INSTITUT FÜR INFORMATIK

UND PRAKTISCHE MATHEMATIK

Analysis of Local Image Structure using Intersections of Conics

Christian B.U. Perwass

Bericht Nr. 0403
Version 1.0, July 2004

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

Institut für Informatik und Praktische Mathematik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Analysis of Local Image Structure using Intersections of Conics

Christian B.U. Perwass

Bericht Nr. 0403
Version 1.0, July 2004

e-mail: chp@ks.informatik.uni-kiel.de

Contents

Abstract

We propose an algorithm for the analysis of local image structure that is able to distinguish between a number of different structures like corners, crossings, y-junctions, t-junctions, lines and line segments. Furthermore, parameters of the detected structures can be evaluated, as, for example, the opening angle of corners. The main idea of the algorithm is to fit the intersection of two conics to the local image structure. The results of the algorithm when applied to synthetic and real data will be presented.

Chapter 1

Introduction

An important problem in Computer Vision is the recognition of objects from images. In order to recognize objects, they have to be represented appropriately. What an appropriate representation of an object is, is still a matter of research. However, a promising approach seems to be the use of "key features" for an object [1, 2, 3]. At the lowest level, local image structures that are intrinsically two dimensional (i2D), are the most useful ones, since they can be identified with a specific position in an image. Intrinsically one dimensional (i1D) structures, on the other hand, allow only for an exact localization in one dimension. Furthermore, i2D structures often stem from defining elements of an object, like the corners of a cube, for example. Due to the importance of i2D structures in many areas of Computer Vision, a lot of effort has gone and still goes into the development of robust edge and corner detectors, e.g. [4, 5, 6, 7]. Typically, corner detectors point to places in an image where the intrinsic dimensionality is two. However, this is the case for many different image structures like corners, line endings, line crossings, y-junctions and t-junctions. It would certainly be advantageous to tell these different types of structures apart. Consequently, there has been some effort to analyze i2D structures further, see e.g. [8, 9, 10]. Apart from determining the type, it would also be useful to extract some parameters of the structures, like the opening angle of a corner, or of a crossing.

In this report we propose a method to extract the type of an i2D image structure and to evaluate its parameters, like the opening angle of a corner, for example. This is done in two main steps. In the first step the edges of an image are extracted. In the second step the edge points are then analyzed locally by finding the two best fitting conics, which enables us to extract the type of structure. Note that we do not use the form of the fitted conics directly to analyze the image structure, as for example in [11], but consider their intersections instead. This report is dedicated to the description of the second step.

Chapter 2

The Algorithm

As mentioned in the introduction, an image is reduced to a set of edge pixels in an initial preprocessing step. The assumption we then make is that within a local area the edge pixels can be segmented into a set of line segments. A corner, for example, consists of two line segments that meet in the local area. Even though we are looking for line segments, it is not obvious of how to fit lines to the data, since it is in general not known how many line segments are present. A y-junction, for example, has three line segments, while a corner, or a t-junction only has two.

The basic idea we follow here is to perform an eigenvector analysis of the edge data in a local area. However, instead of using the data directly we first transform it to some other space, where the eigenvectors represent conics. From a geometric point of view, we try to find the conics that best fit the data. How this can be used to segment the data into line segments will be described later. First the embedding used and the eigenvector analysis are discussed.

2.1 The Vector Space of Conics

From the set of all edge points, we choose a subset of points, which lie in a particular local area. The pixel coordinates of the edge points are then transformed to coordinates relative to the center of the local area, such that the top left corner of the local area is at position $(-1, 1)$ and the bottom right corner at the position $(1, -1)$. The main reason for this transformation is to improve the numerical stability of the algorithm. Let the position vector of the ith edge point in the local area in transformed coordinates be denoted by the column vector $(u_i, v_i)^\mathsf{T}$. In a first step these vectors are embedded in a projective space. The ith edge point vector in this space is then written as $\mathsf{w}_i = (u_i, v_i, 1)^\mathsf{T}$. Next we embed w_i in a 6D-vector space of symmetric matrices, which allows us to fit conics to a set of data points. The details of this embedding are as follows.

