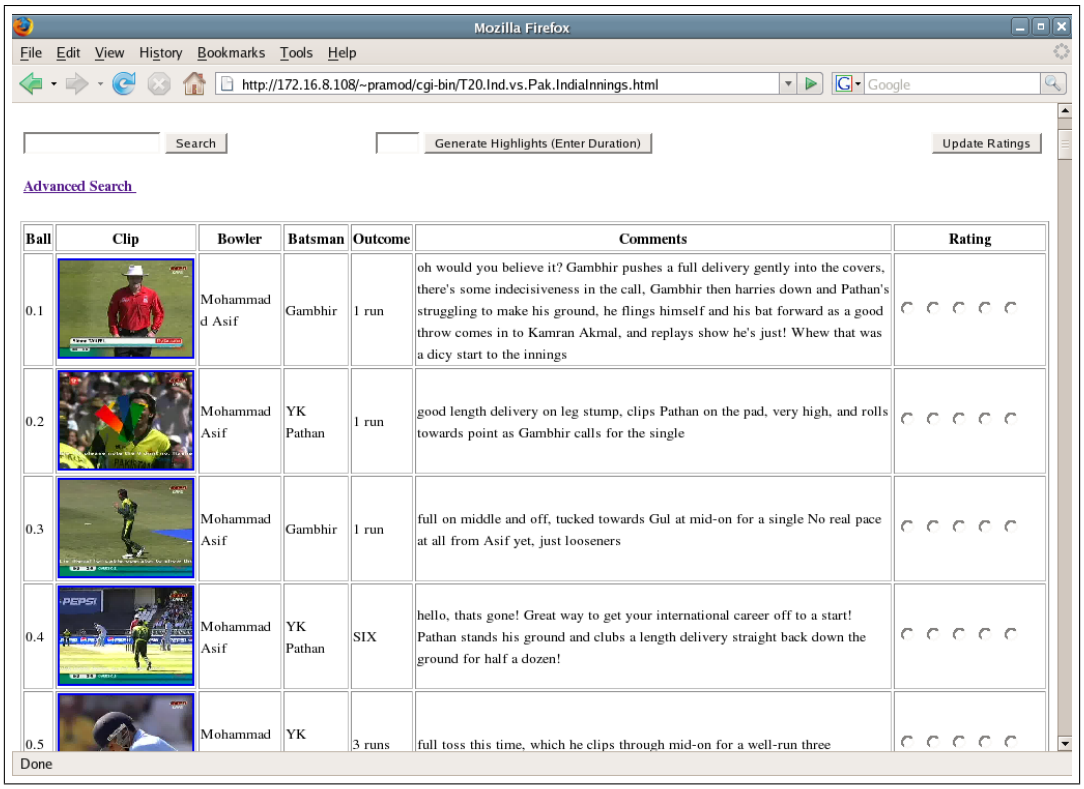


Figure 4.8 A screenshot of the our Match Browser UI. Each column in the table corresponds to various annotations for each ball video. The right most column allows the user to rate the balls based on his liking. The search box on the top-left corner allows user to search and retrieve for his text queries. The second text box in the top takes the desired duration of the user to generate summaries. Personalized highlights are generated using the "Advance Search" link in the second row.

through a webpage; the user need not install any special software to access our system. The screenshot of our interface is provided in Figure 4.8. Each row contains the information regarding one ball. The columns of the table provide information about the ball number, bowler, batsman and outcome. The description of the ball is given in the comments column. The user can easily browse through the entire match by looking at the table. The user can control the number of balls he wishes to see in each view. He could choose to see all the scenes in one view, or browse the match over-by-over. Each ball is depicted with a thumbnail. On clicking the thumbnail, the user can download and play the clip of the scene. A typical clip is about 45 seconds in duration and about 2.5MB in

size. The clips can be downloaded and played in less than 5 seconds, on average, over the Intranet. By choosing to view only those scenes that he might be interested in, the user saves considerable time by using our browser.

4.8.2 Search and Retrieval

Following our annotation, each scene is associated with a set of *textual* tags. We can now build a text-based search engine over the annotated scenes. We perform a standard stop-word detection and removal from the comment tag, before indexing the scenes. Our system accepts text queries given by the user and searches through the index to retrieve the scenes matching the query. The user could thus query for all scenes involving a particular player, or with a certain outcome. The search results are shown preserving the temporal order. Owing to an all text index back-end, the search times of our system are interactive.

However, it is not easy to fully understand and cater to user queries. For e.g., when the user queries for "Yuvraj Six", he probably meant to retrieve all the SIXes hit by Yuvraj Singh. The search system retrieves all the balls which are indexed under both Yuvraj and Six, which contains a large number of results not queried for. To provide results more relevant to the user's desire, the search results should be appropriately ranked.

To be able to retrieve the scenes most suitable to user query, we use standard vector space models popular in text retrieval. Each annotated scene is considered as a bag-of-words of its corresponding annotations. The query too is considered as a bag-of-words. Both the scenes and queries are assumed to be points in a space whose dimensions are given by all unique words (sans stop words) in the annotations. The similarity between the query and annotation is given by the cosine similarity measure between annotated scene S and the query Q as

$$similarity(S, Q) = cos\theta = \frac{S \cdot Q}{|S||Q|} \tag{4.4}$$

The cosine similarity is smallest for overlapping vectors or exactly similar query and document. It is largest when the query and document are un related. This is the distance measure which we use to rank the retrieved results. A suitable threshold is chosen for the similarity measure to avoid retrieving all scenes in ranked order.

4.8.3 Automatic Summarization

Another key application of annotated videos is in generating automatic summaries of videos. Match summaries or highlights generated by the TV studio are fixed and thus might not suit user preference. It is necessary to consider the user preferences in the summary content and allow the user to control the duration of the summaries generated. The user can also search the videos and rank the scenes retrieved for his given queries. This allows the user to rank only those scenes that he would prefer to view and rank, hence encouraging user participation in ranking.

Collecting User Preference. To be able to personalize video summarization, we need to collect the user preferences for the scenes of the video. We allow users to rank the scenes that they like on a scale of one to five. The ratings are collected from a pool of potential users and the cumulative ratings are computed for each scene. Each scene is then re-ranked based on its score. From the ratings obtained from a set of 10 users, it was observed that the outcomes FOUR, SIX and OUT are very popular and are generally included in the highlights. However, there are considerable number of scenes with other outcomes which are also interesting. This could be because of interesting events occurring in the scene. It is in these cases, that user rating will encourage such balls to be included in the highlights.

Controlling Summary Durations. Our application allows the user to choose the duration of the summary as well. Given the required duration, the system automatically identifies scenes that are present in the summary. A threshold is calculated such that

good length delivery on leg stump, clips Pathan on the pad, very high, and rolls towards point as Gambhir calls for the single

good length delivery outside off stump, defended to cover for a sharp single

good length outside off stump, Uthappa gets onto the front foot and just pushes it very so softly in front of cover and calls for the sharp single

good bowling this, Irfan pitching it on a **good length** and forcing Misbah to play across the line, only to mistime it and play it back to the bowler

---

short of a **good length** and outside the off, played down to third-man with an angled bat

Figure 4.9 Retrieved results for the query “good length”, which is a property of the bowling. The top four results are true positives. Retrieved scene videos are shown as a sequence of keyframes. The corresponding comments in the annotation is given beside each scene, the occurrence of the queried words is highlighted. The last result is a false positive. Though the query words are present in the annotation, the comments are describing a different kind of bowling which is “short of a good length”.

System Metric	Value
Number of hours of video	64.3 hours
Size of video	63.3 GB
Average scene duration	45 secs
Average scene clip size	2.5 MB
Average time to play scene	4.5 secs
Retrieval response time	0.66 sec
Average summary duration	50 mins
Average summary file size	165 MB
Average summary generation time	30 secs
Average time to play summary	25 secs

Table 4.2 Details of the dataset that we annotate and segment with parallel commentary, along with some of its properties. Over a large collection of more than 60 hours of video, we could build a semantic retrieval system that answers queries in less than a second, and can build a personalized summary video in about 30 seconds.

the duration of the balls above this threshold is closest to the desired summary duration. Given the ranked list of all the balls, the summary would consists of all the balls with ranks above this threshold. A summary video is generated on-line by concatenating the scenes preserving temporal order, and provided for download to the user.

Summaries from Querying. Another novelty of our system is providing summaries based on search queries. An advanced search and summarize interface is provided for this purpose. This interface contains a list of all the players and outcomes. The user can choose his favorite players and preferred outcomes. He could choose multiple players from both teams, from different innings. He could also give textual queries such as “good length delivery”, “poor shot” (a shot in cricket, refers to the way the ball is played with the bat), etc. The annotations are searched through to identify balls that satisfy the given choices. The retrieved scenes are then concatenated in temporal order to obtain a very personalized summary of the match. This summary can either be browsed through over the webpage or played as a video.

The processing time for summary generation is consumed mainly in concatenating the videos and encoding them into a single file. We use the Mencoder library to merge scene clips into a summary video. A summary of 50 minutes duration is generated in less than 25 seconds, and the file of size 165MB can be downloaded in less than 30 seconds. This means personalized summaries of the match video are less than a minute away for any user in the local network. The performance metrics of our system are summarized in Table 4.2.

4.9 Discussions

We present our results on 10 matches, consisting of 64 hours of video from the 2009 ICC Champions Trophy. We applied the techniques discussed in this Chapter to segment the video into clips of scenes, and have annotated them with the information from the commentary. As an example, we provide search results for the query “good length”, which corresponds to a property of the ball being bowled at a particular place on the pitch. Keyframes for the top four retrieved scenes are given in Figure 4.9, along with the query highlighted in the comments. It should be noted that retrieving “good length” balls using recognition of the visual information alone, is a very difficult problem. Such semantic querying was made possible due to the effective use of parallel text in our annotation procedure.

As can be seen in the Figure 4.9, there are some false positives in the retrieved results. Though the false positives contained the query words “good length”, the comments were actually pertaining to a “short of a good length” ball. Such queries are ambiguous to our search system, resulting in lower precision. On the other hand, the recall is limited by the number of scenes that have been commented about in their full detail. A number of other scenes might have been good length deliveries which was not mentioned in the commentary. As another example, for the query “good shot” we could retrieve only three

scenes, when realistically there would have been large number of scenes which had good cricketing shots.

4.10 Summary

In this Chapter, we have presented a scheme to simultaneously segment and annotate videos with their parallel text, using a unique text-driven temporal segmentation algorithm. Our work has enabled text-based access to a large collection of video data, that was difficult to navigate, search and consume. With this work, we begin with a video-page level alignment and tighten it to a scene-paragraph level.

In future work, we could perform much more *fine-grain* segmentation and annotation of both the visual and textual data. Further, statistical generative models such as Hidden Markov Models (HMMs) could be evaluated for automatic learning and segmentation of the Cricket video into scenes and actions/activities. The training data required to satisfactorily learn the HMMs could be bootstrapped using the segmentation and annotation obtained as a result of the work in this Chapter.

Chapter 5

Reverse Annotation based Retrieval from Large Document Image Collections

In the previous Chapters, we had addressed the problem of annotating *videos* with parallel un-aligned text. In this Chapter, we shall focus on a class of images, that are rich in text data, namely document images created by scanning printed books. The text information is encoded in pixels, making it difficult to access or search these archives easily. However, for many collections, the language of the books is known beforehand. Given the language, we could easily obtain resources such as a list of possible words or a dictionary. The *words* in the image can now be treated as the meaningful entity that could be annotated by the appropriate word in the known vocabulary. This broad goal is depicted in Figure 5.1. We now present the challenges and solutions towards such an annotation on a large-scale document image collection.

5.1 Introduction

The process of creating digital content from paper-based books is now well established, thanks to various digitization projects around the world. Google Books [6], Internet Archive [7], Universal Digital Library (UDL) [8], Digital Library of India (DLI) [2], etc. have scanned millions of books, creating more than a billion document images. The infor-

mation and knowledge captured by these image collections needs to be seamlessly accessible over the web. One of the key mechanism to provide such an access is through enabling a search system, that could retrieve relevant content at the paragraph or sentence level, similar to popular web search engines.

In order to build a search engine over document images, one needs to obtain the text-equivalent for the image content [43]. This was traditionally achieved using Optical Character Recognition (OCR) [42, 155, 166]. OCRs typically perform a bottom-up recognition of characters or character-components. Work in OCRs has focused on identifying the right features and classifiers resulting in successful OCRs for the English language [9, 33]. However, despite considerable effort, robust OCRs are not available for many Indian, Chinese, Arabic and African languages. This is mostly because of the inherent complexity of the languages due to an extended character set, writing style, print variations etc. Moreover, the degradations commonly present in scanned documents result in poor character-segmentation, as well as erroneous recognition output [75, 214].

Another popular approach avoided explicit recognition of text [63, 110, 148, 183]. The concept of *Word Spotting* is used to search for occurrences of a query word in the image collection. The query is either a sub-image selected by the user, or text converted to image by rendering [63]. However, since image matching, in features space, is computationally intensive, the retrieval times using this approach are large. If N is the number of documents in the collection, and M is the number of words in each document, then, the computations required for retrieving a single query would be of $O(N.M.l^2)$ (l is length of feature vector for each word). If we assume that matching a pair of images requires 0.01 second, the retrieval time for a single query, from a collection of 21 million images, would be close to three days. As the collection of images grows, the response time increases accordingly. Thus, a purely recognition-free approach is not scalable to large collections of images and queries.

Figure 5.1 In this Chapter, we use linguistic resources available on the Web, to annotated scanned books from a large corpus [2]. Such an annotation could provide content-level access to the wealth of knowledge in the physical libraries.

In our work, we propose to combine the approaches of Word-Spotting and recognition. We shall perform the labeling of word-segments instead of working at the character level. Word-images contain a lot more context than characters, hence making them easier to match them rather than classify character-components. Even in situations where some of the characters in the words are degraded, matching to other instances of the same word may be accomplished. Further, segmentation at word-level is highly accurate since difficult character-segmentation decisions are avoided, as can been seen in the example of Figure 5.3. Another advantage with recognizing words is that it inherently performs a dictionary based post-processing.

The objective of our work is to convert word-images to text, for the sake of building a text index over the document images. However, as we shall see in Section 5.4, not all words in a document are useful for retrieval. Some words (such as stop-words) that would not be used in the retrieval process, need not be recognized at all. Unlike the task of recognition where all word-images are expected to be converted to text, we shall perform *annotation* of word-images with only those words that would help in retrieval. Now, there are two major issues with annotating word-images in document images: i) what labeling scheme to label the words and ii) how to scale the labeling scheme to large collections.

We address these challenges through a novel *Reverse Annotation* framework that we propose through this work. Unlike traditional annotation where keywords are identified for a given image, in Reverse Annotation, the relevant images are identified for each keyword. The traditional annotation process is depicted in Figure 5.5. The keyword-concepts are learnt from training data, as a feature representation for the keywords. To annotate a test image, its regions are identified, and features are extracted from each region. A distance measure is computed between the extracted features and the keyword features. This distance measure is used to obtain the annotations using a suitable classifier, say a K-nearest-neighbor.

In contrast, the Reverse Annotation framework draws inspiration from generic text retrieval approaches, where words are indexed to speed-up matching. This is achieved by performing an offline clustering of word-images, forming a pseudo-index for the words in the collection. These clusters are then classified against the learnt keyword-concepts, thus reducing the complexity of classification from the collection size to the cluster size, as depicted in Figure 5.6. Reverse Annotation is applicable in cases where the number of keywords for annotation, is far less than the number of images being annotated, which is particularly true for document images.

The main contribution of our work is that we have successfully addressed the problem of large scale document image annotation. Our approach works over those scripts which do not have an OCR, thus making it possible to retrieve from many collections that could not be searched hitherto. In spite of challenges such as: i) complex scripts, ii) degradations and iii) lack of sufficient training data, we built a document image retrieval system that answers text-queries, with instant retrieval. We show that our approach significantly scales to large document image collections, such as a 1000 Telugu book dataset, the largest such collection made searchable. The performance of the retrieval system over this collection is very high, with an average precision of 0.8. Through this work, we also perform a

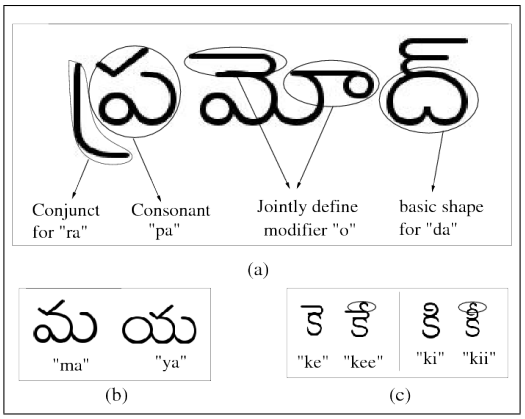
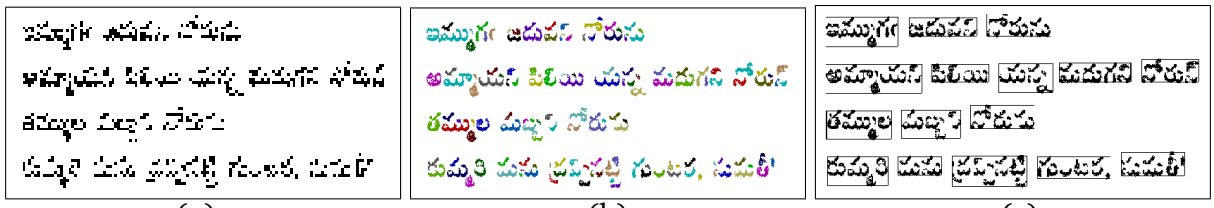


Figure 5.2 Examples demonstrating the subtleties of the Telugu language. In (a) the consonant modifier is shown to be displaced from the consonant in different ways (b) the two characters *ma* and *ya* are distinguished only by the relative size of the circle (c) the small stroke at the top changes the vowel that modifies the consonant.

thorough evaluation of various features, classifiers and indexing schemes towards word-image matching.

5.2 Data: Indian Language Document Images

The visual data that we would like to make searchable is a set of 1000 scanned *Telugu* books, consisting of 120K document images. A significant percentage of the books had been published several decades ago, resulting in severe artifacts in the pages, and consequently in the scanned images. Due to these degradations and the complexity of the *Telugu* script itself, OCRs have not been successful in enabling text-level access to such collections. However, there is large practical value for the information present in these digital libraries, as they cover a wide variety of topics in Indian languages. By providing access to these books, we could augment the information available on the web, for speakers who prefer to read and learn in their native languages.



(a) (b) (c)

Figure 5.3 (a) Example segment from a typical document image of our collection. Notice the considerable degradations, cuts, merges etc. in the passage. (b) The connected-components of the segment in (a). Each component is drawn in one color. One can observe the over segmentation of characters due to degradations. An OCR would not be able to recognize the characters in this situation. (c) Segmentation of the image at the word level. The effect of the degradations is much less severe in this case. We propose the matching of word-images in lieu of recognizing the characters.

