Improved Contour-Based Corner Detection for Architectural Floor Plans

Max Feltes[1], Sheraz Ahmed[1,2(✉)], Andreas Dengel[1,2], and Marcus Liwicki[2,3]

[1] University of Kaiserslautern, Kaiserslautern, Germany
[2] German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
{max.feltes,sheraz.ahmed,andreas.dengel}@dfki.de
[3] University of Fribourg, Fribourg, Switzerland
marcus.liwicki@unifr.ch

Abstract. A new rotation invariant corner detection method for architectural line drawing images is proposed in this paper. The proposed method is capable of finding corners of objects in line drawing images by filtering out unnecessary points without changing the overall structure. Especially, in case of diagonal lines and corners, our method is capable of removing repetitive points. The proposed method is applied to corner detection of walls in floor plans which in turn are used for detection of wall edges. To evaluate the effectiveness of detected corners, gap closing and wall edge detection is performed on a publicly available dataset of 90 floor plans, where we achieved a recognition and detection accuracy of 95 %.

1 Introduction

Even in our modern world it is very hard to find architectural floor plans fulfilling specific criteria. Most of the time, after a client specifies how he imagines his new home, the architect will go through his archive to find similar floor plans matching these criteria. As a next step, he will modify them to fulfill further constraints. However, this manual search takes a long time, and even though it might have a high precision rate, the recall rate is very low. In order to be able to automate the search, the archive has to be scanned and automatically analyzed.

Automated floor plan analysis is the task of extracting information about a building's structure that is embedded inside an image. It is composed of several subtasks, such as, segmenting the text and the graphics from the document, detecting the walls and doors, and finally recognizing the different rooms. Automated floor plan analysis is an ongoing topic of research in pattern recognition and machine learning. Several attempts with varying goals have been made to solve this problem: [1–3] try to reconstruct a 3D model from the 2D floor plans, whereas [4] tries to extract the rooms and their connections. References [5,6] focus on the understanding of hand-drawn and sketched floor plans.

Recently, we have introduced a method for automatic floor plan analysis [7]. An analysis of the results in [7] lead to the conclusion that the room retrieval

B. Lamiroy and J.-M. Ogier (Eds.): GREC 2013, LNCS 8746, pp. 191–203, 2014.
DOI: 10.1007/978-3-662-44854-0_15

works quite good on rooms with walls going horizontally and vertically in the plan but fails on floor plans with diagonal walls. The main reason for these failures were problems while finding the borders of the rooms. The algorithm of [7] closes the gaps occurring at doors and windows and these gaps were not correctly found.

It is to be noted that corner/feature detectors like SIFT [8], FAST [9], etc. cannot be used in context of line drawing images. It is because they are based on blob detection, and the points where a blob is detected is considered as a corner/feature/key point. However, in case of line drawings the goal is to detect corners, which can be used to approximate objects in the image, with high precision and without excessive points.

In this paper, a novel method for corner detection in line drawing images is presented. This method is based on the algorithm introduced in [10] and improves it at different points. The proposed method solves the problem of over-segmentation, especially on diagonal lines. To show the impact of proposed method, detected corners are used for detection of wall edges and gap closing in architectural floor plans as done in [7]. Note, while the method of [10] is already quite old, it is considered as the standard method and being used in different toolkits, e.g. in OpenCV this is a standard method for contour extraction.

The rest of the paper is organized as follows. Section 2 summarizes different methods available for corner detection which can be used in architectural floor plans. Section 3 provides the insight about proposed corner detection method. Section 4 provides an application of detected corners in floor plan analysis. Experimental details and analysis of results are presented in Sect. 5. Finally, Sect. 6 concludes the paper with possible future directions.

2 Related Work

In literature mostly corner detection, vectorization, and key point detection are used alternatively. However, in context of line drawing images corner detection and key point detection are different. This section summarizes different approaches for corner detection/vectorization, which can be used in context of architectural floor plans. An overview of typical selection process of vectorization methods is given in [11].

A corner detection method based on self-similarity is presented in [12]. A pixel is referred to as corner, if similarity between the patch centered at the pixel and neighboring patches, is low. Similarity is computed using sum of square differences between patches. In [13] Harris et al. further improved the method presented in [12] by incorporating directionality into the similarity score. To refer to a pixel as corner it looks for significant changes in all directions.

