Text/Graphics Separation Revisited

Karl Tombre, Salvatore Tabbone, Loïc Pélissier,

Bart Lamiroy, Philippe Dosch

August 2002

Text/Graphics Separation

- X–Y trees, white streams, etc. – adapted to text-rich documents
- RLSA filtering – but few attempts for graphics-rich documents
- Forms – mainly horizontal and vertical lines – look explicitely for lines (Hough, etc.)
- Directional morphological filtering
- Explicit search for lines on DT or vectorization
- **Analysis of connected components** $\longrightarrow$ improvement on [Fletcher & Kasturi]

Why choose F&K?

- Because it’s there
- Stable on variety of documents
- Scalable
- Not many thresholds, and easy to master
- A reference method, well explained, sound, and known to many other people

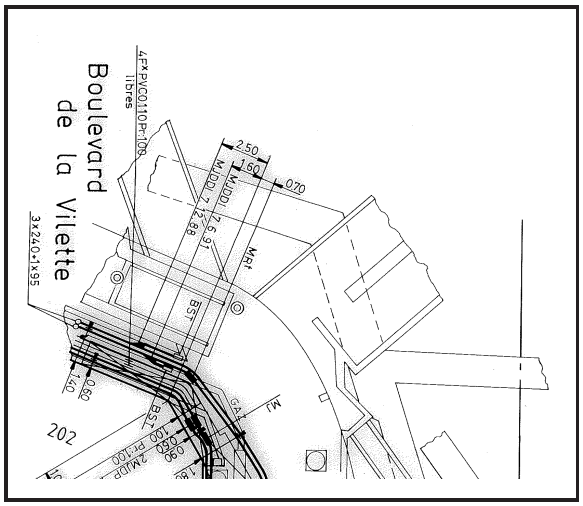
Limitations of F&K

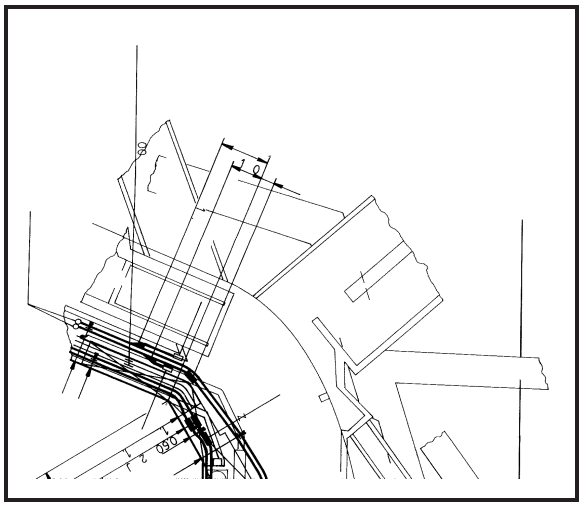
- Designed for mixed text–graphics documents $\Rightarrow$ minor adaptations to graphics-rich documents (absolute constraint on length and width of component)
- Does not separate dashes from elongated symbols (I, l, ...) $\Rightarrow$ separate size filtering from shape filtering, and add a third layer
- Text touching graphics $\Rightarrow$ post-processing text recovery step