5.2.1 Challenges

There are three major challenges toward enabling search over a large collection of Telugu document images:

Script Complexity: The character set of Indian language scripts, especially Telugu, is quite large. Each character is written as a conjoined consonant and vowel modifier. An example word, and the components of each character are shown in Figure 5.2(a). Such formatting makes it hard to segment as well as recognize the character-components. Further, there are a significant number of visually similar character pairs. Subtle changes such as the presence/absence of a dot, or a small stroke could differentiate between characters (Figure 5.2(b,c)).

Degradations: Given that the books that are scanned in digital libraries are quite old, there are significant number of degradations in the document images. These images are generally degraded due to the age of the paper, noise from scanning, etc. An example is shown in Figure 5.3(a). Characters are frequently broken during noise cleaning and binarisation. There is a also significant variety in the fonts used, typeset styles and print quality. Figure 5.4 shows the variety for one word across the collection.

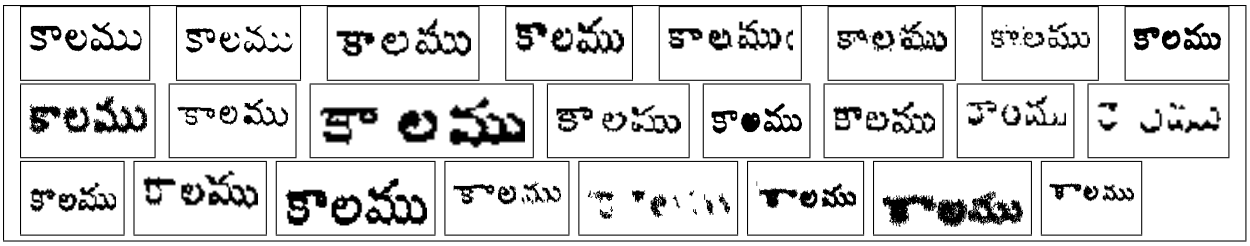


Figure 5.4 Multiple instances of a single word *kaalamu* from the book dataset. Notice the large variations in the font, and the large amount of degradations in the words.

Scale of Data: Given the large amount of data we wish to handle, every step of our algorithm needs to be designed for efficiency. A brute force method to label all words in our collection would require more than 150 years' of compute time. The scale of the data also effects the memory requirements for various tasks. The features extracted from word-segments amount to 210 GB of disk space. Since typical computers cannot handle all the data in one instant, the solution needs to exploit data parallelism whenever possible.

5.3 Related Work on Document Retrieval

5.3.1 Recognition-based

Document recognition methods have had a long history, surveys can be found in [42, 158, 166]. Most recognition based techniques have focused on recognizing character segments. Characters are represented using various features such as patch-based, PCA, LDA, or other statistical features [161]. The classifiers used to recognize characters have evolved over time. In the early days of numeral and character recognition, Artificial Neural Networks were popular such as in LeCun *et al.* [135]. Recently, Francesconi *et al.* [97] combined MLP, with a filtering step based on auto-associators. These days, Support Vector Machines (SVM) are more often used in classifier building. Since SVMs are mostly suitable for two-class classification problems, a chaining architecture is used to combine multiple classifiers, such as a directed acyclic graph in Jawahar *et al.* [62].

In cases where character segmentation is hard, due to ink degradations, word-level classifiers are better suited. Natarajan *et al.* [160] use a HMM to model characters, which are concatenated to form a word model, which is then recognised as a whole. This technique was built into the Byblos commercial OCR system. Chan *et al.* [67] uses a KPCA/LDA features for each vertical strip of a word. A gHMM, with a bi-gram letter transition model is used to infer word transcriptions. However, OCR techniques that use word-level context still rely on inaccurate component-level classification as an intermediary stage.

In the presence of (inevitable) OCR errors, post-processing is very useful to correct classifier mistakes. Kahan *et al.* [122] build a character error model, which is then used to correct frequently confused characters. Another popular approach is the use of a dictionary to rectify errors resulting in invalid words, as detailed in Lehal *et al.* [138]. Statistical language models such as n-gram models are also quite successful in reducing the OCR errors [136, 160]. But, the use of language models for post processing is quite challenging for many Indian languages like Telugu. This is because of the large number of possible words, which makes dictionaries and language models challenging to build (see Bharati *et al.* [51]).

5.3.2 Recognition-free

An alternative approach to OCR called *Word Spotting* [147] , avoids the recognition of characters altogether. In the recognition-free approach, explicit labels are not assigned to characters or words. First proposed for handwritten documents, the word-images are first represented using profile features, and matched against the query-image using a Dynamic Time Warping(DTW) based distance measure. The details of these techniques are provided in Rath and Manmatha [182, 183].

The word-spotting technique was applied to printed document images by Lu *et al.* [144], which uses a weighted Hausdorff distance between templates and test words. For Indian

scripts, Chaudhury *et al.* [68] use geometric feature graphs that exploit the structural characteristics of the Hindi script. Konidaris *et al.* [126] combines synthetic data generation with user feedback to improve performance on historic Greek documents. Profile features were used to retrieve from Ottoman documents by Ataer and Duygulu [38]. Zhang *et al.* [238] use compact features from gradient based binary features, which are then matched by a correlation based measure [238]. A thorough evaluation of features and distance metrics for word matching was performed in Meshesha and Jawahar [150].

However, matching of word-images is computationally intensive, taking on average 1 second for each pair, assuming offline feature extraction (Rath and Manmatha [185]). Searching for a query within a small collection of 1000 pages could take as much as 80 hours, which is quite unacceptable. An index built over word-images would help in faster retrieval. For example, Marinai *et al.* [148] build an index over approximate character labels, which are then used in a multi-step matching and alignment with a query image. Rath and Manmatha [185] built indexes using word-image clustering, while Kumar *et al.* [128] use Locality Sensitive Hashing to index the word-images. The limitation of indexing schemes, is their large memory requirements. An index over a large dataset is typically larger than what computers can handle.

Further, word-spotting systems can only be queried-by-example, while users prefer text querying. Text querying could only be possible by providing text labels to images. Such labels could be assigned manually as in Balasubramanian *et al.* [45] and Rath and Manmatha [185], which is obviously quite expensive. Automatic annotation of word-images are essential for scaling up the word-spotting approach, which is what is performed in this paper.

5.3.3 Dataset Sizes

One of the major claims of our work is the scalability of our approach to large datasets. Much of previous work has been restricted to laboratory settings or small document collections. Due to the unavailability of a large corpus most of the experiments were done only on a limited number of pages. The classification results were reported at character or even at sub-character level. For example, Negi *et al.* [162] report a component level accuracy of 92% on 2524 components, while Neeba and Jawahar [161] evaluate over half-million components.

The Ottoman documents used in Ataer and Duygulu [39] consists of 26 pages. Xiu and Baird [232] demonstrated their work on about 50 images. Konidaris *et al.* [126] use 100 Greek documents for their evaluation. Chan *et al.* [67] use one Arabic book to demonstrate search. The UNLV dataset [188] evaluated OCRs on a set of 2000 pages. For handwriting documents, Lavrenko *et al.* [133] and Rath and Manmatha [185] evaluate their work over 20 documents. Rath *et al.* [184] raised the bar with a 1000 page George Washington collection.

Our previous work in Pramod and Jawahar [175] was the first to build a retrieval system over a collection of 500 (printed) Telugu books, consisting of 75,000 document images. In this paper, we scale the dataset to 120,000 images, which we believe is the largest document image collection made searchable in a non-Latin script.

5.4 Reverse Annotation

Let us begin with the assumption that we are provided with a set of labeled word-images (training set), and the unlabeled words that need to be recognized (test set). We shall henceforth refer to labeled word-images as *exemplars*. Typically, one would begin by building classifiers such as ANN or SVM to classify the test set against the training

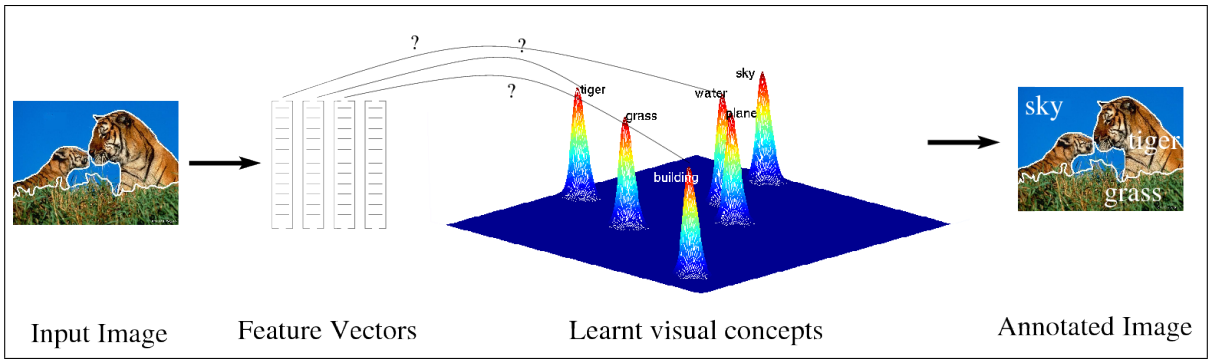


Figure 5.5 Depiction of traditional annotation approaches. Given a test image, features are extracted from its regions. These features are compared against the learnt concept models, and the keyword of the closest match is given to the region.

set. However, unlike character sets which consist of a few hundred symbols, the number of classes for word based recognition is typically hundreds of thousands depending on the language vocabulary. Training classifiers for such large number of classes requires large amount of training data, while using such classifiers in test phase could be highly computationally expensive.

Instead of expensive classifiers, we shall use a quick and reliable Nearest-Neighbor (NN) classifier to label word-images. As long as there is at least one labeled exemplar for a given keyword, the corresponding words of the test set could be labeled. The NN classifier now depends on a reliable matching scheme between the word-images of train and test sets. We carefully select the features and distance metric by evaluating them over a large groundtruth dataset (see Section 5.7). However, the classifier complexity is $O(N_1 \cdot N_2)$, where N_1, N_2 are test and training set sizes respectively.

We further observe that there is abundant repetition in the test image collection, i.e. most of the words occur multiple time across the collection. Let us assume that it was possible to *cluster the word-images* (based on their corresponding features), such that multiple occurrences of a particular word are found in one cluster. Given such clusters, it suffices to annotate one representative of the cluster, whose label is propagated to the rest

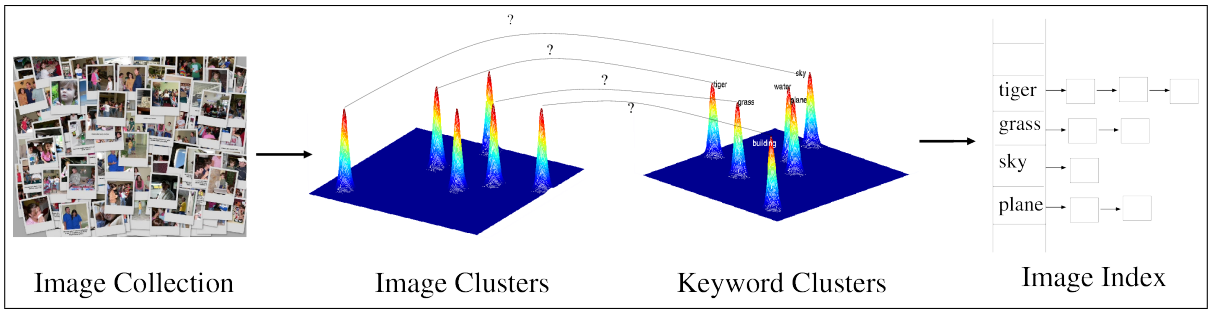


Figure 5.6 Depiction of Reverse Annotation. The features from the image collections are clustered into unknown concepts. To annotate the images, it suffices to find the correspondences between these clusters and that of the keywords. Once the correspondences are identified, the search index could be easily built.

of the collection. Since the classification is performed only once per cluster, it allows for considerable annotation speed-up.

The clustering itself can be sped-up using a hierarchy of clusters [163]. In hierarchical clustering, the data points are clustered into a small number of clusters. Each of these clusters is in-turn clustered into smaller clusters and so on. The advantage of this technique is the considerable speed up in clustering and lookup. We shall elaborate further on this in Section 5.5.

Further, such a clustering could also be performed on the keyword-exemplars as well, since the keyword-exemplars themselves are not totally isolated. *Keyword-exemplars can also be clustered* based on the similarity in their representative features. These clusters can now be matched at the corresponding levels of the hierarchy, such that clusters at a lower level are only matched if their respective parents are close. This technique is depicted in Figure 5.7. The complexity of matching exemplars with features is now reduced to $O(logN_1 \cdot logN_2 \cdot k^2)$, where k is the branching factor for the cluster hierarchy.

It is important to note that a cluster is labeled with a keyword, only if the NN-distance is less than a pre-defined threshold. Thus, it is likely that certain clusters, would not be given any label at all; such as those words with no exemplars in the training set. Unlike traditional auto-annotation, which predicts labels for the given images, our framework

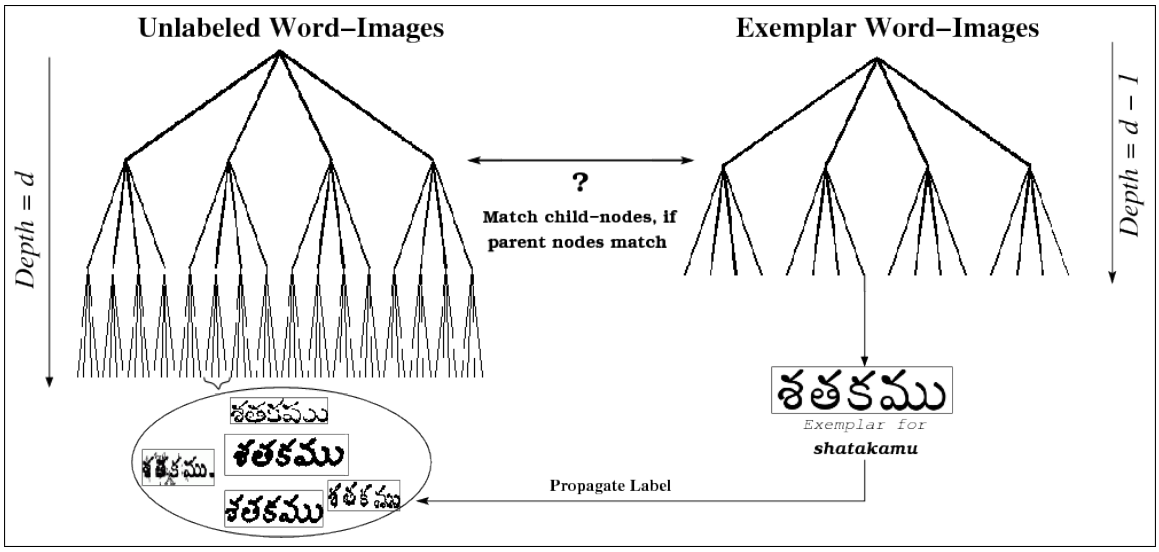


Figure 5.7 Reverse Annotation by efficient comparison of hierarchical clusters of labeled and unlabeled word-images.

$k_1, k_2, ..., k_n$	keyword exemplar cluster
$k_{i_1}, k_{i_2}, ..., k_{i_{n'_i}}$	points in cluster k_i
$t_1, t_2, ..., t_m$	word-image clusters (centroid word-image)
$t_{l_1}, t_{l_2}, ..., t_{l_{n''_l}}$	word-images in cluster t_l

Table 5.1 Naming convention for the Probabilistic Reverse Annotation framework.

predicts the images (or image regions) that correspond to the given label. Hence the term *Reverse Annotation*.

5.4.1 Probabilistic Reverse Annotation

At the end of Reverse Annotation procedure, we have an index of the word-images against the keywords. However, the index does not lend itself to rank the word-images for retrieval purposes, since the associations in the index are binary. For this purpose, we extend the framework to *Probabilistic Reverse Annotation*, where we estimate the probability that each word-image belongs to the keyword.

Let us follow the naming convention presented in Table 5.1. The probability consists of two parts. The first part measures the quality of labeling of a cluster (i.e. its centroid) with the given keyword-exemplar. The second term measures the association of the word-images to the cluster. The probability p_{ij} that word-image t_{l_j} matches keyword k_i is given as a product of these two terms:

$$p_{ij} = \frac{s(k_i||t_l)}{\sum_i \sum_l s(k_i||t_l)} \times \frac{s(t_l||t_{l_j})}{\sum_j s(t_l||t_{l_j})} \tag{5.1}$$

where $s(x||y)$ is a similarity measure (inverse of a distance measure). The denominator of the first term involves the summation over all keywords and all clusters. While computing the denominator would be expensive, we observe that it remains a constant for all word-image and keyword cluster pairs. We can hence ignore the denominator while computing the score, except it would not be a value with range $[0, 1]$. The computation of the score can thus be performed only between the pairs of exemplar and test-set clusters which match in the Reverse Annotation phase, hence preserving the computational advantages of the framework.

5.4.2 Ranking of Retrieved Documents

The final outcome of the probabilistic reverse annotation step, is a ranking of word-images against their corresponding keywords. However, in document retrieval tasks, one needs to rank the entire document image against the given query. We rank documents using a ranking function similar to the Term Frequency/Inverse Document Frequency (TF-IDF) measure. In our ranking function, the sum of number of words is replaced with their probabilities given by the Reverse Annotation step. For the i-th query-word the TF is defined as,

$$tf_{ij} = \frac{\sum_k p_{ik}}{\sum_{l,k} p_{lk}} \tag{5.2}$$

where p_{ik} is the probability score for the k-th word-image to correspond to keyword t_i. TF measures the importance of the keyword for the image, and is normalized by the document length to avoid bias to longer documents. The document length is replaced by the sum of probability scores of all the words within the document, against their respective keywords. This scheme ignores the words that are not labeled in the annotation step.

When the search query consists of multiple words, the relative importance of the words in the query is an important distinguishing factor. This is estimated by the *inverse document frequency* (IDF) measure, which indicates the overall importance of the given keyword in the entire collection. The IDF measure is defined as

$$idf_i = log \frac{1}{\sum_{images} \sum_{l,k} p_{lk}} \tag{5.3}$$

It is basically the logarithm of the total number of documents over those containing the given term. In our case, it is estimated as the inverse of the sum of the cumulative probabilities over all the images. The IDF is used to normalize the TF value across all the images.

5.5 Efficient Implementation of Reverse Annotation

Our framework is built over the clustering of test-data, where multiple occurrences of the same word are grouped together. This is typically performed using a K-Means algorithm over the features of the words. However, such a clustering would be more computationally expensive than direct classification. Scalability to 1000 books or more required that we look at alternative approaches to clustering. We shall look at three such approaches, namely: i) Hierarchical K-Means, ii) Locality Sensitive Hashing and iii) KD-Trees.

