Modeling Automated Inspection Plans for Straight Line Detection in Image Scenes

Edgar Reetz*, Alexander Schlegel, Patrick Werner, Maik Schumann, Jörg Bargenda and Gerhard Linß
*Department of Quality Assurance and Industrial Image Processing
Ilmenau University of Technology, Ilmenau, 98693 / Thuringia, Germany
Email: Edgar.Reetz@tu-ilmenau.de

Abstract

This article explains an approach to further enhance the detection of geometric primitives namely straight lines, based on the spline filter contour decomposition method as shown in [1]. The aim is to split complex features into simple primitives for which further processing steps will lead to automated measurement results. Thus the inspection plan generated is reusable without employing additional CAD software tools. The feature recognition approach proposed in this paper incorporates an adaptive spline filter scale space method for smoothing contour point clouds at subpixel-scale precision with the aim to reduce the user influence on the setup of ROIs (regions of interest) in image scenes and therefore increasing usability of software interfaces.

Keywords: inspection plans; feature extraction; geometric primitives; contour decomposition; straight line detection;

I. Introduction

Using searchline based algorithms in software for high-precision measurement tasks at subpixel-scale usually requires the tester to manually set up the inspection plan by positioning measuring fields (regions of interest with searchline setup) in the image scene. Approaches using CAD data allow the use of algorithms that derive measurement tasks to be executed directly from the design data. In case the design data is not available, the inspection plan has to be set up manually by the tester according to engineering drawings or vague context information. The method presented here focuses on image data, acquired with an optical measurement device. Since the input data for the algorithms used consists of plain point clouds, this approach is not limited to image data but could be implemented as well for touch probe measurement devices. The common procedure to set up the inspection plan, as could be seen in fig. 1, displays the sensor data to the user on a screen. Based on the image displayed and the idea of what kind of feature needs to be measured, the user starts a selection process for every single feature required. That includes selecting the type of feature (e.g. straight line), positioning the region of interest in the image scene where the starting and ending point of the region needs to be set up manually by mouse click interactions, followed by defining the length and the orientation of the searchlines. Before the measurement task can be executed (e.g. measuring distance, angle, radius, etc.) the two dimensional input data needs to be extracted from the grayscale image acquired. For the regions of interest a combination of edge detection algorithm and subpixeling will return point clouds for each one of them. When all simple features are set, the user continues with selecting the features of interest for which the measurement task previously selected will return the measurement results of the linked features (e.g. the angle between two line segments). When all linked features are measured, the inspection plan is fully processed and the results will be displayed to the user. Improving objectivity and accuracy of measurement results requires reduction of user influence while increasing the level of automation. The first step in automating inspection plans is to reduce the user influence during contour decomposition tasks. To analyze the image scene, the contour needs to be split into shape primitives like line segments and arcs, so that measuring fields can be positioned without user interaction. Therefore the edge detection and subpixeling can be autonomously invoked and return point clouds for which the compensation element will be instantly calculated. Employing an increased level of automation for the user interaction phase during the measurement process becomes beneficial not only concerning time (speed up), when considering the workload scaling at least proportional with the required number of regions of interest to be set up for the inspection plan, but also in accuracy when taking the correct positioning of the ROIs into account. Therefore the need for less user interaction enables the software design to provide greater usability and facilitates the software to be used for education and teaching purposes. Inexperienced users will be enabled to use software tools for high precision measurement tasks, without the requirements for detailed knowledge concerning subpixeling methods. Another field of application could be the support for reverse engineering from real world measured data to parameterized CAD models. For the problem shown here, the image preprocessing steps were already completed, therefore the input data for this approach is based on plain point clouds, extracted at subpixel-scaled precision using common interpolation

methods as shown in [2]. Figure 2 displays the extracted point cloud of an image showing the contour of a milling tool, containing 1457 contour points.

