Contextual Junction Finder

J. Matas and J. Kittler
Dept. of Electronic and Electrical Engineering,
University of Surrey,
Guildford, Surrey GU2 5XH, United Kingdom

Abstract

A novel approach to junction detection using an explicit line finder model and contextual rules is presented. Contextual rules expressing properties of 3D-edges (surface orientation discontinuities) limit the number of line intersections interpreted as junctions. Probabilistic relaxation labelling scheme is used to combine the a priori world knowledge represented by contextual rules and the information contained in observed lines.

Junctions corresponding to a vertex (V-junctions) and an occlusion (T-junctions) of a 3D object are detected and stored in a *junction graph*. The information in the junction graph is used to extract higher level features. Results of the most promising method, *the polyhedral object face recovery*, are briefly discussed. The performance of the junction detection process is demonstrated on images from indoor, outdoor, and industrial environments.

1 Introduction

Perceptual groupings of image features have been widely used in computer vision systems to guide scene interpretation and 3D model matching [1, 2, 5, 9, 10]. Of all perceptual groupings studied by psychologists [13, 4] and computer vision researchers we focus our attention on *junctions of line segments* - points of co-termination of lines. As co-termination is a projection-invariant property, the task of junction detection would be relatively simple in an ideal noise-free world. A set of lines terminating at the same point could be interpreted as a projection of edges meeting at a vertex. However, due to the inherent inaccuracy of line (and edge) detection, endpoints of lines can be widely separated even if the lines emanate from a common vertex.

In a novel approach, an explicit error model for line detection in conjunction with *contextual rules* is used to recover junctions. The contextual rules express physical properties of 3D-edges (discontinuities in surface orientation): 1. projections of 3D-edges never cross and 2. visible parts of 3D-edges terminate at either a vertex or a point of occlusion. Both the use of context and an empirically tested explicit error model is a distinguishing feature of the work presented in the paper.

The problem we are facing can be stated as follows: For every line A, find line B which is most likely the line that occluded/had a common vertex in 3D with A. Of all possible assignments for A and B that don't violate rule 1. select the most probable one given the line detector error model. A probabilistic relaxation scheme developed in [6] is applied to the junction detection problem (Section 3). Section 2 specifies the line detection model. Implementation issues

BMVC 1992 doi:10.5244/C.6.13

are addressed in Section 4. Section 5 presents intermediate level groupings built on top of the junction finder The results are summarised in Section 6.

2 Line detector model

Conceptually, the first stage of the junction finder can be viewed as an attempt to recover *projected lines*. A *projected line* is, by definition, a projection of a 3D-edge and therefore must terminate at an intersection of 2 or more projected lines. A set of all projected lines would be very close to an 'ideal line drawing' (see Fig. 1(c)).

Any endpoint detected by a line finder can be treated as a noisy measurement of a projected endpoint. A statistical model of the noise affecting the line finder is a necessary prerequisite for any attempt to recover projected lines. The junction detection becomes trivial if the filtering process is successful - any point where two lines touch is a junction. See Figs. 1(a)-(c) to compare the line finder output, junction finder output (filtered lines) and a set of projected lines.

The uncertainty in line parameters is a function of the particular edge and line finder used. After extensive (but subjectively evaluated) tests we selected Horaud's implementation [5] of the Deriche filter [3] for edge detection and a line detector based on Hough transform [12, 14] for line detection. In agreement with [9, page 367] we observed:

- very precise estimation of the line angle and of the transversal position
- large uncertainty in the localisation of the endpoint

Simplifying the characteristics of the line finder the following model was adopted: 1. the projected endpoint lies on the straight line defined by the line segment. 2. if d denotes the (oriented) distance from detected endpoint E with positive values of d for point outside the line segment, then the endpoint error distribution is

$$P(d) = \begin{cases} \frac{k_{in}k_{out}}{k_{in}+k_{out}} e^{k_{in}d} & \text{if } d < 0 \\ \frac{k_{in}k_{out}}{k_{in}+k_{out}} e^{-k_{out}d} & \text{otherwise} \end{cases} \quad (1)$$

The two constants, k_{in} and k_{out} control the shape of the exponential. At present the values are set to 0.5 and 0.1 pixels respectively (Fig. 2). The choice of exponential is somewhat arbitrary, but Fig. 3 shows that it is in good agreement with empirical data. Test runs with different k_{in} (range 1-0.2) and k_{out} (0.2-0.03) produced similar results suggesting that the performance of the junction finder is not critically sensitive to the shape of the distribution.