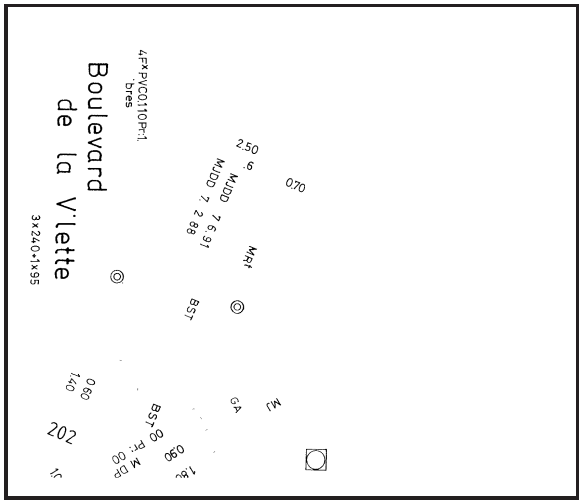
Modified algorithm

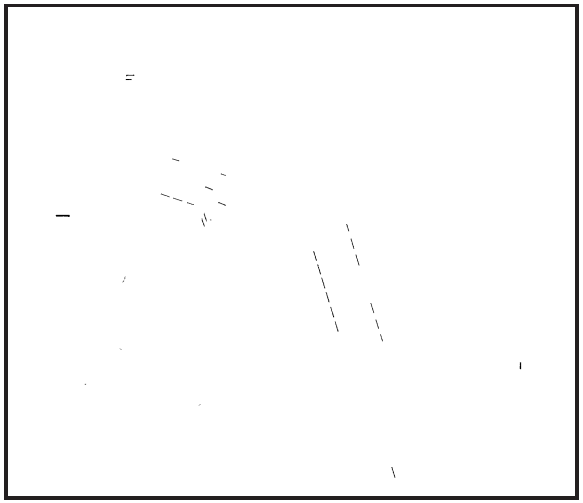
- compute CCs and histogram of BB sizes
- find most populated area A_{mp} and A_{avg}, number of CCs of average size
- set $T_1 = n \times \max(A_{mp}, A_{avg})$ and T_2 (thresholds on BBs)
- move to text layer all black CCs $< T_1$, and $\frac{\text{height}}{\text{width}} \in [\frac{1}{T_2}, T_2]$, and both height and width $< \sqrt{T_1}$
- compute **best enclosing rectangle** (BER) of each “text” component
- set T_3 and T_4 on BERs
- Reclassify “text” CCs with density (wrt BERs) $> T_3$ and elongation $> T_4$ as small elongated shapes

$$T_1 = 1.5 \times \max(A_{mp}, A_{avg}), T_2 = 20, T_3 = 0.5, T_4 = 2$$







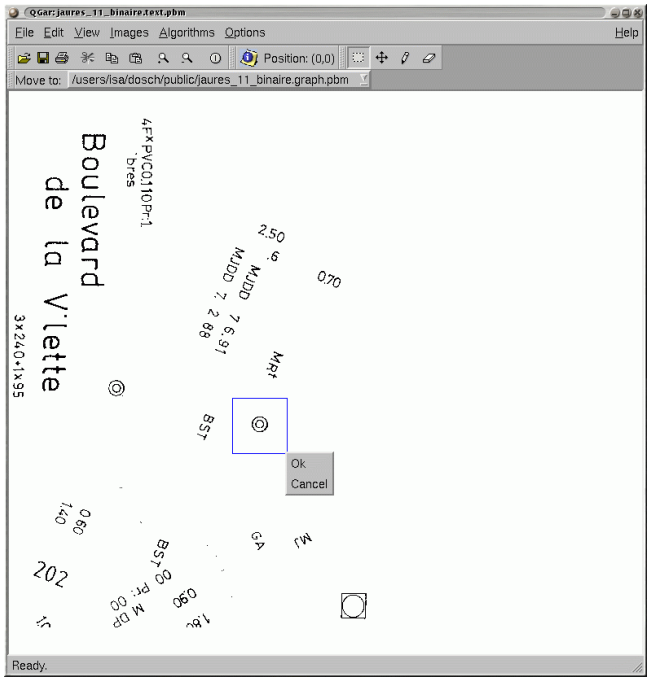


Stability of thresholds

- ✓ T_1 proportionnal to $\max(A_{mp}, A_{avg})$, with n stable if only one character size ($n = 3$ OK for very homogeneous character set)
- ✓ $T_2 = 20$ good for all documents we have worked on
- ✓ $T_3 = 0.5$ if noisy character contours (limitation of BER)
- ✓ T_4 dependent on kinds of dashes present in drawing

Possible improvements

- ✓ Analysis of size and elongation distributions could be made less empirical
- ✓ Better elongation and size descriptor than BER (second-order moments)
- ✓ A fourth layer, that of dots (alignment problems in next step)
- ✓ Still, man must be in the loop...



