N^o d'ordre 03 ISAL 0074 Année 2003

Thèse

Détection de textes dans des images issues d'un flux vidéo pour l'indexation sémantique

Présentée devant
L'Institut National des Sciences Appliquées de Lyon

Pour obtenir
Le grade de docteur

École doctorale:
École Doctorale Informatique et Information pour la Société

Spécialité:
Informatique

Par
Christian Wolf

Soutenue le 5 Décembre 2003 devant la Commission d'examen

Jury MM.

Examinateur	C. Berrut	Université J. Fourier de Grenoble
Examinateur	D. Doermann	University of Maryland
Directeur	J.-M. Jolion	INSA de Lyon
Examinateur	C. Laurent	France Télécom R&D
Rapporteur	A. Montanvert	Université P. Mendès-France de Grenoble
Rapporteur	M. Revenu	Université de Caen

MAI 2003

INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE LYON

Directeur: STORCK A.

Professeurs :

AUDISIO S.		PHYSICOCHIMIE INDUSTRIELLE
BABOT D.		CONT. NON DESTR. PAR RAYONNEMENTS IONISANTS
BABOUX J.C		. GEMPPM***
BALLAND B.		PHYSIQUE DE LA MATIERE
BAPTISTE P.		PRODUCTIQUE ET INFORMATIQUE DES SYSTEMES MANUFACTURIERS
BARBIER D.		PHYSIQUE DE LA MATIERE
BASTIDE J.P.		LAEPSI****
BAYADA G.		MECANIQUE DES CONTACTS
BENADDA B.		LAEPSI****
BETEMPS M.		AUTOMATIQUE INDUSTRIELLE
BIENNIER F.		PRODUCTIQUE ET INFORMATIQUE DES SYSTEMES MANUFACTURIERS
BLANCHARD J.M.		LAEPSI****
BOISSON C.		VIBRATIONS-ACOUSTIQUE
BOIVIN M.		(Prof. émérite) MECANIQUE DES SOLIDES
BOTTA H.		UNITE DE RECHERCHE EN GENIE CIVIL - Développement Urbain
BOTTA-ZIMMERMANN (Mme)	M.	UNITE DE RECHERCHE EN GENIE CIVIL - Développement Urbain
BOULAYE G. (Prof. émérite)		INFORMATIQUE
BOYER J.C.		MECANIQUE DES SOLIDES
BRAU J.		CENTRE DE THERMIQUE DE LYON - Thermique du bâtiment
BREMOND G.		PHYSIQUE DE LA MATIERE
BRISSAUD M.		GENIE ELECTRIQUE ET FERROELECTRICITE
BRUNET M.		MECANIQUE DES SOLIDES
BRUNIE L.		INGENIERIE DES SYSTEMES D'INFORMATION
BUREAU J.C.		CEGELY*
CAVAILLE J.Y.		GEMPPM***
CHANTE J.P.		CEGELY*- Composants de puissance et applications
CHOCAT B.		UNITE DE RECHERCHE EN GENIE CIVIL - Hydrologie urbaine
COMBESCURE A.		MECANIQUE DES CONTACTS
COUSIN M.		UNITE DE RECHERCHE EN GENIE CIVIL - Structures
DAUMAS F. (Mme)		CENTRE DE THERMIQUE DE LYON - Energétique et Thermique
DOUTHEAU A.		CHIMIE ORGANIQUE
DUFOUR R.		MECANIQUE DES STRUCTURES
DUPUY J.C.		PHYSIQUE DE LA MATIERE
EMPTOZ H.		RECONNAISSANCE DE FORMES ET VISION
ESNOUF C.		GEMPPM***
EYRAUD L. (Prof. émérite)		GENIE ELECTRIQUE ET FERROELECTRICITE
FANTOZZI G.		GEMPPM***
FAVREL J.		PRODUCTIQUE ET INFORMATIQUE DES SYSTEMES MANUFACTURIERS
FAYARD J.M.		BIOLOGIE FONCTIONNELLE, INSECTES ET INTERACTIONS
FAYET M.		MECANIQUE DES SOLIDES
FERRARIS-BESSO G.		MECANIQUE DES STRUCTURES
FLAMAND L.		MECANIQUE DES CONTACTS
FLORY A.		INGENIERIE DES SYSTEMES D'INFORMATIONS
FOUGERES R.		GEMPPM***
FOUQUET F.		GEMPPM***
FRECON L.		REGROUPEMENT DES ENSEIGNANTS CHERCHEURS ISOLES
GERARD J.F.		INGENIERIE DES MATERIAUX POLYMERES
GERMAIN P.		LAEPSI****
GIMENEZ G.		CREATIS**
GOBIN P.F.		(Prof. émérite) GEMPPM***
GONNARD P.		GENIE ELECTRIQUE ET FERROELECTRICITE
GONTRAND M.		PHYSIQUE DE LA MATIERE
GOUTTE R. (Prof. émérite)		CREATIS**
GOUJON L.		GEMPPM***
GOURDON R.		LAEPSI****.
GRANGE G.		GENIE ELECTRIQUE ET FERROELECTRICITE
GUENIN G.		GEMPPM***
GUICHARDANT M.		BIOCHIMIE ET PHARMACOLOGIE
GUILLOT G.		PHYSIQUE DE LA MATIERE
GUINET A.		PRODUCTIQUE ET INFORMATIQUE DES SYSTEMES MANUFACTURIERS
GUYADER J.L.		VIBRATIONS-ACOUSTIQUE
GUYOMAR D.		GENIE ELECTRIQUE ET FERROELECTRICITE
HEIBIG A.		MATHEMATIQUE APPLIQUEES DE LYON
JACQUET-RICHARDET G.		MECANIQUE DES STRUCTURES
JAYET Y.		GEMPPM***
JOLION J.M.		RECONNAISSANCE DE FORMES ET VISION
JULLIEN J.F.		UNITE DE RECHERCHE EN GENIE CIVIL - Structures
JUTARD A. (Prof. émérite)		AUTOMATIQUE INDUSTRIELLE
KASTNER R.		UNITE DE RECHERCHE EN GENIE CIVIL - Géotechnique
KOULOUMDJIAN J.		INGENIERIE DES SYSTEMES D'INFORMATION
LAGARDE M.		BIOCHIMIE ET PHARMACOLOGIE
LALANNE M. (Prof. émérite)		MECANIQUE DES STRUCTURES
LALLEMAND A.		CENTRE DE THERMIQUE DE LYON - Energétique et thermique
LALLEMAND M.		(Mme) CENTRE DE THERMIQUE DE LYON - Energétique et thermique
LAUGIER A.		PHYSIQUE DE LA MATIERE
LAUGIER C.		BIOCHIMIE ET PHARMACOLOGIE

LAURINI R.	INFORMATIQUE EN IMAGE ET SYSTEMES D'INFORMATION
LEJEUNE P.	UNITE MICROBIOLOGIE ET GENETIQUE
LUBRECHT A.	MECANIQUE DES CONTACTS
MASSARD N.	INTERACTION COLLABORATIVE TELEFORMATION TELEACTIVITE
MAZILLE H.	PHYSICOCHIMIE INDUSTRIELLE
MERLE P.	GEMPPM***
MERLIN J.	GEMPPM***
MIGNOTTE A. (Mle)	INGENIERIE, INFORMATIQUE INDUSTRIELLE
MILLET J.P.	PHYSICOCHIMIE INDUSTRIELLE
MIRAMOND M.	UNITE DE RECHERCHE EN GENIE CIVIL - Hydrologie urbaine
MOREL R.	MECANIQUE DES FLUIDES ET D'ACOUSTIQUES
MOSZKOWICZ P.	LAEPSI****
NARDON P. (Prof. émérite)	BIOLOGIE FONCTIONNELLE, INSECTES ET INTERACTIONS
NIEL E.	AUTOMATIQUE INDUSTRIELLE
NORTIER P.	DREP
ODET C.	CREATIS**
OTTERBEIN M. (Prof. émérite)	LAEPSI****
PARIZET E.	VIBRATIONS-ACOUSTIQUE
PASCAULT J.P.	INGENIERIE DES MATERIAUX POLYMERES
PAVIC G.	VIBRATIONS-ACOUSTIQUE
PELLETIER J.M.	GEMPPM***
PERA J.	UNITE DE RECHERCHE EN GENIE CIVIL - Matériaux
PERRIAT P.	GEMPPM***
PERRIN J.	INTERACTION COLLABORATIVE TELEFORMATION TELEACTIVITE
PINARD P. (Prof. émérite)	PHYSIQUE DE LA MATIERE
PINON J.M.	INGENIERIE DES SYSTEMES D'INFORMATION
PONCET A.	PHYSIQUE DE LA MATIERE
POUSIN J.	MODELISATION MATHEMATIQUE ET CALCUL SCIENTIFIQUE
PREVOT P.	INTERACTION COLLABORATIVE TELEFORMATION TELEACTIVITE
PROST R.	CREATIS**
RAYNAUD M.	CENTRE DE THERMIQUE DE LYON - Transferts Interfaces et Matériaux
REDARCE H.	AUTOMATIQUE INDUSTRIELLE
RETIF J-M.	CEGELY*
REYNOUARD J.M.	UNITE DE RECHERCHE EN GENIE CIVIL - Structures
RIGAL J.F. MECANIQUE DES SOLIDES	
RIEUTORD E. (Prof. émérite)	MECANIQUE DES FLUIDES
ROBERT-BAUDOUY J. (Mme) (Prof. émérite)	GENETIQUE MOLECULAIRE DES MICROORGANISMES
ROUBY D.	GEMPPM***
ROUX J.J.	CENTRE DE THERMIQUE DE LYON - Thermique de l'Habitat
RUBEL P.	INGENIERIE DES SYSTEMES D'INFORMATION
SACADURA J.F.	CENTRE DE THERMIQUE DE LYON - Transferts Interfaces et Matériaux
SAUTEREAU H.	INGENIERIE DES MATERIAUX POLYMERES
SCAVARDA S.	AUTOMATIQUE INDUSTRIELLE
SOUIFI A.	PHYSIQUE DE LA MATIERE
SOUROUILLE J.L.	INGENIERIE INFORMATIQUE INDUSTRIELLE
THOMASSET D.	AUTOMATIQUE INDUSTRIELLE
THUDEROZ C.	ESCHIL - Equipe Sciences Humaines de l'Insa de Lyon
UBEDA S.	CENTRE D'INNOV. EN TELECOM ET INTEGRATION DE SERVICES
VELEX P.	MECANIQUE DES CONTACTS
VIGIER G.	GEMPPM***
VINCENT A.	GEMPPM***
VRAY D.	CREATIS**
VUILLERMOZ P.L. (Prof. émérite)	PHYSIQUE DE LA MATIERE

Directeurs de recherche C.N.R.S.:

BAIETTO-CARNEIRO M-C. (Mme)	MECANIQUE DES CONTACTS ET DES SOLIDES
BERTHIER Y.	MECANIQUE DES CONTACTS
CONDEMINE G.	UNITE MICROBIOLOGIE ET GENETIQUE
COTTE-PATAT N. (Mme)	UNITE MICROBIOLOGIE ET GENETIQUE
ESCUDIE D. (Mme)	CENTRE DE THERMIQUE DE LYON
FRANCIOSI P.	GEMPPM***
MANDRAND M.A.	(Mme) UNITE MICROBIOLOGIE ET GENETIQUE
POUSIN G.	BIOLOGIE ET PHARMACOLOGIE
ROCHE A.	INGENIERIE DES MATERIAUX POLYMERES
SEGUELA A.	GEMPPM***

Directeurs de recherche I.N.R.A. :

FEBVAY G.	BIOLOGIE FONCTIONNELLE, INSECTES ET INTERACTIONS
GRENIER S.	BIOLOGIE FONCTIONNELLE, INSECTES ET INTERACTIONS
RAHBE Y.	BIOLOGIE FONCTIONNELLE, INSECTES ET INTERACTIONS

Directeurs de recherche I.N.S.E.R.M. :

PRIGENT A.F. (Mme)	BIOLOGIE ET PHARMACOLOGIE
MAGNIN I. (Mme)	CREATIS**

* CEGELY	CENTRE DE GENIE ELECTRIQUE DE LYON
* CREATIS	CENTRE DE RECHERCHE ET D'APPLICATIONS EN TRAITEMENT DE L'IMAGE ET DU SIGNAL
**GEMPPM	GROUPE D'ETUDE METALLURGIE PHYSIQUE ET PHYSIQUE DES MATERIAUX
***LAEPSI	LABORATOIRE D'ANALYSE ENVIRONNEMENTALE DES PROCEDES ET SYSTEMES INDUSTRIELS

MARS 03

Ecoles Doctorales et Diplômes d'Etudes Approfondies

habilités pour la période 1999-2003

ECOLES DOCTORALES n° code national	RESPONSABLE PRINCIPAL	CORRESPONDANT INSA	DEA INSA n° code national	RESPONSABLE DEA INSA
CHIMIE DE LYON **(Chimie, Procédés, Environnement)** **EDA206**	M. D. SINOU UCBL1 04.72.44.62.63 Sec 04.72.44.62.64 Fax 04.72.44.81.60	**M. R. GOURDON** 87.53 Sec 84.30 Fax 87.17	Chimie Inorganique **910643** Sciences et Stratégies Analytiques **910634** Sciences et Techniques du Déchet **910675**	M. R. GOURDON Tél 87.53 Fax 87.17
ECONOMIE, ESPACE ET MODELISATION DES COMPORTEMENTS **(E^2MC)** **EDA417**	M.A. BONNAFOUS LYON 2 04.72.72.64.38 Sec 04.72.72.64.03 Fax 04.72.72.64.48	**Mme M. ZIMMERMANN** 60.91 Fax 87.96	Villes et Sociétés **911218** Dimensions Cognitives et Modélisation **992678**	Mme M. ZIMMERMANN Tél 60.91 Fax 87.96 M. L. FRECON Tél 82.39 Fax 85.18
ELECTRONIQUE, ELECTROTECHNIQUE, AUTOMATIQUE **(E.E.A.)** **EDA160**	**M. D. BARBIER** INSA DE LYON 85.47 Fax 60.82		Automatique Industrielle **910676** Dispositifs de l'Electronique Intégrée **910696** Génie Electrique de Lyon **910065** Images et Systèmes **992254**	M. M. BETEMPS Tél 85.59 Fax 85.35 M. D. BARBIER Tél 85.47 Fax 60.82 M. J.P. CHANTE Tél 87.26 Fax 85.30 Mme I. MAGNIN Tél 85.63 Fax 85.26
EVOLUTION, ECOSYSTEME, MICROBIOLOGIE , MODELISATION **(E2M2)** **EDA403**	**M. J.P FLANDROIS** UCBL1 04.78.86.31.50 Sec 04.78.86.31.52 Fax 04.78.86.31.49	**M. S. GRENIER** 79.88 Fax 85.34	Analyse et Modélisation des Systèmes Biologiques **910509**	M. S. GRENIER Tél 79.88 Fax 85.34
INFORMATIQUE ET INFORMATION POUR LA SOCIETE **(EDIIS)** **EDA 407**	**M. J.M. JOLION** INSA DE LYON 87.59 Fax 80.97		Documents Multimédia, Images et Systèmes d'Information Communicants **992774** Extraction des Connaissances à partir des Données **992099** Informatique et Systèmes Coopératifs pour l'Entreprise **950131**	M. A. FLORY Tél 84.66 Fax 85.97 M. J.F. BOULICAUT Tél 89.05 Fax 87.13 M. A. GUINET Tél 85.94 Fax 85.38
INTERDISCIPLINAIRE SCIENCES-SANTE **(EDISS)** **EDA205**	**M. A.J. COZZONE** UCBL1 04.72.72.26.72 Sec 04.72.72.26.75 Fax 04.72.72.26.01	**M. M. LAGARDE** 82.40 Fax 85.24	Biochimie **930032**	M. M. LAGARDE Tél 82.40 Fax 85.24
MATERIAUX DE LYON **UNIVERSITE LYON 1** **EDA 034**	**M. J. JOSEPH** ECL 04.72.18.62.44 Sec 04.72.18.62.51 Fax 04.72.18.60.90	**M. J.M. PELLETIER** 83.18 Fax 85.28	Génie des Matériaux : Microstructure, Comportement Mécanique, Durabilité **910527** Matériaux Polymères et Composites **910607** Matière Condensée, Surfaces et Interfaces **910577**	M. J.M.PELLETIER Tél 83.18 Fax 85.28 M. H. SAUTEREAU Tél 81.78 Fax 85.27 M. G. GUILLOT Tél 81.61 Fax 85.31
MATHEMATIQUES ET INFORMATIQUE FONDAMENTALE **(Math IF)** **EDA 409**	**M. F. WAGNER** UCBL1 04.72.43.27.86 Fax 04.72.43.00.35	**M. J. POUSIN** 88.36 Fax 85.29	Analyse Numérique, Equations aux dérivées partielles et Calcul Scientifique **910281**	M. G. BAYADA Tél 83.12 Fax 85.29
MECANIQUE, ENERGETIQUE, GENIE CIVIL, ACOUSTIQUE **(MEGA)** **EDA162**	**M. J. BATAILLE** ECL 04.72.18.61.56 Sec 04.72.18.61.60 Fax 04.78.64.71.45	**M. G.DALMAZ** 83.03 Fax 04.72.89.09.80	Acoustique **910016** Génie Civil **992610** Génie Mécanique **992111** Thermique et Energétique **910018**	M. J.L. GUYADER Tél 80.80 Fax 87.12 M. J.J.ROUX Tél 84.60 Fax 85.22 M. G. DALMAZ Tél 83.03 Fax 04.78.89.09.80 M. J. F. SACADURA Tél 81.53 Fax 88.11

En grisé : Les Ecoles doctorales et DEA dont l'INSA est établissement principal

Remerciements

C'est fait, j'ai ma thèse dans mes mains, le fruit de trois années de travail. Il reste à remercier tous ceux qui on partagé ces trois années avec moi.

Tout d'abord, je remercie Mme. Catherine Berrut pour l'honneur qu'elle m'a fait en acceptant d'être la présidente de mon jury de thèse, et Mme. Annick Montanvert et Mme. Marinette Revenu, rapporteurs de cette thèse, qui ont accepté de consacrer beaucoup de leurs temps à la lire et à juger mon travail.

Je tiens à sincèrement remercier Jean-Michel Jolion pour son encadrement. C'est avec lui que j'ai appris ce que c'est, la recherche, la curiosité de savoir plus. Malgré ses nombreuses responsabilités, il était toujours disponible pour toutes les discussions, ce qui m'a permis profiter de ses compétences et de son expérience.

Mes remerciements s'adressent aussi à Frank LeBourgeois pour des nombreuses discussions sur la vision en général et les documents numériques, qui m'ont beaucoup aidés depuis le début de mon premier stage à Lyon il y a 4 ans. De plus, je le remercie aussi pour les discussions sur les sujets hors travail et son accueil des doctorants du laboratoire dans sa piscine pendant les canicules ....

Je remercie Françoise Chassaing et Christophe Laurent de France Télécom R&D pour leur aide et aussi pour leur patience pendant la rédaction de ce mémoire.

Sans oublier, tous les membres de l'equipe du laboratoire LAMP de l'Université de Maryland, qui m'ont accueilli pendant mes deux séjours. Mes remerciements sont plus particulierment destinés à David Doermann et Daniel DeMenthon pour le travail fructueux ainsi que pour leur gentillesse et leur soutien. Merci aussi à Volker, et à Jan et Danitza, l'autre couple multi-culturel de College Park, pour les bons moments passés ensemble.

Un grand merci, à tous les membres du laboratoire LIRIS pour l'accueil chaleureux qu'ils ont donnée à un Autrichien fou qui a décidé de venir en France. D'autant plus que avant chaque soirée que nous avons organisée, ils ont du faire face à la même question (*Qu'est-ce qu'on fait à manger pour Christian?*).

Je remercie aussi Jean-Louis Léonhardt pour nos discussions précieuses sur la recherche selon mon compatriote Sir Karl Popper, qui m'ont profondement influencé.

Enfin, je remercie ma famille, mes parents et mes amis pour leur soutien, soit-il scientifique, humain ou spirituel. Sans eux, mon travail n'aurait pas abouti. Merci à Markus pour sa patience et son écoute. Un grand merci à ma femme Madiha, pour ses encouragements, ses conseilles et son soutien, ainsi que pour sa compréhension.

Détection de textes dans des images issues d'un flux vidéo pour l'indexation sémantique

Résumé

Ce travail rentre dans le cadre de l'indexation d'images et de vidéos. Les systèmes disponibles pour chercher dans les bases des documents audiovisuels travaillent sans connaissance, ils utilisent des méthodes de traitement d'image pour extraire des caractéristiques de bas niveau. Nous utilisons le texte présent dans les images et les vidéos.

Les méthodes de détection de texte présentées dans la littérature sont très simples: la plupart sont basées sur l'estimation de la texture ou sur la détection des contours suivie par l'accumulation de ces caractéristiques. Des contraintes géométriques sont imposées par presque toutes les méthodes, par contre ceci se fait seulement dans une deuxième phase de post-traitement. Cependant, une faible détection ne peut pas toujours être corrigée par un post-traitement.

Nous proposons la prise en compte des caractéristiques géométriques directement dans la phase de détection. Malheureusement il s'agit ici d'une version du problème de la poule et de l'oeuf. En effet, pour estimer les caractéristiques géométriques, il faut d'abord détecter le texte. En conséquence, nous avons proposé un approche en deux étapes: une première détection grossière sert à calculer une image de probabilité de texte; ensuite, pour chaque pixel, nous calculons une estimation robuste des caractéristiques géométriques de la boite de texte de laquelle elle fait éventuellement partie. Ces caractéristiques sont rajoutées aux caractéristiques de la première étape de détection. L'apprentissage se fait avec un classificateur de type "Support Vector Machines".

Pour la segmentation des caractères nous proposons deux algorithmes différents: le premier algorithme est basé sur la maximisation d'un critère de contraste; la deuxième approche exploite des connaissances a priori sur la répartition locale des pixels "texte" et "non-texte" pour aider à la décision de seuillage. Un modèle statistique (en utilisant un modèle de champs de Markov) est élaboré pour les images de vidéo dont les paramètres peuvent être appris par des images d'apprentissage. Le modèle statistique de texte est intégré dans un modèle bayésien d'estimation pour obtenir une estimation de l'image originale binaire.

Pour l'indexation de la vidéo nous présentons une méthode basée sur l'intégration de plusieurs caractéristiques extraites de la vidéo. En outre, le texte extrait est une des sources d'information pour l'algorithme d'indexation.

Mots cléfs

Détection de texte, reconnaissance, OCR, indexation sémantique, indexation de la vidéo par le contenu

Text detection in images taken from video sequences for semantic indexing

Abstract

This work situates itself within the framework of image and video indexation. The systems currently available for the content based image and video retrieval work without semantic knowledge, i.e. they use image processing methods to extract low level features of the data. The similarity obtained by these approaches does not always correspond to the similarity a human user would expect. A way to include more semantic knowledge into the indexing process is to use the text included in the images and video sequences. It is rich in information but easy to use.

Existing methods for text detection are simple: most of them are based on texture estimation or edge detection followed by an accumulation of these characteristics. Geometrical contraints are enforced by most of the methods. However, it is done in a morphological post-processing step only. It is obvious, that a weak detection is very difficult — up to impossible — to correct in a post-processing step. We propose to take into account the geometrical constraints directly in the detection phase. Unfortunately, this is a chicken-egg problem: in order to estimate geometrical constraints, we first need to detect text. Consequently, we suggest a two-step algorithm: a first coarse detection calculates a text "probability" image. Afterwards, for each pixel we calculate geometrical properties of the eventual surrounding text rectangle. These features are added to the features of the first step and fed into a support vector machine classifier.

For the application to video sequences, we propose an algorithm which detects text on a frame by frame basis, tracking the found text rectangles accross multiple frames. For each text appearance, a single enhanced image is robustly created by multiple frame integration.

We tackle the character segmentation problem and suggest two different methods: the first algorithm maximizes a criterion based on the local contrast in the image. The second approach exploits a priori knowledge on the spatial distribution of the text and non-text pixels in the image in order to enhance the segmentation decisions. The a priori knowledge is learned from training images and stored in a statistical Markov random field model. This model is integrated into Bayesian estimation framework in order to obtain an estimation of the original binary image.

We address the video indexing challenge with a method integrating several features extracted from the video. Among others, text extracted with the method mentioned above, is one of the informations sources for the indexing algorithm.

Keywords

Text detection, recognition, OCR, semantic indexing, content based video retrieval

Contents

Part I

Main part

Chapter 1

Introduction générale

Grau, teurer Freund, ist alle Theorie.
Grün ist des Lebens goldener Baum.

Theory, my friend, is grey.
But green is the eternal tree of life.

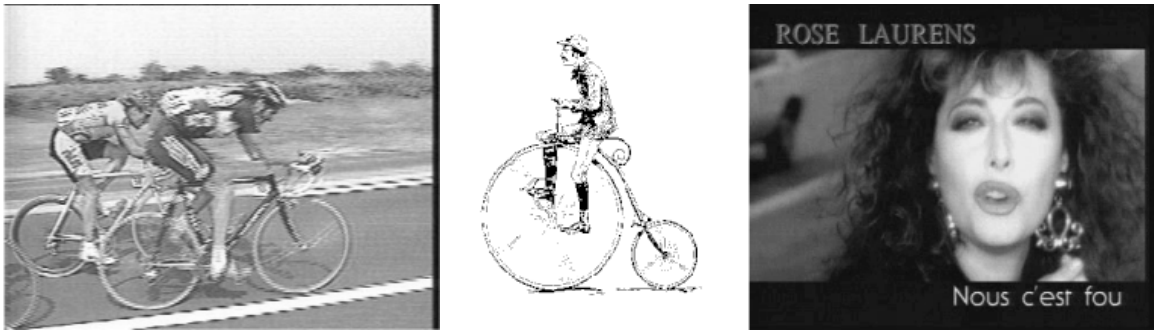
Johann Wolfgang von Goethe
German poet (1749 - 1832)

L'indexation d'images et de séquences vidéos est un domaine de recherche qui a reçu beaucoup d'attention depuis quelques années. Les diverses méthodes présentées dans la littérature permettent de faire des requêtes sur des grandes bases de données de contenu multimédia (images, vidéos etc.). Les caractéristiques employées sont calculées par des méthodes de traitement d'images de bas niveau, les mesures de distance sont construites pour être les plus proches que possible du système visuel humain. Malheureusement les requêtes construites ne se traduisent pas toujours dans des résultats similaires à ceux obtenus par un humain qui interprète le contenu du document.

La cause principale de cet échec est le manque de sémantique dans l'information bas niveau extraite des images . Que représente la sémantique? En cherchant dans le dictionnaire Larousse, on trouve: *"Sémantique : les études de sens des mots"*. Par analogie, dans le contexte de l'image, la sémantique représente donc le sens de l'image.

Considérons l'image de la figure 8.1a. Un utilisateur humain qui choisit cette image comme image requête est probablement intéressé par des images de cyclistes (notons que nous ne pouvons pas deviner le vrai désir de l'utilisateur et par conséquent, la similarité entre deux images n'est donc pas accessible). D'autre part les systèmes d'indexation trouvent des images qui sont similaires par rapport aux caractéristiques de bas niveau. Les images contenant des cyclistes qui ne sont pas similaires par rapport à ces caractéristiques (voir figure 1.1b) ne seront pas trouvées.

Dans la littérature on trouve des approches qui tentent d'extraire de l'information sémantique à partir des caractéristiques de bas niveau [55]. Malheureusement ces



(a) (b) (c)

Figure 1.1: Images d'exemples.

méthodes sont peu robustes, elles dépendent des grandes bases contenant des concepts prédéfinis et leurs représentations de bas niveau. Une autre façon d'inclure plus d'information sémantique dans le processus d'indexation est le bouclage de pertinence. Dans le cadre d'un processus interactif, l'annotation des résultats d'une requête par l'utilisateur est utilisée pour contrôler la prochaine requête. Une bonne introduction à ce sujet peuvent être trouvée dans [68] et [61].

Entre le niveau du pixel et celui de la sémantique, il existe des caractéristiques à la fois riches en information et cependant simples. Le texte présent dans les images et les vidéos fait partie de cette catégorie. La figure 1.1c montre une image extraite d'une publicité fournie en requête par un utilisateur. On peut imaginer que celui-ci cherche l'image de la chanteuse. La sémantique de cette image exemple peut être résumée par: *"Une publicité pour le nouvel album de la chanteuse Rose Laurens avec une photo de son visage"*. Il est impossible de déduire cette information à partir des caractéristiques basses. Pourtant dans ce cas le nom de la chanteuse est présent dans l'image. Par conséquent, en extrayant le texte on obtient une information supplémentaire très valable. Les requêtes classiques peuvent donc être complétées par des mots-cléfs.

En général, les systèmes d'indexation basés sur un mélange de caractéristiques de domaines différents (images et texte) sont très promettants, comparés avec les systèmes mono-domaine [21]. Par contre, les mots-clefs ne sont pas disponibles pour toutes les images. En outre, même s'ils sont très reliés au contenu sémantique de certaines séquences vidéos (voir la figure 1.2), les mots-clefs dépendent du point de vue de l'utilisateur — on parle ainsi de la polysémie de l'image.

Nous présentons dans ce travail un projet visant à la détection et la reconnaissance du texte présent dans des images ou des séquences vidéo. Le texte détecté et reconnu est complété par un lien vers la vidéo et le numéro de frame avant d'être stocké dans une base de données. L'utilisateur peut lancer une requête en soumettant un ou plusieurs mots-clefs, qui sont ensuite appariés d'une façon robuste avec le texte extrait et stocké dans la base. Les vidéos contenant les mots-clefs (ou une partie) sont présentés à l'utilisateur. Ce système peut être combiné avec des caractéristiques différentes, par exemple la couleur,

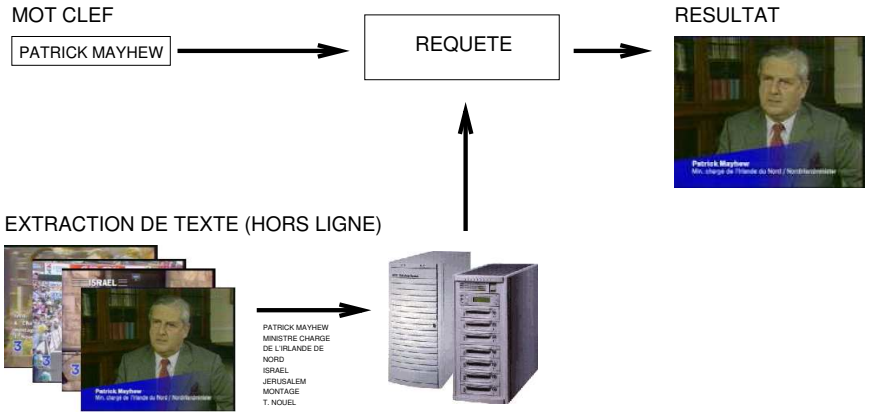


Figure 1.2: Indexation basée mot-clefs utilisant des textes extraits automatiquement.

la texture ou des caractéristiques extraites de la bande sonore, comme la parole reconnue.

Nous avons proposé un tel système "*mixte*" pour la compétition d'indexation de la vidéo dans le cadre de la conférence TREC 2002 (voir chapitre 7). Cette conférence est organisée annuellement par l'organisation américaine *National Institute of Standards and Technology* pour avancer la recherche dans le domaine de l'indexation. L'algorithme d'extraction de texte des séquences vidéos développé dans ce travail était utilisé dans ce but. En collaboration avec une équipe de l'Université de Maryland, nous avons développé un algorithme d'indexation basé sur des caractéristiques diverses telles que le texte, la parole, la couleur ainsi que des caractéristiques binaires [82].

Le travail présenté dans cette thèse était conçu dans le cadre de deux contrats industriels avec France Télécom. Plus précisément, il s'agit des contrats ECAV I & II ("Enrichissement du Contenu Audiovisuel") avec les références respectives 001B575 et 0011BA66. Les résultats donnent lieu à dépôt de deux brevets [83][84].

La partie principale de cette thèse est rédigée en anglais. Cela a pour but de rendre accessible ce document à la communité de recherche internationale.

Le chapitre 2 suit le parcours du problème jusqu'à la solution en commençant avec une motivation et une description du problème. Nous élaborons les propriétés du texte dans les documents multimédia, c.à.d. nous traitons la question: quelle est la nature du texte d'un point de vue d'un traiteur d'image aussi que d'un point de vue d'un documentaliste. Nous donnons un état de l'art de la détection et de la reconnaissance de texte en établissant un rapport entre la littérature et les propriétés de texte introduits auparavant. La chapitre conclue avec un résumé de notre solution proposée.

Les chapitres 3 et 4 traitent le problème de la détection de texte dans les images ou des frames prises des séquences vidéos. Deux méthodes différentes basées sur des philosophies différentes sont présentées. La première méthode suppose qu'il y a du texte dans l'image, pour ensuite séparer les deux populations — les pixels "*textes*" et les pixels "*non-textes*". La deuxième méthode crée un modèle des propriétés du signal et des propriétés géométriques du texte. Les paramètres du modèle sont appris à partir

d'une base d'apprentissage à l'aide des machines à vecteurs supports.

Le chapitre 5 est consacré aux aspects temporels de la détection de texte. Les méthodes pour la détection statique de texte introduites dans le chapitre précédent sont intégrés dans un algorithme capable de détecter et suivre du texte dans les séquences vidéos. Pour augmenter la qualité de reconnaissance, les apparences de texte suivies sont améliorées en intégrant plusieurs frames en une seule image de meilleure qualité.

Dans le chapitre 6 nous adressons le problème de la segmentation de caractères du texte reconnus en vue de les reconnaître. Nous proposons deux algorithmes différents: le premier est basé sur la maximisation d'un critère de contraste pour séparer les pixels "*textes*" et "*non-textes*" dans une fenêtre locale. La deuxième méthode considère le problème de binarisation comme un problème d'estimation d'une image binaire dans un cadre Bayesien. La connaissance a priori apprise sur la répartition spatiale des pixels "*textes*" et "*non-textes*" est stockées dans les paramètres d'un modèle statistique de type champs de Markov. L'estimation est fait avec un algorithme de type recuit simulé. La reconnaissance de caractères est conduite par un produit OCR commercial.

Le chapitre 7 traite le problème de l'indexation sémantique des séquences vidéos. Le texte détecté et reconnu par les algorithmes présentés dans les chapitres précédents est complété par d'autres caractéristiques (la couleur, la parole etc.). Un système de requête et de navigation interactive est introduit, permettant de chercher — à partir d'une demande formulée au niveau sémantique — des plans dans une grande base de vidéos.

Enfin, le chapitre 8 donne une conclusion et des perspectives de notre travail.

Ce travail a donné lieu a plusieurs publications citées dans l'annexe. La partie II du document contient quelques articles publiés en français concernant ce travail.

Notation

Dans ce document nous avons adopté la notation suivante:

- $\boldsymbol{A}$ Les matrices et les images sont écrites en lettres gras et majuscules. Dans la plupart des formules nous utilisons des opérations de traitement de signal plutôt que de l'algèbre linéaire. Pour cette raison les images et les matrices sont accédées d'une façon qui est typique pour le domaine de traitement de signal, c.à.d. les indices sont écrits dans l'ordre suivant: colonne (coordonnée x), ligne (coordonnée y): $\boldsymbol{A}_{x,y}$ or $\boldsymbol{A}(x, y)$. Si la notation diverge de cette convention nous le précisons explicitement.
- $\boldsymbol{v}$ Les vecteurs sont écrits en lettres gras et miniscules.
- i, T Les valeurs scalaires sont écrites en lettres normales et minuscules ou majuscules.

Chapter 2

From the problem to the solution

Deux excés:
exclure la raison,
n'admettre que la raison.

Two extremes:
to exclude reason,
to admit reason only.

Blaise Pascal
French mathematician and physicist (1623 - 1662)

Detection and recognition of text has been a task for at least 30 years in the document analysis community. We now have commercial software, which is able to regenerate a text document with low error rate from a scanned input image, where most parts of the document structure and organization are recognized and rebuilt. We may ask the question why the methods and techniques developed for documents cannot be applied straightforward to videos. The next sections deal with the differences between (printed) document data and multimedia data from a documentation specialists point of view as well as from a computer vision scientist's point of view. We describe the essential properties of text which can be exploited for detection, followed by a state of the art of text detection and a description of the philosophy of our own solution.

2.1 The role of text in videos

Our research team is made of computer vision scientists as well as specialists in the domain of the document in video indexing. In this context, we collaborate with the *Institut National de l'Audiovisuel* (INA)[1] [85], which is in charge of the archive of the French public television broadcasts. It's main task is the preservation of the national

[1] http://www.ina.fr

cultural heritage, i.e. the collection, protection, digitization, restoration if needed and the offer to access the documents.

In order to able to access a database of this dimension, navigation must be made as easy as possible. Browsing of the material can be supported by emphasizing and retrieving scenes and documents according to physical (low level) and/or semantic features. Describing the contents of the videos is a difficult problem since there is no description which is able to "predict" all possible future use-cases. We collide with the well known problem of polysemy of images and a fortiori of video sequences. For example, a video showing Henri-Cartier Bresson in China at the beginning of the nineteen-fifties can be perceived in multiple ways: the interest can be focused on the information about the photographer, on communist China, or even on the shooting methods used in the movie. There is not one, but several reading scenarios. A utopian "good" description should take into account *all* these aspects.

The semantic gap

Before going into detail of the role of text in information retrieval, we should recall the crucial problem in image and video analysis: the problem of the semantic gap. Smeulders et al. [68] illuminate a crucial problem inherent to the automatic treatment of visual documents, namely the conflict between the representation of the image and its meaning. They define the *sensory gap* : "The sensory gap is the gap between the object in the world and the information in a (computational) description derived from a recording of that scene". It is evident that the difficulties in the design of visual information retrieval systems are here: an image can be interpreted in numerous different ways, unlike textual documents which have a unique meaning.

Text is a particular semantic primitive: unlike other features, it possesses an "immediate" meaning and provides information without any necessary interpretation effort. Although text can be (and will be) interpreted subjectively, it is nevertheless easier to exploit than low level image features.

The importance of the documentary notes

At INA, the key point of the documentation process are the documentary notes written by the information officers. These notes have a primordial role during the processes of filing, documentation and retrieval of documents. The contents of these notes is miscellaneous but we shall emphasize the document summary, the names of the actors or the speakers, the names of the technical team and the elementary information as the broadcasting date or the document name. The conception of the notes is centered on a thesaurus organized around 120 fields with 6 deep levels and containing 8600 descriptors with 160 000 named entities.

However, the writing of the most descriptive field, namely that of the summary, is unconstrained: the contents as well as the structure only depend on the subjective interpretation by the information officer. The importance of the notes thus amply justifies

research aiming at the simplification of the work connected to their writing. Indeed, the text appearing on the screen often contains information which can be exploited for the description process. A more quantitative study on the example of a single television news program shows that 48% of the text can be found in the documentary notes. The comparison has been done using a simple spelling equality. A study based on natural language processing would allow to study a more powerful similarity, notably at the semantic level. However, the existing experiments already justify the usage of the text during the process of writing the documentary notes.

Indexing, browsing and segmentation support

Automatic indexing could be more effective if performed on the terms from the documentary notes, which are obviously more numerous. Therefore, exploiting the detected text could be more attractive for documents which have not been annotated yet.

However, text can be used for browsing. In fact, it is common that a query concerns a precise segment of a document rather than the whole document itself. The query does not generally result in this segment but rather in a set of documents which need to be visualized and watched by the officer to refine the selection. Detected text can facilitate the browsing effort by indicating the locations of particular sections, e.g. interviews.

Furthermore, it is possible to create a module which allows to exploit text for segmentation purposes ("storyboard segmentation"). In that case, the detected text does not act as a descriptor but controls the structuring process. This is especially true for television news programs, which possess a well defined structure. It is common that the subjects are presented by an insert mentioning their title, or giving some indications on their contents. It is also common that they are closed by text mentioning the name of the journalists involved in its conception. It is also possible that the segmentation into scenes or even into shots benefits from a multi-modal treatment integrating the information resulting from different sources, as e.g. face detection or speech transcription.

Multi-modal interaction

It is often useful to combine detected text with other features, e.g. speech recognition and face detection (see our experiments in the framework of the TREC video track competition in chapter 7). In fact, in newscasts it is common that a speaker has its name overlaid on the screen, either during the interview itself or in the credits (narrators etc.). This allows the association of the speech transcription or the face with the name of the person, resulting in an identity card of the speakers.

Another possible way to exploit detected text is the support of automatic speech recognition. The efficiency of speech recognition algorithms highly depends on uptodate and rich lexica [2]. The essential contribution of text extraction would be to feed the models with terms connected to current events. The most interesting case are named entities.

The roles between detection and indexing modules may be reversed. The a priori

knowledge on text in television programs may be used to increase the performance of the detection system. Indeed, certain types of text, whose shape and contents can be modeled, play a structuring role in television news programs. By modeling this kind of text it is possible to restrict the detection process to text conforming to the model, i.e. valid in the sense of a specific application. We thus perform "goal-directed" text detection. Researchers of our team are currently working on an adaptation of the detection method presented in this thesis for directed detection [37]. Until now, the work presented by the computer vision community essentially concerns the problem of unconstrained detection and the extraction without practicing any limitation on the kind of text to detect.

The types of text in videos

In computer vision literature, text is usually classified into the following two categories:

Artificial text is superimposed on the signal after the video has been taken with the camera. It is designed to be read, therefore it is most times rendered with high contrast against the background. Examples are subtitles in movies, names of locations or interviewed people, sports or lottery results, etc.

Scene text is part of the original scene taken with the camera. It has not been designed to be read, therefore it may be very small, partly occluded, badly illuminated, heavily distorted by perspective projection or by deformations of the supporting material (e.g. textiles). Examples are text on T-shirts, billboards, traffic signs etc.

However, in addition to this classification, the documentation specialists at INA also distinguish a third category:

Targeted scene text[2] belongs to the class of scene text as it is a part of the scene. However, the director of the video specifically targeted this text in order to guide the viewers attention to it, e.g. by zooming into it. In most cases the text should be readable, although distortions may very well occur. For the computer vision scientist, a particular distinction of targeted scene text does not make much sense.

Our detection system has been designed for indexing, therefore we concentrated our efforts on artificial text. However, our system is not restricted to it.

2.2 Multimedia documents vs. printed documents

The differences between document images and video sequences are due to their different goals and their different transport and usage, as well as their different contents. The sole

[2] "Targeted scene text" = "texte mis en scéne"

purpose of document images is their usage as input to document recognition software (OCR and structure recognition software). The result of this process is a recognized document file, the document image will be almost always be deleted. Therefore, restrictions as a maximum file size are not necessary, the whole process can be optimized for a maximum recognition rate. During the scanning stage a resolution of 300 - 400 dpi is used, which results in large image sizes (100 MB for a grayscale A4 page scanned at 400 dpi). The files are stored uncompressed or compressed without loss, which keeps the full image quality and prevents artifacts. Last but not least, the contents of document images is far less complex than that of video frames. An average document image contains mostly text, where each text word/line is sufficiently long. Background as well as text color are uniform, therefore character segmentation is easy and can be performed by simply thresholding the grayscale image. Pictures in the document are well separated from the text regions and are treated separately.

Video frames on the other hand contain more difficult data from the processing point of view. Text is not separated from the images but either superimposed like subtitles or sports results (artificial text) or even part of the image like the text on a T-shirt of an interviewed person (scene text). The background can be arbitrarily complex, which makes character segmentation difficult. Unlike document images which contain mostly two colors (foreground and background), video frames contain rich gray scale or color information which has to be treated accordingly. The text is not the main information of the image and is not always structured into lines. It is possible, that short single words are floating unconnected in the image.

The primary goal of a digital video file is not to be processed by a text recognition software but to be displayed with sufficient quality — or what is supposed to be good quality *as humans perceive it* — and to be stored in databases and transfered over networks. For this reason, and because image sequences need much more storage space than single images, file size is a big issue. Two methods are applied to decrease the size of the data: Limiting the spatial resolution and data compression. Sophisticated image and video compression methods as JPEG, MPEG 1, 2, 4 etc. have been developed to fulfill these goals. To gain a maximum compression factor, the compression methods are lossy, which means that they attempt to eliminate redundant or unnecessary information and lose information from the original data.

These differences lead to until now unresolved problems and explain why we cannot use document recognition software to process video data, at least not without sufficient pre-processing. We will discuss the main problems and issues in the following paragraphs.

Low resolution One of the consequences of the need for small file sizes is the tremendous cut in resolution videos undergo before they are encoded. The actual resolution depends on the application and ranges from 160×100 pixels for internet based online videos to 720×480 pixels of MPEG 2 encoded DVD videos. Most of our experiments are based on frames of the format 384×288 pixels (equivalent to the CIF standard). This low resolution results in font sizes in pixels, which are

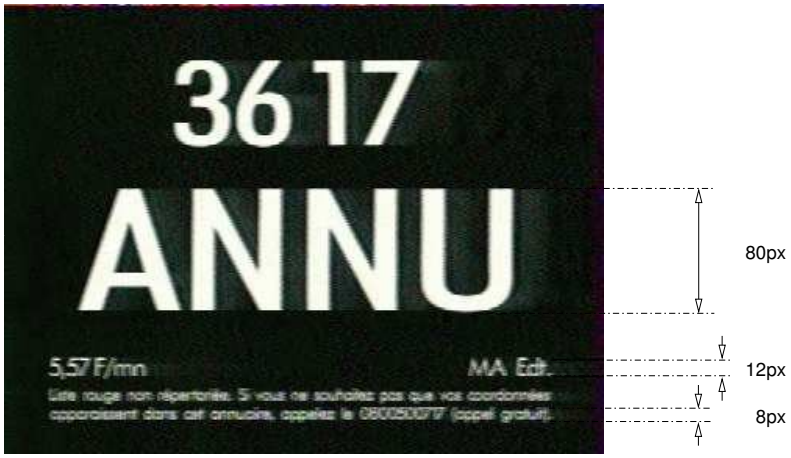


Figure 2.1: Small fonts and high differences in font size.

very much smaller than the font sizes of scanned document images. A character of a typical subtitle displayed with this resolution contains 10×10 to 14×14 pixels, whereas characters scanned for document OCR contain typically 50×50 pixels.

Furthermore the typical contents of video frames contains text with widely varying font sizes, as can be seen e.g in figure 2.1.The text in this frame is varying from 8 pixels to 80 pixels.

The difference of the font sizes poses problems during the detection stage, whereas especially the small fonts pose problems during the recognition phase of the systems. This first property is not actually text extraction dependent and also applies in general scene analysis, where objects can occur at any scale. The main response to this property is the multi-resolution or multi-scale framework.

Anti-aliasing and data compression artifacts Downsampling the video stream into a low resolution as described in the last paragraph creates more difficulties. The videos are mostly produced in higher resolutions, and downsampling only of the original data would create visible aliasing artifacts. To avoid this a low pass filter (e.g. a Gaussian) is applied before downsampling the data. This results in frames with sufficient perceived quality, on the other hand the data is smoothed. Figure 2.4 gives an example: Figure 2.4a shows a part of video frame with some screen text and Figure 2.4b shows the parts of the same text enlarged. One can see the smoothing of the low pass filter, which will disturb the segmentation of the characters.

Besides the low pass filtering, also the data compression techniques add artifacts to the final encoded image. The MPEG compression scheme loses information which is estimated to be redundant for the human vision system. However, the artifacts produced by the encoding scheme do harm the recognition process (e.g. by disturbing the color uniformity). It is well known, that standard compression

(a)

(b)

Figure 2.2: (a) an example of a text box with complex background; (b) an attempt to use fixed thresholding to segment the characters.

techniques like JPEG are not appropriate for synthetic object compression. Text is such an object.

Complex colored background Document images contain mostly two colors only: foreground and background, where both are uniform. Video frames on the other hand contain rich color information, and the background of a text box can be arbitrarily complex. The easy segmentation into foreground and background by thresholding, as it is done for scanned documents, is not possible. Figure 2.2a shows a text box with complex background and figure 2.2b the same box after an attempt to segment it with a fixed threshold. It is clear, that new segmentation techniques have to be found.

Artificial contrast enhancement Not only artificial vision is disturbed by the complex backgrounds in video frames, it also makes text less readable for humans. To increase the readability of text over complex background, the designers of screen text systems created a method to enhance the contrast of the text against it's background: artificial shadows.

An example can be seen in figure 2.3a which displays a sports result. Under and beside each white stroke of the text characters a black shadow has been created, which increases the contrast of the character and highlights it from it's background. However, since the color of the character is not uniform this makes segmentation more difficult. Figure 2.3b shows the same text after a manual fixed threshold.

(a) (b)

Figure 2.3: (a) an example text box where artificial contrast has been added to increase readability; (b) the same text box after optimal manual thresholding.

2.3 What is text?

We are convinced that a perfect text detection algorithm without text recognition is impossible. The only mechanism which is able to discriminate sufficiently between many different kinds of text on one side and non-text on the other side, is an algorithm which is based on the recognition of the characters. Unfortunately, the current state of the art of computer vision systems requires text to be detected before it can be recognized, therefore all current detection systems are pre-attentive systems which employ low level features to discriminate between text and non-text.

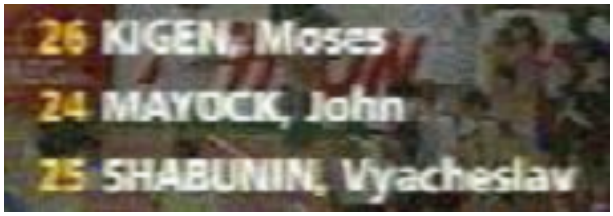
Spatial grouping

This leads us to the question, what is text? What kind of properties does it have and how can we exploit them for detection? A feature which comes to mind quickly is that text is composed of letters if it is written in a western script, or of symbols if it is written in an Asian script. These letters or symbols can be segmented. However, practically this can be very problematic if the text is of poor quality as e.g. in videos. Figure 2.4 shows a part of an MPEG 1 video frame. The original frame has a resolution of 384×288 pixels, the text characters have an average size of 10×12 pixels. A zoom into the digits "26" quickly shows the bad quality of the image[3]. The anti-aliasing effects due to the low resolution lead to color bleeding and smoothing between the characters and the background. Segmentation of the characters at this stage would be very difficult if not impossible.

Texture

A text property which is easier to exploit in noisy and low resolution documents is texture. Text characters and symbols are composed of strokes, which are in the case of horizontal western text for the most part vertical. Text forms pseudo-regular patterns, or texture, and can therefore be identified using standard texture classification methods as e.g. second order statistics or Gabor filtering.

[3]A surprising visual peculiarity of this example, which has been confirmed by many independent observers, is the fact the character which is very well recognizable as "6" in the original image, resembles a "5" when we zoom in.



(a)

(b)

Figure 2.4: (a) an example of a part of a video frame; (b) a zoom into the string "26".

Texture classification is a domain which has already reached a certain degree of maturity with a large quantity of published work. We can therefore ask ourselves, whether it makes sense to exploit its results for text detection. As an example, figure 2.5 shows an image including some scene text ("qantas") in a scene with complex background structures (figure 2.5a) and the normalized Gabor energy features [36] after application of a horizontally oriented Gabor filter (figure 2.5c). The wave length of the sinusoidal wave of the chosen filter roughly corresponds to the inter-stroke spacing of the text in the image. As we can see, the filter does respond to the text, but also to a big part of the background structure.

A bank of Gabor filters is a texture analysis tool, and as such it had been designed to *classify* textures, not to *detect* them. A hypothesis we need to take before we perform texture analysis, is that there is texture. What is texture, and what is the difference between the text and the background texture in image figure 2.5a? Among other things, the difference lies in the spatial extension: text contains several strokes, not just one[4]. It is very difficult with a linear filter, like a Gabor filter, to detect structures with several strokes as opposed to single lines. Gabor filters are known to minimize the product of spatial and frequency localization, however, they cannot break its theoretical limit.

[4]We do not treat short text as e.g. "I". As we already stated before, it is impossible to distinguish text from non-text without recognition.

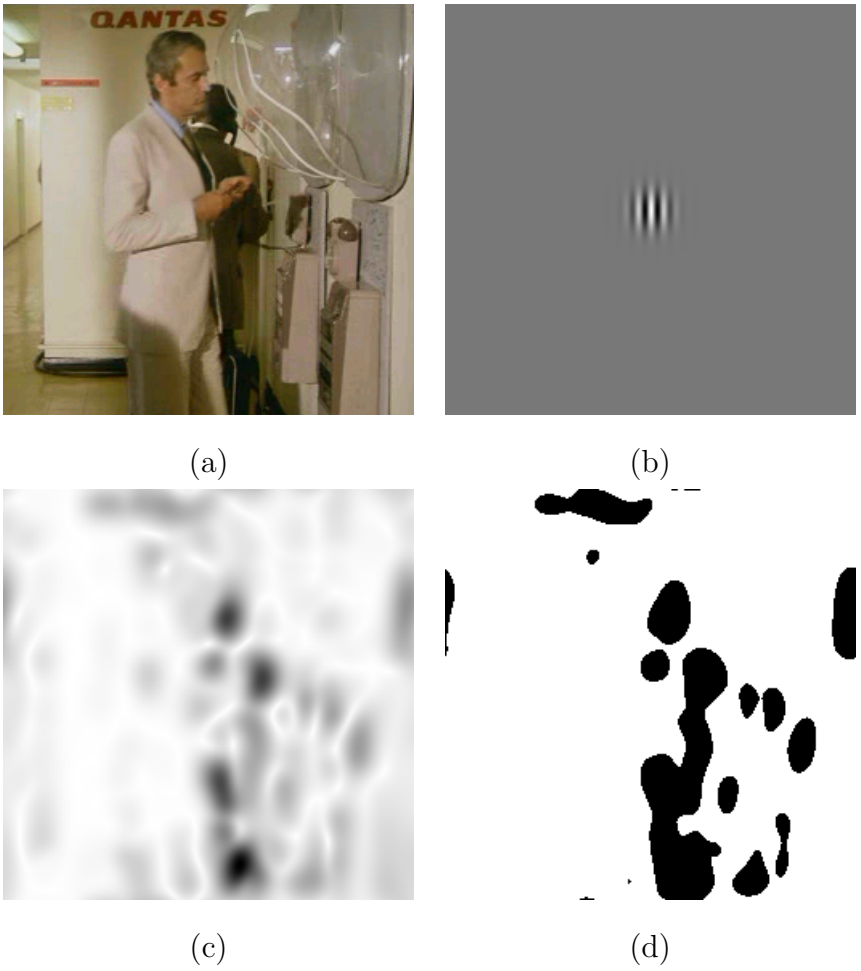


Figure 2.5: Texture analysis: (a) an example image; (b) the impulse response of a horizontally oriented Gabor filter; (c) normalized Gabor energy features after the application of a horizontal Gabor filter (d) the thresholded features.

Typical Gabor filters employed in texture analysis have a half-peak frequency bandwidth of 1 octave [6], which makes both the estimation of the frequency and its localization efficient. However, the spatial support of the filter is not big enough to emphasize large structures. The filter responds equally well to several strokes of medium contrast (amplitude) or to a single bar of high contrast. Non-linear filters, as the grating cell filter [36] could tackle this problem more efficiently. However, this filter introduces geometrical properties into the response, which takes us away from the pure textural features.