3 Junction detection using probabilistic relaxation labelling

Visible parts of 3D edges terminate at either a vertex or a point of occlusion. A *junction* is a projection of such 3D point. The line finder model (section 2) guarantees that all junctions lie at an intersection of straight lines passing through a detected 2D line. Consider the example of Fig. 4. If line A is a

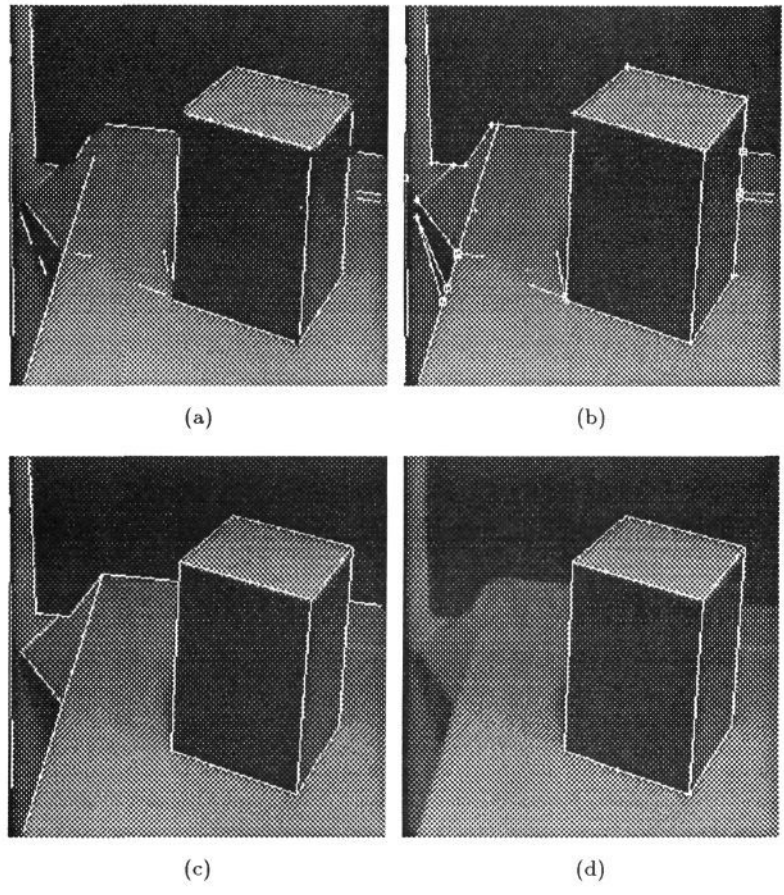


Figure 1: CUBE. A simple scene with a test object used for stereo camera head calibration in the VAP project [15]. In Fig. (a), lines detected by Hough transform are superimposed over the original image. Note that errors in orientation and transversal position of line segments are negligible. Fig. (c) gives an example of an 'ideal line drawing' of the CUBE scene. The image was prepared by editing the result of the junction finder shown in (b). Fig. (b) illustrates junction finder results. Endpoint errors are filtered out by 'stretching' line segments to junctions. The only significant structural error occurred at the top-left vertex of the cube. The left vertical edge of the cube is associated with the rear edge of the table. The result is explainable; the line corresponding to the vertical edge was terminated very close to the projected rear table edge because there is no gray level gradient between the front face of the cube and the background. Results of the *face recovery* postprocessing (Section 5) are shown in Fig. (d). Using geometric information enabled the recovery of the front face despite the top-left vertex problem described above. *Gap bridging* postprocessing (Fig. 6) joint the two lines on the rightmost vertical edge of the cube.