5.5.1 Hierarchical K-Means

One of the limitations of the K-Means algorithm is the necessity to fix K, beforehand. If K is less than the number of unique words in the collection, it would mean that some of the clusters would have different words in them. To avoid different words being clustered together, we would prefer a large K. This ensures that each cluster contains instances of the same word, while allowing multiple clusters for the same word. However, this clearly increases the compute time.

Hierarchical K-Means (HKM) is a way to approximate K-Means fast. The idea [163] is that a small number of clusters - say K - are created at the top level. K is known as the branching factor. Then each of these K clusters is expanded to K more clusters giving K^2 clusters at the second level. This process is repeated up to a certain depth D so that there are K^D clusters or leaf nodes at depth D. Given a new point, to find out which cluster it belongs to, it takes $K \cdot D$ comparisons unlike traditional K-Means which would take K^D comparisons. For example, if $K = 10$, $D = 6$ then there are a million leaf nodes. K-Means requires 10^6 (a million) comparisons while HKM only needs 60 comparisons. To build the entire HKM tree for a dataset of size N requires $O(N \cdot K \cdot D)$ time while K-Means would require $O(N \cdot K^D)$. On a modern desktop processor the difference is 31 yrs for K-Means vs 13 hrs for HKM. For our experiments here we usually chose $D = log_K(N)$.

5.5.2 Locality Sensitive Hashing

In Locality Sensitive Hashing [116] (LSH), data points are projected onto random subspaces, such as a line or hyperplane. The projected subspace is divided into spatial bins; each point is assigned to the bin it falls into. The assumption is that points close to each other in a high dimension space will most likely fall into the same bin. Multiple such hash functions are generally used in collecting additional evidence to determine NNs.

A d dimensional word feature is mapped onto a set of integers by each hash function $h_{a,b}(x)$. Each hash function is indexed by a choice of random a and b. Here, a is a d-dimensional vector with entries chosen independently from a p-stable distribution and b is a real number chosen uniformly from the range $[0, w]$. For a fixed a, b the hash function $h_{a,b}$ is given by,

$$h_{a,b}(x) = \lfloor \frac{a \cdot x + b}{w} \rfloor \tag{5.4}$$

Generally $w = 4$. The dot product $a \cdot x$ projects each vector onto a real line. This line is chopped into equi-width segments of appropriate size w and hash values are assigned to vectors based on which segment they project onto. Algorithmic details are given in [101, 116].

The storage complexity of LSH is of the order $O(n.|H|)$, n being the number of features hashed and H is the set of hash functions. Since the optimal $|H|$, also depends on the size of the dataset, the memory required for a large dataset could be more than what modern computers can handle [35]. The running time complexity for each query consists of two parts: T_c and T_g. T_g is the time required to compute the hash values for the query and to identify the buckets it falls into; it is of order $O(d \cdot k \cdot |H|)$. T_c represents the time for computing the distance to all points encountered in the retrieved buckets; T_c is equal to $O(d|collisions|)$, where $|collisions|$ is the number of points encountered in the buckets corresponding to the query.

5.5.3 KD-Trees

Another approach to address the approximate NN problem uses KD-Trees [157]. A KD-Tree partitions the feature space with axis-parallel hyperplanes. The algorithm splits the data in half at each level of the tree on the dimension for which the data exhibits the greatest variance. When multiple randomized trees are built, the split dimension is chosen randomly from the first D dimensions on which data has the greatest variance. Multiple

trees define an overlapping split of the feature-space. These trees are looked up for NNs, by comparing the query with the bin-boundary at each level of the tree(s). The stopping criterion for the search is determined by parameters set by the user.

Building a KD-tree from n points takes $O(n\ log\ n)$ time, by using a linear median finding algorithm. Inserting a new point takes $O(log\ n)$ and querying for nearest neighbors takes $O(n^{1-1/k} + m)$, where k is the dimension of the KD-Tree and m is the number of nearest neighbors

For KDTrees and HKM, we use the FLANN software provided by [157]. In the implementation, the algorithm first traverses the HKM or KD-tree and adds the unexplored branches in each node along the path to a priority queue. It then finds the closest center in the priority queue to the given query, and uses this node to restart the traversal. The process is stopped when a predetermined number of nodes are visited.

5.5.4 Indexing Issues: Memory Limitation

One of the major hurdles with using the above indexing schemes over large datasets, is that the memory requirements are much larger than what modern computers can handle. To illustrate, the features for the 36M words of our dataset add up to a size of 210 GB in floating point numbers. Building an index over this entire feature set is not possible with existing techniques/code. To simplify this issue, we build indexes over smaller datasets that can be loaded into the RAM of existing machines. It turns out that indexes could be build over features for each subset of 20 books, obtaining 50 indexes in all. The matching of unlabeled word-images with labeled exemplars can be accomplished by lookingup each of the exemplars against these indexes for approximate nearest neighbors.

5.6 Obtaining Labeled Exemplars

Obtaining training data, in the form of labeled exemplars is essential to bootstrap the recognition process. The obvious solution is by manually labeling some portion of the collection. Normally, manual labeling is very expensive, when required at the word-level. Instead of requiring manual labeling of individual word-images, we obtain transcriptions of the document images. Manual typesetting is much quicker when performed at the page-level, rather than at word-level.

The page-level transcriptions are *aligned* with the word-images. This is achieved by segmenting the document image with the information regarding the number of lines and words. The process begins by performing line-segmentation of the document image. If the number of lines from the segmentation is less than the number of lines in the text, it means two text lines have been merged by the segmentation. This often happens due to overlapping ink pixels between the two lines. The tallest lines are thus split, until the number of lines matches between the image and the text. On the other hand, if the number of lines from segmentation is less than those in text, then the smallest lines are merged together. In the second step, each line is similarly segmented into words depending on the number of words in the given line. Exemplars for the keywords are obtained from this labeled dataset. Some examples for one keyword, *kaalamu*, obtained through this method are shown in Figure 5.4.

In case of rare-words, the transcripted data would not yield sufficient labeled exemplars. We address this issue by generating synthetic examples by font-rendering. Unlike our previous work [175], where keyword exemplars were obtained using a single font, in this work, we use *multiple* Unicode fonts to better model the variety of fonts in the dataset. The obtained images are then degraded using the Kanungo degradation model [123]. Rendered exemplars for the keyword *shlookamuloo* are shown in Figure 5.8.

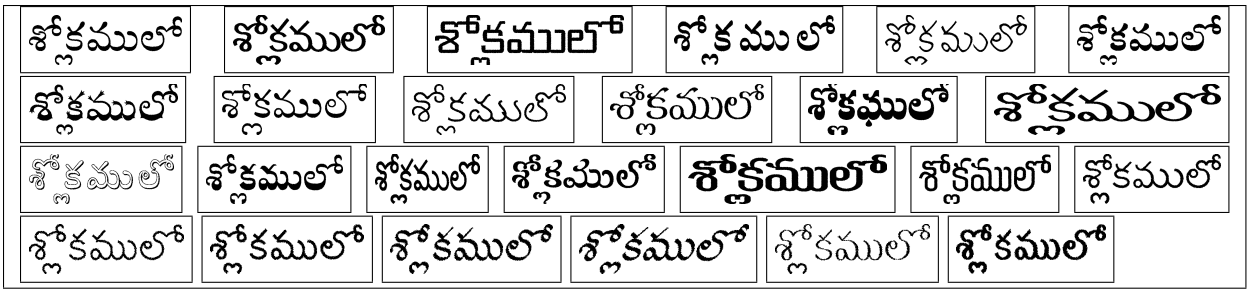


Figure 5.8 Exemplars generated from font rendering for the keyword *shlookamuloo*. Word-images rendered using multiple fonts are degraded using the Kanungo degradation model [123].

Other approaches too, could be explored to obtain keyword exemplars. One such approach is to obtain labeling of word-image using reCAPTCHAs [223]. In a reCAPTCHA, pairs of word-images are presented for recognition by humans. The label for one of the words is known while that of the other is not. If the human recognizes the known image, it is assumed that their judgment about the other image is also valid. The labor provided by humans is essentially free as reCAPTCHAs are required by many websites for sign on. Another approach is to use a crude OCR that uses a rejection strategy so that it recognizes only those words that it is confident about. Whenever it has some doubts, it decides not to recognize. The word accuracy of such an OCR may be poor say 50% or less. However, the exemplars from these approaches could be noisy.

5.7 Feature Selection

Our dataset consists of a wide variety of font-styles, font sizes, and large variation in the amount of degradations. The features chosen for word matching need to be robust to this variety, some of which can be seen in Figures 5.4, 5.8. We examine three types of features: i) Profile-based, ii) SIFT based and iii) PHOG based. Profile features have been previously used for representing word-images [180, 183]. On the other hand, gradient based features

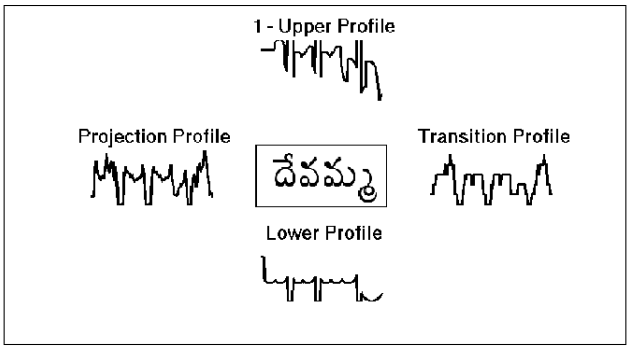


Figure 5.9 Profile features extracted for word images include the Upper, Lower, Transition and Projection profiles. This signal is converted to a fixed length representation by converting them to the Fourier domain where the higher order frequencies are picked as the word representation.

have been popularly used in computer vision [76, 143]. They have also been claimed to perform better than Profile features for handwritten word-images [190].

Profile Features: Profile features were popularized by Rath and Manmatha [183], where these features were used to represent words from handwritten documents. The profile features we extract include (see [183] for more details):

- The Projection Profile is the number of ink pixels in each column.
- Upper and Lower Profile measures the number of background pixels between the word and the word-boundary
- Transition Profile is calculated as number of ink-background transitions per column.

Profiles for an example word are shown in Figure 5.9. Each profile is a vector whose size is the same as the width of the word. The dimensionality of the feature, therefore, varies with the word used. One approach to matching these uses Dynamic Time Warping (DTW) [45, 183].

Profile + DFT: Fixed length representation could be obtained by scaling the word images to a canonical size before extracting the Profiles. However, this distorts the inherent aspect-ratio of words. A more principled method is to compute a Discrete Fourier Trans-

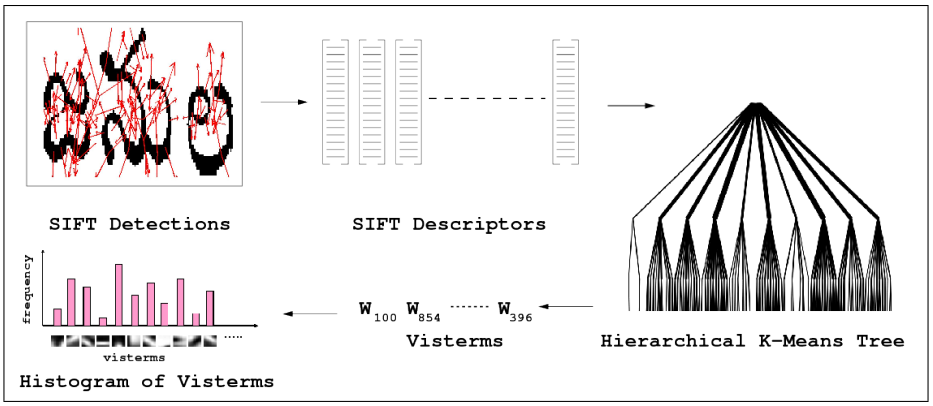


Figure 5.10 Representation of SIFT features in the Bag-of-Words model. Extracted SIFT features from a word image are assigned to the closest *visterm*. This is performed efficiently using Hierarchical K-Means. The word is then represented as a histogram of its constituent visterm occurrences.

form (DFT) of the profiles [128, 184]. The noisy higher order coefficients of the DFT can be discarded, resulting in a robust representation for the word-images. We use 84 Fourier coefficients for each of the profile feature. With this representation, word-images can be matched by comparing feature vectors using a simple distance metric such as Euclidean or Manhattan distances.

SIFT + BoW: An alternative to Profile features is to use a point-based representation for the word-images. SIFT [143] (Scale Invariant Feature Transform) has proved to be a very robust interest point detector and feature descriptor for many computer vision tasks [163]. The SIFT operator contains two parts - an interest point detector and a descriptor. The interest point detector is based on the difference between multi-scale Gaussian of the given image. The scale-invariance property of SIFT and its high repeatability across various affine transformations makes it a good candidate to be used in document images [39].

The pairwise matching of SIFT features between word images is time consuming. To alleviate this, the features are mapped to a unique ID, called the *visterm*. Images can now be matched by counting the overlap between corresponding visterms. The visterms are obtained by vector quantizing the SIFT features using K-Means clustering. Each feature in

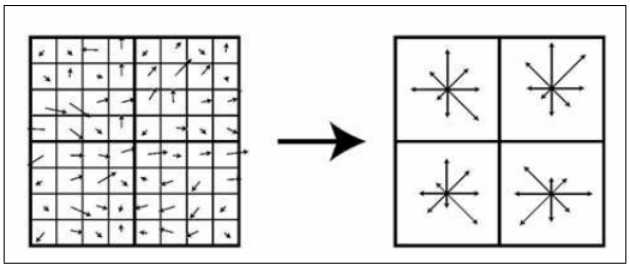


Figure 5.11 A depiction of HoG feature extraction. A 2x2 grid is shown for clarity. In the implementation, a 4×4 grid is used.

a word-image is represented as the index of the cluster visterm. The word-image is then represented as a histogram of the occurrences of each visterm.

For example, if an image has $F_1, F_2, ..., F_{300}$ features, and say the visterm size is 1000. Each feature in the image is assigned to the closest visterm, say $W_{100}, W_{200}, W_{150}, W_{240}, ..., W_{100}$. The number of times a visterm occurs in a word, is stored in a histogram of length 1000. The histograms are then normalized by number of features in the word image. Given this representation, two words are compared by finding the Euclidean distance between the corresponding histogram feature vectors. Since the geometric configuration of the features is ignored and only their incidence is taken into account, it is called a Bag-of-Words representation of the image. This process is depicted in Figure 5.10. Such a representation reduces the time required for explicit match of SIFT features, when matching two images.

PHOG + BoW: Unlike SIFT, which is a sparse detector, one could use a dense representation with similar descriptors. A histogram of oriented gradients (HoG), first proposed for pedestrian detection [76], was recently used for handwritten documents by Rodriguez and Perronnin [190]. A pyramidal HoG at multiple scales was shown to be much more effective in object recognition tasks [54]. In this technique, a fixed size window is moved across the word-image. At each instance of the sliding window, a HoG descriptor is computed similar to the one shown in Figure 5.11. Once the entire word-image is scanned by the window, the window size is doubled and the process repeated. The process is stopped when the size of the window exceeds the smaller dimension of the word-image. The set of

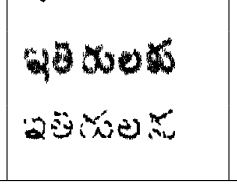
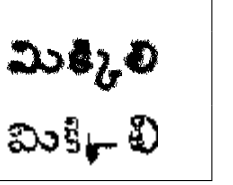
ban'gaaru	itarulaku	mikkili
		

Figure 5.12 Examples from the groundtruth dataset. Notice the severe degradations, noise, apart from the variations in the printing. The text label for the words is given on the top.

extracted features are vector quantized, and the word-image is represented as a histogram of the occurrences of each visterm.

In detail, the initial window size is 16×16 pixels, which is moved across the image by 8 pixels. The 16x16 window is divided into a grid of size 16 with cells of size 4×4 pixels each. The gradients at each pixel are quantized into one of 8 orientation bins. The oriented gradients in each grid are accumulated, and the histograms for all the grids are concatenated together. With this setting, we obtain a 128 dimension representation for each window. In the next iteration, the window size is doubled to 32×32, and moved across by 16 pixels and so on. To ensure that the feature length for PHOG is fixed, all the word-images are initially scaled to a standard size.

5.8 Experimental Results

5.8.1 Groundtruth Dataset

We build a groundtruth dataset of 33,000 word-images, collected from 33 Telugu books. Each of these word-images is labeled with the corresponding text label. The label set consists of 1000 Telugu words represented in the Latin script using the OmTrans transcription scheme. The number of occurrences of each label in the groundtruth dataset varies from

5 to 500. The size of the word-images range from 30×30 to 500×300 pixels. The dataset contains degraded images, and considerable variation in font and print style. Examples of such words in the dataset are shown in Figure 5.12. There is also a certain amount of noise in the word-images.

5.8.2 Feature Evaluation

The goal in this section is to identify the right features to match word-images. The summary of feature evaluation results is given in Table 5.2. The groundtruth dataset is divided into two sets: $Train$ and $Test$, with a random 50:50 split. The $Test$ set is considered to be unlabeled words, that need to be annotated (the labels are only used for evaluation). The labeled data could be either of i) the $Train$ set (referred to as *Labeled Samples* in Table 5.2), or ii) templates generated by font-rendering. The classifier is either an NN classifier or an SVM.

The first feature we consider is the classical Profile features. These features are of unequal length across the dataset. The matching in such cases is performed using DTW (Dynamic Time Warping). The score from the DTW matrix is normalized by the length of the backtrack path of the DP array. Though the result from this matching is very good (81%), DTW has the disadvantage of high time complexity. The recognition of the $Test$ set alone takes around 75 hours. This expense can be avoided by using a fixed-length description for each image, so that the Euclidean distance may be used to compare the word-images quickly.

Fixed-length representations were evaluated for both profiles from scaled word-images and from Profile+DFT. With either representation, the time for recognizing the $Test$ set is about 4 hours. For SIFT and PHOG representations for word-images, each word-image is represented as a histogram of visterms and matched using an NN classifier or with an SVM.

