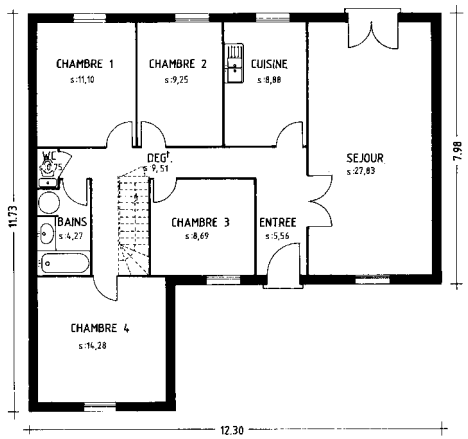
Vectorization in Graphics Recognition: To Thin or not to Thin

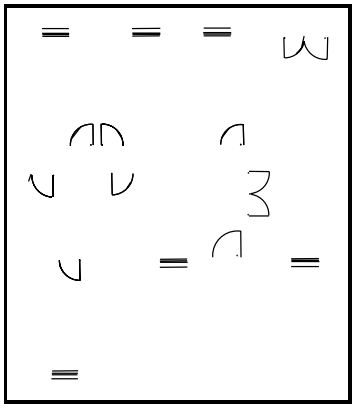
Karl Tombre and Salvatore Tabbone

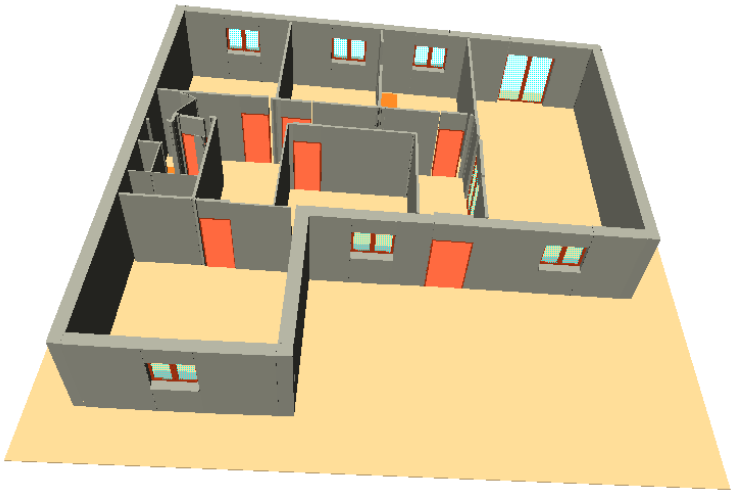
September 2000

Graphics recognition

Analysis of graphics-intensive documents







Main feature = the *vector* $\Rightarrow$ raster-to-vector conversion plays a central rôle

Choice of vectorization method

Compute a medial axis or not?

- ➥ medial axis computations are classical for image processing people
- ➥ skeletons often used
- ➥ faithful representation of binary shape $\neq$ retrieve actual thoughts of draftsman
- ➥ Special problems with junctions and line extremities

The steps of vectorization

1. Find the lines ←
2. Approximate the lines into vectors
   - ➥ Rosin & West split and merge
   - ➥ Wall & Danielsson iterative
3. Add contextual information
4. Dashed-line detection
5. Arc detection → *poster presented yesterday*