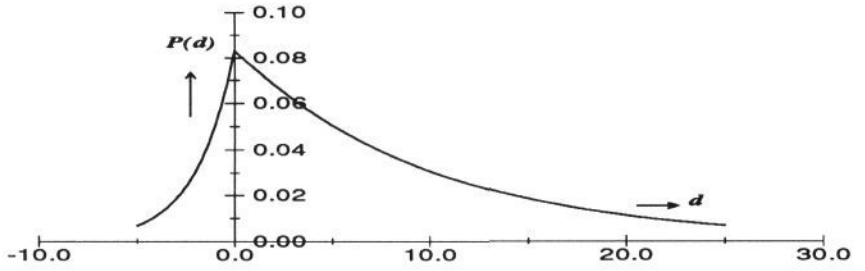


Figure 2: Assumed endpoint error distribution of the line finder. d denotes the distance from an enpoint along the line segment. Negative values refer to points inside the line segment.

projection of a 3D edge it must terminate at a junction that is identical to one of the intersection I_{ab}, I_{ac}, I_{ad}. We are concerned with the problem of computing the probabilities of all events $\{A_e = I_{ai}, \forall i\}$. To exploit the contextual information conveyed by these interacting events we compute the probabilities using a dictionary based relaxation scheme whereby the probabilities at stage $n+1$ are obtained from probabilities $P^n(A_e = I_{ai})$ at the previous iteration, ie.

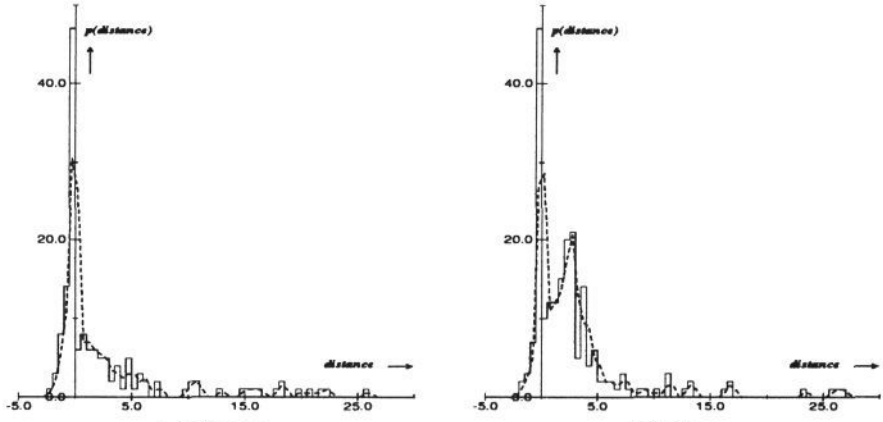
$$P^{n+1}(A_e = I_{ai}) = \frac{P^n(A_e = I_{ai}) \sum_{\Lambda^k} P(\Lambda^k) \prod_l \frac{P^n(l_e = I_{em})}{P(l_e = I_{em})}}{\sum_j P^n(A_e = I_{aj}) \sum_{\Lambda^k | A_e = I_{aj}} P(\Lambda^k) \prod_l \frac{P^n(l_e = I_{em})}{P(l_e = I_{em})}} \quad (2)$$

The initial probabilities $P^0(A_e = I_{ai})$ are computed using the Bayes formula from the probability distribution of endpoint errors (Eq. 1, Fig. 2) and prior probabilities. All prior probabilities in Eq. 2 are assumed to be equal. The context is introduced using a dictionary Λ that contains all permissible configurations of junction assignments. The dictionary is constructed using to rules obeyed by projections of 3D edges: 1. projected lines must not cross and 2. every projected line must terminate at a single junction.

4 Implementation of the Junction finder

The junction detection process is performed in three stages - initialisation and preprocessing, relaxation, and postprocessing. In the first stage, all intersections of line pairs are considered as possible junctions. Any intersection with endpoint distance outside the margin of error of the line detector is immediately discarded. Each intersection is assigned an initial, non-contextual probability according to Bayes formula. The information associated with every intersection is stored in a node of an *intersection network*. Each node in the network is linked to four other nodes representing its predecessor and successor (with respect to distance) in the list of intersections of one line (Fig. 5(a)).

