Language-Free Layout Analysis

D. J. Ittner and H. S. Baird
AT&T Bell Laboratories
Murray Hill, NJ 07974, USA

Abstract

We describe a system for isolating blocks, lines, words, and symbols within images of machine-printed textual documents that is, to a large extent, independent of language and writing system. This is achieved by exploiting a small number of nearly universal typesetting and layout conventions. Our system does not require prior knowledge of page orientation (modulo 90°*), and copes well with nonzero skew and shear angles (within* 10°*). Also, it locates blocks of text without reliance on detailed* a priori *layout models, and in spite of unknown or mixed horizontal and vertical text-line orientations. Within blocks, it infers text-line orientation and isolates lines, without knowledge of the language, symbol set, text sizes, or the number of the text lines. Segmentation into words and symbols, and determination of reading order, normally require some knowledge of the language: we hold this to a minimum by relying on shape-driven algorithms. The underlying algorithms are based on Fourier theory, digital signal processing, computational geometry, and statistical decision theory. Most of the computation occurs within algorithms that possess unambiguous semantics (that is, heuristics are kept to a minimum). We discuss the effectiveness of the method on English, Japanese, Hebrew, Thai, and Korean documents.*

1 Introduction

We describe a system for the automatic segmentation of images of machine-printed pages into text blocks, lines, words, and symbols. In this paper, we emphasize features that allow it to be applied, with little or no modification, to text printed in a wide variety of writing systems and languages. We report algorithmic details of the system, discuss its strengths and weaknesses, and illustrate its behavior on English, Japanese, Hebrew, Korean, and Thai pages.

This is a subsystem of an experimental page reader [4] capable also of polyfont symbol recognition, contextual analysis, etc. Each of these topics has a large and interesting literature; for lack of space, we confine ourselves to a few references to papers which provide a careful review of the literature.

The system was implemented in the C programming language. Runtimes quoted are for a Silicon Graphics Power Series Model 4D/480S, with a 40MHZ IP7 processor.

Layout analysis consists of the following sequence of steps: (a) correct for non-zero skew and shear angles; (b) partition the page image into blocks of text; (c) infer text-line orientation within each block; (d) partition each block into text lines; (e) isolate symbols within each text line; and finally (f) merge symbols into words. Each of these steps will be described in Sections 3 through 8; Section 2 defines our problem.

2 Manhattan Textual Layouts

We focus on a class of layouts which we call *Manhattan textual layouts*, having the following properties. Pages contain blocks of text lines of symbols; in addition, many writing systems provide typographical breaks between words. The text should consist predominantly (not necessarily exclusively) of detached symbols (characters), as is customary in machine-print in almost all of the writing systems [10] of West and East Europe and the Americas (Latin, Greek, Cyrillic, Armenian, etc), East Asia (Japanese, Chinese, Korean, Tibetan, etc), Southeast Asia (Burmese, Thai, Lao, etc), India (Kannada, Sinhalese, etc), West Asia (Hebrew, Urdu, etc), and Africa (Amharic) – but not including Arabic and Devanagari and their derivatives.

We assume that all of the symbols in the page are printed *upright*. Text sizes must lie within known bounds $[s_{min}, s_{max}]$ (*e.g.* [5,24] point), but otherwise may vary arbitrarily within the page. We also assume that non-textual graphics or black noise fragments either do not occur, or are so large or small that they may be easily discarded during preprocessing.

The majority of characters in each text line must share a straight *reference line*, defined implicitly by typographical convention: for horizontal lines, this is usually called a baseline, and in vertical lines a centerline. In Manhattan layouts, after affine correction, text lines can have one of two *orientations*: horizontal or vertical. In most writing systems, one or the other of these is used exclusively; but in some both may occur on the same page. Within vertical lines, in all languages of which we are aware the *reading order* of symbols is top-to-bottom. In horizontal text lines, reading order may be, depending on the language, either left-to-right or right-to-left (we assume never both in the same language).

We define an *isolated block* of text to be a set of text lines all of the same orientation, spaced apart so that their symbols do not overlap when projected along their reference lines. This is consistent with normal typographical convention in which the closest line-spacing used is *set-solid* spacing, defined for this purpose. Within horizontal blocks the lines are read top-to-bottom. Within vertical blocks reading order may be, depending on the language, either left-to-right or right-to-left, but never both. Finally, each block in a Manhattan layout, after affine correction, should be surrounded by vertical and horizontal gaps of white space, which serve to separate it from other blocks.

Finally, a Manhattan layout possesses a single affine transformation that describes the skew and shear alignments over the entire image; this restriction may exclude some types of advertising copy and some large imperfectly-printed tabloid and newspaper layouts. In spite of these and other restrictions, the class of Manhattan textual layouts is large and useful.

3 Skew and Shear Correction

We assume that the page possesses a single dominant alignment orientation, at an angle within a few degrees of horizontal or vertical: this is normally determined by the reference lines of the majority of the text lines. The page may also possess an alignment roughly perpendicular to this: this may be due to the presence of fixed-pitch text, or to subtler alignments along the justified margins of text blocks.

We use a fast and accurate method of skew estimation [1]. For a given angle, we compute an alignment measure over the