Sometimes it is assumed that text contains strokes with regular spacing. Unfortunately, strokes are spaced in pseudo-regular structure only, since the characters themselves are very different from each other, sometimes resulting in big differences in the

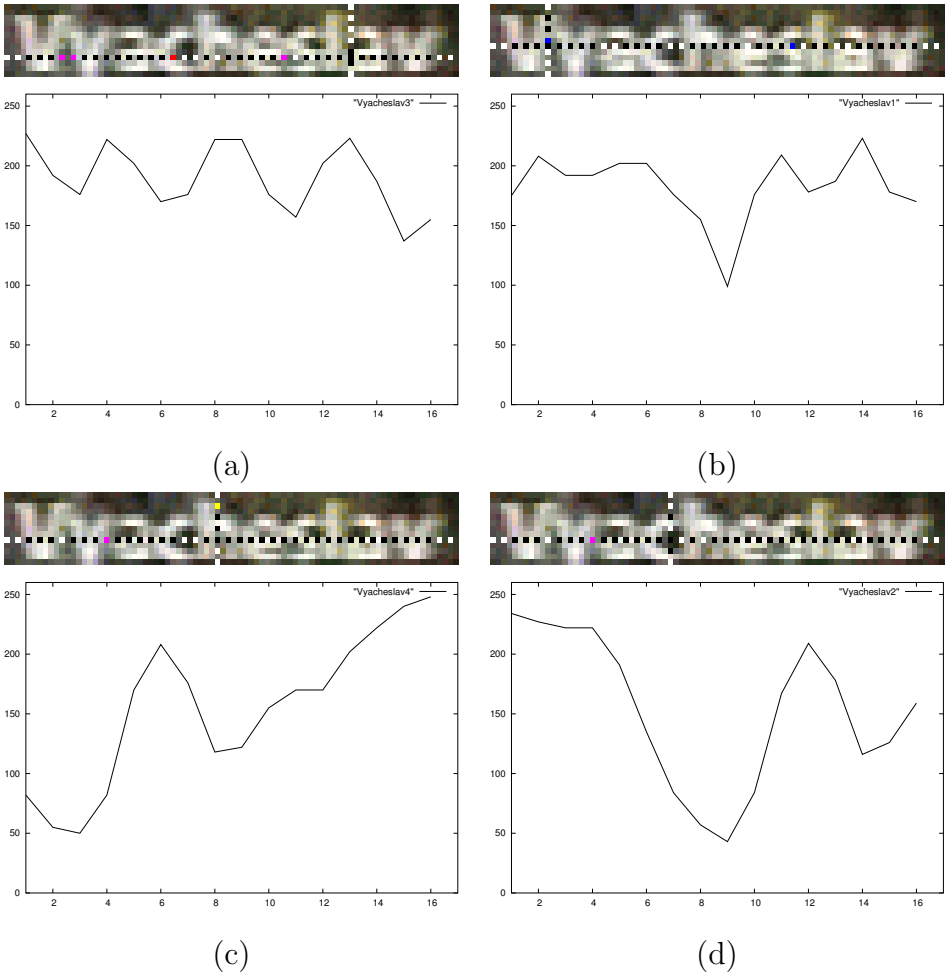
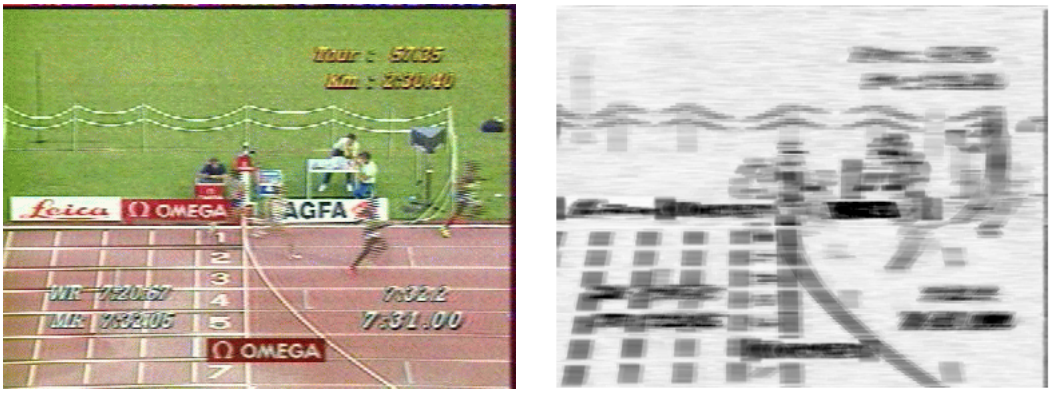


Figure 2.6: Horizontal intervals of gray values at 4 different positions taken from the word "Vyacheslav" in figure 2.4a.

inter-stroke spacing. These differences are negligible if the text is long enough, but can be very important when short text needs to be considered as well. Most detection methods are designed to detect text consisting of 2-3 or more characters. As an example, clearly the inter-stroke spacing of the strings CLJ and MWM is different, resulting from the different numbers and positions of the vertical strokes in the characters C, L, J, M and W. As another example, figure 2.6 shows the gray value profiles of four horizontal neighborhoods (±8 pixels) around four positions in the text string Vyacheslav taken from the image shown in figure 2.4a. The profile shown in figure 2.6a spans across the area taken by the characters sla, therefore 4 regular and sharp peaks are visible in the profile. Figure 2.6b shows only two very broad peaks, since the corresponding interval is located at the bottom of the characters Vy.

The shown examples are taken from a single word written in a single font. Irregu-



(a) (b)

Figure 2.7: (a) an example image; (b) the result of a gradient density filter.

larities in the texture are more evident with different fonts, styles, slants, orientations of the text etc. Because of this irregularities, low order statistics (first order) are more useful then higher order statistics as a discriminative feature when we want to be able to detect shorter pieces of text.

Contrast and geometry

The decision whether a pixel is part of text or not cannot be taken locally, because texture alone is not sufficient as a discriminative feature. Rather, a wider context has to be verified, as e.g. the geometric properties of the text area to be detected. Text strings, i.e. words or sentences, form rectangles (or quadrilaterals in the general case of scene text which is subject to perspective distortion).

Figure 2.7 shows an example image and a filter which replaces each image pixel by the mean of the Sobel magnitude in a horizontal neighborhood. The text in the example image is visible as rectangular areas in the filtered image and it is possible to guess what parts of the filtered image contain text on the basis of the shape of these areas.

Color

When a computer vision system is designed, one must decide whether the system needs to process color images or whether grayscale processing is enough. Most computer vision architectures and algorithms are concentrated on grayscale processing since color adds additional information in specific situations only. After all, color blind people cope very well with their limitation in the real world. Several reasons suggest the use of gray value processing, if the situation does not explicitly require color processing. Of course there is an obvious gain in computational complexity if we can restrict an algorithm to gray value processing. Moreover, mathematical modeling is easier in the 1D grayscale space than the 3D color space or even multi-dimensional multi-spectral spaces. Finally, the

(a) (b) (c)

Figure 2.8: Color vs. grayscale processing: (a) an example image; (b) the Sobel edges; (c) the Sobel edges in LUV color space using the Euclidean distance.

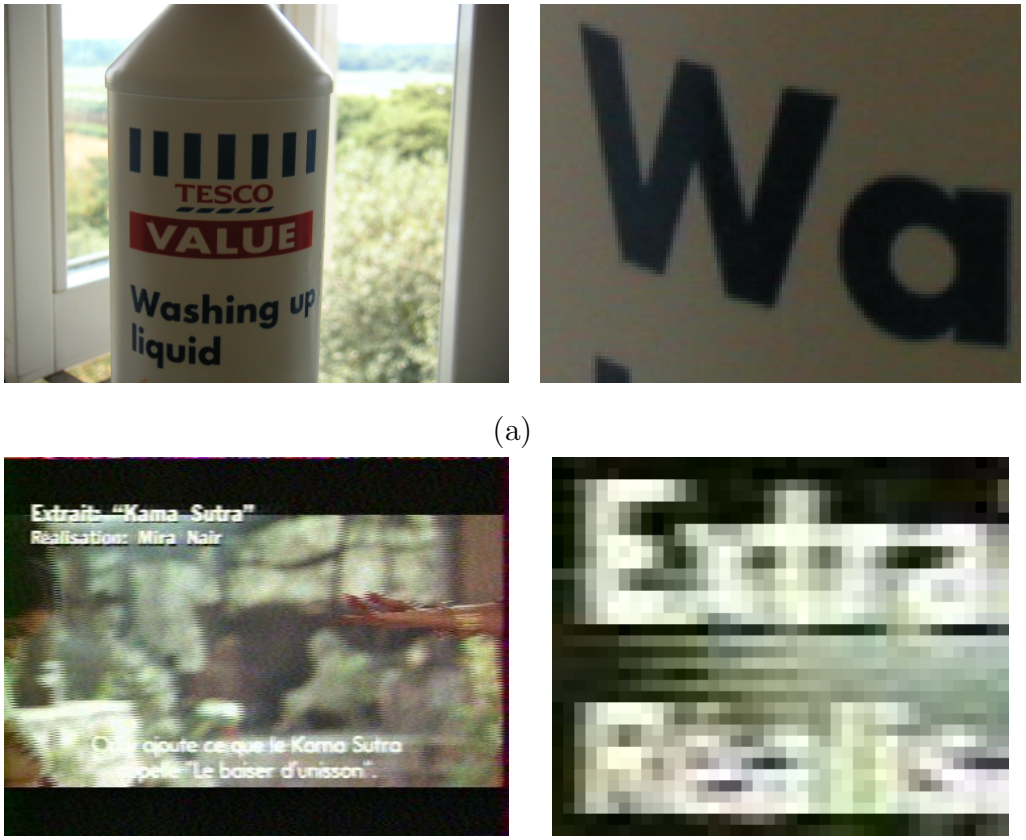
decision to process color opens a box of worms: what kind of color system is the right one? How does the equipment (sensors, converters etc.) treat the different color bands, etc.

One might think that for text detection, color is irrelevant since text needs to be read on black and white TVs (which actually should be called grayscale TVs). However, this is true for artificial text only. Scene text detection may very well benefit from color processing, as figure 2.8 suggests. The example image contains some scene text with disturbing reflections. Figure 2.8b shows the Sobel edges calculated on the grayscale image. We remark, that there are no strong gradients of the luminance information on the character borders of the word "La Téléphonie". The Sobel edge filter can also be adapted for color images by applying the vertical part of the linear filter to all three color bands and replacing the horizontal filter — which is basically a distance between two pixels — by the Euclidean distance in the 3D color space. The results of this operation applied on the example image are shown in figure 2.8c. We notice, that the edges of the characters show strong color gradients.

However, this result can not be generalized to all types of text or even all types of scene text. In fact, our experience showed us, that for most text color processing does not add additional information. However, we should keep in mind that it may help in rare cases.

What kind of text?

Although text presents itself in various styles and orientations and moving text is encountered frequently, for indexing purposes it is often useful to restrict the detection process to a narrower class of text in order to increase precision. In fact, the most important type of text for the indexing of news casts and television journals is horizontal



(a)

(b)

Figure 2.9: Example images and zooms into the text area.

artificial text[5]. Movements are mostly restricted to vertical movement of large quantities of text (the movie casting type) or a single line of text which scrolls from right to left in the lower part of the screen. Both types of movement are special cases which are easier to detect then the general case.

A text detection algorithm needs to be adapted to the type of target document. Figure 2.9 shows two different images (left) and a zoom into a part of the text (right). Figure 2.9a has been taken from the test data used for the text localization competition in the framework of the International Conference for Document Analysis and Recognition. The image has been shot with a digital camera and has a very high resolution of 1280×960 pixels. The displayed character "W" is of size 70×60 pixels. The detection algorithm needs to be different for this kind of image then for the one shown in 2.9b, which has been taken from a French TV channel and is in format CIF (384×288 pixels) with average character sizes of 10×10 pixels. A system optimized for text shown in

[5] Even the majority of scene text is horizontal or close to horizontal with a tolerance of $\pm 20^o$, although considerable perspective distortions occur frequently

figure 2.9a might use character segmentation and regrouping for detection, for instance. However this system would perform badly on images as the one in figure 2.9b.

2.4 State of the art of text detection

Extraction of text from images and videos is a very young research subject, which nevertheless attracts a large number of researchers. The whole extraction process involves different techniques, starting with the detection, tracking, enhancement, binarization to recognition. In order to increase the readability of this thesis, we delegated the description of the previous work in this domain to the respective chapters. Therefore, in this section we concentrate ourselves on a description of the state of the art of detection in still images. The previous work concerning text detection in video sequences is explained in chapter 5. The state of the art of enhancement and character segmentation is given in chapter 6.

The vast number of existing work on text detection can be classified according different criteria. The cited methods are classified according to the type of algorithms they employ (see figure 2.10). However, a historical point of view is taken into account.

Detection through segmentation and spatial grouping

The first text detection algorithms, introduced by the document processing community for the extraction of text from colored journal images and web pages, segment characters before grouping them to words and lines. Jain et al. [29] perform a color space reduction followed by color segmentation and spatial regrouping to detect text. Although processing of touching characters is considered by the authors, the segmentation phase presents major problems in the case of low quality documents, especially video sequences. A similar approach, which gives impressive results on text with large fonts, has been presented by Lienhart [44]. A segmentation algorithm and regrouping algorithm are combined with a filter detecting high local contrast, which results in a method which is more adapted to text of low quality. Still, the author cannot demonstrate the reliability of his algorithm in the case of small text. False alarms are removed by texture analysis, and tracking is performed on character level, which might pose considerable problems in the case of text as presented in figure 2.9b. Similar methods working on color clustering or thresholding followed by a regrouping of components have been presented by Lee and Kankanhalli [39], by Zhou and Lopresti [92] and by Sobottka, Bunke et al. [69]. Hase et al. cluster the components and allow for spatial arrangements which follow a quadratic function [25].

Scanline processing

Some methods are scanline based, i.e. they proceed line by line during the classification phase. Mariano and Kasturi perform a color clustering of the pixels of each scan line in order to find the pixels of the text cluster [48]. Histograms of line segments of uniform

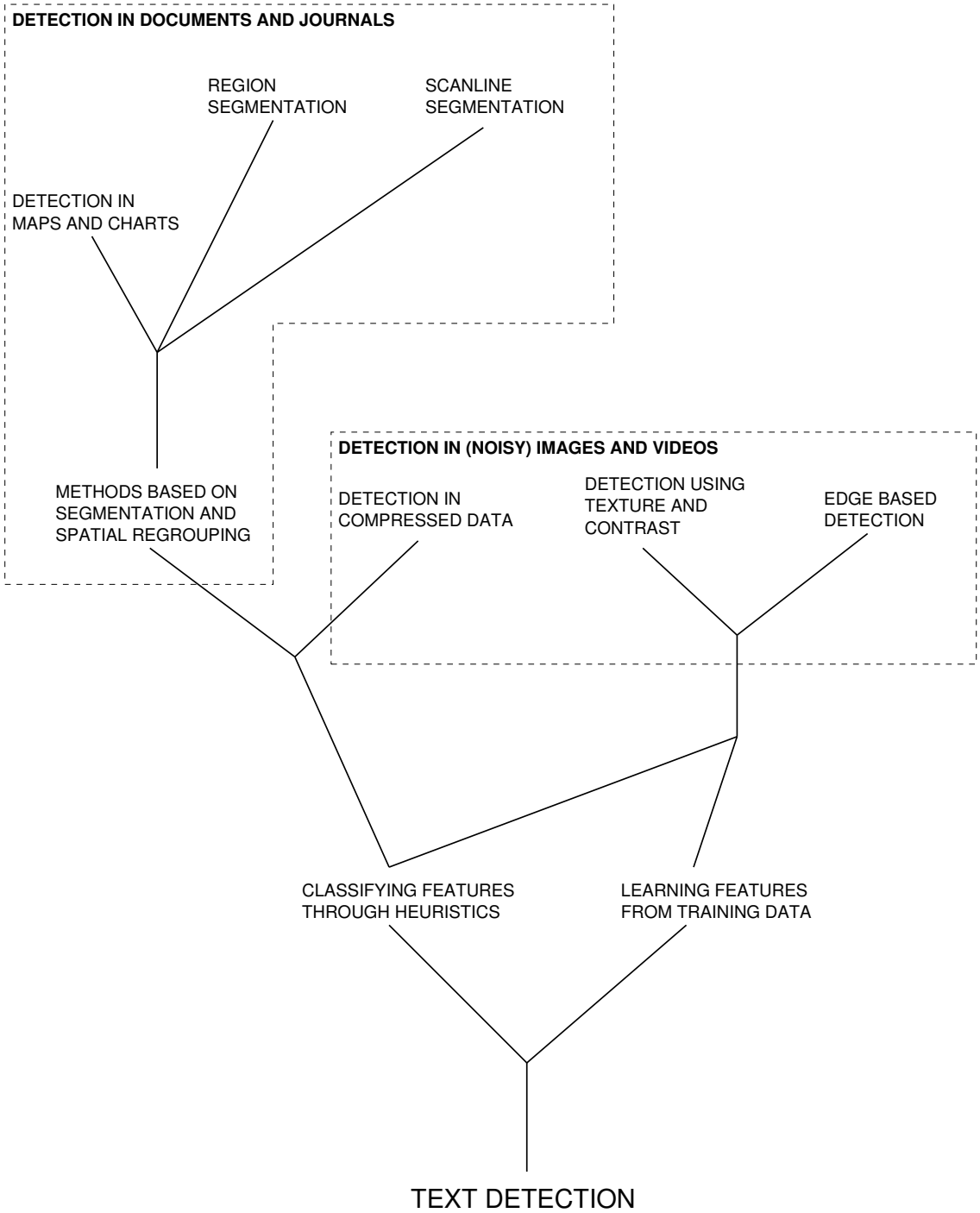


Figure 2.10: A classification of previous work in text detection.

color are computed and compared across lines to form rectangles. Wong and Chen calculate gradient measures for each line and cut the lines to segments of similar gray value [86]. Adjacent scanlines are merged using a statistical similarity criterion.

Detection in maps and charts

Methods for text extraction from very graphical documents, e.g. maps, followed similar patterns as the ones developed by the document processing community. Tan et al. use a hierarchical processing of the connected components in the image to find text as regrouped components [70]. Brès and Eglin [7] binarize the map and glide a rectangular map across the image. Inside each window, measures as the number of vertical segments, spacing, regularity etc. are calculated and used for the decision whether a pixel contains text or not.

The methods based on segmentation work fine for high resolution images as newspapers and journals but fail in the case of low resolution video, where characters are touching and the font size is very small. New methods developed by the image and video processing community based on edge detection or texture analysis were soon introduced when the attention focused to video.

Edge based detection

The video indexing system introduced by Sato, Kanade et al. [63] combines closed caption extraction with super-imposed caption (artificial text) extraction. The text extraction algorithm is based on the fact that text consists of strokes with high contrast. It searches for vertical edges which are grouped into rectangles. The authors recognized the necessity to improve the quality of the text before passing an OCR step. Consequently, they perform an interpolation of the detected text rectangles before integrating multiple frames into a single enhanced image by taking the minimum/maximum value for each pixel. They also introduced an OCR step based on a correlation measure. A similar method using edge detection and edge clustering has been proposed by Agnihotri and Dimitrova [1]. Wu, Manmatha and Riseman [88] combine the search for vertical edges with a texture filter to detect text. Unfortunately, these binary edge clustering techniques are sensitive to the binarization step of the edge detectors. A similar approach has been developed by Myers et al. [54]. However, the authors concentrate on the correction of perspective distortions of scene text after the detection. Therefore, the text must be large so that baselines and vanishing points can be found. Since the detection of these features is not always possible, assumptions on the imaging geometry need to be made.

LeBourgeois [38] moves the binarization step after the clustering by calculating a measure of accumulated gradients instead of edges. The coarse detection step of our work is based on a slightly modified variant of this filter, but our detection technique also uses higher level features based on a robust estimation of the geometry of the coarsely detected features. In his work, LeBourgeois proposes an OCR algorithm which uses

statistics on the projections of the gray values to recognize the characters.

A couple of methods use mathematical morphology in the detection step. Hori dilates Sobel edges into text regions [27]. The emphasis of his work is set on binarization and removal of complex backgrounds. Hasan and Karam dilate edges detected with a morphological detector [24]. The main drawback of their algorithm are the strong assumptions on the geometry and the size of the text.

In the previously introduced methods, the features calculated for the discrimination between text and non-text pixels are of very basic nature. This is due to the fact, that in a neighborhood of small size, text does not have a very distinctive signature. Most people use edge density and related features, as high frequency components of wavelet decompositions etc. Very sophisticated texture features are of limited use since the texture of text is very irregular, especially in case of short text. Sin et al. detect the text using features calculated on the autocorrelation function of a scanline [67]. However, they exploit the fact, that text in their application (billboard detection) is very long and enclosed by a rectangular frame (e.g. the panel of the billboard). Furthermore, several scanlines of a text rectangle are concatenated, which creates problems due to the non-alignment of the phases of the Fourier-transforms of the different scanlines. The method cannot be applied to short text.

Detection through learning methods

Various methods based on learning have also been presented. Li and Doermann use a Haar wavelet for feature extraction [42]. By gliding a fixed size window across the image, they feed the wavelet coefficients to a MLP type neural network in order to classify each pixel as "text" or "non-text". Clark and Mirmehdi also leave the classification to a neural network fed with various features, as the histogram variance, edge density etc. [14]. Similarly, Wernike and Lienhart extract overall and directional edge strength as features and use them as input for a neural network classifier [80]. In a more recent paper, the same authors change the features to a 20×10 map of color edges, which is fed directly to the neural network [45]. Jung directly uses the gray values of the pixels as input for a neural network to classify regions whether they contain text or not [31]. Kim et al. also use the gray values only as features, but feed them to a support vector machine for classification [33]. Tang et al. use unsupervised learning to detect text [71]. However, their features are calculated from the differences between adjacent frames of the same shot, resulting in a detection of appearing and disappearing text. Text which is present from the beginning of a shot to the end (a type of text frequently encountered for locations in news casts) is missed. Chen et al. [11] introduce a two step process of fast candidate text line detection by edge detection and morphological processing and text rectangle identification by SVM learning. The features fed to the SVM are an edge distance map of the scaled text box. The drawback of the method is the simple coarse detection step. Text on complex background is likely to touch the background, thus cannot be segmented and therefore not be verified by the more complex identification step.

As usual, learning methods depend on the quality of the training data used to train the systems and on the features which are fed into the learning machine. However, in video, text appears in various sizes, fonts, styles etc., which makes it very difficult to train a generalizing system. The features which have been presented in the previous work are very basic (gray values, normalized gray values, distances to the edge map, etc.), it must be feared that they do not generalize well across the set of possible text types.

Detection in compressed data

Methods working directly in the compressed domain have also been introduced. Zhong, Zhang and Jain extract features from the DCT coefficients of MPEG compressed video streams [91]. Detection is based on the search of horizontal intensity variations followed by a morphological clean up step. Randall and Kasturi [15] compute horizontal and vertical texture energy from the DCT coefficients. Unfortunately they only obtain good results for large text. Gu [22] also uses the DCT coefficients to detect static non-moving text and removes false alarms by checking the motion vector information. One of the problems of these methods is the large number of existing video formats (MPEG 1&2, MPEG 4, Real Video, wavelet based methods etc.) and the high rate of innovation in the video coding domain, which makes methods working on compressed data quickly obsolete.

Conclusion

Although the research domain is very young, there exist already a high number of contributions. Only few of them present complete methods beginning from the detection of the text until the last binarization step before passing the results to an OCR. In most cases the model behind the detection algorithms is very simple: edge detection, regrouping into rectangles.

2.5 Our text extraction method

One of the major design goals of our method was the ability to extract text from still images as well as video sequences. A video sequence is a 3 dimensional object with spatial dimensions x and y and the time dimension t. The additional time dimension creates questions concerning the detection process itself as well as the representation and the usage of the detected text "appearances". We call text appearance an entity of text perceived as such by the viewer, which appears on the screen and disappears after some time. Appearances are 3D sub-objects of the input sequence, which cover a part of the the spatial area and which have a defined temporal range. The detection process itself, i.e. the creation of the text appearances, may follow different principles:

- Spatio-temporal approaches treat the 3D input stream as such and detect the text directly in the 3D space. The text appearance is a direct result of the detection process.
- Tracking methods treat the 3D input stream as a temporal sequence of 2D images and detect text on a frame by frame basis. The detected text objects are tracked across several frames in order to create the text appearance, which is a result of the detection *and* the tracking process.

We designed our method for the later use in semantic video indexing systems. According to our partner INA, these systems are mostly applied to news casts and television journals, where most text is either static or follows very simple movements (the horizontal movement of a single line or the vertical movement of the "movie casting type"). For this reason, and because of the fact that we also want to be able to detect text in still images, we decided to use the tracking approach, detecting text on a frame by frame basis. In this work, we concentrated on static, non moving text. A generalization of our work to simple, linear movement has been done by Marquis [49] and will be described shortly in section 5.4.

Nevertheless, a spatio-temporal detection process should be feasible and could be investigated in future work. This type of algorithm falls into the class of "spatio-temporal segmentation" methods, which are currently under investigation in our research team [51].

Recognition

Apart from the detection itself, the additional temporal component in video sequences also leads to consequences and open questions concerning the usage of the detected appearance. The detected text forms a 3D object, which cannot be recognized by an OCR software directly, at least not by existing OCR technology. Several possible techniques may be considered:

- Choose a single frame out of all possible frames of the appearance. This is a simple technique, which has the disadvantage of losing the possible additional temporal information in the appearance.
- Use the 3D text object as it is and develop a new recognition algorithm which exploits the temporal information directly in order to increase recognition performance.
- Integrate the 3D appearance into a single 2D image, exploiting the temporal information in order to "clean up" the image and to create a single image of better quality.
- Apply the recognition algorithm to *each* of the frames in order to create a set of recognized text strings. Apply symbolic statistics to the set in order to create a single string with the most probable contents.

The first solution — choosing the text rectangle from a single frame for recognition — fails due do the miserable quality of most text taken from videos. At least some processing is necessary to enhance the image quality.

The second solution is beyond the scope of this thesis. The development of a new OCR technology needs a tremendous amount of engineering experience in order to find the necessary heuristics which allow these systems to obtain their excellent recognition performance. Instead, we apply commercial software, which delivers excellent results on scanned printed or faxed documents.

Unfortunately, commercial OCR software is not adapted to the type of data (see section 2.2), which is why we chose the third solution: We integrate the text appearance into a single image of better quality, which is better closer to the type of data expected by commercial OCR software.

Our research team also successfully worked on the fourth way to use the text appearance. The algorithm which uses new theoretical research on statistical processing of character strings done by our team, is described shortly in section 5.5.

A global scheme of our proposed system for text extraction from video sequences is presented in figure 2.11. As already stated, the detection algorithm for still images is applied to each frame of the sequence separately. The detected text rectangles are passed to a tracking step, which finds corresponding rectangles of the same text appearance in different frames. From several frames of an appearance, a single enhanced image is generated and binarized, i.e. segmented into characters and background, before passing it to a standard commercial OCR software. The respective steps are given in the next chapters.

Text in videos has gray level properties (e.g. high contrast in given directions), morphological properties (spatial distribution, shape), geometrical properties (length, ratio height/length etc.) and temporal properties (stability). Our method makes use of these properties, starting from the signal and going sequentially to the more domain dependent properties. The final step (the character segmentation presented in chapter 6) results in a set of binary boxes containing text which need to be recognized by a classical commercial OCR system.

Detection in still images

The heart of the extraction system is the detection algorithm for still images. In the next two chapters, we propose two different algorithms with two different philosophies. The common issue is the problem of finding a decision function for each pixel deciding whether it is part of text or not. The pre-attentive nature of the state of the art of computer vision tends to produce algorithms whose results are not binary, but continuous. This is a very common problem in computer vision, which is for example also encountered during edge detection algorithms. Gradient based edge detection algorithms produce continuous outputs with smaller responses to noise and light background texture and larger responses for edges, which (hopefully) correspond to object boundaries. The problem of automatically determining a threshold which separates these two cases

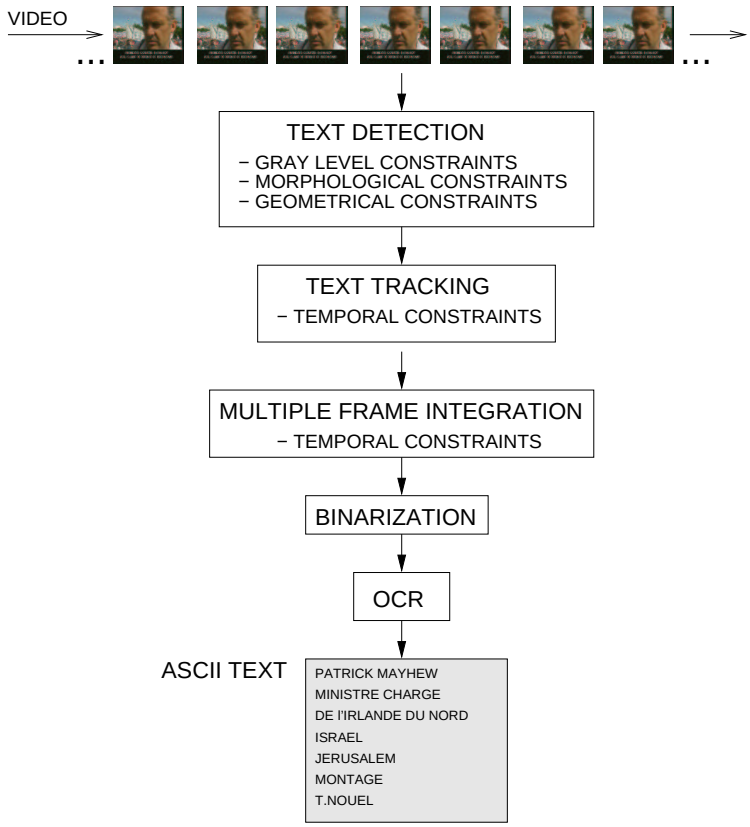


Figure 2.11: The scheme of our system.

is far from being resolved.

The same is true in the case of text detection, the decision function needs to be thresholded. The problem of automatically finding the threshold can be tackled in different ways. We propose two different solutions (See figure 2.12):

- We assume that there is text present in the image and search the threshold by statistical modeling of the two distributions (*text* and *non-text*).
- Thresholds and/or text features are learned from training data.

The first method is described in detail in the next chapter. Figure 2.13 illustrates the principle of this approach. Figure 2.13b shows an image whose gray values correspond to a "probability" of text. Assuming that there is text present in the image, we can threshold the data with a threshold which in some sense optimally divides the two populations, which results in a binary image (see figure 2.13c). An additional post processing step can further clean up noise and impose geometrical constraints on the detected text areas. On the other hand, if the hypothesis is not fulfilled, i.e. if there is no text present in the image, then the threshold will separate two non-text populations,

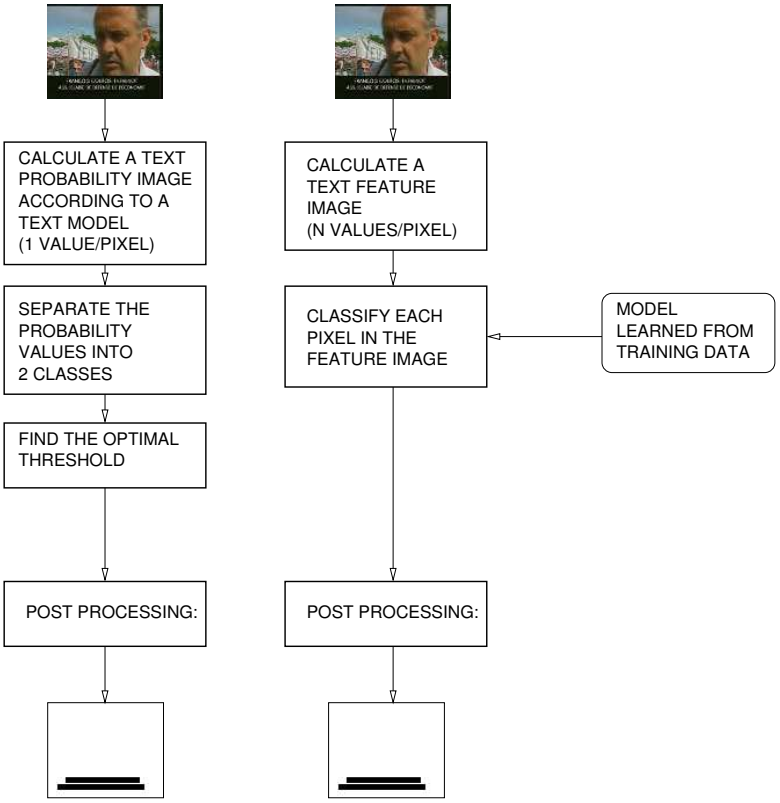


Figure 2.12: Two different detection principles.

effectively creating false alarms (see figure 2.13f). The success of the method depends on the ability of the post processing step to remove these falsely detected text rectangles.

A second possible approach, which will be pursued in chapter 4, is to learn text features from training data. This increases the detection precision by improving the performance in images which do not contain text, with the tradeoff of a dependence on the training data. Text appears in a wide range of styles and sizes, a complete coverage of the training data is very difficult.

An important advantage of a learning solution is the ability to increase the complexity of the text model. State of the art learning machines as e.g. Support Vector Machines (SVM) are very powerful and give optimal classification performance even in the case of high dimensional feature spaces. This allowed us to include geometrical features directly into the detection phase, as opposed to the post processing phase in the first algorithm.

The existing text detection algorithms presented in section 2.4 enforce geometrical constraints in a post processing step after the detection step, mostly using mathematical morphology. The disadvantage of this procedure is evident: a bad detection cannot be corrected by sophisticated post processing. However, the integration of geometrical features into the detection phase is not straightforward. We adopted a two step

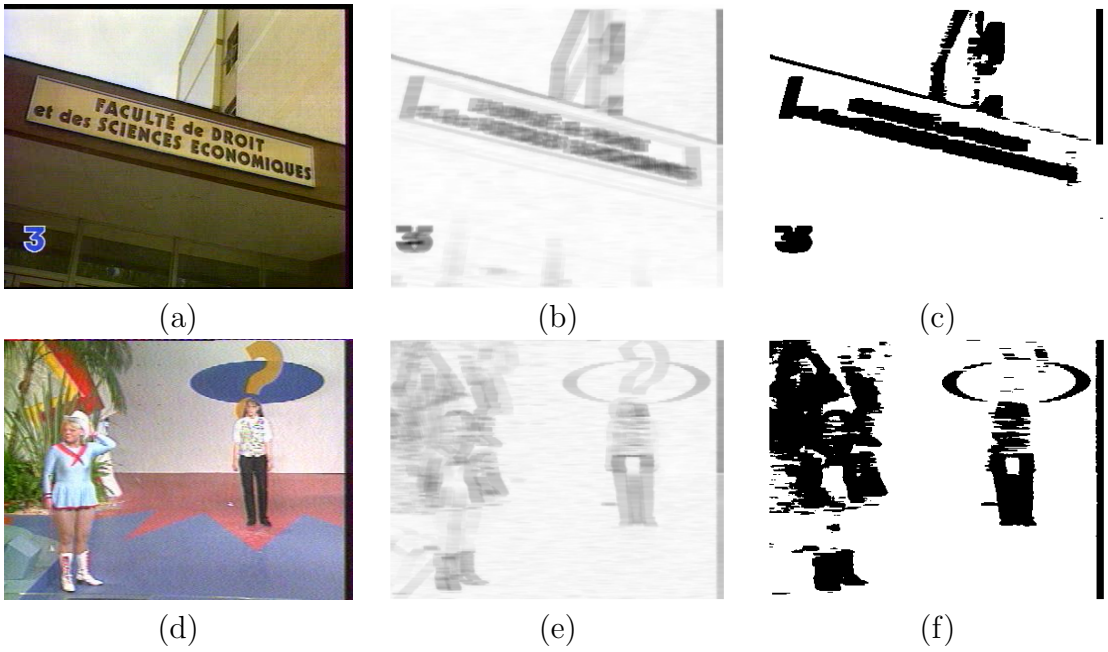


Figure 2.13: From left to right: the original image, a gradient density measure image, the thresholded image.

approach:

- Perform a coarse detection without taking into account geometrical features.
- For each pixel, calculate geometrical features of its neighborhood based on the detection results from step 1. Use these features together with the features calculated in step 1 and perform a new refined detection.

More details of this algorithm are presented in chapter 4.

As said above, the distinction of our work to other methods based on SVM learning lies in the choice of features. The approaches [31] and [11] already cited in section 2.4 feed very simple features into the SVM, namely directly the gray values in a local window around the pixel or an edge distance map. In these works, the scientists delegated the difficult task of feature design to the learning machine. It is well known, that implicit feature extraction does not give the same results as wisely done manual feature design — finally, only the scientist knows what he or she needs to detect, as opposed to the learning machine.

Chapter 3

Detection in still images - local contrast

Wenn ihr's nicht fühlt, ihr werdet's nicht erjagen,
wenn es nicht aus der Seele dringt,
und mit urkräftigem Behagen
die Herzen aller Hörer zwingt.

You can't accomplish it, if you don't feel it,
unless it comes straight from your soul
and with the force of deep contentment
compels the hearts of all who hear.

Johann Wolfgang von Goethe
German poet (1749 - 1832)

The text detection method presented in this chapter has been designed to detect horizontal artificial text. We exploit various properties of this kind of text, which can be translated into gray level constraints, morphological constraints and constraints on the geometry of the detected text areas. The constraints are enforced in this order during the detection. Figure 3.1 shows the intermediate results after imposing these respective constraints. The details of these transformations will be explained in the following sections.

3.1 Gray level constraints

Artificial text has been designed to be read easily. Therefore, the contrast of the characters against the background can be assumed to be high. On the other hand, we cannot assume uniform background, arbitrary complex background needs to be supported. The second assumption concerns the color properties of the text. Since the text is designed

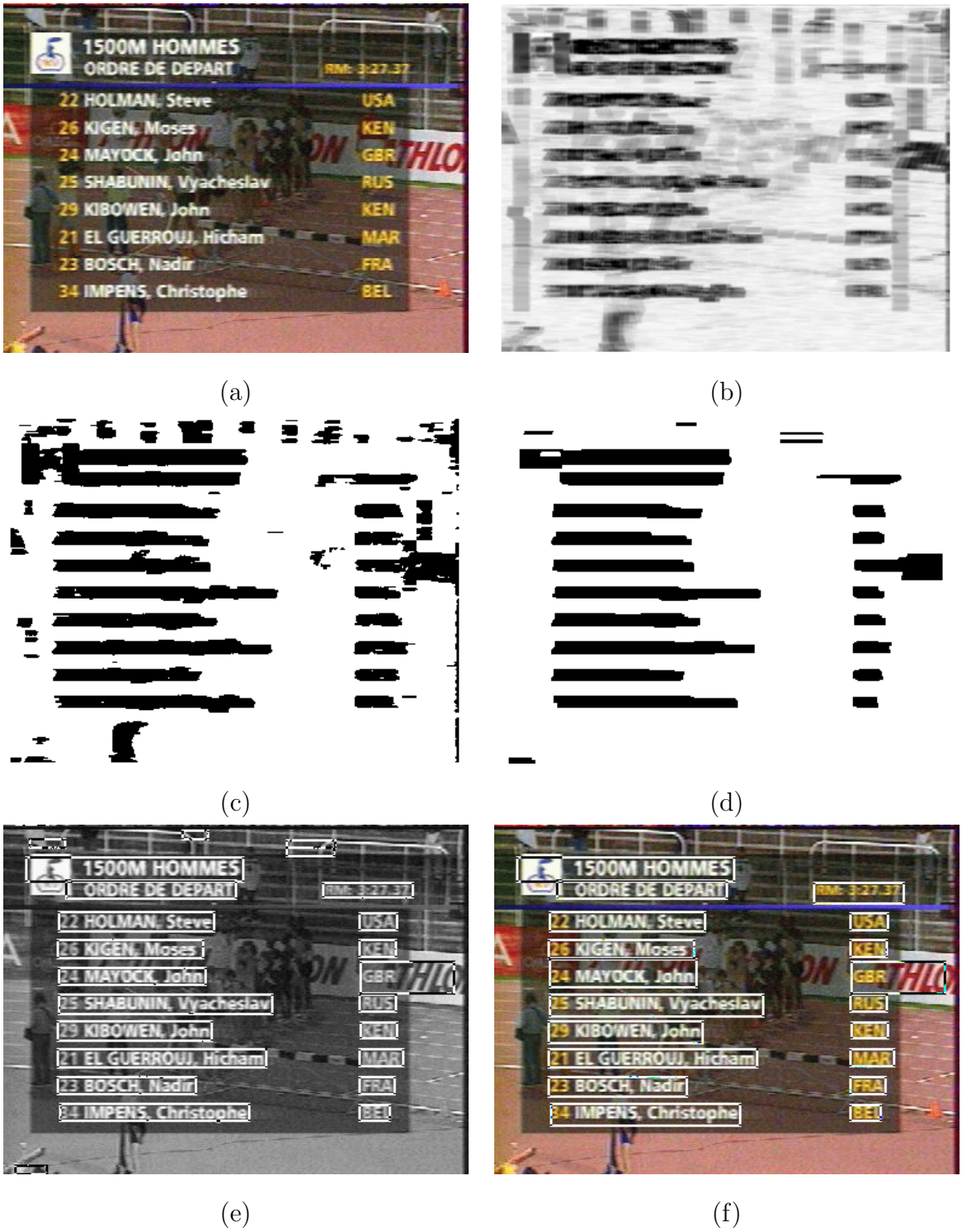


Figure 3.1: The intermediate results during the detection process: (a) the input image; (b) the gray value constraints (the accumulated gradients); (c) the binarized image; (d) the image after morphological post processing; (e) the rectangles superimposed on the input image; (f) the final result after imposing geometrical constraints.

to be read easily, it can be extracted using the luminance information of the video signal only (else news cast would not be readable on black and white TVs). We therefore convert all frames into gray scale images before we start processing.

Our detection method is based on the fact that text characters form a regular texture containing vertical strokes which are aligned horizontally. We slightly modified the algorithm of LeBourgeois [38], which detects the text with a measure of accumulated gradients:

$$\boldsymbol{A}(x,y) = \left[\sum_{i=-\lfloor S/2 \rfloor}^{\lfloor S/2 \rfloor} \left(\frac{\partial \boldsymbol{I}}{\partial x}(x+i,y) \right)^2 \right]^{\frac{1}{2}} \tag{3.1}$$

The parameters of this filter are the implementation of the partial derivative and the size S of the accumulation window. We chose the horizontal version of the Sobel operator as gradient measure, which obtained the best results in our experiments. The size of the accumulation window depends on the size of the characters and the minimum length of words to detect. Since the results are not very sensitive to this parameter, we set it to a fixed value. The filter response is an image containing for each pixel a measure of the probability to be part of text.

Binarization of the accumulated gradients is done with a two-threshold version of Otsu's global thresholding algorithm [59]. Otsu calculates an optimal threshold from the gray value histogram by assuming two distributions in the image (class 0 for "non-text" and class 1 for "text" in our case) and maximizing a criterion used in discriminant analysis, the inter-class variance:

$$t_{opt} = \arg\max_t (\omega_0 \, \omega_1 (\mu_1 - \mu_0)^2) \tag{3.2}$$

where $\omega_0 = \sum_{i=1}^{t} \frac{\boldsymbol{h}_i}{N}$ is the normalized mass of the first class, $\omega_1 = \sum_{i=t+1}^{L} \frac{\boldsymbol{h}_i}{N}$ is the normalized mass of the second class, μ_0 and μ_1 are the mean gray levels of the respective classes, $\boldsymbol{h}$ denotes the histogram, N the number of pixels and L the number of bins of the histogram.

In order to make the binarization decision more robust, we added a second threshold and changed the decision for each pixel as follows:

$$\begin{array}{lcl}
\boldsymbol{I}_{x,y} < k_l & \Rightarrow & \boldsymbol{B}_{x,y} = 0 \\
\boldsymbol{I}_{x,y} > k_h & \Rightarrow & \boldsymbol{B}_{x,y} = 255 \\
k_l \leq \boldsymbol{I}_{x,y} \leq k_h & \Rightarrow & \boldsymbol{B}_{x,y} = \begin{cases} 255 & \text{if there is a path}^1 \text{ to} \\ & \text{a pixel } \boldsymbol{I}_{u,v} > k_h \\ 0 & \text{else} \end{cases}
\end{array} \tag{3.3}$$

where k_h is the optimal threshold calculated by Otsu's method and k_l is calculated from k_h and the mean μ_0 of the non-text class of the histogram: $k_l = \mu_0 + \alpha(k_h - \mu_0)$, where α is a parameter. The result of this step is a binary image containing text pixels and background pixels.

[1] only made of pixels (x,y) such that $\boldsymbol{I}_{x,y} > k_l$.

3.2 Morphological constraints

A phase of mathematical morphology follows the binarization step for several reasons:

- to reduce the noise;
- to correct classification errors using information from the neighborhood of each pixel;
- to distinguish text from regions with texture similar to text based on some assumptions (e.g. minimal length);
- to connect loose characters in order to form complete words;
- to make the detection results less sensitive to the size of the accumulation window and to cases where the distances between characters are large;
- to remove false alarms produced in frames without text, i.e. where the initial hypothesis is not fulfilled.

The morphological operations consist of the following steps:

Step 1: Close (1 iteration). Structuring element:

$$\boldsymbol{B} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & \textcircled{1} & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.4}$$

This first preliminary step closes small holes in the components and connects components separated by small gaps.

Step 2: Suppression of small horizontal bridges between connected components.

This step of the morphological phase has been designed to deconnect connected components corresponding to text regions from connected components corresponding to non-text regions. It removes small bridges between connected components by suppressing pixels of columns whose local connected component's height is under a certain threshold. To do this, we first build a matrix $\boldsymbol{A}$ of the same size as the input image. Each element of the matrix corresponds to a pixel of the image and holds the local height of the connected component it belongs to. Finally, a thresholding step removes pixels (x, y) such that $\boldsymbol{A}_{x,y} < t_1$, where t_1 is a fixed parameter. Figure 3.2 shows an example image and the intermediate result images before and after the removal of the bridges between components.

A side effect of this operation may be the separation of a noisy text region into two or more connected components. Also, in cases where the distances between characters are high, text regions may be decomposed in several connected components. Steps 3

(a)

(b)

(c)

Figure 3.2: The removal of small bridges between components: (a) the original image; (b) before the removal; (c) after the removal.

and 4 of the morphological phase address this problem:

Step 3: Conditional dilation (16 iterations). Structuring element:

$$\boldsymbol{B}_C = \begin{bmatrix} \textcircled{1} & 1 \end{bmatrix} \tag{3.5}$$

The conditional dilation and erosion algorithms have been developed to connect loose characters in order to form words, we thus want to merge all connected components which are horizontally aligned and whose heights are similar. The dilation algorithm consists of a standard dilation algorithm using additional conditions on the shape of the connected components of the image. First, a connected components analysis for 4-connected neighborhoods is performed on the binary input image. We assume, that each component C_i corresponds to a character or a word. The component height is used in the following dilation step. To ease the implementation, we build a height matrix $\boldsymbol{H}$ of the same size as the binary input image. Every element of the matrix corresponds to a pixel of the input image and contains the height of the respective connected component. A second matrix $\boldsymbol{P}$ is constructed, which holds for each pixel the y coordinate of the corresponding connected component. Next, the non-zero values of the matrices are "smeared" to the left over the zero values, i.e. each zero value is replaced with its closest non-zero neighbor to the right of the same row:

$$\boldsymbol{H}_{x,y} = \begin{cases} \boldsymbol{H}_{x,y} & \text{if } \boldsymbol{H}_{x,y} \neq 0 \\ \boldsymbol{H}_{v,y} : \; v = \min\limits_{u>x}(\boldsymbol{H}_{u,y} > 0) & \text{else} \end{cases} \tag{3.6}$$

The same treatment is also applied to the matrix P.

The matrices $\boldsymbol{H}$ and $\boldsymbol{P}$ contain the information necessary to perform the conditional dilation. The algorithm traverses each line of the image from the left to right and each non-text pixel (x,y) preceded by a text pixel is set to text (i.e. dilation) under certain conditions. These conditions are based on the relative differences of height (respectively the y coordinate) of the component corresponding to the set pixel to the left of the current pixel and the height of the neighboring component to the right. These heights can be taken directly from the matrix $\boldsymbol{H}$. Since the test for dilation is performed on a non-set pixel, the matrix $\boldsymbol{H}$ contains the height of the next neighboring component to the right (see equation 3.6). This situation is illustrated in figure 3.3.

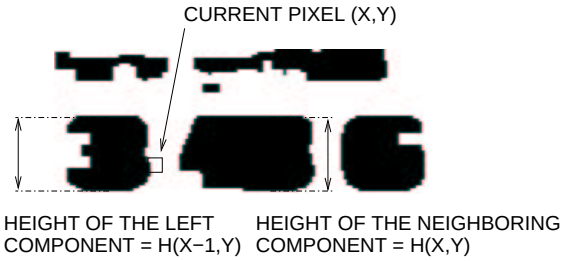


Figure 3.3: Determining the heights of neighboring connected components.

The difference function used for the conditions is defined as follows:

$$\Delta(a,b) = \frac{\mid a - b \mid}{\min(a,b)} \tag{3.7}$$

Dilation is performed, if:

- The relative difference of the heights of the connected component including the given pixel and the neighboring component to the right does not exceed a threshold t_2, i.e. $\Delta(\boldsymbol{H}_{x-1,y}, \boldsymbol{H}_{x,y}) < t_2$
- The relative difference of the y positions of these two connected components does not exceed a threshold t_3, i.e. $\Delta(\boldsymbol{P}_{x-1,y}, \boldsymbol{P}_{x,y}) < t_3$

If the test is valid and the maximum number of dilations (i.e. the number of iterations) since the last dilated text pixel has not been passed, then the pixel is set to a pseudo color which is neither 0 (non-set) nor 255 (set).

Step 4: Conditional erosion (16 iterations). Structuring element: $\boldsymbol{B}_C$.

The conditional erosion step performs a "standard" morphological erosion using the same structuring element $\boldsymbol{B}_C$, but contrary to the conditional dilation algorithm, there are additional conditions on the gray levels of the pixels instead of the shape. The image is eroded using the structuring element $\boldsymbol{B}_C$ with the condition that only pixels marked with the pseudo color are eroded. Finally, all pixels marked with the pseudo color are marked as text. The effect of this step is the connection of the all connected components which are horizontally aligned and whose heights are similar.

An example for the two conditional operations (Steps 3 and 4) is shown in figure 3.4. Figure 3.4a shows the original input image. Figure 3.4b shows the binarized accumulated gradients. The words are separated and there are also gaps between the digits. Figure 3.4c shows the pseudo color image after the conditional dilation step. Figure 3.4d shows the image after the conditional erosion. Since the neighboring components are similar in height and they are aligned horizontally, the pixels between them are dilated, which connects these components.

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

Figure 3.4: Original image (a) binarized accumulated gradients (b) after conditional dilation (c) and after conditional erosion (d).

Step 5: Horizontal erosion (12 iterations). Structuring element:

$$\boldsymbol{B}_H = \begin{bmatrix} 1 & \textcircled{1} & 1 \end{bmatrix} \tag{3.8}$$

This very strong horizontal erosion removes components which do not respect the hypothesis, that text has a certain minimum length. This hypothesis, which has already been imposed on the gray levels as well (text consists of vertical strokes arranged horizontally with a minimum length), distinguishes between small components containing high contrast strokes and text with a minimum length.

Step 6: Horizontal dilation (6 iterations). Structuring element: $\boldsymbol{B}_H$.

This dilation step resizes the remaining components to almost their original size (i.e. the estimated size of the text rectangles). Instead of performing a horizontal dilation with the same number of iterations, which bears the danger of reconnecting text components with non-text components, only half of the iterations are performed. A connected components analysis extracts the remaining components, whose bounding boxes are used for further processing. Finally, the bounding boxes are grown horizontally 3 pixels to the left and 3 pixels to the right in order to compensate for the difference in the horizontal erosion and dilation step.

Figure 3.1 shows the intermediate results of the detection chain. Figures 3.1a and 3.1b display the original image and the accumulated gradients, where text areas are visible as emphasized white rectangles. Figure 3.1c shows the binarized image which contains white text components and black non-text background, but also white non-text components due to the background texture in the original image. Almost all these are removed after imposing the morphological constraints, shown in figure 3.1d and 3.1e. Figure 3.1f shows the original image with superimposed bounding boxes of the connected components, after the geometrical constraints on the rectangles are imposed.

3.3 Geometrical constraints

After the morphological processing, the following geometrical constraints are imposed on the rectangles in order to further decrease their number:

$$\frac{width}{height} > t_4 \tag{3.9}$$

$$\frac{number\ of\ text\ pixels\ of\ the\ component}{area\ of\ the\ bounding\ box} > t_5 \tag{3.10}$$

where t_4 and t_5 are fixed thresholds.

Some special cases are considered, which increase the size of the bounding boxes of the rectangles to include the heads and tails of the characters, which have been removed by the morphological step. Finally, rectangles are combined in order to decrease their number. Two (partly) overlapping rectangles are combined into a single one of bigger dimension, if *one* of the following two conditions holds:

$$\begin{aligned} C_1: &\quad \frac{Area(R_s) - Area(R_b \cap R_s)}{Area(R_s)} < t_6 \\ C_2: &\quad \frac{Area(R_s) - Area(R_b \cap R_s)}{Area(R_s)} < t_8 \quad \wedge \quad \frac{Area(R_s)}{Area(R_b)} < t_7 \end{aligned} \tag{3.11}$$

where R_b is the bigger rectangle, R_s is the smaller rectangle, $Area(R)$ is the area of rectangle R, and $t_6 - t_8$ are fixed thresholds which respect the condition $t_6 < t_8$. In plain words, if the non-overlapping part of the small rectangle (i.e. the part which does not overlap with the bigger rectangle) is smaller than threshold t_6, then the two rectangles are joined. Condition 2 states, that the two rectangles are even joined if the non-overlapping part is bigger than t_6 but still smaller than the second threshold t_8 (since $t_8 > t_6$), if the difference in size of the two rectangles is big enough. In other words, if a very small rectangle is glued to a bigger rectangle, then the two are joined even if the overlap area is not as big as required by condition 1.

3.4 Experimental results

To estimate the performance of our system on still images, we used a database containing 384 ground-truthed images in format CIF (384 × 288 pixels), among which 192 contain text and 192 do not contain text. The ground truth also allowed us to optimize the parameters of the system. Table 3.4 shows the parameters we determined from this test database.

Please note, that the algorithm described in this chapter is also a part of the algorithm developed for the detection in video streams. The experimental results for video sequences are given in chapter 5.

3.4.1 Evaluation measures

Traditionally, the performance of information retrieval systems is evaluated with the measures of precision and recall:

$$\begin{aligned} \text{Recall}_{IR} &= \frac{\text{number of correctly retrieved items}}{\text{number of relevant items in the database}} \\ \text{Precision}_{IR} &= \frac{\text{number of correctly retrieved items}}{\text{total number of retrieved items}} \end{aligned} \tag{3.12}$$

In order to have a single performance value for the ranking of methods, the harmonic mean of precision and recall has been introduced by the information retrieval community [76]:

$$Mean = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

For object detection systems (and as a special case, text detection systems), the notion of "the object has been detected" is not well-defined, the question cannot be answered with a simple "yes" or "no", since objects may be partially detected. Therefore, the familiar precision/recall measures need to be changed to incorporate the quality of the detection. Unfortunately, until now there is no widely used evaluation scheme which is recognized by the scientists of the domain. We therefore used different evaluation schemes, among which are schemes proposed by other researchers and our own techniques which address some short comings of the already existing ones.

The goal of a detection evaluation scheme is to take a list of ground truth text rectangles $G_i, i = 1..|G|$ and a list of detected text rectangles $D_j, j = 1..|D|$ and to measure the quality of the match between the two lists. The quality measure should be goal-oriented, i.e. it should depend on the task at hand. It should penalize information loss, which occurs if text has not been detected, and it should penalize information clutter, i.e. false alarms. There are several possibilities:

- A goal oriented evaluation scheme for an information retrieval system which exploits the text content (as opposed to its position) should take into account the results of the recognition step, and therefore should penalize lost text characters and additional characters which are not present in the ground truth.
- A goal oriented evaluation scheme for an information retrieval system which exploits the text position (as opposed to its content) should take into account geometrical information, e.g. the amount of overlap between the ground truth rectangles and the detected rectangles, as well as the size of the non-overlapping parts.

We concentrated our detection evaluation on schemes which exploit geometrical information of the rectangles. The evaluation of the binarization and the recognition techniques, which takes into account lost and added characters, is presented in section 6.3.

In the following, we will present three different evaluation schemes which we used. They differ in the way how they treat the correspondence problem between rectangles, i.e. whether they consider single matches only or multiple matches, and in the way

how they incorporate the geometrical information (overlap) into the precision and recall measures, i.e. if the information is thresholded or not.

The ICDAR measure

A simple but effective evaluation scheme has been used to evaluate the systems participating at the text locating competition in the framework of the 7^{th} International Conference on Document Analysis and Recognition (ICDAR) 2003 [46]. The two measures, recall and precision, are changed slightly in order to take into account the amount of overlap between the rectangles:

$$\begin{aligned} \text{Recall}_{ICDAR} &= \frac{\sum_i Match_G(G_i)}{|G|} \\ \text{Precision}_{ICDAR} &= \frac{\sum_j Match_D(D_j)}{|D|} \end{aligned} \tag{3.13}$$

where $|D|$ is the number of detected rectangles, $|G|$ the number of ground truth rectangles, and $Match_G$ and $Match_D$ are functions which deliver the quality of the closest match of a rectangle in the opposing list:

$$Match_G(G_i) = \max_j \frac{2 \cdot Area(G_i \cap D_j)}{Area(G_i) + Area(D_j)}$$

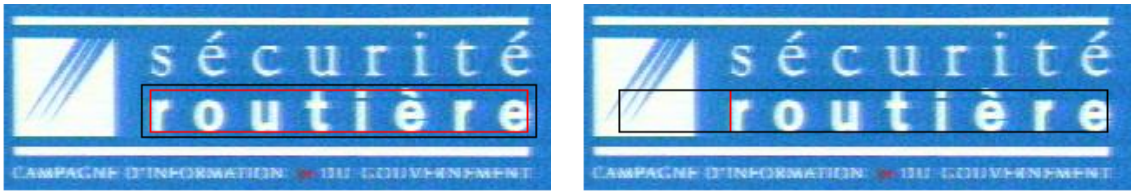
$$Match_D(D_j) = \max_i \frac{2\ Area(D_j \cap G_i)}{Area(D_j) + Area(G_i)}$$

If a rectangle is matched perfectly by another rectangle in the opposing list, then the match functions evaluate to 1, else they evaluate to a value < 1. Therefore, the measures given by (3.12) are upper bounds for the measures given by (3.13). Both, precision and recall given by (3.13), are low if the overlap region of the corresponding rectangles is small.

The CRISP measure

The ICDAR evaluation scheme has several drawbacks:

- Only one-to-one matches are considered. However, in reality sometimes one ground truth rectangle may correspond to several text rectangles or vice-versa. This is problem the authors themselves report in [46].
- The amount of overlap between two rectangles is not a perceptively valid measure of quality, as can be seen in figure 3.5. Precision and recall are equivalent for both detection examples, but the detection shown in figure 3.5a might be considered as better, since the additional detected space is distributed over all sides of the ground truth rectangle.



(a) (b)

Figure 3.5: ground truth rectangle and detected rectangles for an example image. Precision and recall for figures (a) and (b) are equivalent.

- The inclusion of overlap information into the evaluation measures leaves room for ambiguity: a recall of 50% could mean that 50% of the ground truth rectangles have been matched perfectly, or that all ground truth rectangles have been found but only with an overlap of 50%, or anything in between these two extremes.

We developed an evaluation scheme which addresses these problems. Inspired by the method presented in [43], it takes into account one-to-one as well as many-to-one and one-to-many matches. However, the algorithm aims at an exact determination — controlled by thresholds — whether a ground truth rectangle has been detected or not.