In the relaxation stage, repeated sweeps through the intersection network are made. At each endpoint, the probability distribution is updated according to context-conveying formula 2. Generally, the probabilities of an intersection being a junction gradually shift either towards 0 or 1. At the end of the sweep, intersections with 0 probabilities are deleted (it follows from formula 2 that



(a) HOUSE (b) TOYS

Figure 3: Histogram of enpoint distance errors. The output of the junction finder, the *junction graph* is assumed to represent the ground truth (see Fig. 8 to check the validity of this assumptions). Histogram of line end to corresponding junction distances (solid line) shows good agreement with the line detection model (Fig. 2). The dashed line graph shows the running average of two consecutive histogram bins (of 0.5 pixel size).

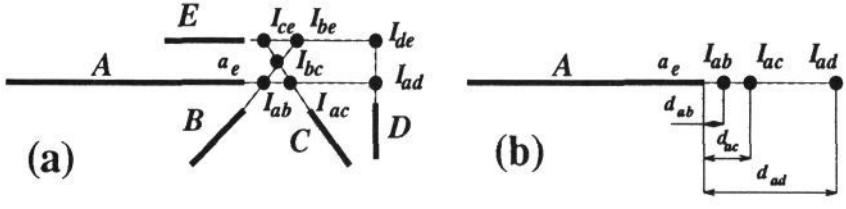


Figure 4: A set of interacting lines

the probability would remain 0). The originally dense network (Fig. 5(a)) is gradually transformed into a structure similar to Fig. 5(b). The iteration loop is exited when the average relative change of intersection probability falls under a preset threshold (default: 2%).

Finally, only intersections with maximum probability at an endpoint are retained and labelled as either V or T junctions. An intersection having a maximum probability with respect to both participating lines is assumed to be a projection of a vertex; the intersection is labelled a V-junction. If the intersection probability is a maximum with respect to one line only then the other line must 'pass through' (other possibilities are suppressed by the relaxation process). The situation indicates that the intersection is a projection of a point where a 3D-edge was occluded; the intersection is labelled as T-junction. The network of intersections is transformed into an attributed graph structure called a *junction graph*, (Fig. 5(c)). Every node of the junction graph represents a junction relation (either V or T) between a pair of lines. The junction

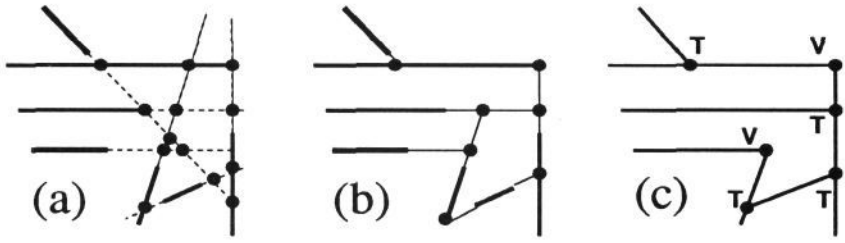


Figure 5: From *intersection network* to *junction graph*. The large margin of error of the line detector allows for multiple interpretations of an endpoint. An *intersection network* structure is created to facilitate the probabilistic updating of intersection probabilities. During the relaxation process, the network becomes sparser as probabilities of some intersections drop to 0 (see Tab. 1). Finally, all non-maximum junctions are discarded - a *junction graph* is created (c).

graph is a semi-symbolic structure; numerical information is attached to both junctions (position, probability) and lines (position). The junction graph can be used for both symbolic and geometric reasoning about the scene (see section 5).

image	HOUSE		WIDGET		TOYS		CUBE	
time	1.6s		0.7s		2.1s		0.4s	
lines	78		44		90		28	
iteration	inters.	change	inters.	change	inters.	change	inters.	change
1	563	84.16	247	84.01	800	86.60	107	83.72
2	368	75.08	164	67.00	489	73.34	64	47.00
3	142	24.03	85	50.29	204	25.65	43	7.87
4	142	2.09	85	7.15	204	2.80	43	1.26
5	142	5.73	85	3.20	204	5.65	43	6.49
6	142	0.38	85	20.41	204	1.04	43	0.99
7			85	3.36				

Table 1: Junction finder performance. The processing time was measured on a SPARC 2 machine. The 'inters.' column shows the number of intersections processed in the $n-th$ iteration. The 'change' column contains information about an average change (in %) of the intersection probabilities