An approach for contour detection is presented in [10]. It can be used as corner detector/vectorization method, because it simplifies contours points by approximating them as a polygon. These simplified points serve as detected corners. It is based on a simple border following algorithm, with the option to either detect only outer contours or to include inner borders.

In [14], an algorithm is proposed to approximate a digital line by recursively including points based on a distance measure. It finds the point farthest

away from the approximated line, and includes it if the distance exceeds a given threshold. The segments formed by this are then recursively approximated. These approximated points are referred to as detected corners.

In [15] a method is presented to detect dominant/corner points on closed digital curves. It is a parameter free approach that first determines the region of support for each point based on its local properties. Using these regions relative significance (e.g. curvature) of each point is computed. To finally detect dominant/corner points non maximum suppression is applied.

A method for segmentation of edges into lines and arcs is presented in [16]. The main idea is to extract corners based on edges of object. It uses a recursive algorithm that analyzes lists of connected edge points and convert them into polygons. These polygonal descriptions are analyzed to groups of connected lines. Finally circular arcs and lines are obtained as image description, which are representing the corners of objects in the image.

Reference [17] approached the task of edge detection by finding points in the binary image and combining several vectors to form more complex forms. As this approach only return bars and poly-lines, it could lead to problems with some walls in floor plan analysis, as some walls are curved, e.g., corner towers. Similarly, [18] proposed an approach for corner detection based on the Chord-to-Point distance accumulation.

Another corner detection/vectorization approach was introduced by [19], which works on skeletonized shapes. The drawback is that the thickness of the lines are only approximated. This thickness however is used for the gap closing algorithm and should reflect the thickness of the line at the extremities. Reference [20] extracts local interest points as junctions by creating a skeleton connective graph and using a wavelet transform. Similarly, [21] proposed a contour based corner detector method. It is based on magnitude of imaginary part of Gabor filters response on contours.

All of the above-mentioned methods try to approximate objects in given image. However, there is another class of corner/feature detector where the goal is not to approximate the object but to locate important points (referred to as key points) in the image. For example, LoG [22], DoG [23], SIFT [8], SURF [9], FAST [24], BRISK [25], SUSAN [26], etc. These methods try to locate blobs using different masks and other information. All the points where blob is detected are referred as keypoint. This class of corner/key point detection is not suitable for line drawing images, where goal is to approximate objects with precision and as less points as possible.

3 Proposed Corner Detection

Most of the systems for analysis of line drawings/technical drawings/architectural floor plans are based on vectorization. The vectorization results are considered as corners of objects in these images. These corners are further processed to extract different structural information in these drawing images. If the corner detection/vectorization has errors, these errors are propagated to the next steps

(a) Detected points on a diagonal wall (b) Detected points near an indentation

Fig. 1. Two cases which lead to over-segmentation using [10]

in the analysis, as next steps are based on processing these points. In order to explain our method in terms of real application, we applied it on architectural floor plans and compared our results with the method used in [7].

In [7] corners for wall image in architectural floor plans are detected using the method in [10]. However, using the method in [10] for corner detection from wall image inherits different drawbacks. As shown in Fig. 1a, the method in [10] worked correctly only for perfectly horizontal and vertical walls. However, if the walls were diagonal, many excess points would be detected along those walls, thus splitting a long edge on wall into several smaller ones and leading to over-segmentation. This over-segmentation will lead to errors in next steps, where each wall edge is processed for closing the gaps on the probable locations of doors and windows.

A second case which could lead to over-segmentation is when the walls don't have clean lines, but have some noise added to the edges. The most notable causes were binarization and removal of doors/windows, where small indentations were left on some of the wall segments, as shown in Fig. 1b.

The proposed corner detection improves the method in [10] by filtering out the points that appeared through over-segmentation on diagonal lines. To filter every point, it calculates the distance to the line connecting the previous and the next point. This distance indicates whether the point was necessary or if it appeared because of an over-segmentation of a horizontal edge.