From the two lists G and D of detected rectangles and ground truth rectangles, two overlap matrices σ and τ are created. The lines $i = 1..|G|$ of the matrices correspond to the ground truth rectangles and the columns $j = 1..|D|$ correspond to the detected rectangles. The values are calculated as follows:

$$\sigma_{ij} = \frac{Area(G_i \cap D_j)}{Area(G_i)} \quad \text{and} \quad \tau_{ij} = \frac{Area(G_i \cap D_j)}{Area(D_j)}$$

In a certain way, the values σ_{ij} and τ_{ij} may be considered as recall and precision, respectively, considering a detection of ground truth rectangle i by rectangle j. The matrices can be analyzed in order to determine the correspondences between the two lists:

- one-to-one matches: G_i matches against D_j if row i of both matrices contains only one non-zero element at column j and column j of both matrices contains only one non-zero element at row i. We enforce the additional conditions of $\sigma_{ij} \geq e_1 = 0.8$ and $\tau_{ij} \geq e_2 = 0.4$, i.e. the overlap area needs to have a certain size compared to the rectangles in order to be considered as successfully detected. This situation is shown in figure 3.6a.

- one-to-many matches with one ground truth rectangle: G_i matches against several detected rectangles if row i of the matrices contains only one non-zero element at column j. We enforce the additional constraints of $\sum_j \sigma_{ij} \geq e_3 = 0.8$ and $\forall j : \tau_{ij} \geq e_4 = 0.4$. Figure 3.6b illustrates this match type. The threshold e_3 ensures

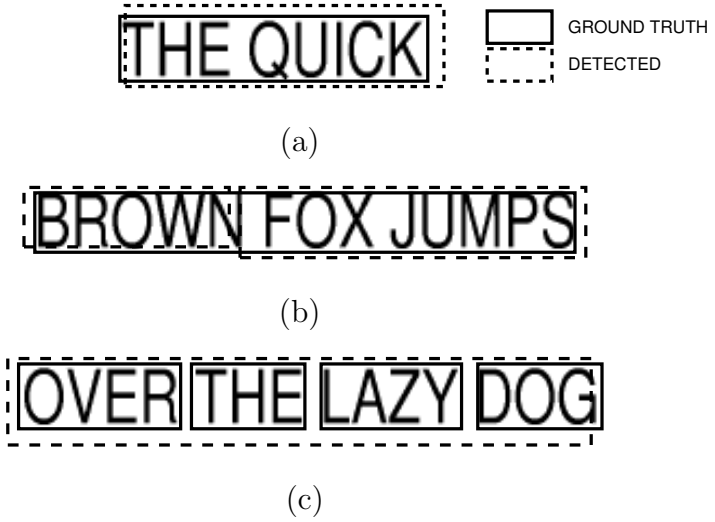


Figure 3.6: Different match types between ground truth rectangles and detected rectangles: (a) one-to-one match; (b) one-to-many with one ground truth rectangle; (c) one-to-many with one detected rectangle.

that the single ground truth rectangle is sufficiently detected (i.e., with high recall) and the threshold e_4 ensures that each of detected rectangles is precisely enough.

- one-to-many matches with one detected rectangle: D_j matches against several ground truth rectangles if column j of the matrices contains only one non-zero element at row i. We enforce the additional constraints of $\sum_i \tau_{ij} \geq e_5 = 0.4$ and $\forall i : \sigma_{ij} \geq e_6 = 0.8$. Figure 3.6c shows this situation. Here, threshold e_5 guarantees the precision of the detection and threshold e_6 ensures that each of the ground truth rectangles is detected sufficiently (i.e. with high recall).

One-to-one matches need to fulfill an additional geometrical constrained in order to be validated: the differences of the left (respectively right) coordinates of the rectangles need to be smaller than a threshold which depends on the width of the ground truth rectangle. This constraint, which does not depend on the overlap information, makes sure that a situation depicted in figure 3.5b is unlikely to occur. Table 3.1 gives a summary of the thresholds and their meaning. The threshold values have been chosen empirically. However, in section 3.4.2 we present performance graphs for different threshold values.

Based on this matching strategy, the recall and precision measures are given as follows:

$$\begin{aligned} \text{Recall}_{CRISP} &= \frac{\sum_i MatchCrisp_G(G_i)}{|G|} \\ \text{Precision}_{CRISP} &= \frac{\sum_j MatchCrisp_D(D_j)}{|D|} \end{aligned} \tag{3.14}$$

Thr.	Match type	Description	Value
e_1	one to one	Thresholds the detection recall.	0.8
e_2	one to one	Thresholds the detection precision.	0.4
e_3	one ground truth to many detected	Thresholds the detection recall for the single ground truth rectangle.	0.8
e_4	one ground truth to many detected	Thresholds the detection precisions for each of the detected rectangles.	0.4
e_5	one detected to many ground truth	Thresholds the detection precision.	0.4
e_6	one detected to many ground truth	Thresholds the detection recall for each of the ground truth rectangles.	0.8
e_7	one to one	Thresholds the differences of the x-coordinates of the borders	adaptive

Table 3.1: The thresholds controlling the crisp evaluation scheme and their meaning.

where $MatchCrisp_G$ and $MatchCrisp_D$ are functions which take into account the different types of matches described above and which evaluate to the quality of the match:

$$MatchCrisp_G(G_i) = \begin{cases} 1 & \text{if } G_i \text{ matches against a single detected rectangle} \\ 0 & \text{if } G_i \text{ does not match against any detected rectangle} \\ 0.8 & \text{if } G_i \text{ matches against several detected rectangles} \end{cases}$$

The function $MatchCrisp_D$ is defined accordingly. This evaluation scheme takes into account one-to-many matches, but does "punish" them slightly (with a value of 0.8 as opposed to 1). As the traditional measures given by (3.12), these measures provide an intuitive impression on how many rectangles have been detected correctly and how many false alarms have been produced. Please note, however, that text which is only partly detected and therefore not matched against a ground truth rectangle, will decrease the precision measure, in contrast to the ICDAR evaluation scheme.

The AREA measure

As a third detection evaluation scheme we want to present a combination of the ICDAR scheme (3.13) and the CRISP scheme (3.14): Recall and precision reflect the overlap information **and** take into account one-to-many matches:

$$\begin{aligned} \text{Recall}_{AREA} &= \frac{\sum_i \sum_{j\ MatchCrisp\ i} Area(G_i \cap D_j)}{\sum_i Area(G_i)} \\ \text{Precision}_{AREA} &= \frac{\sum_j \sum_{i\ MatchCrisp\ j} Area(D_j \cap G_i)}{\sum_j Area(D_j)} \end{aligned} \tag{3.15}$$

Scheme	Advantages	Disadvantages
ICDAR	+ simple + overlap may provide additional information	- only one-to-one matches - overlap is ambiguous
CRISP	+ one-to-many matches + intuitive	- depends on thresholds - partly detected rectangles decrease precision
AREA	+ one-to-many matches + overlap may provide additional information	- overlap is ambiguous - depends on thresholds - partly detected rectangles decrease precision

Table 3.2: A summary of the text detection evaluation schemes used.

A summary of the text detection evaluation schemes described above with their advantages and disadvantages is given in table 3.2.

Generality

As for information retrieval (IR) tasks, the measured performance of an object detection algorithm highly depends on the test database. It is obvious, that the nature of the images determines the performance of the algorithm. As an example we could think of the text type (artificial text or scene text), its size, the image quality, noise, fonts and styles, compression artifacts etc. For this reason, an objective comparison between different algorithms will only be possible if the text detection community decides on a shared common test database. We decided to tackle this problem partly by performing different experiments for different test databases (e.g. including scene text, only artificial text etc.).

On the other hand, the nature of the images is not the only variable which determines the influence of the test database on the detection performance. The structure of the data, i.e. the ratio between the relevant data and the irrelevant data, is a major factor which influences the results. This simple but important fact has been overlooked by the information retrieval community for a long time.

In [28], Huijsmans et al. call attention to this fact and adapt the well known precision/recall graphs in order to link them to the notion of generality for an IR system, which is defined as follows:

$$\text{Generality}_{IR} = \frac{\text{number of relevant items in the database}}{\text{number of items in the database}}$$

Very large databases with low generality, i.e. much irrelevant clutter compared to the relevant material, produce results with lower precision than databases with higher generality. This makes sense, since the probability to retrieve a relevant item is lower if there is more irrelevant noise present in the database. A standard IR system presents the retrieved items to the user in a result set of predefined size. Since this size is fixed, with falling generality the amount of relevant material in the result set — thus the recall — will tend to be smaller. Thus, recall and precision depend on the generality of the database. In IR one is interested in the retrieval performance with respect to the generality as well as with respect to the size of the result set, which determines the search effort for the user. The dependence on two parameters makes three-dimensional performance graphs necessary. Details on how to overcome this problem can be found in [28].

However, unlike IR tasks, text detection algorithms do not work with items (images, videos or documents). Instead, images (or videos) are used as input, and text rectangles are retrieved. Nevertheless, a notion of generality can be defined as the amount of text which is present in the images of the database. We define it to be

$$\text{Generality} = \frac{\text{Number of text rectangles in the database}}{\text{Number of images in the database}} \tag{3.16}$$

Another difference to IR systems is the lack of a result set window, because all detected items are returned to the user. Therefore, the generality of the database does influence precision, but *not* recall. Thus, the influence of the database structure on the system performance can be shown with simple two-dimensional precision/generality graphs. The graphs introduced by Huijsmans are displayed on a logarithmic scale, since the generality in very large IR databases may attain very low values. On the other hand, the amount of text per image (or per video frame) should remain relatively high, therefore we decided to display the graphs on a linear scale.

Finally, a decision needed to be made concerning the generality level of the database when result tables or graphs are displayed which contain a fixed level a generality. In other words, we needed to decide how many images with zero ground truth (no text present) should be included in the database. We concluded, that a mixture of 50% images with text and 50% images without text should be a reasonable level. We should keep in mind that this amount is not representative for realistic video streams. However, a larger part of non-text images would introduce a higher bias into the detection system. The detection results for realistic videos are given in chapter 5.

3.4.2 Results

Figures 3.7 and 3.8 show the performance of the contrast based detection system on, respectively, 15 images containing text and 15 images not containing any text. As can

Dataset	#†	G††	Eval. scheme	Recall	Precision	H. mean
Figure 3.7:	15	6.20	ICDAR	66.2	60.7	63.3
Artificial text			CRISP	84.3	72.0	77.7
			AREA	88.8	64.3	74.6
Figures 3.7+3.8:	30	3.10	ICDAR	66.2	27.5	38.8
Artificial text +			CRISP	84.3	32.6	47.0
no text			AREA	88.8	36.1	51.4
Artificial text +	144	1.49	ICDAR	70.2	18.0	28.6
no text			CRISP	81.2	20.1	32.3
			AREA	83.5	26.3	40.0
Artificial text +	384	1.84	ICDAR	55.9	17.3	26.4
scene text +			CRISP	59.1	18.1	27.7
no text			AREA	60.8	21.9	32.2

†Number of images

††Generality

Table 3.3: The detection results of the contrast based method for different datasets and different evaluation schemes.

be seen, the performance on the images containing text is excellent. Most of the text is detected and the precision is very high. Unfortunately, the system also detects text on the images which do not contain text.

In order to relate the detection results illustrated on the images in figures 3.7 and 3.8 to the evaluation measures defined in the previous section, we give the performance measures for these images in the upper two segments of table 3.3. Evaluated with the CRISP evaluation scheme, we achieve on the 15 images containing text a recall of 84.4% and a precision of 71.9%. The performance values given with the mixed scheme (AREA) is close to the ones calculated with the CRISP scheme. The lower performance evaluated with the ICDAR method can be explained with the restriction to one-to-one matches. As an example we observe the string "1 40 3 43 6" of lotto numbers in the right image of the top row which has been detected as a single rectangle, whereas the ground truth contained separate rectangles for each number. This is taken into account by the CRISP and AREA methods but not by the ICDAR method.

As described in the previous section, the recall remains the same if we add the 15 images not containing text to the dataset, which now contains 30 images. However, since the generality falls from 6.2 to 3.1, also the precision drops considerately from 72% to 32.6%.

The lower two segments of table 3.3 give the figures for the whole dataset. The third segment shows the performance for the images including artificial text only, and the fourth segment the performance for the artificial text and scene text. In both cases, the datasets contain 50% images with text and 50% images not containing text. If the

Figure 3.7: Some detection examples on images with text.

Figure 3.8: Some detection examples on images without text.

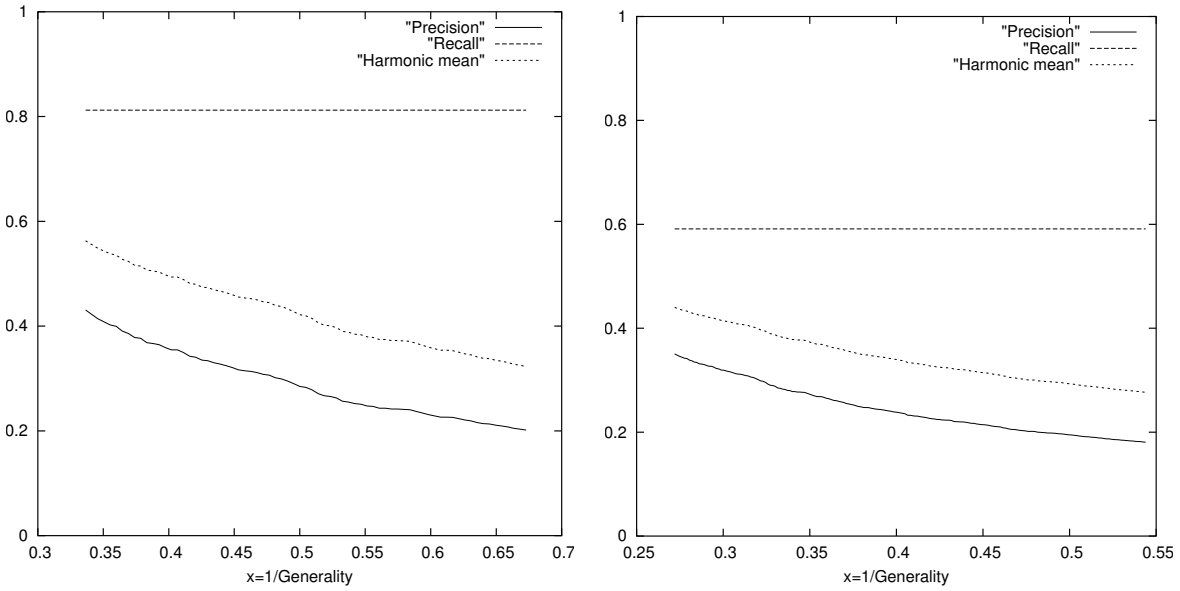


Figure 3.9: Precision$_{CRISP}$ for different generalities. Reciprocal generality is displayed on the x-axis: (a) artificial text + no text; (b) artifical text + scene text + no text.

system is used in order to index broadcasted video, then an emphasis on artificial text is reasonable. In this case, we achieve a recall of 81.2%. However, results on videos are presented in section 5. As expected, the performance on a dataset including scene text is much lower. In our case, the recall drops to 59.1%.

The influence of generality

The dependence of precision on the generality of the dataset is given in figures 3.9a and 3.9b for, respectively, the dataset containing artificial text only and the dataset containing both types of text. Since information retrieval scientists are used to performance curves with dropping performances (e.g. precision/recall curves), we displayed precision on the y-axis and the reciprocal of the generality on the x-axis. The highest generality value — situated at the left border of the curve — is achieved if only images with text are included in the dataset. As we traverse the curve to the right, images with zero ground truth are added to database which results in a dropping precision value. In our experiments, the furthest point on the right is reached when we added as many images with zero ground truth as there are images with text.

The influence of the evaluation thresholds

As described in section 3.4.1, the CRISP evaluation method depends on thresholds which decide whether a given ground truth rectangle has been detected sufficiently or not and

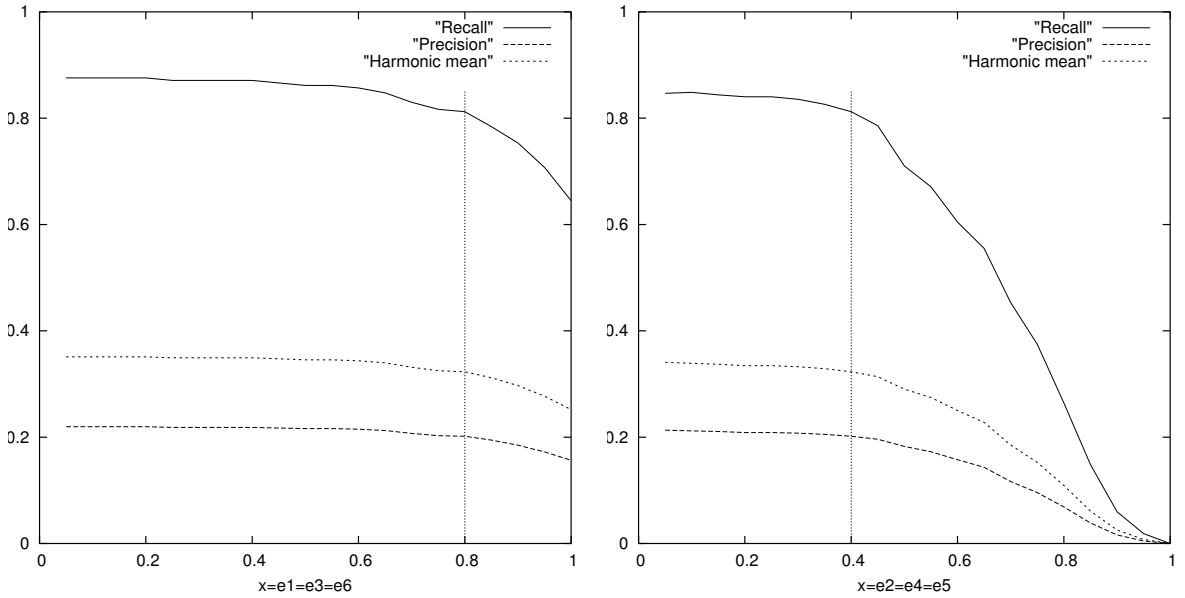


Figure 3.10: The system performance evaluated with the CRISP method for different evaluation thresholds (the vertical bars indicate the threshold chosen throughout this work): (a) changing thresholds e_1, e_3 and e_6; (b) changing thresholds e_2, e_4 and e_5.

whether the detection is precise enough or not. The former condition is controlled by the thresholds e_1, e_3 and e_6, the latter is controlled by the thresholds e_2, e_4 and e_5. We performed experiments with different values for these two groups and give the result curves in figures 3.10a and 3.10b, respectively.

As can be seen in figure 3.10a, the evaluated detection performance only slightly depends on the "recall" thresholds e_1, e_3 and e_6. Even when a surface recall of 100% per rectangle is enforced, overall detection recall only drops to 64.5%, down from 81.2% at the chosen 'standard" threshold of 80% surface recall. This may be explained by the fact, that most text is detected with a slightly larger rectangle than the retangle given in the ground truth.

On the other hand, the detection performance significantly depends on the "precision" thresholds e_2, e_4 and e_5, explained by the same reason mentioned above. When a surface precision of 100% per rectangle is enforced, the performance drops to zero. Unfortunately, as explained in section 3.4.1, the surface precision per rectangle alone is not a perceptively valid measure capable of deciding whether the detection performance is sufficiently precise. As an example we cited figure 3.5, which shows two different detection results with a precision value of only 77.1%. This relatively low value is astonishing given the detection in figure 3.5a, which is perceived as rather precise. For this reason we set the "standard" threshold throughout this work to a surface precision of 0.4, a rather small value. However, we added an additional constraint through threshold e_7

Par.	Description
S	Size of the gradient accumulation filter.
α	Placement of the second threshold for the binarization of the accumulated gradients.
t_1	Threshold on column height.
t_2	Threshold on height difference.
t_3	Threshold on position difference.
t_4	Threshold on ratio width/height.
t_5	Threshold on ratio pixels/area.
t_6	Threshold for combining rectangles.
t_7	Threshold for combining rectangles.
t_8	Threshold for combining rectangles.

Table 3.4: The parameters of the system.

on the differences of the x-coordinates of the ground truth rectangle and the detected rectangle, as mentioned in section 3.4.1.

The influence of the system parameters

Figure 3.11 shows the CRISP performance measures for different values of the four parameters S (accumulation length), α (binarization) and t_4 and t_6 (geometrical constraints). The size S of the accumulation window should be chosen according to the character size in the dataset. However, as can be seen in figure 3.11a, a change of the parameter in the range 10-20 pixels does not change the system performance significantly. As figure 3.11b confirms, the detection algorithm is robust against small changes of the parameters α.

The parameters t_4 and t_5 are thresholds on the geometrical constraints of the detected text (shape of the rectangle and amount of text pixels in the rectangle). They therefore act as text rectangle filters, which can be used to control the ratio between recall and precision. Enforcing tighter thresholds increases precision with the drawback of a lower recall.

Finally, table 3.4 summarizes the total set of parameters of the detection system.

3.4.3 Execution time

The execution time of the algorithm implemented in C++ on a Pentium III with 700 Mhz[2] running under Linux is about 0.3 seconds for an image in CIF format.

[2]The PC was bought in July 2000.

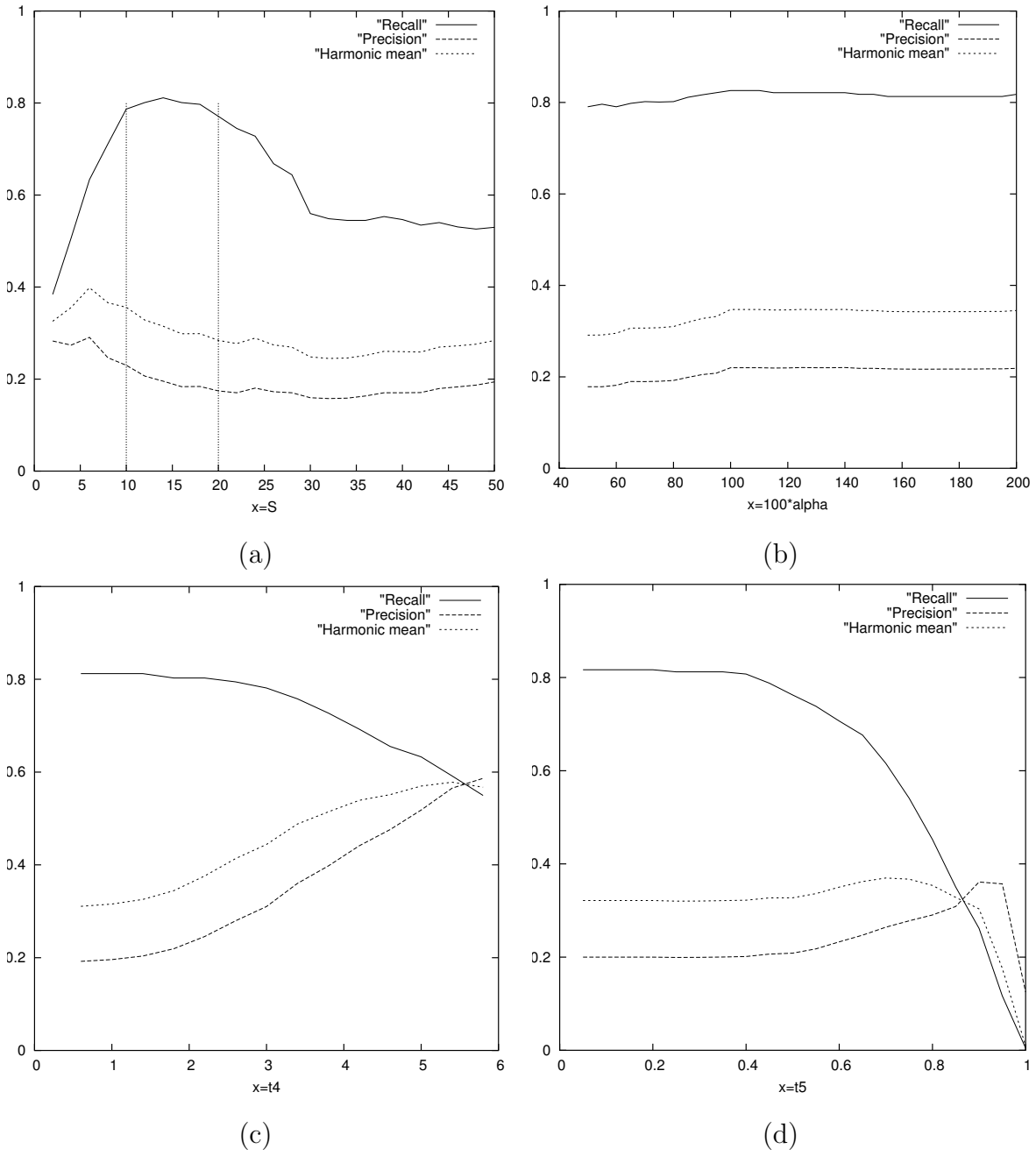


Figure 3.11: Precision and recall on the still images database for different parameter values: (a) the influence of S; (b) the influence of t_2; (c) the influence of t_4; (d) the influence of t_5.

Chapter 4

Detection in still images - a learning approach

The big question about a theory is whether it's right or wrong.
Unfortunately, it's impossible to know that a scientific theory is right.
The theory may agree beautifully with all the evidence - today.
But science isn't like mathematics.
There can be no guarantee about what evidence we will discover tomorrow.

Sir Karl Popper
Austrian epistemologist (1902-1994)

In this chapter we present our text detection method based on learning. In contrast to the method introduced in the last chapter, we do *not* need to assume that there is text present in the image. The threshold, which has been found by statistical modeling of two populations in the last algorithm, is learned from training data. More precisely, this "thresholding" is done internally by the learning machine. The use of machine learning also allows us to benefit from several advantages:

- We hope to increase the precision of the detection algorithm by learning the characteristics of text.
- We are able to use more complex text models, which would be very difficult to derive analytically.
- The discovery of support vector machine (SVM) learning and it's ability to generalize even in high dimensional spaces opens the door to complex decision functions and feature models.

Of course, the advantages of machine learning come with several trade-offs. As for all learning algorithms, the training set needs to be chosen carefully to ensure high generalization. The danger of specialization to a specific type of text needs to be avoided.

Unfortunately, text exists in wide varies of forms, fonts, sizes, orientations and deformations (especially scene text), which makes generalization difficult.

We tackle the generalization problem with the introduction of higher level features instead of simple gray values or high frequency components. We reuse the methods presented in the last chapter, and complete it by adding higher level features. Generalization to multiple orientations is achieved by applying the directional filter to different principal orientations and by merging the responses. In the remainder of this chapter, directions as "horizontal" or "vertical" should be understood as meant for the horizontal text filter only. The adaptation to different orientations is straightforward.

As discussed in chapter 2.4, research in text detection algorithms passed through two phases: the phase of segmentation based algorithms, which has been initiated by the document processing community, and the phase of texture/edge based techniques, which has been adopted by the image/video processing community. In our opinion, research needs to advance going one more step further by the usage of higher level features for detection. The distinction between text and non-text cannot be done considering very local characteristics of the signal only. One of the main properties of text is its geometry: The presence of a rectangular bounding box and/or the presence of a baseline[1]. On the other hand, the texture properties of text tend to get more and more unreliable, as the detected text gets shorter and shorter.

Recapitulating, our text model contains the following hypotheses:

Text contains a high amount of *vertical strokes*.

Text contains a *baseline*, i.e. its border area forms a regular rectangle.

Until now, text detection features have been very simple: gradient or edge densities, texture features based on filtering (Gabor, derivatives of a Gaussian etc.) or high frequency components of wavelet decompositions. While it is true, that higher level features as geometrical constraints etc. are enforced by most of the existing methods, they are employed at a second stage for verification of a segmented rectangle only. Very often, mathematical morphology is used to clean up noise and also to enforce geometrical constraints, but once again this happens after the classification of the pixels — whether they are text or non-text — has been done. As a consequence, the weakness of these approaches is the hard (up to impossible) improvement of weak classification decisions. A very badly segmented image, where the text box touches a complex background in many places, or where a text box and the background make a single convex region, are impossible to correct.

[1] In document analysis, the baseline is traditionally the lower boundary of the text area. In our case, when we refer to "baseline", we mean the upper and/or the lower boundary line of the text, since the presence of these two lines is a necessary condition for text.

A logical step is to include the geometrical features directly into the decision process for each pixel. Two strategies are possible:

- The geometrical features are calculated directly in the image. This is rather difficult due to the irregular, texture nature of text.
- Calculate the geometrical features on an image containing already detected text. This is a chicken and egg problem: to detect the geometrical features of text, very often it is necessary to first detect the text, hence the usage of the geometrical features for verification only in the classical methods.

Figures 4.1a and 4.1b, respectively, present schemes for the two strategies. The first method directly calculates two different features in the image : the accumulated gradients introduced in chapter 3.1, which respond to areas containing a high amount of vertical strokes, and propagated and accumulated corner responses. In text areas, corners are typically found in the baseline or the upper borderline of the text. Propagating or accumulating the corner responses into the text area will result in a filter which responds to text with a consistent baseline.

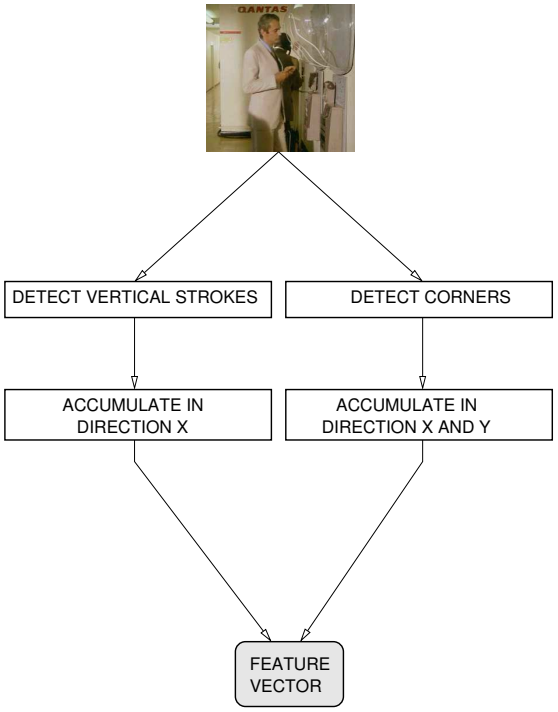
The second method (figure 4.1b) consists of a two step approach: at first, a coarse detection gives an indication about the presence of text for each pixel. Please note, that this is not a classification, since no decision whatsoever is taken at this step. From the resulting "text probability image", we calculate geometrical features, which are used for the classification on pixel level.

4.1 Corner response propagation

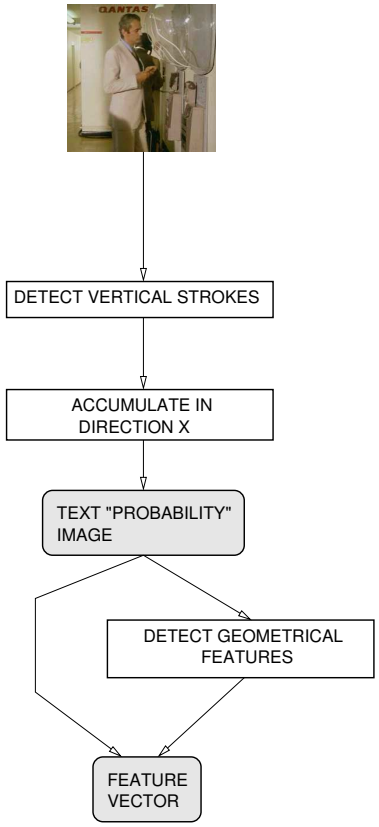
As already stated, we take advantage of the fact that the baseline of text tends to produce high responses to corner detection algorithms. This is due to the fact, that the characters contain stroke end points at this area. If we want to create a filter which exploits this property, we need to translate this high response at the baseline into a high response at the whole text area.

Figure 4.2 illustrates the way how we perform this propagation of corner responses into the text area. The corner responses are calculated with four directional corner detectors described below. Then, we propagate the responses into the direction of the gradient (see figure 4.1c). After the propagation, the responses are accumulated in the same way as the gradients, which has been described in section 3.1.

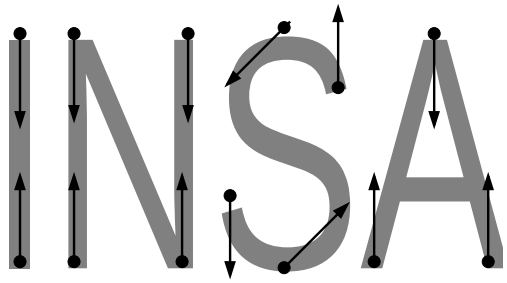
Although there are sophisticated corner detectors in the computer vision literature (Harris, Susan, etc.), we did not use them in our method since their responses are very non-linear and their response has only a weak correspondence to the contrast of the corner. These detectors have been used successfully in robotics and image or video indexing, where a fixed number of points needs to be extracted, but the extraction of a variable number of points with a certain amount of contrast is difficult.



(a)



(b)



(c)

(d)

Figure 4.1: Two different methods to include geometrical features into the detection phase: (a) searching the baseline using corners; (b) detecting geometrical features using an intermediate coarse detection; (c) propagating the corners of the characters into the center; (d) the baseline of a scene text example.

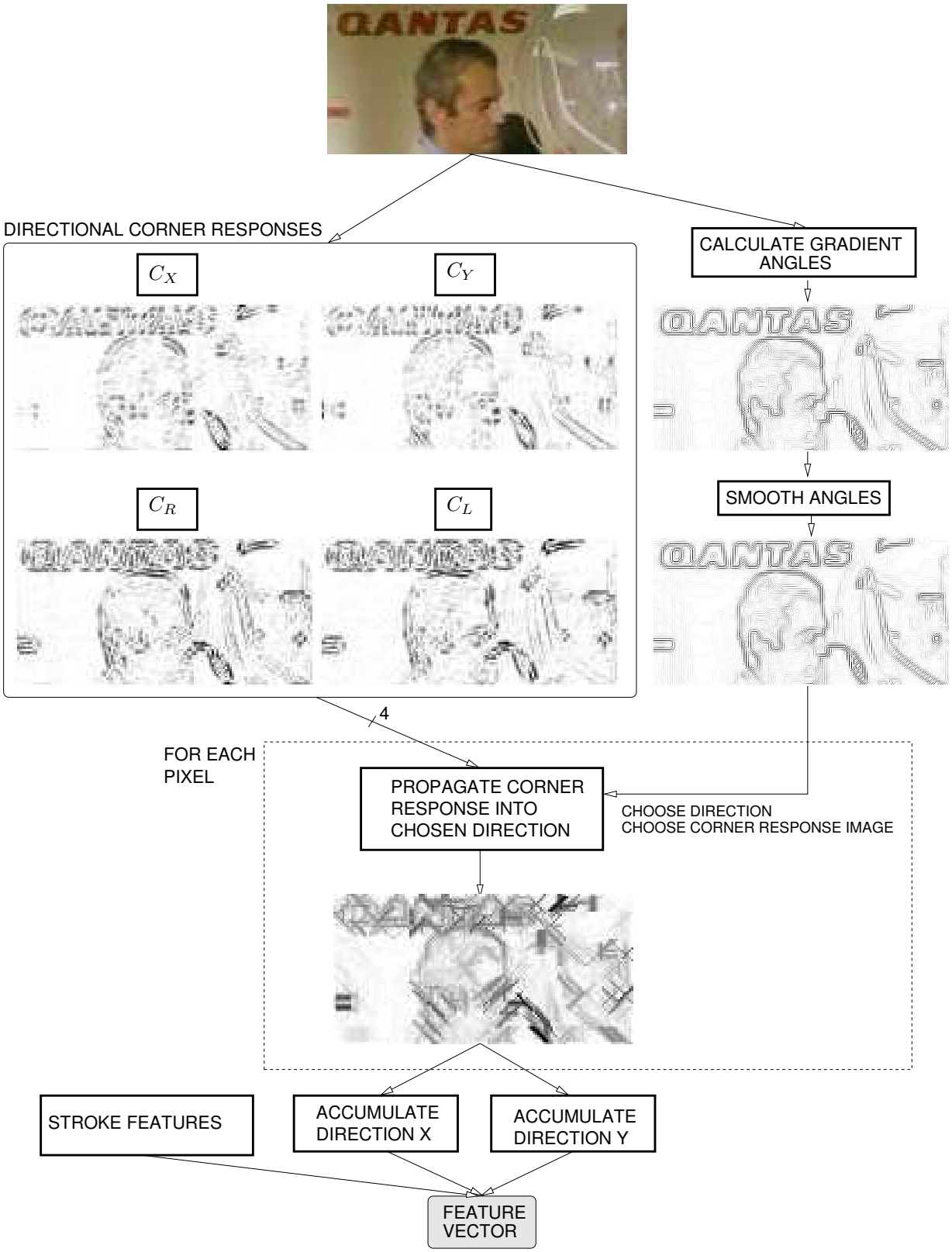


Figure 4.2: Accumulation of a baseline presence feature by corner propagation.

Instead, we calculate a set of 4 directional and very simple corner measures based on derivative filters:

$$C_X = \left|\frac{\partial}{\partial y}\left|\frac{\partial I}{\partial x}\right|\right| \qquad C_Y = \left|\frac{\partial}{\partial x}\left|\frac{\partial I}{\partial y}\right|\right| \qquad C_R = \left|\frac{\partial}{\partial l}\left|\frac{\partial I}{\partial r}\right|\right| \qquad C_L = \left|\frac{\partial}{\partial r}\left|\frac{\partial I}{\partial l}\right|\right|$$

where C_X detects the end points of vertical strokes (hence of horizontal text), C_Y detects the end points of horizontal strokes and C_R and C_L are used for diagonal text. These measures have the disadvantage of not being invariant to rotations, but their responses are easier to handle and they have a stronger correspondence to the contrast of a corner.

The propagation is carried out in one of four possible discrete orientations:

$$\Theta = \{X, Y, R, L\}$$

where X is the horizontal orientation, Y the vertical orientation, R right diagonal and L the left diagonal orientation. The orientation is determined from the gradient angle ϕ of a pixel, which is also quantized into these principal orientations according to the following function:

$$\Phi = q(\phi) = \begin{cases} X & \text{if } \phi \in [-\frac{\pi}{2}, -\frac{3\pi}{8}[\text{ or } \phi \in [\frac{3\pi}{8}, \frac{\pi}{2}[\\ R & \text{if } \phi \in [-\frac{3\pi}{8}, -\frac{\pi}{8}[\\ Y & \text{if } \phi \in [-\frac{\pi}{8}, \frac{\pi}{8}[\\ L & \text{if } \phi \in [\frac{\pi}{8}, \frac{3\pi}{8}[\end{cases}$$

The gradient angle of each pixel p is smoothed using the gradient information of the 8 neighbor pixels. A histogram quantized by the four orientations counts the number of pixels for each orientation in the neighborhood. The entries are weighted by a Gaussian mask $\boldsymbol{W} = [1\ 2\ 1]^T \otimes [1\ 2\ 1]$ centered on the pixel to smooth, and by the gradient magnitude of the neighbor pixel. The smoothed angle Φ_s is therefore calculated as follows:

$$\Phi_s = \arg\max_{\Phi' \in \Theta} \sum_{i \in \text{Neighbors}(p)} \delta_{\Phi' \Phi_i} \boldsymbol{W}_i ||\nabla_i||$$

where δ_{ij} is the Kronecker delta defined by

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{else} \end{cases}$$

The corner propagation algorithm works as follows:

- Initialize the corner response image C_P with zero.
- Calculate four corner response images C_H, C_V, C_R and C_L for the four principal orientations.

- Calculate the gradient angles for the image.
- Smooth the gradient angles using the algorithm described above.
- Traverse *all* pixels (x, y) of the image C_P and for each pixel do:
  - Choose a direction Φ_s and one of the four corner responses C_{Φ_s}, $\Phi_s \in \{X, Y, R, L\}$ according to the smoothed gradient angle at this pixel and the sign of the corresponding second derivative at this point. E.g. if the gradient angle is $\approx$0, then the propagation direction is vertical. In this case, the sign of the corner response $\frac{\partial}{\partial x}\left|\frac{\partial I}{\partial y}\right|$ (as opposed to its absolute value) determines whether we propagate upwards (positive sign) or downwards (negative sign).
  - Save the corner response value of this pixel: $C^{start} = C_{\Phi_s}(x, y)$
  - Walk along the chosen direction and set the walked pixels to the maximum of it's original value and the saved corner response of the start pixel C^{start}. The length of this walk is specified by the parameter S_p, which is related to the a priori text size.

Figure 4.3 illustrates the different steps of the propagation algorithm. Figure 4.3b shows the smoothed and discretized gradient angles. For each pixel, a short line in the direction of the gradient is drawn with a grayvalue which is proportional to the norm of the gradient. Figure 4.3c presents one of the four corner response images (C_X, i.e. the one which is relevant for horizontal text). Figure 4.3d shows the image of propagated responses. As can be seen, high corner responses result in strokes with length S_p, the propagation parameter.

We traverse the whole image, and each pixel of the image is a starting point for a walk in the direction of the smoothed gradient at this pixel. Since during the walk the encountered pixels are replaced with a maximum of their value and the start value, the end result after the traversal of the image does not depend on the way how we traverse the image, i.e. in which order we choose the starting points.

Note, that the corner response values are *not* thresholded. All pixels of the image are traversed and all values are propagated. However, areas with do not contain text are less likely to contain corners with high contrast, therefore the image of propagated responses only shows isolated strokes in these areas. In text areas, on the other hand, the frequent corners result in a large density of strokes *inside* of the area. Finally, figure 4.3e and 4.3f show the horizontal accumulation of the image of propagated corners and the accumulated gradients, respectively.

The filter responses described above serve as a basis for a feature vector, which is computed for each pixel. The feature vectors are used by the learning machine in order to learn the characteristics of text and non-text pixels (see section 4.4).

More precisely, each feature vector for a given pixel P contains accumulated gradients and accumulated values from the image of propagated corner responses (see figure 4.4 and table 4.1). The values are accumulated in two different directions and for each

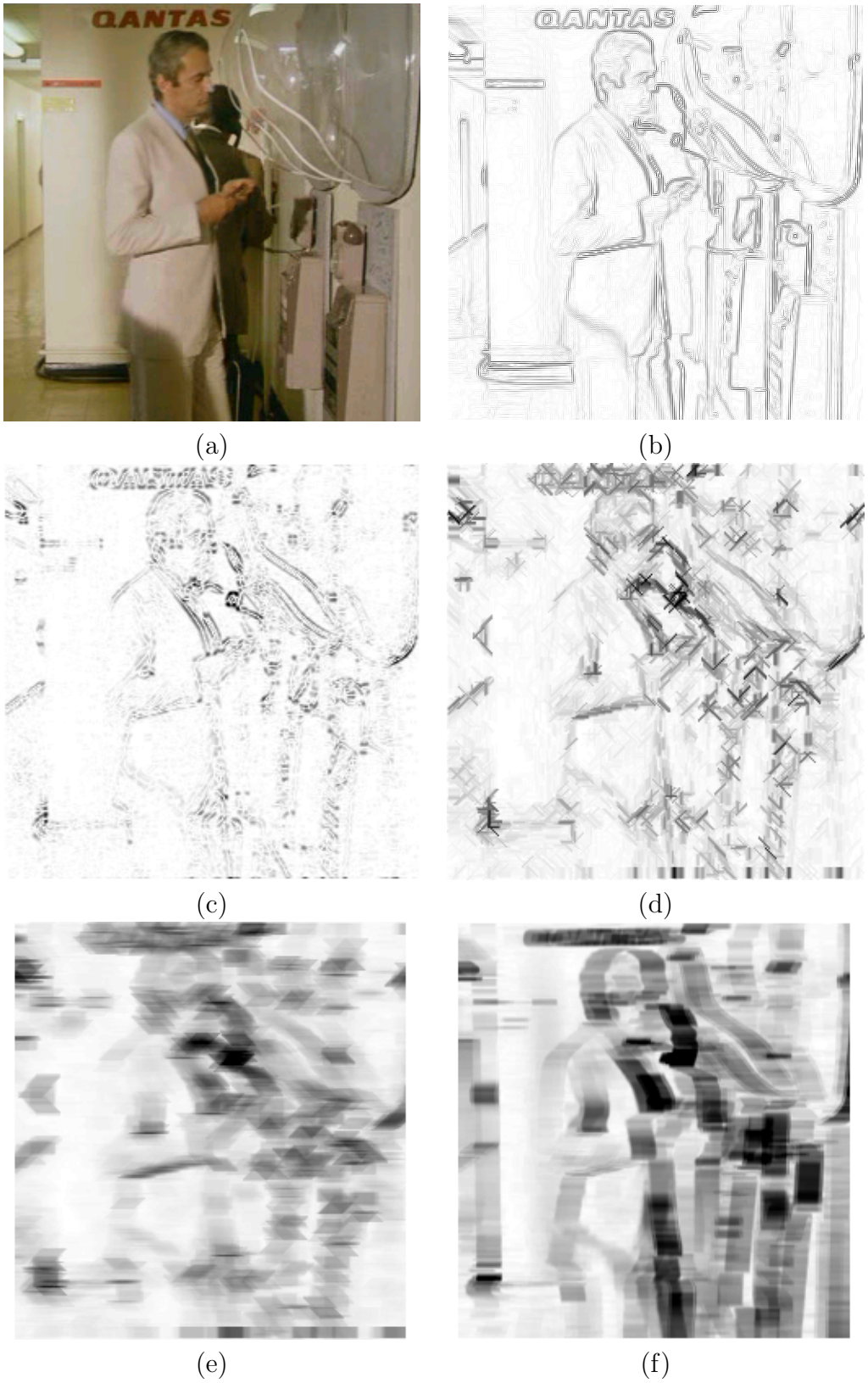


Figure 4.3: The corner propagation: (a) the original image; (b) the smoothed gradient angles; (c) the corner response C_X; (d) the propagated responses; (e) the accumulated propagated responses; (f) the accumulated gradients from section 3.1.

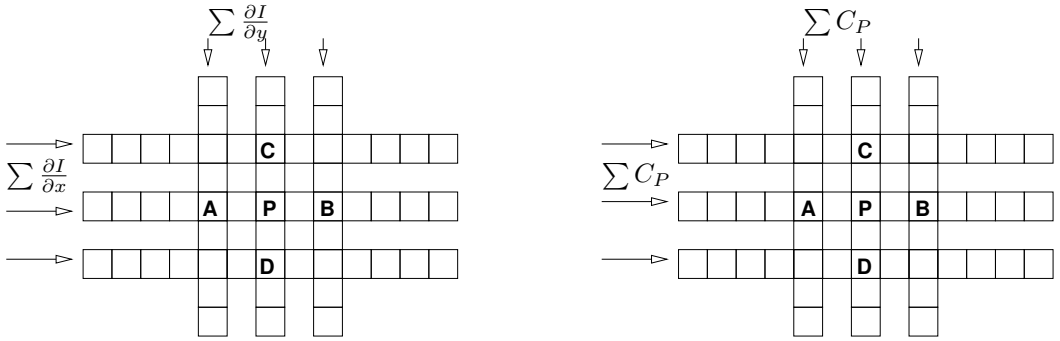


Figure 4.4: The features for each pixel P are calculated by directional accumulation of gradients and propagated corners at P and P's neighbor pixels A, B, C and D.

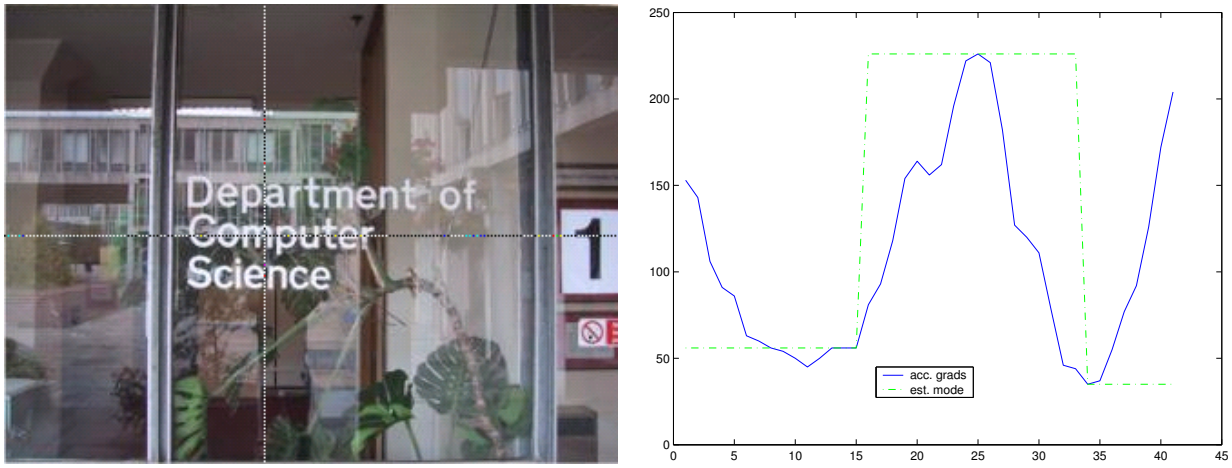
# Values	Feature
3	First derivative $\frac{\partial I}{\partial x}$, horizontally accumulated at pixels C, P and D.
3	First derivative $\frac{\partial I}{\partial y}$, vertically accumulated at pixels A, P, B.
3	Propagated corner response C_P, horizontally accumulated at pixels C, P and D.
3	Propagated corner response C_P, vertically accumulated at pixels A, P, B.
12	Total

Table 4.1: The contents of a feature vector of type I for a pixel P at one level of the pyramid.

direction around three different points: the pixel P itself and the two neighbors A and B, or C and D, depending on the direction of the accumulation. In order to introduce more information about the spatial distribution of the feature values into the feature vector, we chose neighbors with a distance of two pixels to the given pixel P.

4.2 Text height estimation

In contrast to the previous approach, which collects evidence for a baseline presence directly in the grayscale image by accumulating corner responses, in the second method we measure the baseline as the boundary of a high density area in an image which is the result of a first, rough text detection. Detecting the baseline explicitly, i.e. by using a Hough transform or similar techniques, is inherently difficult due to the irregularity of the boundary and the possibility of very short text. Furthermore, the boundaries between the text region and the non-text background may be fuzzy, especially if the text orientation does not totally coincide with the text filter direction (e.g. if a text with 15° slope needs to be detected with a horizontal filter). Therefore, the direct detection



(a) (b)

Figure 4.5: Searching the mode in an interval (parameter $T = 41$): (a) the original image; (b) the vertical interval of gradients and the estimated mode.

of lines and contours is rather difficult. Instead we detect, at each pixel, the height of the eventually associated text rectangle, and check for differences in these heights across a neighborhood. In text areas, the text height should remain approximately constant. Using the height instead of the vertical position of the rectangle is more robust, especially to rotations of the text.

We apply the text filter using horizontally accumulated gradients to create a first filter response. The rectangle height is computed from the vertical profile of the filter response, i.e. at each pixel a vertical interval of T pixels centered on the given pixel is considered. In this interval, we search for a mode (peak) containing the center pixel. Figure 4.5a shows an example image whose rectangle height is estimated at the pixel position indicated by the cross hair. Figure 4.5b shows the vertical interval of accumulated horizontal gradients with a visible peak in the center region of the interval (the center value corresponds to the examined pixel). The dashed-dotted line shows the estimated borders of the mode.

Mode estimation

The peak may be more or less flat, depending on the orientation of the text. For the horizontal filter, horizontal text creates a rectangular shaped peak, whereas slightly rotated text creates a flatter, more trapezoidal peak. The areas of the interval which are outside of the mode, are undefined.

The detection of peaks has already been tackled in the framework of edge detection. In contrast to classical step edges, “roof” edges or “ridge” edges are peak shaped functions. For instance, Ziou presents a solution for the detection of roof edges [93]

which is derived from Canny's criteria of localization quality and detection quality [10]. These solutions are based on linear filtering with infinite impulse response followed by a maximum search in the filter response. In our case, we are not so much interested in the localization of the maximum of the peak, which may be anywhere in the text rectangle, but in the localization of the peak borders. The peak may either be situated in a flat environment, if the respective rectangle lies in an unstructured background, or neighbor other peaks, if the text is part of several lines. Therefore, a simple model of the mode as a roof function with added white noise is not possible.

Instead of trying to find the closed form of an optimal filter kernel for a linear filter, we posed the peak detection as optimization problem over the space of possible peak borders. We exploit various properties of the situation at hand, based on the following assumptions:

- A high filter response inside the mode.
- A high contrast to the rest of the profile, i.e. the difference between the maximum of the mode and the values at the borders should be high.
- The size of the mode, which corresponds to the height of the text rectangle, needs to be as small as possible, in order to avoid the detection of multiple neighboring text rectangles.

Given an interval of N values $G_1 \dots G_N$ where the center value $G_{N/2}$ corresponds to the pixel to evaluate, the following values are possible for the mode borders a and b: $a \in [1, N/2-1]$, $b \in [N/2+1, N]$. The border values are estimated maximizing the following criterion:

$$(a,b) = \arg\max_{a,b} \left[\alpha_1 \left(1 - \frac{width}{N}\right) + \alpha_2 \frac{height}{\max_i(G_i) - \min_i(G_i)} + \alpha_3 \frac{\mu(G_i)}{\max_i(G_i)} \right]$$

where $width = b - a + 1$ is the width of the mode (i.e. height of the text rectangle),

$$height = \left[\max_{i \in [a+1, b-1]} (G_i) \right] - \frac{1}{2}(G_a + G_b)$$

is the height of the mode (i.e. the contrast of the text rectangle to its neighborhood), $\mu(G_i),\ i \in [a,b]$ is the mean of the mode values and the $\alpha_j,\ j \in 1..3$ are weights. The criterion searches for a mode with large height and mean but small width. The weights have been experimentally set to the following values:

$$\alpha_1 = 0.25 \quad \alpha_2 = 1.0 \quad \alpha_3 = 0.5$$

These weights emphasize the height criterion and still allow the width criterion to avoid the detection of multiple modes.

Note, that the three mode properties — height, width and mean — are combined additively instead of multiplicatively, as the criteria proposed by Canny. Multiplication favors configurations where all three properties are high. However, in our case we wanted to put the emphasis on high mode height, which is the main characteristic of text modes. The other two properties, width and mean, have been added to further increase the performance and to restrict the choice of borders to a single peak. This choice to combine the mode properties may be compared to the combination of multiple experts (see section 7.2.2). In the case where some of the experts are weakly trained, i.e. less reliable than others, the sum rule of classifier combination is more powerful than the product rule.

Text height regularity

Once the mode is estimated, we can extract a number of features which we already used for its detection: width, height, mean, etc. Combining the properties of the modes of several neighboring pixels, we are able to extract features on a larger spatial neighborhood, which is schematically shown in Figure 4.6. Figure 4.6b shows the accumulated gradients of an example image. As already mentioned, around each pixel we want to classify, a vertical interval of accumulated gradients is considered and the mode in this interval is estimated. Then the variability of the mode width across horizontally neighboring pixels is verified (figure 4.6c) by calculating the difference in mode width between neighbors and accumulating this difference across a horizontal window of size S_w, where S_w is a parameter which depends on the text size:

$$\Delta W(x,y) = \sum_{i=-\lfloor S_w/2 \rfloor}^{\lfloor S_w/2 \rfloor} |W(x+i-1,y) - W(x+i,y)| \tag{4.1}$$

where $\Delta W(x,y)$ is the accumulated difference in mode width (which we call baseline presence) at pixel (x,y) and $W(x,y)$ is the mode width at pixel (x,y).

Figure 4.7 shows an example image and the resulting feature images, where the feature values are scaled for each feature independently into the interval $[0,255]$ in order to be able to display them as gray values, and the negative image is displayed. Note, that the mode width is approximately uniform in the text areas, which results in low accumulated mode width differences in this area, displayed as close to white in the respective feature image.

The final seven feature values corresponding to a single orientation are listed in table 4.2. In contrast to the features proposed in section 4.1, the mode estimation features are not rotation invariant, although they are robust to slight changes up to 25^o. Therefore, we calculate the features for the four principal orientations of the image $\{X,Y,R,L\}$ resulting in a $7 \cdot 4 = 28$ dimensional feature vector.

For speed reasons, the classification is done on every 16^{th} pixel only, i.e. on every 4^{th} pixel in x direction and every 4^{th} pixel in y direction. The classification decisions for the other pixels are bi-linearly interpolated.