Feature	Training Data	Classifier	Distance	Accuracy
Profile Features	Labeled Samples	Nearest-Neighbor	DTW	81%
Profiles (Scaled)	Labeled Samples	Nearest-Neighbor	Euclidean	68.8%
Profiles + DFT	Labeled Samples	Nearest-Neighbor	Euclidean	78.6%
Profiles + DFT	Labeled Samples	Nearest-Neighbor	Manhattan	**82.8%**
Profiles (Scaled)	Labeled Samples	SVM	Linear Kernel	54.1%
Profiles (Scaled)	Labeled Samples	SVM	Polynomial Kernel	48.6%
Profiles + DFT	Labeled Samples	SVM	Linear Kernel	58%
Profiles + DFT	Labeled Samples	SVM	Polynomial Kernel	46.8%
SIFT (1K visterms)	Labeled Samples	Nearest-Neighbor	Euclidean	26.4%
SIFT (100K visterms)	Labeled Samples	Nearest-Neighbor	Euclidean	8%
PHOG (100 visterms)	Labeled Samples	Nearest-Neighbor	Euclidean	35.2%
PHOG (1000 visterms)	Labeled Samples	Nearest-Neighbor	Euclidean	41.8%
PHOG (100K visterms)	Labeled Samples	Nearest-Neighbor	Euclidean	14.1%
PHOG (100 visterms)	Labeled Samples	SVM	Linear Kernel	20.8%
PHOG (1000 visterms)	Labeled Samples	SVM	Linear Kernel	40%
Profile Features	One-font Template	Nearest-Neighbor	DTW	29%
Profiles (Scaled)	One-font Template	Nearest-Neighbor	Manhattan	38%
Profiles + DFT	One-font Template	Nearest-Neighbor	Manhattan	33.3%
Profiles + DFT	28-fonts Templates	Nearest-Neighbor	Manhattan	54.5%

Table 5.2 Word Recognition accuracy across various features and matching schemes. The best recognition accuracy was obtained using Profile features scaled using DFT. Nearest neighbor classifiers have outperformed the more advanced Kernel-SVMs. The Manhattan distance seems to work better than Euclidean distance for the Profile feature representation. Further, the matching against exemplars from a small labeled dataset result in much better accuracies than that over a large set of synthetic exemplars.

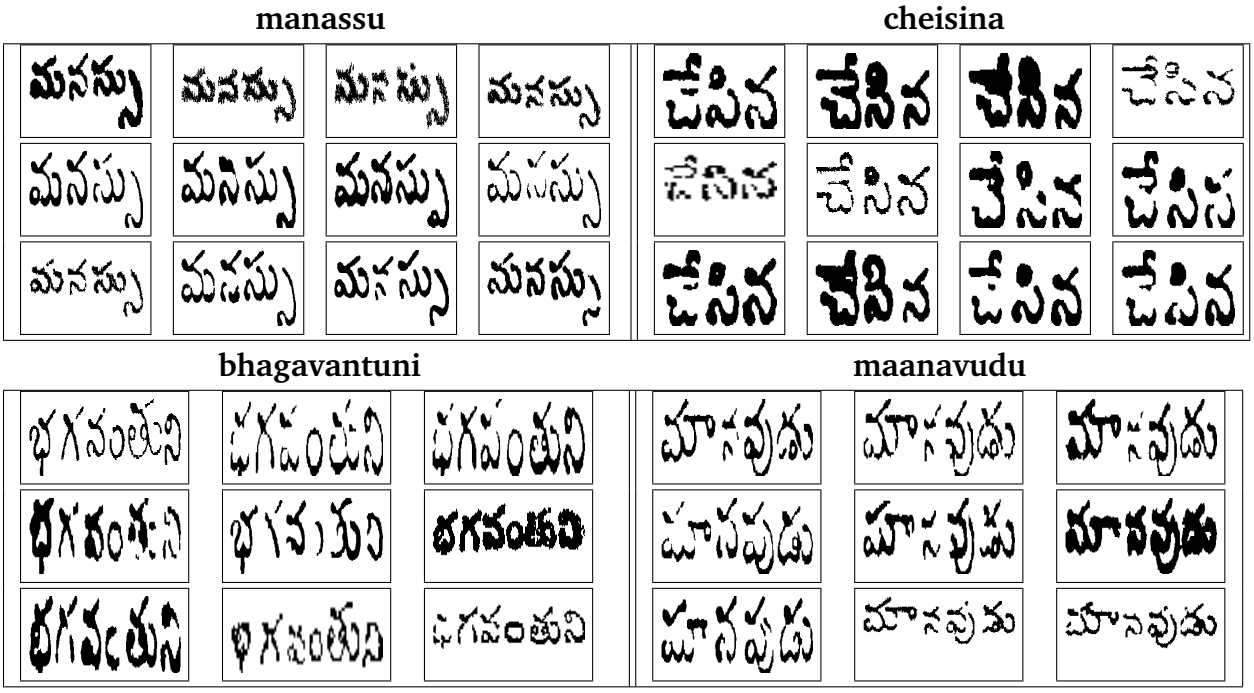


Figure 5.13 Example recognition results from the book collection using our method. Notice that in spite of severe degradations such as cuts and merges, they were correctly recognized. An OCR would easily fail over all these images.

As Table 5.2 shows the best performing feature was the Profiles + DFT with L1 distance matching. Surprisingly, the gradient features - SIFT and PHOG - perform rather poorly in spite of their great success with generic vision tasks. Rodriguez and Perronnin [190] show that for word spotting in handwriting, gradient features work better than profile features. However, our results are of the contrary. We believe that this is because of the large amount of degradations present in our document images which drastically affect the oriented gradients. The profile features are thus more robust to degradations. We shall use the Profiles + DFT features for the rest of this paper.

From our experiments, we further observe that given the same features, an SVM always performs poorer than a Nearest Neighbor classifier. The possible reason for this behavior could be that the kernels we used were not quite suitable for transforming the feature space of word-images. Another disadvantage with SVMs is their high computational cost.

We refrain from further evaluation of kernels and limit ourselves to the NN classifier which gives us satisfactory performance in lesser time.

Example results from our recognition module are given in Figure 5.13. As we observe, a large number of words are correctly recognized in spite of heavy degradations. We benefit from the presence of similar looking labeled exemplars in the training set. In cases where our procedure fails, the features are unable to distinguish between different words.

Finally, using the labeled samples as training data always performs better than that with font-rendered templates. However, by using multiple fonts to render exemplars, the accuracy of recognizing the test data is about 55%. This means that rare-words, which do not have sufficient examples in the manually-labeled data, could also be recognized with decent accuracy.

5.8.3 Performance of Indexing Schemes

Our next experiment evaluates the performance of various indexing schemes for efficient Reverse Annotation. The evaluation results from this experiment are given in Table 5.3. The features used here are Profiles + DFT, and the distance measure is either Euclidean or Manhattan. As can be seen, the indexing schemes give tremendous speedup of over 500 times as compared to a brute-force NN classifier. The loss in accuracy from the use of indexing is quite acceptable for such a speedup. The best performing scheme is the composite of KD-Tree and HKM, used with the L1-Norm. We shall use this particular indexing scheme for Reverse Annotating our 1000 books collection.

5.8.4 Effect of Labeled Data Quantity

One of the questions we address in this work is what amount of labeled word-images, is required to reliably recognize unlabeled word-images. Unlike a 50:50 split that was used in the previous experiments, we shall vary the $Train$, $Test$ proportions here. We stick to

Matching Scheme	Distance	Accuracy	Time
NN Classifier	Euclidean	78.6%	4 hours
NN Classifier	Manhattan	82.8%	3.5 hours
HKM ($K = 8$)	Euclidean	76.5%	31 secs
HKM ($K = 32$)	Euclidean	75.1%	38 secs
HKM ($K = 8$)	Manhattan	77.5%	2 m 25 secs
HKM ($K = 32$)	Manhattan	77.2%	3 m 19 secs
KD-Tree ($T = 8$)	Euclidean	69.4%	28 secs
KD-Tree ($T = 32$)	Euclidean	72.9%	41 secs
KD-Tree ($T = 8$)	Manhattan	71.5%	24 secs
KD-Tree ($T = 32$)	Manhattan	75.2%	45 secs
LSH	Euclidean	78.3%	18 m 14 secs
LSH	Manhattan	80.1%	16 m 38 secs
HKM + KD-Tree	Euclidean	76.8%	43 secs
HKM + KD-Tree	Manhattan	80.3%	3 m 3 secs

Table 5.3 Word Recognition accuracy using various indexing schemes.

the Profiles + DFT features and the Euclidean distance based NN classifier as well as the HKM based approaches with the two parameters in Table 5.3. The recognition accuracy for different percentage of training data is shown in Figure 5.14.

For the NN classifier, it can be seen that for a labeled dataset of as small as 20%, the recognition performance is a respectable 71%. The performance improves with additional data until it tapers around 76.5% (for the 50:50 split), after which there is no significant improvement. For the HKM approach with $B = 8$, we can achieve 70% accuracy with just 30% of the data.

5.9 Performance of Word-Retrieval

To evaluate our word-retrieval performance, we propose using an approximate estimate of average precision. Popular Information Retrieval evaluation measures such as precision,

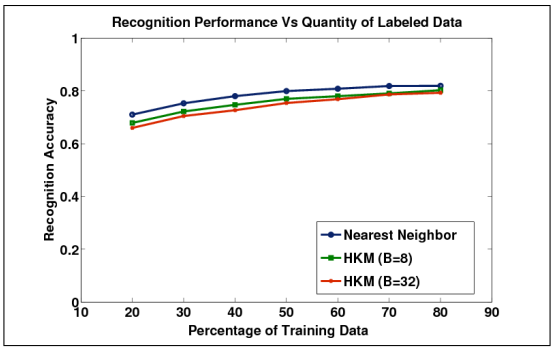
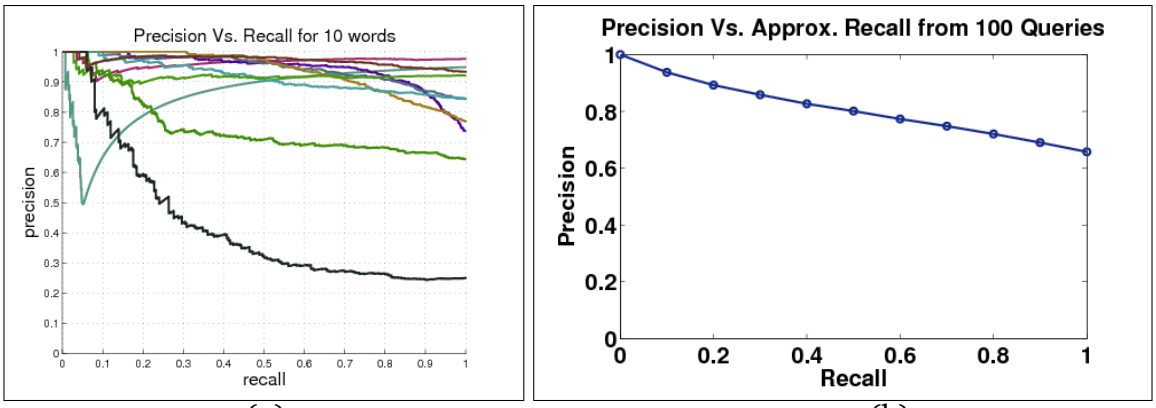


Figure 5.14 Graph of % Training data Vs Recognition performance.

recall and f-measure, assume that the number of relevant documents is known in the collection. However, given our massive collection, it is impossible for us to obtain the exact recall value for the queries.

We manually label only the top-1000 retrieved results, as relevant or irrelevant to the query. By labeling the search results, we are effectively evaluating results over the entire collection; instead of evaluating over a small groundtruthed subset, like in previous works. The number of true-positives in the top-1000 are considered as an approximation of the recall value. Hence, when the top-1000 retrieved results are considered the Recall value is always 1. This is very similar to the evaluation performed in the TREC web retrieval challenge [10]. We evaluate the Precision across various values for this approximate recall. The PRCurves for 10 example words are shown in Figure 5.15(a). It can be seen that among the top 1000 retrieved results, the relevant words are retrieved quite accurately.

The Average Precision (AP) is computed as the average of precision at each relevant retrieval for the given query. The Mean Average Precision (mAP) is the mean of the AP for multiple queries. The PRCurve averaged over 100 queries is shown in Figure 5.15(b), from which the mAP was calculated to be **0.8**. This is a significant achievement, and we are unaware of any other approaches that can generate a similar result for our data. Some example word retrieval results are shown in Figure 5.16. The words outlined in green are



(a) (b)

Figure 5.15 (a) Precision Vs. Approx.Recall for 10 queries evaluated on the top-1000 retrieved results. We can see that the Precision is generally high across various values of Recall. (b) Precision-Recall curve evaluated on the top-1000 retrieved results for 100 queries.

correct results, while those in red are errors. It is evident that the top results are mostly correct for the queries.

A few qualitative examples of document retrieval are shown in Figure 5.18 and Figure 5.17. In Figure 5.18, books are retrieved by querying for the title, in this case books consisting of a particular set of poems. The Figure 5.17 shows specific poems retrieved by querying the first line. The retrieved documents in this example also contain full transcription and explanation of these popular poems, which would be quite useful to the users.

Computational Time

The superiority of our method is further highlighted by the reasonable time it takes for enabling search. The times taken by each step of the pipeline are given in Table 5.4. The total compute time for processing the 1000 books was close to 2700 Hours. The process can be easily adapted to work incrementally, thus making our method easily scalable to large collections.

Figure 5.16 The retrieved word-images for a few example queries. These results are obtained from our framework over a large 1000 book collection. The words relevant to the query are marked in green and errors are marked red. The top-30 results for the queries *jnj-ananamu* and *maatramei* are all correct retrievals. A few errors can be seen for the query *sharand-u*. It is evident that our method overcomes many of the challenges inherent in the problem.

Aspect of the Pipeline	Time Taken
Segmentation	42 Hours
Feature Extraction	2140 Hours
Feature file pre-processing	3.5 Hours
Indexing and Lookup	445 Hours
Post-processing	58 Hours
Total	2688.5 hours

Table 5.4 Time consumed by each stage of the pipeline; and the total time for making 1000 books searchable.

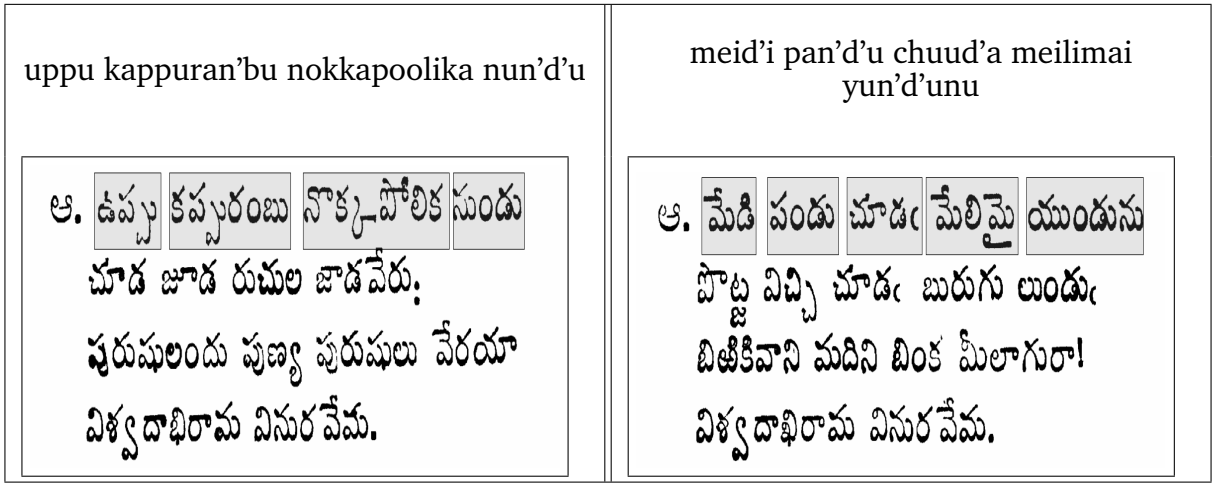


Figure 5.17 In this example, two popular poems are retrieved by querying their first line.

5.10 Summary

In this Chapter, we have presented a successful framework to annotate a large collection of document images with text-labels obtained from the language's vocabulary. The method we propose is applicable to similar instances where the segmentation of the visual and textual entities is known, but there are no temporal constraints to exploit (like in the previous Chapters). Instead, we leverage the occurrence statistics to avoid repetitive evaluations of correspondence, which is further speeded-up with indexing schemes. As an outcome of

this approach, we were able to build a retrieval system on 1000 Telugu books, that could not be converted to text with an OCR.

Further, the Reverse Annotation framework could be directly extended to build a retrieval solution over a mixed library of documents across various languages and scripts. Towards disambiguating scripts that are visually similar (e.g. Hindi and Bangla), more robust features such as HoG features computed over vertical strips [102], could be used instead of the Profile features. In fact, any feature representation that could be matched in a metric space could be used within the Reverse Annotation scheme.

While this Chapter, addressed the annotation at the word level, in the next Chapter, we delve into a finer granularity called Character N-Grams (CNG). The limitations placed by the lexicon, shall be overcome with the CNG labeling. The resulting search mechanism that can answer queries from an unlimited vocabulary.

Figure 5.18 Example retrieval results for queries based on book titles. Queries are given in a *transliteration* format called OmTrans, which is a Latin script representation for Telugu characters. Here we query for a set of Telugu poems, given the names *veinkateswara shatakamu* and *shrii kaalahastiishwara shatakamu*. Multiple books are retrieved from the 1000 book collection, that are relevant to the given query, a few sample document segments are shown here.

Chapter 6

Alignment and Annotation of Document Images at Sub-Word Level

6.1 Introduction

In the previous Chapter, we showed a mechanism to obtain text-labels for word-images of scanned documents. While this is a big stride forward towards recognition-free retrieval, the word-level annotation can address only a limited vocabulary that it was trained for. Further, word-spotting and word-annotation are not directly applicable to partial word matching of a query with prefixed/suffixed words in the collection. For example, let us assume that the document collection has an instance of *bicycle*, while the query is *cycling*. Ideally, the document containing *bicycle* is relevant to the given query, and should thus be retrieved. However, that would not be possible with a regular word-spotting/annotation setup. This could possibly be addressed using a DTW based partial matching [45]. But, such techniques are not scalable to large collections due to the high computational complexity, especially at run-time.. Typical partial matching on even a few thousand document images could require many hours of computing time per query, thus making it infeasible for practical applications. Language models, that could be applied for stemming text-words are hard to build for Indian languages [51].

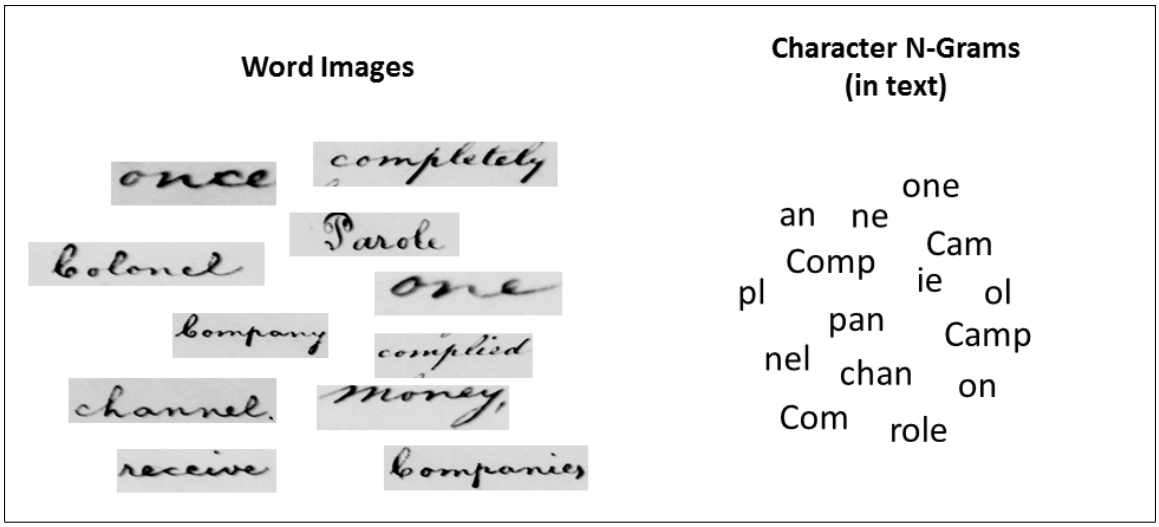


Figure 6.1 Word images extracted from handwritten documents are presented alongside some of their Character N-Grams. A reasonable size CNG-dictionary could cover an unlimited word-vocabulary. In this Chapter, we show that CNG is a novel and useful primitive to represent document images with.