It is well known that given a symmetric 3×3 matrix A, the set of vectors $\mathsf{x} = (x, y, 1)^\mathsf{T}$ that satisfy

$$\mathsf{x}^\mathsf{T}\,\mathsf{A}\,\mathsf{x} = 0, \tag{2.1}$$

lie on a conic. This can also be written using the scalar product of matrices, denoted here by $\cdot$, as

$$(\mathsf{x}\mathsf{x}^\mathsf{T}) \;\cdot\; \mathsf{A} = 0. \tag{2.2}$$

It makes therefore sense to define a vector space of symmetric matrices in the following way. If a_{ij} denotes the component of matrix A at row i and column j, we can define the transformation $\mathcal{T}$ that maps elements of $\mathbb{R}^{3\times3}$ to $\mathbb{R}^6$ as

$$\mathcal{T}\;:\;\mathsf{A} \in \mathbb{R}^{3\times3} \mapsto (a_{13},\, a_{23},\, \tfrac{1}{\sqrt{2}}\,a_{33},\, \tfrac{1}{\sqrt{2}}\,a_{11},\, \tfrac{1}{\sqrt{2}}\,a_{22},\, a_{12})^\mathsf{T} \in \mathbb{R}^6. \tag{2.3}$$

A vector $\mathsf{x} \in \mathbb{R}^3$ may now be embedded in the same six dimensional space via

$$\mathbf{x} := \mathcal{T}(\mathsf{x}\mathsf{x}^\mathsf{T}) = (x,\, y,\, \tfrac{1}{\sqrt{2}},\, \tfrac{1}{\sqrt{2}}\,x^2,\, \tfrac{1}{\sqrt{2}}\,y^2,\, x\,y)^\mathsf{T} \in \mathbb{R}^6. \tag{2.4}$$

If we define $\mathbf{a} := \mathcal{T}(\mathsf{A})$, then equation (2.1) can be written as the scalar product

$$\mathbf{x}^\mathsf{T}\,\mathbf{a} = 0 \iff x^2\,a_{11} + y^2\,a_{22} + 2xy\,a_{12} + 2x\,a_{13} + 2y\,a_{23} + a_{33} = 0. \tag{2.5}$$

Finding the vector $\mathbf{a}$ that best satisfies the above equation for a set of points $\{\mathbf{x}_i\}$ is usually called the algebraic estimation of a conic [12].

In the following we will denote the 6D-vector space in which 2D-conics may be represented by $\mathbb{D}^2 \equiv \mathbb{R}^6$. A 2D-vector $(x, y) \in \mathbb{R}^2$ is transformed to $\mathbb{D}^2$ by the function

$$\mathcal{D}\;:\quad (x,y) \in \mathbb{R}^2 \;\mapsto\; (x,\, y,\, \tfrac{1}{\sqrt{2}},\, \tfrac{1}{\sqrt{2}}\,x^2,\, \tfrac{1}{\sqrt{2}}\,y^2,\, xy) \in \mathbb{D}^2. \tag{2.6}$$

2.2 The Eigenvector Analysis

In order to analyze the edge data, we embed the data vectors $\{\mathsf{w}_i\}$ in the vector space of symmetric matrices as described above, i.e. $\mathbf{w}_i := \mathcal{D}(\mathsf{w}_i)$. Denote by $\mathbf{W}$ the matrix constructed from the $\{\mathbf{w}_i\}$ as $\mathbf{W} = (\mathbf{w}_1,\, \ldots,\, \mathbf{w}_N)^\mathsf{T}$, where N is the number of data vectors. A conic $\mathbf{a} = \mathcal{T}(\mathsf{A})$ that minimizes $\|\mathbf{W}\,\mathbf{a}\|^2$ is then a best fit to the data in an algebraic sense. The key to our algorithm is not just to look at the best fit but at the *two* eigenvectors of $\mathbf{W}$ with the two smallest eigenvalues.

In order to obtain real valued eigenvalues and orthogonal eigenvectors, we evaluate the eigenvectors and eigenvalues of $\mathbf{W}$ by performing a singular value decomposition (SVD) on $\mathbf{W}^{\mathsf{T}}\,\mathbf{W}$, which is symmetric. The singular vectors are then simply the eigenvectors and the square root of the singular values gives the eigenvalues of $\mathbf{W}$.