(a)

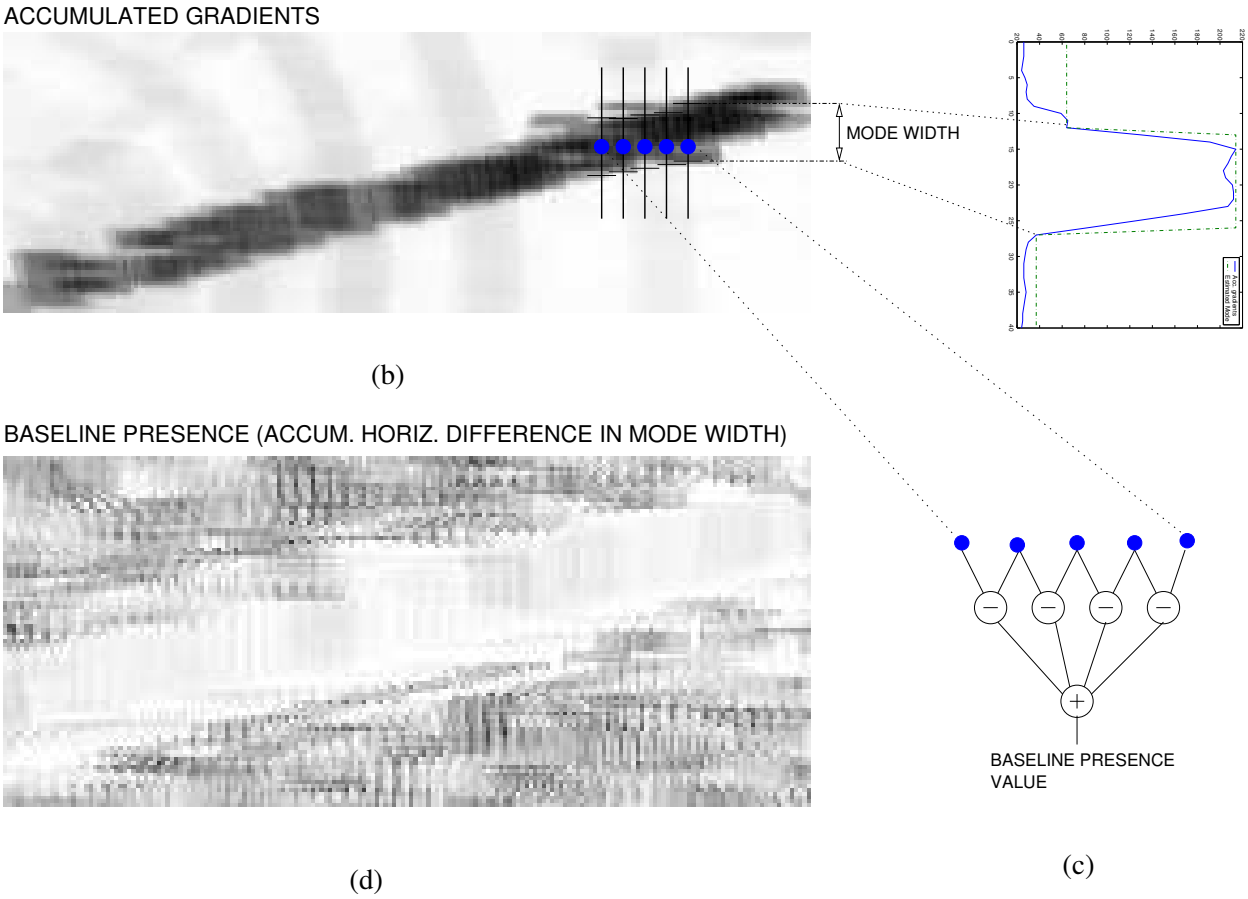


Figure 4.6: Combining the mode features of neighboring pixels: (a) the original image; (b) estimating the mode at different pixels; (c) calculating the accumulated difference of mode widths; (d) the baseline presence image.

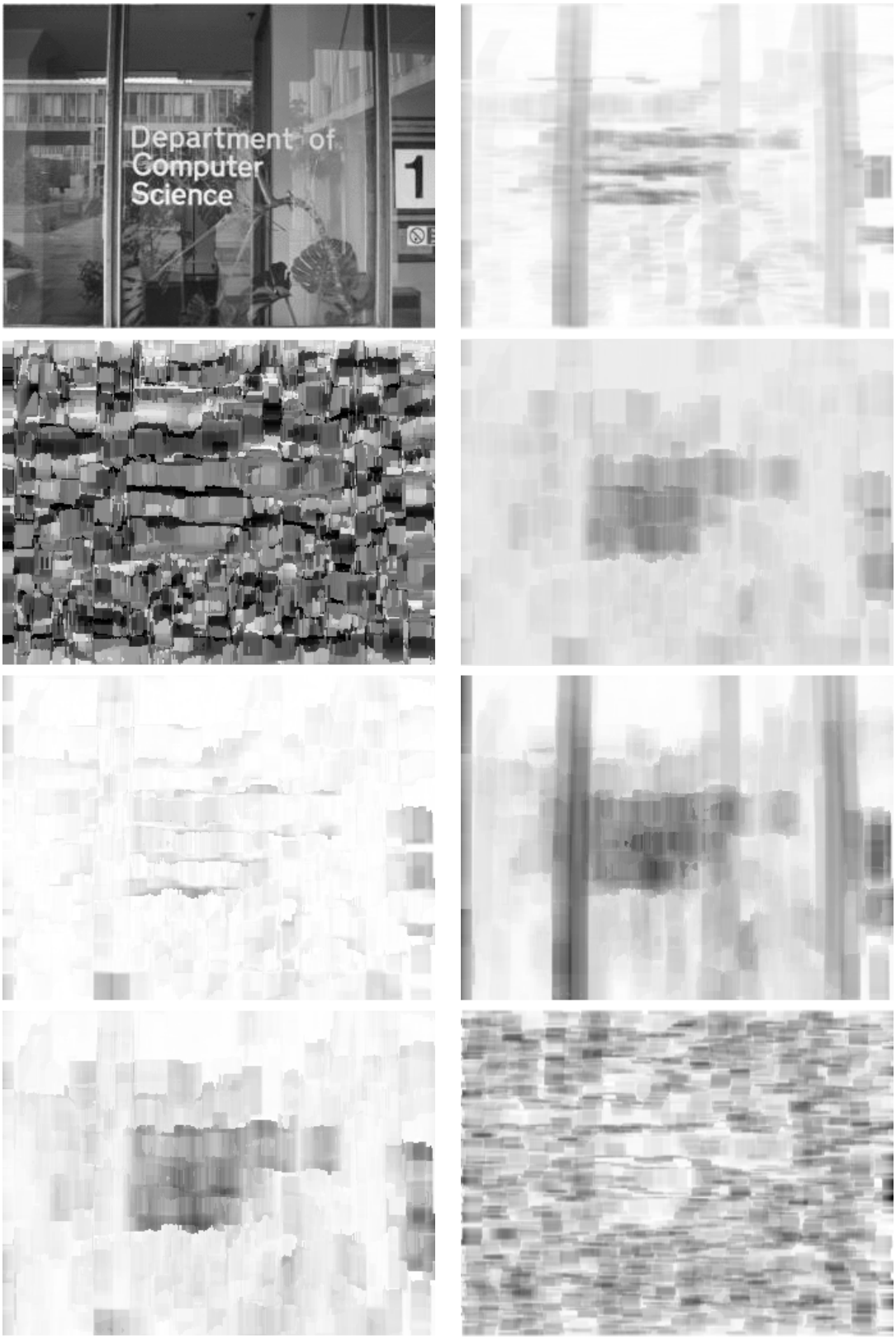


Figure 4.7: From left to right and top to bottom: The original image, the accumulated gradients (darker→ better), mode width, mode height (darker→ better), difference of mode heights (brighter→ better), mean (darker→ better), standard deviation (darker→ better), baseline presence (brighter→ better).

# Values	Feature
1	Horizontally accumulated first derivative $\frac{\partial I}{\partial x}$.
1	The width of the detected mode.
1	The height of the detected mode.
1	The difference of the heights of the mode to the left and to the right border.
1	The mean of the gradient values in the mode.
1	The standard deviation of the gradient values in the mode.
1	The baseline presence (accumulated differences of the mode widths across several modes in a horizontal neighborhood).
7	Total per orientation
28	Total

Table 4.2: The contents of a feature vector of type II.

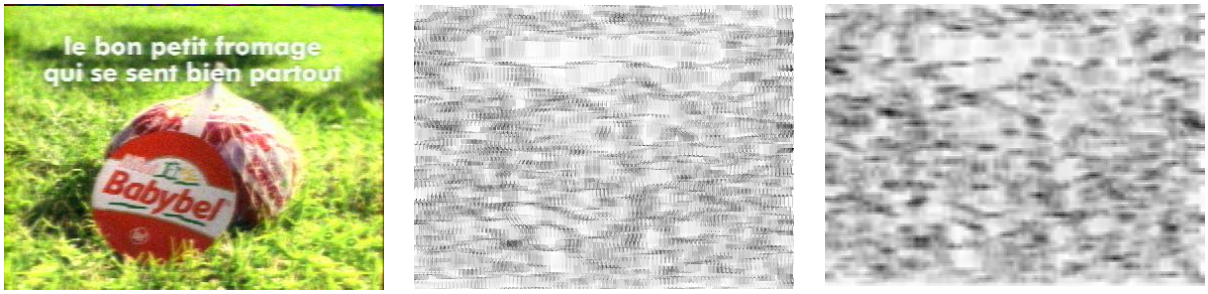
The most complex step during the feature calculation phase is the mode estimation. It is possible to perform this calculation only on pixels which are evaluated later in the classification phase, which reduces the complexity tremendously (see table 4.9). However, the mode properties are reused to calculate the baseline presence feature (equation 4.1), so the calculation of this feature needs to be changed since the mode properties of the immediate neighbors of a pixel are not available anymore. Instead, the differences in mode width between the nearest horizontally neighbors with available mode properties are accumulated:

$$\Delta W(x,y) = \sum_{i=-\lfloor S_w/2 \rfloor}^{\lfloor S_w/2 \rfloor} |W(x+4(i-1),y) - W(x+4i,y)|$$

where of course the length parameter S_w needs to be adapted to the new situation. Figure 4.8 shows an example image and the baseline presence feature image with feature calculation on every pixel (4.8b) and with feature calculation on every 16^{th} pixel and bi-linear interpolation of the other pixels (4.8c). As can be seen, no significant difference can be noted between the two feature images.

4.3 From classification to detection

The last two sections introduced two different ways to calculate features, i.e. to translate a pixel and its local neighborhood into a descriptive feature vector. The feature vectors of a training set are fed into a learning machine in order to learn the differences between the features of text and non-text, as we will explain in section 4.4. In this section we describe how the classification of the pixels is carried out on images and how it is translated into detection results.



(a) (b) (c)

Figure 4.8: (a) example image; (b) the baseline presence with feature calculation at every pixel; (c) the baseline presence with feature calculation at every 16^{th} pixel only.

Type	Dim. per level	Total dimension	Description
I	12	24	Corner propagation
II	28	56	Text height estimation

Table 4.3: The total dimensions of the feature vector for two levels of the pyramid.

In order to be able to detect text of various sizes in images of various sizes, the detection algorithm is performed in a hierarchical framework. Each input image forms the base of a Gaussian pyramid, whose height is determined by the image size. The classical detection approaches apply a classification step at each level of the pyramid and collapses the pyramid by combining these intermediate results (see, e.g. [80]).

The combination of the results of different levels is a difficult problem. Boolean functions as *AND* and *OR* have drawbacks: The *OR* function creates a response for each response on one of the levels of the pyramid, therefore large images with high pyramids tend to create many responses and many false alarms. The *AND* function only responds if the text is detected at all levels, and therefore eliminates the advantages of a hierarchical solution. We decided to partly delegate this problem to the learning machine by creating feature vectors which contain features taken from two levels. Figure 4.9 illustrates the principle: a Gaussian pyramid is built for the image as already stated above. As done in the classical approaches, classification is done at each level of the pyramid for each pixel of the image. However, each feature vector is doubled: it contains the features calculated on the level itself as well as the features calculated for the central parent pixel. Therefore the dimensions of the feature vectors are doubled (see table 4.3).

Since a single feature vector now contains the features from two levels, the classification decisions are more robust to changes of text size. Text, which is normally detected at a certain level l, will also be detected on a lower level $l-1$ with a higher probability, since the features for level l are included in the feature vector for level $l-1$.

Of course, the fact that we create vectors with features from two different levels does

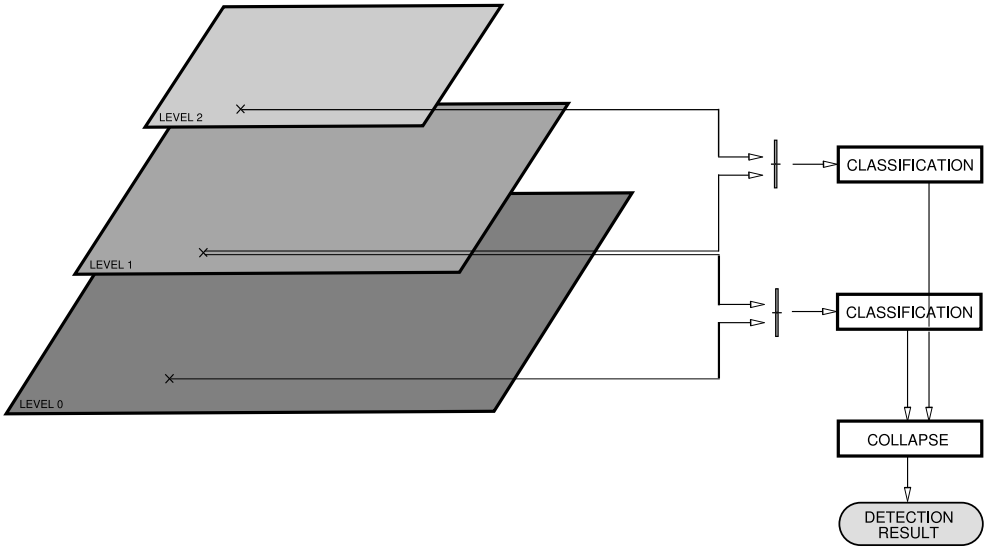


Figure 4.9: The pyramid structure of the algorithm.

not solve the problem of finding a way to collapse the hierarchical decisions into a flat result. We chose a simple OR function for the combination of the results of the different levels. However, we do not combine the results on pixel level but on rectangle level. Hence, the detection of text on one level of the pyramid involves the following steps:

- Calculation of the features on each pixel of the level, using the neighborhood of the pixel itself as well as the neighborhood of the central parent pixel.
- Classification of the feature vector, i.e. of the pixel, using the learned model.
- Post processing of the decision image and extraction of text rectangles as bounding boxes of connected components. We perform the morphological and geometrical post-processing already described in sections 3.2 and 3.3.

The hierarchical detection and post-processing results in a list of detected rectangles per pyramid level. The final list of rectangles consists of the union of all rectangles, where each rectangle is projected to the base of the pyramid in order to normalize their sizes and positions. Overlapping and touching rectangles are merged according to the rules described in section 3.3.

4.4 Learning

The feature vectors given in the previous section have been designed to distinguish single pixels between text and non-text. It is the task of learning machines to learn this distinction from training data, i.e. from a set of positive and negative examples.

From the large pool of existing learning machines, we chose support vector machines (SVM) for the task of classification between the two classes. Lately, they received tremendous attention in the learning community and have been applied successfully to a large class of pattern recognition problems, where they performed better then competing techniques, e.g. artificial neural networks.

A major advantage of SVMs is their smaller sensitivity to the number of dimensions of the feature space (the curse of dimensionality), which hurts the performance of neural networks. Indeed, whereas a reduction of the dimensionality of the feature space with tools as e.g. the principal components analysis is crucial when "traditional" learning techniques are employed, learning with SVM often does not require this reduction.

4.4.1 Support Vector Machines

Support Vector Machine learning has been introduced by Vapnik to tackle the problem of learning with small data sets. The interesting point which distinguishes SVM learning from classical neural network learning is the fact, that SVMs try to minimize the generalization error using a principle called the structural risk minimization. A detailed introduction would be too long for this thesis, the interested reader is referred to [9] for a tutorial on SVM learning or Vapnik's books for a detailed treatment of the theory [77][78].

Linear SVM learning separates a dataset consisting of 2 classes by a hyperplane and maximizes the margin, i.e. the sum of the two distances from the hyperplane to the nearest vector of the respective classes. The vectors whose removal would change the solution are called support vectors.

The well known decision function for a linear classifier for data points of dimension d is given by

$$f(\boldsymbol{x}) = (\boldsymbol{w} \cdot \boldsymbol{x}) + b$$

where $\boldsymbol{x} \in \boldsymbol{R}^d$ is the data point to classify, $\boldsymbol{w}$ is the normal vector of the separating hyper plane and

$$\frac{b}{||\boldsymbol{w}||} \tag{4.2}$$

is the distance between the plane and the origin. As we can see, there is a degree of liberty in this equation: scaling the normal vector $\boldsymbol{w}$ and changing b accordingly leaves the function unchanged. We cope with this by a normalization: by definition, the vectors nearest to the hyperplane evaluate to

$$(\boldsymbol{w} \cdot \boldsymbol{x}) + b = \pm 1$$

where the function evaluation to $+1$ for positive class labels and -1 for negative class labels. Then, given the points $\{(\boldsymbol{x}_i, y_i)\},\ y_i \in \{-1, +1\}$ of a training set, where y_i is

the class label, the plane maximizing the margin (4.2) between the two classes can be found minimizing the norm of the normal vector

$$||\boldsymbol{w}||^2/2$$

under the constraints:

$$\begin{aligned} \boldsymbol{x}_i \cdot \boldsymbol{w} + b \geq +1 \quad \text{for} \quad y_i = +1 \\ \boldsymbol{x}_i \cdot \boldsymbol{w} + b \leq -1 \quad \text{for} \quad y_i = -1 \end{aligned}$$

The constraints are imposed by the fact, that the two classes need to separated linearly without outliers. The problem is unsolvable, if the dataset is not linearly separable. Since most real world problems are not linearly separable, slack variables ξ_i are introduced, which compensate for data points which violate the margin. Therefore, the constraints have to be changed to:

$$\begin{aligned} \boldsymbol{x}_i \cdot \boldsymbol{w} + b \geq +1 - \xi_i \quad \text{for} \quad y_i = +1 \\ \boldsymbol{x}_i \cdot \boldsymbol{w} + b \leq -1 + \xi_i \quad \text{for} \quad y_i = -1 \end{aligned}$$

The usage values of the slack variables is punished with a new objective function:

$$||\boldsymbol{w}||^2/2 + C \sum_i \xi_i \tag{4.3}$$

where C is a parameter which controls the amount of punishment for data points which violate the margin.

Until now, only linear problems have been treated. SVMs treat non-linear problems in a different way then neural networks, which create non-linear decision functions. SVMs keep linear hyper planes as decision functions but apply a non-linear function $\Phi(\boldsymbol{x})$ in order to project the data in a — possibly very high dimensional — space and try to separate it linearly in this space:

$$\boldsymbol{x} \rightarrow \Phi(\boldsymbol{x})$$

Due to a well known mathematical tool called "kernel trick", the function $\Phi(\boldsymbol{x})$ does not need to be known. We exploit the fact that the only calculations performed with the data points are dot products. In order to avoid calculations in the high dimensional feature space, a kernel function in the original data space is developed, which corresponds to an inner product in the feature space:

$$K(\boldsymbol{x}, \boldsymbol{y}) = \Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{y})$$

On the other hand, we now cannot directly access the normal vector of the separating hyperplane. The decision function for the classifier is given as:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N_s} \alpha_i y_i K(\boldsymbol{s_i}, \boldsymbol{x}) + b \tag{4.4}$$

Description	Kernel function
Polynomial kernel	$K(\boldsymbol{x}, \boldsymbol{y}) = \gamma + (\boldsymbol{x} \cdot \boldsymbol{y})^p$
Radial Basis Function (RBF) kernel	$K(\boldsymbol{x}, \boldsymbol{y}) = exp(-\frac{1}{2\sigma^2}\|\|\boldsymbol{x} - \boldsymbol{y}\|\|^2)$
Sigmoid kernel	$K(\boldsymbol{x}, \boldsymbol{y}) = tanh(\boldsymbol{x} \cdot \boldsymbol{y} - \Theta)$

Table 4.4: Some widely used SVM kernel functions.

where $\boldsymbol{s_i},\ i = 1..N_s$ are the support vectors.

The choice of the kernel function and its eventual parameters determines the structure of the feature space and therefore the class of possible decision functions. Unfortunately, the issue of choosing the right kernel function for a given problem is an open research problem. The classical solution is to use a set of several well known kernels and determine the right one and its eventual parameters by cross validation. Different known kernel functions are respectively, the polynomial kernel, the radial basis function kernel and the sigmoid kernel (see table 4.4).

During the learning process, apart from the kernel type and the kernel parameters, another parameter needs to be set: the parameter C from the objective function (4.3). It controls the contribution of the slack variables to the objective function and therefore the "smoothness" of the decision function.

In this work, we conducted ν-SVM learning[65] instead of $C-$SVM learning. In ν-SVM learning, the training parameter C, which depends on the dimension of the problem and the size of the training set, is replaced by a a new parameter $\nu \in [0, 1]$. This parameter is independent of the size of the data set, which allows to estimate it on a smaller training set before training the classifier on a large training set.

The drawback of the SVM approach is the high computational complexity in the test phase, which becomes clear looking at the equation of the decision function (4.4). The distance between the vector $\boldsymbol{x}$ and the hyperplane invokes a sum over all support vectors. The complexity of the classification is therefore $O(d \cdot N_s)$. We will address this problem in the next section.

4.4.2 Reducing the complexity

Support vector machines are currently significantly slower than learning machines with a similar generalization performance. As already indicated above, the complexity of the classification process is proportional to the dimension of the problem and the number of support vectors in the model. Since the dimension of the problem cannot always be changed easily, the key to the reduction of the computational complexity is a reduction of the number of support vectors.

Different methods exist in literature for this purpose. Burges proposes the minimization of the quadratic error of the initial hyper plane and a new one with a — fixed — lower amount of vectors which are not necessarily data vectors [8]. The method has the disadvantage of being difficult to implement. Osuna and Girosi propose the usage

of SVM regression to approximate the original hyper plane with a new function with less support vectors [58]. We used this approach for the reduction of our model. The algorithm to reduce the model can be outlined as follows:

1. Train the full classification model with a given kernel and parameters.
2. Run epsilon support vector machine regression (SVMR) on the hyper plane evaluated at the support vectors, i.e. $(\boldsymbol{s_i}, f(\boldsymbol{s_i}))$. This results in a new hyper plane with fewer support vectors.

The parameters of the SVMR algorithm are C (the sum of the slack-variables as in C-SVM classification) and the ϵ for Vapnik's ϵ-insensitive cost function defined as:

$$|x|_\epsilon = \begin{cases} 0 & \text{if } |x| < \epsilon \\ |x| - \epsilon & \text{else} \end{cases}$$

The two parameters define the accuracy of the approximation and therefore the performance of the reduced classifier and the number of support vectors.

4.4.3 The training algorithm

The learning machine as described above needs a training set with positive and negative samples in order to learn to model. In the case of object detection problems, and as a special case text detection problems, the positive examples are not hard to determine. On the other hand, which samples shall be chosen as negative samples? Practically speaking, the size of the training set is limited since the complexity of the training algorithm grows non-linearly with the number of the training samples. Hence, the negative samples need to be chosen wisely in order to represent the class of non-text as closely and completely as possible.

To tackle this problem, we employed the well known bootstrapping approach for the selection of the training set, i.e. the negative samples depend on the positive ones and are chosen in an iterative algorithm as follows:

1. The initial training set consists of the positive training sample set T_P and $\frac{1}{K}|T_P|$ randomly chosen vectors from a large set of negative samples T_N, where K is the number of bootstrap iterations.
2. Training is performed on the training set.
3. The returned model is applied to the rest of the negative samples and the correctly classified samples are removed from this set. From the remaining set, the $\frac{1}{K}|T_P|$ samples with the smallest distance to the separating hyperplane are added to the training set.
4. Go to step 2 until the number of iterations has been performed.

During the training phase, we perform another iterative algorithm: We employ n-fold cross validation in order to estimate the generalization error of the model we obtained by the learning process. The two iterative processes, bootstrapping and N-fold cross validation, are combined into a single training algorithm as follows:

1. Create two sets of training samples: a set T_P of positive samples and a large set T_N of negative samples.
2. Shuffle the two sets and partition each set into N distinctive and non-zero subsets of sizes $\frac{1}{N}|T_P|$ and $\frac{1}{N}|T_N|$, respectively, where N is the parameter from the N-fold cross validation.
3. $i = 1$.
4. For each of the two sets, choose subset i.
5. Train the SVM on the two sets using the iterative bootstrapping method described above.
6. Test the model on the samples not chosen in step 4 and calculate the error.
7. $i = i + 1$.
8. Go to step 4 until the number of iterations is reached (i.e. i=N).

At each iteration, $\frac{N-1}{N}(|T_P| + |T_N|)$ samples are used for training and $\frac{1}{N}(|T_P| + |T_N|)$ samples are used for testing. The final generalization error is computed as the mean of the errors computed in step 6 over all iterations.

4.5 Experimental results

As already stated, the proposed method consists of a classification step and post-processing step (morphological and geometrical post-processing and combination of the levels of the pyramid). Hence, the experiments and the evaluation of the detection algorithm need to be conducted on two different levels:

- We performed an evaluation on pixel level, i.e. feature vector level, which evaluates the discrimination performance of the features and the performance of the classifier. We also based the choice of the learning parameter of the learning machine and the choice of the kernel and its parameters on the results of this evaluation, in order to keep the choice independent of the parameters of the post-processing step.
- An evaluation on text rectangle level has been done. The evaluation criteria we used are the same as the ones developed for the detection method using local contrast (see section 3.4.1).

4.5.1 Classification performance

The estimation of the classification error with 5-fold cross validation helped us to determine various choices:

- The choice between the two feature types, type I (the propagated corner responses) and type II (the estimated text heights);
- The choice of the kernel;
- The kernel parameter;
- The learning parameter ν;
- The reduction parameters C and ϵ for the reduction of the number of support vectors;

For speed reasons, we carried out the evaluation on two training sets of different sizes. For the selection of the kernel, its parameters and the training parameters, we chose for each feature type 3,000 vectors corresponding to text pixels (positive examples) and 100,000 feature vectors corresponding to non-text pixels (negative examples). Once the optimal parameters were selected, we trained the system with 50,000 positive feature vectors and 200,000 negative samples. All learning was done with combined boot strapping (3 iterations) and 5-fold cross validation as described in section 4.4.3, i.e. not all negative samples were actually used in the final training set.

Figure 4.10 illustrates the dependence of the classification performance on the kernel parameters and the learning parameter ν. The figure shows the classification error, the number of support vectors, recall, precision, and the harmonic mean of recall and precision for feature type I as a function of ν and the degree of the polynomial SVM kernel. Please note, that the number of support vectors very much depends on the learning parameter ν. As can be seen from figure 4.10e and 4.10f, the optimal parameters for this feature type are a kernel with degree 3 and a learning parameter of $\nu = 0.45$. For feature type II we determined an optimal polynomial kernel of degree 6 and $\nu = 0.37$.

Table 4.5 shows the classification results compared between the two feature types I & II and the effect of the reduction of the number of support vectors. As expected, the second feature type outperforms the first one by an increase in recall of 8.9 percentage points, although the number of support vectors is lower by 8,244 vectors. As presumed, the geometrical information carried by the second feature type is more discriminative then the one by the first type.

The full, unreduced trained SVM models contain between 22,000 and 30,000 support vectors, depending on the feature type. This amount of support vectors results in very high classification complexity. The classification of an image of size 384×288 pixels takes around 15 minutes on a Pentium III-700 (if only every 4^{th} pixel in each dimension, i.e. every 16^{th} pixel is classified!!). However, by applying the technique described in section 4.4.2, a large reduction of the number of support vectors can be achieved without

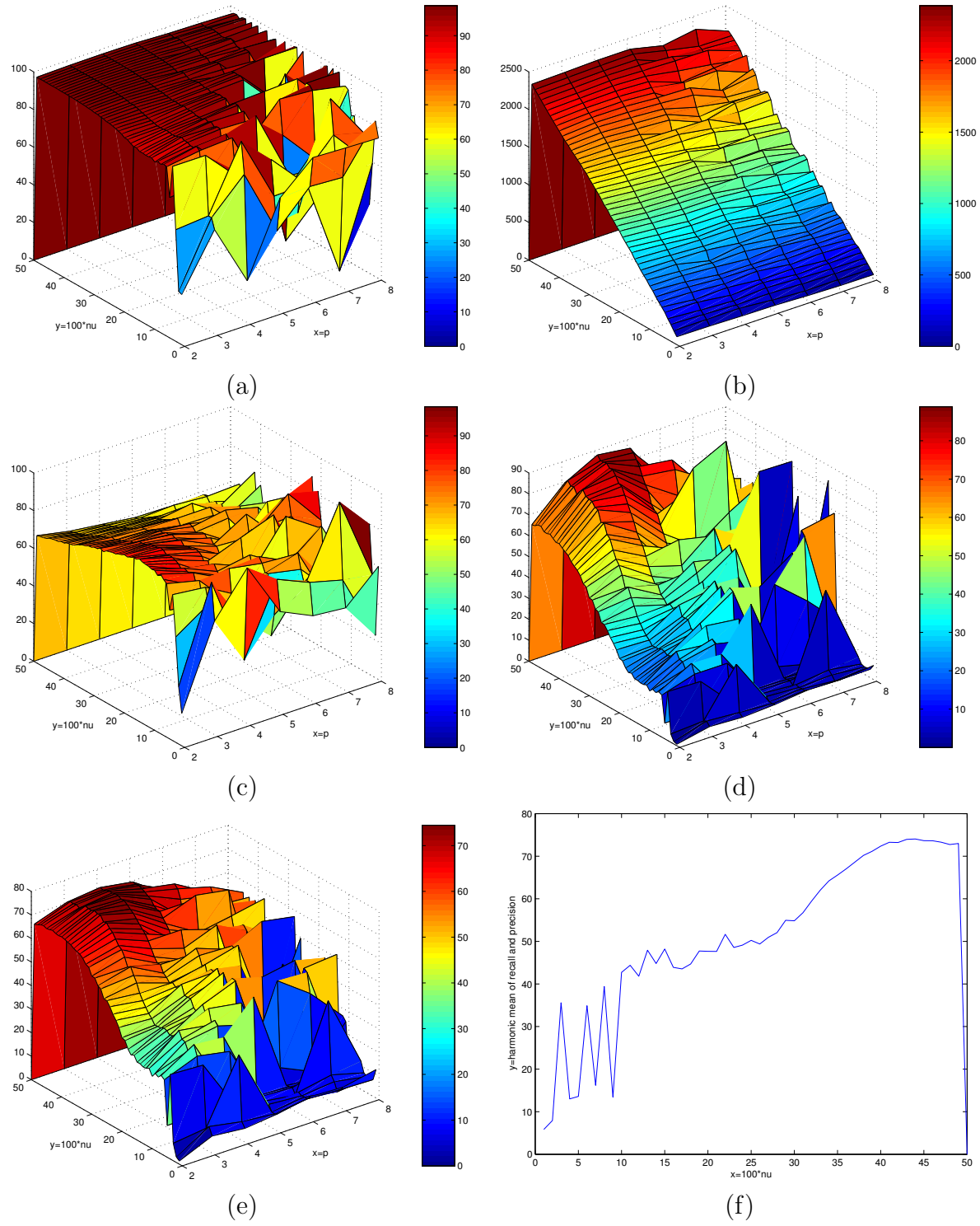


Figure 4.10: Parameter evaluation as a function of ν and the degree p of the polynomial kernel: (a) the classification error; (b) the number of SVs; (c) recall; (d) precision; (e) the harmonic mean of precision and recall; (f) the harmonic mean for a polynomial of degree 3 only.

F.type	C	ϵ	Recall	Precision	H.Mean	#SVs
I	no reduction		72.1	95.5	**82.2**	30344
I	1000	0.01	72.1	95.5	82.2	809
		0.1	72.3	95.1	82.1	337
		1	73.2	91.1	81.2	70
I	10000	0.01	72.1	95.5	82.2	822
		0.1	72.1	95.0	82.0	344
		1	73.3	91.0	81.2	70
II	no reduction		81.0	96.0	**87.9**	22100
II	1000	0.01	80.5	95.9	87.5	2581
		0.1	80.3	95.8	87.4	806
		1	75.5	96.6	84.8	139
II	10000	0.01	80.5	95.9	87.5	2581
		0.1	80.3	95.8	87.4	806
		1	75.5	96.6	84.8	139
Δ	no reduction		8.9	0.5	**5.7**	8244

Table 4.5: Comparison between the two feature types I & II and the effect of the reduction of the number of support vectors on the classification performance. The row Δ contains the difference in performance between the two feature types.

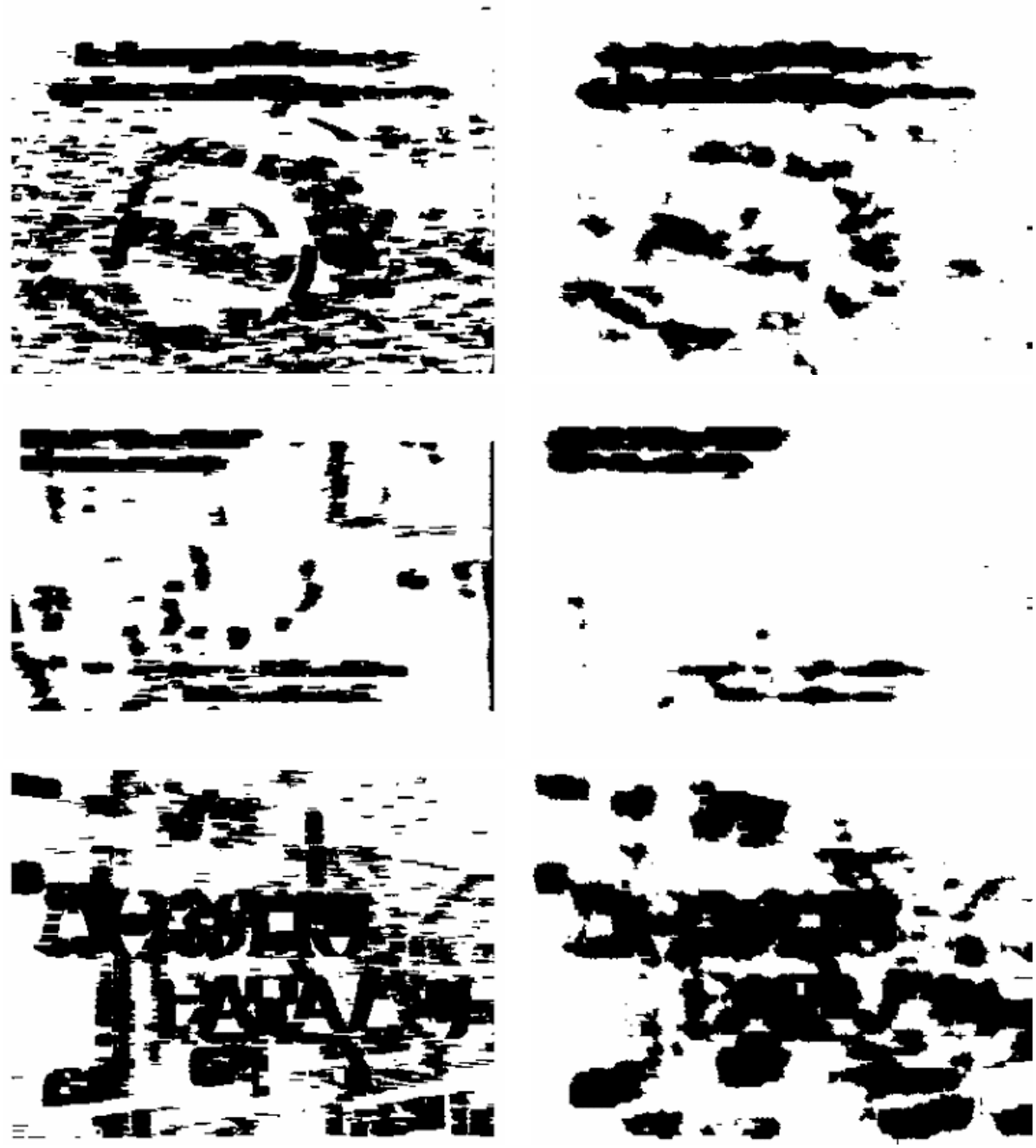
losing any classification performance. For feature type II, a reduction from 22,100 support vectors to 806 support vectors decreases the classification performance only by 0.5 percentage points (table 4.5). The tradeoff between the number of support vectors and the classification performance can be controlled conveniently by tuning the parameter ϵ of the model reduction algorithm. More information on the run-time complexity of the classification algorithm for different models with numbers of support vectors is given in section 4.5.3.

The classification performance figures given above have been achieved with mode features which are calculated for each pixel. However, as explained in section 4.2 and shown in figure 4.8, the calculation may be accelerated by calculating the features on

C	ϵ	Recall	Precision	H.Mean	#SVs
no reduction		84.0	93.2	88.4	19058
100	1.0	**78.3**	**93.0**	**85.0**	**610**
100	1.5	69.3	93.7	79.7	236

Table 4.6: The classification performance with partial calculation of the features only (each 16^{th} pixel). The model reduced with parameters $C = 100$ and $\epsilon = 1.0$ is the final model used in our experiments.

(a)



(b)

Figure 4.11: Classification results: (a) three example images; (b) result images using feature type II (right) compared to the results using the non-learning method presented in section 3 (left).

each 4^{th} pixel in each dimension only. The classification results for this calculation mode are given in table 4.6. The final model we used throughout this thesis is the one reduced with parameters $C = 100$ and $\epsilon = 1.0$ resulting in 610 support vectors.

Finally, figure 4.11 shows three result images and the text detection results — without post processing — for the method using local contrast only and the new learning base method. The result images are calculated on the original scale of the image (as opposed to the feature vectors, which are calculated on two scales). As can be seen, the new method produces far less false alarms, which can be explained by the discriminative power of the new features.

4.5.2 Detection performance

Figures 4.13 and 4.14 show the performance of the learning based detection system on, respectively, 15 images containing text and 15 images not containing any text. These images are the same 30 images which already have been used in chapter 3 for the illustration of the performance of the local contrast based method.

The two sets of images illustrate the difference in performance characteristics between the new method and the method presented in chapter 3: at the first sight we remark that detection precision has been significantly improved on the images which do not contain text. Even the images which do contain text result in less false alarms. However, we can also see that the bounding boxes of the detected rectangles are slightly larger then the ones produced by the local contrast based method, although the detection stays very well within the limits we can expect from a detection system[2]. Furthermore, detection recall is visibly lower then the recall of the local contrast method. The concerned text is mostly text with low contrast, and some rectangle borders seem to be affected. More research in the training data and special post-processing to correct the border problem could solve these issues.

The first two segments of table 4.7 give the performance measures of the images in figures 4.13 and 4.14. As expected, compared to the local contrast method, the recall measures dropped. However, the large difference measured by the CRISP evaluation method is also due to the fact, that most partly detected rectangles are not counted at all. This also holds for the precision measure. While the learning method visibly produces less false alarms, its precision measure is lower then the one produced by the local contrast method. This can be explained by the fact, that partially detected rectangles count as false alarms and thus lower precision. Looking at the ICDAR evaluation scheme, we remark that the precision increased from 27.5% for the local contrast based method to 32.4% for the learning method.

The lower two segments of table 4.7 give the figures for the whole dataset. We observe, that with the learning method we achieve a slight gain in precision (from 20.1%

[2]These limits of precision are hard to define. A goal oriented definition could be possible, which takes into account the response of an OCR system, but this approach is impossible if we are interested in the location of the detected text only. We limit ourselves to a subjective visual judgement of precision, in addition to the given unthresholded precision values, of course.

(a) (b)

(c) (d)

Figure 4.12: Examples for the classification with partial calculation of the features only (each 16^{th} pixel): (a) example image; (b) classification with the full model; (c) classification with a reduced model ($C = 100$, $\epsilon = 1.0$). This model is the final one we chose for the following experiments; (d) classification with a reduced model ($C = 100$, $\epsilon = 1.5$).

Figure 4.13: Some detection examples on images with text.

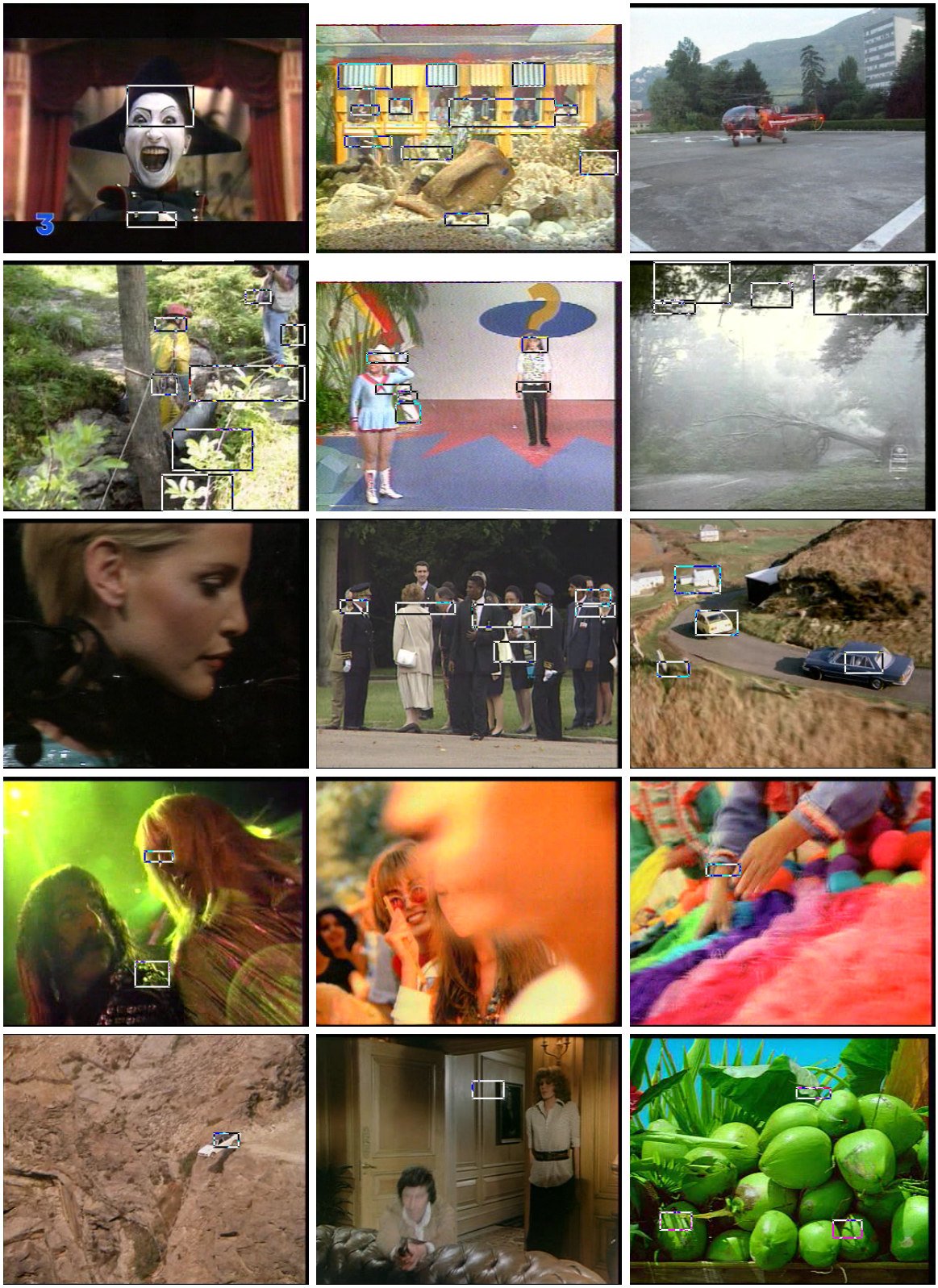


Figure 4.14: Some detection examples on images without text.

Dataset	#†	G††	Eval. scheme	Recall	Precision	H. mean
Figure 4.13:	15	6.20	ICDAR	44.0	54.6	48.7
Artificial text			CRISP	49.5	50.9	50.2
			AREA	59.1	43.7	50.2
Figures 4.13+4.14:	30	3.10	ICDAR	44.0	32.4	37.3
Artificial text +			CRISP	49.5	30.2	37.5
no text			AREA	59.1	35.9	44.7
Artificial text +	144	1.49	ICDAR	54.8	23.2	32.6
no text			CRISP	59.7	23.9	34.2
			AREA	68.8	25.5	37.3
Artificial text +	384	1.84	ICDAR	45.1	21.7	29.3
scene text +			CRISP	47.5	21.5	29.6
no text			AREA	53.6	24.1	33.3

†Number of images

††Generality

Table 4.7: The detection results of the learning based method for different datasets and different evaluation schemes.

→ 23.9%) with the tradeoff of a quite large drop in recall (from 81.% → 59.7%). The tradeoff between the gap in recall and the gain in precision depends on the generality of the database. We will see in chapter 5 how this tradeoff affects the detection performance in video sequences, which are usually characterized by a very low generality.

The ICDAR text locating competition

We participated with the learning based method at the text locating competition organized in the framework of the International Conference on Document Analysis and Recognition 2003 (ICDAR). A description of the competition can be found in [46].

We participated at the competition, although the image dataset it used was clearly different from the image dataset we had in mind when we designed our systems: while we aimed at video frames with low resolution and a high noise level, the ICDAR image dataset consisted of images taken with digital photo cameras, mostly with a resolution of 1600×1200 pixels. In these images, the text characters are very large, hence algorithms performing full character segmentation are favored (see the state of the art of text detection in section 2.4). Our hierarchical processing system tended to produce 5 to 6 pyramid levels for these big images. Since our detection system has been designed for small and noisy text, we went as far as to ignore the base level of the pyramid in order to increase the precision of the algorithm. In other words, we did not take into account valuable information. As a last handicap, we did not have enough time to train the system on the ICDAR test dataset. Instead, we submitted a model trained with our

own test dataset described in section 3.4.

Table 4.8 gives the competition results published by the organizers in [46]. Four academic researchers participated at the competition: Ashida from the Department of Computer Engineering, Shinshu University; H.W. David from the Institute of Automation, Chinese Academy of Sciences; Todoran from the Department of Computer Science and Logic, University of Amsterdam, Netherlands, and myself. The entry labeled "Full" in the table is a virtual system which for each image returns a single text rectangle which covers the whole input image.

System	Precision (%)	Recall (%)	H. mean (%)	time (s)
Ashida	55	46	50	8.7
HWDavid	44	46	45	0.3
Wolf	**30**	**44**	**35**	**17.0**
Todoran	19	18	18	3
Full	01	06	08	0.2

Table 4.8: The results of the competition organized in the framework of the International Conference of Document Analysis and Recognition (ICDAR).

The influence of generality

The dependence of CRISP precision on the generality of the dataset is given in figures 4.15a and 4.15b for, respectively, the dataset containing artificial text only and the dataset containing both types of text. Comparing these figures to the ones for the local contrast method (figures 3.9a and 3.9b), we remark that the precision/generality curves drop less fast for the SVM based method. Hence, this advantage of this method increases for datasets with lower generality, i.e. less text, which is more representative of real life cases. Again we should be aware of the fact, that the CRISP precision counts partially detected rectangles as false alarms, thus lowering precision.

The influence of the evaluation thresholds

Figures 4.16a and 4.16b illustrate the dependence of the CRISP performance measures for, respectively, the dataset containing artificial text only and the dataset containing both types of text. If we compare these figures to the ones for the local contrast method (figures 3.10a and 3.10b), we observe that the first (left) figures are very similar for both methods, although the curves for the SVM based method are lower. These figures represent the dependence on the "recall" threshold.

The latter (right) figures are a little bit different: the performance curves drop faster after the "standard" threshold on precision has been reached. This basically means that the SVM method tends to produce detected text rectangles which are slightly bigger than the counterparts produced by the local contrast based method.

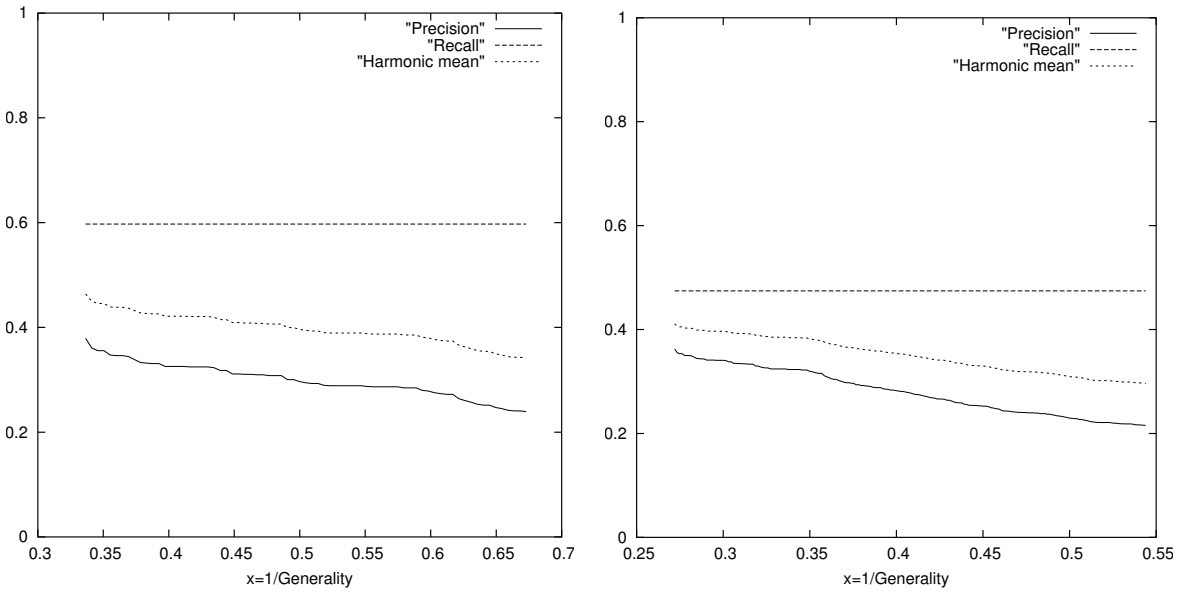


Figure 4.15: Precision for different generalities. Reciprocal generality is displayed on the x-axis: (a) artificial text + no text; (b) artificial text + scene text + no text.

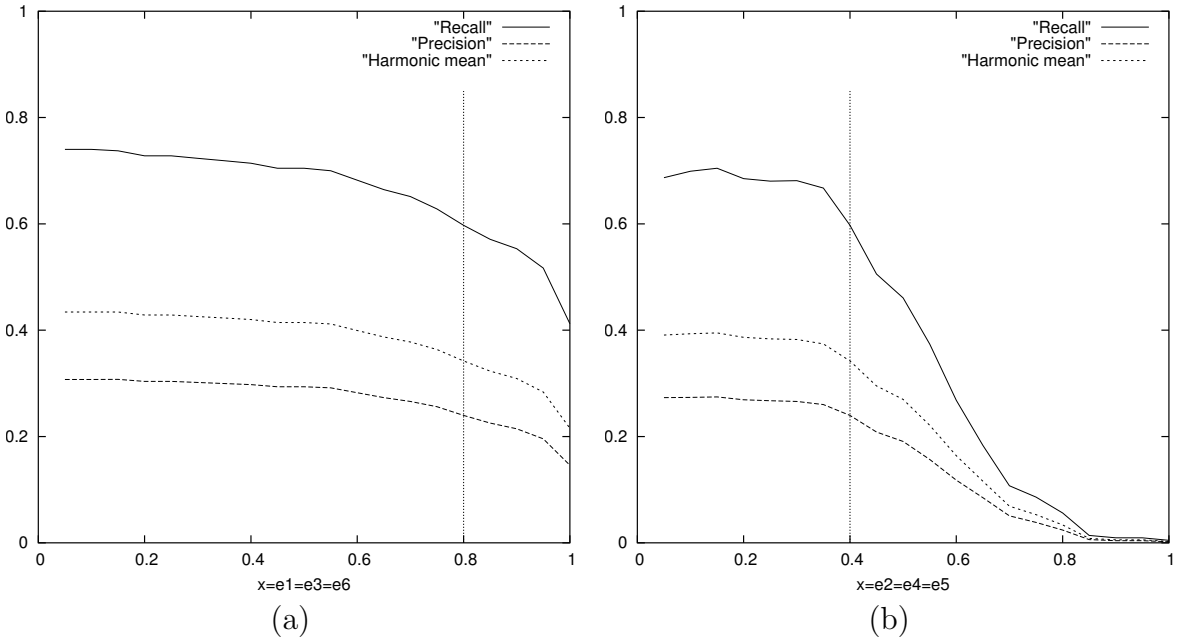


Figure 4.16: The system performance evaluated with the CRISP method for different evaluation thresholds (the vertical bars indicate the threshold chosen throughout this work): (a) changing thresholds e_1, e_3 and e_6; (b) changing thresholds e_2, e_4 and e_5.

4.5.3 Execution time

The execution time of the algorithm implemented in C++ on a Pentium III with 700 Mhz running under Linux is shown in table 4.9. The execution time has been measured for an input image in CIF format, i.e. of size 384×288 pixels. As already mentioned, the classification largely depends on the number of support vectors, which in turn depends on the parameters of the regression algorithm used to approximate the decision function. The classification time can be reduced from 8 minutes and 49 seconds with the full model down to 3 seconds with a very reduced model. The final model we chose is the one with 610 support vectors.

The times and performances given in table 4.9 have been achieved by calculating the mode feature properties only on the pixels which are classified. If the mode properties are calculated on each pixel, the feature calculation time increases from 1 second to 11 seconds.

Model	Full	reduced	reduced
# Support vectors	19,058	**610**	236
Feature calculation (sec)	1	**1**	1
Classification (sec)	528	**18**	3
Total	529	**19**	4
Classification Recall (%)	84.0	**78.3**	69.3
Classification Precision (%)	93.2	**93.0**	93.7
Classification H. mean (%)	88.4	**85.0**	79.7

Table 4.9: Execution time on a Pentium III with 700 Mhz for different SVM model sizes. The final model is the one with 610 support vectors.

Chapter 5

Text detection in video sequences

If it's green, it's biology,
If it stinks, it's chemistry,
If it has numbers it's math,
If it doesn't work, it's technology.

Unknown author

Whereas the last two chapters described text detection in still images, this chapter is dedicated to the process of text detection in video sequences. The fundamental difference is, of course, the additional temporal aspect in movies.

Previous work

Most of the previous work on text detection concentrates on still images. The methods which do treat video sequences perform a frame by frame detection, tracking the detected rectangles across multiple frames. Multiple frame integration in order to produce a single enhanced image is widely done, although mostly simple averaging and minimum/maximum algorithms are employed.

Lienhart and Wernike propose text tracking on overlap information with an additional contents check using signatures calculated from projection profiles [45]. Li and Doermann track text with a pure translational model using a sum of square differences matching algorithm [40]. The same authors propose "classical" multiple frame averaging for artificial text [40] as well as a super-resolution enhancement algorithm for scene text, which is based on the projection on complex sets [41].

Gargi et al. exploit the motion vector information in compressed MPEG video to track text and register text blocks with least square error matching [19]. Assuming non-moving text, Gu suppresses false alarms eliminating moving macro-blocks in MPEG videos [22].

As most previous work, and as we already mentioned in section 2.5, we also detect text on a frame by frame basis, tracking the detected text rectangles across several

frames in order to create text appearances. However, as we will describe in this chapter, we extended the classical algorithms in order to make them more robust and to further exploit properties of artificial text.

This chapter is organized as follows: We describe the tracking process in the next section. Section 5.2 explains our robust multiple frame integration algorithm, which creates a single image of enhanced quality for each text appearance. Section 5.3 describes the experimental results of our system applied to video sequences. Finally, sections 5.4 and 5.5 describe, respectively, extensions of our work concerning the treatment of moving text and statistical processing of recognized text strings.

5.1 Tracking

In the context of video processing, it is obvious that text needs to remain on the screen during a certain amount of successive frames in order to be readable. We take advantage of this property in two different ways:

- Instead of detecting text in the raw input frames, we apply a temporal average filter to the video stream and detect the text in an image which constitutes an average of N frames. More precisely, instead of detecting text in a stream of frames $\boldsymbol{F}_i$, we detect the text in a stream of averages of frames

$$\overline{\boldsymbol{F}_i} = \frac{1}{N} \sum_j \boldsymbol{F}_j \ , \quad j = i - N/2..i + N/2$$

  This initial integration leaves non moving objects (e.g. text) unchanged, whereas the edges of changing objects (some backgrounds) get smoothed. As a consequence, the amount of false alarms decreases.

- From the results of the text detection on the successive frames, we create text appearances and use statistical information on the appearances in order to filter out false alarms. This, of course, holds only if N is not too high.

The tracking algorithm resorts to the overlap information between the list of rectangles detected in the current frame and the list of currently active rectangles (i.e. the text detected in the previous frames which is still visible in the last frame) in order to create the text appearances. Figure 5.1 shows a scheme of the tracking process. During the tracking process we keep a list $\bar{L}$ of currently active text appearances. For each frame i, we compare the list L^i of rectangles detected in this frame with list $\bar{L}$ in order to check whether the detected rectangles are part of the already existing appearances.