Extracting the Strings

Based on Hough Transform working on bounding boxes of text layer components:

- sampling step of HT set to $chdr \times H_{avg}$
- look for alignments by voting in (ρ, θ) space
- segment each alignment into words:
  - compute mean height $\bar{h}$
  - group all successive characters separated by less than $\mu \times \bar{h}$

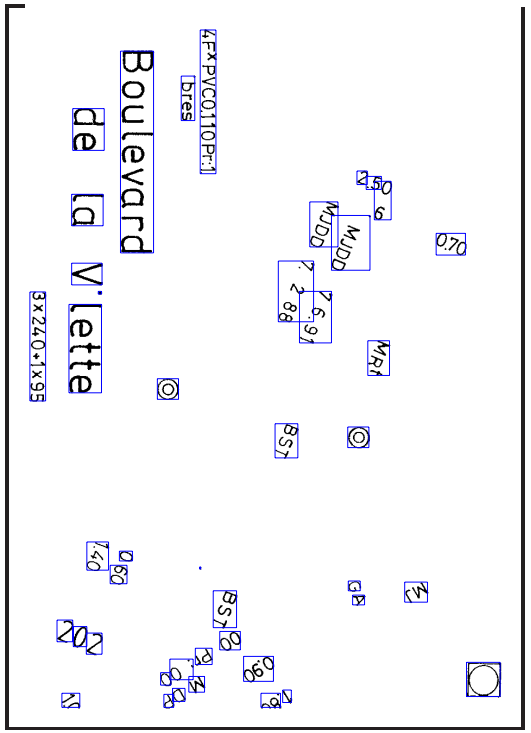
2 options:

1. process first the highest votes of the HT, and do not consider characters already grouped in a first alignment when processing lower votes;
2. give the possibility to each character to be present in more than one word hypothesis, and wait until all votes are processed before eliminating multiple occurrences, by keeping the longest words.

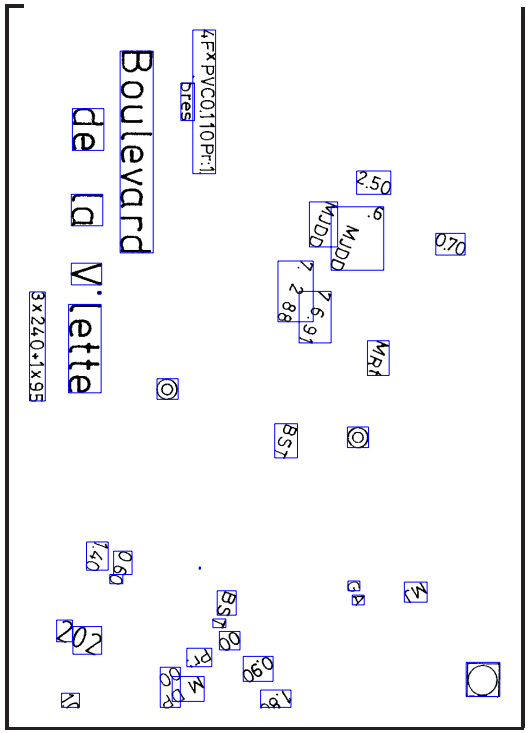
$\Rightarrow$ No clear winner

Choice of parameters

✓ $chdr$: adjusts sampling step of HT. Difficult to stabilize – false clusters, or over-segmentation