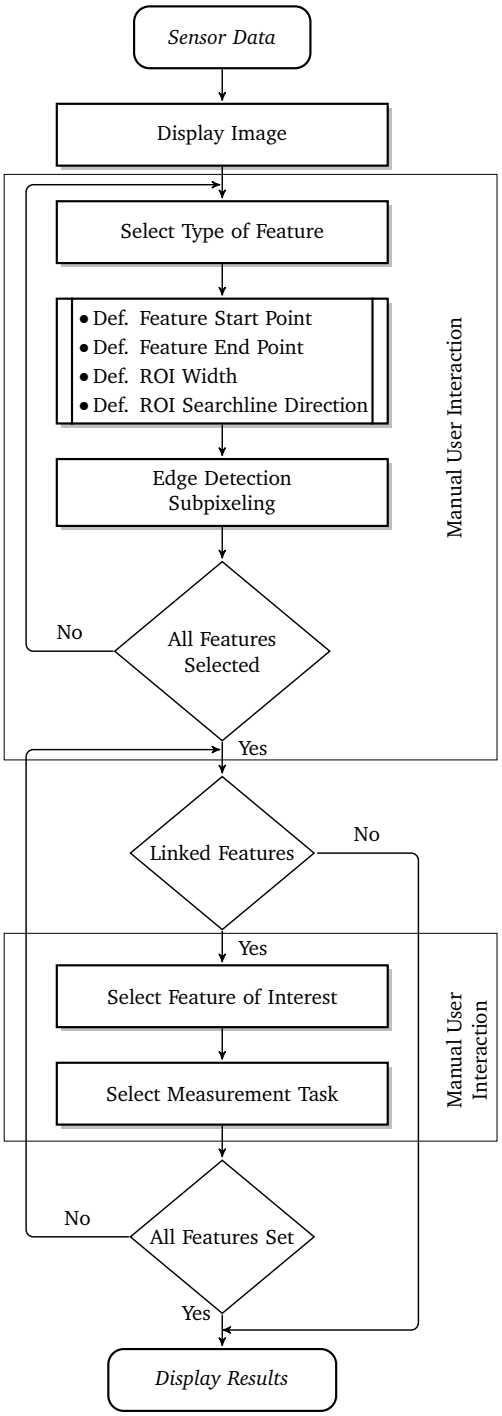

Fig. 1. Manual user interaction during the measuring process using optical measurement devices that operate at subpixel-scale precision employing searchline based algorithms.

II. State Of The Art

Using local contour curvature information for contour decomposition problems is a common approach in computer vision. The segmentation of the contour point cloud

Fig. 2. Point cloud representing object contour at subpixel-scaled precision, containing 1457 contour points.

shown in fig. 2 is based on curvature information only. The contour curvature criterion is used to discriminate between line segments and arcs:

$$\kappa(t) \approx const. \wedge \kappa(t) \approx 0 \quad \text{for line segments} \tag{1}$$
$$\kappa(t) \approx const. \wedge \kappa(t) \not\approx 0 \quad \text{for arcs} \tag{2}$$

For the continuous 2D space, the principle contour curvature is calculated using equ. (3).

$$\kappa(t) = \frac{d\phi(t)}{dt}. \tag{3}$$

For the discrete contour points t, the local contour curvature $\kappa(t)$ is calculated using equ. (4) where ϕ_t utilizes the $\mathrm{atan2}$-function which generalizes the $\arctan(\frac{y}{x})$-approach:

$$\kappa(t) = \phi_{t+1} - \phi_{t-1}. \tag{4}$$

Using subpixeling methods as described in [2] usually results in a noisy signal for the contour curvature data caused by interpolated spatial positions of the contour points. The corresponding contour curvature $\kappa(t)$ for the contour shown in fig. 2 is displayed in fig. 3. For most object recognition tasks and shape analysis problems pixel-scaled precision is sufficient. To further condition the curve, methods like $\frac{\pi}{4}$-clockwise smoothing based on the pixelgrid, could not be applied due to the non-equidistantly positioned contour points caused by subpixel interpolation methods.

Treshold Methods

Contour decomposition using a priori set threshold values for feature point detection at higher curvature amplitudes (corner points) is not only time consuming in tweaking the right parameter, but will result in loss of important feature points or many false positives, according to the profile's roughness. Consequently the contour is split into