The comparison is done by calculating the overlap area between all rectangles $\bar{L}_k$ of the list of active rectangles and all rectangles L^i_l of the current frame i. For all pairs (k, l) with non-zero overlap $A(\bar{L}_k \cap L^i_l)$, the following conditions are verified:

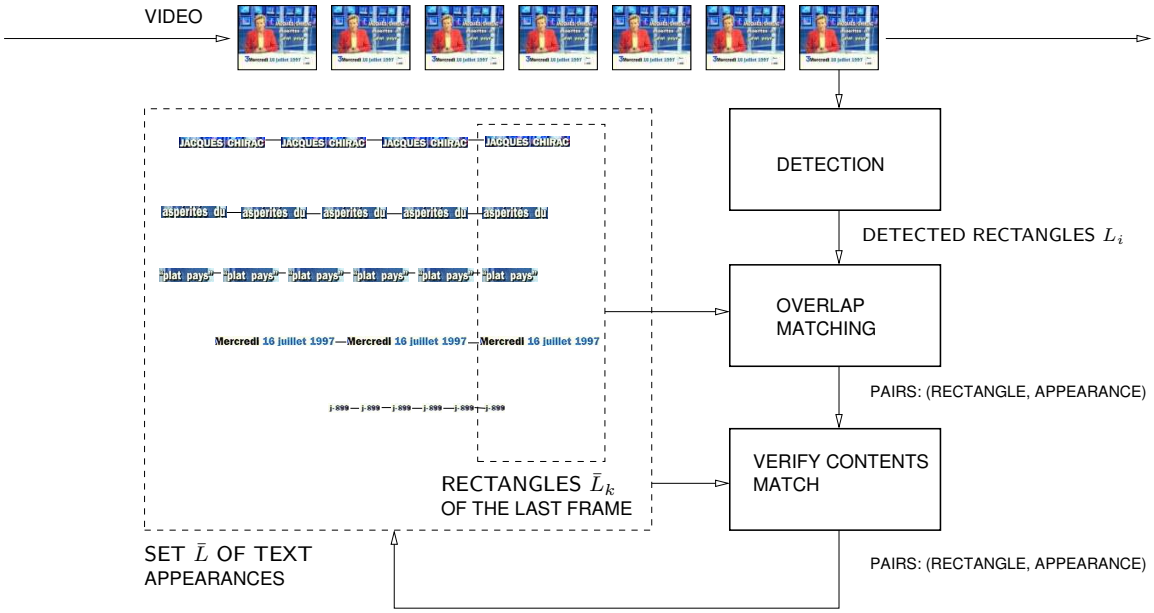
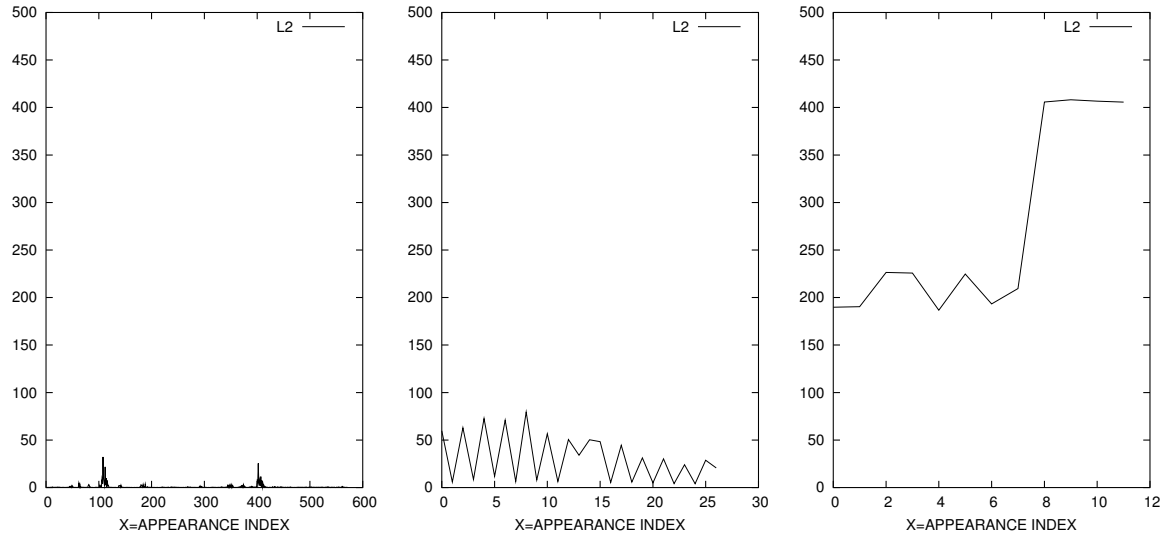


Figure 5.1: The scheme of the text tracking process.

- The difference in size is below a certain threshold.
- The difference in position is below a certain threshold (We assume non moving text and will comment motion in section 5.4).
- The size of the overlap area is higher than a certain threshold.

If for a new rectangle L_l^i one or more active appearances respecting these conditions are found, then the one having the maximum overlap area is chosen and the rectangle L_l^i is associated to this active appearance $\bar{L}_k$. If no appearance is found, then a new one is created (i.e. new text begins to appear on the screen) and added to the list $\bar{L}$. Active appearances in list $\bar{L}$, which are not associated to any rectangles of the current frame, are not sent to the processing step immediately. Instead, they are kept another 5 frames in the list of active frames. This compensates for possible instabilities in the detection algorithm. If during these 5 frames no new rectangle is found and associated to it, then the appearance is considered as finished, extracted from the list $\bar{L}$ and processing of this appearance begins.

Unfortunately, the overlap information used to match the newly detected rectangles to the existing text appearances is not sufficient for an efficient tracking process. In some cases, especially in commercials, text on the screen is replaced with different text on the same position without a significant gap between the two appearances. In some rare cases there is not a single frame between the two different texts, therefore they are recognized by the system as parts of the same text appearance. This can be avoided by performing a content check before adding a newly detected rectangle to a text appearance. Similarly



(a) (b) (c)

Figure 5.2: L_2 distances between the signatures of text boxes: (a) distances between boxes containing the same text; (b) distances between boxes containing a mixture of the same text and other content (fading text); (c) distances between boxes containing different text.

to the solution proposed by Lienhart et al. [80], we compute a signature on the contents of each detected rectangle and compare this signature to the ones computed on the last rectangles in the corresponding appearance.

In our case, the signature is calculated as the vertical projection profile of the Sobel gradient of the sub image. The new rectangle (detected in current frame t) is accepted if the L_2 distance against the last rectangle of the corresponding appearance is below a threshold. In order to take text fading into account, we do not only check against the last frame of the appearance (frame $t-1$) but also against a frame further back. In practice, a check against frame $t-5$ works very well. Figure 5.2 shows the L_2 distances between the signatures of text boxes. Three different types of pairs are compared: text boxes which contain the same text (figure 5.2a), text boxes which contain fading text (figure 5.2b) and text boxes which contain different texts (figure 5.2c). The set of displayed distances contains all calculated distances for a video sequence of 3000 frames. For reasons of clarity, we do not present histograms of the distances but curves which show the distance in a chronological order. The figures illustrate the clear discrimination of the L_2 distance between the different types of matches.

Once the text appearances are created, two additional temporal constraints are imposed. First we use their length as a stability measure, which allows us to distinguish between temporarily unstable appearances and stable text. Only frames which stay on the screen for a certain minimum time are considered as text. Since we concentrate on

artificial text, which has been designed to be readable, this hypothesis is justified.

The second temporal constraint relates to the detection stability. As mentioned above, an active appearance is not finished immediately if new corresponding text is found in the current frame, but kept for another 5 frames. If text is found within these 5 frames then it is associated to the appearance. Therefore, "holes" are possible in each appearance, i.e. frames which do not contain any detected text. If the number of these holes is too large, then we consider the appearance as too unstable to be text. The two conditions, which result in rejection, can be summarized as:

$$number\ of\ frames < t_9 \tag{5.1}$$

$$\frac{number\ of\ frames\ containing\ text}{number\ of\ frames} < t_{10} \tag{5.2}$$

5.2 Multiple frame integration

Before we hand the images of a text appearance to the OCR software, we enhance their contents by creating a single image of better quality from all images of the sequence. If our goal is to create additional information in the enhanced image, i.e. compared to a single image taken from the appearance, then there are several possibilities to ameliorate this information according to the type of text:

- Using statistical techniques, exploit the sub-pixel movements of the text in the image to increase the effective resolution of the image.
- Exploit the fact, that text is static but background may move in order to increase the ratio $\frac{\text{text}}{\text{non-text}}$.

Since we assume that text is static, an exploit of sub-pixel movements is impossible. We therefore chose to take advantage of this hypothesis and combined its exploit with an increase in image resolution, which does not add any additional information, but is necessary because the commercial OCR programs have been developed for scanned document pages and are tuned to high resolutions. Both steps, interpolation and multiple frame enhancement, are integrated into a single, robust algorithm (see figure 5.3).

The area of the enhanced image $\hat{\boldsymbol{F}}$ consists of the median bounding box of all text images $\boldsymbol{F}_i$ taken from the T frames i of the sequence. Each image $\boldsymbol{F}_i$ is interpolated robustly in order to increase its size:

$$F_i' = \uparrow (\boldsymbol{F}_i, \boldsymbol{M}, \boldsymbol{S}) \qquad i = 1..T$$

where the image $\boldsymbol{M}$ contains for each pixel the temporal mean of all images and the image $\boldsymbol{S}$ the temporal standard deviation. Hence, the interpolation function $\uparrow$ takes into account not only the original image but also information on the temporal stability of each pixel. Since the size of the interpolated image is larger than the size of the

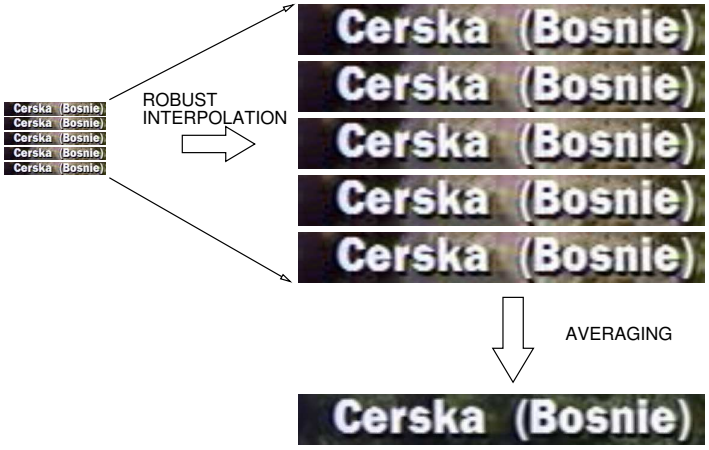


Figure 5.3: A scheme of the multiple frame integration algorithm.

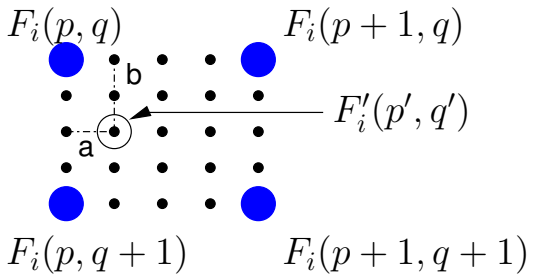


Figure 5.4: Bi-linear interpolation (factor 4x). The interpolated pixel with coordinates (p´,q´) is calculated from its 4 neighbors with coordinates (p,q), (p+1,q), (p,q+1) and (p+1,q+1).

original image, from now on we will denote coordinates in the interpolated images with a prime (e.g. $\hat{\boldsymbol{F}}(p',q'), \boldsymbol{F}'_i(p',q')$) as opposed to the coordinates in the un-interpolated text images (e.g. $\boldsymbol{F}_i(p,q), \boldsymbol{M}(p,q)$). The interpolation is applied to all frame images $\boldsymbol{F}_i$, the final enhanced image $\hat{\boldsymbol{F}}$ being the mean of the interpolated images:

$$\hat{\boldsymbol{F}}(p',q') = \frac{1}{T}\sum_{i=1}^{T} \boldsymbol{F}'_i(p',q')$$

We used two different interpolation functions, the bi-linear and the bi-cubic methods, and adapted them to be more robust to temporal outliers. In the bi-linear algorithm, the gray value of each pixel $\boldsymbol{F}'_i(p',q')$ is a linear combination of the gray values of its 4 neighbors ($\boldsymbol{F}_i(p,q)$, $\boldsymbol{F}_i(p+1,q)$, $\boldsymbol{F}_i(p,q+1)$ and $\boldsymbol{F}_i(p+1,q+1)$), where

$$p = \left\lfloor \frac{p'}{u} \right\rfloor \tag{5.3}$$

$$q = \left\lfloor \frac{q'}{u} \right\rfloor$$

and u is the interpolation factor[1] (see figure 5.4).

The weights $w_{m,n}$ for each neighbor $\boldsymbol{F}_i(p+m, q+n)$ $(m, n \in [0,1])$ depend on the distance between the pixel and the respective neighbor (calculated through the horizontal and vertical distances a and b respectively, between the pixel and the reference neighbor $\boldsymbol{F}_i(p,q)$):

$$a = \frac{p'}{u} - \left\lfloor \frac{p'}{u} \right\rfloor \tag{5.4}$$

$$b = \frac{q'}{u} - \left\lfloor \frac{q'}{u} \right\rfloor$$

From this distance, the weights are calculated as follows:

$$\begin{aligned} w_{0,0} &= (1-a)\cdot(1-b) \\ w_{1,0} &= a\cdot(1-b) \\ w_{0,1} &= (1-a)\cdot b \\ w_{1,1} &= a\cdot b \end{aligned} \tag{5.5}$$

To increase the robustness of the integration process, we added an additional weight $g^i_{m,n}$ for each neighbor $\boldsymbol{F}_i(p+m, q+n)$ to the interpolation scheme which decreases the weights of outlier neighbor pixels by taking into account the temporal stability by means of the temporal standard deviation:

$$g^i_{m,n} = \left(1 + \frac{|\boldsymbol{F}_i(p+m,q+n) - \boldsymbol{M}(p+m,q+n)|}{1 + \boldsymbol{S}(p+m,q+n)}\right)^{-1} \tag{5.6}$$

The final weight for neighbor $\boldsymbol{F}_i(p+m, q+n)$ is therefore $w_{m,n}g^i_{m,n}$, resulting in the following equation for the interpolated pixel $\boldsymbol{F}'_i(p',q')$ of a given frame $\boldsymbol{F}_i$:

$$\boldsymbol{F}'_i(p',q') = \frac{\sum_{m=0}^{1}\sum_{n=0}^{1} w_{m,n}g^i_{m,n}\boldsymbol{F}_i(p+m,q+n)}{\sum_{m=0}^{1}\sum_{n=0}^{1} w_{m,n}g^i_{m,n}} \tag{5.7}$$

In computer vision, bi-linear interpolation is not considered as one of the best interpolation techniques, because of the low visual quality of the produced images. Bi-cubic interpolation produces images which are more pleasing to the human eye, so we adapted it in the same manner as the bi-linear technique to be robust to temporal outlier pixels.

[1] We assume a unique interpolation factor in both x and y direction, but a generalization to two factors is straightforward.

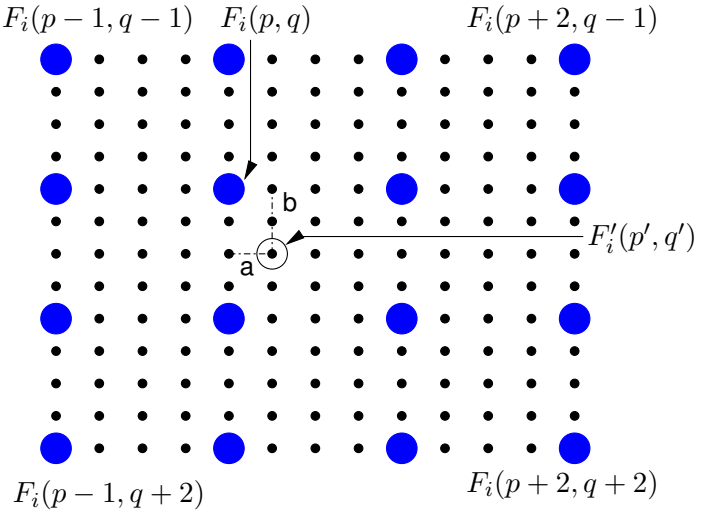


Figure 5.5: Bi-cubic interpolation (factor 4x).

The bi-cubic interpolation method passes a cubic polynomial through the neighbor points instead of a linear function. Consequently, there are 16 neighbor points needed instead of 4 to be able to calculate a new image pixel $\boldsymbol{F}_i'(p',q')$ (see figure 5.5).
As in the bi-linear interpolation scheme, the weights for the different neighbors $\boldsymbol{F}_i(p+m,q+n)$ depend on the distances to the interpolated pixel and are calculated using the horizontal distances a and b to the reference point $\boldsymbol{F}_i(p,q)$ (see figure 5.5). However, instead of setting the weights proportional to the distance, the weight $w_{m,n}$ for each neighbor is calculated as

$$w_{m,n} = R_c(m-a)R_c(-(n-b)) \tag{5.8}$$

where R_c is the cubic polynomial

$$R_c(x) = \frac{1}{6}(x^3 - 3x^2 - 12x + 9) \tag{5.9}$$

and

$$(x)^m = \begin{cases} x^m & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{5.10}$$

The interpolated pixel $\boldsymbol{F}_i'(p',q')$ can therefore be calculated by the equation

$$\boldsymbol{F}_i'(p',q') = \frac{\sum_{m=-1}^{2}\sum_{n=-1}^{2} w_{m,n}g_{m,n}^i \boldsymbol{F}_i(p+m,q+n)}{\sum_{m=-1}^{2}\sum_{n=-1}^{2} w_{m,n}g_{m,n}^i} \tag{5.11}$$

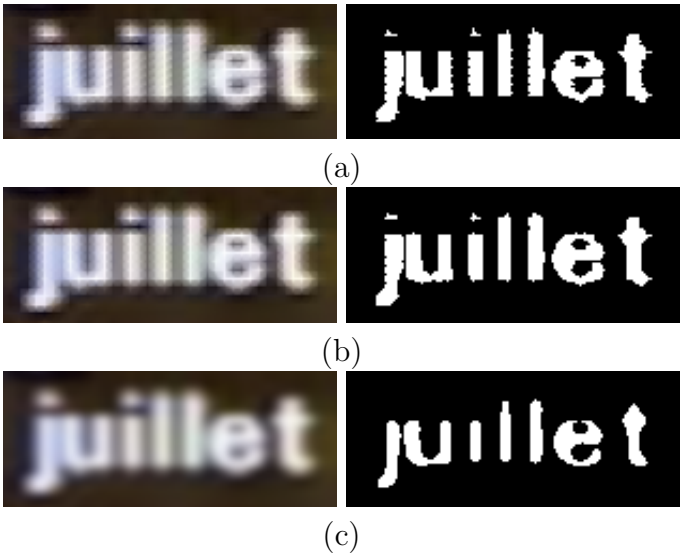


Figure 5.6: Interpolated images (left), after a manual thresholding (right): (a) bi-linear; (b) robust bi-linear; (c) robust bi-cubic.

Figure 5.6 shows an example for a sequence of multiple frames integrated into a single enhanced image by three different methods. Each result image is also thresholded by a manually selected threshold (right column). Figure 5.6a shows the result using a standard bi-linear interpolation on each frame and averaging for the integration into a single image. The result of the robust version of the bi-linear interpolation presented in Figure 5.6b is less noisy at the edges, which is clearly visible at the edges of the thresholded image. This approach takes into account small displacements of the text ($\pm$ 1 pixel) in the temporal sequence which we can assume to be discretization errors. The image created by the robust bi-cubic method presented in Figure 5.6c is visually more attractive but also much smoother. Which interpolation technique is the best will be decided by the OCR results, i.e. we pursue a goal directed selection of techniques.

After the multiple frame integration step, the enhanced image is of a better quality than the original frame images. This allows us to perform some additional segmentation algorithms at this stage of the process. Due to noise and background texture in the original signal, the detected text boxes may contain several lines of text in a single box. The smoothed background in the enhanced image makes it easier to separate the boxes into several boxes containing only one text line each. To do this, we redo the steps already performed to detect the text in a single frame, i.e. we recalculated the accumulated gradients on the enhanced image and binarize them. By searching peaks in the horizontal projection profiles of the text pixels of the binarized image, we are able to extract single lines. Figure 5.7 shows an example rectangle before and after the separation into single lines.

(a) (b)

Figure 5.7: The enhanced image: (a) before separation; (b) after separation into two images with one text line each.

Additionally, since the obtained text boxes now contain only one line of text each, we again impose geometrical constraints on these rectangles, i.e. we remove all rectangles having a ratio of width/height smaller than a threshold t_{11}, where $t_{11} > t_4$.

5.3 Experimental results

To estimate the performance of our system we carried out exhaustive evaluations using a video database containing 60.000 frames in 4 different MPEG 1 videos with a resolution of 384×288 pixels (See table 5.1). The videos provided by INA contain 323 appearances of artificial text from the French television channels TF1, France 3, Arte, M6 and Canal+. They mainly contain news casts and commercials. We manually created ground truth for each video sequence to be able to evaluate the results[2].

5.3.1 Evaluation measures

The automatic evaluation schemes for still images described in section 3.4.1 is very difficult to extend to the detection in video sequences and bears the risk to be error prone. Furthermore, during the experiments on video sequences we discovered that there are less text objects which are detected partially only: text objects are either detected as a whole or not at all. This may be explained with the initial integration in the video algorithm, which has a smoothing effect on the background, and hence increases the contrast between text and background.

We therefore decided to perform a manual evaluation of the detection in video sequences by deciding manually for each ground truth text object whether it has been detected or not and for each detected text rectangle whether it is present in the ground truth or not. Thus, the precision and recall values we give in this section are not based on surfaces but on counts only (i.e. the number of detected rectangles, number of existing rectangles etc.).

[2]The ground truth files can be downloaded from http://rfv.insa-lyon.fr/~wolf/groundtruth

Sample	minutes	frames	text boxes	Contents
#1	10	15000	110	Commercials (France 3, TF1)
#2	10	15000	85	News (M6, Canal+)
#3	10	15000	63	Cartoon, News (Arte)
#4	10	15000	65	News (France 3)
Total	40	60000	323	

(a)

Video #1: aim1mb02 provided by INA: commercials (France 3, TF1)

Video #2: aim1mb03 provided by INA: news (M6, Canal+)

Video #3: aim1mb04 provided by INA: cartoon, news (Arte)

Video #4: aim1mb05 provided by INA: news (France 3)

(b)

Table 5.1: The videos used in the experiments: (a) description; (b) screenshots.

For object classification tasks, where a special object property (e.g. *the object is text*) needs to be detected, the evaluation results are traditionally given in a confusion matrix. The matrix consists of 4 different cases:

- the object is text and has been detected as text (*true positive*).
- the object is non-text and has been detected as non-text (*true negative*).
- the object is text and has been detected as non-text (*false negative*).
- the object is non-text and has been detected as text (*false positive, false alarm*).

Our system for text detection in videos is different to traditional classification systems in various ways. There is no such figure as *the object is non-text and has been detected as non-text*, since text which does not exist and is not detected, cannot be counted. Furthermore, an actual text string present in the ground truth may probably correspond to several detected text rectangles, since the detection process may split up a text appearance and detect it more than once[3]. Hence, the rows and columns of such a confusion matrix do not sum up correctly.

We therefore present detection results in a different way, as shown in table 5.2. The table consists of two parts. The upper part is ground truth oriented, i.e. the categories are parts of the set of ground truth rectangles. It shows the number of rectangles in the ground truth which have been correctly detected (*Classified as text*) and missed (*Classified as non-text*). The total number of rectangles in the ground truth is shown as *Total in ground truth*. From these figures the recall of the experiment can be calculated as

$$\text{Recall} = \frac{\text{Classified as text}}{\text{Total in ground truth}} \tag{5.12}$$

The lower part of the table is detection related, i.e. the categories are parts of the set of detected rectangles. It shows the number of correctly detected artificial text rectangles (*Positives*) [4], the number of false alarms (*False alarms*), the number of rectangles corresponding to logos (*Logos*) and detected scene text rectangles (*Scene text*). A total number of detected rectangles is given as *Total detected*. The precision of the experiment can be calculated as

$$\text{Precision} = \frac{\textit{Total - false alarms}}{\textit{Total}} \tag{5.13}$$

This figure is based on the number of artificial text, scene text and logos, since the latter two types of text cannot be considered as false alarms, although the detection system has not been designed with their detection in mind.

[3] Let us remember that the system has been designed for video indexing, so we do not want to punish multiple detections.

[4] As mentioned above, the number of positives may differ from the number of corrected classified ground truth rectangles (*Classified as text*) if text has been detected as more than one text appearance.

5.3.2 Results

Table 5.2 shows the results for the detection in video sequences. We used the two algorithms for the detection in still images introduced in chapters 3 and 4. The results for the first algorithm, which assumes that there is text in an image in order to create a statistical model of the two populations, is shown in table 5.2a. We achieve an overall detection rate of 93.5% of the text appearing in the video. The remaining 6.5% of missing text are mostly special cases, which are very difficult to treat, and very large text (larger than a third of the screen), which could be detected using a multi resolution method, but at the additional cost of a higher number of false alarms. The precision of the system is situated at 34.4%, which means that there are unfortunately a large number of false alarms.

Table 5.2b gives the results for the second algorithm, which learns contrast and geometrical features from training images. As expected, the precision of the method is much higher (34.4% $\rightarrow$ 75.0%) at the cost of a slightly lower recall (93.5% $\rightarrow$ 88.2%).

The influence of generality

Together with the detection performance, table 5.2 also gives information on the generality of the videos. In this context, we regard the video as a set of frames, i.e. a set of still images, and calculate the generality of the video as given in (3.16). Note, that this notion of generality measures text rectangles per frame, not text appearances per frame, i.e. that a video of length 100 frames containing a single text appearance of length 50 frames possesses a generality of $\frac{1}{2}$ and not $\frac{1}{100}$. We chose this way of calculation because the detection process itself operates on a frame by frame basis.

The generality of the four test videos is situated between 0.70 and 1.04 text rectangles per frame, whereas the performance values for still images in chapters 3 and 4 have been given for image sets with generalities between 1.49 and 6.20. Obviously, video data is of very low generality. Another measure which is related to the generality given in (3.16) is the amount of text frames in the total set of frames:

$$\text{Ratio text frames} = \frac{\text{Number of images containing text}}{\text{Total number of images}}$$

As said above, this measure is related to generality, but it also conveys information which is independent of the generality. Datasets may have a high generality value but a low ratio of text frames, e.g. if the set consists of a large part of images without text and a smaller part of images with very much text. This effects text detection with the local contrast detection method, which is characterized by a low detection precision if applied to images without text. For the still image datasets presented in chapters 3 and 4 and the precision/generality curves shown in figures 3.9 and 4.15, this ratio ranged between 0 and 0.5. For the four test video sequences, the ratio takes values between 0.31 and 0.40.

If we compare the performance for the two detection algorithms on still images with

Category	Video files #1	#2	#3	#4	Total
Classified as text	103	80	59	59	301
Classified as non-text	7	5	4	5	21
Total in ground truth	110	85	63	64	322
Positives	114	78	72	86	350
False alarms	138	185	374	250	947
Logos	12	0	34	29	75
Scene text	22	5	28	17	72
Total - false alarms	148	83	134	132	497
Total detected	286	268	508	382	1444
Recall	93.6	94.1	93.7	92.2	**93.5**
Precision	51.4	31.0	26.4	34.6	**34.4**
Harmonic mean (%)	66.7	46.6	41.2	50.3	**50.3**
Generality	0.80	1.04	0.85	0.70	0.85
Ratio text frames	0.39	0.32	0.31	0.40	0.34

(a)

Category	Video files #1	#2	#3	#4	Total
Classified as text	92	77	55	60	284
Classified as non-text	18	8	8	4	38
Total in ground truth	110	85	63	64	322
Positives	121	112	70	81	384
False alarms	48	17	66	40	171
Logos	12	1	9	17	39
Scene text	28	8	27	27	90
Total - false alarms	161	121	106	125	513
Total detected	209	138	172	165	684
Recall (%)	83.6	90.6	87.3	93.8	**88.2**
Precision (%)	77.0	87.7	61.6	75.8	**75.0**
Harmonic mean (%)	80.2	89.1	72.3	83.8	**81.1**
Generality	0.80	1.04	0.85	0.70	0.85
Ratio text frames	0.39	0.32	0.31	0.40	0.34

(b)

Table 5.2: The detection results for video sequences given on text object level: (a) local contrast based detection; (b) SVM-learning based detection.

the detection performances of the same algorithms on video sequences, then we remark two differences:

- Looking at the performance of the detection in still images, we notice a small advantage of the SVM based algorithm over the local contrast method regarding detection precision. This small advantage turns into a significant advantage when the detection algorithms are applied to video images (from a precision of 34.4% to 75.0%). This may be explained with the low generality of video sequences, as the differences in precision between different algorithms become more and more important as the generality of a dataset decreases.

- The high difference in recall between the two detection algorithms for still images becomes smaller when the algorithms are applied to video sequences. Apparently, the smaller robustness of the SVM algorithm is compensated by the fact that each text appearance may be detected at several frames. Most appearances are between 50 and 100 frames long (the median length of all appearances in ground truth is 89 frames), which provides several chances for the detection process to detect a part of the appearance which is longer than the minimum length t_9.

We may conclude, that the performance advantage shifts from the local contrast method to the SVM method as the generality of the dataset decreases.

The influence of the system parameters

Table 5.3 shows the detection performance for different values of parameter t_9, the threshold on appearance length of text. For reasons of complexity, we present the results for the detection on one of the video files only. The video in question (video #1) has been chosen for its particular content: Among other sequences, it contains some commercials with text appearing only for a short time. In some extreme cases, text appears only 10-15 frames (less than half a second). Furthermore, in some cases, different text appears on the same position on the screen without empty frames between two appearances.

We performed experiments for three different values of t_9: 15, 25 and 40 frames. As table 5.3a illustrates, precision increases as the minimum appearance length is increased. As expected, at the same time recall drops as these short text appearances are filtered out. The chart in table 5.3b shows these results in the form of a graph.

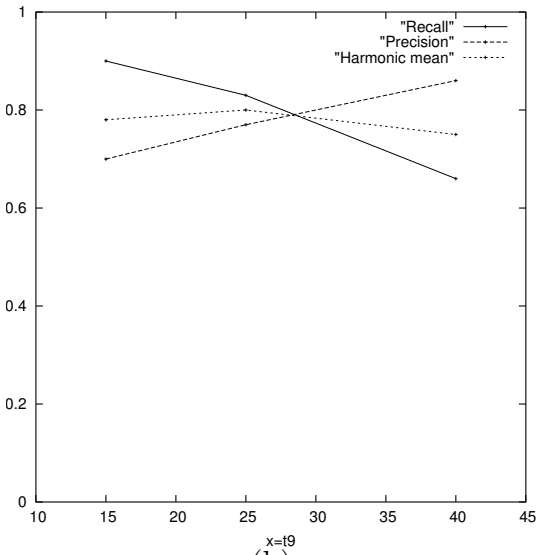
The parameter can be controlled by an operator: guided by a priori knowledge on the type of video, the parameter may be adjusted to the data.

5.4 Moving text

The detection methods presented in this thesis are restricted to static, non-moving text. An extension of our work to moving text has been introduced by Marquis et Brès [49].

Category	Parameter t_9 15	25	40
Classified as text	99	92	73
Classified as non-text	11	18	37
Total in ground truth	110	110	110
Positives	147	121	93
False alarms	85	48	18
Logos	22	12	8
Scene text	30	28	16
Total - false alarms	199	161	117
Total detected	284	209	135
Recall (%)	90.0	**83.6**	66.4
Precision (%)	70.1	**77.0**	86.7
Harmonic mean (%)	78.8	**80.2**	75.2

(a)



(b)

Table 5.3: Detection results on video #1 for different minimum appearance lengths: (a) confusion matrix; (b) performance graph.

Their algorithm has been designed for linear scrolling-like movements as e.g. the castings at the end of movies, which move vertically, or the newsticker-like horizontal moving text on news channels. The text is tracked comparing the image contents on a frame by frame basis. In order to be sure that only text is tracked, a previous detection stage removes all non-text from the frame. A correlation in the frequency domain is employed in order to match text blocks.

To enhance the image quality, the method resorts to the multiple frame integration algorithm presented in section 5.2. However, the registered images are moved and aligned before the integration is done.

5.5 Statistical string processing

As pointed out in section 2.5, multiple frame integration — to produce a single image out of the frames of a text appearance — is not the only way to treat text appearances, which are 3D objects with a temporal dimension. Another possibility is the application of the OCR algorithm to each of the frames of the text appearance. In this case, the resulting single text string for the appearance must be some kind of combination of the results for each frame.

Researchers of our team are currently working on a theoretical framework which can be used to tackle this problem: the statistical processing of symbolic strings [30]. The combination method takes recourse to an algorithm called *reweighted set median*. Its

goal is to calculate a robust median string from a set of strings, which is assumed to contain noisy versions of the same string and eventual additional noise. Based on the Levenstein distance between two strings [79] (see section 6.3.1), two concepts are derived, which correspond to the median and the variance in $\Re$, respectively: the set median string (sms) and set deviation string (sds). The re-weighted set median is calculated by an iterative algorithm, which computes the median string and iteratively discards strings from the set, which have a high distance to the previous median. The distance function uses weights which depend on the set deviation strings. In other words, strings which are close to the set deviation are less likely to be discarded.

The algorithm is powerful enough to even remove high amounts of noise in the set of strings, as for example the presence of other text strings which were not taken from the same text appearance[5]. Figure 5.4 shows some examples of text extracted from a target appearance ("Hôpital Michallon (Grenoble)") mixed with some strings of two other appearances ("J.-Hazard" and "montage") and handed to an OCR program frame by frame. The complet set contains 197 strings in three clusters of size 79, 18 and 49. The set median of the ensemble is "l$montagne". However, using the reweighted set median we get the correct result "Hôpital Michallon (Grenoble)".

Details to the theoretical framework of set median and set deviation, as well as details to the application to the treatment of OCR results can be found in [30].

[5]This situation rarely occurs in our text detection system because of the signature check descibed in section 5.1.

```
HopjtaJ Michalion (Grenoble)
HopjtaJ Michalion (Grenoble)
fji-kVufe-ii .on -,
Hopjtaj Michallon (Grenoble)
Hôpital Michallon (Grenoble)
Hopjtal Michallon (Grenoble)
HÆopital Michallon (Grenoble)
.-yMTJifeiÆ1fc ¡MV .
+-HÆ opital Michallon (Grenoble)
*M5¡!tefe.aF : - *
HÆopital Michallon (Grenoble)
,.*rateM *HI .-. , r
J. -HÆazard ;
J.-Hizard ;
J.-Hizard ;
J. rd 1 j.
HÆopital Michallon (Grenoble)
.4ei. 4f r  X   rateMW ;ji v *
Hopjtal Michallon (Grenoble)
Hopjta[ .Michallon (Grenoble)
Hopjtal.Michallon (Grenoble) -
Hopttal Michallon (Grenoble) .
HÆopital Michallon (Grenoble)
lW. vx.   Mm* i .m - *
HÆopital Michallon (Grenoble)
tlmm i*h - . Jmmmmmmmi mmmMi
i * *  jrt**.4
HÆopital Michallon (Grenoble)
montage-* *-
morn jTga
montage-
montage*
montage
montage-
montage-L
nrontage
I .montage
montage
```

Table 5.4: Examples for text detected and recognized without multiple frame integration. The set of strings contains text from three different text appearances.

Chapter 6

Character segmentation and recognition

Ein Kunstwerk schön finden heißt,
den Menschen lieben, der es hervorbrachte.

To like a piece of art
means to love the person who created it.

Christian Morgenstern
German writer and poet (1871 - 1914)

As a final step in the text extraction process, the enhanced image needs to be segmented before we can hand it to an OCR program. Although most OCR software is equipped with a binarization step, the algorithms employed are mostly tuned to the text images taken from scanned documents, which are inherently different from video data (see section 2.2). For this reason, we developed our own character segmentation algorithms. The difficulties of the existing methods are twofold:

- Text images created from scanned documents are characterized by a large contrast between characters and background. In most cases, text characters are very close to black and the background is very close to white. On the other hand, video text images may be of low contrast.
- When documents are scanned, the resolution may be chosen high enough by the user to simplify the recognition process. As a result, document images are usually of very high resolution (300-400dpi) with character sizes of around 100×100 pixels, whereas video text contains characters with sizes as low as 10×10 characters and lower.

As a consequence, we developed two different algorithms which address these problems: an adaptive thresholding algorithm which has been designed to increase the local contrast in the text image, and a character segmentation algorithm which takes into account

a priori knowledge on the image — modeled by a Markov Random Field (MRF) — in order to achieve a better segmentation. These two methods are presented, respectively, in the next two sections. Section 6.3 gives results for both methods.

Previous work

To propose a new binarization algorithm may be — at a first glance — a little bit surprising, as these algorithms have been studied extensively in the past, even since the very beginning of research on digital images. Several types of methods may be cited.

The first methods had been developed for the general binarization case. A classical approach is the global thresholding algorithm introduced by Otsu [59], which takes recourse to theoretical work on discriminant analysis by Fisher. We employed this algorithm to threshold the image of accumulated gradients in one of our text detection methods (see section 3.1). A method minimizing the thresholding error has been given by Kittler et al. [35]. The histogram of the image is modeled as a sum of normal distributions and the optimal threshold is calculated as the intersection of the distributions. A good survey of algorithms for the general thresholding case has been written by Sahoo et al. [62].

Several binarization techniques have been presented by the document analysis community, which was soon confronted with the problem of thresholding very large images with changing illumination conditions. As a consequence, most algorithms are adaptive instead of global. One of the simplest but also one of the most efficient methods has been suggested by Niblack in [57]. It glides a window across the image and calculates a threshold for each pixel based on simple statistics on the gray values inside the window. We present this algorithm together with Sauvola et al.'s improved version [64] in more details in the next section. Seeger and Dance improve Niblack's method in order to take into account lighting variations which occur in images taken with digital cameras [66]. The method proposed by Yanowitz and Bruckstein is characterized by a higher complexity [90]. A threshold surface is created which passes through the edge pixels of the image. Between the edge pixels, the surface is relaxed in an iterative algorithm.

A survey and evaluation of binarization algorithms for document images is the well known paper of Trier and Jain [74]. Therein, the methods have been evaluated by comparing the results of a following OCR phase. They concluded, that the best performance had been achieved by Niblack's algorithm and Yanowitz and Bruckstein's algorithm, where Niblack's method is one of the simplest and fastest and Yanowitz-Bruckstein's one of the most complex and slowest. Trier et al. performed their experiments on "traditional" documents produced by scanners.

In the framework of text detection and recognition in images and videos, several binarization methods have been proposed. In this case, the domains of binarization, character segmentation and character restoration are intermingled. As mentioned in section 2.4, the early text detection algorithms were based on segmentation and spatial regrouping. For these methods, character segmentation is an integral part of the detection algorithm and is mostly done by color clustering. As example we may cite Sobotka

et al. [69] and Lee and Kankanhalli [39].

Some methods use standard binarization methods but adapt them to the situation at hand. For instance, LeBourgeois applies a maximum entropy method to binarize a text rectangle and searches character borders as peaks in a projection profile [38]. Antani et al. perform logical level thresholding on a pre-processed text rectangle and post-process the connected components of the binary image, applying heuristics to remove non-text objects [4]. Lienhart and Wernike use simple global thresholding [45]. However, the threshold is determined from the text and background color, which are estimated using histograms on the border and at the center of the text box.

Wu, Manmatha et al. calculate the threshold from the histogram of text "chips", which are detected with texture analysis methods [87]. Sato, Kanade et al. assume that text is composed of horizontal, vertical or diagonal line segments and design a filter to extract these lines, followed by fixed thresholding [63]. This algorithm is further improved by Natarajan et al., who combine it with a color clustering step [56]. In a similar work, Chen et al. propose asymmetric Gabor filters to enhance the strokes of text [13]. However, instead of just assuming that the strokes are oriented in the principal directions of image grid, for each pixel the stroke direction is estimated using the same set of Gabor filters.

An appealing idea is to introduce a priori knowledge on the images into the segmentation algorithms in order to increase their efficiency. Different ways to model the a priori knowledge have been suggested. For instance, Allier et al. resort to active contours to segment and restore characters [3].

Markov Random Fields (MRF) and Gibbs Distributions have been used successfully for image restoration, as presented in the seminal work of Geman and Geman [20]. We believe, that MRF models are very well suited for modeling text properties, but the models found in the literature are not rich enough and do not exploit enough the properties of text characters. Thouin et al. use 3×1 and 1×3 cliques and energy functions stimulating horizontal and vertical homogeneous strokes to restore bimodal text images from low resolution samples [73]. Cui and Huang use 2×2 cliques with similar energy functions to extract binarized car license plate images from gray scale sequences [16]. Both approaches use a simple model, where the MRF is used to model the prior for an MAP estimator which tends to remove simple noise and augments small straight strokes.

Chen et al. also model text with a MRF based on 2×2 cliques [12]. However, instead of estimating each pixel label separately, they estimate two global thresholds on the image, which best validate the MRF model. The estimation is performed with the Expectation-Maximization algorithm.

6.1 Binarization by contrast maximization

As already mentioned, the images extracted from videos of low quality do not have the same characteristics as scanned document images. In our experiments we found out,

that for our purposes, the simpler algorithms are more robust to the noise present in video images. We therefore chose Niblack's method for our system, and finally derived a similar method based on a criterion maximizing local contrast.

Niblack's algorithm calculates a threshold surface by shifting a rectangular window across the image. The threshold T for the center pixel of the window is computed using the mean m and the variance s of the gray values in the window:

$$T = m + k \cdot s \tag{6.1}$$

where k is a constant set to -0.2. The results are not very sensitive to the window size as long as the window covers at least 1-2 characters. However, the drawback is noise created in areas which do not contain any text, due to the fact that a threshold is created in these cases as well. Sauvola et al. [64] solve this problem by adding a hypothesis on the gray values of text and background pixels: text pixels are assumed to have gray values near 0 and background pixels are assumed to have gray values near 255. This hypothesis results in the following formula for the threshold:

$$T = m \cdot (1 - k \cdot (1 - \frac{s}{R})) \tag{6.2}$$

where R is the dynamics of the standard deviation fixed to 128 and the parameter k is fixed to 0.5. This method gives better results for document images, but creates additional problems for video frames whose contents do not always correspond to the hypothesis, even if images containing bright text on dark background are reversed to resemble better the desired configuration (see section 6.3 on the experimental results for examples).

We propose to normalize the different elements used in the equation which calculates the threshold T, i.e. to formulate the binarization decision in terms of contrast instead of in terms of gray values, which is a natural way considering the motivation behind Niblack's technique[1]. How can we define contrast? If we refer to Niblack [57], p. 45, then the local contrast of the center pixel of a window of gray levels is defined as

$$C_L = \frac{|m - I|}{s} \tag{6.3}$$

where I is the gray value of the center pixel and, as mentioned above, m is the mean of the gray values and s is the standard deviation of the gray values of the window. Since we suppose dark text on bright background, we do not consider points having a gray value which is higher than the local mean, therefore the absolute value can be eliminated. Denoting by M the minimum value of the gray levels of the whole image, the maximum value of this local contrast is given by

$$C_{max} = \frac{m - M}{s} \tag{6.4}$$

[1]Niblack derived his method from algorithms to transform gray level histograms in order to increase the contrast in images, e.g. to display them.

It is difficult to formulate a thresholding strategy defined only on the local contrast and its maximum value, since this does not take into account the variation of the window with respect to the rest of the image. This is the reason why we also propose the definition of a more global contrast, the contrast of the window centered on the given pixel:

$$C_W = \frac{m - M}{R} \tag{6.5}$$

where $R = max(s)$ is the maximum value of the standard deviations of all windows of the image. This contrast tells us, if the window is rather dark or bright with respect to the rest of the image (a high value characterizes the absence of text). Now we can express our binarization strategy in terms of these contrasts. A simple thresholding criterion is to keep only pixels which have a high local contrast compared to it's maximum value corrected by the contrast of the window centered on this pixel:

$$I : C_L > a(C_{max} - C_W) \tag{6.6}$$

where a is a gain parameter. Developing this equation we obtain the following threshold value:

$$T = (1 - a)m + aM + a\frac{s}{R}(m - M) \tag{6.7}$$

In the case where the given pixel is the center of a window with maximum contrast, i.e. $s = R$, we get $T = m$. Thus, the algorithm is forced to keep the maximum number of points of the window. On the other hand, if the variation is low ($s << R$), then the probability that the window contains text is very low. We therefore keep a pixel only if it's local contrast is very high. The threshold is therefore $T \approx (1 - a)m + aM$. The gain parameter a allows to control the uncertainty around the mean value. A simple solution is to fix it at 0.5, which situates the threshold between m and M.

We remark that global parameters have been introduced into the underlying adaptive algorithm: the minimum gray value M and the dynamic range R are calculated on the whole image. Consequently, the method is less suited for large images from scanned documents. On the other hand, images extracted from video sequences are small and generally do not contain large illumination changes. The global parameters allow the algorithm to adapt itself to the brightness and the contrast of the text box. The actual threshold itself is still calculated in an adaptive manner based on the statistics of the gray values in the local window.

The computational complexity of this new technique is slightly higher than Niblack's version, since one of the thresholding decision parameters is the maximum standard deviation of all windows of the image. Therefore two passes are required. On the other hand, it is not necessary to shift the window twice across the image, if the mean value and standard deviations for each pixel (i.e. each window) are stored in two-dimensional tables.

Furthermore, for the calculation of the mean and standard deviation, only the first window for the first pixel of each line needs to be traversed entirely. The mean value and the standard deviation of the other windows can be calculated incrementally from the preceding windows, if for each window the sum of the gray values and the sum of the squares of the gray values are held.

6.2 Character segmentation using a priori knowledge

Most original text images without noise are binary and consist of text pixels and background pixels. Moreover, the spatial distribution of the pixels of these two populations in the image is not arbitrary. Even considering multiple languages and scripts, there is only a limited number of characters and therefore also a limited number of binary pixel "layouts" which form these characters. An appealing idea is to learn the possible pixel configurations from "perfect" training data and to use this a priori knowledge for the binarization of noisy images.

Of course, given all the different possible types of text in different fonts, sizes, styles, orientations, colors etc., it is difficult or near impossible to create an exact model of text which fits all possible text observations. In part, this is the reason for the simplicity of the existing text models. However, the fonts of most of the low quality text in video broadcasts are very simple and without serifs so they remain readable when rendered at low resolution. The method described in this section creates a Markov random field model of text from training data and uses it in a Bayesian estimation algorithm to segment noisy text images.

The section is outlined as follow: sub section 6.2.1 gives a short introduction into MRFs and Gibbs Distributions. Sub section 6.2.2 presents the observation model and sub section 6.2.3 the prior model. Sub section 6.2.4 explains the annealing process used to minimize the energy function.

6.2.1 Markov random fields and Gibbs Distributions

MRF models treat images as stochastic processes. A field X of random variables X_{s_1}, X_{s_2}, ... X_{s_N} is a MRF iff

$$P(X{=}z) > 0 \quad \forall z \in \Omega$$
$$\text{and}$$
$$P(X_s{=}x_s | X_r{=}x_r, r \neq s) \;=\; P(X_s{=}x_s | X_r{=}x_r, r \in g_s)$$

where z is a configuration of the random field, Ω is the space of all possible configurations and g_s is the neighborhood of the pixel X_s. In other words, the conditional probability for a pixel of the image depends only on the pixels of a pre-defined neighborhood around this pixel. This neighborhood is defined in terms of cliques, where a clique is the set

of pixels which are neighbors of the given pixel. It has been proven that the joint probability density functions (p.d.f.) of MRFs are equivalent to Gibbs distributions, i.e. are of the form

$$\pi(z) = \frac{1}{Z} e^{-U(z)/T}$$

where Z is a normalization constant, T is a temperature factor which can be assumed to be 1 for simplicity,

$$U(z) = \sum_{c \in C} V_c(z) \tag{6.8}$$

is an energy function, C is the set of all possible cliques of the field and V_c is the energy potential defined on a single clique. Hence, the model of the spatial relationship in the image can be defined in terms of energy potentials V_c of the clique labelings.

In this paper the MRF models the prior $P(z)$ in a Bayesian maximum a posteriori (MAP) estimator which determines the binary estimate from a degraded observation:

$$\hat{z} = \arg\max_z P(z|f) = \arg\max_z P(z)P(f|z) \tag{6.9}$$

The likelihood $P(f|z)$ depends on the observation and the noise process. The prior and likelihood models are described in the next two sub sections.

6.2.2 The observation model

The likelihood or conditional density is the probability of the observation given an estimate. It depends on the observation model assumed for the process. Most approaches use simple models, e.g. Gaussian noise with zero mean and variance σ_n^2 and text and background gray values of 0 and 255 respectively, which leads to the probability density function

$$P(f|z) = (2\pi\sigma_n^2)^{-N/2} \ \exp\left\{\frac{||z-f||^2}{2\sigma_n^2}\right\}$$

where f is the observed gray scale image, z is the estimated binary image and N is the image size in number of pixels. However, in real life this is rarely the case. Modeling the image degradation this way results in fixed thresholding with threshold 127.5 if the prior distribution is set to uniform — a method which is not acceptable.

Given a reliable estimate of the text gray value(s), background gray value(s) and the variance of the noise process, a model assuming Gaussian noise results in the p.d.f.

$$P(f|z) = (2\pi\sigma_n^2)^{-N/2} \ \exp\left\{\frac{||c(z)-f||^2}{2\sigma_n^2}\right\} \tag{6.10}$$

where $c(z)$ is the function delivering the text gray value for text pixels and background gray value for background pixels. Unfortunately, estimating the text and background

color (or gray value) is a difficult task and far from trivial. We performed experiments with an MRF based binarization scheme using the observation model (6.10) together with a uniform prior. Due to inaccurate estimates, the results are not as good as the ones obtained by existing binarization techniques such as Sauvola et al's method (see section 6.1). The goal of our approach is to improve the classic techniques by using spatial relationships defined in the prior p.d.f. Therefore, as a desired property, we want the new technique to give at least the same results as these existing techniques, if the prior distribution is set to uniform. We therefore incorporated Sauvola et al.'s method into the observation model for the new technique as follows:

$$P(f|z) = (2\pi\sigma_n^2)^{-N/2} \ \exp\left\{\frac{||z - f + T - 127.5||^2}{2\sigma_n^2}\right\} \tag{6.11}$$

where T is the threshold for the given pixel produced by Sauvola et al.'s method. Fixed thresholding is replaced by thresholding using the threshold surface computed by Sauvola's algorithm. The last parameter to estimate is σ_n^2, the variance of the noise process. If the prior is uniform, then a change of the variance does not effect the result. However, if a non uniform prior is taken into account, then the noise variance should be estimated as closely as possible since it controls the weight between the prior model and the observation model.

We use local statistics of the gray levels of the image to estimate the noise variance. A window is shifted across the image and the gray level histogram of the pixels in the window is computed. Then the histogram is separated into two classes by calculating Otsu's optimal threshold (which is equivalent to minimizing the intraclass variance, see section 3.1). Finally, the standard deviation of the noise is estimated as $\sigma_n = w\ \sigma_{ic}$, where σ_{ic} is the intraclass standard deviation and w is a weighting factor which we fixed to $w = 0.5$.

6.2.3 The prior distribution

As equation (6.8) suggests, the MRF for the prior is defined in terms of cliques, i.e. a set of predefined neighborhood configurations and their corresponding binary labeling. For each labeling, a "clique energy" or "clique potential" is given, which corresponds to the probability to find this labeling in the model. The sizes of the cliques depend on the order of the MRF, which in our case depends on the properties of the text we want to model. In reality, the clique sizes necessary to accurately model text are unrealistically high, resulting in prohibitive complexity. Instead, the MRF is approximated with smaller cliques, which is a strong limitation.

As mentioned above, the cliques used in previous work are comparatively small (3×1, 1×3, 2×2, etc.). Our proposed prior model is defined on a large neighborhood (4×4 pixel cliques), which makes it more powerful. On the other hand, simple tabularization of the clique potentials is not possible anymore, since we would need to specify the energy potentials for a large number of clique labelings ($2^B = 65535$, where B=16 is the

number of pixels in the clique). We decided on a different approach, learning the clique potentials from training data. The probability of a clique labeling θ_i can be estimated from the frequency of its occurrence in the training images. The probability can be converted into a clique potential as follows:

$$V_c(\theta_i) = -\frac{1}{B} ln(P(\theta_i))$$

Unfortunately, not all theoretically possible clique labelings are usually (and realistically) found in the training images. In our case, learning cliques labelings of size 4×4, only 8.4% of the possible labelings are found in the training data. The question arises how to find the potentials for the missing cliques. One solution has been proposed by Milun and Sher [53] as an application of the work of Hancock and Kittler [23]. After initializing the unfound probabilities with zero, the probability distribution of the clique labelings $\theta_i \in \Theta$ is smoothed using the following function:

$$P'(\theta_i) = \sum_{\theta_j:\ P(\theta_j)>0} P(\theta_j) p^{d(i,j)} (1-p)^{u(i,j)}$$

where $d(i,j)$ is the number of bits which differ between the clique labelings θ_i and θ_j, and $u(i,j)$ are the number of bits which are the same. p is the smoothing coefficient, higher values denote more smoothing. Thus, the probability of each clique labeling, also the ones which did occur in the training data, is replaced by a new probability which is a function of the found probabilities.

A typical training document image contains far more background pixels than text pixels (a typical value is about 1% of text pixels). This leads to very low energy potentials for the cliques containing many background pixels. We therefore normalized the clique potentials by a term which only depends on the histogram of the clique. Let θ_a be a clique labeling which contains $|\theta_a|_t$ text pixels and $|\theta_a|_{nt}$ background pixels. In an image randomly generated from a stationary but biased source, which generates text pixels with probability $P(X{=}0) = \alpha$ and background pixels with probability $P(X{=}255) = \beta$, the probability to find θ_a on a location is

$$P_i(\theta_a) = \alpha^{|\theta_a|_t} \beta^{|\theta_a|_{nt}}$$

Dividing the measured clique probability by this factor we normalize the energy distribution by increasing the energy of cliques with many background pixels and decreasing the energy of cliques with many text pixels (assuming that $\alpha < \beta$, as it is the case for document images). The final energy potential is calculated as follows:

$$V_c(\theta_i) = -\frac{1}{B} ln(\frac{1}{Z} \frac{P'(\theta_i)}{P_i(\theta_i)})$$

where $Z = \sum_{\theta_j} \frac{P'(\theta_j)}{P_i(\theta_j)}$ is a normalization constant.

6.2.4 Optimization

To estimate the binary image, equation (6.9) must be maximized. Unfortunately, the function is not convex and standard gradient descent methods will most likely return a non global solution. Simulated Annealing has been proven to return the global optimum under certain conditions [20].

Simulated Annealing received its name from physical processes, which decrease temperatures to allow particles (e.g. atoms in an alloy) to relax into a low energy configuration. Similarly, for the optimization of a non-convex function, the simulated annealing process lowers a — virtual — temperature factor. During the annealing process, pixels of the estimated binary image are flipped in order to bring the estimation closer to the model. However, a certain amount of randomness is included in the optimization process, which allows the system to flip to more unfavorable estimates at certain times. This amount of randomness depends on the temperature factor: it is set relatively high at the beginning to allow the system to "jump" out of local minimums, and is gradually lowered together with the temperature factor.

More precisely, during the annealing process, for each pixel the energy potential is calculated before and after flipping it. The decision whether to flip the pixel or not is based on the value

$$q = e^{-\Delta/T}$$

where Δ is the difference of the energy potentials before and after the pixel change. If $q > 1$ then the change is favorable and accepted. If $q < 1$ then it is accepted with probability q, which depends on the global temperature factor T. For the cooling schedule we used the suggestions in [17] (page 356), where the temperature T is set to

$$T(k) = T(1) \cdot c^{k-1}$$

where c is a constant controlling the speed of the cooling process and k denotes the current iteration. The start temperature must be sufficiently high to switch to energetically very unfavorable states with a certain probability. We set $T(1) = 2 \cdot \max(\Delta)$ which allows a change from the clique with lowest temperature (i.e. very favorable) to the clique with the highest temperature with probability 0.61.

6.3 Experimental results

This section presents the experimental results of the character segmentation and of the recognition steps. The results of the different character segmentation algorithms are presented as binary example images, as well as through goal oriented evaluation, i.e. we give the OCR results for different segmentation algorithms. Additionally, we present the goal oriented evaluation of the different robust multiple frame integration algorithms given in chapter 5.

For reasons which will become clear as we present more results, we evaluated the two character segmentation algorithms differently. We applied the binarization method using contrast maximization directly to the text images extracted with the method presented in the previous chapters. However, the text in these images proved to be too diverse for the MRF based segmentation method. Therefore, we applied this algorithm to subsampled document images of very low quality. The next two sub sections give, respectively, the results on the two methods.

6.3.1 Contrast based segmentation

In figure 6.1 we show four images taken from the set of detected and enhanced images together with results of the different binarization techniques. We decided to compare the results of our contrast-maximization algorithm with the original algorithms by Niblack and Sauvola et al., respectively, as well as with some standard binarization methods:

- Otsu's method derived from discriminant analysis (see section 3.1 on the binarization of the accumulated gradients).
- A local version of Otsu's method, where a window is shifted across the image and the threshold is calculated from the histogram of the gray values of the window.
- Yanowitz-Bruckstein's method [90], which detects the edges in image and then passes a threshold surface through the edge pixels.
- Yanowitz-Bruckstein's method including their post-processing step, which removes ghost objects by calculating statistics on the gray values of the edge pixels of the connected components.
- Niblack's method [57] with parameter $k = -0.2$.
- Sauvola et al.'s method [64] with parameters $k = 0.5$ and $R = 128$.
- Our contrast based method with parameter $a = 0.5$.

As can be seen in figure 6.1, Yanowitz-Bruckstein's edge based method is very sensitive to noise and very much depends on the quality of edge detection, even if the post-processing step is employed. Otsu's method suffers from the known disadvantages of global thresholding methods, i.e. inexact separation of text and background in case of high variations of the gray values. The windowed version improves the result (the characters are better segmented), but creates additional structure in areas where no text exists. This is due to the fact, that a threshold separating text and background is calculated even in areas where only background exists.

The same problem can be observed for Niblack's method, which segments the characters very well, but also suffers from noise in the zones without text. Sauvola's algorithm overcomes this problem with the drawback that the additional hypothesis causes thinner characters and holes. Our solution solves this problem by combining Sauvola's robustness vis-à-vis background textures and the segmentation quality of Niblack's method.