In this Chapter, we overcome the limitations of the approaches that operate either at component level or at word-level, by proposing a new retrieval/recognition primitive based on Character N-Gram Images (CNG-Img) [80, 212]. CNG-Imges are formed as sequences of character segments from a given word-image. This formulation is different from text n-grams which are used to provide a statistical prior on character labels [109]. The CNG-Img are represented and matched entirely in the image space. Due to the representative capability of CNG-Img, the system allows for retrieving morphologically similar words also. A small set of words, along with some of their CNGs are depicted in Figure 6.1. It can be observed that several words contain common sub-strings, making it possible to represent a large vocabulary with a small set of CNGs.

In the text classification community [65], n-grams over words have been well studied. However, their use in OCR systems has been limited [208], mostly due to the fact that the frequency information is not decisive enough to disambiguate the confusing recognition. In another study [88], n-grams of confused characters in words are extracted and processed

using IR methods to correct errors. Unlike previous n-gram approaches that mostly used n-gram frequencies for post-processing, we shall rather exploit the joint appearance of characters in n-grams for improving the matching itself.

Using the CNG-Img as a retrieval primitive, a retrieval system i) can retrieve morphologically similar words without explicit stemming, ii) answers queries from an almost unlimited vocabulary, iii) ensure the retrieval is robust to degradations such as cuts and merges. However, to build a text-based retrieval system (or QBK), we need to either i) obtain labeled exemplars for text CNGs or ii) recognize the CNG-Img. We address both possibilities in this Chapter.

For the first approach, we begin with the word-level labeling achieved from the previous Chapter. Given word-labels, the task is to infer the labeling at the sub-word or the character n-gram (CNG) level. We propose a weakly-supervised scheme for a simultaneous CNG-Img segmentation and annotation using a framework similar to expectation-maximization (EM).

Following the process of obtaining labeled exemplars, we present a recognition scheme that involves two steps: i) the recognition of individual CNGs in a given word image, and ii) inference of the word label given the labels for each of the CNGs. The recognition scheme using CNGs has shown to outperform both character and word based recognition approaches. We demonstrate the approach on both printed and handwritten document images from multiple languages.

6.2 Character N-Gram-Image as a Primitive

The Character N-Gram Spotting framework begins with segmenting word-images to candidate character segments. All possible contiguous sequences of segments are considered as the CNG-Imges. The word-image is in-turn treated as a bag of its constituent

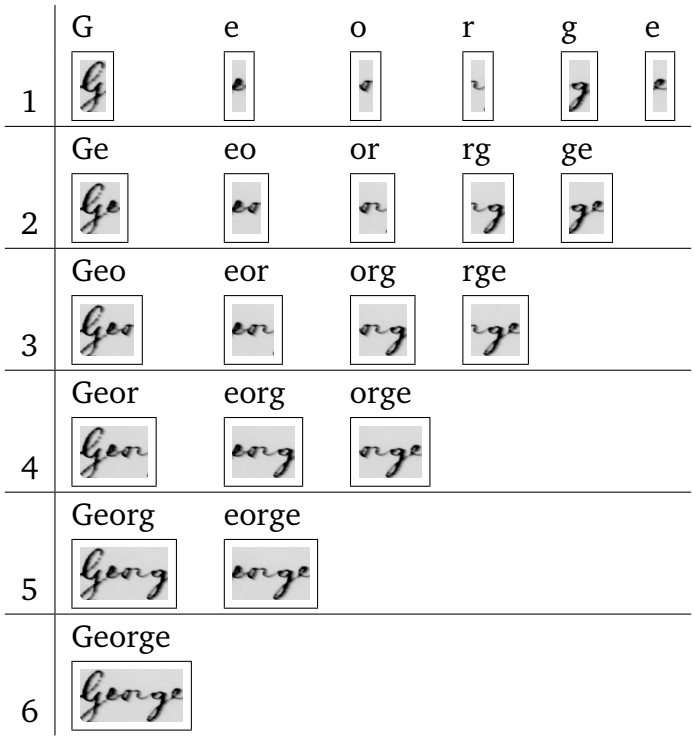


Figure 6.2 An example word image and its corresponding Character N-Gram Image set. It is important to note that we process character n-grams in the *image space only*, avoiding the need for explicit recognition.

CNG-Imges. Example CNG-Img set for a given word image is shown in Figure 6.2. The Character N-Grams representation combines the advantages of both characters and words.

- **Better disambiguation:** Through the presence of multiple characters, n-grams have more context than characters. The joint appearance of multiple characters in an n-gram is more distinctive than each character in isolation, which allows for better character disambiguation.

- **Robustness to Segmentation:** By virtue of considering all possible CNGs, the mechanism is robust to segmentation and degradation errors. All the n-grams are used in a unified recognition framework, which does not bias an n-gram based on its size. For example, if a character is over-segmented into two components, the entire

character will be represented as a bi-gram of these two components. Similarly, a merge between two characters will be treated as a unigram, but match a corresponding bi-gram from a similar word.

- **Large Training Data:** Each word-image consisting of L characters emits $L \cdot (L+1)/2$ CNGs. In the training phase, this makes it easier to obtain considerable amount of labeled n-gram exemplars from a small collection of labeled word-images. In the classification phase, n-grams generated from test word-images are recognized using n-gram models learned during training. For reasonable size of n-grams, the number of unique CNG is limited, hence allowing for easy indexing of the CNG-Img associated with them.

- **Partial Language Model:** The scheme implicitly performs a validity check of an n-gram, by always finding the closest valid n-gram seen during training. For example, in the word "Illinois", the first quad gram is always recognized as "Illi" instead of a visually similar "liil", which is most likely unseen in training data.

- **Unifying Character & Word based Approaches:** By treating the characters as uni-grams and words as L-grams, and by augmenting both with the intermediary n-grams, the CNG-Img approach both unifies and subsumes the character and word-level approaches known in literature.

In order to bootstrap the CNG-Img based recognition and retrieval, we need to extract labeled exemplars for the CNGs. The following section will describe the method we propose to achieve this.

6.3 Weakly-Supervised CNG-Img Segmentation

Given labels for word-images, the goal is to obtain character/CNG segmentation and annotation. In case of clean printed documents, the propagation of word to character labels can be performed if the number of characters and connected-components is exactly the same. This is not true for documents that contain degradations, documents that contain complex scripts such as for Indian languages and for handwritten documents due to cursive writing.

Though much of previous transcript-alignment work operated at the word-level [112, 115], there are a few recent works that explore automatic character level annotation given the word labels [189, 242]. For example in [189], character labeling is speed-ed up using character clustering/retrieval, while in [242], a conditional random field is used to align the labels to the word-image components over Chinese/Japanese documents. However, the underlying matching occurs at character or component level. We believe that better segmentation accuracy could be obtained by matching at CNG level.

We begin with three sets of data: i) weakly-annotated data that is labeled at word level, ii) strongly-annotated data of 300 words that is labeled at character level and iii) un-annotated data. Over the weakly-annotated data, the character segmentation is initialized using a weak feature, in our case we use character position in the word and the estimated character width. Let us denote the segment of character $c(i,j)$ of word W_i, by $\{L(i,j), R(i,j)\}$. The goal is to optimize the position of each $c(i,j)$ within the word, as well ensure that the segment appears similar to other instances of the same character. This can be represented by the following objective function:

$$E = \sum_{i,j} \sum_{Ex_{c(i,j)}} dist(Ex_{c(i,j)}, \{L(i,j), R(i,j)\}) \tag{6.1}$$

where $dist$ is a function that computes dissimilarity between two image segments (or features); and $Ex_{c(i,j)}$ are exemplars for the character $c(i,j)$. In the CNG-Img-Spotting setting,

$c(i,j)$ can represent the CNG-Img in place of isolated characters. The exemplars $Ex_{c(i,j)}$ could be generated either from strongly annotated data (a case of weakly-supervised learning), or by the putative character segments from weakly annotated data (semi-supervised setting). In this work, we restrict ourselves to the weakly supervised setting. The segmentation of the word-image into characters (and CNG-Img) is the unknown parameter in this function. Optimizing the above objective function is typically performed using a two-step optimization algorithm.

In the first step, we assume that the segmentation of the CNG-Imges is provided and optimize the objective function on the appearance of the CNG-Img segment against its expected appearance. The features from the segments are matched against those from exemplars in the strongly-annotated dataset. In cases where a CNG does not have an exemplar, it is generated by concatenating its corresponding character exemplars. The $dist$ function is the distance between the feature vectors of the character segment and the exemplar. We use profile features [185] which describe the upper/lower, projection and transition characteristics for each column of the image. The length of these features varies with the width of the image. These variable-length features are compared by finding the cost of Dynamic Time Warping (DTW) alignment.

In the second step, the appearance of the CNG-Img segment is assumed fixed and the segmentation is optimized. This is performed using the alignment information provided by the DTW. Since we begin with a segment bigger than the actual n-gram, the backtrack of the DTW path will align the character sequences, while the beginning and the end of the segmented image would have a high "insertion" cost. The steps followed are: 1) compute the average cost of points in the backtrack path, 2) traverse from the beginning on the backtrack path and stop when the cost is below average; call this point estimated left-boundary, 3) perform similar traversal from the end of the backtrack path for the estimated right-boundary.

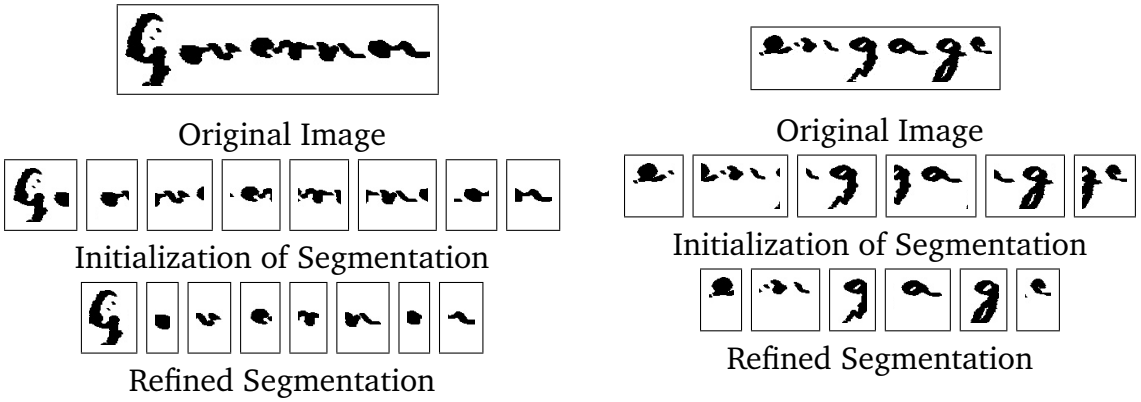


Figure 6.3 Example of successful annotation refinement. The words "Governor" and "engage" are first segmented into characters using a very weak feature (character width). The segmentation is then refined using strong features (Profiles) and matching technique (DTW).

We propose two ways for refining the character segments given the estimates from DTW. The first method, called *nGramAvg.*, finds the character boundaries as an aggregate of the boundaries defined by all CNG that constitute the given character. For example, given the word "George", the left boundary of "o" is obtained as the aggregate of the left boundaries of "or", "org" and "orge"; similar procedure is followed for the right boundary. In the second method, called *nGramSub.*, the new estimate of character "o" is found by subtracting the surrounding n-gram boundaries from the word, i.e. $\text{“}o'' = \text{“}George'' - \text{“}Ge'' - \text{“}rge''$. The same procedures are extended to refine CNG segments as well.

The new estimates for the segmentation is used in the next iteration of the algorithm, which is said to have converged when the segmentation estimates do not change beyond a certain empirical threshold. Example results from the segmentation/annotation procedure are shown in Figure 6.3.

Algorithm	Annotation Error
Initialization	71.2%
Characters Only	33.0%
nGramAvg.	31.8%
nGramSub.	**26.1%**

Table 6.1 Performance of segmentation refinement algorithms. The proposed segmentation procedure outperforms the other methods by a large margin.

6.3.1 Segmentation Evaluation

Experimental Setup: We evaluate our approach over the popular George Washington (GW) handwritten dataset [185]. The GW dataset consists of 20 pages containing more than 4700 words, written by a single author. We divide the dataset into two sets for training and testing, each contain 2300 words each. The training dataset is used to create labeled exemplars for the search system, using the segmentation approach presented in Section 6.3. The testing dataset is used to evaluate the retrieval performance. The word-images are pre-processed to remove the slant from handwriting using a shear transform. We use the profile features [185] to represent the character n-gram images, which are known to be better suited for handwritten documents [185] and were shown to be robust to degradations [177]. Since profile features are dependent on word width, the images are scaled to a canonical size before feature extraction, to ensure uniform feature length while indexing.

The segmentation error is defined as

$$\frac{SegmentWidth - Overlap}{GroundtruthWidth} \tag{6.2}$$

where $Overlap$ is the intersection between the segment and the groundtruth.

The annotation refinement results are shown in Table 6.1. The baseline method, given as **Characters Only** which uses isolated character segments refined by matching against

character exemplars. We evaluate the error over character segments alone, in order to compare fairly with the baseline method. The two nGram based methods out-perform character based methods by a large margin. Among the two nGram based re-estimation methods, *nGramSub*. performs slightly better than *nGramAvg*. The best performing setting has an error of 25%, which amounts to about 2 pixel error on either side of a typical 10 pixels width character. Much of the error is owing to the tight groundtruth segments, while the obtained segments contain some amount of cursive-connector pixels.

6.4 Retrieval with CNG-Img-Spotting

The process of CNG-Img Spotting can be summarized in three steps. Firstly, CNG-Imges are obtained from the document collection and represented in a suitable feature space. The features are indexed for quick retrieval. In the second step, given a QBE, the query is expanded into its constituent CNG-Imges. The features from the expanded query are looked up in the index of features. The final step consists of obtaining individual retrieval lists for each of the query-CNG-Img, and merging them appropriately to present the user with one ranked list of word-images.

By using a single index for all the CNG-Img, the approach robustly matches similar CNG-Imges inspite of degradations such as cuts and merges. Since the words are inherently represented at a sub-word level, CNG-Img-Spotting allows for easy matching of morphologically related words. Further, the spotting approach can be easily extended to QBK, by converting the query-keyword to a query-image by identifying an exemplar.

Indexing Phase: The matching of CNG-Imges in the feature space is a computationally expensive task, since each word in the collection emits a large number of CNG-Imges. In order to speed-up the matching, we build an index using a combination of Hierarchical K-Means (HKM) and a random forest of KD-Trees [157]. To enable building an index, the features extracted from the CNG-Img are ensured to be of the same dimensions.

Retrieval Phase: Using the built index, we obtain a list of approximate nearest neighbors for each Query-CNG-Img. Thus, a candidate retrieval list is obtained for each unigram, bigram, etc. The task now, is to fuse the individual retrieval lists to obtain the final relevant image set for the given query. This is achieved by providing a ranking of the individual retrieved lists using a ranking function as described below. If Q is the query word with length L, then let us denote as $Q_1, Q_2, \ldots, Q_L$, the sets of CNG-Img for the query. For each Q_i^j, the approximate NNs list is given as R_i^j. Each point P_k in the retrieved list R_i^j is weighted by its distance from the query as

$$S_i^j(P_k) = (2^{L-i} \cdot (L - i + 1))^{-1} \cdot (1 - dist(P_k, Q_i^j)) \tag{6.3}$$

The first term of the ranking function ensures that longer n-grams are given more weight than shorter n-grams. We choose to reduce the cumulative weight of each n-gram by half for each step of the n-gram. Thus, a K-gram will be given twice the weight of K-1-gram and so on. The second term is the distance of the retrieved CNG-Img to the query CNG-Img. The unique words from the retrieved lists of all query-CNG-Img are scored by aggregating their corresponding $S_i^j(P_k)$ measure, across the respective retrieved lists that they occur in. The unique words are then re-ranked and presented to the user as the retrieval list for the given query image.

In-Vocabulary Queries: In the case of a text-query being present in the training data, the query is said to be "in-vocabulary". In such cases, the exemplars for the query can be directly obtained from the training dataset. The index scheme can now be queried against the indexed CNG-Imgs from the dataset. If multiple exemplars are present for the given query, better results could be obtained by using each of them as a separate QBE and aggregating the retrieved list across all of them. Multiple exemplars are particularly useful while retrieving documents from different writers.

Out-of-Vocabulary Queries: Given a text query that was not seen previously in the training dataset, it is called an OOV query. The OOV query is first expanded to its nGrams

Retrieval Scheme	Prec @ 10	mAP	Time/Query(sec.)
QBE Word Spotting (DTW)	0.47	0.44	15
QBE Word Spotting (L2)	0.23	0.14	0.24
QBE CNG-Img-Spotting	0.36	0.32	0.27
QBK In-Vocabulary	**0.54**	**0.50**	0.59
QBK Out-of-Vocabulary	0.24	0.18	0.59

Table 6.2 Retrieval performance of the proposed system, across various query and algorithm settings.

in the text space (CNG-Text). For each CNG-Text, the training dataset is searched for the presence of an exemplar. If such an exemplar is present, it is used to query the index over the document collection to retrieve the approx-NN list. In cases where an exemplar is not present in the labeled dataset, such an exemplar is synthetically created by concatenating exemplars of its constituent characters/nGrams. The synthetic exemplar is now used to query the index. Due to the ability to construct any given query from its constituent CNG, the OOV querying mechanism can answer queries from an *unlimited vocabulary* set.

6.4.1 Retrieval Evaluation

The dataset used for evaluating retrieval is the same George Washington handwritten documents used in the previous section. A few example retrieval results are presented in Figure 6.4. Given the query "Companies", our system was able to retrieve similar words such as "Company" in the Top-10 results. In case of the the query "receive", the erroneous result "inconceivable" is found due to the matching of the quad-gram "ceiv".