The equation of the line passing through $P_1(x_1, y_1)$ and $P_3(x_3, y_3)$ has the general form:

$$l{\equiv}a * x + b * y + c = 0 \tag{1}$$

The distance of a given point $P(x, y)$ to l is defined as:

$$distance(P, l) = \frac{a * x + b * y + c}{\sqrt{a^2 + b^2}} \tag{2}$$

Points can now be filtered out using the following formula:

$$discard(P_2, l) = \begin{cases} True & \text{if } distance(P_2, l) < \Theta_d \\ False & \text{else} \end{cases} \text{ with } \Theta_d = theshold_{distance} \tag{3}$$

The proposed method needs as an input only the ordered list of detected points, as well as the threshold Θ_d mentioned above. The order list of corner points is created by traversing and arranging all the detected points in clock wise direction.

Algorithm 1. filter_cornerpoints

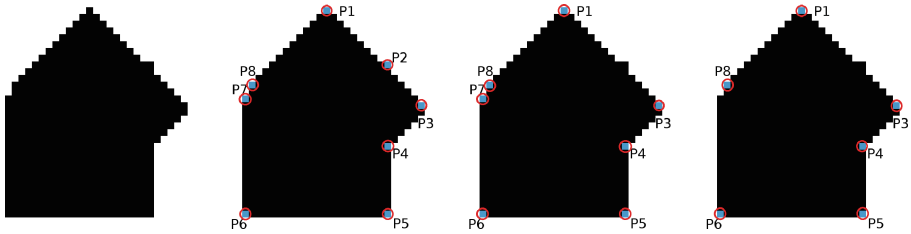
Input: A threshold *threshold*
An ordered list containing the detected points *Ordered_points_list*
Output: A filtered ordered list of points

```
point_list ← Ordered_point_list
i ← 0
while i + 3 < len(point_list) do
  P1 ← point_list[i]
  P2 ← point_list[i + 1]
  P3 ← point_list[i + 2]
  line ← get_line(P1, P3)
  d ← distance(P52, line)
  if d < threshold then
    del point_list[i + 1]
  else
    i ← i + 1
  end if
end while
return point_list
```

In Fig. 3 there is an over-segmentation of the two diagonal lines because of the detected points P_2 and P_7/P_8. The proposed method will consecutively check the detected points if they meet the distance threshold (Fig. 2).

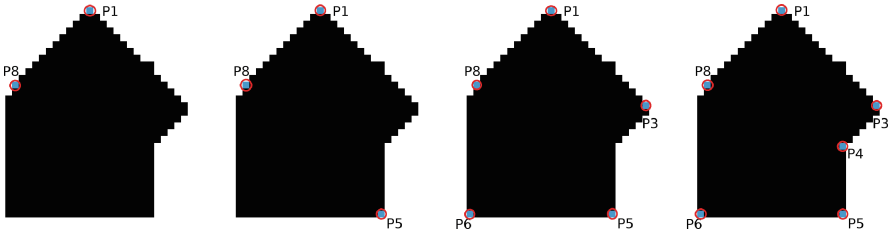
It is important to note that if consecutive points are close together, only the last one will be kept. In rare cases however, each point is needed to retain the shape of the wall, most notably in corner towers. This can be solved by not only considering the three consecutive points, but increasing the index of the right delimiter as long as all the points between the two corners defining the line lie within the defined distance threshold. However, as these are very rare, the very small increase in the detection rate did not outweight the decrease in recognition accuracy of the gap closing evaluated in Sect. 4. In the example in Fig. 3, it is not clear which point should be kept (P_7 or P_8). As these are very rare, it is acceptable to lose some information in this case. In another rare case, a small indentations indicates that a door closes as this position. This information is unfortunately lost (see Fig. 4).

Another important point to note is that this proposed method is different from [14] in the way that it is more robust against over-segmentation by calculating the distances of the points to different segments than the method proposed by [14]. This can lead to different results, as seen in Fig. 5. Figure 7 shows an



(a) input wall segment (b) Detected corner points using [10] (c) step 1 discards P_2 (d) step 6 discards P_7

Fig. 2. Different stages of the proposed method with $\Theta_d = 1.5$



(a) step 1 selects the first and the last corners as starting conditions (b) step 2 selects the furthest point to the segment, P_5 (c) step 3 recursively reestimates both segments until $distance < \Theta_d$ for every corner (d) step 4 includes P_4

Fig. 3. Different stages of [14]

undersegmentation that happens when applying [14]. Our proposed approach works locally, meaning it only needs to calculate the distance of a single corner, whereas [14] needs to compute the distances of every corner on the segment.