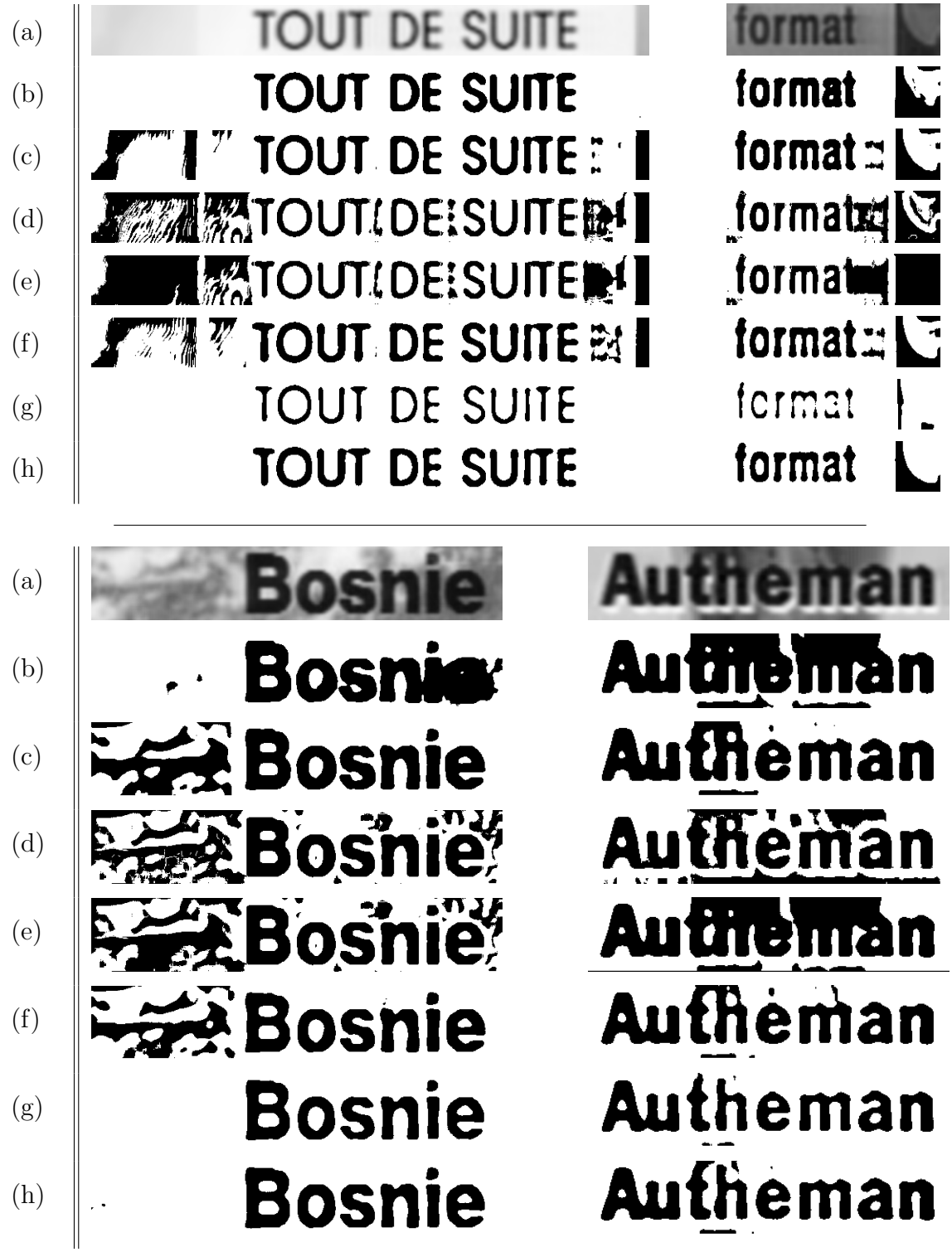


Figure 6.1: The results of the different binarisation techniques for 4 different example images: (a) the original image; (b) Otsu's method; (c) a local windowed version of Otsu's method; (d) Yanowitz and Bruckstein's method; (e) Yanowitz and Bruckstein's method including their post processing step; (f) Niblack's method; (g) Sauvola et al.'s method; (h) Our contrast based method.

Goal oriented evaluation

The algorithm has been evaluated on the video dataset described in section 5.3. The text detected by the algorithm introduced in chapters 3 and 5 is binarized and handed to a commercial OCR product (Abbyy Finereader 5.0) for recognition, processing each of the 4 MPEG files seperately. Furthermore, the rectangles for file #3, which contains contents of the French-German channel Arte, are split up into two sets containing French and German text respectively, and treated with different dictionaries during the OCR step.

To measure the OCR performance, we used the similarity measure introduced by Wagner and Fisher [79], also called "Levenstein distance" or "edit distance". The resemblance is defined as the minimal cost of transformations $\delta(x, y)$ of the ground truth string x to the corresponding output string y. The transformation is realized as a sequence of the following basic operations:

Substitution of a character:	$a \rightarrow b$	cost $\gamma(a, b)$
Insertion of a character:	$\lambda \rightarrow b$	cost $\gamma(\lambda, b)$
Deletion of a character:	$a \rightarrow \lambda$	cost $\gamma(a, \lambda)$

Under the assumption that $\delta(a, b) = \gamma(a, b)$, i.e. that the minimal distance of two characters is the cost of changing one character into another, this transformation also assigns each correctly recognized character its partner character in the ground truth string. It is therefore easy to compute the number of correctly recognized characters. To be able to compare the costs between files with different sizes, we normalize the cost by the number of characters:

$$\text{Normalized cost} = 100 \frac{\text{Cost}}{\text{Characters in the ground truth}}$$

Additionally to the normalized cost, we give the traditionally used measures of precision and recall to measure the performance of the OCR phase:

$$\text{Recall}_{OCR} = \frac{\text{Correctly recognized characters}}{\text{Characters in the ground truth}}$$

$$\text{Precision}_{OCR} = \frac{\text{Correctly recognized characters}}{\text{Characters in the OCR output}}$$

The evaluation measures, cost as well as recall and precision, depend on the character cost function, which we implemented for our experiments as follows (λ specifies the empty character):

$$\gamma(\alpha,\beta) = \begin{cases} 0 & \text{if } \alpha = \beta \quad , \quad \alpha,\beta \in X \cup \{\lambda\} \\ 0.5 & \text{if } \alpha = \beta \quad , \quad \alpha,\beta \in X \cup \{\lambda\} \\ & \alpha \text{ and } \beta \text{ have different cases} \\ 1 & \text{else} \end{cases} , \tag{6.12}$$

$$\gamma(\lambda,\beta) = \begin{cases} 0.5 & \text{if } \beta \text{ is a white space} \\ 1 & \text{else} \end{cases} \tag{6.13}$$

$$\gamma(\lambda,\beta) = \gamma(\beta,\lambda) \tag{6.14}$$

where X is the alphabet of the input characters and λ specifies the empty character.

Table 6.1a presents the results of the OCR step for different binarization methods:

- Otsu's method.
- Niblack's method with parameter $k = -0.2$.
- Sauvola et al.'s method with parameters $k = 0.5$ and $R = 128$.
- Our method with parameter $a = 0.5$.

The figures confirm the known disadvantages of global techniques, as Otsu's method. Although the precision is very high at 90%, only 47% of the rectangles are correctly recognized. Niblack obtains a rate of 80% of corrected characters. Sauvola's algorithm, which has been developed to overcome the problems of Niblack, obtains worse results. This comes from the quality of the video frames, where the contrast is not always as high as in document images, and the hypothesis assumed by Sauvola et al. does not hold. Our normalization contains the advantages of Sauvola's method without the sensitivity to contrast and gray value range. The results obtained by our method are better than Niblack's and Sauvola's.

Evaluation of the image enhancement methods

Table 6.1b shows the results of the OCR step for the two different enhancement methods presented in section 5.2, robust bi-linear interpolation and robust bi-cubic interpolation. As can be seen, the figures for the two methods are comparable. Thus, although the robust bi-cubic interpolated images are of a better quality for a human viewer, the OCR system does not transform this better quality into better recognition results.

6.3.2 Segmentation using a priori knowledge

We evaluated the performance of the MRF based character segmentation algorithm with the same goal oriented method we used for the contrast maximization algorithm: we binarized images and passed them to the same commercial OCR program Finereader 5.0. However, the test documents were taken from two well known document databases: the

Input	Bin. method	Recall	Prec.	H.Mean	N.Cost
#1	Otsu	39.6	87.5	54.5	65.1
	Niblack	79.0	78.3	78.6	43.4
	Sauvola	66.5	75.8	70.8	51.4
	Our method	**81.5**	**88.3**	**84.8**	**29.7**
#2	Otsu	54.8	**93.2**	69.0	48.5
	Niblack	92.4	79.6	85.5	33.6
	Sauvola	82.8	88.5	85.6	26.6
	Our method	**94.8**	91.5	**93.1**	**15.3**
#3-fr	Otsu	51.1	**96.1**	66.7	52.8
	Niblack	85.1	93.4	89.1	22.8
	Sauvola	61.5	68.2	64.7	64.6
	Our method	**88.1**	92.9	**90.4**	**18.1**
#3-ge	Otsu	57.5	**98.2**	72.5	43.5
	Niblack	**99.0**	93.4	96.1	09.1
	Sauvola	80.3	98.1	88.3	26.3
	Our method	98.9	97.4	**98.1**	**04.0**
#4	Otsu	45.7	84.8	59.4	59.6
	Niblack	62.8	70.5	66.4	62.8
	Sauvola	76.0	85.5	80.5	33.5
	Our method	**76.2**	**89.3**	**82.2**	**30.1**
Total	Otsu	47.3	90.5	62.1	56.8
	Niblack	80.5	80.4	80.4	40.0
	Sauvola	72.4	81.2	76.5	42.3
	Our method	**85.4**	**90.7**	**88.0**	**23.0**

(a)

Input	Enh. method	Recall	Prec.	H.Mean	N.Cost
#1	Bilinear robust	81.5	88.3	84.7	29.7
	Bicubic robust	80.5	86.5	83.4	29.2
#2	Bilinear robust	94.8	91.5	93.1	15.3
	Bicubic robust	96.7	92.8	94.7	12.5
#3-fr	Bilinear robust	88.1	92.9	90.4	18.1
	Bicubic robust	85.4	92.0	88.6	20.1
#3-ge	Bilinear robust	98.9	97.4	98.1	04.0
	Bicubic robust	97.5	95.4	96.4	05.3
#4	Bilinear robust	76.2	89.3	82.2	30.1
	Bicubic robust	82.7	87.1	84.9	28.4
Total	Bilinear robust	85.4	90.7	88.0	23.0
	Bicubic robust	86.4	89.6	88.0	22.3

(b)

Table 6.1: OCR results: (a) for different binarization methods; (b) for different image enhancement methods.

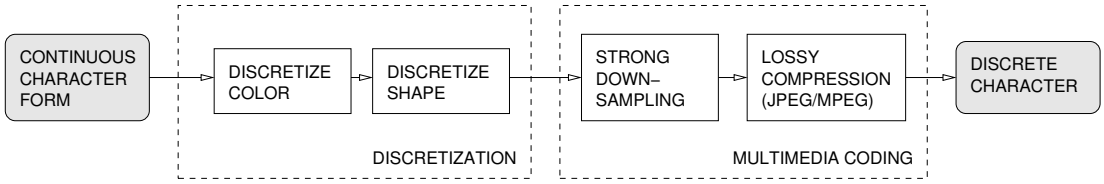


Figure 6.2: The degradation model.

Pink Panther database [89] and the University of Washington document image collection [75]. The test dataset consisted of five gray scale documents with various fonts of the same size.

To be able to perform experiments in realistic conditions we degraded the images. Several theoretical document image degradation models are known in the document image processing community. A continuous character faces several processing steps before it is transformed into its digital pendant (see figure 6.2): the continuous shape is digitized, the color is digitized and transformed into multi-band color values, the resolution of the digital image is decreased by strong downsampling, and finally strong artifacts are created by lossy data compression.

Already the first step, discretization, creates information loss by removing all information from the spectrum which is below the Nyquist limit and introduces a dependence on the position of the text on the discretization grid. However, since we wanted to concentrate the experiments on low quality text (e.g. taken from multimedia documents), we ignored the degradation following from the discretization of the continuous characters, because these effects are normally overshadowed by the stronger degradations following from the multimedia coding. To simulate these degradations we down sampled the gray scale input images by a factor of two and compressed them with the JPEG algorithm with a quality factor of 75%.

The choice of the "noiseless" training images proved to be a tough one. In fact, a perfect training image set does not exist, since only continuous characters are perfect characters. All digital training images do necessarily have to go through a discretization process, which in turn depends on the exact location of the image on the discretization grid. Hence, the same continuous character may result in a whole range of different discrete images depending on its position on the grid. We tackled this problem in a quantitative manner: using 13 synthetic documents (produced with LaTeX) as training images, we hope to cover a wide range of possible discrete forms for each character.

As already mentioned, the parameters of the segmentation algorithm were estimated from the training images. The maximum difference Δ we calculated from the clique energy function was 2.5 so we set the start temperature of the annealing algorithm to $T(1)=2\Delta=5$. About 10% of all possible clique labelings were found in the training set, the probability for the others was set to 0 if Hancock/Kittler (HK) smoothing was applied and to $\frac{1}{2}\min(P(\theta_i))$ otherwise. We ran the annealing procedure with 400 iterations and a cooling factor of $c=0.97$.

(a)   (b)

θ_b	θ_a	$V_c(\theta_b)$	$V_c(\theta_a)$	θ_b	θ_a	$V_c(\theta_b)$	$V_c(\theta_a)$
		1.05	0.95			1.82	1.38
		1.48	1.15			1.85	1.30
		2.00	1.36			2.14	1.40
		1.80	1.79			1.77	1.52
		1.87	1.16			1.84	1.57
		1.72	1.32			1.66	1.42
		2.00	1.28			2.08	1.57
		1.89	1.50			1.93	1.69
					$\sum$	28.91	22.38

(c)

Figure 6.3: (a) an example binarized with Sauvola's method; (b) with the MRF method; (c) The clique labelings of the repaired pixel.

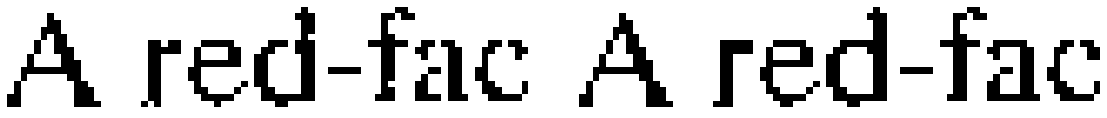
Document	1	2	3	4	5	Total
Sauvola	77.1	39.8	77.1	99.0	98.7	79.0
MRF	81.0	40.5	87.3	99.3	98.8	82.0

Table 6.2: The OCR results.

Figures 6.3a and 6.3b illustrate an example of a character repaired using the spatial information in the prior. Figure 6.3c shows the clique labelings of the repaired pixel before (θ_b) and after (θ_a) the pixel has been flipped. Note, that all 16 cliques favor the change of the pixel.

We performed experiments with HK smoothing parameters p=0 (no smoothing) and p=0.001, and with or without normalization of the clique probabilities by the factor $P_i(\theta_i)$. We could not confirm the results of Milun and Sher, which reported improvements of the results if HK smoothing was applied. On the other hand, application of the normalization step improved the quality of the results.

The OCR experiments have been conducted with two classes of images: documents binarized with Sauvola's method and documents binarized with the MRF method (normalization of the clique potentials, no HK smoothing). As can be seen from Figure 6.4, the MRF prior is capable of repairing some damage in the characters. The recognition rates are 79.0% for Sauvola's method and 82.0% for the MRF method. Table 6.2 gives the results for each document.



(a) (b)

Figure 6.4: An example: (a) binarized with Sauvola's method; (b) binarized with the MRF method.

6.4 Conclusion

In this chapter we presented two character segmentation techniques: a method maximizing the local contrast in the image and an algorithm based on Bayesian estimation. The contrast based method proved to be very efficient when applied to the enhanced images extracted from video sequences. The performance of the previous algorithms proposed by Niblack and Sauvola et al., respectively, has been improved significantly.

On the other hand, for the moment we failed in our attempts to further improve these results by taking into account a priori knowledge. The probabilistic MRF based character segmentation only slightly improves the recognition rate on a dataset with text of the same size and style (albeit different fonts). Moreover, the performance on a dataset with large differences in text size and style are not encouraging for the moment. This is partly due to the sensitivity of the method to model parameters (the noise variance) and to the energy function. We have shown that learning the clique parameters from training data does not result in an energy function invariant to changes in the text properties. We nevertheless believe that — given the nature of existing binarization techniques — further research in probabilistic modeling of text is necessary in order to create a deeper understanding of the properties of text.

Chapter 7

Semantic video indexing

Where is the wisdom we have lost in knowledge?
Where is the knowledge we have lost in information?

T.S. Eliot
American poet (1888-1965)

The TREC conference series is sponsored by the U.S. National Institute of Standards and Technology (NIST) with additional support from other U.S. government agencies. The goal of the conference series is to encourage research in information retrieval from large amounts of text by providing a large test collection, uniform scoring procedures, and a forum for organizations interested in comparing their results. TREC researchers address a number of related but different retrieval tasks. These research efforts are organized as tracks within TREC. For TREC-2001 a new track was defined - one that went beyond text retrieval. The high-level goal of the Video Retrieval Track is the investigation of content-based retrieval from digital video.

In a close collaboration, our team participated together with the team from the University of Maryland (UMD) in the video track of the year 2002's competition[1]. In a second collaboration we planned to share the results of the screen text detection with the laboratory CLIPS of the University of Grenoble. Unfortunately we had to abandon this project because of the late completion of our feature detection run (see section 7.3.4. The time schedule of that year's competition envisioned a conference procedure between February 2002 and November 2002 (see table 7.1). However, unfortunately, organizational constraints of the research exchange between Lyon and Maryland restricted the effective work on the project to the three months between July and September 2002.

The video test collection used for development, test and evaluation contained 68.45 hours of MPEG 1 video, which was taken from the *Internet Archive*[2] and the *Open Video*

[1]A part of this Ph.D. thesis has been done at the Language and Media Processing Laboratory at the University of Maryland, USA, during two exchange periods of 3 months each.

[2]http://www.archive.org/movies

Date	Milestone
February	Application was due at NIST.
May	Common shot boundary reference was made available.
July	Automatic speech recognition transcripts were made available.
August	Binary features were made available.
	Search topics were made available.
September	Feature extraction submissions and search submissions were due.
October	Results were made available.
November	Conference at NIST in Gaithersburg, USA.

Table 7.1: The time schedule of the TREC 2002 video track.

Project[3]. The movies in these collections are freely available and are not protected by copyright laws. However, a drawback is the age of the material: the videos are from the 1950s, which means that the signal quality is quite bad. Figure 7.6 shows some example keyframes of these movies, which illustrate the washed-out colors of the old material.

In order to be able to train, test and evaluate the systems on different data, the video collection had been partitioned into three sets by NIST:

- Search test collection (40.12 hrs.)
- Feature development collection (23.26 hrs.)
- Feature test collection (5.07 hrs.)

Each participating team was free to submit results to one or several different sub tasks of the video track. Our team participated in two available tasks:

- Feature extraction: detection of artificial overlay text.
- General search

At the opening of the TREC competition, a common shot boundary reference was distributed among the participants. All features refer to this common set, e.g. binary features detect whether a certain feature has been detected in a given shot and the automatic speech recognition results are given per shot. Participants were allowed and encouraged to donate feature detection results to other teams in order to make a participation in the search task possible for teams which did not have enough feature detectors available. It was also possible to donate feature results, which had been submitted to the feature extraction task.

[3]http://www.open-video.org

Non text examples	Text examples
a yen Pu s1c˜	TONY RIYERA
v\˜˜˜	EUGENE PODDANY
.˜a˜˜ 1r.	EMERY NAWK˜N5
˜-	GERALD NEYIU
.1	D i recto r
j_ I	CARL URBAN
i ˜˜˜t˜ ..˜f: a1 t.	Art Direction
ad	URANIUM
˜! ˜ ˜._’_‘	92
,(1f˜rJ1	GzOTONS 92
I˜rs˜-’.’ i˜u .r	arrtoms 142
lsYf,7	234

Table 7.2: Examples for text and non-text OCR output.

This chapter is outlined as follows: section 7.1 presents the adaptations we did on our text detection system for the TREC competition. The features used for the search task and the techniques we developed to query them are given in Section 7.2. Experiments are presented in Section 7.3, and Section 7.4 provides a conclusion on the TREC competition.

7.1 Detection of overlay text

Our team participated in the feature extraction subtask with the text detector presented in this thesis. The participating system used the framework for the detection in video sequences introduced in chapter 5 together with the text detection system based on local contrast, presented in chapter 3. The SVM based detector would have provided a higher detection precision, however, this algorithm was not yet developed at the time of the competition. Unlike in our previous experiments, for the TREC competition we used OCR software from Scansoft, which does not perform as well on our data as the Finereader software we used for the evaluation in the previous chapters, but it does come with a programmable API which makes it easier to integrate it into our system. This is imperative given the amount of data we needed during the competition.

In order to reduce the number of false alarms and to increase the precision of the detector, we developed a classifier which takes the OCR output from each detected text box and classifies it either as a positive text result or a false alarm.
Table 7.2 shows OCR output examples for text and non-text. At first glance it seems to be straightforward for a human to judge between “meaningful” text and junk strings. However, we did not want to use a dictionary based approach for two reasons:

- Screen text often contains words which cannot be found in a dictionary, such as names and identifiers containing digits.

- A part of the OCR output is very noisy (e.g. “arrtoms” instead of “atoms” in the examples).

We therefore used simple features based on the type and distribution of the characters in the string. We manually classified the set of possible characters into the 4 different groups “upper case”, “lower case”, “digits”, “non-text”. The input to the classifier is a feature vector containing the following two features:

$$F_1 = \frac{\text{Number of good characters (upper+lower+digits)}}{\text{Number of characters}}$$

$$F_2 = \frac{\text{Number of group changes}}{\text{Number of characters}}$$

We trained a linear classifier with Fisher’s linear discriminant [5] on a set of training strings and ranked all text instances by the classifier output.

7.2 General search

For the second task — general search — we used the entire set of donated features, i.e. the results of the automatic speech recognition (ASR), the binary features (outdoors, indoors, contains people, contains face, cityscape, landscape, etc..), as well as the results of the automatic text recognition (ATR) and the corresponding transcripts, and the temporal color correlograms already used during TREC 2001 and described in [60].

The main problem in video indexing and retrieval is the “semantic gap” between the way people think — and therefore formulate queries to a system — and the way computers analyze data and extract information from videos. In our case, the queries are formulated as requests for semantic concepts on the available features: for instance, “retrieve shots containing Bruce Willis”, “retrieve shots containing a paratrooper swinging on his chute”, “retrieve shots containing an airplane doing a loop”). Some features are related to semantics, as for example the outputs of the speech recognition and the text recognition, others are more related to the low level information on the raw data (temporal color correlograms). Some features are of ”semantic nature”, but one can never be sure how well the feature detectors did their job since they only have been trained statistically on low level data, e.g. detectors for outdoors scenes, landscapes, music etc.

The organisation of the search task

Three weeks before the submission deadline, 25 search topics were made available by NIST. Table 7.3 shows some of these topics. The requests came as XML files with a textual description of the request (one or two paragraphs) and links to one or several example images and videos. For instance, topic nr. 76 contained example images and videos showing Mr. Chandler talking while sitting behind his desk in his office. The

Topic	Description
75	Find shots of Eddie Rickenbacker.
76	Find shots of James Chandler.
77	Find shots of George Washington
79	Find shots of people at the beach.
80	Find shots of musicians with music playing.
82	Find shots of women standing in long dresses.
84	Find shots of the Price tower.
86	Find shots of overhead views of cities.
88	Find shots show a map of the continental US.
89	Find shots of a living butterfly.
91	Find shots of parrots.
93	Find shots of beef or dairy cattle, cows.
95	Find shots of nuclear explosions with mushroom clouds.
97	Find shots of microscopic views of living cells.
99	Find shots of a Rocket or missile taking off.

Table 7.3: A short description of some of the search topics. Each of the topics is accompagnied by one or several example images and videos.

processing of the topic requests occurred manually, i.e. the users themselves read the requests and adjusted the query system. Automatic topic processing is not yet treated by TREC.

As for the feature extraction, the reference shot detection was the common basis for the organization of the search task. The search results were submitted in form of XML files, which contained the IDs of the shots which had been found by the retrieval system.

Two different types of queries have been defined for the TREC search task. Manual runs and interactive runs (see figure 7.1). In manual runs, queries are performed without any user feedback. The user manually looks at the query request statement (i.e. no automatic query request parsing is necessary) and then chooses the type of features he wants to use and adjusts the parameters manually. The results of the first and only run are submitted. During a query in an interactive run on the other hand, a user is allowed to reformulate his queries after inspecting previous result sets or the data set.

In the following subsections we describe the different features and the query techniques necessary to use them in a query system. Finally, Sections 7.2.4 and 7.2.5 describe the two tools employed for our experiments.

7.2.1 Recognized text and speech

Recognized video text and speech transcriptions are treated in the same way by our system. For retrieval we used the Managing Gigabytes system [81] which is freely avail-

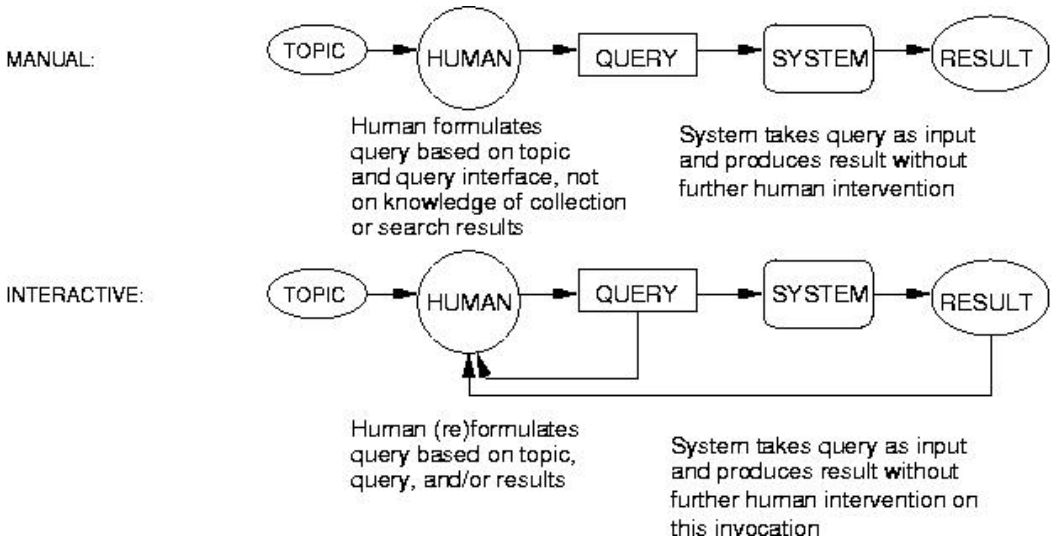


Figure 7.1: The two different types of runs defined for the search task.

able[4]. This software is able to retrieve "documents" from one or more "collections", defined by the user. In our case, documents were the text contents obtained for different shots, with the three collections corresponding to different feature sources: text, speech and text + speech. Managing Gigabytes (MG) allows users to choose between two different query metrics: Boolean queries and ranked queries, ranked queries being the default, where the relevance of each shot is determined by the cosine similarity [81].

MG stems all document and query terms, so for example, looking for the term "produced" will also retrieve shots containing the term "producer" or "producing". Unfortunately this feature cannot be turned off, so we could not measure it's influence on the query results in the case of noisy transcripts.

The MG system was developed for large collections of noise free documents, and therefore all hits are computed as exact matches (apart from the stemming technique). The ASR transcripts we received from the feature donors had been post processed with a dictionary and therefore contained only noise free text. The ATR transcripts on the other hand were quite noisy. For example the string "`Nick Chandler`"[5] had been recognized by the OCR in the shot as "`ni ck l6 tia ndler`", so a search for the keyword "chandler" would not retrieve the shot. We included a possibility of performing inexact matches using N-grams. The N-Gram of a string is the set of all sub strings of length $\geq N$. By performing a Boolean query and OR-connecting all the substrings, exact matches of these substrings will be retrieved. The minimum length N of the substrings determines the level of "exactness". Lower values of N will retrieve documents with noisier keywords but also more irrelevant documents. We set N empirically to 5.

[4]http://www.cs.mu.oz.au/mg

[5]The person "Nick Chandler" is not related to the "James H. Chandler" from TREC search topic 75.

Applied to the example given above, an inexact query on the keyword "chandler" would result in the following Boolean query:

`chand|handl|andle|ndler|chandl|handle|andler|chandle|handler|chandler`

which retrieves the desired shot but also a shot containing the transcript "`colleges cattlemen handlers of livestock`", clearly a false alarm.

7.2.2 Binary features

Ten different binary features had been specified by NIST in the framework of the TREC competition:

Outdoors	Landscape	Instrumental sound	Text overlay
Indoors	Face	Speech	
Cityscape	People	Monologue	

Users were allowed to use several features at the same time, e.g. searching for indoor shots with people and instrumental sound. Combining these features can be viewed as the classical problem of combining the results of multiple classifiers which has already received considerate attention in the pattern recognition and in the retrieval community [34, 26, 18, 47]. Training the combining classifier has been reported to considerately improve the performance of the total system in some cases, especially if the base classifiers use different features [18]. In our case, however, training data is not available since we are not allowed to use features from TREC's search test collection. Therefore, using fixed rules to combine the classifiers is the only available option.

In the following discussion, $C_{ij}(x)$ denotes the output of classifier j for class i, given the measurement vector x and $Q_i(x)$ the output of the combining classifier. We did not formulate the rules in a probabilistic way, since we cannot know if the output of the classifiers quantifies the posterior probabilities of the class memberships. The two most well known fixed combination rules [34, 18] are the product rule:

$$Q_i(x) = \prod_j C_{ij}(x)$$

and the sum rule:

$$Q_i(x) = \sum_j C_{ij}(x)$$

If the feature representations are statistically independent and the classifiers produce posterior probabilities, then the product rule will produce an optimal classifier in the sense that it quantifies the true likelihood of the measurements given class k. Unfortunately, this is rarely the case [34]. The sum rule is considered to work well in cases where base classifiers have independent noise behavior as errors in one classifier are averaged out by the other classifiers.

A third type of rule can be derived from robust statistics — the median rule:

$$Q_i(x) = \text{Median}_j \; C_{ij}(x)$$

This rule tends to ignore outliers, which introduce a bias into the decision of the classical classifiers. However, robust statistics is less efficient in cases where the number of experts is very low, as in our case. Let's keep in mind that we combine 2 - 3 classifiers for each feature.

In our case there are only two classes for each classifier: "feature present" and "feature not present". The donated classifiers specify an output for each shot, where the information given are the confidence values of the classifiers, which are scaled into the interval [0,1].

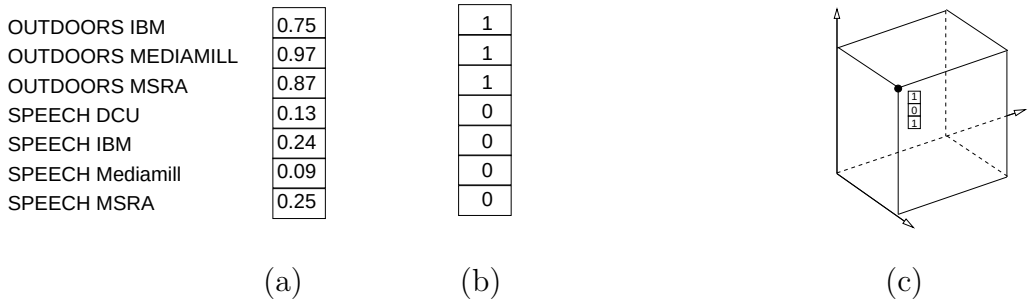
We combined the classifiers in two different ways in order to provide two different query modes. On one hand we need to combine the different classifiers from different donators for a *single* given type of feature (e.g. indoors) in order to derive a classifier output for each shot for this feature. This way, is possible to create filters on the binary features, retrieving only shots with a certain confidence of containing a feature (e.g. "Give me all shots whose confidence for being outdoors is greater or equal to 0.5".).

We chose the more robust sum rule for this purpose, since we do not have enough information on the type of classifiers used for the base classifiers and how they have been trained. If one of the classifiers is weakly trained, then the product rule will create a combining classifier with very poor results, whereas the sum rule tends to reduce the effect of a poor base classifier. If we denote with F_t the set of classifiers for the single feature type t, the output of the combining classifier is given as follows:

$$Q_i(x) = \frac{1}{|F_t|} \sum_{j \in F_t} C_{ij}(x) \tag{7.1}$$

Since we do not only want to classify the shot by the combined classifier but are also interested in the output value of the classifier, we divide the output by the number of classifiers, which gives us finally the average of the classifier outputs.

On the other hand, a user might prefer to submit a ranked query returning a set of shots "closest to " the desired features. In this case it is more convenient to use a vector space model for the set of classifiers since vector space distance functions can be used to rank the shots. The actual space depends on the query specified by the user. The user may wish to retrieve shots with constraints on certain features but may also want to ignore others. As an example, in order to retrieve people walking on a beach, a user might want to return shots containing high confidences on the features *outdoors* and *people* and low confidences on the features *indoors* and *cityscape*, but is likely to ignore the feature *speech*. In our vector space model, each dimension in the vector space corresponds to a donated feature, which is constrained in the query, so the vector space is a sub space of the space spanned by all features. If, for example, the query puts constraints on the features *outdoors* and *speech* and *outdoors* has been donated by 3 teams whereas *speech* has been donated by 4 teams, then the feature space has $N = 7$



(a) (b) (c)

Figure 7.2: An example shot vector (a) and an example query vector (b) for the query "*outdoors but not speech*". The feature space for a query on 3 features (c). The query vector is a corner point in the hypercube.

dimensions (see Figure 7.2a). For each shot s, the confidence outputs can be placed into a N dimensional feature vector u_s.

In order to rank the shots in the database, the shot vectors are compared with a query vector q. The elements of the query vector are set to 1 if the corresponding feature is desired by the user and to 0 if the corresponding feature is not desired by the user (see figure 7.2b). It does *not* make sense to provide an intermediate value of 0.5 for a feature on which the user does not have enough information, since in this case the feature is ommited from vector. As already pointed out, the feature space is a sub space of the complete space of features. The space of all possible shot vectors is therefore bounded by an N-dimensional hypercube and the query vector is one of the corners of the hypercube (see Figure 7.2c).

Given the fact that the confidence output of all feature detectors are already scaled to the interval $[0-1]$, but that the distribution of the confidence values in the interval may be different for different detectors, as is the case for the two features for *Indoors* donated by IBM and MSRA, we considered the Euclidean distance

$$D_E(q, u_s) = \{(q - u_s)^T (q - u_s)\}^{-\frac{1}{2}}$$

as well as the Mahalanobis distance

$$D_M(q, u_s) = (q - u_s)^T \sum^{-1} (q - u_s)$$

which scales the distances according to the covariances Σ of the data set. During the experiments, however, the Euclidean distance outperformed the Mahalanobis distance by far. This can be explained with the different distributions of the confidence values of the different detectors.

7.2.3 Temporal color features

We incorporated a capability for color motion search by the temporal color features developed by the University of Maryland in collaboration with the University of Oulu

and presented in [60]. These features model the statistical distribution of the colors in the shot, but, unlike histograms, they also capture the spatial and temporal distribution of the colors. A correlogram is defined as:

$$\gamma_{c_i,c_j}^{(d)} = \Pr_{p_1 \in I_{c_i}^n, p_2 \in I^n}(p_2 \in I_{c_j}^n \big| |p_1 - p_2| = d)$$

where I^n are the frames of the shot, p_1 and p_2 are two pixels and c_i and c_j are two colors. In each bin, the feature stores the probability that given any pixel p_1 of color c_i, a pixel p_2 at distance d is of color c_j among the shots frames I^n. For the Video TREC evaluation, the features have been computed as auto correlograms, and therefore $c_i = c_j$, resulting in a two dimensional structure. The distance between two features is calculated using the L_1 norm.

7.2.4 Querying

A command line interface allowed the users to submit queries on the features with the techniques described above. The query results could be stored in 30 different result frames, allowing combinations of the results using different schemes. The following operations were supported by the interface:

- Keyword based queries on text, speech or both; with or without n-grams; and Boolean or ranked.
- Ranked color queries.
- Ranked queries on binary features and filters on binary features.
- Combination of query results (AND/OR) including weighted combinations of the ranking of both queries. Assigning each query i a weight α_i, the shots in the combined set are ordered by a measure[6] m_s, which can be calculated from each shot s's ranking $r_{s,i}$ in the query results:

  $$m_s = \sum_i \alpha_i \left(1 - \frac{r_{s,i} - 1}{N + 1}\right)$$

  where N is the total amount of shots in the combined set.
- Inspection of the keyframes of queries (thumbnails and full sized image).

[6]The measure is not real ranking information since a value may occur several times in the combined set.

7.2.5 Browsing

In addition to the command line query interface, the data set was viewed graphically in a browsing tool developed at the University of Maryland. The users had the choice to either view the complete data set or to select a set of query results and to view only shots which have been returned by one or more of these queries. For each shot, the following features were available:

- The confidence of the binary features (one value per feature type).
- The text and speech recognition output and it's confidence values.
- The rank of the shot. The ranking information was scaled and normalized to make it intuitively conform to the confidence values of the binary features ("*higher means better*"): $m_s = 1 - \frac{r_{s,i}-1}{N+1}$
- A OR-combined rank-like measure computed on the ranking of the shot in all chosen queries. The weights α_i were chosen by the user when the queries are exported to the browsing tool.

Browsing allowed users to view the shots as visual objects on a dynamic two-dimensional grid. The available features could be used in multiple ways to visualize the data set. The user could visualize the shots in two dimensions by placing any two features on the horizontal and vertical axis. Additional dimensions were visualized by adding attributes to each object. Color, for example, could be used to represent a third feature dimension. Dynamic range sliders were provided for all features (not just the visible ones) so the users were able to dynamically modify the visibility of each feature and navigate through the collection. Attributes of each object could be configured and displayed as "tips" when the mouse passed near them. Additionally, by clicking on a visual object or choosing a set of objects, their keyframes could be displayed and a media player was launched to view the shot.

Figure 7.3 shows an example for a browsing session. The rank of a color query has been assigned to the y-axis and the results of a text/speech query (keyword: "president") has been assigned to the x-axis. All shots inside or touching the circle are listed in the blue boxed list, for each shot the text/speech recognition results are displayed.

7.3 Experimental Results

In accordance with the TREC task definitions, we submitted 3 different runs:

Manual run The user adjusted the query parameters in the command line query tool to manually run queries and combined them to a single query. Our users where asked to write down the manual queries on a sheet of paper before actually performing them with the tool in order to avoid potential feedback from the sub queries.

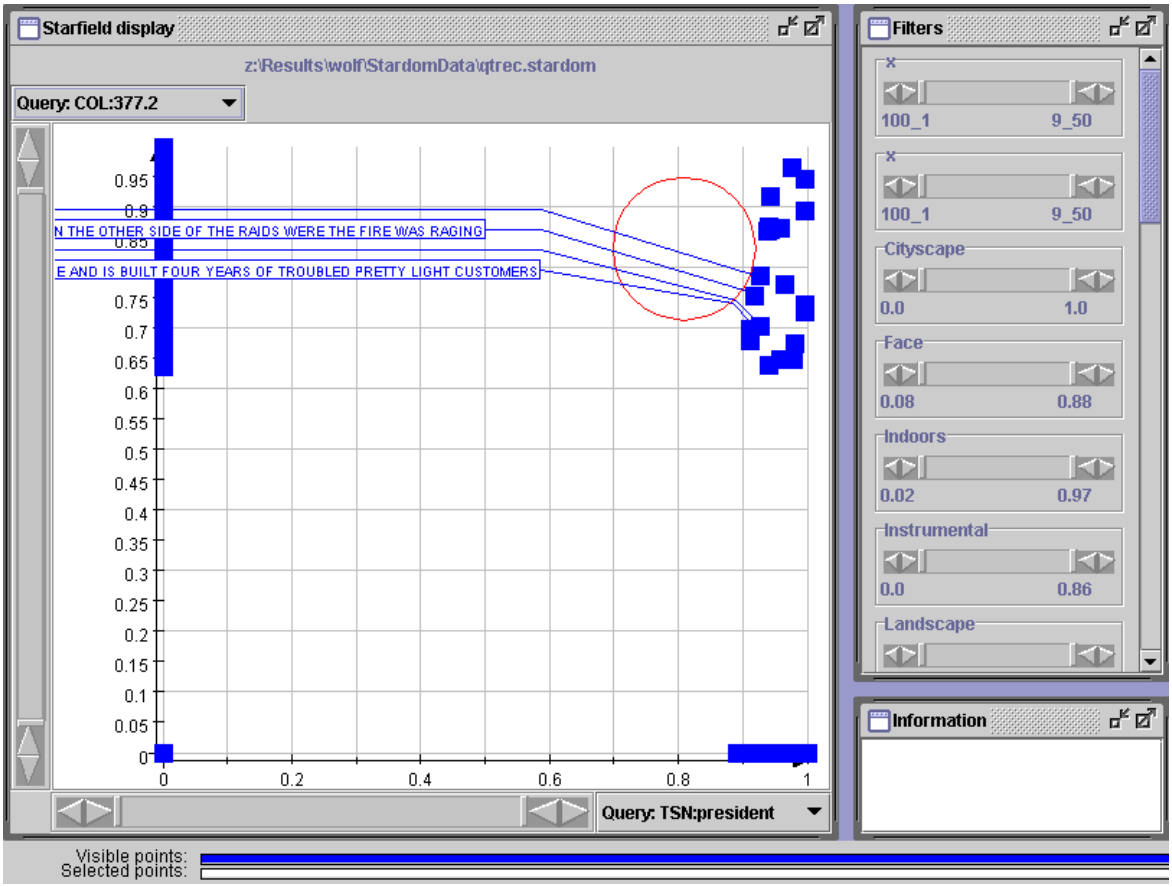


Figure 7.3: The graphical browsing tool "stardom".

Manual run/no ASR The same queries as above, but without speech recognition.

Interactive run The users were free to look at the results of each intermediate query returned by the command line tool and to adjust parameters and re-run them. Additionally, the query results could be viewed with the browsing tool for a closer inspection of the result sets and the mappings into the feature space. Based on these observations, queries could be re-run in the query tool. Hence, submitted results were queries run and combined in the query tool.

Four different users participated in the experiments. The users had been given dummy topics in order to be able to get used to the system and the features. They did not know the real topics before they started working on them.

Figure 7.4 shows the precision curves vs. result size and recall, respectively, in graphical form. Unfortunately, the generality of the database has not been provided by NIST. Since all teams used the same database, it is possible to compare the different methods with the classic precision/recall graphs in order to determine which method is the best. However, a quantitative evaluation of the differences between the methods also makes the inclusion of the database generality necessary.

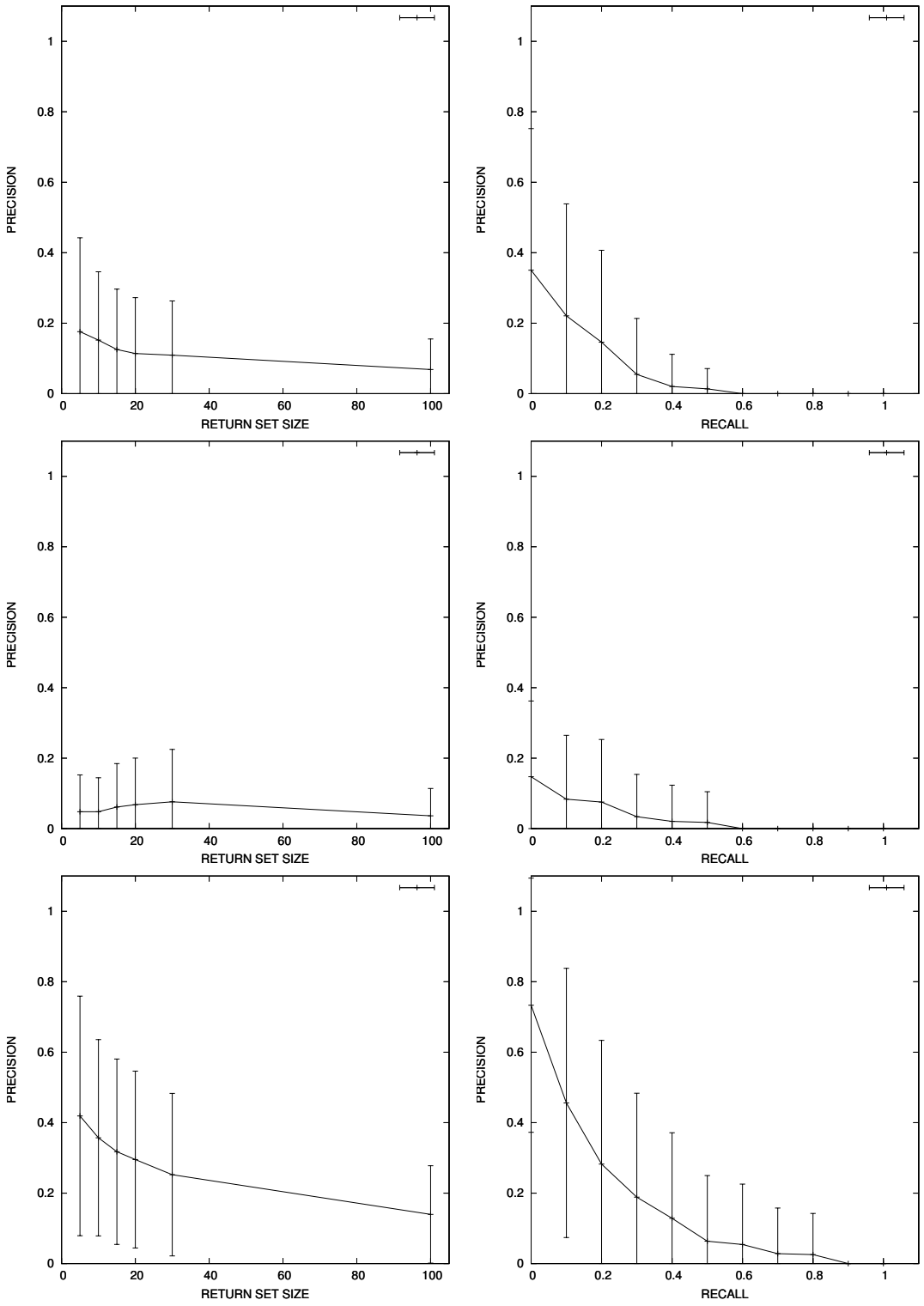


Figure 7.4: Precision at different result sizes averaged across the topics (left), and precision at different levels of recall averaged across the topics (right). From top to bottom: the manual run, the manual run without speech recognition and the interactive run.

The interactive system performed best for query topics *76-James Chandler* (54% precision at 80% recall) and *97-living cells* because of the color features, and for *89-Butterfly* because of the color features and the ASR transcripts. Topics with medium sucess were *80-Musicians*, where a combination of color and ASR queries was used, and *77-George Washington*, *78-Abraham Lincoln*, *90-mountain* and *99-rockets*, which had been tackled with ASR only. For these topics, users performed queries with different keywords which were more or less related to the topic, and filterted the output with the binary features. For topic *86-Cities* a recall of 20% was achieved by a sole usage of color features. The user fed the results of the color queries back to new color queries and combined these results.

During the experiments we had significant difficulty with the topics *79-Beach*, *87-Oil fields*, *88-Map*, *96-flag*, *98-locomotive approaching*. For four topics our experimenters could not find a single relevant shot: *75-Eddie Rickenbacker*, *81-Football players*, *85-Square arch*, *91-Parrots*. These are cases where the ASR transcripts did not return any relevant shots and the color features were not useful. Searching for *Eddie Rickenbacker*, even the usage of the context taken from the request statement (e.g. the knowledge that he is the CEO of an airline) did not help to find a shot. The color features were particularly powerless in the topics *football*, *Beach* and *locomotive*, which are specified where a connection between the semantic concept and colors is very hard. A feature detecting drawings and cartoons could have been useful for the map topic. In some cases the users missed a possibility to explicitly specify colors or color ranges (e.g. for the topics *flag* and *parrots*).

7.3.1 The impact of ASR

Naturally, as the speech recognition transcripts are very close to the semantic meaning of the video, they have a significant impact on the search results. Nevertheless, the quality of keyword based queries on the transcripts highly depends on the topic. In general, the result sets of speech queries are very heterogeneous and need to be filtered, for example, by the results of binary filters.

Consider for example the results on the keywords "rocket, missile", shown in Figure 7.5. Adding additional keywords (e.g. "taking off") could increase the chances of retrieving rockets actually taking off as opposed to rockets on desks etc., but it also "thins" the semantic meaning of the query, in this case decreasing the probability to find shots about rockets or missiles in the result set. Another possibility is filtering the results of speech queries by the results of binary queries, a method applied frequently during our experiments.

7.3.2 The impact of automatic text recognition

The TREC 2002 video data set is a very bad collection to test the usability of recognized text for video retrieval. Text detection and recognition algorithms have been developed

Figure 7.5: The results of a query on the speech transcripts using the keywords "rockets missile".

with different video material in mind such as newscasts, television journals and commercials, where text occurs much more frequently. In newscasts, the frequent appearance of names (interviewed people etc.) and locations rendered in overlay text makes indexing very powerful, especially in combination with face detection.

The TREC data set, on the other hand, comprises documentaries with rarer occurrences of text. Most of them only contain title and related information at the beginning, casting-like text at the end but rarely text in the body of the video. Another problem, although possible to circumvent, is that much of the overlay text actually occurs on blank uniform and unchanging background, which is (correctly) segmented into a single shot by the shot detection mechanism. Unfortunately, this shot is not interesting to return for most query topics. In the future, returning the shots before or after these shots may improve the quality of the results.

Figure 7.6 shows the keyframes of shots returned on keyword based queries on the result of the text recognition only. The query results displayed in this figure are a part of the actual queries run during our TREC experiments and submitted to NIST. Unfortunately, only one of the returned shots is relevant to one of the search topics: Figure 7.6a shows the keyframe of the shot returned on the keyword "music". The query definition required shots containing musicians playing music. The shot actually shows musicians playing, and overlay text rendered over the scene.

Examining the query results of our experiments we found out, that there have been

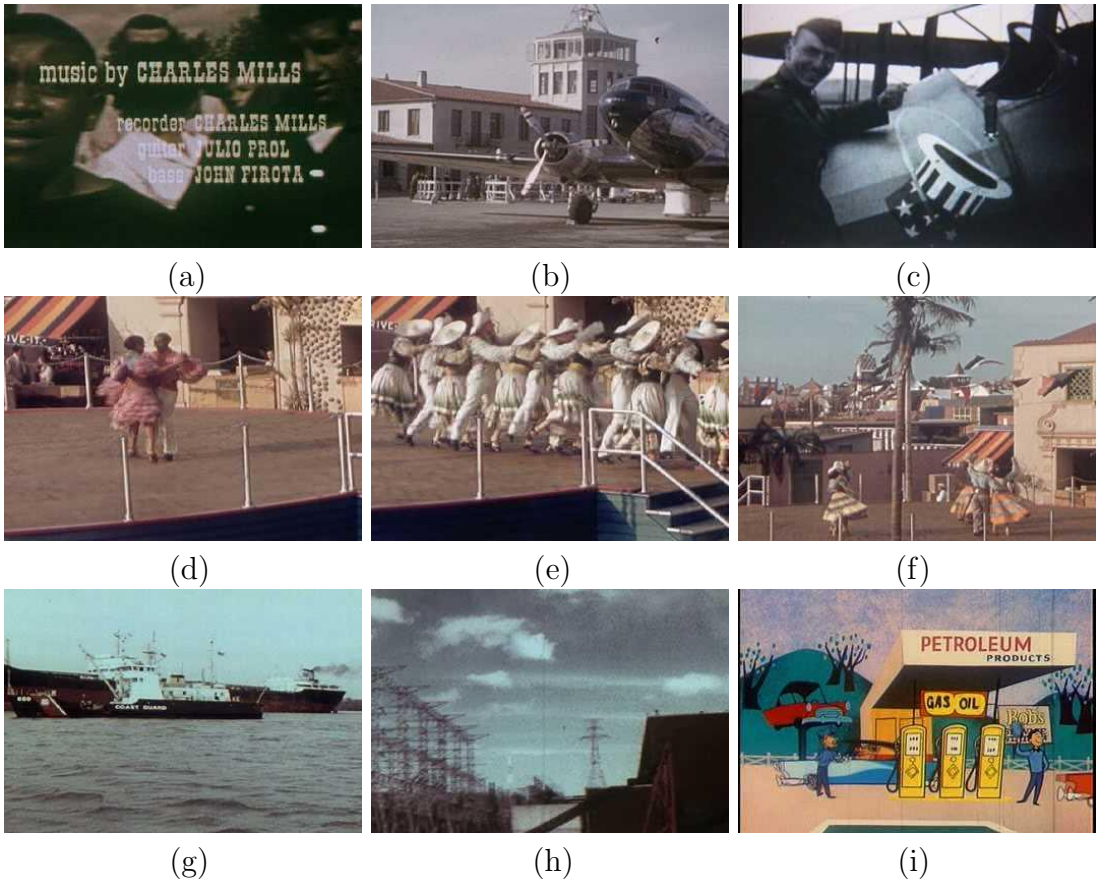


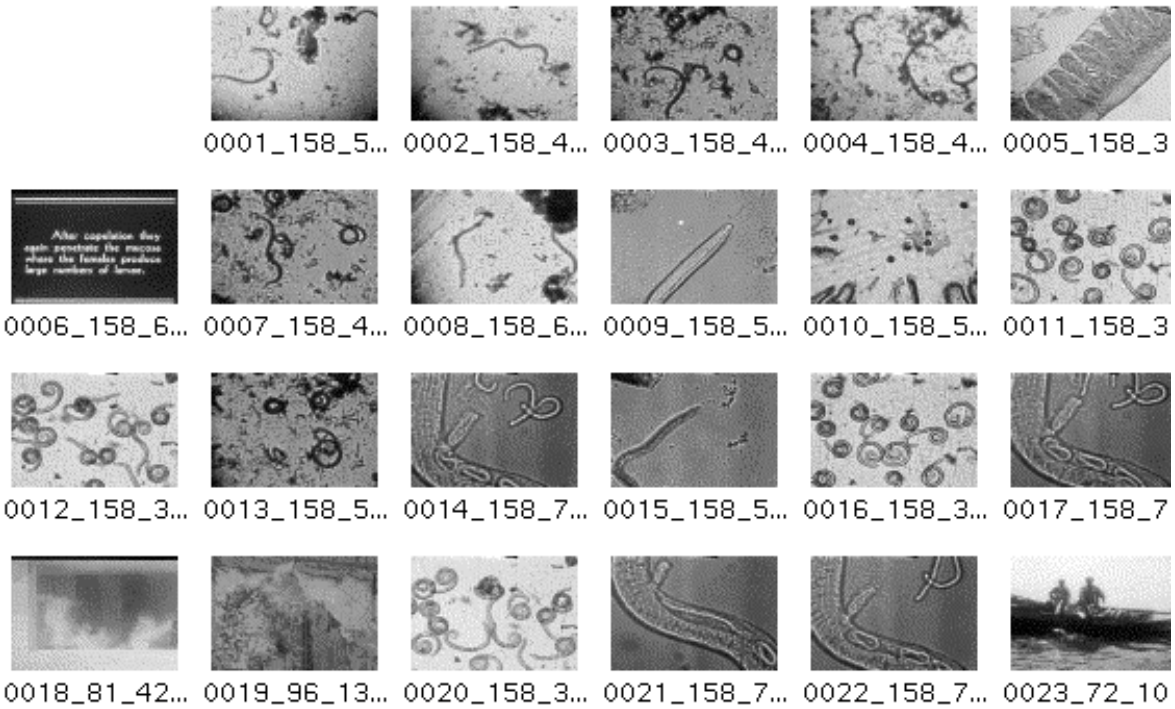
Figure 7.6: Keyframes of shots returned by text recognition queries on the keywords: (a) "music"; (b) "air line"; (c) "air plane"; (d-f) "dance"; (g,h) "energy"; (i) "oil".

other query results which are actually relevant to the submitted keyword, but unfortunately they were not relevant to the topic. As an example, a query to the keyword "air line" and "air plane" actually returned planes (see figure 7.6b-c), but unfortunately the query topic required a shot of airline commander James H. Chandler and the experimenters tried to show creativity in finding these shots. Similarly, e.g. the queries for "dance" (figure 7.6d-f), "energy" (figure 7.6g-h) and "oil" (figure 7.6i) returned meaningfull shots but were not relevant to the query topic.

Finally, please note, that in many keyframes the text is not visible, but the text is never the less present in the shot.

7.3.3 The impact of Color

As expected, the color features have been very useful in cases where the query shots where very different from other shots in terms of low level features, or where the relevant shots in the database share common color properties with the example query. For



(a)

(b)

Figure 7.7: The results of color queries using example shots: (a) searching for living cells under a microscope; (b) searching for shots showing James H. Chandler.

example, if they have been shot in the same environment. Figure 7.7a, shows the result set of a color query on an example shot showing living cells under a microscope as requested for TREC search topic 97.

As another example of success, Figure 7.7b shows the results on the color queries on example shots showing James H. Chandler. Due to the special brownish color of the videos showing Mr. Chandler, the queries were very successful in retrieving shots from the same set of videos. In general, however, the color features need to be used very carefully, since no semantic meaning whatsoever is attached to them.

7.3.4 Execution time

The processing complexity of the indexing algorithm can be divided into two parts: the calculation of the features and the retrieval itself. Most of the features we used had been calculated by third parties, except for the text detection. The text detection algorithm employed at the time of participation was not yet optimized for speed. The processing of one TREC video sequence of about 20 minutes took around 17 hours on a Pentium III with 500Mhz used at that time. The processing of the whole dataset would have taken 4,110 hours, or 171 *days*! Fortunately, we had access to a multi-processor Linux cluster at the University of Maryland, which shortened the processing time to 1.5 months. Nevertheless, we finished feature extraction one week before the submission deadline of the search results, which made it impossible for our partners at CLIPS of the University of Grenoble to use them.