The efficiency-minded readers may express doubt about the speed of the process. The computational complexity of the implementation of preprocessing is $O(N^2)$ (where N is the number of input lines) as all line pairs are examined (an $O(NlogN)$ algorithm can be found in [16]). The theoretical worst-case complexity of the iterative relaxation is even worse. Fortunately, the worst-case complexity is not of practical importance (representing a situation where all lines terminate in a tight cluster); the *average* complexity is extremely hard to derive analytically, but empirical results suggest $O(NlogN)$. Table 1 summarises the junction finder performance. For run-times in the order of seconds 1. optimising performance of the junction finder is not of particular importance

and 2. the performance depends more on system specific constants (i/o speed etc.) than on the computational complexity. The good performance is a consequence of the *focus of attention* property of the relaxation labelling. After two or three iterations, app. $2N$ (not the theoretical N^2) intersection probabilities are updated. Most of the processing in later iterations revolves around a set of ambiguous intersections (Tab. 1, column 'inters.'). Empirical data (column 'relative change' in Tab. 1) suggest good stability and fast, although not monotonic, convergence.

5 Beyond the Junction Graph

The *junction graph* can be viewed as a final result of the junction finder. The richness of the information represented by the junction graph invites further exploitation. Three methods, *gap bridging, V3-junction detection and polyhedral object face recovery*, are presented in this section.

Gap bridging (see fig. 6 for full description) corrects edge detector failures at T-junctions. The V3-junction detector finds subgraphs of the junction graph that are likely to be projection of a *3-vertex*, a vertex where 3 visible 3D-edges meet. In Section 2 we made an assumption that transversal positions of lines are error-free. This implies that all 3 (or, more generally n) lines terminating at a common 3(or n)-vertex should intersect at a single point. In practise, the line intersections are clustered in a small region. The implementation is straightforward; for every V-junction: check for V-junctions in a small (default: 1 pixel) neighbourhood. The size of the neighbourhood makes false positives virtually impossible. Examples of V3-junctions can be found in figs. 8 and 1.

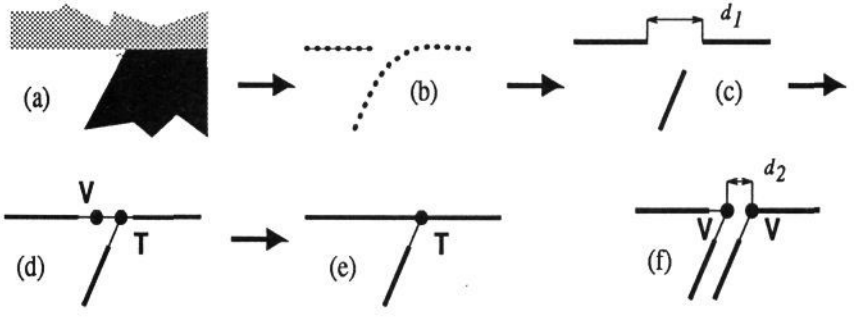
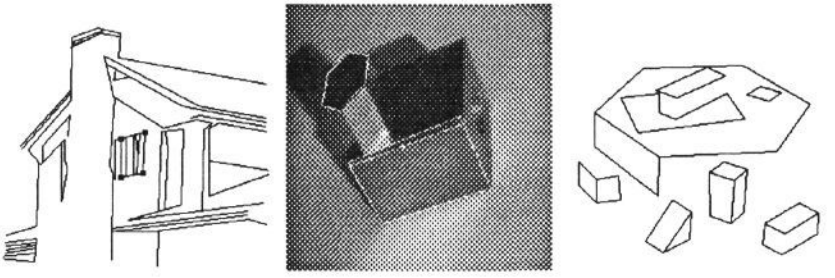


Figure 6: Gap bridging. Conventional edge detectors perform poorly (subfig. (b)) in areas where 3 regions (subfig. (a)) meet (see [11, 8]). This causes a projection of one 3D-edge to be broken into two line segments (c). Spatial arrangement (c) of line segments give rise to subgraph (d) of the junction graph. Situation (d) can be interpreted as either a view of a vertex from a highly improbable, accidental viewpoint or as a manifestation of the above mentioned edge detector problem. The latter interpretation is assumed (the former having a negligible probability) and the subgraph is transformed into the form depicted in subfig. (e). The gap bridging process is *context dependent*; gap of subfig. (f) of length d_2, although significantly smaller then d_1, is left intact. Examples of the gap bridging can be found in the HOUSE (see e.g. roof), WIDGET (upper vertical edge of the from face) and CUBE images (Figs. 8, 1)