If $\mathbf{W}$ has two small eigenvalues, this means that the whole subspace spanned by the corresponding eigenvectors is a good fit to the data. In mathematical terms this can be written as follows. Throughout this text we will use $\mathbf{c}_1, \mathbf{c}_2$ to denote the two eigenvectors with smallest eigenvalues of $\mathbf{W}$. For any $\alpha, \beta \in \mathbb{R}$, $\mathbf{c} = \alpha\,\mathbf{c}_1 + \beta\,\mathbf{c}_2$ is a good fit to the data. This may also be termed a pencil of conics. The base points that define this pencil of conics are those that lie on all conics in this pencil. These points are simply the intersection points of the conics $\mathbf{c}_1$ and $\mathbf{c}_2$. This can be seen quite easily. If $\mathbf{x}$ is an intersection point of $\mathbf{c}_1$ and $\mathbf{c}_2$, then $\mathbf{x}^{\mathsf{T}}\,\mathbf{c}_1 = 0$ and $\mathbf{x}^{\mathsf{T}}\,\mathbf{c}_2 = 0$. Hence,

$$\mathbf{x}^{\mathsf{T}}\,\mathbf{c} = \alpha\,(\mathbf{x}^{\mathsf{T}}\,\mathbf{c}_1) + \beta\,(\mathbf{x}^{\mathsf{T}}\,\mathbf{c}_2) = 0\;\forall\,\alpha, \beta \in \mathbb{R}. \tag{2.7}$$

It therefore seems sensible that the intersection points of $\mathbf{c}_1$ and $\mathbf{c}_2$ also contain important information about the structure of the data from which $\mathbf{W}$ was constructed. An example that this is indeed the case can be seen in figure 2.1. The dots in this figure represent the data points and the two hyperbolas are the conics represented by the two eigenvectors of the corresponding $\mathbf{W}$ matrix with the smallest eigenvalues. It can immediately be seen that each conic by itself does not represent the data distribution. However, their intersection points lie exactly in the four clusters formed by the data.

By intersecting conics $\mathbf{c}_1$ and $\mathbf{c}_2$, we basically try to represent the data in terms of up to four points. In effect, this is not much different from a principal component analysis (PCA). However, instead of looking for the eigenvectors that best describe the range of $\mathbf{W}$, we are interested in those that that best describe the null space of $\mathbf{W}$. A PCA would be useful if $\mathbf{W}$ were constructed from a set of conics, from which we wanted to extract the conic that best represents all. In the case presented here, the matrix $\mathbf{W}$ is constructed from a set of points (represented as degenerate conics), and we try to find the conics that best lie on all data points by looking for the null space of $\mathbf{W}$. The space of intersections of eigenvectors of $\mathbf{W}$ may actually be expressed as the vector space of bivectors in a Clifford algebra over the vector space of symmetric matrices. In this sense, the space of intersections may be regarded as a kind of "second order" null space of $\mathbf{W}$. See appendix B for more details.

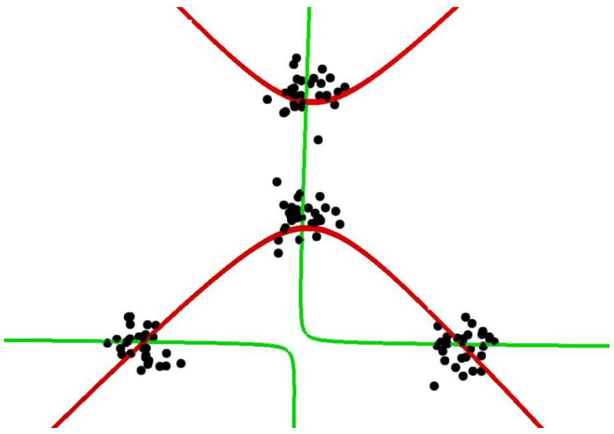

Figure 2.1: Four clusters of data points and the conics represented by the two eigenvectors with the smallest eigenvalues.

2.3 Analyzing Line Pairs

In figure 2.1 it was shown how the intersection of two conics can be used to fit a point quadruplet to clustered data. In this section we will show how the idea of intersections of conics can be used to analyze line pairs.

The first observation we can make is that two lines in $\mathbb{R}^2$ can always be represented by a projective conic. For example, two intersecting lines can be constructed by the intersection of a cone with a plane that passes through the cone's origin. When we are working with projective conics, then it is also possible to represent two parallel lines as a conic section. Some examples of the representation of conics centered on the origin may help to further the understanding of this.