The retrieval performance of the CNG-Img spotting framework is evaluated using two metrics: i) Precision among top-10 results, and ii) mean average precision (mAP). The precision is computed such that morphologically similar words are labeled as correct matches. This is obtained by finding the longest common sub-sequence (LCS) between the query and

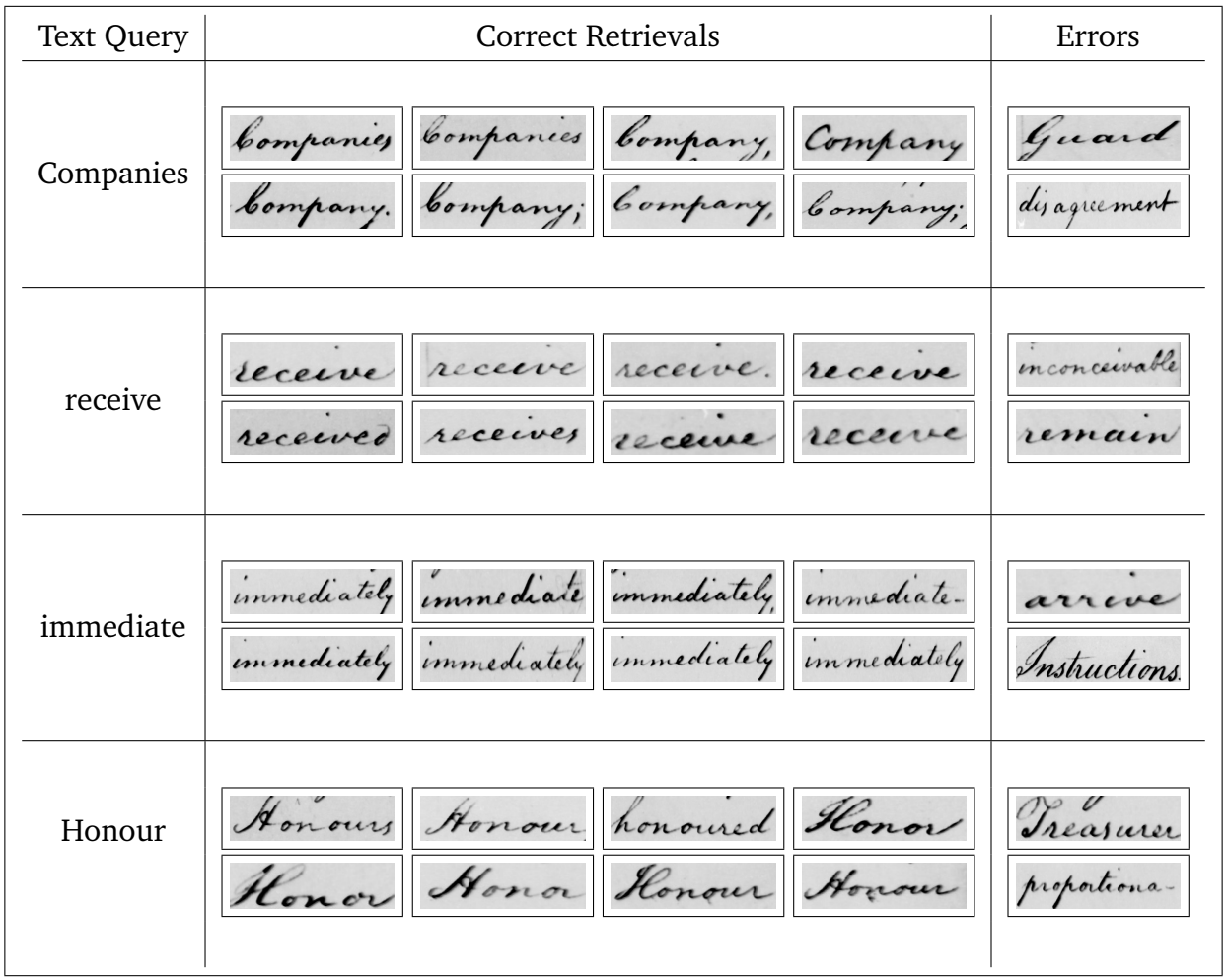


Figure 6.4 Example retrieval results from our QBK retrieval system on the George Washington dataset. The results are obtained without explicit recognition or morphological analysis. As we can see the results are quite accurate, with similar words being retrieved automatically. A few errors in the retrieval are also presented.

the retrieved word, normalized by query length. If the LCS is greater than a threshold of 0.5, they are said to match. While the precision is a representative of accuracy among the retrieved results, the mAP is a combination of both precision and recall. The Average Precision (AP) is computed as the average of precision at each relevant retrieval for the given query. The Mean Average Precision (mAP) is the mean of the AP for multiple queries. It is essentially the area under the PR curve obtained from the retrieval evaluation.

The results are presented in Table 6.2. The performance of our approach is compared against two word-spotting baselines, one that uses DTW to match the query with the collection and another that uses Euclidean distance. The DTW based word spotting performs well in the QBE setting, but takes close to 15 seconds per query. The L2-based word spotting allows the use of a feature index that speeds up retrieval time to about 0.24 seconds, but performs poorly compared to the proposed CNG-Img spotting approach. In the QBK approach, we obtain a significant performance, given by a mean average precision of 0.5, for in-vocabulary queries. The performance drops for OOV queries, which is mostly due to similar CNG across words that are not morphologically related.

Time & Memory The proposed approach is also computationally efficient. The retrieval system uses 500 MB of RAM to index the feature set over the GW collection, which takes less than 440 seconds to build. Example query-time using different approaches is provided in Table 6.2. All methods, with the obvious exception of DTW, have a sub-second retrieval time. During exemplar-building, the step of segmentation refinement takes about 1.1 seconds per word. The index size and indexing time scales linearly with the dataset, which means our framework is applicable to much larger collections. Further improvements could always be obtained by using more compact features or better indexing schemes.

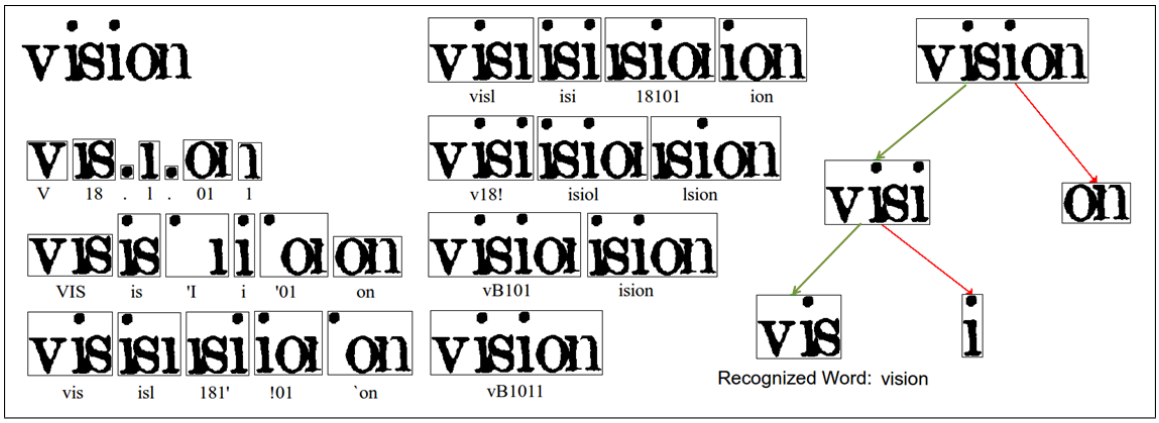


Figure 6.5 A depiction of word recognition using n-grams. The given word is segmented to its constituent n-grams, each of which is recognized independently. The labels obtained for each n-gram are then fused using dynamic programming to obtain the optimal sequence of n-gram labels. The word in this example is recognized as a bigram (of 3 characters), unigram (1 char.) and a unigram (2 char.).

6.5 Recognition with CNG-Img

In this section, we shall apply the CNG-Img primitive towards robust recognition in the presence of heavy degradations in document images. As a consequence of the CNG-Img properties, the method can potentially recognize an unlimited vocabulary. The goal is to infer the label of a given word by recognizing its constituent n-grams.

We have shown in Section 6.3, how to obtain considerable amount of labeled n-gram exemplars from a small collections of labeled word-images. In the training phase, Towards building a recognition system, the CNG-Img set is generated from each test word-image and recognized using n-gram models learnt during training. The different n-grams extracted from an example word image, are shown in Figure 6.5 (b). The individual n-gram recognitions are merged together to obtain the most suitable label for the word.

CNG-Img recognition presents these challenges:

1. Building a recognition scheme for n-grams involves classifying against 100K n-gram classes. Classification at such large class sizes is a non-trivial problem.

2. Recognizing n-grams in the test-phase is expensive, as the number of features to classify is multiplied by a factor of $(k+1)/2$ (for a k length word).

3. The label of the word-image needs to be inferred by aggregating the recognition of individual n-grams.

6.5.1 Recognizing Individual CNG-Imgs

In the training phase, the design of the n-gram recognizer consists of identifying the features and the classifier for the task. In the presence of degradations and multiple fonts, as we saw in Section 5.7, it was observed that profile-features [185] are quite robust in word-level matching. The profile-features proposed in [185] are of variable length, depending on the width of the word. In order to ensure that the features are of a constant size, all n-grams are scaled to a canonical size before profile-features are extracted.

In the presence of thousands of classes to recognize, the classifier of choice needs to be very robust. This means that classifiers should be able to learn from a small set of exemplars per class and also be easy to train. A Nearest Neighbor (NN) classifier would require no training and is highly scalable with the class sizes. An NN classifier was shown to work better than SVMs [177] for a task of classifying 33K words of 1000 classes. Further, the context present in the n-gram is sufficiently distinguishing for many pairs of characters, thereby obviating the need for strong classifiers (and in some cases strong features). We use scaled profile-features with a NN classifier for the n-gram recognition.

The challenge during the test phase is that the test dataset size is increased by an order of magnitude, since each word-image generates a large collection of n-grams. The computational cost of NN classifiers can be significantly reduced by using an Approximate Nearest Neighbor (ANN) search, similar to the ones used in Section 5.5. In ANN, the labeled exemplars are indexed using Hierarchical K-Means and KD-Trees [157]. Due to the indexing, the test point need not be compared against all the exemplars in the labeled

data. By looking up a given test n-gram in the built index, one could identify the ANN in about 10 milli-seconds, while a regular NN would take about 5 seconds (about 500$\times$ speedup). The obtained ANN is used to identify the label of the given n-gram.

6.5.2 Fusing N-Gram Recognition for Word Recognition

Each n-gram provides certain evidence for what the word's label should be. In the situation where every n-gram is correctly recognized, the n-gram labels reinforce the evidence from one another. For example, given the word "most", if the first trigram is correctly recognized as "mos" and the second trigram as "ost", the overlapping "os" implies that the word is very likely to be "most". However, if one of the trigrams is erroneously recognized, the inference is not immediate, thereby requiring to include evidence from the bigrams and unigrams, etc. In this paper, we use an OR scheme where it suffices to correctly recognize only a small subset of all the n-grams.

Given the recognition of each i^{th} n-gram $w_{n,i}$, with the corresponding confidence of $c_{n,i}$. The objective is to identify the sequence of n-grams that would result in the most confident prediction for the entire word. The label for the word can be defined recursively as optimizing this objective function:

$$\begin{aligned} W_{n,i} &= w_{n,i} && \text{, if } n = 1 \\ &= min_{m \in [1,n-1]} \{ W_{n,i}, (n-m) \cdot \tfrac{(W_{n-m,i} + m \cdot W_{m,n+i-m})}{n} \} && \text{, otherwise} \end{aligned}$$

The first condition returns the unigram label, while the second condition finds the minimum cost between the label for the n^{th} gram and its constituent $(n-1)^{th}$ grams. This objective function lends itself to be optimally solved using dynamic programming (DP). Each entry in the DP array stores the cumulative confidence of the n-grams that contribute to the word label. The backtracked path of the DP array is the most confident sequence of n-grams. The word label is obtained by simply concatenating these n-grams. This process is shown in Figure 6.5. The n-grams formed by the CCs are individually recognized and a label is obtained from the closest match, this label is shown below each n-gram image.

Aspect	HW Recognition	Word Spotting (DTW based)	CNG-Img Spotting
Text querying	Yes	No	Yes
Retrieval time	Instantaneous	Time-consuming	Interactive
Degradations	Serious Effects in Segmentation	Lesser Effects in Segmentation	Segmentation errors do not matter
Data Scalability	Scalable	Not Scalable	Scalable
Vocabulary Coverage	Limited by Post-processing	Limited by Manual Annotation	Almost Unlimited
Morphologically-related Word Retrieval	Requires Language Model	Requires Partial Matching	Inherently Addressed

Table 6.3 A comparison of our approach with other popular approaches for handwriting retrieval. Our approach has multiple advantages over both approaches.

The word label is obtained as a concatenation of the most confident n-gram recognition sequence. For the given example, the last unigram is confidently recognized as “on”. The trigram corresponding to “visi” is recognized as a trigram corresponding to “vis” and a unigram for “i”, resulting in the word label “vision”.

6.5.3 Analysis of the Algorithm

A summary of the n-gram based recognition process is provided in Algorithm 6.1. A comparison of our approach to the popular existing approaches such as word recognition and word-spotting is presented in Table 6.3. It is clear that our approach overcomes many of their limitations.

Recognizing an Unseen Word: Consider the case where the word “modulation” is OOV. A holistic word-recognition would fail to obtain a label for such a word-image. In our approach, suppose that exemplars are present for the words “module” and “integration”, the constituent n-grams from these exemplars would be indexed. During the test phase,

Algorithm 6.1 N-Gram based Word Recognition Framework

Training Phase
for all Word-images in *Training* data **do**
  Segment words to connected-components.
  Obtain character n-gram images and extract features.
end for
Build index over all features extracted over *Training* data.
Test Phase
for all Word-images in *Test* data **do**
  Segment words to connected-components, extract features for character n-grams.
  Recognize n-grams against index built on *Training* data. Obtain confidence of each recognition.
  Apply Dynamic Programming on the confidence scores and the n-gram size to fuse n-gram recognitions. Obtain the most likely word label.
end for

the n-grams "modul" and "ation" in the test word are correctly recognized against the corresponding n-gram exemplars. When the recognition result is fused, our algorithm would generate the correct label for the *unseen* word.

Effect of Cuts & Merges: In the case of cuts, a character would be split to two components. A standard OCR would treat them as separate characters and classify them separately, deferring error-correction to the post-processing step. In our algorithm, the split components form a valid bigram and is thus correctly classified inspite of a cut. Similarly, in the case of merges multiple characters in the test word would be combined to appear like one CC. In such cases, a unigram from test data would match an n-gram from training data, resulting in a correct recognition. We have effectively negated the effect of cuts and merges by using a single indexing scheme over all n-grams, such that it ignores the number of components and only focuses on how they appear together. Hence, the approach is quite robust to degradations such as cuts and merges.

Limitations of Approach: One of the limitations of our approach is that it assumes word segmentation to be provided as input. This assumption might be tough to satisfy for Arabic documents, in which cases HMM solutions can perform well. We could address this limitation easily by recognizing at the line-level instead of word-level. Another limitation

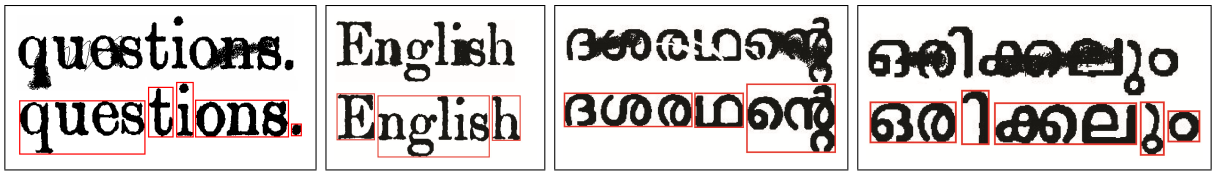


Figure 6.6 Examples of words where our algorithm correctly recognizes despite degradations. Popular OCRs have failed to recognize these images. We propose a novel n-gram based recognition scheme that addresses challenges in character recognition. In each example, the top word is the test image and the bottom word is the concatenation of the matched n-grams, outlined in red.

	# Words	# N-Grams	# Cuts	# Merges
Good	81K	3M	5388 (6%)	783 (0.01%)
Bad	141K	6.7M	55,149 (39%)	16781 (12%)
Ugly	23K	0.5M	1575 (6%)	16,923 (74%)

Table 6.4 Details of the test datasets for the printed Malayalam collection.

is that our approach cannot disambiguate between similar looking n-grams, such as “lil” and “ill” from *lily* and *pill*.

6.5.4 Recognition Results

The data for our experiments is obtained from multiple sources such as scanned books and newspapers for the Indian language of Malayalam. Groundtruth was obtained by manual typesetting. The *Training* dataset consists of 100K non-degraded words. The *Test* dataset is divided into three groups based on their degradations: Good, Bad and Ugly. The details of the number and type of degradations in the test datasets are given in Table 6.4. The Bad dataset consists of more cuts while the Ugly dataset has lot more merges. We also build an English dataset from 140K words.

There are two baselines that we compare our approach against: i) a character classification scheme and ii) a word recognition scheme. To ensure that there is no bias in terms of features and classifiers, we use the same features and classifier as used for n-gram recog-

Character Error Rate(CER)				
Malayalam	**MOCR**	**Char-Rec (Uni).**	**Word-rec(k)**	**N-Gram Rec.**
Good	4.43	4.98	50.07	4.27
Bad	19.62	12.87	59.02	7.07
Ugly	37.64	36.03	67.5	18.12
English	**Tesseract**	**Char-Rec (Uni).**	**Word-rec(k)**	**N-Gram Rec.**
Good	0.64	2.36	17.48	2.0
Bad	3.7	25.27	20.72	7.9
Word Error Rate (WER)				
Malayalam	**MOCR**	**Char-Rec(Uni).**	**Word-Rec(k)**	**N-Gram Rec.**
Good	20.86	24.35	64.48	20.26
Bad	55.23	41.8	73.21	29.27
Ugly	62.94	64.76	84.65	47.73
English	**Tesseract**	**Char-Rec(Uni).**	**Word-rec(k)**	**N-Gram Rec.**
Good	3.24	8.15	19.98	7.08
Bad	10.66	49.47	24.74	21.21

Table 6.5 Character and word error rates for Malayalam and English datasets. N-Gram based recognition consistently outperforms the character and word recognition baselines. By using n-grams, we are able to improve character recognition by 19%.

nition (namely scaled profile-features and NN classification). The recognition accuracy is measured at both character and word levels respectively by Character Error Rate (CER) and Word Error Rate (WER). We also compare our approach with a state-of-the-art OCR for both languages.

The quantitative results are provided in Table 6.5. Consider the character recognition results on the Malayalam datasets. It is clear that the performance of all the recognition systems degraded in performance with poor quality data. However, the loss of accuracy is much more pronounced in the case of the standard OCR, as opposed to the proposed N-Gram based recognition (column 5). The difference in accuracy of the Good and Ugly datasets for the OCR was more than 33% while that of our method was less than 14%. In

column 3 we present the results from classification using unigrams and column 4 contains results from recognizing the word as a whole. We can see that the word-recognition performs poorly over all datasets, since words that are unseen in training do not receive valid labels in the test-phase. The improvement from unigram to n-gram is more pronounced in the Bad dataset, which contains more cuts than merges. On the other hand, in Ugly dataset which has more merges, the unigram performance is comparable to n-grams, since the so-called unigrams are correctly matched to the corresponding n-gram in training data. The trends are similar when considering the WER, and the improvements are more substantial, about 24% over Ugly Malayalam dataset.