Recovery of polyhedral object faces is more complex. If 2D line segments are projections of 3D-edges and junctions projections of vertices then a completely visible face of a polyhedral object must project into a closed loop of V-junctions in the junction graph. Fig. 7 illustrates results of the *face recovery* procedure. Application of *face recovery* for fast 3D pose estimations is described in [7].



(a) (b) (c)

Figure 7: Results of *polyhedral face recovery*. Closed loops in the *junction graph* are assumed to correspond to 3D-edges and vertices of a polyhedral object face. Only one incorrect face is recovered in the TOYS scene (the non-convex polygon inside the large hexagon of Fig. (c); a careful scrutiny of (c) and Fig. 8(f) reveals an accidental alignment of the block and pad leading to junction graph distortion). Fig. (a) shows *corrected lines* - lines stretched to terminate at a junction (compare with Fig. 8(b)). The single face recovered in the HOUSE scene is a projection of the window frame (corners marked by □).

6 Conclusion

We have presented an algorithm for junction detection with two new features: exploitation of spatial context and use of an explicit line detector model. The combination of contextual evidence is not based on an ad hoc method; a well established method, *probabilistic relaxation*, is employed to accomplish the task. The tests performed to validate the line detector model could prove valuable in its own right as an evaluation tool for line detectors.

The junction finder performance has been tested on hundreds of images, mostly running as a part of a continuously operating vision system [7]. Four scenes, HOUSE, TOYS, WIDGET, and CUBE, were selected as representative of different environments (indoor, outdoor, industrial). Results (Fig. 8) show that our main objective has been achieved - vast majority of V and T junctions indeed correspond to vertices and occlusions in the 3D world. Success of the polyhedral face recovery strongly supports this claim. The junction finder possesses two key features vital for continuous operation - it is fast and it doesn't require any user-defined thresholds even in changing conditions. Experiments have shown [7] that the junction finder output provides salient intermediate level features for model invocation and 3D pose estimation.