A projective conic centered at the origin with its main axes along the coordinate axes may be represented by a diagonal 3×3 matrix. If $\mathsf{A} \in \mathbb{R}^{3\times 3}$ is a diagonal matrix, then these types of conics are given by the set of vectors $\mathsf{x} \in \mathbb{R}^3$ that satisfy $\mathsf{x}^\mathsf{T}\mathsf{A}\mathsf{x} = 0$. Note that A represents a conic in $\mathbb{R}^2$ and x is a projective embedding of points in $\mathbb{R}^2$ just as in section 2.1. We will write A as

$$\mathsf{A} = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & -\rho \end{pmatrix}. \tag{2.8}$$

Different values of λ_1, λ_2 and ρ represent different types of conic. This can be seen best by expanding the condition $\mathsf{x}^\mathsf{T}\mathsf{A}\mathsf{x} = 0$.

$$\mathsf{x}^\mathsf{T}\mathsf{A}\mathsf{x} = 0 \iff \lambda_1\, x^2 + \lambda_2\, y^2 = \rho, \tag{2.9}$$

where $\mathsf{x} = (x, y, 1)$. Clearly, if $\lambda_1 = \lambda_2 = 1$, then A represents a circle of radius $\sqrt{\rho}$. Furthermore,

- **Ellipse**. $\lambda_1, \lambda_2 > 0$ and $\rho > 0$.
- **Hyperbola**. $\lambda_1 > 0$ and $\lambda_2 < 0$ or vice versa, and $\rho > 0$.
- **Two intersecting lines**. $\lambda_1 > 0$ and $\lambda_2 < 0$ or vice versa, and $\rho = 0$.
- **Two parallel lines**. $\lambda_1 > 0$ and $\lambda_2 = 0$ or vice versa, and $\rho > 0$.
- **Line**. $\lambda_1 > 0$ and $\lambda_2 = 0$ or vice versa, and $\rho = 0$.

Note that the two intersecting lines actually represent a 2D cone. In general one can say that ρ always gives some kind scaling or separation of the object in question. For example, for a circle $\sqrt{\rho}$ is the radius, for hyperbolas it gives the separation of their branches and for parallel line pairs it is the distance between the two lines.

These conics can now be translated and rotated by simply applying rotation and translation matrices to A. If R is a rotation matrix and T a translation matrix, such that $\mathsf{y} := \mathsf{R}\mathsf{T}\mathsf{x}$ is a rotated and translated point. This transformation of points can also be applied to A by observing that

$$\mathsf{y}^\mathsf{T}\,\mathsf{A}\,\mathsf{y} = 0 \iff \mathsf{x}^\mathsf{T}\,(\mathsf{T}^\mathsf{T}\,\mathsf{R}^\mathsf{T}\,\mathsf{A}\,\mathsf{R}\,\mathsf{T})\,\mathsf{x} = 0. \tag{2.10}$$

The matrix $(\mathsf{T}^\mathsf{T}\,\mathsf{R}^\mathsf{T}\,\mathsf{A}\,\mathsf{R}\,\mathsf{T})$ now represents a rotated and translated conic.

Let us return now to the fitting of conics to data as described in section 2.2. If the data points $\{\mathsf{w}_i\}$ are points on two lines, then the singular vector with the smallest singular value of W will give a conic that represents just those two lines. The singular vector with the next higher singular value has to be perpendicular to the first singular vector. Nevertheless, it will still fit the data in some respect. In particular the intersection points of the two conics have to lie on or near the lines, since the first conic is a good fit to the two lines.

2.4 Analyzing Image Data

The type of data that we want to analyze with the above described method, are sets of a few line segments, like those shown in figure 2.2. A standard PCA approach on 2D position vectors will not be of any use in this case, since this would not allow us to distinguish between differently oriented