The English "Bad" dataset also contains a number of cuts and almost no merges, hence the trends are similar to Malayalam Bad. However, we observe that the k-gram recognition is better for English than Malayalam. This is a result of the smaller vocabulary size that reduces the number of OOV words in the test set. Comparing with an OCR, we do not observe any improvement in performance using n-gram recognition. This could be due to better features/classifiers and a strong post-processor module used in Tesseract. Also, Tesseract has the advantage of better design and engineering due to contributions from the open-source community. Instead of competing with OCR, it might be a good strategy to use n-gram recognition in those cases where the OCR is known to fail, hence complementing existing systems with robustness for degraded documents.

Error Analysis:

We observed that many of the errors in our recognition scheme are due to erroneous, yet confident recognition of n-grams. For example, the letter "c" is sometimes confused with "e" with very similar confidence values. In the absence of more confident n-grams, this error is retained in the final word. Similarly, some errors are found in similar looking n-grams that cannot be disambiguated, such as "lil" (in *lily*) and "ill" (in *pill*). This could possibly be addressed in the future by using stronger features for matching, or by using

multiple labels for each n-grams from which the most appropriate is picked in the fusing scheme.

6.6 Summary

In this Chapter, we presented an alignment of visual and textual data for document images at a sub-word level. Consequently, we were able to build a retrieval and recognition system over printed and handwritten documents that allows: i) text-queries without explicit recognition, ii) sub-word retrieval without morphological analyzer and iii) search over unconstrained vocabularies without partial matching. The segmentation of the visual data is inferred from the weakly-supervised text labels provided at the word-level. The temporal ordering, as well as the hierarchy of the CNGs in the word image is used to build a reliable annotation as well as recognition mechanism.

Chapter 7

Conclusions and Future Directions

We now present a summary of the achievements, outcomes and learnings from this thesis, and discuss possible future directions of applying and extending these ideas.

7.1 Summary and Conclusions

In this thesis, we address the problem of building text-based retrieval over multimedia collections, using additional information obtained from parallel text sources. We proposed novel approaches that perform alignment across information domains, across various levels of spatial and temporal binding, and across the scale of the data. Our work is important in light of the acknowledged understanding that vision-alone approaches are still far from realizing the dream of semantic multimedia understanding.

We demonstrate our approach over four multimedia-text pairs: i) Movies with transcripts, ii) Cricket videos with commentary, iii) Document images with lexicon and iv) word images with labels. The temporal binding between the parallel annotation and the multimedia, is relaxed as we move from one setup to the next, calling for newer approaches to tackle the problem. In Chapter 3, we present the possibility of matching multimedia with descriptive text, resulting in a semantic annotation of multimedia. We extend the usability of parallel text to not only annotate, but also to *segment* the multimedia into meaningful

entities, as described in Chapter 4. With the work presented in Chapter 5, we demonstrate the feasibility of building large-scale retrieval systems using such annotation schemes. Further, in Chapter 6 we showcase the utility of annotation in building a robust recognition system over a challenging dataset. We show that effectively leveraging parallel text is a promising direction toward multimedia retrieval, recognition and understanding, for the near future.

The overall contributions of our work can be summarized as follows.

- A direction towards explicitly mapping semantic textual information to weakly-aligned multimedia, across multiple semantic granularities and temporal bindings.

- A multitude of "model-and-match" approaches, that have shown to work better than the classical "recognize and annotate" approach, especially with higher level semantics.

- An indexing based approach to annotation, that shows tremendous speed up such that large-scale labeling is now made feasible.

- A demonstration that practical retrieval systems could be built over large collections of multimedia by leveraging additional information available freely.

- Retrieval systems that can answer text-queries in less than a second, searching over large quantities of multimedia data.

- A significant amount of labeled multimedia data, that could be used to learn a host of new classifiers such as for recognizing actions, activities, events, gestures, excitement, etc.

7.2 Future Directions

One direction of future research would be application of the techniques presented in this work to other multimedia-text pairs. We have picked those cases, which were challenging to begin with, yet reasonably solvable. There are several other image/video collections that have parallel information that have yet to be exploited for annotation and retrieval, for example i) news videos with news articles, ii) lecture videos with lecture notes/slides, etc. Similarly, there is large scope in multimedia processing with non-text information, for example, i) the on-board camera of a vehicle could be aligned with local information extracted from geographical maps, ii) personal multimedia collections could be automatically aligned with the appropriate "circles" in the social network, iii) cameras monitoring patients in a hospital/ambulance could be augmented with additional sensors, etc.

Secondly, it is important to build models and solutions for multimedia data the does not contain any parallel information. This is particularly useful in novel domains of multimedia where generating parallel information is either expensive or unreliable due to subjective bias. One could look at using models learned over weakly-aligned data (such as those presented in this thesis) to those data that has no such parallel information. This perspective could be considered an extension of semi-supervised learning, where both the labeled and unlabeled data belong to the same domain. For example, one could learn to predict user response to an advertising campaign, by mining the reactions of previous campaigns from social media. Similarly, by learning the relationships between a student's behavior in the classroom and their academic performance, one could provide focused guidance to the next generation of students.

Finally, we would like to believe that annotation and retrieval is a stepping stone for large scale multimedia understanding. This is the flip side of treating the understanding problem as a means to the retrieval goal. The annotation/retrieval modules could be a part of the understanding pipeline. Naturally, one can think of a feedback loop between

annotation modules and understanding approaches, so that both could be improved simultaneously.

With the success of the Internet, through the sharing of common knowledge, we have seen a significant boost in human learning, understanding and achievements. A similar gain could be possible, especially in developing countries such as India, China, and those in Africa, could benefit significantly, through the effective use of multimedia as a tool for dissemination of information and assimilation of knowledge. We believe that our work could take us a step forward in this direction.

Bibliography

[1] Google Images search engine at: http://www.images.google.com. xvii, 15

[2] Digital Library of India at: http://dli.iiit.ac.in. xxiii, 91, 93

[3] Google search engine at: http://www.google.com. 1, 3, 18, 23

[4] Bing search engine at: http://www.bing.com. 18

[5] Cricinfo at: http://www.cricinfo.com. 64, 67

[6] Google Books at: http://books.google.com. 91

[7] Internet Archive at: http://www.archive.org. 91

[8] Universal Library at: http://www.ulib.org. 91

[9] Tesseract OCR Engine at: http://code.google.com/p/tesseract-ocr. 92

[10] C.L.A. Clarke, N. Craswell, I. Soboroff, Overview of the TREC 2009 Web Track (online resource) at: http://trec.nist.gov/pubs/trec18/papers/WEB09.OVERVIEW.pdf. 120

[11] 500px photo sharing service at: http://500px.com/. 22

[12] Apache lucene - opensource search engine tool at: http://lucene.apache.org/. 5, 17, 31

[13] Charlie Chaplin scripts from: http://www.charliechaplinarchive.org/. 37, 56

[14] CMU Sphinx from: http://cmusphinx.sourceforge.net/html/cmusphinx.php. xx, 48, 49

[15] Demo of the Video Google system at: http://www.robots.ox.ac.uk/˜vgg/research/vgoogle/. xvii, 17

[16] Dragon Naturally Speaking software from: http://www.nuance.com/naturallyspeaking/. xx, 48, 49

[17] Facebook social network at: http://www.facebook.com/. 1

[18] Flickr photo sharing service at: http://www.flickr.com/. 22

[19] Flow smartphone app by amazon at: http://a9.com/whatwedo/mobile-technology/flow-powered-by-amazon/. xvii, 15, 29

[20] Getty Images from: http://www.gettyimages.com. 22

[21] Google goggles smartphone app at: http://www.google.com/mobile/goggles. xvii, 15, 29

[22] Greenstone digital library software at: http://www.greenstone.org/. 5, 17

[23] Hulu video broadcast service at: http://www.hulu.com/. 22

[24] Internet Movie Database: http://www.imdb.com/. 33

[25] Khan academy open courseware at: https://www.khanacademy.org/. 2

[26] Movie scripts from IMSDb: http://www.imsdb.com/. 34

[27] Movie scripts from Script-o-Rama: http://www.script-o-rama.com/. 34

[28] Netflix video broadcast service at: http://www.netflix.com/. 22

[29] Scripts for TV shows from Twiz TV: http://www.twiztv.com/. 34

[30] Seinfeld scripts from: http://www.seinfeldscripts.com. xix, 35, 37

[31] Siri personal assitant for the iPhone, cursory details at: http://www.apple.com/ios/siri/. 3

[32] YouTube video sharing service at: http://www.youtube.com/. 1, 20, 22

[33] ABBYY FineReader software from. http://finereader.abbyy.com/. 58, 92

[34] J. K. Aggarwal and M. S. Ryoo. Human activity analysis: A review. *ACM Computing Surveys*, 43(3), 2011. 20

[35] A. Andoni, M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. *Nearest Neighbor Methods in Learning and Vision: Theory and Practice*, 2006. 15, 107

[36] S. Antani, R. Kasturi, and R. Jain. A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video. *Pattern Recognition*, 35(4), 2002. 13

[37] J. Ashley, M. Flickner, J. Hafner, D. Lee, W. Niblack, and D. Petkovic. The query by image content (QBIC) system. In *Proc. SIGMOD*, 1995. 13

[38] E. Ataer and P. Duygulu. Retrieval of Ottoman documents. In *Proc. MIR*, 2006. 99

[39] E. Ataer and P. Duygulu. Matching Ottoman words: An image retrieval approach to historical document indexing. In *Proc. CIVR*, 2007. 100, 112

[40] N. Babaguchi, Y. Kawai, and T. Kitahashi. Event based indexing of broadcasted sports video by intermodal collaboration. *IEEE Transactions on Multimedia*, 4(1), 2002. 23

[41] J. R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. C. Jain, and C.-F. Shu. Virage image search engine: an open framework for image management. In *Proc. SPIE*, volume 2670, 1996. 13

[42] H. S. Baird. Anatomy of a versatile page reader. *IEEE Special Issue on OCR*, 80(7), 1992. 92, 97

[43] H. S. Baird, V. Govindaraju, and D. P. Lopresti. Document analysis systems for digital libraries: Challenges and opportunities. In *Proc. DAS*, 2004. 92

[44] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 92(1), Mar. 2011. 20

[45] A. Balasubramanian, M. Meshesha, and C. V. Jawahar. Retrieval from document image collections. In *Proc. DAS*, 2006. 99, 111, 127

[46] D. Barnard, K.; Forsyth. Learning the semantics of words and pictures. In *Proc. ICCV*, volume 2, 2001. 24

[47] H. Bay, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. In *Proc. ECCV*, 2006. 16

[48] R. Bayer. Binary B-Trees for virtual memory. In *Proc. ACM-SIGFIDET Workshop on Data Description, Access and Control*, 1971. 15

[49] T. Berg and D. Forsyth. Animals on the web. In *Proc. CVPR*, volume 2, 2006. 23

[50] T. L. Berg, A. C. Berg, J. Edwards, and D. A. Forsyth. Who is in the picture. In *Proc. NIPS*, 2006. 24, 26

[51] A. Bharati, P. Rao, R. Sangal, and S. M. Bendre. Basic statistical analaysis of corpus and cross comparision. In *Proc. ICON*, 2002. 98, 127

[52] M. Black, Y. Yacoob, A. Jepson, and D. Fleet. Learning parameterized models of image motion. In *Proc. CVPR*, 1997. 20

[53] D. Blei and M. I. Jordan. Modeling annotated data. In *Proc. SIGIR*, 2003. 24

[54] A. Bosch, A. Zisserman, and X. Munoz. Scene classification using a hybrid generative/discriminative approach. *TPAMI*, 30(4), 2008. 113

[55] R. Bowden, D. Windridge, T. Kadir, A. Zisserman, and J. M. Brady. A linguistic feature vector for the visual interpretation of sign language. In *Proc. ECCV*. Springer-Verlag, 2004. 20

[56] M. Bray, P. Kohli, and P. H. S. Torr. Posecut: Simultaneous segmentation and 3d pose estimation of humans using dynamic graph-cuts. In *Proc. ECCV*, 2006. 29

[57] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proc. CVPR*, 1998. 20

[58] C. Bregler, J. Malik, and K. Pullen. Twist based acquisition and tracking of animal and human kinematics. *IJCV*, 56(3), Feb. 2004. 20

[59] R. Brunelli, O. Mich, and C. M. Modena. A survey on video indexing. *Journal of Visual Communications and Image Representation*, 10, 1996. 16

[60] P. Buehler. *Automatic Learning of British Sign Language from Signed TV Broadcasts*. PhD thesis, University of Oxford, 2010. xviii, 26, 27

[61] P. Buehler, M. Everingham, and A. Zisserman. Learning sign language by watching TV (using weakly aligned subtitles). In *Proc. CVPR*, 2009. xviii, 26, 27

[62] C. V. Jawahar, M. P. Kumar, and S. S. Ravikiran. A bilingual OCR system for Hindi-Telugu documents and its applications. In *Proc. ICDAR*, 2003. 97

[63] C. V. Jawahar, M. Meshesha, and A. Balasubramanian. Searching in document images. In *Proc. ICVGIP*, 2004. 92

[64] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *TPAMI*, 29(3), 2007. 26

[65] W. B. Cavnar and J. M. Trenkle. N-gram-based text categorization. In *Proc. DAIR*, 1994. 128

[66] G. S. Chambers, S. Venkatesh, G. A. W. West, and H. H. Bui. Segmentation of intentional human gestures for sports video annotation. In *Proc. MMM*, 2004. 65

[67] J. Chan, C. Ziftci, and D. A. Forsyth. Searching off-line Arabic documents. In *Proc. CVPR*, 2006. 98, 100

[68] S. Chaudhury, G. Sethi, A. Vyas, and G. Harit. Devising interactive access techniques for Indian language document images. In *Proc. ICDAR*, 2003. 99

[69] R. Chellappa, A. R. Chowdhury, and S. Zhou. *Recognition of Humans and Their Activities Using Video*. Morgan Claypool, 2005. 20

[70] H.-W. Chen, J.-H. Kuo, W.-T. Chu, and J.-L. Wu. Action movies segmentation and summarization based on tempo analysis. In *Proc. ACM Multimedia*, 2004. 16, 20

[71] M. Chen and A. G. Hauptmann. Multi-modal classification in digital news libraries. In *Proc. JCDL*, 2004. 48

[72] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Proc. ICCV*, 2007. 30

[73] T. Cour, C. Jordan, E. Miltsakaki, and B. Taskar. Movie/script: Alignment and parsing of video and text transcription. In *Proc. ECCV*, 2008. 34

[74] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs. A maximum likelihood stereo algorithm. *CVIU*, 63(3), 1996. 76

[75] W. B. Croft, S. M. Harding, K. Taghva, and J. Borsack. An evaluation of information retrieval accuracy with simulated OCR output. In *Proc. DAIR*, 1992. 92

[76] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, 2005. 16, 111, 113

[77] R. Datta, J. Li, and J. Z. Wang. Content-based image retrieval: Approaches and trends of the new age. In *Proc.MIR*, 2005. 13

[78] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and L. Fei-fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 23

[79] L.-Y. Duan, J. Wang, Y. Zheng, J. S. Jin, H. Lu, and C. Xu. Segmentation, categorization, and identification of commercial clips from tv streams using multimodal analysis. In *Proc. ACM Multimedia*, 2006. 20

[80] S. Dutta, N. Sankaran, Pramod Sankar K., and C. V. Jawahar. Robust recognition of degraded documents using character n-grams. In *Proc. DAS*, 2012. 11, 128

[81] P. Duygulu, K. Barnard, N. de Freitas, and D. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proc. ECCV*, 2002. 24

[82] J. Eakins, K. Riley, and J. Edwards. Shape feature matching for trademark image retrieval. In *Proc. CIVR*, 2003. 14

[83] M. Everingham, J. Sivic, and A. Zisserman. "Hello! My name is... Buffy" – automatic naming of characters in TV video. In *Proc. BMVC*, 2006. 23, 26, 34, 45, 46, 47

[84] M. Everingham, J. Sivic, and A. Zisserman. Taking the bite out of automatic naming of characters in TV video. *IVC*, 27(5), 2009. 47

[85] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 88(2), June 2010. xviii, 3, 19, 23

[86] A. Farhadi and D. A. Forsyth. Aligning ASL for statistical translation using a discriminative word model. In *Proc. CVPR*, 2006. 26

[87] A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth. Every picture tells a story: Generating sentences from images. In *Proc. ECCV*, 2010. xviii, 25, 26

[88] Y. Fataicha, M. Cheriet, Y. Nie, and Y. Suen. Retrieving poorly degraded OCR documents. *IJDAR*, 8, 2006. 128

[89] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 2010. 19

[90] S. Feng, R. Manmatha, and V. Lavrenko. Multiple bernoulli relevance models for image and video annotation. In *Proc. CVPR*, 2004. 24

[91] X. Feng and P. Perona. Human action recognition by sequence of movelet codewords. In *Proc. 3D Data Processing Visualization and Transmission*, 2002. 20

[92] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *Proc. CVPR*, 2008. xx, 43

[93] M. M. Fleck, D. A. Forsyth, and C. Bregler. Finding naked people. In *Proc. ECCV*, 1996. 18

[94] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9), Sept. 1995. xvii, 13, 15

[95] D. Forsyth and M. Fleck. Body plans. In *Proc. CVPR*, 1997. 18

[96] D. A. Forsyth, O. Arikan, and D. Ramanan. Computational studies of human motion: Part 1, tracking and motion synthesis. In *Foundations and Trends in Computer Graphics and Vision*, 2006. 20

[97] E. Francesconi, M.Gori, S.Marinai, and G.Soda. A serial combination of connectionist-based classifiers for OCR. *IJDAR*, 3, 2001. 97

[98] C. Frankel, M. Swain, and A. V. WebSeer: An Image Search Engine for the World Wide Web. Technical Report 96–14, University of Chicago Technical Report, 1996. 13

[99] F. Fraundorfer and H. Stewenius. A binning scheme for fast hard drive based image search. In *Proc. CVPR*, 2007. 17

[100] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith. Query by humming: Musical information retrieval in an audio database. In *Proc. ACM Multimedia*, 1995. 3

[101] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. VLDB*, 1999. 107

[102] V. Goel, A. Mishra, K. Alahari, and C. V. Jawahar. Whole is greater than sum of parts: Recognizing scene text words. In *Proc. ICDAR*, 2013. 124

[103] K. Grauman. *Matching Sets of Features for Efficient Retrieval and Recognition*. PhD thesis, MIT, 2006. 17