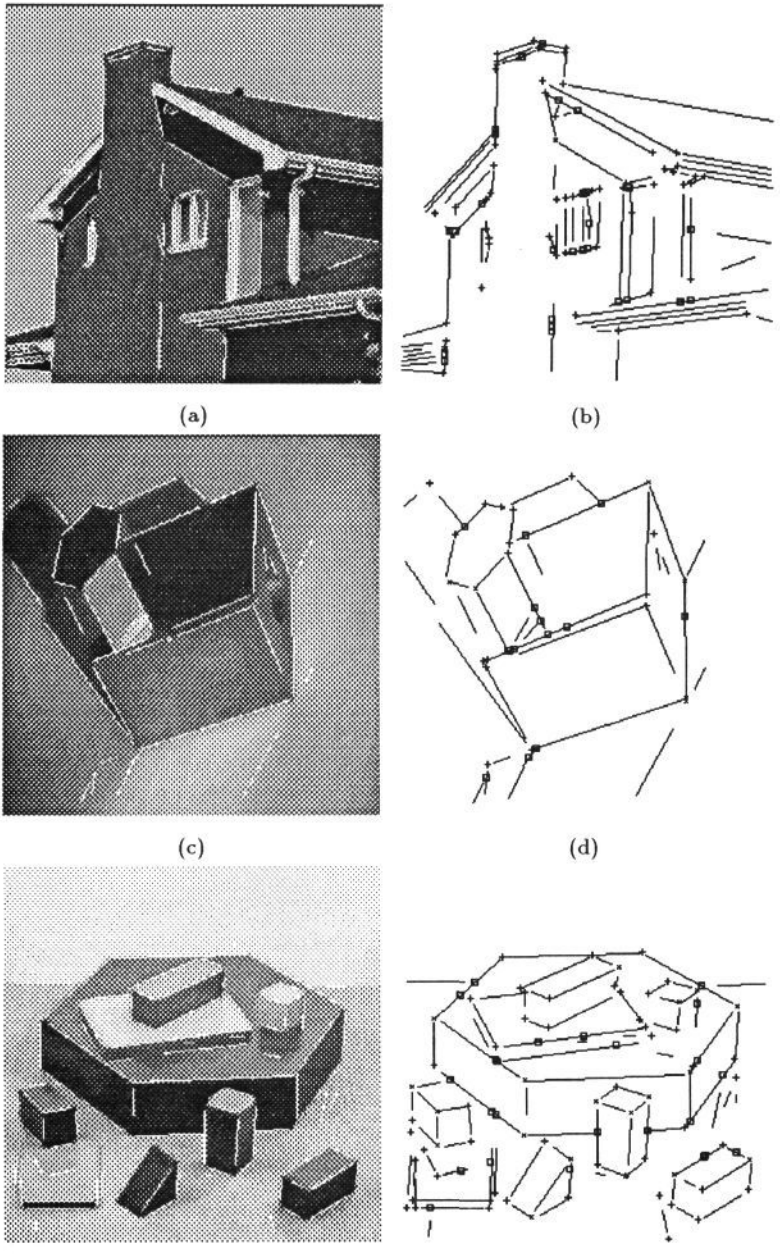


Figure 8: HOUSE, WIDGET and TOYS scenes. Figures (a), (b) and (c) show lines detected by Hough transform superimposed over the original image . Figures (b), (c) and (d) depict lines corrected by contextual gap filling (described in section 6) and the results of the junction finder. *T-junctions* (indicating an occlusion) are marked by □; *V-junctions* (indicating a vertex of two 3D edges) by + and *V3-junctions* (indicating a vertex of three 3D-edges) by ×.

References

[1] R. Bergevin and M. D. Levine. Extraction of line drawing features for object recognition. In *Proceedings of IEEE International Conference Pattern Recognition*, pages 496–501, 1990.

[2] C. Coelho, M. Straforini, and M. Campani. Using geometrical rules and a priori knowledge for the understanding of indoor scenes. In *Proc. British Machine Vision Conference*, pages 229–234, 1990.

[3] R. Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1(2):167, 1987.

[4] R.N. Haber and M. Hershenson. *The Psychology of Visual Perception.* Holt, Rinehart and Winston Inc., U.S.A, 1973.

[5] R. Horaud, F. Veillon, and T. Skordas. Finding geometric and relational structures in an image. In *European Conference on Computer Vision*, pages 373–384, 1990.

[6] J. Kittler and E.R. Hancock. Combining evidence in probabilistic relaxation. *International Journal of Pattern Recognition and Artificial Intelligence*, 3:29–51, 1989.

[7] J. Kittler, J. Illingworth, J. Matas, P. Remagnino, K. C. Wong, H. Christensen, J-O. Eklundh, G. Olofsonn, and M. Li. Symbolic scene interpretation and control of perception. Technical report, ESPRIT BRA Project 3038, March 1992.

[8] Du Li, G.D. Sullivan, and K.D. Baker. Edge detection at junctions. In *AVC*, pages 121–125, 1989.

[9] D. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artifical Intelligence*, 31:355–395, 1987.

[10] R. Mohan and R. Nevatia. Using perceptual organisation to extract 3-d structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1121–1139, 1989.

[11] J. A. Nobel. Finding corners. In *AVC*, pages 267–274, 1988.

[12] P. L. Palmer, J. Kittler, and M. Petrou. A Hough transform algorithm with a 2D hypothesis testing kernel. In *Proceedings of IEEE International Conference Pattern Recognition*, September 1992.

[13] J.R Pomerantz. Perceptual organization and information processing. In *Perceptual Organization*, pages 141–180, 365 Broadway, Hillsdale, New Jersey, 1981. Lawrence ERLBRAUM associates.

[14] J. Princen. *Hough Transform Methods for Curve Detection and Parameter Estimation.* PhD thesis, University of Surrey, June 1990.

[15] VAP project. Vision as process. Technical annex, EEC, March 1989.

[16] R. Sedgewick. *Algorithms.* Addison-Wesley, Reading, 1983.