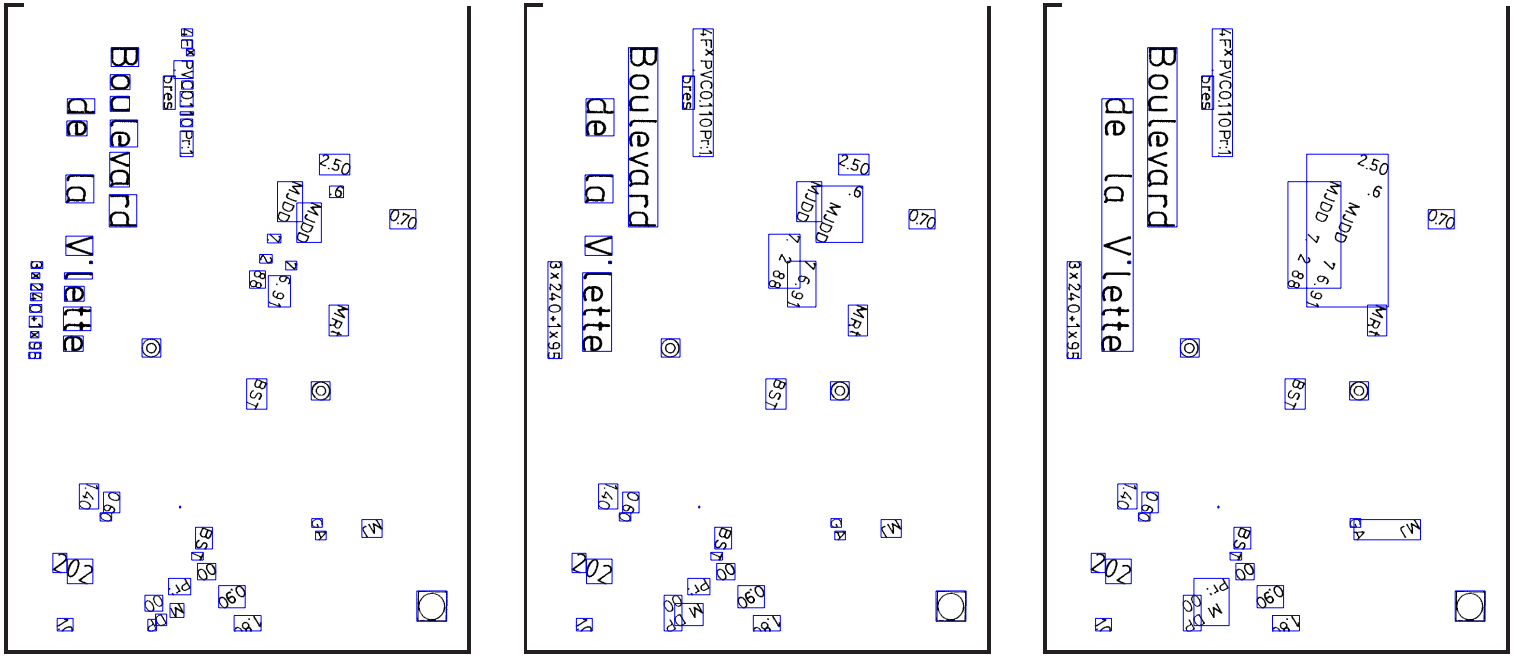


$chdr = 0.2$



$chdr = 0.4$

✓ μ: adjusts maximum distance allowed between characters in a same string. Default value 2.5 seems to be quite stable



$\mu = 1.5$ $\mu = 2.5$ $\mu = 5.0$

Possible improvements

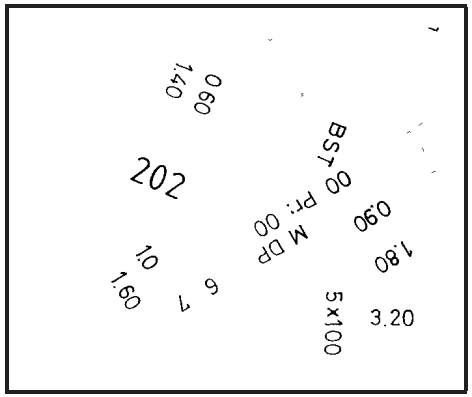
- ✓ Short strings not reliably detected $\longrightarrow$ hierarchical strategy to refine thresholds when lowering string length
- ✓ Artificial diagonal alignments $\longrightarrow$ heuristics on privileged directions
- ✓ Refinement of string orientation for short strings $\longrightarrow$ post-processing by Radon transform for short strings (3–4 chars)
- ✓ Punctuation signs, points on “i” characters and other accents $\longrightarrow$ extract them to a 4th layer and add them after string segmentation