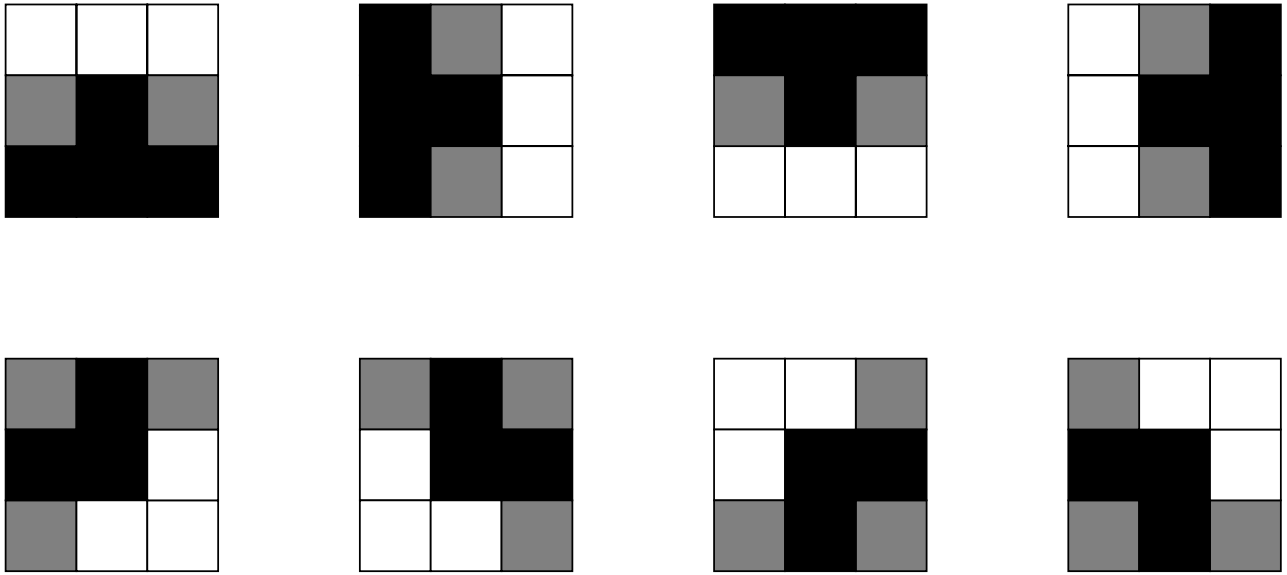
Finding the lines

From raster image to lines (chains of pixels) $\longrightarrow$ significant medial axes

1. Compute the skeleton
   - iterative thinning
   - distance transform
2. Match the opposite sides of the line
3. Sparse-pixel approaches

Skeletonization

Iterative thinning : “peeling an onion”



- small image buffer size
- multiple passes, sensitive to noise

Distance transform : 3–4 DT [Sanniti di Baja]

V_2	V_3	V_4
V_1	X	V_5
V_8	V_7	V_6

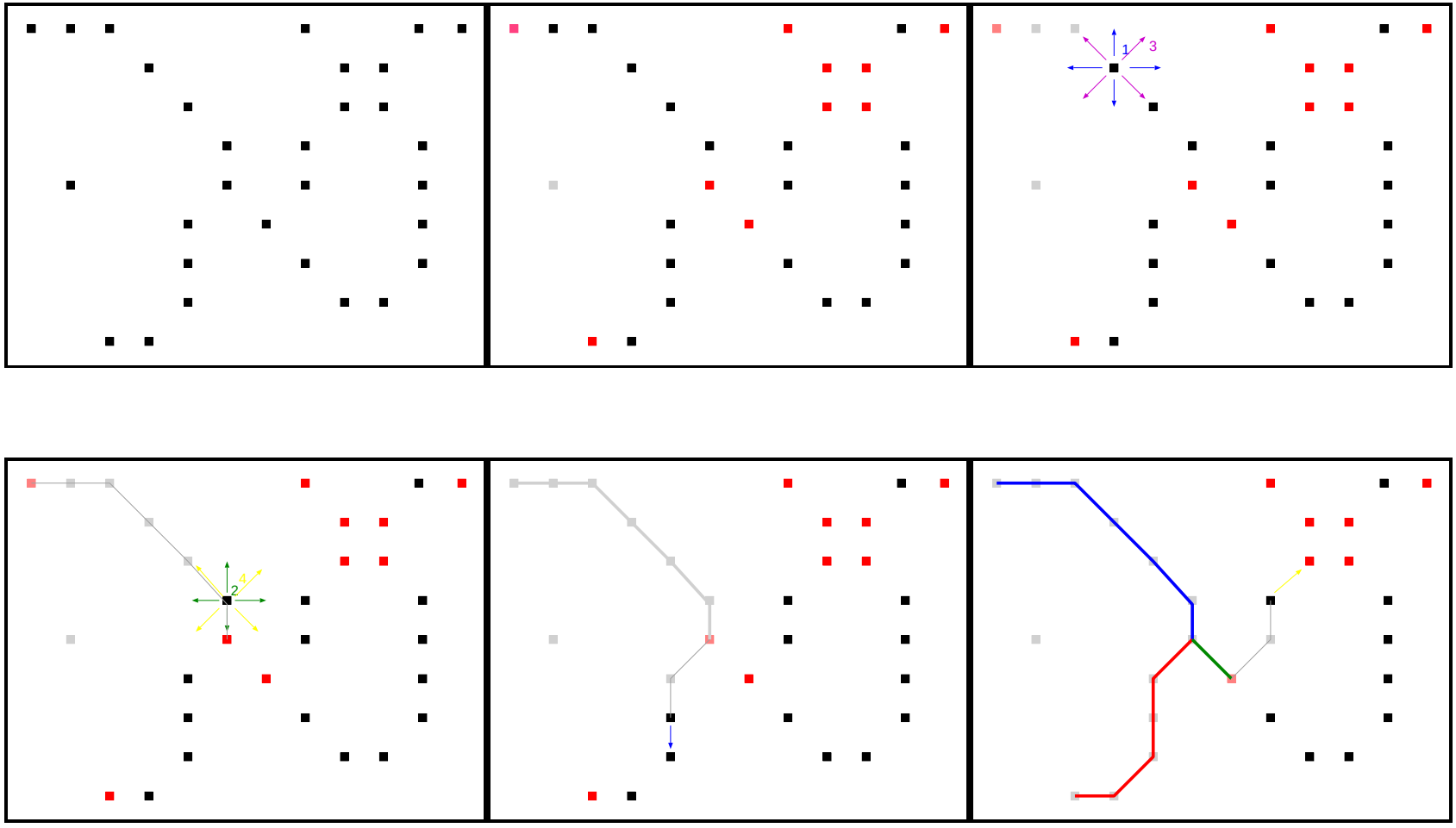
Pass 1 : $Y = \min(V_1 + 3, V_2 + 4, V_3 + 3, V_4 + 4)$