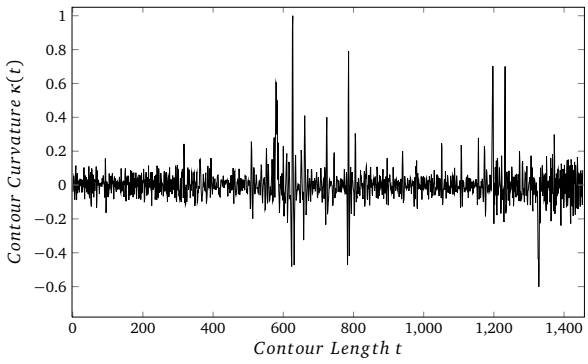

Fig. 3. Computed contour curvature $\kappa(t)$ for the object contour shown in fig. 1

many separated segments which have to be joined by extensive postprocessing operations [3]. Even applying evolutionary algorithms will not necessarily lead to an optimal parameter setup. The genetic algorithm applied is trained using ground truth data, which resembles a specific combination of profile roughness and waviness. The critical component is the low-pass filter setup (see section III and [1]) – if the contour profile to be segmented is of significantly different roughness and waviness, the parameter setup trained over a long time will not solve the decomposition task.

Hough Transform

The transform method proposed by Hough [4] suffices pixel-scaled precision for image data using a model based approach to describe shape (e.g. lines, circles, ellipses). For each pixel that supports a model, a counter in parameter space will be incremented. When applied to subpixel-scale, the computational costs of the method would be tremendous. The Hough method itself does not localize. To get the position of each element, a windowed contour tracking followed by a split and merge algorithm will return the contour elements. The major disadvantage is the trade-off between precision and computational costs when setting the parameter for the transformation.

Polygonal Fitting

Approximating the contour data by simple 2D polygons usually requires a priori set threshold values. The basic idea is to approximate a complex shaped contour by shape primitives like line segments. The method is a successively refining split and merge approach, where the first and the last vertex of the contour will be connected by a line segment. The point with the maximum distance to the approximating element is the new breakpoint. Consequently, from the starting vertex to the breakpoint and from the breakpoint to the end vertex two new approximations will be set. The previous steps will be continued until the contour is completely split into approximation elements according to termination criteria, e.g. a minimal length of approximating line segments. Thus, traversing the generated tree structure for the object's contour enables the use of measurements for the quality of fitting to the contour segment. Using such significance measurements, as proposed by Lowe [5], West/Rosin [6] and Faber [7] could improve the overall quality of the shape approximation, especially when comparing the significance value for each type of shape primitive used for the approximation (e.g. lines, circles, ellipses). On the other hand this method depends on initially set starting points and breakpoints. As a consequence this could cause deletion of important feature points as shown in [6]. Therefore they were not applied at initial contour decomposition stages.

III. The Problem Setting

Detecting corner points means finding contour points that provide a local peak in curvature. Therefore a low-pass filter is set up, to discriminate the contour curvature of the underlying shape from noise, caused by spatial interpolation due to the subpixeling. Smoothing the contour curvature data using methods like triangular window averaging or methods using local polynomial regression like Savitzky-Golay filters (see [8] and [9]) always requires correct estimation of the smoothing window length (see [1]). While the Savitzky-Golay filter preserves higher moments of the underlying data, it will return fairly smooth data, when used within a filter cascade (in combination with triangular filters [1]). The problem with the filter setup still remains. Using evolutionary algorithms for training contour peak point detection depends on representative image material. The feature extraction will fail if the contour to be processed has higher roughness due to the highly fragmented initial segmentation results. In That case postprocessing steps are required to join the split segments into continuous contour segments for further processing or measurement tasks. The method to be applied needs to be more suitable due to the different quality of input data.

IV. The Novel Approach

Corner points break the object contour into less complex segments. Therefore the following section will present an approach to obtain less fragmented segmentation results, taking corner points into consideration. The aim is to split the contour into a suitable set of shape primitives from where on further processing steps could be implemented. To distinguish feature points from the set of contour points P:

$$P = \{p_i\}, \qquad i = 1 \dots n \tag{5}$$

the profile's deviation in waviness and roughness will be separated from the shape deviation using surface profile filters. Applying spline filters as proposed in ISO/TS 16610-22 as surface profile filters, an input function $z(x)$ is mapped to an output function $w(x)$ using spline

function filter kernels like the following for non-periodic spline filters:

$$(1 + \alpha^4 Q)w = z, \tag{6}$$

In contrast to extrapolating the curve endpoints as shown in [10], the non-periodic spline filter, based on natural cubic splines, already provides natural boundary conditions, i.e. second derivative at the free boundaries equals zero. The periodic spline filter for closed profiles uses an adapted matrix to ensure continuous second derivatives as boundary condition. For further details on spline filters as linear operators see [11] and [12]. The piecewise polynomials smoothen the input data in a way that minimizes the bending and therefore results in low pass filtered surface profiles. To control the smoothness of the result, the concept of wavelength based filtering is implemented using the parameter α, given by:

$$\alpha = \frac{1}{2sin(\frac{\pi \Delta x}{\lambda_c})}. \tag{7}$$

The cutoff-wavelength λ_c as smoothing parameter is used to separate the waviness and roughness of the input profile data. The point spacing Δx is computed by averaging the distance between all points along the contour. The spline filter returns relocated spatial positions for each contour point, according to the smoothing parameters. To avoid improperly chosen parameters, the average contour point distance for the two dimensional case:

$$d_j = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}, \quad i = 1 \dots n \tag{8}$$

is used to calculate the average point spacing Δx:

$$\Delta x := \overline{d} = \frac{1}{n-1} \sum_{j=1}^{n-1} d_j. \tag{9}$$

When increasing the cutoff-wavelength λ_c, the overall contour smoothing increases in a way that the noisy curvature signal for the contour will be smoothened while the underlying shape is still preserved. By raising the scale the contour corner points will be smoothened as well. To loop through the scales, a scale space[1] using different smoothing parameters λ_c is utilized by the ratio:

$$\frac{\lambda_c}{\Delta x}. \tag{10}$$

When utilizing $\frac{\lambda_c}{\Delta x}$ as scale space interval, the contour smoothing increases by raising the scale. Figure 4 depicts the spline filter scale space using 85 scales (loop through $\frac{\lambda_c}{\Delta x}$ in the range $15 \dots 100$), drawn on top of the contour profile from fig. 2. It is important to note, that the shape to be filtered could collapse, when rising the scale to high.

[1]Scale space methods employing Gaussian filter kernels for shape description and analysis were already proposed by [10], [13]

In the case presented here, the shape will deviate more and more from its original form. The major corner in the middle of the contour will stay throughout all scales and it will become the major feature at all scales, while the two smaller corner points at the end of the contour will be completely leveled out and therefore they will be adapted by the smoothing scale to the shapes more and more evident oval form.
To display the detected regions, the binary encoding of the smoothened contour curvature is indicated by dark red color, for regions, where:

$$\kappa(t) > \frac{1}{n-1} \sum_{j=1}^{n-1} \kappa_j. \tag{11}$$

Regions with smaller curvature are displayed in light

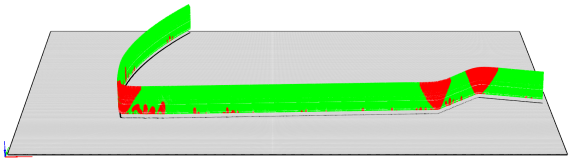

Fig. 4. Spline filter scale space indicating feature points of higher curvature than the profiles average in dark red color.

green color. Figure 4 depicts that contour irregularities are still present in lower scales, while at higher scales the contour is fairly smooth. Even the corner points were smoothened and therefore bend in a way that the length of possible corner segments decline, but they still remain the major feature points over all contour points. The serious difference in curvature between feature points and profile points superimposed by waviness could be seen at every scale (indicated using dark red color, see fig. 4). Detecting feature points means finding high curvature points at every scale. When iterating through the scales, the regions of high curvature are still present, if they resemble a corner region. Contour irregularities disappear by rising the scale. Since the regions containing features increase in length, only the minimal length throughout all scales is of interest. Further constraints could be the minimal length of feature point segments, in order to exclude small segments deviated by contour irregularities. Since a line intersecting two points does not resemble deviation, the minimum amount of points could be set to three or higher. The feature point detection throughout all scales is displayed in fig. 5, where the input contour data is split into segments using the feature regions detected. To discriminate straight segments from others, threshold based heuristics could be applied. Sezgin [14] proposed to compute the ratio of accumulated point distances along the extracted segment and the direct distance between beginning and end of the segment:

$$r = \frac{l_2}{l_1} \tag{12}$$

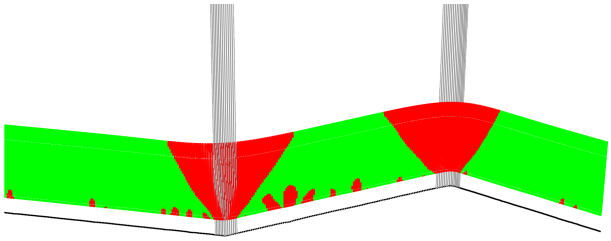

Fig. 5. Profile feature points at minimum width detected at every scale indicated by vertical lines. The original profile is drawn using black dotted line style.