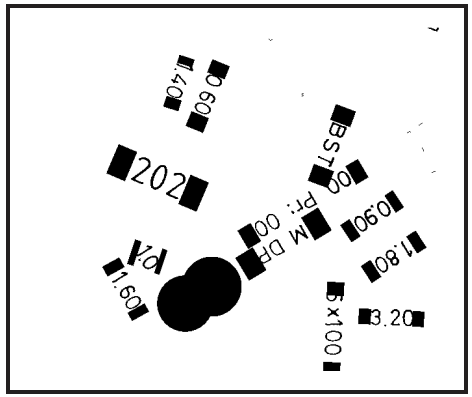
Recovering Touching Characters

- General problem with CC based methods
- In our case, no a priori knowledge on orientation (such as in forms) or on stroke width
- General idea: extend strings found by previous step (thus, method does not work if everything touches!)

Outline of method

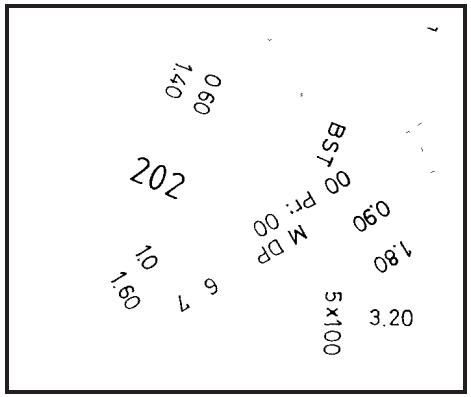
- compute equation of best line passing through all string characters
- compute enclosing rectangle of string along direction, and define search areas (circle if only 1 char in string)
- look for characters in these areas, first in 3rd and 4th layer, then by segmenting skeleton

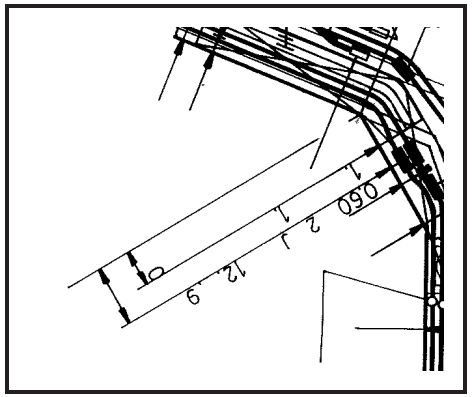


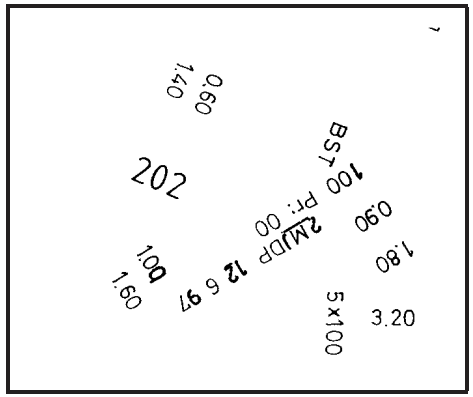


Segmentation of the Skeleton

- Compute 3–4 distance skeleton in search area
- Segment skeleton into subsets connected to skeleton outside search area by one and only one multiple point
- Retrieve candidate character fragments
- Reconstruct using inverse distance transform







Limitations

- ✓ method does not retrieve a string completely connected to the graphics (no seed string)
- ✓ if string orientation not correct (regression for short strings not robust), some characters may be missed
- ✓ heuristic leads to non extraction of characters intersecting search area at 2 ore more points

Evaluation

Image	Nb. ch.	T/G	Retr.	Total	Errors
IMG1	63	50 (79%)	8/13	58 (92%)	7
IMG2	92	66 (72%)	5/16	71 (77%)	24
IMG3	93	78 (84%)	3/15	81 (87%)	5
IMG4	121	95 (78%)	9/26	104 (86%)	71
IMG5	31	7 (22%)	0/0	7 (22%)	1

Conclusion

- ✓ Robust, stable and well-mastered method
- ✓ Recovery of touching characters for a given class of problems
- ✓ Still room for improvements
- ✓ No panacea $\longrightarrow$ we still need to put man in the loop