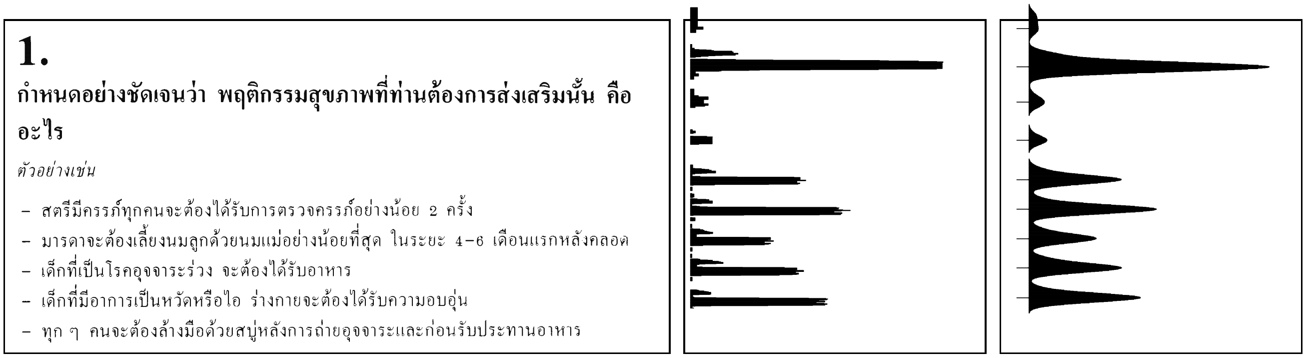

Figure 4: Left: Thai text block (after skew correction). Center: horizontal projection P; notice projection of well-separated vowels and tone marks. Right: smoothed projection S, also W shown as tick marks to the left. All text lines are isolated properly.

7 Isolating Symbols in a Text Line

The preliminary segmentation of components into symbols fails on touching or badly fragmented symbols. For these cases we have developed two methods: one specialized to fixed-pitch text, and the other applicable also to variable ("proportional") pitch. Both are sharply restricted to writing systems in which symbols can be separated by straight-line cuts roughly at right angles to the reference line (and so are not suitable for printed Arabic).

So that the system need not be told in advance, we have developed an algorithm, based on spectrum analysis, to automatically detect for the presence of, and then to estimate, fixed pitch. We first compute the projection profile for the text line (optionally, for the block) perpendicular to the orientation. We project both pixels and slightly shrunken rectangular boxes of components (a compromise between too much and too little detail). The projection is normalized to produce a signal in the range [-1,1] with mean 0. The squared magnitude of the output values from an FFT performed directly on this 1-D signal is analyzed for a single dominant frequency component. If a single dominant frequency is present, it implies a pitch p; if not, we assume the text is printed in variable pitch.

Given pitch p, we make a second pass over each text line, sweeping a "cut point", splitting and merging components as necessary. When splitting, we use the point of minimum energy in the detailed pixel projection within $0.1p$ of the expected break point. This is fast and often sufficient.

But we do not require that the block be printed on a strict grid in both directions. We accommodate small local deviations from p by slowly adjusting the cut point based on local conditions. Figure 5 shows a block from the Japanese page of Figure 1. The vertical projection profile produced a signal of length 1621; it was decimated by 4, then zero-padded to 1024. The DFT frequency with maximum squared magnitude corresponds to a pitch of 9.24 symbols per inch (note a secondary peak at twice that frequency). Pitch detection and estimation for this block required 0.22 CPU s.

Figure 5 also illustrates some challenges of Japanese text. When Latin text (Romaji) occurs, it is typically not printed using the same pitch as the surrounding Kanji and Kana. A more subtle difficulty can be seen in the sixth line: the symbols in the right half of the line are squeezed to accommodate an extra character. Our segmentation algorithm merged 7 pairs of Latin letters in the second line. All segmentation of the Japanese Kanji and Kana was correct, including the sixth line.

The use of spectrum analysis is efficient and highly accurate for detecting and estimating fixed pitch. The signal can be strongly decimated and the method will still succeed on lines containing as few as 4 symbols. Zero padding in the image domain also allows us to increase resolution in the frequency domain, achieving high accuracy in estimation. However, one troubling characteristic of the method is that there are often secondary peaks in the frequency domain at multiples of the pitch. We compensate by including into the major peak the energy of secondary peaks which are near-integral multiples of the first. This simple heuristic is effective, although we hope to better understand the causes of multiple peaks, and if possible to eliminate them.

Fixed-pitch analysis, although often accurate, is only a first stage: afterwards, we attempt resegmentation wherever symbol-recognition confidence scores are low. In this case, if fixed-pitch processing has occurred, then we first reexamine the original segmentation. Thereafter, splitting and merging are guided by classification confidence scores in a branch and bound manner described in [9].

8 Isolating Words in a Text Line

While linguists may not agree on a definition for "word," they do agree that words are essential units of every language [7]. In most alphabetic writing systems word-breaks are marked typographically, often by spacing (in Tibetan a printing word-mark symbol is used). Chinese and Japanese, by contrast, do not mark word-breaks. Other writing systems mark some but not all, for example in Sanskrit where word-spaces appear only when they coincide with breath pauses. For these reasons, isolating words within a line of text is impossible, in general, without prior detailed knowledge of the writing system.

Our system must be instructed at run-time whether or not the input uses word spaces. If it does not, we make no attempt to locate words at this stage, deferring it until contextual analysis. We believe that in the hardest cases, linguistic or semantic models may be required[1].

If the input language uses word spaces, we infer a scalable word-space threshold t for each text block separately. Because we want the threshold to be independent of text size, we cannot locate words until text size is estimated. Since an accurate estimate of point size cannot be made without knowledge of the symbol set (or even font style), we do not isolate words until after symbol recognition.

The word-space threshold t is arrived at by first extracting all inter-symbol distances parallel to the text-line orientation, scaled by local text size. The distribution of these scaled distances is typically bimodal with one mode for inter-symbol spaces and the other mode, typically smaller, for inter-word spaces. Locating a reliable threshold between peaks is done with a simple minimization technique: the distribution is smoothed by a Gaussian kernel of standard error 0.03 em, and the minimum (or mean of multiple minima) is chosen. This typically lies

[1] But may not be available: in some languages word segmentation, even given correct text, appears to be an open problem in computational linguistics.