The detected corner points can be used for different purposes. In [7] these detected corner points are used for extraction of parallel wall edges, which in turn are used to close the gaps on the probable location of door and windows. To show the effectiveness of our improved corner detection method, we have applied our and several different corner detection methods on architectural floor plans and evaluated the gap closing (See Sect. 4).

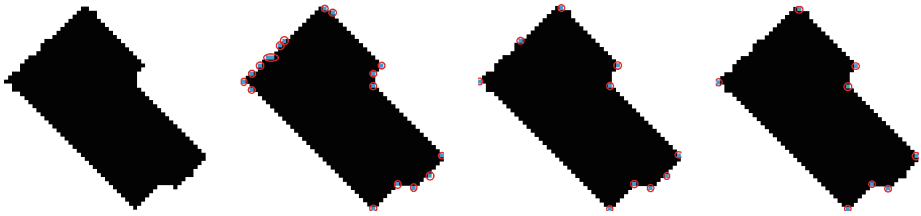
4 Gap Closing: An Application of Corner Detection

Gap closing is a process in architectural floor plan analysis which is performed to find the closed regions of rooms in the floor plan. It is performed on the edges which correspond to the parallel walls, as these are the probable locations of door and/or windows. Wall edges are constructed using detected corner points (for more details on wall edges see [7]). Therefore, if there is an error in corner point

(a) Detected corner points using method in [10]

(b) Detected corner points after filtering

Fig. 4. Failure case: a small indentation is lost where a door might possibly connect



(a) Input wall image

(b) Extracted corners using [10]

(c) Detected corner points using [14] with $\Theta_d = 1.5$

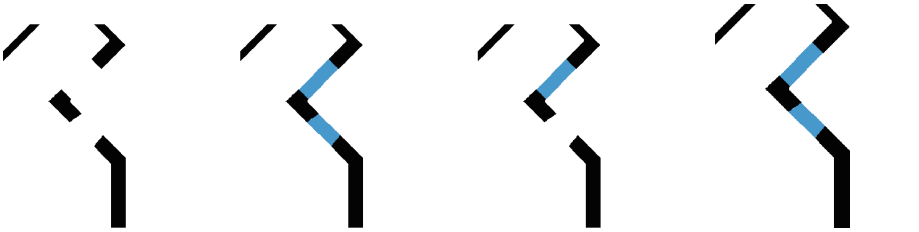
(d) Detected corner points using the proposed method with $\Theta_d = 1.5$

Fig. 5. Comparison of the proposed method and [14], both with $\Theta_d = 1.5$ Note that [14] was not able to filter 2 noisy points on diagonal lines.

detection (over/under segmentation) it would propagate to wall edge detection and then to gap closing.

Here gap closing is presented to show that our method has improved detection and recognition accuracy remarkably by resolving errors in corner point detection in [10] which was used in [7]. Gaps are closed by connecting pairs of previously detected edges, which fulfilled several conditions:

- The angles of the rectangle created by connecting the two edges should be $\sim$ 90°. This ensures that the edges are aligned.
- The area between the two edge candidates should be empty. This ensures that two edges will not be connected if they are separated by another wall.
- $length_{edge1} <= 2 * length_{edge2}$. This ensures that edges are only connected if they have approximately the same length (as is almost always the case in architectural floor plans).



(a) Input for gap closing

(b) Expected result for gap closing

(c) Gap closing with corner points by [10] and [14]

(d) Gap closing with corner points by proposed approach

Fig. 6. Gap closing

- $length_{edge} > threshold_1$. This removes some edges which exist because of noise.
- $length_{edge} < threshold_2$. This removes connections that would be formed by connecting the walls of a hallway for example.

Figure 6 clearly shows that the gap closing using the detected corner points by proposed method solves the over-segmentation of the diagonal lines, and thus introduces a rotation invariance to [10]. It is important to note that if even one of two corresponding edges is over-segmented, the gap closing will not work. It is therefore crucial that the proposed method solves this problem reliably.