The contour segments with values close to 1 represent the candidate segments for straight line estimation. Setting this threshold value highly depends on the surfaces roughness as well. An almost perfectly smoothened point cloud used to evaluate the threshold could lead to many straight line misses, if the subsequent profiles inspected contain a higher deviation among the contour points. This might be an useful application for significance measurements, using the contours deviation from an approximation element to check whether a line fitting is appropriate or not. The straight line segment fit afterwards is determined by using two dimensional total least squares regression method. The calculated center of gravity of the segments point cloud equals the position vector of the straight line representation. The direction could be estimated calculating the eigenvector with the biggest absolute eigenvalue of the covariance matrix V:

$$V = \frac{1}{n} \begin{bmatrix} \sum_{j=0}^{n} (x_j - \overline{x})^2 & \sum_{j=0}^{n} (x_j - \overline{x})(y_j - \overline{y}) \\ \sum_{j=0}^{n} (x_j - \overline{x})(y_j - \overline{y}) & \sum_{j=0}^{n} (y_j - \overline{y})^2 \end{bmatrix} \quad (13)$$

The segments extracted can be used for further computation or for measurement tasks as proposed by Werner in [15]. Figure 6 shows the detected profile segments using measuring fields as overlay. The indicated rectangular measuring fields in fig. 6 are set automatically, providing perpendicular set searchlines to scan the image scene. The new automated workflow is shown in fig. 7, where subsequent to the edge detection and subpixeling process, the automated feature extraction replaces the former extensive block of manual user interaction. The selection process of linked measurement tasks for the features extracted remains.

V. Conclusion

The method presented provides further improvements to state of the art methods, especially through its adaptive character when applied to object contours which were superimposed by roughness and waviness. The smoothened contour preserves relevant shape information for further

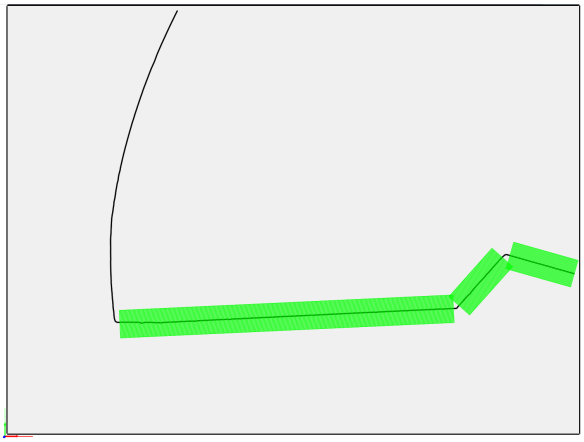

Fig. 6. Results for straight line segment candidates indicated by measuring fields for the searchline based measurement approach.

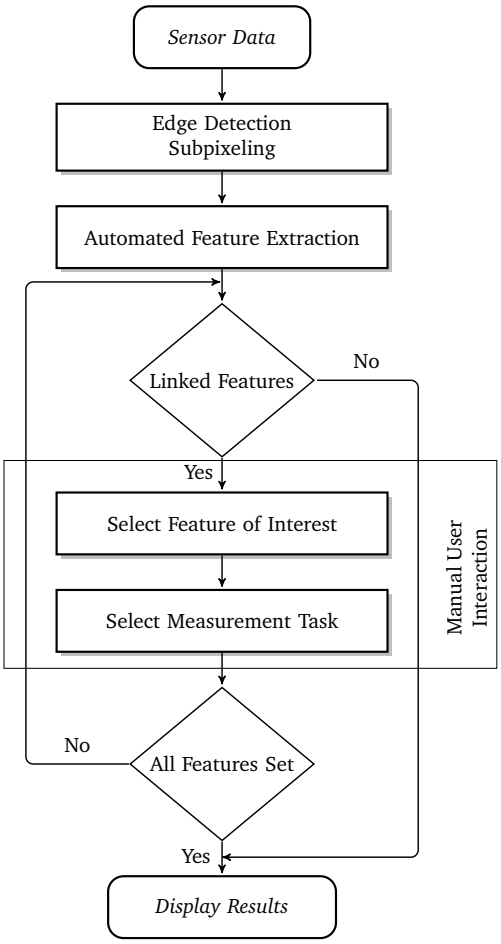

Fig. 7. The workflow for the manual user interaction to set up regions of interest is replaced by the automated feature extraction.