The execution time of the retrieval system was close to negligible: the search time for all features was under a second. An exception were searches on color features, which took around 30 second's. This is due to the implementation in the scripting language perl by the original authors of the features. Clearly, there is a large potential for optimization. Although it has been written in java, the browsing tool responded very fast as long as query result sets of up to 3000 result items were inspected. Loading the whole database of around 14,000 shots into stardom slowed down reaction times considerably. However, inspecting the whole set of shots proved to be ineffective for retrieval.

7.3.5 Discussion of the experiments

As mentioned above, the usefulness and impact of the different queries highly depends on the topics and on the example images and videos. In general, the speech recognition transcripts proved to be very important, although queries on them produced very heterogeneous outputs. The color features were only useful in isolated cases, where the targeted shots had very similar low level characteristics to the query images or videos. As expected, the binary features were very useful for filtering queries on the other features but not very useful as only features. Furthermore, the different donated features for the same feature type were not very correlated and tended to have a high amount of noise.

text/speech	*swimming*	2	OR	10000	AND 2 OR
text/speech	*shore*	1			
Landscape ≥ 0.3 Cityscape ≤ 0.5 Outdoors ≥ 0.5				1	
text/speech	*water*	2	OR		1
People≤ 0.5 Outdoors ≥ 0.5 Cityscape≤ 0.05 Indoors ≤ 0.75 Landscape≥ 0.5		1			

Table 7.4: The submitted query formulations for the interactive run of topic 79 *"People spending leisure time on the beach"*.

Table 7.4 shows an example of an interactive query submitted. The user realized, that the color features are not very useful in this case and concentrated on queries on the text and speech transcripts using different and more diverse keywords, filtered by binary features whose boundaries have been found also using the browsing tool.

7.4 Conclusion and future work

A major emphasis in our work was put on the user and the experimental part of the work. We presented a query tool retrieving shots from user requests and a graphical tool which permits to browse through the feature and query space. Our future research will include:

- Exploiting the temporal continuities between the frames, e.g. as already proposed by the Dutch team during TREC 2001 [72]. This seems to be even more important using overlay OCR results, since text is often rendered on neighboring shots.
- Combining the binary features (normalization, robust outlier detection etc.).
- Improving the graphical browsing interface. As our experiments showed, quick access to the keyframes and to the movie itself is a keystone in the success of the browsing tool. A browser showing tiny and enlargeable keyframes in the viewing grid instead of video objects should remarkably improve the performance of interactive queries.
- Adding additional features: e.g. explicit color filters, query by sketched example, motion, etc.

The most promising research in video retrieval points into the direction of the use of different features in order to attack the semantic gap from a materialistic point of view. Techniques combining different sources seem to be very successful [52, 26, 55]. However, sophisticated methods combining the large amounts of features are still missing and more theoretical research needs to be done in this area.

Chapter 8

Conclusion générale

Es ist nicht das Ziel der Wissenschaften,
der unendlichen Weisheit eine Tür zu öffnen,
sondern eine Grenze zu setzen dem unendlichen Irrtum.

The aim of science
is not to open the door to infinite wisdom,
but to set a limit to infinite error.

Bertold Brecht
German writer and director (1898 - 1956)

Dans ce travail nous avons présenté des nouvelles méthodes pour détecter et traiter du texte dans les images et les séquences vidéos. Nous avons proposé une approche intégrale pour la détection dans les vidéos, commençant par la localisation du texte dans des frames simples, le suivi, l'amélioration de l'image à partir de plusieurs frames et la segmentation des caractères avant de les passer à un produit d'OCR commercial. Un algorithme pour exploiter le texte pour l'indexation sémantique des vidéos a été introduit dans ce travail.

La détection

Pour la détection du texte dans les images ou dans les frames pris d'une vidéo, nous avons proposé deux algorithmes qui sont basés sur des philosophies différentes. Par conséquence, ces algorithmes n'ont pas les mêmes propriétés et caractéristiques — ils doivent être appliqués à des données de type différents.

L'avantage du premier l'algorithme basé sur le contraste local est sa haute performance dans le cas où il est appliqué aux images qui contiennent du texte. C'est pourquoi nous recommandons son usage pour les images, puisque dans la majorité des cas on sait qu'elles contiennent du texte.

L'avantage du deuxième l'algorithme basé sur l'apprentissage est sa grande précision, surtout quand il est appliqué à des données d'une faible généralité. Nous conseillons l'application aux séquences vidéos, qui dans la majorité des cas contiennent plus de frames sans texte que de frames avec du texte. De plus, par apprentissage, cet algorithme peut être adapté aux différents types de texte. Cela peut s'avérer important dans les cas où un opérateur connaît a priori le type de source, par exemple une chaîne de télévision.

Les deux méthodes de détection sont capables d'exploiter plusieurs propriétés du texte. Parmi ces propriétés nous nous servons des caractéristiques géométriques. Dans le cas du deuxième algorithme, ces caractéristiques sont introduites très tôt dans le processus de détection — c.à.d. avant la classification — contrairement aux algorithmes existants, qui imposent des contraintes géométriques dans une phase de post traitement.

Dans ce contexte, les perspectives de notre travail portent sur l'amélioration ultérieure des caractéristiques, e.g. la normalisation par rapport au contraste global dans l'image, l'intégration de l'analyse de texture géométrique [36] et une amélioration de la complexité du calcul. En effet, en comparant les caractéristiques de l'algorithme sans optimisation avec la version optimisée, nous constatons une perte de performance due à l'amélioration de la vitesse de calcul, nécessaire pour réduire la complexité à un facteur raisonnable. Ces changements concernent le sous-échantillonnage des pixels à classifier, le sous-echantillonage des pixels utilisés pour le calcul des caractéristiques et l'approximation du modèle SVM.

Une autre perspective de notre travail concerne la généralisation des méthodes à une classe plus large de texte. Actuellement, nos algorithmes ne sont pas restreint sur le texte artificiel. Par contre, les textes aux orientations générales ne sont pas encore traités. Bien que nous ayons conçu les caractéristiques et l'algorithme de classification pour une classe de texte aux orientations multiples, une adaptation de l'algorithme de post traitement doit être réalisée. De plus, la détection de texte de scène déformé demande la conception d'un module de détection et de correction d'orientation et de déformation.

Enfin, comme nous l'avons déjà mentionné, notre travail a été adapté aux mouvements linéaires de type générique de film [49]. Une extension ultérieure aux mouvements plus complexes peut être envisagée.

La détection et l'extraction de texte semble avoir atteint une certaine maturité. Nous croyons que dans l'avenir l'intégration des différentes méthodes, e.g. des méthodes statistiques et des méthodes structurelles, sera à l'origine des améliorations de la performance de détection, surtout dans les cas où le type de données n'est pas connu a priori. A titre d'exemple on peut considérer, dans un cadre hiérarchique, l'application des méthodes structurelles à un bas niveau pour confirmer la détection de texte par des méthodes basées sur l'analyse de texture et de contour appliquées sur un niveau plus haut.

La binarisation

Nous avons proposé deux méthodes différentes pour la segmentation de caractères. Nous conseillons la méthode basée sur la maximisation du contraste local pour l'application au texte extrait des séquences vidéos. Grâce à la prise en compte des paramètres statistiques globaux, une meilleure adaptativité à la variété des contenus des vidéos est obtenue. Le prix à payer se situe au niveau computationnel puisque ces paramètres supplémentaires requièrent un traitement en deux passes. Cependant, ce coût est très faible au regard, d'une part du gain sur les performances, et d'autre part du coût global incluant les phases de détection des boites de texte et de reconnaissance par le logiciel d'OCR.

Cette nouvelle méthode n'est *a priori* pas définitive en ce sens que d'autres paramètres peuvent être introduit. A titre d'exemple, nous avons évoqué (cf. Eq. 8.8) le cas où la variabilité locale est faible ($s \ll R$). Il serait intéressant de pas lier cette propriété à la seule comparaison de s à la valeur maximale mais également à la valeur minimale de l'écart type. Cette autre valeur extrême permet de quantifier le bruit additif à l'image et par là même d'accéder au rapport signal/bruit de l'image. De nombreuses techniques existent pour l'estimation de cette variance minimale (par exemple [50]). Elles sont souvent coûteuses en temps et leur adaptativité aux spécificités de la vidéo doit être préalablement étudiée.

Au delà du seul objectif de reconnaissance, nous souhaitons aussi étudier l'apport de ces indicateurs pour filtrer plus efficacement les fausses alarmes, *i.e.* les boites de texte qui ne contiennent pas de texte, et qui induisent un surcoût de traitement dans la phase de reconnaissance du texte. Notre approche s'appuie sur une formalisation en termes de contraste et peut donc être reliée à tous les travaux, très nombreux en traitement d'images. En particulier, il sera nécessaire de faire le lien avec la méthode de Kholer [32].

La reconnaissance

La majorité des algorithmes connus dans la littérature réalise la reconnaissance des caractères à l'aide de produits commerciaux, qui font preuve d'une performance tout à fait satisfaisante dans le cas de texte d'une qualité raisonnable. Cependant, la plupart de ces produits étaient conçus pour des documents de papier scannés, une adaptation des données est donc nécessaire. Dans la communité de la reconnaissance optique des caractères on peut constater une émergence d'algorithmes qui traitent directement des images en niveaux de gris. Etant donné que le seuillage d'images de basse résolution cause une grande perte d'information, nous croyons que dans le cas de la vidéo un gain considérable est possible par la reconnaissance de caractères à partir des images en niveaux de gris.

L'indexation par le contenu

Sur un niveau plus haut, nous voyons les challenges du domaine dans l'intégration des algorithmes de détection et de reconnaissance avec les algorithmes de l'indexation

manuelle ou automatique. La communauté de l'indexation par contenu exprime un véritable besoin d'outils et de techniques pour l'indexation sémantique. Le texte présent dans les images et dans les vidéos semble d'être une des pistes les plus prometteuses pour répondre à ces besoins à court et moyen termes. Les contributions que le texte peut apporter à l'indexation sont étudiées par des documentalistes dans le but de trouver des moyens "intelligents" pour son exploitation, i.e. des moyens qui vont au-delà de la simple utilisation des mots-clefs.

La plupart des émissions de télévision suivent une charte graphique, qui impose des régles fixes pour le texte par rapport au message qui est transporté par ce texte. Ces règles concernent sa taille, son style, sa position, ses contraintes temporelles etc. A titre d'exemple, dans les journaux télévisés les noms des gens interrogés sont affichés dans une position et un style fixe. Nous pouvons envisager différentes possibilités pour l'exploitation d'une charte graphique:

- l'information sur la charte graphique peut être exploitée pour réaliser une macro segmentation d'un journal télévisé en segments de sujet unique;
- la connaissance a priori sur les propriétés de texte dans les journaux télévisé peut être à l'origine d'une suppression contrôlée de fausses alarmes.

Concernant la dernière possibilité, un projet de collaboration entre notre équipe et l'INA est en cours. Il a pour but de contrôler les techniques de détection de texte avec la connaissance a priori disponible sur le contenu de la vidéo [37]. Les paramètres de notre système de détection sont reliés aux propriétés du texte. Ils peuvent donc être réglés par un opérateur afin d'adapter les algorithmes à la charte graphique de la vidéo pour diminuer les fausses alarmes de la détection. A titre d'exemple nous pouvons citer un paramètre spécial pour la catégorie "film sous-titré" avec une taille et position fixées des sous-titres. Des contraintes plus générales doivent être admises pour la catégorie "publicité" qui contient des textes de styles plus divers.

En perspective

En conclusion, nous pouvons dire que les défis à venir se situent à la fois dans l'utilisation du texte pour l'indexation et dans l'extraction du texte elle même. Le système que nous avons proposé a été conçu pour la détection et pour l'indexation des textes artificiels. Les systèmes futurs devront faire face à une classe plus génerale de texte et à une utilisation encore plus sophistiquée des textes extraits.

Bibliography

[1] L. Agnihotri and N. Dimitrova. Text detection for video analysis. In *IEEE Workshop on Content-based Access of Image and Video Libraries*, pages 109–113, 1999.

[2] A. Allauzen and J.L. Gauvain. Adaptation automatique du modèle de langage d'un système de transcription de journaux parlés. *Traitement Automatique des langues*, 44/1:11–31, 2003.

[3] B. Allier and H. Emptoz. Degraded character image restoration using active contours: A first approach. In *Proceedings of the ACM Symposium on Document Engineering*, pages 142–148, 2002.

[4] S. Antani, D. Crandall, and R. Kasturi. Robust Extraction of Text in Video. In *Proceedings of the International Conference on Pattern Recognition*, volume 1, pages 831–834, 2000.

[5] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1996.

[6] A.C. Bovik, M. Clark, and W.S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):55–73, 1990.

[7] S. Brès and V. Eglin. Extraction de textes courts dans les images et les documents à graphisme complexe. In *Colloque International Francophone sur l'Ecrit et le Document, Lyon*, pages 31–40, 2000.

[8] C.J.C. Burges. Simplified support vector decision rules. In *Proceedings of the 13th International Conference on Machine Learning*, pages 71–77, 1996.

[9] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

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

[11] D. Chen, H. Bourlard, and J.P. Thiran. Text identification in complex background using svm. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 621–626, 2001.

[12] D. Chen, J.M. Odobez, and H. Bourlard. Text segmentation and recognition in complex background based on markov random field. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, volume 4, pages 227–230, 2002.

[13] D. Chen, K. Shearer, and H. Bourlard. Text enhancement with asymmetric filter for video OCR. In *Proceedings of the 11th International Conference on Image Analysis and Processing*, pages 192–197, 2001.

[14] P. Clark and M. Mirmehdi. Combining Statistical Measures to Find Image Text Regions. In *Proceedings of the International Conference on Pattern Recognition*, pages 450–453, 2000.

[15] D. Crandall and R. Kasturi. Robust Detection of Stylized Text Events in Digital Video. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 865–869, 2001.

[16] Y. Cui and Q. Huang. Character Extraction of License Plates from Video. In *Proceedings of the 1997 IEEE Conference on Computer Vision and Pattern Recognition*, pages 502–507, 1997.

[17] R. Duda, P. Hart, and D. Stork. *Pattern Classification, 2nd Edition.* Wiley, New York, NY, November 2000.

[18] R. Duin. The combining classifier: to train or not to train? In IEEE Computer Society, editor, *Proceedings of the International Conference on Pattern Recognition*, volume 2, pages 765–770, August 2002.

[19] U. Gargi, D. Crandall, S. Antani, T. Gandhi, R. Keener, and R. Kasturi. A system for automatic text detection in video. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 29–32, 1999.

[20] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 11 1984.

[21] G. Giocca, I. Gagliardi, and R. Schettini. Quicklook: An Integrated Multimedia System. *Journal of Visual Languages and Computing*, 12(1):81–103, 2001.

[22] L. Gu. Text Detection and Extraction in MPEG Video Sequences. In *Proceedings of the International Workshop on Content-Based Multimedia Indexing*, pages 233–240, 2001.

[23] E.R. Hancock and J. Kittler. A Label Error Process for Discrete Relaxation. In IEEE Computer Society, editor, *Proceedings of the International Conference on Pattern Recognition*, volume 1, pages 523–528, 1990.

[24] Y.M.Y. Hasan and L.J. Karam. Morphological text extraction from images. *IEEE Transactions on Image Processing*, 9(11):1978–1983, 2000.

[25] H. Hase, T. Shinokawa, M. Yoneda, M. Sakai, and H. Maruyama. Character string extraction from a color document. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 75–78, 1999.

[26] G.E. Hinton. Products of experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks*, volume 1, pages 1–6, 1999.

[27] O. Hori. A video text extraction method for character recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 25–28, 1999.

[28] N. Huijsmans and N. Sebe. Extended Performance Graphs for Cluster Retrieval. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 26–32, 2001.

[29] A.K. Jain and B. Yu. Automatic Text Location in Images and Video Frames. *Pattern Recognition*, 31(12):2055–2076, 1998.

[30] J.M. Jolion. The deviation of a set of strings. *Pattern Analysis and Applications*, 2004. (to appear).

[31] K. Jung. Neural network-based text location in color images. *Pattern Recognition Letters*, 22(14):1503–1515, 2001.

[32] R. Kholer. A segmentation system based on thresholding. *Computer, Graphics and Image Processing*, 15:241–245, 1981.

[33] K.I. Kim, K. Jung, S.H. Park, and H.J. Kim. Support vector machine-based text detection in digital video. *Pattern Recognition*, 34(2):527–529, 2001.

[34] J. Kittler, M. Hatef, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, March 1998.

[35] J. Kittler and J. Illingworth. Minimum error thresholding. *Pattern Recognition*, 19(1):41–47, 1986.

[36] P. Kruizinga and N. Petkov. Nonlinear operator for oriented texture. *IEEE Transactions on Image Processing*, 8(10):1395–1407, 1999.

[37] R. Landais, C. Wolf, L. Vinet, and J.M. Jolion. Utilisation de connaissances a priori pour le paramétrage d'un algorithme de détection de textes dans les documents audiovisuels. Application à un corpus de journaux télévisés. In *14ème congrès francophone de reconnaissance des formes et intelligence artificielle*, 2004.

[38] F. LeBourgeois. Robust Multifont OCR System from Gray Level Images. In *Proceedings of the 4th International Conference on Document Analysis and Recognition*, pages 1–5, 1997.

[39] C.M. Lee and A. Kankanhalli. Automatic extraction of characters in complex scene images. *International Journal of Pattern Recognition and Artificial Intelligence*, 9(1):67–82, 1995.

[40] H. Li and D. Doerman. Text Enhancement in Digital Video Using Multiple Frame Integration. In *Proceedings of the ACM Multimedia*, pages 19–22, 1999.

[41] H. Li and D. Doerman. Superresolution Enhancement of Text in Digital Video. In *Proceedings of the International Conference on Pattern Recognition*, volume 1, pages 847–850, 2000.

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

[43] J. Liang, I.T. Phillips, and R.M. Haralick. Performance evaluation of document layout analysis algorithms on the UW data set. In *Document Recognition IV, Proceedings of the SPIE*, pages 149–160, 1997.

[44] R. Lienhart. Automatic Text Recognition for Video Indexing. In *Proceedings of the ACM Multimedia 96, Boston*, pages 11–20, 1996.

[45] R. Lienhart and A. Wernike. Localizing and Segmenting Text in Images, Videos and Web Pages. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(4):256–268, 2002.

[46] S.M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. ICDAR 2003 robust reading competitions. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, volume 2, pages 682–687, 2003.

[47] C.C. LuVogt. Learning to listen to the right information retrieval system. *Pattern Analysis and Applications*, 5(2):145–153, 2002.

[48] V.Y. Mariano and R. Kasturi. Locating uniform-colored text in video frames. In *Proceedings of the International Conference on Pattern Recognition*, volume 4, pages 539–542, 2000.

[49] D. Marquis and S. Brès. Suivi et amélioration de textes issus de génériques vidéos. In *Journées d´Études et d´Echanges "Compression et Réprésentation des Signaux Audiovisuels"*, pages 179–182, 2003.

[50] P. Meer, J.M. Jolion, and A. Rosenfeld. A Fast Parallel Algorithm for Blind Estimation of Noise Variance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):216–223, 1990.

[51] R. Megret and D. DeMenthon. A survey of spatio-temporal grouping techniques. Technical Report CS-TR-4403, Language and Media Processing Laboratory, University of Maryland, August 2002.

[52] K. Messer, W. Christmas, and J. Kittler. Automatic sports classification. In IEEE Computer Society, editor, *Proceedings of the International Conference on Pattern Recognition*, volume 2, pages 1005–1008, August 2002.

[53] D. Milun and D. Sher. Improving Sampled Probabilty Distributions for Markov Random Fields. *Pattern Recognition Letters*, 14(10):781–788, 1993.

[54] G. Myers, R. Bolles, Q.T. Luong, and J. Herson. Recognition of Text in 3D Scenes. In *Fourth Symposium on Document Image Understanding Technology, Maryland*, pages 85–99, 2001.

[55] M.R. Naphade and T.S. Huang. Semantic Video Indexing using a probabilistic framework. In *Proceedings of the International Conference on Pattern Recognition 2000*, pages 83–88, 2000.

[56] P. Natarajan, B. Elmieh, R. Schwartz, and J. Makhoul. Videotext OCR using hidden Markov models. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 947–951, 2001.

[57] W. Niblack. *An Introduction to Digital Image Processing*, pages 115–116. Englewood Cliffs, N.J.: Prentice Hall, 1986.

[58] E. Osuna and F. Girosi. Reducing the run-time complexity in support vector machines. In C.Burges B. Schölkopf and A. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 271–284. MIT Press, Cambridge, MA, 1999.

[59] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979.

[60] M. Rautiainen and D. Doermann. Temporal color correlograms in video retrieval. In IEEE Computer Society, editor, *Proceedings of the International Conference on Pattern Recognition*, volume 1, pages 267–270, August 2002.

[61] Y. Rui and T.S. Huang. *Principles of Visual Information Retrieval*, chapter Relevance Feedback Techniques in Image Retrieval, pages 219–258. Springer Verlag, 2001.

[62] P.K. Sahoo, S. Soltani, and A.K.C. Wong. A Survey of Thresholding Techniques. *Computer Vision, Graphics and Image Processing*, 41(2):233–260, 1988.

[63] T. Sato, T. Kanade, E.K. Hughes, M.A. Smith, and S. Satoh. Video OCR: Indexing digtal news libraries by recognition of superimposed captions. *ACM Multimedia Systems: Special Issue on Video Libraries*, 7(5):385–395, 1999.

[64] J. Sauvola, T. Seppänen, S. Haapakoski, and M. Pietikäinen. Adaptive Document Binarization. In *International Conference on Document Analysis and Recognition*, volume 1, pages 147–152, 1997.

[65] B. Schölkopf, A. Smola, R.C. Williamson, and P.L. Bartlett. New support vector algorithms. *Neural Computation*, 12(5):1207–1245, 2000.

[66] M. Seeger and C. Dance. Binarising camera images for OCR. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 54–58, 2001.

[67] B.K. Sin, S.K. Kim, and B.J. Cho. Locating characters in scene images using frequency features. In *Proceedings of the International Conference on Pattern Recognition*, volume 3, pages 489–492, 2002.

[68] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[69] K. Sobottka, H. Bunke, and H. Kronenberg. Identification of text on colored book and journal covers. In *Proceedings of the 5th International Conference on Document Analysis and Recognition*, pages 57–62, 1999.

[70] C.L. Tan and P.O. NG. Text extraction using pyramid. *Pattern Recognition*, 31(1):63–72, 1998.

[71] X. Tang, X. Gao, J. Liu, and H. Zhang. A Spatial-Temporal Approach for Video Caption Detection and Recognition. *IEEE Transactions on Neural Networks*, 13(4):961–971, 2002.

[72] The Lowlands Team. Lazy users and automatic video retrieval tools in (the) lowlands. In NIST, editor, *The Tenth Text REtrieval Conference, TREC 2001*, pages 159–168, 2001.

[73] P. Thouin, Y. Du, and C.I. Chang. Low Resolution Expansion of Gray Scale Text Images using Gibbs- Markov Random Field Model. In *2001 Symp. on Document Image Understanding Techn., Columbia, MD*, pages 41–47, 4 2001.

[74] O.D. Trier and A.K. Jain. Goal-Directed Evaluation of Binarization Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1191–1201, 1995.

[75] Univ. of Washington. Document image database collection. Intelligent Systems Laboratory, Dept. of Electrical Engineering. 352500 U. of Washington, Seattle, WA 98195.

[76] C.J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 2nd edition, 1979.

[77] V.N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., 1998.

[78] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 2nd edition, 2000.

[79] R.A. Wagner and M.J.Fisher. The string to string correction problem. *Journal of Assoc. Comp. Mach.*, 21(1):168–173, 1974.

[80] A. Wernike and R. Lienhart. On the segmentation of text in videos. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME) 2000*, pages 1511–1514, 2000.

[81] I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes - Compressing and Indexing Documents and Images*. Academic Press, San Diego, 2nd edition, 1999.

[82] C. Wolf, D. Doermann, and M. Rautiainen. Video indexing and retrieval at UMD. In National Institute for Standards and Technology, editors, *Proceedings of the text retrieval conference - TREC*, 2002.

[83] C. Wolf, J.M. Jolion, and F. Chassaing. Procédé de détection de zones de texte dans une image vidéo. Patent France Télécom, Ref.No. FR 01 06776, June 2001.

[84] C. Wolf, J.M. Jolion, and C. Laurent. Détermination de caractéristiques textuelles de pixels. Patent France Télécom, Ref.No. FR 03 11918, October 2003.

[85] C. Wolf, R. Landais, and J.M. Jolion. *Trends and Advances in Content-Based Image and Video Retrieval*, chapter "Detection of Artificial Text for Semantic Indexing". Lecture Notes in Computer Science. Springer Verlag, 2003. (to appear).

[86] E.K. Wong and M. Chen. A new robust algorithm for video text extraction. *Pattern Recognition*, 36(6):1397–1406, 2003.

[87] V. Wu, R. Manmatha, and E.M. Riseman. Finding Text In Images. In *Proceedings of the 2nd ACM International Conference on Digital Libraries*, pages 3–12, 1997.

[88] V. Wu, R. Manmatha, and E.M. Riseman. Textfinder: An automatic system to detect and recognize text in images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1224–1229, 1999.

[89] B. Yanikoglu and L. Vincent. Pink Panther: A complete environment for ground-truthing and benchmarking document page segmentation. *Pattern Recognition*, 31(20):1191–1204, 1998.

[90] S.D. Yanowitz and A.M. Bruckstein. A new method for image segmentation. *Computer Vision, Graphics and Image Processing*, 46(1):82–95, 1989.

[91] Y. Zhong, H. Zhang, and A.K. Jain. Automatic Caption Localization in Compressed Video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(4):385–392, 2000.

[92] J. Zhou and D. Lopresti. Extracting text from www images. In *Proceedings of the 4th International Conference on Document Analysis and Recognition*, pages 248–252, 1997.

[93] D. Ziou. Line detection using an optimal IIR filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):465–478, 1991.

Publications of the author

Patents

International journals

International conferences

National conferences

Technical reports

Part II

Publications en français

ORASIS 2001

Vidéo OCR - Détection et extraction du texte

Résumé

Nous présentons dans cet article les premières étapes d'un projet visant à la détection et la reconnaissance du texte présent dans des images ou des séquences vidéo. Nous insisterons ici sur la caractérisation de ce type de texte en regard des textes présents dans les documents classiques. Nous proposons un schéma de détection s'appuyant sur la mesure du gradient directionnel cumulé. Dans le cas des séquences, nous introduisons un processus de "fiabilisation" des détections et l'amélioration des textes détectés par un suivi et une intégration temporelle.

Mots-clés

Indexation de la vidéo, OCR, détection de texte

1 Introduction

Depuis quelques années, les documents audiovisuels numérisés sont de plus en plus fréquents. De grandes bases de données audiovisuelles ont été créées par des entreprises, des organisations et aussi par des personnes privées. Cependant, l'utilisation de ces bases reste encore problématique. Tout particulièrement, ces nouveaux types de données (images et vidéos) ont conduit à de nouveaux systèmes d'indexation où la recherche par le contenu se fait à partir d'une image exemple.

Les systèmes disponibles actuellement travaillent sans connaissance (systèmes pré-attentifs). Ils utilisent des méthodes de traitement d'images pour extraire des caractéristiques de bas niveau (couleur, texture, forme, etc.). Malheureusement les requêtes construites à partir de ces caractéristiques ne correspondent pas toujours aux résultats obtenus par un humain qui interprète le contenu du document.

La cause de cet échec est le manque de sémantique dans l'information extraite. Que représente la sémantique? En cherchant dans le dictionnaire Larousse on trouve: *"Sémantique : les études de sens des mots"*. Par analogie, dans le contexte de l'image, la sémantique représente donc le sens de l'image.

Considérons l'image de la figure 8.1a. Un utilisateur humain qui choisit cette image comme image requête est probablement intéressé par des images de cyclistes (notons que ne nous pouvons pas deviner le vrai désir de l'utilisateur et par conséquent, la similarité entre deux images n'est donc pas accessible). D'autre part les systèmes d'indexation trouvent des images qui sont similaires par rapport aux caractéristiques de bas niveau. Les images contenant des cyclistes qui ne sont pas similaires par rapport à ces caractéristiques (voir figure 8.1b) ne seront pas trouvées.

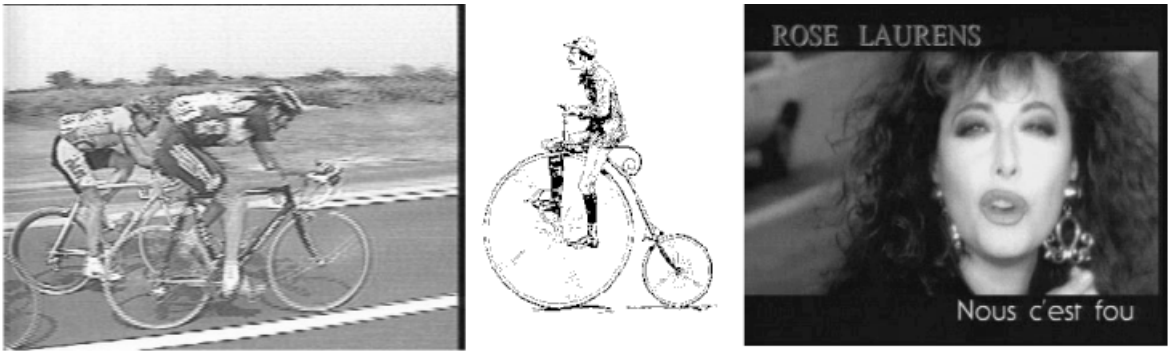


Figure 8.1: Images d'exemples.

Entre le niveau du pixel et celui de la sémantique, il existe des caractéristiques à la fois riches en information et cependant simples. Le texte présent dans les images et les vidéos fait partie de cette catégorie. La figure 8.1c montre une image extraite d'une publicité fournie en requête par un utilisateur. On peut imaginer que celui-ci cherche l'image de la chanteuse. La sémantique de cette image exemple peut être résumée par: *"Une publicité pour le nouvel album de la chanteuse Rose Laurens avec une photo de son visage"*. Il est impossible de déduire cette information à partir des caractéristiques basses. Pourtant dans ce cas le nom de la chanteuse est présent dans l'image. Par conséquent, en extrayant le texte on obtient une information supplémentaire très valable. Les requêtes classiques peuvent donc être complétées par des mots-cléfs. Le texte automatiquement extrait peut être stocké. Le processus que nous étudions dans ce travail concourt donc à une meilleure automatisation de la phase d'indexation d'images et de vidéos.

2 Formulation du problème

Les recherches en détection et l'extraction du texte à partir des séquences vidéo sont encore confrontées à de sérieux problèmes. Pourtant la recherche dans le domaine de l'OCR des documents classiques (c'est-à-dire les documents écrits, les journaux etc.) a développé de bonnes méthodes et des outils commerciaux produisent de bons résultats pour des images de documents. Le problème principal peut être expliqué par la différence

entre l'information présente dans un document et celle donnée par une séquence vidéo ainsi que les méthodes de stockage de chaque type. Nous allons décrire ces différences dans les paragraphes suivants.

Les images de documents sont créées en vue de les numériser pour les passer ensuite à une phase "OCR" pour reconnaître la structure et le texte. Pour améliorer la qualité et le taux de la reconnaissance, les images sont numérisées à très haute résolution (200-400 dpi), ce qui donne des fichiers de taille très élevée (un fichier de 100 Mo résultant d'une page A4 numérisée à 400 dpi). Les fichiers sont comprimés sans perte pour garder la qualité et empêcher des artefacts de compression. Ces grandes tailles ne sont pas une limite puisque ni leur transport ni leur stockage ne sont prévus. La plupart du temps les images de documents sont bien structurées et contiennent un fond uniforme et la couleur du texte est également uniforme. Ceci permet de séparer les caractères du fond avec un seuillage fixe ou adaptatif. La majorité de la page contient des caractères structurés dans différents paragraphes, quelques images étant incluses dans la page.

Par contre les images des séquences vidéo contiennent de l'information plus difficile à traiter. Le texte n'est pas séparé du fond. Il est soit superposé (le "texte artificiel" comme les sous-titres, les résultats de sport, etc.) soit inclus dans la scène de l'image (le "texte de scène", par exemple le texte sur le tee-shirt d'un acteur). Le fond de l'image peut être très complexe ce qui empêche une séparation facile des caractères. De plus, contrairement aux documents écrits, les séquences vidéo contiennent de l'information très riche en couleurs. Enfin, le texte n'est pas structuré en lignes, et souvent quelques mots courts et déconnectés flottent dans l'image.

Le but principal de la conception des formats de fichiers vidéo est de garder une qualité suffisante pour l'affichage, pour le stockage et le transport par des réseaux informatiques. Pour cette raison et pour une quantité de données vidéo plus importante, il est nécessaire de réduire fortement l'information avant son stockage dans le fichier vidéo. Pour limiter la taille de l'information deux méthodes sont souvent appliquées : la forte réduction de la résolution et le codage avec perte, c'est-à-dire avec élimination des données redondantes et perte d'une partie de l'information originale, en utilisant les algorithmes de compression *JPEG* et *MPEG*.

En conséquence, il est clair que les méthodes et les résultats développés pour les documents traditionnels ne peuvent être utilisés pour les documents audiovisuels. Nous allons discuter les problèmes principaux dans les paragraphes suivants.

La résolution basse

La résolution spatiale de la vidéo dépend de l'application et prend des valeurs entre 160×100 pixels (diffusion par Internet) et 720×480 pixels (DVD vidéo codée en MPEG 2). Un format typique est le format CIF avec 384×288 pixels. Notons que la taille du texte affiché avec cette résolution est beaucoup moins grande que le texte résultant d'un document numérisé. Les *frames* d'une vidéo de cette résolution contiennent des caractères d'une hauteur de moins de 10 pixels, alors que dans les documents numérisés, les caractères ont une hauteur comprise entre 50 et 70 pixels.

Ce sont également les différences de taille de la police qui rendent le processus plus

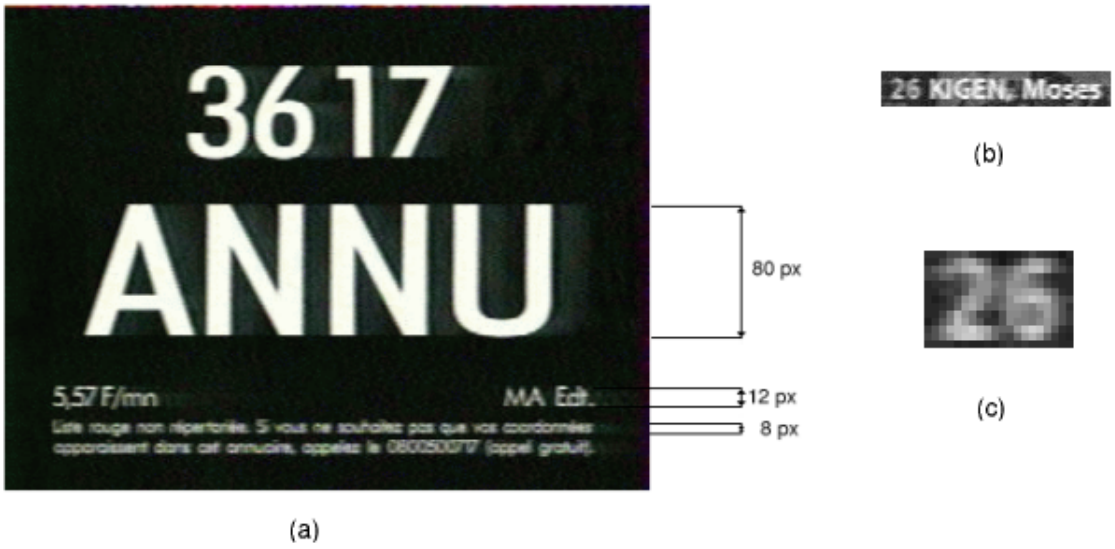


Figure 8.2: Les petites polices et la grande différence de tailles (a). Artefacts d'*anti-aliasing* (b,c).

difficile. La figure 8.2a montre une image contenant des polices entre 8 pixels et 80 pixels. Cette variété de tailles pose des problèmes pendant la phase de détection, induisant la nécessité d'une approche multi-résolution. D'un autre côté les polices de petites tailles rendent difficile la phase de reconnaissance des caractères.

Les artefacts d'*anti-aliasing* et de la compression

Pendant la production de la vidéo, le signal original est sous-échantillonné plusieurs fois. Pour empêcher des effets d'*aliasing*, des filtres passe-bas sont appliqués. Le signal résultant a alors une qualité suffisante pour la lecture mais il est complètement lissé. La figure 8.2b montre un exemple d'image extraite d'une émission de sport. La hauteur de la police du texte affiché est seulement de 11 pixels, pourtant les caractères sont suffisamment lisibles. La figure 8.2c montre deux caractères agrandis. Le lissage du signal rend la segmentation des caractères très difficile.

Après la réduction de la résolution, la compression du signal ajoute également des artefacts. L'information supprimée par le schéma MPEG est considérée comme redondante pour le système visuel humain. Cependant, les artefacts de l'encodage cause des problèmes pendant la reconnaissance (par exemple en perturbant l'uniformité des couleurs). Même si les nouvelles normes comme JPEG 2000 apportent un plus en regard de ce type de problèmes, l'existence de très nombreuses données codées selon le format JPEG classique justifie nos recherches dans cette direction.

Le fond complexe et détaillé

Le fond sur lequel est inscrit le texte ne peut être considéré comme constant. Le plus souvent, un seuillage fixe, même determiné localement, n'est pas suffisant pour clairement

mettre en évidence le texte comme le montre l'exemple des figures 8.3a et 8.3b.

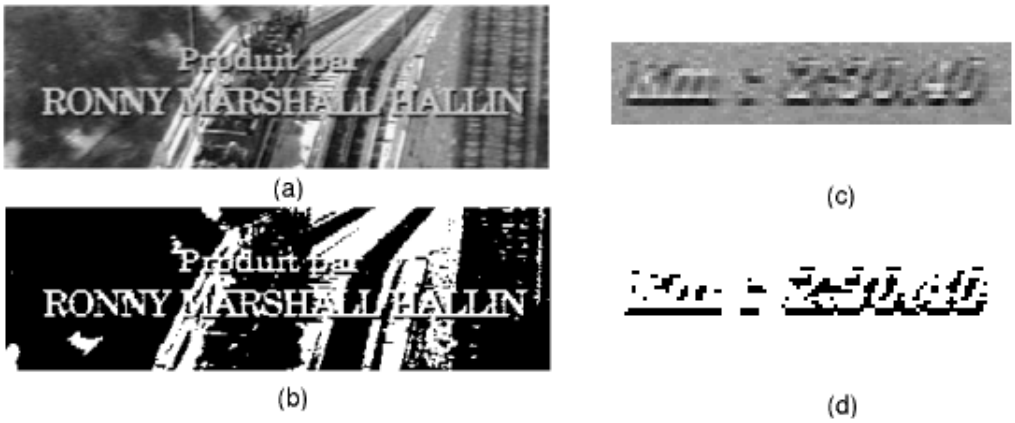


Figure 8.3: Texte sur un fond complexe (a). Un essai de seuillage fixe (b). Rehaussement artificiel du contraste (c). Un essai de seuillage fixe (d).

Le rehaussement artificiel du contraste
Afin de faciliter la lecture du texte, des artifices d'inscription du texte sont utilisés. Ces techniques ne sont malheureusement pas un avantage pour la détection comme le montrent les figures 8.3 où le texte est artificiellement rehaussé par l'ajout d'un soulignement.

3 État de l'art

L'extraction du texte des flux de vidéo est un domaine de recherche très récent mais cependant très fécond. La plupart des travaux traite le problème de la détection et de la localisation du texte (nommé "détection"). Ainsi, il existe très peu de recherches sur la reconnaissance, surtout à cause des problèmes présentés dans la section 2.

Les premières méthodes pour la détection ont été proposées comme des généralisations de techniques du domaine de l'OCR dans le cadre de l'analyse automatique des documents. La problématique de la détection est en effet proche de celle de l'analyse de la structure d'un document contenant du graphisme (journaux, page web etc, ...). A titre d'exemple, on peut citer les travaux de Jain et al. [**?**]. En supposant que la couleur des caractères est uniforme, Jain et al. utilisent une réduction des couleurs suivie d'une segmentation et d'une phase de regroupement spatial pour trouver les caractères du texte. Bien que la présence des caractères joints est prévue par les auteurs, la phase de segmentation pose des problèmes dans le cas de documents de mauvaise qualité et notamment les séquences vidéo de basse résolution.

Une approche similaire, qui donne des résultats impressionnants sur le texte de grande police, est présentée par Lienhart et al. [**?**]. Ils sont d'ailleurs parmi les premiers

qui ont présenté une étude aussi complète partant de la détection jusqu'à la reconnaissance et l'indexation. Étant mieux adapté aux propriétés de la vidéo, l'algorithme proposé combine une phase de détection par segmentation des caractères et la détection par une recherche de contraste local élevé. Ensuite une analyse de texture supprime les fausses alarmes. Le suivi du texte est effectué au niveau des caractères. Pour la reconnaissance les auteurs utilisent un produit commercialisé. Malheureusement ils ne pouvaient pas montrer l'aptitude de l'algorithme de segmentation appliqué aux textes de petite taille.

La méthode de Sato, Kanade et al. [63] repose sur le fait que le texte est composé de traits de contraste élevé. L'algorithme de détection cherche des contours groupés en rectangles horizontaux. Les auteurs reconnaissent la nécessité d'augmenter la qualité de l'image avant la phase d'OCR. Par conséquent, ils effectuent une interpolation pour augmenter la résolution de chaque boîte de texte suivie par une intégration de plusieurs *frames* (prenant les valeurs minimale/maximale des niveaux de gris). La reconnaissance est réalisée par une mesure de corrélation. Wu, Manmatha et Riseman [88] combinent la recherche des contours verticaux et l'application d'un filtre de texture.

Le travail de LeBourgeois [38] est basé sur l'accumulation des gradients horizontaux. La reconnaissance dans le système proposé est effectuée par des règles statistiques sur les caractéristiques de projections des niveaux de gris.

Li et Doermann présentent une approche d'apprentissage [42]. En laissant glisser une sous-fenêtre sur l'image, ils utilisent un réseau de neurones de type MLP pour classifier chaque pixel comme "texte" ou "non texte". Les caractéristiques sont extraites des échelles de hautes fréquences d'une ondelette de Haar. Clark et Mirmehdi utilisent aussi un réseau de neurones pour la détection du texte [14]. Ils extraient des caractéristiques diverses comme la variance de l'histogramme, la densité des contours etc.

Une méthode qui travaille directement dans le domaine de la vidéo comprimée est proposée par Zhong, Zhang et Jain [91]. Leurs caractéristiques sont calculées directement à partir des coefficients de DCT des données MPEG. La détection est basée sur la recherche des variations horizontales d'intensité, suivie par une étape de morphologie mathématique.

Les différentes approches reflètent l'état de l'art dans le domaine de *vidéo OCR*. La plupart des problèmes sont identifiés, mais pas encore résolus. Surtout, l'aspect d'amélioration du contenu, bien qu'abordé par Sato, Kanade et al. [63] et par Li et Doermann [42], ne donne pas encore des résultats tout à fait satisfaisants. Nous considérons aussi la segmentation des caractères du fond complexe comme un problème de grande importance, qui demande encore une recherche intensive.

4 Un système d'extraction

Nous pouvons décrire le but principal d'un système d'extraction de texte par quelques mots: accepter des fichiers d'images et de vidéo, détecter le texte, l'extraire et produire un fichier ASCII incluant le texte dans un format utilisable pour l'indexation. En tenant

compte de la problématique expliquée précédemment, la structure d'un système proposée dans la figure 8.4 s'impose naturellement.

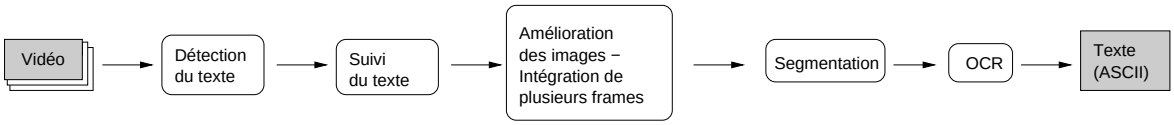


Figure 8.4: Le fonctionnement général du système.

La détection du texte est réalisée dans chaque *frame* de la vidéo[1]. Les rectangles figurant la localisation du texte sont suivis pendant leur période d'apparition pour associr les rectangles se correspondant dans les différents *frames*. Cette information est nécessaire pour améliorer le contenu de l'image, ce qui peut être atteint par l'intégration de plusieurs rectangles contenant le même texte. Cette phase doit produire des sous-images d'une qualité en accord avec les pré-requis d'un processus OCR. Par conséquent il est aussi nécessaire d'augmenter la résolution, en utilisant l'information supplémentaire prise dans la séquence d'images. La phase finale avant la reconnaissance par un algorithme d'OCR s'occupe de la segmentation des caractères en tenant compte de la complexité du fond de l'image.

Dans cette section nous proposons un système de détection et de suivi du texte. Notre système est capable de détecter et de localiser le texte dans une image ou dans un *frame*, et de suivre les occurrences du texte dans une séquence vidéo. Nous ne présenterons que peu de détails dans cette section (uniquement les principes généraux) car cette recherche est effectuée dans le cadre d'un contrat et les premiers résultats donnent lieu à dépôt de brevets[2]. Les perspectives que nous pouvons envisager pour notre projet sont donc fondées sur le développement d'un système complet qui sera capable d'extraire le texte, de le reconnaître et enfin de l'utiliser pour l'indexation.

4.1 Détection

Notre algorithme de détection repose sur une succession de traitements. Dans un premier temps, on utilise le fait que les caractères du texte forment une texture régulière contenant des contours verticaux allongés horizontalement. Une première mesure met en évidence les pixels de l'image conformes à cette hypothèse. Afin de passer des pixels à des zones compactes, nous utilisons ensuite une binarisation suivie d'un post-traitement afin d'extraire les rectangles englobants des zones de texte. Cette dernière étape permet d'atteindre plusieurs buts:

- Réduire le bruit.

[1]Un sous-échantillonnage de la vidéo peut accroître la rapidité du système mais cela réduit la facilité du suivi des zones de texte.

[2]Une démonstration en ligne pour les images statiques est accessible à l'adresse : http://telesun.insa-lyon.fr/∼wolf/demos/textdetect.html

- Corriger des erreurs de classification à partir de l'information du voisinage.
- Connecter des caractères afin de former des mots complets.

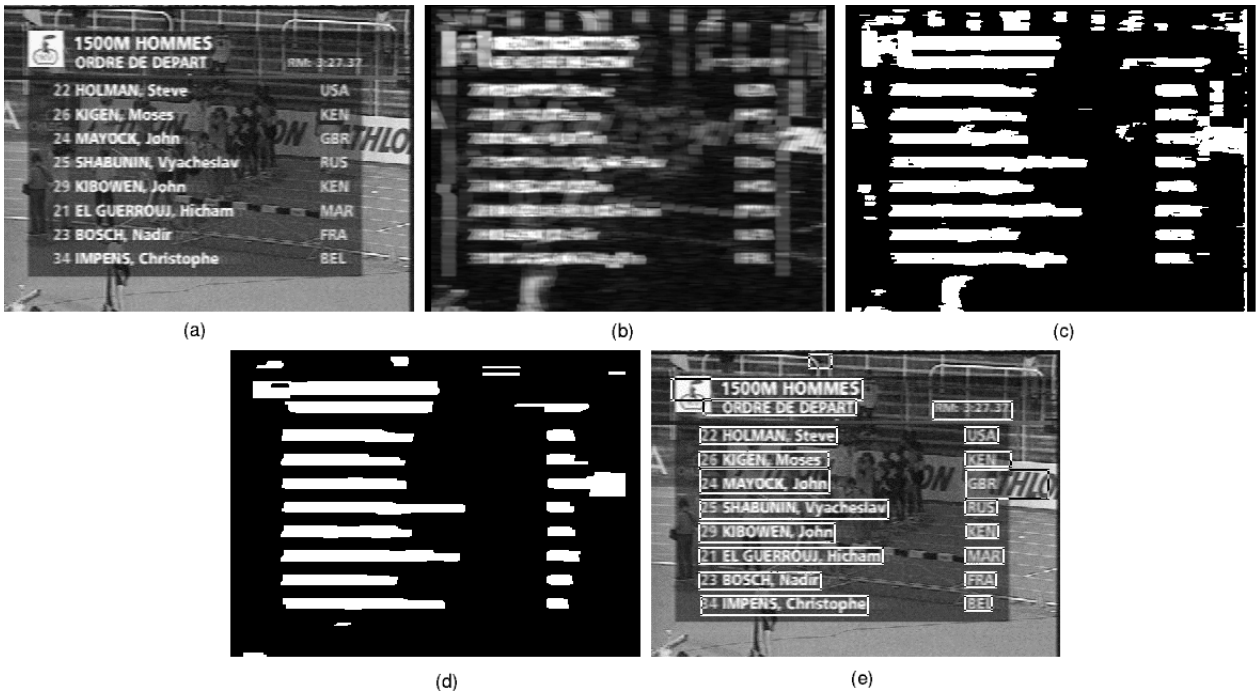


Figure 8.5: Les résultats intermédiaires pendant la détection: l'image d'entrée (a), les mesures de probabilité de présence d'un texte (b), l'image binarisée (c), l'image après le post-traitement (d), les rectangles finaux (e).

L'effet principal de cette phase réside dans la séparation du texte de la plupart des fausses alarmes. Ensuite, nous effectuons une analyse de chaque composante connexe pour trouver celles qui correspondent aux zones de texte. A ce stade, nous passons du traitement global de l'image au traitement local par composante. Chaque région est vérifiée en imposant des contraintes sur sa géométrie de façon à encore réduire le nombre de fausses alarmes. A ce stade, une suite de tests est effectuée afin de prendre en compte les spécificités de polices particulières ou d'effets d'incrustation.

A partir de ce point le traitement des composantes de texte s'effectue au niveau de leurs boîtes englobantes seulement. Pour cela, nous avons choisi le terme "*rectangle*". Le résultat de la détection est donc une liste de rectangles. Dans le cadre global du système, nous construisons cette liste des rectangles détectés pour chaque *frame* de la séquence. Les listes sont passées au module de suivi pour détecter les occurrences de texte pendant plusieurs *frames* de la vidéo.

La figure 8.5 montre des images de résultats intermédiaires pendant les différentes phases de la détection. L'image d'entrée est prise dans une émission de sport. L'image de probabilité de présence d'un texte (figure 8.5b) montre la forte réponse du filtre sur les zones de texte, mais également sur les endroits possédant des textures régulières (la grille). Bien que ce bruit reste présent dans l'image binarisée (figure 8.5c), la phase de post-traitement a réussi à le supprimer (figure 8.5d). Le résultat final est affiché sur la figure 8.5e.

4.2 Suivi du texte

L'algorithme de détection produit une liste de rectangles détectés par *frame*. Le but de notre module de suivi est l'association des rectangles correspondants afin de produire des apparitions de texte pendant plusieurs *frames* de la vidéo. Ceci est réalisé en gardant une liste L des apparitions actuelles dans la mémoire, initialisée avec les rectangles du premier *frame*. Pour chaque *frame* i, nous comparons la liste des rectangles L_i détectés avec cette liste pour vérifier si les rectangles détectés font partie d'une apparition déjà existante.

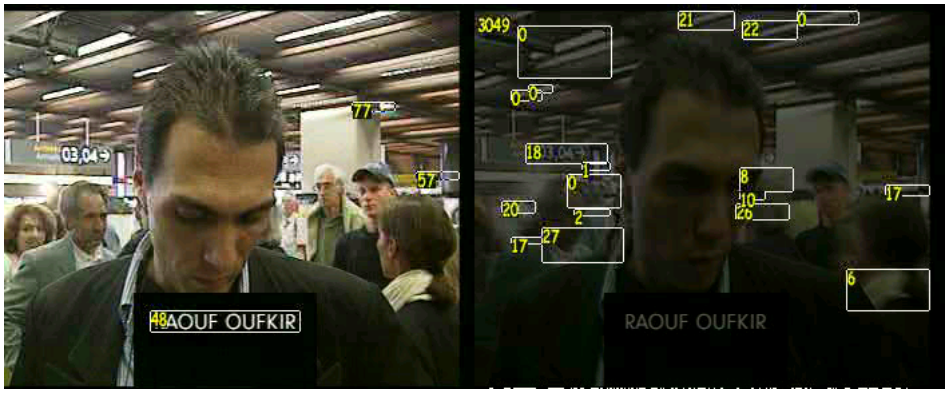


Figure 8.6: Suppression des fausses alarmes.

Les résultats de la phase de suivi sont des occurrences du texte contenant de l'information sur la localisation du texte dans chaque *frame*. La longueur de l'apparition, c'est-à-dire le nombre de *frames* où elle apparaît, nous sert comme mesure de stabilité du texte. En supposant que chaque occurrence de texte doit rester à l'écran un minimum de temps, nous considérons les apparitions de longueur plus courte qu'un seuil fixe comme des fausses alarmes. La figure 8.6 montre dans l'image gauche les rectangles considérés comme stables d'un *frame* exemple, et dans l'image droite les rectangles associés aux apparitions trop courtes, donc considérés comme des fausses alarmes.

4.3 Résultats

Pour l'évaluation de la détection statique dans chaque *frame* nous avons créé une base de test. La table 8.1 présente le taux de détection atteint. Nous avons imposé des critères d'évaluation stricts, ce qui entraîne que les rectangles qui n'ont pas été détectés entièrement, n'ont pas été comptabilisés dans le taux de reconnaissance. La figure 8.7 montre quelques exemples de texte que nous considérons comme détectés. La suite de notre travail impliquera également la création d'une base de vérité terrain pour l'évaluation du processus temporel.

Nombre de rectangles dans la vérité terrain	339
Nombre de rectangles détectés	306
Pourcentage	90.48%

Table 8.1: Les résultats.

5 Temps d'exécution

Le temps d'exécution de notre algorithme est présenté dans la table 8.2. Le système est implementé en C++ (non optimisé) sur Linux et le temps est spécifié pour un processeur Intel Pentium III de 700 Mhz. Les chiffres donnés correspondent à l'exécution d'un *frame*.

Décodage MPEG + conversion d'image	0.05 s
Intégration initiale + stockage interne	0.12 s
Détection	0.46 s
Suivi	0.002 s
Total	0.64 s

Table 8.2: Temps d'exécution par *frame*.

6 Conclusion

Le travail présenté dans cet article constitue la première phase de notre projet. Les différents éléments de l'état actuel de notre système doivent être optimisés mais la structure générale est maintenant stable. La continuité de ce travail se situe dans la phase de reconnaissance. Nous avons déjà procédé à des essais sur les principaux outils commerciaux de reconnaissance de caractères [**?**]. De cette étude, nous pouvons déduire les caractéristiques minimales que doivent valider les textes inclus dans la vidéo pour

être reconnus. Si les caractéristiques sont trop contraignantes, il sera alors nécessaire de développer un outil spécifique de reconnaissance.

Figure 8.7: Quelques résultats de détection statique.

References

CORESA 2001

Vidéo OCR - Détection et extraction du texte

Résumé

Les systèmes d'indexation ou de recherche par le contenu disponibles actuellement travaillent sans connaissance (systèmes pré-attentifs). Malheureusement les requêtes construites ne correspondent pas toujours aux résultats obtenus par un humain qui interprète le contenu du document. Le texte présent dans les vidéos représente une caractéristique à la fois riche en information et cependant simple, cela permet de compléter les requêtes classiques par des mots clefs.

Nous présentons dans cet article un projet visant à la détection et la reconnaissance du texte présent dans des images ou des séquences vidéo. Nous proposons un schéma de détection s'appuyant sur la mesure du gradient directionnel cumulé. Dans le cas des séquences vidéo, nous introduisons un processus de fiabilisation des détections et l'amélioration des textes détectés par un suivi et une intégration temporelle.

Mots Clef

Vidéo OCR, détection de texte

7 Introduction

L'extraction du texte des flux de vidéo est un domaine de recherche très récent mais cependant très fécond. Les débuts de notre travail sont inspirés par des travaux dans le domaine de traitement de documents. La recherche du texte dans les journaux a résulté dans les premières travaux sur la vidéo. Par contre, le domaine de la vidéo rencontre de nouveaux problèmes dus aux données très différentes (pour une description détaillée du problème, consultez [**?**]). Les algorithmes évolués de traitement de texte étaient remplacés par des algorithmes spécialement conçus pour les données de la vidéo [42][80][91].

La plupart des travaux traitent le problème de la détection et de la localisation du texte (nommé "détection"). En revanche, il existe très peu de recherche sur la reconnaissance. Même si la plupart des problèmes sont identifiés, tout n'est pas encore

résolu. Cela concerne, entre autres, l'amélioration du contenu. Bien que ce sujet abordé par plusieurs auteurs (e.g. par Li et Doermann[42]), les résultats ne sont pas encore tout à fait satisfaisants. On peut aussi citer la segmentation des caractères du fond complexe. La motivation de notre système étant l'indexation de la vidéo, nous avons concentré notre travail sur le texte statique horizontal (sous-titrage, infos, etc.).

Dans le chapitre 8, nous décrivons les détails de notre système de détection, suivi et binarisation. Le chapitre 9 montre les expériences poursuivies et les résultats obtenus, chapitre 10 donne la conclusion et les perspectives de ce travail.

8 Notre système d'extraction

Le but principal d'un système d'extraction de texte est le suivant: accepter des fichiers d'images et de vidéo, détecter le texte, l'extraire et produire un fichier ASCII incluant le texte dans un format utilisable pour l'indexation.