5 Experiments

5.1 Evaluation Method

The evaluation method introduced by [27] is used to evaluate the proposed methods. It is able to evaluate exact matches as well as partial matches.

The detected regions are compared to the ground-truth regions by calculating the overlap between them and determining several parameters:

one_to_one	is the number of detected regions which overlap with exactly one ground-truth region
g_one_to_many	is the number of ground-truth regions that overlap with more than one detected region
g_many_to_one	is the number of detected regions where more than one detected region overlaps with a single ground-truth region
d_one_to_many	is the number of detected regions that overlap with more than one ground-truth region
d_many_to_one	is the number of ground-truth region where more than one ground-truth region overlaps with a single detected region

To determine these parameters, the overlap between each pair of detected regions and ground-truth regions is detected using the following formula:

Let d[i] be the i^{th} detected region, g[j] be the j^{th} ground-truth region:

$$match\_score(i,j) = \frac{area(d[i] \bigcap g[j])}{max(area(d[i]), area(g[j]))} \tag{4}$$

If $match\_score(i,j) > acceptance\_threshold$, the overlap is kept. By building a table containing every possible combination of detected regions and ground-truth regions (called *match score table*), the aforementioned parameters can easily be deduced. These parameters are then used to calculate the detection rate and recognition accuracy of the evaluated method:

$$DetectionRate = \frac{one\_to\_one}{N} + \frac{g\_one\_to\_many}{N} + \frac{g\_many\_to\_one}{N} \tag{5}$$

$$RecognitionAccuracy = \frac{one\_to\_one}{M} + \frac{d\_one\_to\_many}{M} + \frac{d\_many\_to\_one}{M} \tag{6}$$

with N = number of ground-truth regions
M = number of detected regions

5.2 Results

Gap closing has been evaluated for the original system presented by [7]. It is compared to the gap closing using the corner point detected in this paper. The results for gap closing with the different types of corner point detection are summarized in Table 1.

The data set consists of 90 architectural floor plans[1], which are rescaled to a smaller size. This ensures that all the walls have the same thickness, and the same thresholds can be used on all images for edges that are to small/big. If the images had not been resized to the same size, a different threshold would possibly have to be chosen for different sizes of the images. An example of the floor plans can be seen in Fig. 7.

5.3 Performance Analysis

The proposed algorithm needs to calculate the distance from every point to the line connecting its neighbouring points exactly once. To compute this, it needs to first compute the parameters a, b, c of the line, and subsequently the distance. It runs in $O(n)$.

In comparison, the method proposed by [14] needs the same computation steps to calculate the distance. However, as it needs to calculate all distances of the given segment in every iteration, it runs in $O(n\ log(n))$, with a worst case scenario of $O(n^2)$. The algorithm proposed in this paper works more efficiently, while still conserving the original shape of the contour.

[1] The actual image size is $2479 * 3508$. For making the analysis process more efficient, isotropic down scaling to $1413 * 2000$ has been applied.

Table 1. Results for gap closing

	Detection rate	Recognition accuracy
Original system	54.5 %	51.27 %
[15] with L curvature	95.88 %	92.46 %
[15] with K curvature	95.77 %	92.34 %
[14] with $\Theta_d = 1.5$	93.39 %	93.46 %
Proposed method ($\Theta_d = 1.5$)	94.99 %	94.40 %
Proposed method with multiple corners ($\Theta_d = 1.5$)	95.01 %	94.14 %

5.4 Analysis

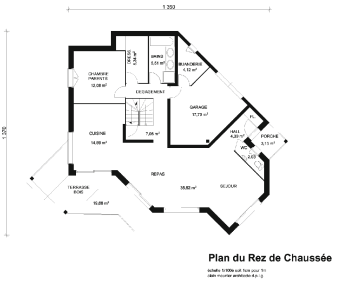
First, it should be mentioned that the threshold has been determined manually. The idea behind the value of this threshold is that over-segmentation happens because of binarization on not perfectly horizontally aligned lines. Therefore, our proposed value is small enough to correct small errors that lead to the over-segmentation, but not too high that it would lead to cutting of actual corners.