Pass 2 : $Z = \min(Y, V_5 + 3, V_6 + 4, V_7 + 3, V_8 + 4)$

- Detection of local maxima
- Tracking the skeleton
- Reduce to thickness 1
- Barb removal

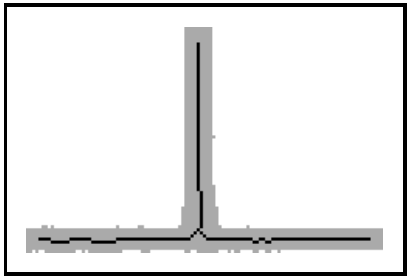
$\longrightarrow$ good compromise w.r.t. precision, simplicity and robustness

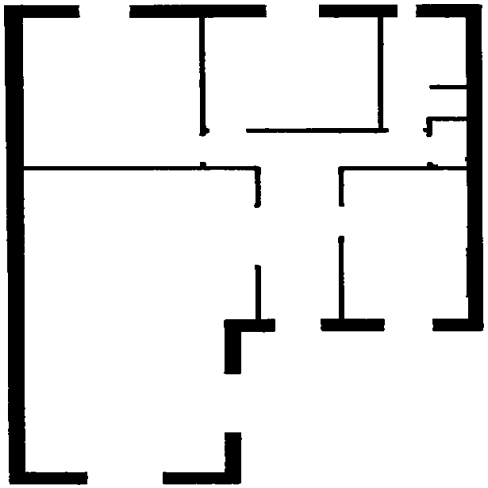
Linking the skeleton

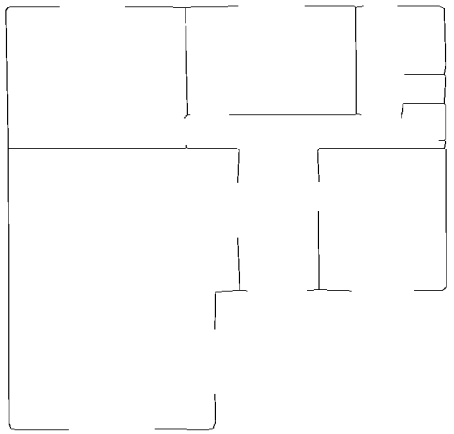


Main weakness

Skeletonization displaces junction points



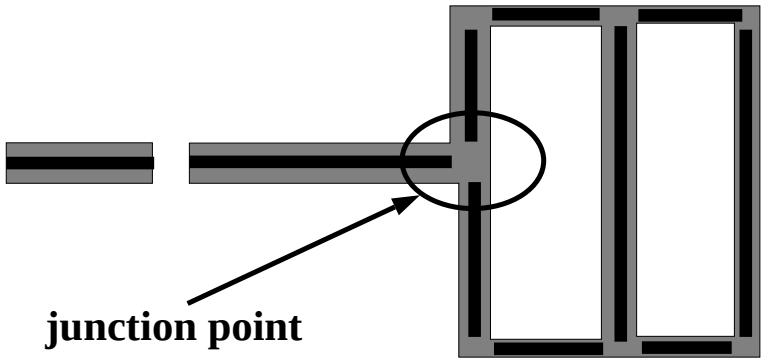




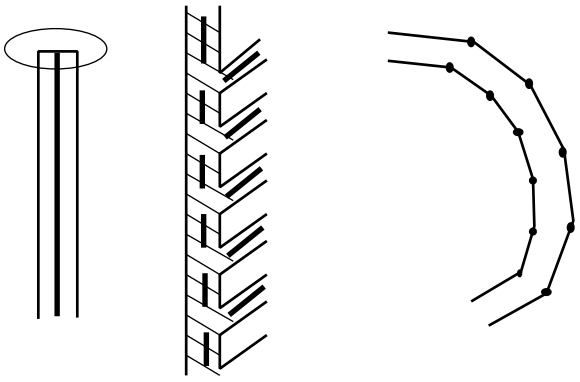
Original image Polygonal approximation of 2–3 skeleton

Matching opposite contours

REDRAW: polygonal approximation of connected components contours, and match segments