feature detection and extraction. The voting of curvature points throughout all scales was used to detect feature points. In the outlook, further improvements like split and merge of already detected straight line segments could be applied to reduce false positive detections and especially to extend the line segments more closely to the corner

points. Furthermore the method needs to be extended to distinguish not only straight line segments from corner points but also circular and elliptical segments as well. Based on the feature detection method presented, automated measurement tasks could directly be executed without further user interaction. Less user interaction increases the usability of highly complex measurement software and therefore enables a more broad use of it. The inspection plan generated is reusable and was provided without prior knowledge of the milling tools CAD data. Therefore this approach provides fast and automated generation of inspection plans reducing the user influenced workflow during the measurement process.

Acknowledgment

The support of the TMBWK ProExzellenz initiative, Graduate School on Image Processing and Image Interpretation at Ilmenau University of Technology is greatfully acknowledged.

References

[1] E. Reetz et al., "Application of spline surface profile filters to subpixel contour decomposition problems," in *Advanced Mathematical and Computational Tools in Metrology and Testing*, Series on Advances in Mathematics for Applied Sciences, vol. 84, F. Pavese et al., Eds., vol. 9. Singapore: World Scientific, 2012, pp. 334–342.

[2] O. Kühn, "Ein Beitrag zur hochauflösenden Geometriemessung mit CCD-Zeilensensoren," Dissertation, Technische Universität Ilmenau, Ilmenau, Mai 1997.

[3] M. Schumann et al., "Extraction of geometrical primitives from a set of contour points," in *Proceedings of the 10th International Symposium on Measurement and Quality Control*, Osaka, Japan: JSPE Technical Committee for Intelligent Nano-Measure, September 2010, pp. F5-029-1 – F5-029-4.

[4] P. V. C. Hough, "Methods and means for recognizing complex patterns," United States of America Patent 3 069 654, December 18, 1962.

[5] D. G. Lowe, "Three-dimensional object recognition from single two-dimensional images," *Artificial Intelligence*, vol. 31, pp. 355–395, 1987.

[6] G. L. West and P. L. Rosin, "Techniques for segmenting image curves into meaningful descriptions," in *IEEE Transactions On Pattern And Machine Intelligence*, vol. 24, no. 7. Pergamon Press, Jan. 1991, pp. 643–652.

[7] P. Faber, "Parameterlose Konturesegmentierung," in *DAGM-Symposium*, 1999, pp. 172–180.

[8] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[9] J. Steinier, Y. Termonia, and J. Deltour, "Comments on smoothing and differentiation of data by simplified least square procedure," *Analytical Chemistry*, vol. 44, no. 11, pp. 1906–1909, 1972.

[10] F. Mokhtarian and A. Mackworth, "Scale-based description and recognition of planar curves and two-dimensional shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 1, pp. 34–43, Jan. 1986.

[11] M. Krystek, "Discrete linear profile filters," in *X. International Colloquium on Surfaces Chemnitz (Germany)*, M. Dietsch and H. Trumpold, Eds. Aachen: Shaker, 2000, pp. 145–152.

[12] B. Muralikrishnan and J. Raya, *Computational Surface and Roundness Metrology*. London: Springer, 2009.

[13] D. G. Lowe, "Organization of smooth image curves at multiple scales," in *Second International Conference on Computer Vision*, Dec 1988, pp. 558–567.

[14] T. M. Sezgin, T. Stahovich, and R. Davis, "Sketch based interfaces: early processing for sketch understanding," in *ACM SIGGRAPH 2006 Courses*, ser. SIGGRAPH '06. New York, NY, USA: ACM, 2006.

[15] P. Werner et al., "A novel method for an automated analysis of a measurement scene," in *Advanced Mathematical and Computational Tools in Metrology and Testing*, Series on Advances in Mathematics for Applied Sciences vol. 84, F. Pavese et al., Eds., vol. 9. Singapore: World Scientific, 2012, pp. 334–342.