The gap closing criteria using the proposed corner points clearly performed very well. Unclosed gaps are mostly edge cases, where gaps have to be closed on curved walls (i.e. corner towers) or the walls don't align (i.e. windows or doors on corners). The proposed corner point detection improved mostly diagonal edges, even if noise is added due to binarization.

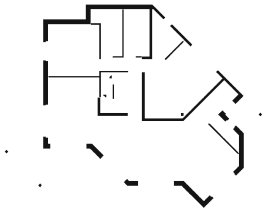
The wall edge detection using the corner points detected by the proposed approach performed very well on the evaluation data. It was able to solve the over-segmentation problem of the previous method. Also, this method of edge detection is rotation invariant, which is a big advantage for analysing scanned floor plans.

It is important to note that the data set consists of floor plans that are all perfectly aligned with the frame of the image. Thus, only diagonal walls really profit from the improvements of the edge detection. If the data set had featured scanned floor plans, the difference would have been a lot more noticeable, as the horizontal and vertical lines would have been subject to the over-segmentation problem mentioned earlier.

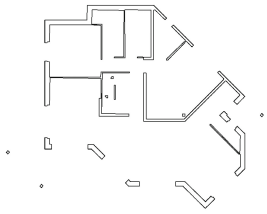
The disadvantages of the proposed method are that fine details on the walls (i.e. small indentations) can be lost during the filtering process. This effect can be seen in Fig. 4. A second noteworthy point is that if wrong corners have been detected close to a real corner, it is possible that the real corner will be filtered out, and one of the other wrong corners will be kept. This can lead to the effect that the corner appear to drift a small amount from the expected position. However, as this method is used as a preprocessing step to perform gap closing to delimit the rooms from outside, both of these drawbacks have little or no impact on the final performance due to the use of information from other steps, e.g. door detection.



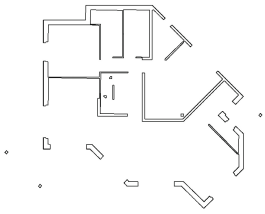
(a) Floor plan from the data set



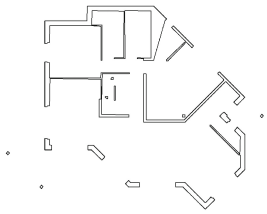
(b) Extracted walls



(c) Walls redrawn after extracting the corners with[15] (n.b. the walls are heavily over-segemented)



(d) Walls redrawn after filtering the corners with the proposed method (the walls are no longer over-segmented)



(e) Walls redrawn after filtering the corners with [14]

Fig. 7. A floorplan from the dataset

6 Conclusion and Future Work

In this paper, an improved corner point detection method based on border following algorithm is presented by improving the method in [10]. This method rendered the algorithm rotation invariant, as the diagonal edges are no longer over-segmented into smaller ones.

Afterwards, several criteria for gap closing based on the distance measure are used in order to close the outer walls of a building. Although they seem simple

and straight-forward, they achieved a very high detection rate, as well as an equally high recognition accuracy.

At present, the thresholds used during the evaluation were estimated by trial and error. This achieved good results, as the analysed floor plans are rescaled to a fixed size before the analysis process. To further improve wall edge detection, the distance threshold Θ_d parameter could be dynamically calculated by estimating the thickness of the walls. This would render the process less prone to varying input sizes.

The gap closing could also benefit from dynamically calculated edge length thresholds, as well as the error tolerance that they are granted when deciding whether to connect the edges or not.