[104] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007. xviii, 19

[105] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *Proc. ICCV*, 2009. 25

[106] A. Gupta. *Beyond Nouns and Verbs*. PhD thesis, University of Maryland, College Park, 2009. xviii, 27

[107] A. Gupta, P. Srinivasan, J. Shi, and L. Davis. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *Proc. CVPR*, 2009. 27

[108] A. Hanjalic, G. C. Langelaar, P. M. V. Roosmalen, J. Biemond, and R. L. Langendijk. *Image and Video Databases: Restoration, Watermarking and Retrieval*. Elsevier Science Inc., 2000. 16

[109] S. Harding, W. B. Croft, and C. Weir. Probabilistic retrieval of OCR degraded text using n-grams. In *Proc. ECDL*, 1997. 128

[110] G. Harit, S. Chaudhury, and H. Ghosh. Managing document images in a digital library: An ontology guided approach. In *Proc. DIAL*, 2004. 92

[111] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 2006. 19

[112] C. Huang and S. N. Srihari. Mapping transcripts to handwritten text. In *Proc. IWFHR*, 2006. 132

[113] M. Huijbregts, R. Ordelman, and F. de Jong. Annotation of heterogeneous multimedia content using automatic speech recognition. In *Semantic Multimedia*, 2007. 48

[114] N. Ikizler and D. Forsyth. Searching video for complex activities with finite state models. In *Proc. CVPR*, 2007. 20

[115] E. Indermuhle, M. Liwicki, and H. Bunke. Combining alignment results for historical handwritten document analysis. In *Proc. ICDAR*, 2009. 132

[116] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. STOC*, 1998. 106, 107

[117] R. Jain, M. Sudhapraveen, Pramod Sankar K., and C. V. Jawahar. An indexing approach for speeding-up image classification. In *Proc. ICVGIP*, 2010. 11

[118] N. Jammalamadaka, A. Zisserman, M. Eichner, V. Ferrari, and C. V. Jawahar. Video retrieval by mimicking poses. In *Proc. ICMR*, 2012. 29

[119] H. Jegou, H. Harzallah, and C. Schmid. A contextual dissimilarity measure for accurate and efficient image search. In *Proc. CVPR*, 2007. 17

[120] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *Proc. ACM SIGIR*, 2003. 24

[121] J. Jeon and R. Manmatha. Automatic image annotation of news images with large vocabularies and low quality training data. In *Proc. ACM Multimedia*, 2004. 24

[122] S. Kahan, T. Pavlidis, and H. S. Baird. On the recognition of printed character of any font and size. *TPAMI*, 9(2), 1987. 98

[123] T. Kanungo, R. M. Haralick, H. S. Baird, W. Stuezle, and D. Madigan. A statistical, nonparametric methodology for document degradation model validation. *TPAMI*, 22, 2000. xxiii, 109, 110

[124] J. Kim, S. M. Seitz, and M. Agrawala. Video-based document tracking: Unifying your physical and electronic desktops. In *Proc. UIST*, 2004. 29

[125] M. H. Kolekar and S. Sengupta. A hierarchical framework for generic sports video classification. In *Proc. ACCV*, 2006. 65, 69

[126] T. Konidaris, B. Gatos, K. Ntzios, I. Pratikakis, S. Theodoridis, and S. J. Perantonis. Keyword-guided word spotting in historical printed documents with synthetic data and user feedback. *IJDAR*, 2007. 99, 100

[127] I. Koprinska and S. Carrato. Temporal video segmentation: A survey. *Signal Processing: Image Communication*, 2001. 16

[128] A. Kumar, C. V. Jawahar, and R. Manmatha. Efficient search in document image collections. In *Proc. ACCV*, 2007. 99, 112

[129] K. Kunze, H. Kawaichi, K. Yoshimura, and K. Kise. The wordometer–estimating the number of words read using document image retrieval and mobile eye tracking. In *Proc. ICDAR*, 2013. 29

[130] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proc. CVPR*, 2008. 26, 34

[131] I. Laptev and P. Perez. Retrieving actions in movies. In *Proc. ICCV*, 2007. 26

[132] V. Lavrenko, R. Manmatha, and J. Jeon. A model for learning the semantics of pictures. In *Proc. NIPS*. MIT Press, 2003. 24

[133] V. Lavrenko, T. M. Rath, and R. Manmatha. Holistic word recognition for handwritten historical documents. In *Proc. DIAL*, 2004. 100

[134] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, 2006. 17

[135] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Proc. NIPS*, 1989. 97

[136] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 1998. 98

[137] S. Lefevre, J. Holler, and N. Vincent. A review of real-time segmentation of uncompressed video sequences for content-based search and retrieval. *Real-Time Imaging*, 9(1), 2003. 16

[138] G. S. Lehal, C. Singh, and R. Lehal. A shape based post processor for gurumukhi OCR. In *Proc. ICDAR*, 2001. 98

[139] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2(1), 2006. 13, 16

[140] Y. Li, L. G. Shapiro, and J. A. Bilmes. A generative/discriminative learning algorithm for image classification. In *Proc. ICCV*, 2005. 26

[141] C. Liao, Q. Liu, B. Liew, and L. Wilcox. Pacer: Fine-grained interactive paper via camera-touch hybrid gestures on a cell phone. In *Proc. SIGCHI*, 2010. 29

[142] X. Liu and D. Doermann. Mobile retriever: Access to digital documents from their physical source. *IJDAR*, 11(1), Sept. 2008. 29

[143] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2), 2004. 16, 40, 41, 42, 111, 112

[144] Y. Lu, C. L. Tan, W. Huang, and L. Fan. An approach to word image matching based on weighted Hausdorff distance. In *Proc. ICDAR*, 2001. 98

[145] W. Ma and B. Manjunath. NeTra: A toolbox for navigating large image databases. In *Multimedia Systems*, 1999. xvii, 15

[146] A. Makadia, V. Pavlovic, and S. Kumar. A new baseline for image annotation. In *Proc. ECCV*, 2008. 25

[147] R. Manmatha and W. B. Croft. Word spotting: Indexing handwritten archives. In M. Maybury, editor, *Intelligent Multimedia Information Retrieval Collection*, 1997. xix, 29, 98

[148] S. Marinai, E. Marino, and G. Soda. Font adaptive word indexing of modern printed documents. *TPAMI*, 28(8), Aug., 2006. 92, 99

[149] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. BMVC*, 2002. 16

[150] M. Meshesha and C. V. Jawahar. Matching word images for content-based retrieval from printed document images. *IJDAR*, 2008. 99

[151] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *TPAMI*, 27(10), 2005. 16

[152] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *IJCV*, 65(1/2), 2005. 41

[153] G. A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11), 1995. 23

[154] A. Mishra, K. Alahari, and C. V. Jawahar. Image retrieval using textual cues. In *Proc. ICCV*, 2013. 23

[155] M. Mitra and B. B. Chaudhuri. Information retrieval from documents: A survey. *Information Retrieval*, 2(2-3), 2000. 92

[156] F. Monay and D. Gatica-Perez. On image auto-annotation with latent space models. In *Proc. ACM Multimedia*, 2003. 24

[157] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proc. VISAPP*, 2009. 15, 107, 108, 136, 142

[158] G. Nagy. At the frontiers of OCR. *Proceedings of the IEEE*, 80, 1992. 97

[159] T. Nakai, K. Kise, and M. Iwamura. Camera-based document image mosaicing using LLAH. In *Proc. DRR*, 2009. xix, 30

[160] P. Natarajan, E. MacRostie, , and M. Decerbo. The BBN Byblos Hindi OCR System. *Guide to OCR for Indic Scripts*, 2009. 98

[161] N. V. Neeba and C. V. Jawahar. Empirical evaluation of character classification schemes. In *Proc. ICAPR*, 2009. 97, 100

[162] A. Negi, C. Bhagvathi, and B. Krishna. An OCR System for Telugu. In *Proc. ICDAR*, 2001. 100

[163] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proc. CVPR*, 2006. 15, 102, 106, 112

[164] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42(3), 2001. 19

[165] A. Opelt, A. Fussenegger, A. Pinz, and P. Auer. Weak hypotheses and boosting for generic object detection and recognition. In *Proc. ECCV*, volume 2, 2004. 17

[166] U. Pal and B. Chaudhuri. Indian script character recognition: A survey. *Pattern Recognition*, 37(9), 2004. 92, 97

[167] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *Proc. ECCV*, 2010. 17

[168] J. Philbin. *Scalable Object Retrieval in Very Large Image Collections*. PhD thesis, University of Oxford, 2010. 17

[169] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proc. CVPR*, 2008. 17

[170] J. Philbin, M. Marin-Jimenez, S. Srinivasan, A. Zisserman, M. Jain, S. Vempati, Pramod Sankar K., and C. V. Jawahar. Oxford/iiit TRECVID 2008 – notebook paper. In *Proc. TRECVID Workshop*, 2008. xviii, 20

[171] J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, editors. *Toward Category-Level Object Recognition*, volume 4170 of *Lecture Notes in Computer Science*. Springer, 2006. 19

[172] Pramod Sankar K., R. Manmatha, and C. V. Jawahar. Large scale document image retrieval by automatic word annotation. *IJDAR*, 17(1), 2014. 11

[173] Pramod Sankar K., S. Pandey, and C. V. Jawahar. Text driven temporal segmentation of cricket videos. In *Proc. ICVGIP*, 2006. 12, 34

[174] Pramod Sankar K. and C. V. Jawahar. Enabling search over large collections of telugu document images - an automatic annotation based approach. In *Proc. ICVGIP*, 2006. 12

[175] Pramod Sankar K. and C. V. Jawahar. Probabilistic reverse annotation for large scale image retrieval. In *Proc. CVPR*, 2007. 12, 100, 109

[176] Pramod Sankar K., C. V. Jawahar and A. Zisserman. Subtitle-free movie to script alignment. In *Proc. BMVC*, 2009. 12

[177] Pramod Sankar K., C. V. Jawahar and R. Manmatha. Nearest neighbor based collection OCR. In *Proc. DAS*, 2010. 11, 135, 142

[178] D. Ramanan. *Tracking People and Recognizing their Activities*. PhD thesis, University of California, Berkeley, 2005. 20

[179] D. Ramanan, D. A. Forsyth, and A. Zisserman. Tracking people by learning their appearance. *TPAMI*, 29(1), 2007. 20

[180] V. Rasagna, A. Kumar, C. V. Jawahar, and R. Manmatha. Robust recognition of documents by fusing results of word clusters. In *Proc. ICDAR*, 2009. 110

[181] Z. Rasheed and M. Shah. Scene detection in hollywood movies and tv shows. In *Proc. CVPR*, volume 2, 2003. 16

[182] T. Rath and R. Manmatha. Features for word matching in historical manuscripts. In *Proc. ICDAR*, 2003. 98

[183] T. Rath and R. Manmatha. Word image matching using dynamic time warping. In *Proc. CVPR*, 2003. xvii, 14, 92, 98, 110, 111

[184] T. Rath, R. Manmatha, and V. Lavrenko. A search engine for historical manuscript images. In *Proc. SIGIR*, 2004. 28, 100, 112

[185] T. M. Rath and R. Manmatha. Word spotting for historical documents. *IJDAR*, 9(2-4), 2007. xix, 29, 99, 100, 133, 135, 142

[186] S. Ray. *The Filmscript of Agantuk - The Stranger*. TLM Books, 2003. 37, 58

[187] X. Ren and J. Malik. Learning a classification model for segmentation. In *Proc. ICCV*, volume 1, 2003. 16

[188] S. V. Rice, F. R. Jenkins, and T. A. Nartker. The fifth annual test of OCR accuracy. Technical report, UNLV, 1996. 100

[189] J. Richarz, S. Vajda, and G. A. Fink. Annotating handwritten characters with minimal human involvement in a semi-supervised learning strategy. In *Proc. ICFHR*, 2012. 132

[190] J. A. Rodriguez and F. Perronnin. Local gradient histogram features for word spotting in unconstrained handwritten documents. In *Proc. ICFHR*, 2008. 111, 113, 117

[191] U. Roy, N. Sankaran, Pramod Sankar K., and C. V. Jawahar. Character n-gram spotting on handwritten documents using weakly-supervised segmentation. In *Proc. ICDAR*, 2013. 11

[192] Y. Rui, A. Gupta, and A. Acero. Automatically extracting highlights for tv baseball programs. In *Proc. ACM Multimedia*, 2000. 20

[193] Y. Rui, T. S. Huang, and S.-f. Chang. Image retrieval: Past, present, and future. *Journal of Visual Communication and Image Representation*, 10, 1997. xvii, 14

[194] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: a power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5), 1998. 30

[195] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: A database and web-based tool for image annotation. *IJCV*, 77, 2008. 23

[196] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4), 1990. 30

[197] T. Sato, T. Kanade, E. Hughes, and M. Smith. Video OCR for digital news archives. In *Proc. Workshop on Content-Based Access of Image and Video Databases*, 1998. 23

[198] C. Schmid and R. Mohr. Matching by local invariants. Technical Report 2644, INRIA Rhne-Alpes, Grenoble, France, 1995. 16

[199] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting Image Databases from the Web. *TPAMI*, 33(4), apr 2011. 23

[200] E. D. Sciascio and M. Mongiello. Query by sketch and relevance feedback for content-based image retrieval over the web. *Journal of Visual Languages and Computing*, 10(6), 1999. 29

[201] N. Serrano, A. Savakis, and J. Luo. A computationally efficient approach to indoor/outdoor scene classification. In *Proc. ICPR*, 2002. 18

[202] J. Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 22, 1997. 16

[203] J. Sivic. *Efficient Visual Search of Images and Videos*. PhD thesis, University of Oxford, 2006. 17

[204] J. Sivic, M. Everingham, and A. Zisserman. Person spotting: video shot retrieval for face sets. In *Proc. CIVR*, 2005. 23

[205] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, 2003. xvii, 14, 16, 17, 40, 42

[206] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *Proc. MIR*, 2006. xviii, 19, 20, 31

[207] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *TPAMI*, 22(12), 2000. xvii, 3, 13, 14

[208] R. Smith. Limits on the application of frequency-based language models to OCR. In *Proc. ICDAR*, 2011. 128

[209] C. G. M. Snoek. *The Authoring Metaphor to Machine Understanding of Multimedia*. PhD thesis, University of Amsterdam, 2005. xix, 21, 28

[210] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *Proc. Workshop on Internet Vision*, 2008. 23

[211] T. Starner, A. Pentland, and J. Weaver. Real-time american sign language recognition using desk and wearable computer based video. *TPAMI*, 20(12), Dec. 1998. 20

[212] M. Sudhapraveen, Pramod Sankar K., and C. V. Jawahar. Character n-gram spotting in document images. In *Proc. ICDAR*, 2011. 11, 128

[213] M. Szummer and R. W. Picard. Indoor-outdoor image classification. In *Proc. Workshop on Content-based Access of Image and Video Databases*, 1998. 18

[214] K. Taghva, J. Borsack, and A. Condit. Evaluation of model-based retrieval effectiveness with OCR text. *ACM Transactions on Information Systems*, 14(1), 1996. 92

[215] K. Takeda, K. Kise, and M. Iwamura. Real-time document image retrieval for a 10 million nopages database with a memory efficient and stability improved LLAH. In *Proc. ICDAR*, 2011. xix, 30

[216] W. Tavanapong and J. Zhou. Shot clustering techniques for story browsing. *IEEE Transactions on Multimedia*, 6(4), 2004. 16

[217] E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. In *Proc. CVPR*, 2008. 16

[218] A. Torralba, R. Fergus, and W. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *TPAMI*, 30(11), Nov 2008. xviii, 22, 23

[219] Y. Varma and C. V. Jawahar. Image annotation using metric learning in semantic neighbourhoods. In *Proc. ECCV*, 2012. xviii, 25, 26

[220] Y. Varma and C. V. Jawahar. Exploring SVM for image annotation in presence of confusing labels. In *Proc. BMVC*, 2013. 26

[221] S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *Proc. CVPR*, June 2008. 16

[222] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 57(2), 2004. 45

[223] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. reCAPTCHA: Human-based character recognition via web security measures. *Science*, 321(5895), 2008. 110

[224] C. Vondrick, D. Ramanan, and D. Patterson. Efficiently scaling up video annotation with crowdsourced marketplaces. In *Proc. ECCV*, 2010. 23

[225] G. Wang and D. Forsyth. Object image retrieval by exploiting online knowledge resources. In *Proc. CVPR*, June 2008. 23

[226] J. Wang, J. Li, G. Wiederhold, and O. Firschein. System for screening objectionable images. *Computer Communications*, 21(15), 1998. 18

[227] J. Z. Wang, J. Li, and G. Wiederhold. Simplicity: Semantics-sensitive integrated matching for picture libraries. *TPAMI*, 23(9), 2001. 18

[228] K. Wang and S. Belongie. Word spotting in the wild. In *Proc. ECCV*, 2010. 23

[229] J. J. Weinman, E. Learned-Miller, and A. R. Hanson. Scene text recognition using similarity and a lexicon with sparse belief propagation. *TPAMI*, 31(10), 2009. 23

[230] P. Welinder. *Hybrid Human-Machine Vision Systems: Image Annotation using Crowds, Experts and Machines*. PhD thesis, California Institute of Technology, 2012. 23

[231] A. Wilson and A. Bobick. Parametric hidden markov models for gesture recognition. *TPAMI*, 21(9), 1999. 20

[232] P. Xiu and H. S. Baird. Scaling up whole-book recognition. In *Proc. ICDAR*, 2009. 100

[233] C. Xu, J. Wang, K. Wan, Y. Li, and L. Duan. Live sports event detection based on broadcast video and web-casting text. In *Proc. ACM Multimedia*, 2006. 65

[234] Y. Yacoob and L. S. Davis. Learned models for estimation of rigid and articulated human motion from stationary or moving camera. *IJCV*, 36(1), 2000. 20

[235] J. Yang and A. G. Hauptmann. 3wnews: who, where, and when in news video. In *Proc. ACM Multimedia*, 2006. 28

[236] J. Yang and A. G. Hauptmann. Annotating news video with locations. In *Proc. CIVR*, 2006. xix, 24, 28

[237] R. Zabih, J. Miller, and K. Mai. A feature-based algorithm for detecting and classifying production effects. *Multimedia Systems*, 7(2), 1999. 16

[238] B. Zhang, S. N. Srihari, and C. Huang. Word image retrieval using binary features. In *Proc. Document Recognition and Retrieval*, 2004. 99

[239] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, 73(2), 2007. 17

[240] Y. Zhang, X. Zhang, C. Xu, and H. Lu. Personalized retrieval of sports video. In *Proc. Workshop on Multimedia Information Retrieval*, 2007. 30, 65

[241] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computer Survey*, 35(4), 2003. 18

[242] X. D. Zhou, F. Yin, D. H. Wang, Q. F. Wang, M. Nakagawa, and C. L. Liu. Transcript mapping for handwritten text lines using conditional random fields. In *Proc. ICDAR*, 2011. 132