La détection du texte est réalisée dans chaque frame de la vidéo [3]. Les rectangles figurant la localisation du texte sont suivis pendant leur période d'apparition pour associer les rectangles se correspondant dans les différents frames. Cette information est nécessaire pour améliorer le contenu de l'image, ce qui peut être atteint par l'intégration de plusieurs rectangles contenant le même texte. Cette phase doit produire des sous-images d'une qualité en accord avec les pré-requis d'un processus OCR. Par conséquent, il est aussi nécessaire d'augmenter la résolution en utilisant l'information supplémentaire prise dans la séquence d'images.

8.1 Détection

Notre approche s'appuie sur le fait que les caractères du texte forment une texture régulière contenant des contours verticaux allongés horizontalement. L'algorithme de détection reprend la méthode de LeBourgeois [38], qui utilise l'accumulation des gradients horizontaux pour détecter le texte:

$$A(x,y) = \sum_{i=-t}^{t} \frac{\partial I}{\partial x}(x+t, y)$$

Les paramètre de ce filtre sont l'implémentation de l'opération dérivatif (nous avons choisi la version horizontale du filtre Sobel, qui a obtenu les meilleurs résultats) et la taille de la fenêtre de l'accumulation, qui correspond à la taille des caractères. Comme les résultats ne dépendent pas trop de ce paramètre, nous l'avons fixé à $2t + 1 = 13$ pixels. La réponse est une image contenant une mésure de la probabilité de chaque pixel d'être un pixel de texte.

[3] Un sous échantillonnage de la vidéo peut accroître la rapidité du système mais cela réduit la fiabilité du suivi.

La binarisation des gradients cumulés est poursuivie par une version deux-seuils de la méthode d'Otsu [59]. Cette méthode calcule un seuil global à partir de l'histogramme de niveau de gris. Nous avons changé la décision de binarisation pour chaque pixel comme suit:

$$\begin{array}{lll} I_{x,y} < k_{bas} & \Rightarrow & B_{x,y} = 0 \\ I_{x,y} > k_{haut} & \Rightarrow & B_{x,y} = 255 \\ k_{bas} > I_{x,y} > k_{haut} & \Rightarrow & B_{x,y} = \end{array}$$

$$\begin{cases} 255 & \text{s'il existe un chemin} \\ & \text{à un pixel } I_{u,v} > k_{haut} \\ 0 & \text{sinon} \end{cases}$$

où le seuil k_{haut} correspond au seuil calculé par la méthode d'Otsu, et le seuil k_{bas} est calculé à partir du seuil k_{haut} et le premier mode m_0 de l'histogramme: $k_{bas} = m_0 + 0.87 \cdot (k_{haut} - m_0)$.

Avant de passer des pixels à des zones compactes, nous utilisons un post-traitement morphologique afin d'extraire les rectangles englobant des zones de texte. Cette dernière étape permet d'atteindre plusieurs buts:

- Réduire le bruit.
- Corriger des erreurs de classification à partir de l'information du voisinage.
- Connecter des caractères afin de former des mots complets.

La phase morphologique est composé des étapes suivantes:

- Fermeture (1 itération).
- Suppression des ponts (Suppression de tous les pixels qui font partie d'une colonne de hauteur inférieur à 2 pixels).
- Dilatation conditionnelle (16 itérations) suivie par une érosion conditionnelle (16 itérations).
- Érosion horizontale (12 itérations).
- Dilatation horizontale (6 itérations).

Nous avons conçu un algorithme de dilatation conditionnelle pour connecter les caractères d'un mot ou d'une phrase, qui forment différentes composantes connexes dans l'image binarisée. Cela peut arriver si soit la taille de la police soit les distances entre les caractères sont relativement grand. L'algorithme est basé sur une dilatation "standard" avec l'élément structurant $B_H = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ et des conditions suivantes vérifiées pour chaque pixel: Un pixel P est dilaté seulement, si

- la différence de hauteur de la composante C_a incluant le pixel P et la composante C_b voisin à droite ne dépasse pas un seuil t_1.

- la différence de positions y de ces deux composantes ne dépasse pas un seuil t_2.
- la hauteur de la boite englobante incluant ces deux composantes ne dépasse pas un seuil t_3.

Les pixels dilatés sont marqués avec une couleur spéciale. L'érosion conditionnelle suivant la dilatation conditionnelle utilise le même élément structurant B_H. Les pixels supprimés par cette opération sont restreint aux pixels marqués en couleur spéciale par la dilatation. Après les deux opérations tous les pixels encore marqués en couleur spéciale sont marqués comme texte.

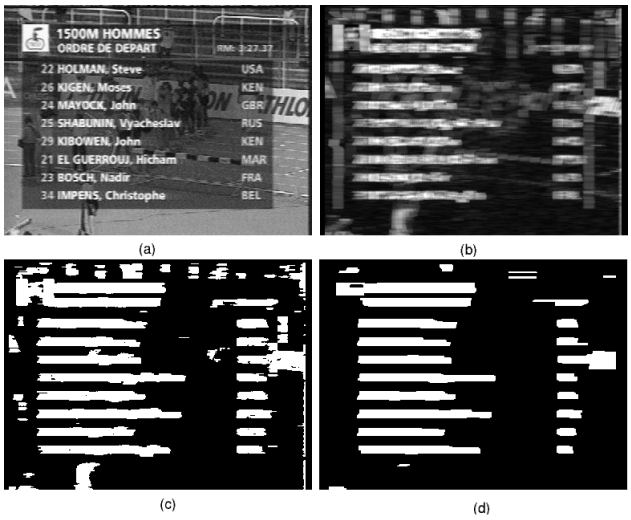


Figure 8.8: Les résultats intermédiaires pendant la détection: l'image d'entrée (a), le gradient (b), l'image binarisée (c), l'image après le post-traitement (d).

La figure 8.8 montre les résultats pendant la détection. Les figures 8.8a et 8.8b affichent l'image originale et l'image des gradients accumulés. Les régions de texte sont visiblement marquées en blanc. La figure 8.8c montre l'image binarisée, encore portant du bruit, qui est supprimé dans la figure 8.8d.

Après le traitement morphologique, chaque région (composante connexe) est vérifiée en imposant des contraintes sur sa géométrie de façon à encore réduire le nombre des fausses alarmes. Enfin, une recherche des cas particuliers est appliquée pour considérer aussi les traits détachés de la zone de chaque composante (les hauts de casse et les bas de casse).

8.2 Suivi

Le but de ce module est le suivi et l'association des rectangles se correspondant afin de produire des apparitions de texte pendant plusieurs frames de la vidéo. Pour ceci, nous utilisons des mesures du recouvrement calculées entre chaque rectangle du frame

et chaque rectangle de la liste des apparitions. Les résultats de la phase de suivi sont des occurrences du texte contenant de l'information sur la localisation du texte dans chaque frame. La longueur de l'apparition, c.à.d le nombre de frames où elle apparaît, nous sert comme mesure de stabilité du texte. Nous considérons les apparitions de longueur plus courte qu'un seuil fixe comme fausses alarmes.

8.3 Amélioration du contenu et binarisation

Pour qu'un OCR soit en mesure d'extraire le texte contenu dans un rectangle, il est nécessaire d'améliorer la qualité de l'image. Nous utilisons les contenus de tous les frames F_i d'une apparition de texte pour produire une seule image améliorée. Cela est fait d'une façon robuste, basée sur des statistiques calculées sur le niveau de gris de chaque pixel pendant le temps d'apparition.

De plus, le processus d'amélioration comprend une phase d'augmentation de résolution. Cela n'ajoute pas d'information, mais adapte l'image au format nécessaire pour un traitement avec un logiciel commercial. Nous avons choisi l'interpolation bi-linéaire, qui calcule le niveau de gris d'un pixel comme moyenne des niveaux de gris de ses voisins. Le poids de chaque voisin correspond à la distance au pixel calculé: $(1-a)\cdot(1-b)$ (voir figure 8.9). Cette augmentation est appliquée à chaque frame de l'apparition, l'image finale étant la moyenne de tous les frames.

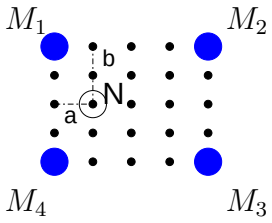


Figure 8.9: L'interpolation bi-linéaire.

L'amélioration du contenu est accomplie par un poids additionnel ajouté au schéma d'interpolation agrandissant chaque frame. Le but de cette amélioration est une meilleur robustesse vis-à-vis du bruit en diminuant le poids pour les pixels voisins aberrants en regard de la moyenne des niveaux de gris. Le facteur g_k^i correspondant au frame F_i et au voisin M_k est calculé comme suit:

$$g_k^i = \frac{1}{1 + \frac{|F_i(M_k) - M(M_k)|}{V(M_k)}}$$

où M est l'image moyenne et V est l'image d'écart type de l'appartion. Le poids final pour le voisin M_k est $(1-a)\cdot(1-b)\cdot g_k^i$.

L'image améliorée doit être binarisée avant le traitement par un OCR afin de supprimer le bruit de fond. Nous avons choisi la version amélioré par Sauvola et al. [64] de

la méthode de Niblack [57]. Cette approche calcule une surface de seuillage en glissant une fenêtre sur l'image. Pour chaque pixel un seuil T est calculé à partir de quelques statistiques sur les niveaux de gris dans la fenêtre:

$$T = m \cdot (1 - k \cdot (1 - \frac{s}{R}))$$

où m est la moyenne, s est l'écart type, k est un paramètre fixé à $k = 0.5$ et R est la dynamique de l'écart type, fixé à $R = 128$. Cet algorithme a prouvé sa capacité à binariser des documents numérisés. Par contre, face aux documents multimédias caractérisés par des propriétés différentes (faible contraste, écart de niveaux de gris etc.), les résultats sont insuffisants. Nous avons changé le calcul du seuil T pour améliorer l'efficacité face aux vidéos:

$$T = m - k\,\alpha\,(m - M) \quad , \quad \alpha = 1 - \frac{s}{R} \quad , \quad R = max(s)$$

où M est le minimum des niveaux de gris de toute l'image et la dynamique d'écart type R est fixée au maximum de l'écart type s de toutes les fenêtres. Plus de détails sur cet algorithme sont proposès dans [**?**].

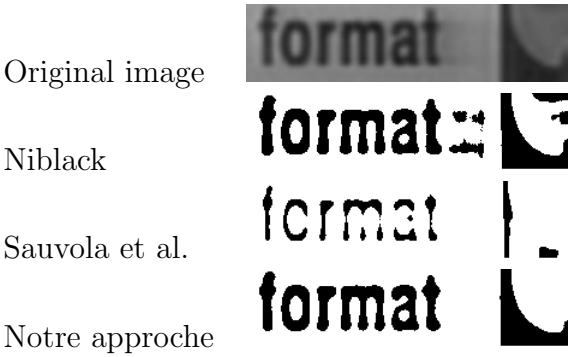


Figure 8.10: Différentes méthodes de binarisation.

La figure 8.3 montre une image exemple et les résultats obtenus par différentes méthodes. La méthode de Niblack segmente bien les caractères de texte, par contre du bruit est crée dans les zones sans texte. Les résultats obtenus avec la méthode de Sauvola et al. montrent moins de bruit dus aux hypothèses sur les données. Par contre, ces hypothèses ne sont pas toujours justifiées dans le cas de données vidéo, résultant dans des trous dans les caractères. Notre méthode résout ce problème en gardant les bonnes performance au niveau de bruit.

9 Résultats

Pour évaluer notre système nous avons utilisé 60.000 frames de différentes vidéos (voir figure 8.11) en format MPEG 1 (384×288). Ces vidéos contiennent 371 occurrences

de texte avec 3519 caractères. Pour la reconnaissance nous avons utilisé le produit commercial Finereader 5.0.

Figure 8.11: Exemples de la base de vidéo utilisée.

Les résultats sont regroupés dans la table 8.3. Nous obtenons un taux de détection de 93.5% de rectangles. 79.6% de caractères détectés était reconnus correctement par l'OCR.

Détection	OCR		
	Méthode	Recall	Precision
Taux 93.5%	Niblack	73.1%	82.6%
	Sauvola	58.4%	88.5%
	Notre méth..	**79.6**%	**91.5%**

Table 8.3: Résultats de détection et OCR.

Les nombreux paramètres permettent une grande flexibilité du système et une bonne adaptation aux caractéristiques intrinsèques des vidéos auxquelles ils sont directement liés.

10 Conclusion

Le travail présenté dans cet article constitue la première phase de notre projet. Les différents éléments de l'état actuel de notre système doivent être optimisés mais la structure générale est maintenant stable. Les taux de détection sur le texte horizontal sont prometteurs.

La continuité de ce travail est consacrée au texte ayant des propriétés moins spéciales (texte animé, texte avec des orientations générales etc.) et sur la phase de reconnaissance.

References

RFIA 2002

Extraction de texte dans des vidéos: le cas de la binarisation

Résumé

Dans cet article nous abordons la problème de la binarisation de "boites", *i.e.* sous-image, contenant du texte. Nous montrons que la spécificité des contenus vidéos amène à la conception d'une nouvelle approche de cette étape de binarisation en regard des techniques habituelles tant du traitement d'image au sens large, que du domaine de l'analyse de documents écrits.

Mots Clef

Binarisation, seuillage, vidéo, textes artificiels.

11 Introduction

Depuis quelques années, les documents audiovisuels numérisés sont de plus en plus fréquents. Des grandes bases de données audiovisuelles ont été crées par des entreprises, des organisations et aussi par des personnes privées. Cependant, l'utilisation de ces bases reste encore problématique. Tout particulièrement, ces nouveaux types de données, image et vidéo, ont conduit à de nouveaux systèmes d'indexation où la recherche par le contenu se fait à partir d'une image exemple (Qbic, VisualSEEK, SurfImage, MARS ...), grâce à des mots clefs ou les deux (WebSEEK, Two.Six, Virage).

Notre objectif n'est pas ici de traiter de la problématique de l'indexation en général. On peut cependant constater que les résultats ne sont pas encore à la hauteur des attentes des usagers potentiels. La principale raison se situe dans l'extrême difficulté rencontrée lors du passage des informations liées au signal à celles qui relèveraient de la sémantique portée par celui-ci. Cette difficulté n'est d'ailleurs pas nouvelle [**?**].

Cependant, entre le niveau du pixel, et celui de la sémantique, il existe des caractéristiques à la fois riche en information et cependant simples. Le texte présent dans les images et les vidéos fait partie de cette catégorie. Une fois extrait, ce texte peut

alimenter les index qui caractérisent un plan ou une séquence au même titre que des indicateurs plus proches du signal.

Bien sûr, afin de soulager le travail du documentaliste, il est nécessaire de faire en sorte que la détection (et la reconnaissance) du texte soit la plus automatique possible.

Nous avons abordé le problème général de la détection de textes dans les vidéos dans notre projet ECAV (Enrichissement de Contenu Audiovisuel) en collaboration avec France Télécom. Celui-ci a donné lieu à un brevet [**?**]. Une présentation générale peut être trouvée dans [**?**].

Dans cet article, nous n'abordons qu'un des aspects de cette étude. En effet, au delà de l'aspect ingénierie inhérent au développement d'un système efficace et robuste, nous avons, au cours de notre étude, mis en évidence des manques en termes de modélisation formelle et donc d'outils qui en découlent.

Tout particulièrement, nous nous focaliserons ici sur le cas de la segmentation, *i.e.* binarisation d'une sous-image dans lequel un processus approprié de détection a mis en évidence la présence de textes artificiels avec une forte probabilité. L'étape qui nous intéresse ici se situe donc en aval de cette détection et en amont de la phase de reconnaissance du texte par un logiciel d'OCR qui nécessite une donnée de type binaire.

Nous allons tout d'abord présenter les spécificités des données de type vidéo. Nous verrons ensuite un rapide survol des principales approches de la binarisation et tout particulièrement de celles qui sont utilisées dans le traitement des documents. Nous pourrons alors voir en détails notre approche que nous comparerons sur des exemples variés.

12 Sur la nature des données

12.1 Vidéo vs document numérique

Les recherches en détection et extraction du texte à partir des séquences vidéo sont encore confrontées à de sérieux problèmes. Pourtant la recherche dans le domaine de l'OCR des documents classiques (c.a.d. les documents écrits, les journaux etc.) a développé des méthodes performantes et des outils commerciaux produisent de bons résultats pour des images de documents. Le problème principal peut être expliqué par la différence entre l'information présente dans un document et celle donnée par une séquence vidéo ainsi que les méthodes de stockage de chaque type de données.

Les images de documents sont créées en vu de les numériser pour les passer ensuite à une phase OCR pour reconnaître la structure et le texte. Pour améliorer la qualité et le taux de la reconnaissance, les images sont numérisées à très haute résolution (200-400 dpi) donnant des fichiers de taille très élevée (un fichier de 100 Mo résulte d'une page A4 numérisée à 400 dpi). Les fichiers sont comprimés sans perte pour garder la qualité et empêcher des artéfacts de compression. Ces grandes tailles ne sont pas une limite puisque ni leur transport ni leur stockage ne sont prévus. La plupart du temps les images de documents sont bien structurées et contiennent un fond uniforme et la couleur du

texte est également uniforme. Ceci permet de séparer les caractères du fond avec un seuillage fixe ou adaptatif. La majorité de la page contient des caractères structurés dans différents paragraphes, quelques images sont inclues dans la page.

Par contre, les images des séquences vidéo contiennent de l'information plus difficile à traiter. Le texte n'est pas séparé du fond. Il est soit superposé (le "texte artificiel" comme les sous-titres, les résultats de sport etc.) soit inclut dans la scène de l'image (le "texte de scène", par exemple le texte sur le tee-shirt d'un acteur). Le fond de l'image peut être très complexe ce qui empêche une séparation facile des caractères. De plus, contrairement aux documents écrits, les séquences vidéo contiennent de l'information très riche en couleurs. Enfin, le texte n'est pas structuré en lignes, et souvent quelques mots courts et déconnectés flottent dans l'image.

12.2 La faible résolution

Le but principal de la conception des formats de fichiers vidéo est de garder une qualité suffisante pour l'affichage, pour le stockage et le transport par des réseaux informatiques. Pour cette raison et pour une quantité de données vidéo plus haute, il est nécessaire de réduire fortement l'information avant son stockage dans le fichier vidéo. Pour limiter la taille de l'information deux méthodes sont souvent appliquées : la forte réduction de la résolution et le codage avec perte, c.a.d. avec élimination des données redondantes et perte d'une partie de l'information originale, en utilisant les algorithmes de compression JPEG et MPEG. La résolution spatiale de la vidéo dépend de l'application et prend des valeurs entre 160×100 pixels (diffusion par Internet) et 720×480 pixels (DVD vidéo codé en MPEG 2). Un format typique est le format CIF avec 384×288 pixels. Notons que la taille du texte affiché avec cette résolution est beaucoup moins grande que le texte résultant d'un document numérisé. Les frames d'une vidéo de cette résolution contiennent des caractères d'une hauteur de moins de 10 pixels, alors que dans les documents numérisés, les caractères ont une hauteur comprise entre 50 et 70 pixels.

12.3 Les effets d'aliasing

Pendant la production de la vidéo, le signal original est sous échantillonné plusieurs fois. Pour empêcher des effets d'aliasing, des filtres passe bas sont appliqués. Le signal résultant a alors une qualité suffisante pour la lecture mais il est complètement lissé.

Après la réduction de la résolution, la compression du signal ajoute également des artéfacts. L'information supprimée par le schéma MPEG est considérée comme redondante pour le système de visuel humain. Cependant, les artéfacts de l'encodage causent des problèmes pendant la reconnaissance (par exemple en perturbant l'uniformité des couleurs). Même si les nouvelles normes comme JPEG 2000 apportent un plus en regard de ce type de problèmes, l'existence de très nombreuses données codées selon le format JPEG classique justifie nos recherches dans cette direction.

12.4 Le fond complexe et détaillé

Le fond sur lequel est inscrit le texte ne peut être considéré comme constant. Un seuillage global, même déterminé localement, n'est le plus souvent pas suffisant pour clairement mettre en évidence le texte.

Enfin, afin de faciliter la lecture du texte, des artifices d'inscription du texte sont utilisés (e.g. des ombres artificielles qui augmentent le contrast avec le fond). Ces techniques ne sont malheureusement pas un avantage pour la détection. L'environnement du texte peut être considéré comme bruité mais pas avec un modèle simple de type additif.

12.5 Quelques hypothèses

Dans ce qui suit, nous ferons les hypothèses suivantes (qui découlent des spécificités du texte dans les vidéos). Tout d'abord, nous assumerons une gamme fixe de taille de police. Plus exactement, nous supposerons que dans une boite de texte potentiel, il n'existe qu'une seule police de caractères. Ensuite, nous supposerons que le texte peut être discriminé du fond sur la composante luminance (ce qui est en fait l'hypothèse principale de toute méthode de seuillage). Sans que cela soit une contrainte trop forte, nous supposerons par la suite que le texte est plus foncé que le fond. Enfin, compte tenu de la variété du fond, nous imposons une recherche adaptative de seuil, *i.e.* par fenêtre locale. La taille de la fenêtre devra être fonction de la taille des caractères afin que toute fenêtre ne puisse être totalement incluse dans un caractère mais contienne toujours une part significative de pixels du fond.

13 Sur la binarisation

Proposer une nouvelle approche de la binarisation est, au premier abord, un peu surprenant tant ce domaine a été étudié par le passé et ceci dès le début des travaux sur les images numériques. En se référant à notre travail, on pourra citer deux types d'approches.

13.1 Les méthodes générales

Il s'agit là des méthodes à usage général. On peut y inclure les deux approches les plus célèbres. Tout d'abord, la méthode de Fisher/Otsu (issue du domaine de l'analyse de données et optimisée par Otsu [59]), encore dénommée méthode de minimisation de la variance (implicitement on fait ici référence à la variance intraclasse), s'appuie sur des caractéristiques statistiques des deux classes de pixels définies par le choix d'un seuil. Il n'y a donc pas vraiment de modèle sous-jacent sur la forme de l'histogramme. Par contre, la taille des populations doit permettre une estimation des paramètres statistiques, *i.e.* il y a explicitement référence à deux classes.

Nos tests ont montré que cette approche, bien que très générale, présentait de nombreux avantages pour les images qui nous intéressent, c.à.d des images bruitées de résolution basse (voir section 12). Cependant, les résultats sont en deçà de ceux obtenus avec les méthodes que nous présenterons par la suite. Nous ne l'avons donc pas inclue dans les comparaisons présentées dans cet article.

La méthode de minimisation de l'erreur de seuillage s'appuie quant à elle sur une modélisation de l'histogramme en terme de somme de distributions. La position optimale du seuil est alors définie par l'intersection de ces distributions. Une solution simple est connue dans le cas où les distributions sont normales [35]. Dans notre approche, il n'est pas possible d'obtenir des histogrammes pour lesquels une approximation de courbes donne de bons résultats. En effet, la taille des fenêtres est typiquement de l'ordre de 30×30 pixels. La forme de l'histogramme induit est très fortement bruitée et ne donne pas une bonne approximation des éventuelles distributions sous-jacentes.

Il est nécessaire de s'appuyer sur des critères plus globaux.

La principale critique que l'on peut faire à ces techniques est leur généralité, c'est à dire leur faible adaptabilité à un contexte particulier. C'est pourquoi il semble plus judicieux de s'orienter vers des techniques relevant d'un domaine similaire à la vidéo.

13.2 Les méthodes du domaine des documents numériques

Comme nous l'avons évoqué, l'extraction d'une zone de texte en vue de sa reconnaissance/transcription par un logiciel d'OCR est une étape clef des logiciels d'analyse de documents. C'est pourquoi l'on y retrouve des techniques adaptées aux spécificités de ce type d'images (tout particulièrement la grande densité). On pourra se référer à [74] pour une étude comparative. La plus performante de ces méthodes est sans doute celle proposée par Niblack [57]. Il s'agit d'une approche adaptative où un seuil T est déterminé en tout point par la formule

$$T = m + k.s \tag{8.1}$$

où m, s et k désignent respectivement la moyenne des niveaux de gris sur la fenêtre centrée sur le point, l'écart type de cette distribution et une constante que Niblack a fixée empiriquement à -0.18. Le résultat est peu dépendant de la taille de la fenêtre même si l'on constate que les meilleurs résultats sont obtenus lorsque celle-ci englobe 2 ou 3 caractères. Bien que très simple, cette méthode ne permet pas de traiter le cas de la présence de textures dans le fond qui sont le plus souvent détectées comme du texte à faible contraste.

On doit à Sauvola *et al.* [64] une amélioration qui permet de tenir compte de cette remarque.

$$T = m.(1 + k.(\frac{s}{R} - 1)) \tag{8.2}$$

où R traduit la dynamique du paramètre s [4] et est fixée empiriquement à 128. Si le seuil se situe de nouveau proche de la moyenne, celui-ci s'adapte mieux au contenu de l'image sans pour autant répondre à toutes les contraintes que nous avons évoquées.

La figure 8.12 montre un exemple d'application de ces méthodes sur une boite de texte. Comme nous l'évoquions précédemment, la méthode de Niblack extrait non seulement des structures importantes (à droite) ce qui est logique compte tenu de leur luminance très sombre, mais aussi des textures du fond (juste après la dernière lettre du mot, moins visible à l'oeil dans l'image originale) du fait de la localité du seuillage. Sur le même exemple, la méthode de Sauvola est moins sensible à ces bruits mais son seuillage, plus strict, conduit à des caractères extraits plus fins et présentant par endroit des coupures très préjudiciables à toute reconnaissance sans un post traitement de type morphologique.

(a) Image originale

(b) Niblack

(c) Sauvola

Figure 8.12: Exemple de binarisation par les méthodes de Niblack et Sauvola sur une image contenant du texte.

14 Notre proposition

Nous avons focalisé notre étude sur les approches statistiques, c'est à dire celles qui utilisent une caractérisation statistique de l'image à seuiller. Compte tenu de ce que nous avons spécifié dans nos hypothèses, nous supposerons qu'une fenêtre centrée en un point ne peut être que de deux types, soit elle contient uniquement des pixels du fond, soit elle contient à la fois des pixels du fond et des pixels du texte.

La formule de Sauvola peut être réécrite sous la forme

$$T = m + k\alpha m \tag{8.3}$$

[4] De plus Sauvola propose la valeur 0.5 pour la constante k.

où $\alpha = \frac{s}{R} - 1$.

Le terme correctif par rapport à la moyenne dépend donc de la dynamique locale mais aussi de la valeur de la moyenne. Plus celle-ci sera haute, plus le seuil sera diminué [5]. Les textures du fond qui seraient claires induiront donc un seuil bas qui induira une fenêtre binarisée appartenant totalement à la classe fond. Sous l'hypothèse que le texte est plus foncé que le fond, cela semble résoudre le problème des textures de fond. La motivation de cette amélioration trouve sa justification dans le domaine des documents écrits lorsque le contraste local est faible. Cette hypothèse est cependant trop stricte pour les vidéos où il est nécessaire de prendre en compte une plus grande variété qui ne peut se réduire dans une valeur constante de la dynamique de référence, *i.e.* $R = 128$.

La figure 8.13 montre un effet de surluminance de l'ensemble de l'image.

(a) Image originale

(b) Niblack

(c) Sauvola

Figure 8.13: Exemple de binarisation par la méthode de Sauvola lorsque l'image est globalement claire. L'ensemble de l'image est classée comme fond. La méthode originelle de Niblack fournit quant à elle un résultat correct.

Afin de coller au mieux aux données, nous proposons de normaliser les éléments intervenant dans l'équation qui définit le seuil T. Cela revient à ne plus raisonner sur les valeurs absolues mais sur les contrastes, ce qui est naturel si l'on regarde les différentes justifications des méthodes de Niblack [6] et de Sauvola qui argumentent sur la notion de contraste.

Comment peut-on définir ces contrastes ? Si l'on se réfère à [57], p. 45, le contraste, au centre de la fenêtre de niveau de gris I, est défini par

$$C_L = \frac{|m - I|}{s} \tag{8.4}$$

[5] Sauvola propose R fixé à 128. On a donc $s < R$ qui conduit à des valeurs négatives pour α.

[6] Niblack situe l'origine de son travail dans les méthodes de transformation d'histogramme de niveaux de gris pour augmenter le contraste d'une image.

Comme nous avons supposé un texte sombre sur un fond clair, nous ne considérerons pas les points ayant un niveau de gris plus grand que la moyenne locale, la valeur absolue pourra donc être éliminée.

Si l'on note M la valeur minimale des niveaux de gris de l'image, la valeur maximale de ce contraste local, notée C_{max} est donnée par

$$C_{max} = \frac{m - M}{s} \tag{8.5}$$

Il est difficile d'appuyer une stratégie de seuillage sur la seule comparaison entre le contraste local et sa valeur maximale car cela ne permet pas de prendre en compte la variabilité de la fenêtre centrée sur le point en regard du restant de l'image. C'est pourquoi nous proposons de définir un contraste plus global, le contraste de la fenêtre, noté C_F, par

$$C_F = \frac{m - M}{R} \tag{8.6}$$

où R désigne la valeur maximale des écarts type pour toutes les fenêtres de l'image.

Ce contraste permet de savoir si la fenêtre est plutôt sombre ou claire par rapport à l'ensemble de l'image (une valeur forte caractérisera l'absence de texte). Nous pouvons maintenant exprimer notre stratégie en fonction de ces contrastes.

Un critère de seuillage simple consiste à ne conserver que les points ayant un contraste local proportionnellement fort en regard de la valeur maximale corrigée par le contraste de la fenêtre centrée sur ce point.

$$I : C_L > a(C_{max} - C_F) \tag{8.7}$$

où a est un paramètre de gain.

En développant cette équation, on obtient la valeur du seuil

$$T = (1 - a)m + aM + a\frac{s}{R}(m - M) \tag{8.8}$$

Dans le cas où l'on se trouve sur la fenêtre de variabilité maximale, *i.e.* $s = R$, on obtient $T = m$. On incite le processus à conserver le maximum de points sur la fenêtre. Par contre, dans le cas où la variabilité est faible (*i.e.* $s \ll R$), il y a une probabilité forte d'absence de texte. On ne conservera un pixel que si il a un fort contraste local. Le seuil est obtenu par $T \approx (1 - a)m + aM$. Le paramètre a permet de régler la marge d'incertitude autour de la moyenne. Une solution simple consiste à fixer la valeur de ce paramètre à 0.5, le seuil se situant alors à mi-distance entre M et m.

La figure 8.14 montre un exemple d'application de notre méthode sur les deux images utilisées aux figures 8.12 et 8.13. Les résultats sont corrects comparativement aux autres méthodes.

La figure 8.15 montre un autre exemple pour les trois méthodes que nous avons détaillées pour une image qui contient un fond majoritaire (donc influent sur les paramètres statistiques) ainsi qu'une dérive lumineuse. Notre approche combine la robustesse de la

(a) Image de la figure 1

(b) Image de la figure 2

Figure 8.14: Exemple de binarisation par notre méthode pour les deux images présentées aux figures 1 et 2.

méthode de Sauvola vis à vis du fond et la qualité de délimitation des caractères de la méthode de Niblack.

(a) Image originale

(b) Niblack

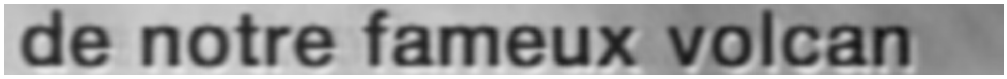
(c) Sauvola

(d) Notre approche

Figure 8.15: Exemple de binarisation par les différentes méthodes présentées dans cet article.

La figure 8.16 présente un exemple plus complexe de part la mise en forme des caractères (effet de relief) et la présence d'un fond très sombre. Le résultat confirme le comportement des trois méthodes.

Enfin, la figure 8.17 présente un dernier exemple où le fond a une texture complexe. Nous avons, pour cet exemple, ajouté la méthode de Fisher globale (seuil unique pour toute l'image) et locale (un seuil par fenêtre). La méthode de Fisher locale est très similaire à celle de Niblack de part son comportement sur les textures du fond. Notre approche fournit un résultat comparable à la méthode de Sauvola.



(a) Image originale

(b) Niblack

(c) Sauvola

(d) Notre approche

Figure 8.16: Exemple de binarisation par les différentes méthodes présentées dans cet article.

15 Évaluation

Il est souvent plus simple de proposer une nouvelle technique que d'en prouver l'utilité en regard des techniques existantes. Comment pouvons-nous, en dehors de quelques exemples bien choisis, prouver l'intérêt de ce nouveau schéma de binarisation. Une approche possible est de créer artificiellement des images de référence perturbées en utilisant des modéles des dégradations déjà connus. Cependant, nous pensons que cette approche ne démontrera pas grand chose sur la réelle utilité de notre technique dans le cas de la vidéo. En effet, celle-ci n'a pas été conçue pour devenir une technique générique pour toutes les images mais pour répondre à un objectif bien précis, l'extraction de texte dans des vidéos **en vue de leur reconnaissance**. Nous jugerons donc de l'intérêt de la technique par la performance induite sur la phase de reconnaissance. En effet, meilleure sera la binarisation, plus facile sera la reconnaissance.

Le protocole que nous avons choisit est donc le suivant. Les boites de texte une fois binarisées [7] sont passées à un OCR (Finereader). Chaque chaîne de caractères est ensuite évaluée par la méthode de Wagner et Fisher [79]. Soient α et β deux caractères appartenant respectivement à une chaîne reconnue et à une chaîne de référence, *i.e.* la vérité terrain. La fonction de coût, γ, est définie par

[7] Dans tous nos essais, la taille de la fenêtre a été fixée à 30×30 pixels.

$$\gamma(\alpha,\beta) = \begin{cases} 0 & \text{si } \alpha = \beta \ , \ \alpha,\beta \in X \bigcup \{\lambda\} \\ 0.5 & \text{si } \alpha = \beta \ , \ \alpha,\beta \in X \bigcup \{\lambda\} \\ & \text{et } \alpha \text{ et } \beta \text{ sont dans des formats} \\ & \text{(min./maj.) différents} \\ 1 & \textit{sinon} \end{cases} \tag{8.9}$$

où X est l'ensemble des caractères et λ le caractère "vide" pour lequel une fonction de coût spécifique est requise..

$$\gamma(\lambda,\beta) = \begin{cases} 0.5 & \text{si } \beta \text{ est un espace} \\ 1 & \textit{sinon} \end{cases} \tag{8.10}$$

et

$$\gamma(\lambda,\beta) = \gamma(\beta,\lambda) \tag{8.11}$$

En plus de cette fonction de coût, il est aussi possible de caractériser la reconnaissance du texte sur la base de test par les mesures de précision et de rappel.

$$\text{Précision} = \frac{\text{Nombre de caractères reconnus}}{\text{Nombre de caractères proposés par l'OCR}}$$

$$\text{Rappel} = \frac{\text{Nombre de caractères reconnus}}{\text{Nombre de caractères dans la référence}}$$

Les résultats qui sont résumés dans le tableau 8.4 ont été obtenus pour 4 vidéos extraites de la base mise à disposition par l'INA. Cela représente 60000 frames (environ 40 minutes) contenant 322 occurrences de textes. La figure 8.18 propose des extraits de ces vidéos qui montre bien la grande variété des apparitions de textes.

Dans cette étude, deux versions de la méthode de Sauvola ont été testées. La première correspond à celle que nous avons présentée (cf. Eq. 8.2). Dans la deuxième, nous avons simplement remplacé la valeur fixe de R (=128) par une valeur adaptative identique à celle que nous avons introduite dans notre approche, *i.e.* $R = max(s)$. Sur les trois indicateurs, rappel, précision et coût, cette simple modification induit un gain significatif (surtout sur le rappel) et lui permet de dépasser la méthode de Niblack. A l'exclusion d'un indicateur, la précision, et sur une seule vidéo, notre approche fournit les meilleurs résultats ce qui montre sa pertinence et sa robustesse.

16 Discussion

L'approche que nous avons proposée dans cet article permet, grâce à la prise en compte de paramètres statistiques globaux, une meilleure adaptativité à la variété des contenus des vidéos. Le prix à payer se situe au niveau computationnel puisque ces paramètres supplémentaires requièrent un traitement en deux passes. Cependant, ce coût est très faible au regard, d'une part du gain sur les performances, et d'autre part du coût global

incluant les phases de détection des boites de texte et de reconnaissance par le logiciel d'OCR.

Cette nouvelle méthode n'est *a priori* pas définitive en ce sens que d'autres paramètres epuvent être introduit. A titre d'exemple, nous avons évoqué (cf. Eq. 8.8) le cas où la variabilité locale est faible ($s \ll R$). Il serait intéressant de pas lier cette propriété à la seule comparaison de s à la valeur maximale mais également à la valeur minimale de l'écart type. Cette autre valeur extrême permet de quantifier le bruit additif à l'image et par là même d'accéder au rapport signal/bruit de l'image. De nombreuses techniques existent pour l'estimation de cette variance minimale (par exemple [**?**]). Elles sont souvent coûteuses en temps et leur adaptativité aux spécificités de la vidéo doit être préalablement étudiée.

Au delà du seul objectif de reconnaissance, nous souhaitons aussi étudier l'apport de ces indicateurs pour filtrer plus efficacement les fausses alarmes, *i.e.* les boites de texte qui ne contiennent pas de texte, qui sont encore trop nombreuses dans notre système (en conséquence de notre soucis de ne pas louper de texte) et qui induisent un surcoût de traitement dans la phase de reconnaissance du texte.

Notre approche s'appuie sur une formalisation en termes de contraste et peut donc être reliée à tous les travaux, très nombreux en traitement d'images. En particulier, il sera nécessaire de faire le lien avec la méthode de Kholer [32].

Enfin, nous avons, pour le moment, focalisé notre étude sur le texte dit artificiel, c'est à dire incrusté dans une scène par surimpression. Afin de fournir des informations plus complètes pour construire des index plus informatifs, il est nécessaire d'étendre cette recherche aux textes dits de scène, c'est à dire totalement inclus dans le contenu de la scène. Ceci constituera le cadre de la poursuite de notre collaboration avec France Télécom Recherche & Développement.

References

Vidéo	Méthode	Rappel	Précision	Coût
AIM2	Niblack	67.4%	87.5%	499
	Sauvola R=128	53.8%	87.6%	616.5
	Sauvola R ad.	75.3%	87.8%	384.5
	Notre approche	**78.4%**	**90.4%**	**344.5**
AIM3	Niblack	92.5%	78.1%	196
	Sauvola R=128	69.9%	89.6%	206
	Sauvola R ad.	85.3%	92.5%	110
	Notre approche	**96.2%**	**95.3%**	**51**
AIM4	Niblack	78.5%	**92.0%**	252
	Sauvola R=128	48.6%	87.7%	490.5
	Sauvola R ad.	69.8%	84.6%	360.5
	Notre approche	**80.1%**	90.4%	**211.5**
AIM5	Niblack	62.1%	71.4%	501.5
	Sauvola R=128	66.7%	89.3%	324.5
	Sauvola R ad.	64.9%	90.1%	328
	Notre approche	**69.0%**	**91.0%**	**294.5**
Total	Niblack	73.1%	82.6%	1448.5
	Sauvola R=128	58.4%	88.5%	1637.5
	Sauvola R ad.	73.0%	88.4%	1183
	Notre approche	**79.6**%	**91.5%**	**901.5**

Table 8.4: Comparaison des méthodes de seuillage par la qualité de la reconnaissance des caractères sur les images binaires produites.

(a) Image originale

(b) Fisher

(c) Fisher local

(d) Niblack

(e) Sauvola

(f) Notre approche

Figure 8.17: Exemple de binarisation par les différentes méthodes présentées dans cet article.

AIM2 (INA) : publicités (France 3, TF1)

AIM3 (INA) : journal télévisé (M6, Canal+)

AIM4 (INA) : dessin animé et journal télévisé (Arte)

AIM5 (INA) : journal télévisé (France 3)

Figure 8.18: Extraits de la base vidéos de test.

CORESA 2003

Détection de textes de scènes dans des images issues d'un flux vidéo[8]

Résumé

La plupart des travaux sur la détection de texte se concentre sur le texte artificiel et horizontal. Nous proposons une méthode de détection en orientation générale qui repose sur un filtre directionnel appliqué dans plusieurs orientations. Un algorithme de relaxation hiérarchique est employé pour consolider les résultats locaux de direction. Une étape de vote entre des directions permet d'obtenir une image binaire localisant les zones de textes.

Mots clefs

Détection de texte, indexation, multi orientation

17 Introduction

Lors de précédentes études, nous avons mis au point une technique permettant de détecter et reconnaître des textes artificiels dans des images et dans des séquences audiovisuelles [**?**, **?**]. Dans la suite de cette démarche, nous présentons nos résultats sur la détection de textes de scène dans des images numériques.

Le passage de la notion de texte artificiel à celle de texte de scène s'accompagne d'une relaxation dans les contraintes qui définissent le texte artificiel:
Luminance : Un texte artificiel horizontal est assimilé à une accumulation de gradients forts dans cette direction. Cette hypothèse peut être maintenue mais il faut l'étendre à une direction inconnue.
Géométrique : Un texte artificiel obéit le plus souvent à une charte graphique et tout particulièrement en ce qui concerne la police donc la taille. Pour un texte de scène, cette contrainte n'est plus possible. Il est donc nécessaire de développer une approche multirésolution.

[8]Cette étude est financée par France Télécom R&D dans le cadre du contrat ECAV2 001B575

Forme : Un texte artificiel est un texte fait pour être lu, superposé au signal. Il ne subit pas de distorsion et sa forme est donc régulière. Ce n'est plus le cas pour un texte de scène qui peut subir les transformations/distorsions du support sur lequel il est imprimé (et surtout si ce support, comme dans le cas d'une banderole, n'est pas rigide). Compte tenu de la diversité des distorsions, nous supposerons encore une non déformation du texte.
Temporel : La principale caractéristique d'un texte artificiel est sa stabilité temporelle, indépendamment du contenu du signal. Dans le cas d'un texte de scène, cette contrainte doit être abandonnée.

La seule vraie contrainte d'un texte qui subsiste est la propriété texturelle (succession de gradients orientés). C'est pourquoi notre méthode s'appuie sur un détecteur de gradients cumulés similaire à celui présenté dans [**?**] mais déployé dans plusieurs directions.

La bibliographie relative à la détection de textes dans des directions quelconques est très réduites. Les seuls travaux connus utilisent des détecteurs de textes non directionnels. Li and Doermann [**?**] utilisent un réseau de neurones entraînés sur les coefficients de la transformée de Haar et estiment la direction du texte grâce aux moments d'inertie. Crandall and Kasturi [15] utilisent les coefficients de la DCT du flux MPEG pour détecter les textes et estiment son orientation par maximisation de son recouvrement par un rectangle. Ces deux approches utilisent donc des informations directionnelles de bas niveau et combinent ces informations pour créer un détecteur non directionnel de texte. Cependant, ces deux modèles restent très pauvres.

Nous pensons au contraire que le texte présent dans une vidéo a des caractéristiques directionnelles propres. Un détecteur directionnel est par conséquence nécessaire pour accéder à cette information particulière. L'objectif par cette recherche est, par l'utilisation de caractéristiques plus pertinentes, d'aboutir à une réduction du taux de fausses alarmes qui est un des principaux problèmes de la détection de textes (artificiels ou de scènes).

L'article est organisé de la façon suivante: La section 18 introduira le schéma général de la méthode. La section 19 présentera en détail la relaxation hiérarchique. Les résultats de nos expériences sont donnés dans la section 20, suivie par une conclusion.

18 Le schéma général

Pour rendre notre détecteur moins sensible à la taille du texte à détecter, nous employons une pyramide à plusieurs échelles (normalement 2 niveaux suffisants). La figure 8.19 montre le schéma du traitement appliqué à chaque échelle.

Le texte est détecté dans plusieurs orientations, *i.e.* 4 (0°,45°,90°,135°). Chaque détecteur fournit une *probabilité* pour un pixel d'appartenir à une zone de texte dans une direction donnée. Cependant, un texte est une texture qui ne répond pas à une direction unique lorsque l'on examine les réponses au niveau du pixel. La notion de direction d'un texte ou d'un mot est une caractéristique plus globale. C'est pourquoi nous utilisons un schéma de relaxation pour consolider et/ou modifier les informations

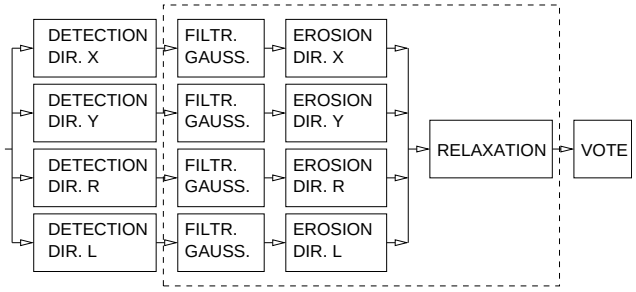


Figure 8.19: Détection de textes de scènes de direction quelconque

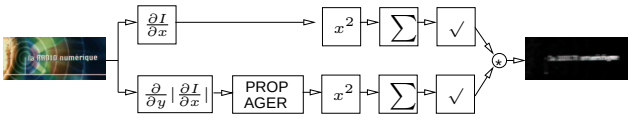


Figure 8.20: Un filtre pour la détection directionelle du texte

locales de direction. Une étape supplémentaire de vote entre des directions permet d'obtenir une image binaire localisant les zones de textes. Enfin, comme nous l'avons évoqué et pour tenir compte de la variabilité des tailles, l'ensemble de ce schéma est reproduit à plusieurs échelles grâce à une structure pyramidale.

Le filtre utilisé pour la détection directionnelle du texte est une amélioration du filtre introduit dans [**?**]. Ce détecteur utilisait une accumulation de gradients horizontaux, justifiée par le fait que les textes forment une texture régulière contenant des frontières verticales qui est alignée horizontalement dans le cas des textes artificiels. Cependant, ce type de détecteur réagit aussi en présence textures qui ne sont pas des textes. Nous avons donc ajouté une contrainte supplémentaire, l'alignement, majoritaire, des ruptures verticales. La nouvelle version de notre filtre tient donc compte de cette contrainte comme le montre la figure 8.20. Ce nouveau filtre est une combinaison entre un filtre à accumulation de gradient et un filtre répondant au coin. Comme détecteur de coin nous utilisons une forme particulière de la dérivée seconde (estimée par la sortie du filtre sobel dans la direction y appliquée sur la valeur absolue du filtre de sobel dans direction x) [9]. Ces valeurs sont propagées dans la direction des traits des caractères (direction y). L'orientation peut alors être estimée par le signe de la sortie du filtre de sobel dans la direction y. Une accumulation dans la direction x met en évidence les régions avec une concentration de coins alignés. La réponse finale du détecteur est le produit des deux parties du détecteur.

[9] Les détecteurs de coins connus (e.g. [**?**]) sont plus sophistiqués et produisent des réponses de coins plus ciblées et plus invariantes. Bien qu'il est facile de les régler pour produire un nombre specifique de points, nécessaire pour l'indexation ou la robotique, il est très difficile de seuiller leur réponse absolue.

19 La relaxation hiérarchique

Une estimation de direction est obtenue classiquement par un filtre local. Une information plus stable est obtenue avec la prise en compte de voisinages de tailles significatives dont le coût est souvent prohibitif. Pour palier cet inconvénient, nous utilisons une structure pyramidale (différente de celle utilisée pour la prise en compte de textes de tailles variables). Dans notre approche, nous nous limitons à la détection de 4 directions principales. Cette détection est bien sûr accompagnée de bruit qui peut être atténué par la prise en compte de la cohérence spatiale des réponses par un processus de relaxation. Dans ce processus, chaque sortie de chaque filtre directionnel est réévalué par la prise en compte des directions détectées dans le voisinage du pixel considéré. Notre implantation dans une structure pyramidale permet de limiter l'accès au voisinage à un seul accès aux informations du parent du pixel dans la structure.

L'algorithme de relaxation peut être résumé par les étapes suivantes :

1. Initialiser les bases des pyramides directionnelles (une par direction testée) par les sorties des filtres directionnels.
2. Construire les 4 pyramides directionnelles des niveaux 1 à $N-1$.
3. Recalculer, pour chaque pixel, la sortie de chaque filtre directionnel par la prise en compte de son voisinage par une descente de la pyramide (du niveau $N-1$ jusqu'à la base).
4. Retourner à l'étape 2.

L'algorithme est itéré classiquement jusqu'à convergence, stagnation des valeurs ou épuisement d'un nombre limité d'itérations fixé *a priori*. En pratique, 10 itérations sont suffisantes pour permettre à la cohérence locale de s'affirmer.

Les 4 pyramides que nous utilisons sont construites de manière à tenir compte de leur objectif, *i.e.* un texte dans une direction particulière, dans leur processus de filtrage/sous-échantillonnage. Par exemple, un texte horizontal est un rectangle ayant son axe principal dans la direction horizontale. Le filtre gaussien utilisé dans la construction de la pyramide horizontale a un support rectangulaire avec une variance dans la direction X plus grande que dans la direction Y. De même pour le filtre appliqué dans la pyramide verticale. Ces deux filtres sont séparables. Ce n'est plus le cas des directions diagonales où nous devons employer un filtre gaussien non séparable. Ces filtres anisotropics seront comparés avec des filtres gaussien isotropics de taille 3×3. Dans tous les cas, nous avons adopté une réduction 2×2 pour le passage entre deux niveaux consécutifs.

Lors de la descente du niveau $N-1$ à la base, la valeur G_i^l du niveau l pour chaque direction i ($i \in \Theta = \{X, Y, R, L\}$) est mise à jour à partir des informations du niveau précédent (nous omettons les coordonnées (x, y) pour des raisons de lisibilité) par

$$G_i^{l'} = \frac{G_i^l U_i}{Z} \quad , \quad Z = \sum_{j \in \Theta} G_j^l U_j$$

où Z est une constante de normalisation et U_i le facteur de mise à jour donné par

$$U_i = \sum_{j \in \Theta} \left(\sum_{Q \in Parents(Q)} W(Q) G_i^l(Q) \right)^{\alpha} c(i,j)$$

$c(i,k)$ désigne la compatibilité entre deux directions. Le coefficient α permet une accélération de la convergence. Classiquement, α augmente avec le nombre d'itérations. Le facteur de mise à jour U_i est calculé par une somme sur les 4 directions des produits entre l'avis des parents sur la direction j, et la compatibilité entre cette direction et la direction considérée i.

19.1 Le label "non-texte"

Le mécanismes de relaxation que nous venons de décrire impose une estimation de direction même pour les pixels qui se sont pas liés à du texte. De manière générale, tous les pixels des zones de textes sont perturbés par les pixels n'appartenant pas au texte et pour lesquels l'information de direction n'est pas pertinente. Pour palier ce problème, nous proposons de retirer du processus de relaxation les pixels dont la non appartenance au texte est quasi certaine. Pour cela, il suffit, dans le mécanisme de relaxation, de rajouter un nouveau label "N" qui désigne les zone de non texte. La décision sur la non appartenance à une zone de texte s'appuie sur la valeur maximale des réponses directionnelles. Une approche intuitive serait de procéder à une comparaison avec la valeur maximale théorique du filtre

$$G_N = G_{max} - \max_{i \in \Theta - \{N\}} (G_i)$$

où G_{max} désigne la sortie maximale du filtre. Malheureusement, celle-ci est liée à une configuration très particulière très peu probable dans une image, elle est donc inutilisable pour ce type de test. La seule mesure disponible est la valeur maximale observée dans l'image en cours de traitement.

$$G_{max} = \max_{i \in \Theta - \{N\}} (\max_{x,y} G_i(x,y)))$$

Cette définition du label "non-texte" suppose que l'image contient du texte. Dans ce cas, on peut admettre que la valeur maximale observée est liée à une zone de texte observée dans la direction adéquate. Dans le cas contraire, notre hypothèse conduit à la production de fausses alarmes qui devrons être détectées par les étapes ultérieures de type filtrage morphologique. En effet, on peut attendre des composantes connexes issues de zones de non textes (typiquement des textures) qu'elles ne valident pas les contraintes géométriques et morphologiques associées à une zone de texte.

Le seuil séparant le texte du non-texte est obtenu classiquement par seuillage automatique [59] des distributions des sorties des filtres directionnels. Comme nous avons 4 filtres, nous retenons la valeur maximale du seuil comme seuil final (G_t).

Nous incorporons ces valeurs maximales et de seuil dans un processus de normalisation des sorties des filtres

$$G_i{}' = s(G_i) \quad , \quad i \in \Theta = \{X, Y, R, L, N\}$$

où s est la fonction de normalisation qui tient compte de G_t et G_{max} :

$$s(x) = \begin{cases} \dfrac{x - G_t}{2(G_{max} - G_t)} + 0.5 & \text{si } x \geq G_t \\ \dfrac{x}{2G_t} & \text{sinon} \end{cases}$$

Dans tout le processus de relaxation, le label "N" est traité comme les labels de directions. Sa pyramide associée est construite en utilisant un filtre gaussien isotropique 3×3.

La fonction de compatibilité $c(i, j)$ est proche de zéro pour des directions nettement différentes et proche de 1 pour des directions similaires. (cf. table 8.5). Un label non-texte est uniquement jugé compatible avec un autre label non-texte et très incompatible avec toutes les labels de directions.

Label	X	Y	R	L	N
X	1	0	0.5	0.5	0
Y	0	1	0.5	0.5	0
R	0.5	0.5	1	0	0
L	0.5	0.5	0	1	0
N	0	0	0	0	1

Table 8.5: La fonction de compatibilité entre les directions.

19.2 Binarisation

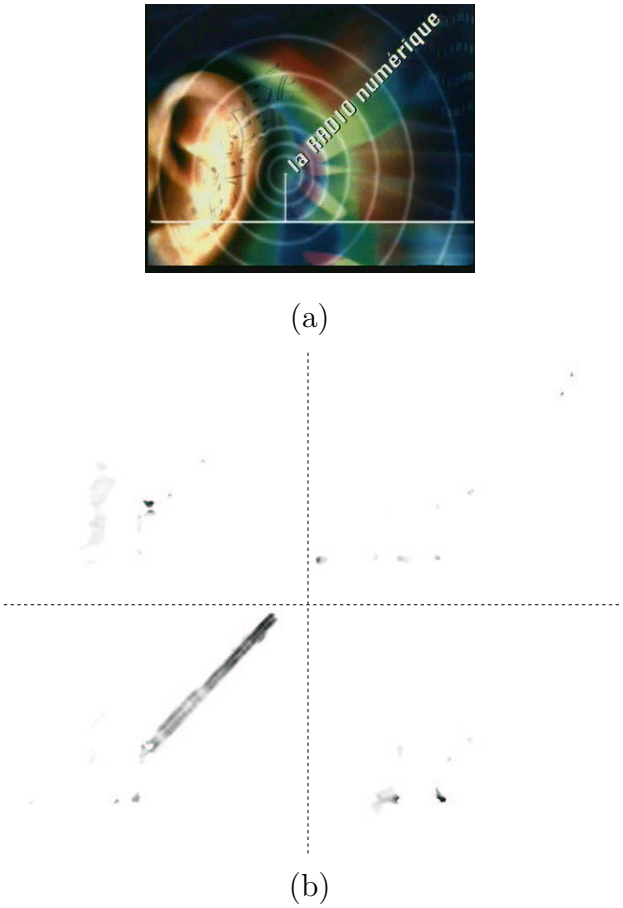
Un simple système de vote permet de déterminer, pour chaque pixel, son label de direction, associé à la valeur maximale de la sortie des différents filtres. Cette segmentation est combinée avec une segmentation "frustre" obtenue par seuillage automatique de la sortie initiale de chaque filtre. La sortie finale pour chaque segmentation est un "et" entre les deux segmentations. La sortie associée à l'absence de texte est un "ou" sur les 4 directions testées.

20 Résultats

Nous avons crée une base de 232 images afin de déterminer des valeurs appropriées pour les différents paramètres de notre méthode et procéder à des tests. Ces images

ont été partiellement extraites du corpus de l'INA [10] et partiellement d'une base fournit par notre partenaire France Télécom. Environ la moitié de ces images contiennent du texte artificiel et l'autre moitié des textes de scènes sans caractéristique constante. Les pixels de textes ont été marqués manuellement afin de construire une vérité terrain pour chaque type de texte (artificiel ou de scène) et pour les 4 directions.

Les résultats obtenus sont regroupés dans la table 8.6 en utilisant des mesures classiques de précision et de rappel. En plus, nous indiquons une mesure unique de performance (H) estimée par la moyenne harmonique des deux mesures précédentes [76]. On peut constater que la relaxation, surtout combinée avec des pyramides anisotropique et un cinquième label, augmente la performance de la détection de façon significative. La figure 8.21 montre le résultat de la relaxation appliquée à une image. On peut constater que les pixels détectés se concentrent sur quadrant bas-gauche de l'image, qui correspond au filtre diagonal, donc l'orientation du texte.



(a)

(b)

Figure 8.21: Image source (a) et résultat du vote après une relaxation anisotropique avec 5 labels (b).

[10] http://www.ina.fr

Type d. rel.	Labels	Rappel	Précision	H
Pas de Relax.	4	25.0	11.1	7.69
Pyr. isotr.	4	29.5	15.3	10.07
Pyr. anisotr. X,Y	4	34.1	23.7	13.98
Pyr. anisotr.	4	44.7	29.7	17.84
Pyr. anisotr.	5	**48.4**	**36.8**	**20.91**

Table 8.6: Résultats pour différents types de relaxation.

21 Conclusion et développements futurs

Les résultats de détection obtenus au niveau du pixel sont comparables avec ce que l'on obtient pour des textes artificiels. Il est maintenant nécessaire de prendre en compte des contraintes géométriques et de forme (sans prendre en compte des distorsions). Pour cela, il faut adapter les filtres morphologiques employés pour le texte artificiel (horizontal) à une direction variable. Enfin, un post-traitement pour réduire les fausses alarmes est possible en tenant compte de la dimension temporelle par *block matching* sous certaines hypothèses de mouvement. Ceci constitue le coeur de notre recherche actuelle.

References