References

1. Yang, R., Cai, S., Lu, T., Yang, H.: Automatic analysis and integration of architectural drawings. Int. J. Doc. Anal. Recogn. (IJDAR) **9**(1), 31–47 (2007)
2. Masini, G., Dosch, P.: Reconstruction of the 3d structure of a building from the 2d drawings of its floors. In: Proceedings of the Fifth International Conference on Document Analysis and Recognition, pp. 487–490 (1999)
3. Or, S.H., Wong, K.H., Yu, Y.K., Chang, M.M.Y.: Abstract highly automatic approach to architectural floorplan image understanding & model generation (2005)
4. Valveny, E., Tabbone, S., Macé, S., Locteau, H.: A system to detect rooms in architectural floor plan images. In: Proceedings of the 9th IAPR International Workshop on Document Analysis Systems, DAS '10, pp. 167–174 (2010)
5. Arai, H., Aoki, Y., Shio, A., Odaka, K.: A prototype system for interpreting handsketched floor plans. In: Proceedings of the 13th International Conference on Pattern Recognition, vol. 3, pp. 747–751 (1996)
6. Liwicki, M., Weber, M., Dengel, A.: A sketch-based retrieval for architectural floor plans. In: 12th International Conference on Frontiers of Handwriting Recognition, pp. 289–294 (2010)
7. Ahmed, S., Weber, M., Liwicki, M., Langenhan, C., Dengel, A., Petzold, F.: Automatic analysis and sketch-based retrieval of architectural floor plans. Pattern Recogn. Lett. **35**, 91–100 (2014). (Frontiers in Handwriting Processing)
8. Lowe, D.G.: Object recognition from local scale-invariant features. In: The Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, pp. 1150–1157 (1999)
9. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Comput. Vis. Image Underst. **110**(3), 346–359 (2008)
10. Suzuki, S., Abe, K.: Topological structural analysis of digitized binary images by border following. Comput. Vis. Graph. Image Process. **30**(1), 32–46 (1985)
11. Tombre, K., Ah-Soon, C., Dosch, P., Masini, G., Tabbone, S.: Stable and robust vectorization: how to make the right choices. In: Chhabra, A.K., Dori, D. (eds.) GREC 1999. LNCS, vol. 1941, pp. 3–16. Springer, Heidelberg (2000)
12. Moravec, H.: Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University and doctoral dissertation, Stanford University, number CMU-RI-TR-80-03, September 1980
13. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings of Fourth Alvey Vision Conference, pp. 147–151 (1988)

14. Peuker, T.K., Douglas, D.H.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The Int. J. Geogr. Inf. Geovisualization **10**(2), 113–122 (1973)
15. Teh, C.H., Chin, R.T.: On the detection of dominant points on digital curves. IEEE Trans. Pattern Anal. Mach. Intell. **11**(8), 859–872 (1989)
16. Rosin, P.L., West, G.A.W.: Segmentation of edges into lines and arcs. Image Vis. Comput. **7**(2), 109–114 (1989)
17. Dori, D., Liu, W.: Sparse pixel vectorization: an algorithm and its performance evaluation. IEEE Trans. Pattern Anal. Mach. Intell. **21**(3), 202–215 (1999)
18. Awrangjeb, M., Lu, G.: Robust image corner detection based on the chord-to-point distance accumulation technique. IEEE Trans. Multimedia **10**(6), 1059–1072 (2008)
19. Hilaire, X., Tombre, K.: Robust and accurate vectorization of line drawings. IEEE Trans. Pattern Anal. Mach. Intell. **28**(6), 890–904 (2006)
20. Barrat, S., Ramel, J., Pham, T.-A., Delalandre, M.: A robust approach for local interest point detection in line-drawing images. In: 2012 10th IAPR International Workshop on Document Analysis Systems (DAS), pp. 79–84 (2012)
21. Zhang, W.-C., Wang, F.-P., Zhu, L., Zhou, Z.-F.: Corner detection using gabor filters. IET Image Processing, May 2014
22. Lindeberg, T.: Feature detection with automatic scale selection. Int. J. Comput. Vision **30**(2), 79–116 (1998)
23. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**(2), 91–110 (2004)
24. Rosten, E., Drummond, T.: Fusing points and lines for high performance tracking. In: IEEE International Conference on Computer Vision, vol. 2, pp. 1508–1511, Oct 2005
25. Leutenegger, S., Chli, M., Siegwart, R.: Brisk: binary robust invariant scalable keypoints. In: ICCV, pp. 2548–2555 (2011)
26. Smith, S.M., Michael Brady, J.: Susana new approach to low level image processing. Int. J. Comput. Vision **23**(1), 45–78 (1997)
27. Phillips, I.T., Chhabra, A.K.: Empirical performance evaluation of graphics recognition systems. IEEE Trans. Pattern Anal. Mach. Intell. **21**, 849–870 (1999)