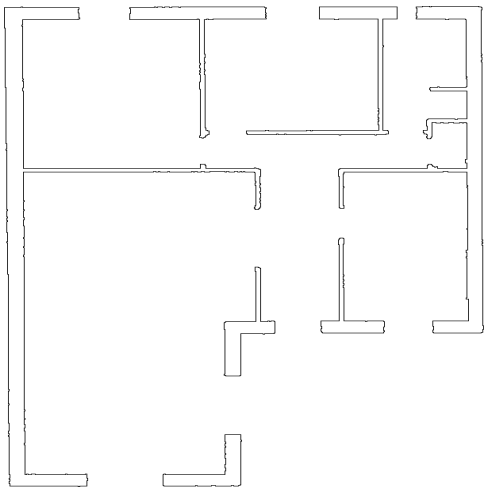


Problems: thresholds, one-to-many and many-to-many matches

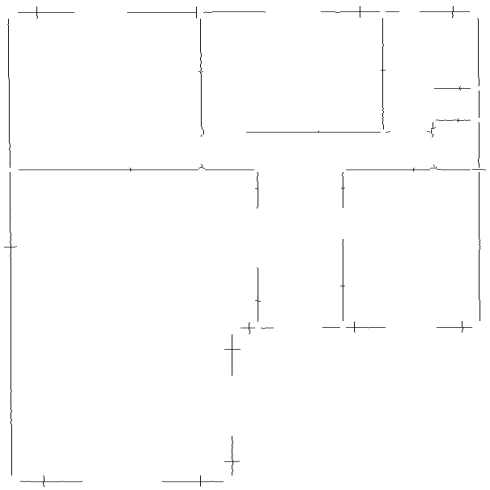


Improvement: use the gradient orientation

Gradient vector (Canny operator) perpendicular to contour $\Rightarrow$ dense and robuste matching of each contour point in the direction of the gradient



Contour detection



“Skeleton” obtained by dense matching

The steps of vectorization

❶ Find the lines

❷ Approximate the lines into vectors

- ➥ Rosin & West split and merge
- ➥ Wall & Danielsson iterative

❸ Add contextual information ←

❹ Dashed-line detection

❺ Arc detection → *poster presented yesterday*

Contextual information

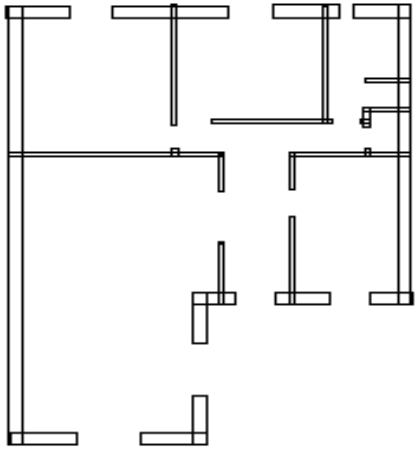
Add domain knowledge

- ✓ specific documents $\longrightarrow$ ad hoc vectorization (e.g. [Chhabra])
- ✓ heuristic correction $\longrightarrow$ adds new thresholds and parameters
- ✓ add constraints [Röösli & Monagan]
- ✓ optimize junction position through iterative fitting [Janssen]
- ✓ introduce models of ideal junctions (T, L, Y . . .) and correct by fitting

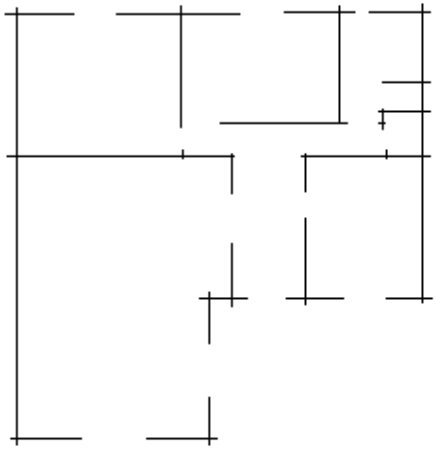
Knowing what you look for

The case of architectural drawings: looking for the walls

Dense contour-matching followed by search for horizontal and vertical rectangles



Reconstructed rectangles



Medial axes of rectangles

Discussion

- Contour-matching + contextual information yield good results...
- ... but in general case, contextual information not exhaustive
- Skeletonization is generic and robust, but yields weak junctions
- skeleton: emphasizes *distance* criterion
- dense matching: emphasizes *direction* criterion

A possible solution

Three-step vectorization under study:

- first crude vectorization to segment a drawing into pieces having same direction
- directional distance transforms on each piece
- junction models for better precision at intersections

Conclusion

- ✓ Skeletons still most general and generic
- ✓ Junctions and extremities still not satisfactory
- ✓ Combine directional and distance criteria
- ✓ Keep an eye on emerging methods (Markov Random Fields, ...)