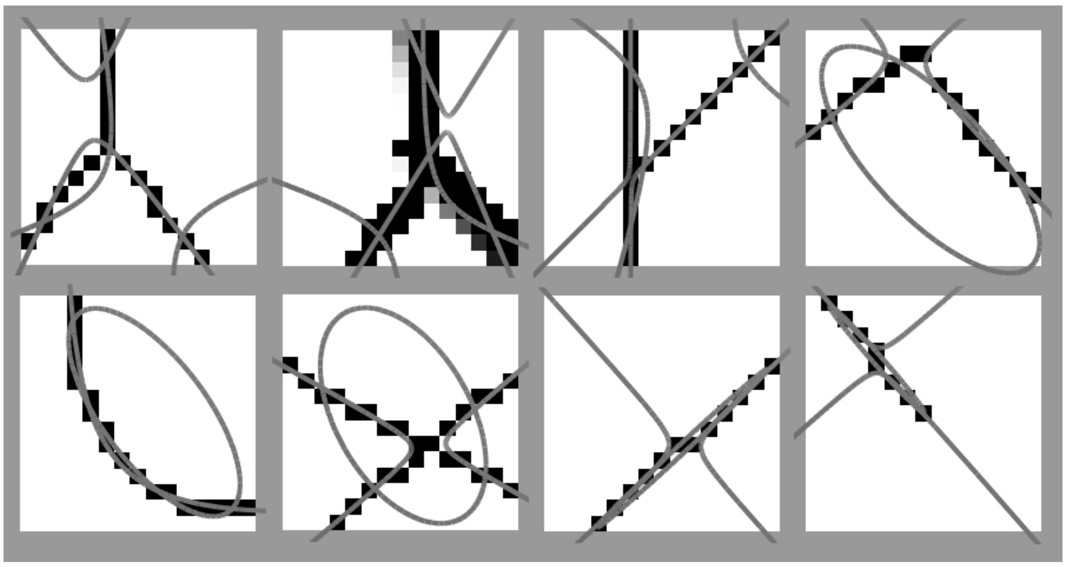

Figure 2.2: Examples of typical image structures.

line segments in the same data subspace. Instead we observed that the intersections of conics $\mathbf{c}_1$ and $\mathbf{c}_2$, represent the data in a very useful way. This can also be seen in figure 2.2, where the two conics are drawn on top of the image structures. The images show that the intersection points of $\mathbf{c}_1$ and $\mathbf{c}_2$ lie on the line segments, and that the line segments always lie between two intersection points. The image in the first row, second column of figure 2.2 shows that this also works if the lines are not thin, i.e. one pixel wide. Furthermore, the influence of the pixels can be weighted, since the vectors from which $\mathbf{W}$ is constructed, are embedded in a projective space. Scaling the rows of $\mathbf{W}$ therefore does not change the geometric content, but it changes the numerical influence of the points on the least squares fit performed by the SVD on $\mathbf{W}$. Mathematically this can be seen quite easily. Recall that the coordinates of the i^{th} data point are embedded as $\mathsf{w}_i = (u_i, v_i, 1)^\mathsf{T}$ and then mapped to the space of symmetric matrices via $\mathbf{w}_i := \mathcal{T}(\mathsf{w}_i\, \mathsf{w}_i^\mathsf{T})$. If for some conic $\mathbf{a}$ in this space we have $\mathbf{w}_i^\mathsf{T}\, \mathbf{a} = 0$, then $\alpha\, \mathbf{w}_i^\mathsf{T}\, \mathbf{a} = 0$ also has to be true for all $\alpha \in \mathbb{R}$. The $\mathbf{W}$ matrix is now constructed as $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_N)^\mathsf{T}$. Suppose that for some conic $\mathbf{a}$ we have $\mathbf{w}_i^\mathsf{T}\, \mathbf{a} = \epsilon_i \in \mathbb{R}$, then

$$\mathbf{a}^\mathsf{T}\, \mathbf{W}^\mathsf{T}\, \mathbf{W}\, \mathbf{a} = \sum_{i=1}^{N} \epsilon_i^2. \tag{2.11}$$

The singular vector with smallest singular value of a SVD on $\mathbf{W}^\mathsf{T}\, \mathbf{W}$, minimizes this sum. By scaling the vectors $\mathbf{w}_i$ separately, their influence on the final result can be changed. Suppose $\mathbf{W}' = (\alpha_1 \mathbf{w}_1, \ldots, \alpha_N \mathbf{w}_N)^\mathsf{T}$, $\alpha_i \in \mathbb{R}$, then

$$\mathbf{a}^{\mathsf{T}}\,\mathbf{W}'^{\mathsf{T}}\,\mathbf{W}'\,\mathbf{a} = \sum_{i=1}^{N} (\alpha_i\,\epsilon_i)^2. \tag{2.12}$$

In this way the influence of data pixels on the null space of $\mathbf{W}$ can be weighted, and it is therefore not necessary to use binary edge data for the algorithm to work.

The examples in figure 2.2 show that the intersection points of $\mathbf{c}_1$ and $\mathbf{c}_2$ are potentially very useful to describe different image structures, like y-junctions, t-junctions, crossings, corners, lines and line endings. In the following we will denote the set of intersection points of $\mathbf{c}_1$ and $\mathbf{c}_2$ in $\mathbb{R}^2$ as $\mathbb{S}_E \subset \mathbb{R}^2$. We use the subscript E for $\mathbb{S}$, since $\mathbb{S}_E$ contains the intersection points in Euclidean space $\mathbb{R}^2$. If $|\mathbb{S}_E| = 4$, that is, $\mathbf{c}_1$ and $\mathbf{c}_2$ intersect in four points, then there are six unique point pairs between which lines could occur. Typically, only a few of these lines are actually present in the image, though. Therefore, we are not finished once we have found $\mathbb{S}_E$. We also have to check which of the possible six lines have support in the image. Once we have identified such a subset, the last step will be to analyze the extracted line segments and to decide which type of structure, if any, is currently present.

2.4.1 Intersection of Conics

Finding the intersection points of two 2D conics is not trivial. In general one has to solve a polynomial equation of degree at most four. The method we use is described in detail in appendix C. In short, given two conics we find a linear combination of them that represents a degenerate conic, for example a line pair. This degenerate conic then also passes through the intersection points of the two initial conics. This allows us to evaluate the intersection points of the two conics by evaluating the intersection of the degenerate conic with one of the initial conics. This is much simpler than solving a polynomial of degree four, since it results in two polynomial equations of degree two. The only numerically sensitive operation we have to use is the evaluation of eigenvectors and eigenvalues, for which many stable numerical algorithms exist.

Figure 2.3 shows three examples of the intersection points found for two fitted conics. It can be seen that the line segments that make up the image structures can be described by the lines between the intersection points. Note that if an intersection point lies outside the local area under investigation, it will be neglected, since an extrapolation of line segments can lead to spurious line structures.

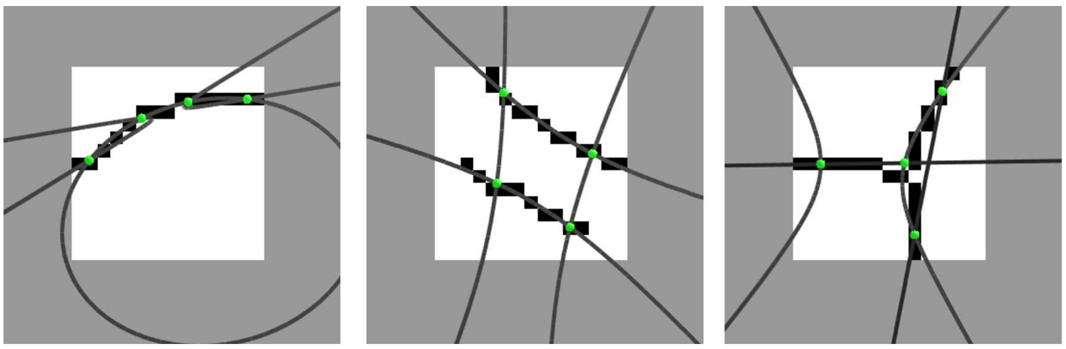

Figure 2.3: Examples of intersections of conics fitted to local image structures.

2.4.2 Finding Support for Lines

Given the set of intersection points $\mathbb{S}_E$ of two conics, the question now is which of the $\binom{|\mathbb{S}_E|}{2}$ lines, is actually present in the data, if any. The method we present here is rather simple but shows good results. The basic idea is as follows: the number of data points along a line segment should be at least as high as the separation between the two corresponding intersection points measured in pixels. Since the data points give the coordinates of edge pixels, this condition basically says that there is a closed line of pixels between two intersection points. In order to weaken this condition somewhat, we use the following mathematical approach to implement the idea.

Denote by $\mathbb{W} \subset \mathbb{R}^2$ the set of data points, i.e. the set of edge pixels in a local area. Furthermore, let $N_W = |\mathbb{W}|$ be the number of data points. The ith line segment will be written in parameterized form as

$$\mathsf{m}_i := \mathsf{a}_i + \alpha\, \mathsf{r}_i, \quad \alpha \in [0,1], \tag{2.13}$$

where $\mathsf{a}_i, \mathsf{r}_i \in \mathbb{R}^2$ and $\|\mathsf{r}_i\|$ gives the length of the line segment. The projection of a data point $\mathsf{w}_i \in \mathbb{W}$ onto the line segment m_j is defined as

$$\mathcal{P}_j(\mathsf{w}_i) := (\hat{\mathsf{r}}_j^\mathsf{T}\, (\mathsf{w}_i - \mathsf{a}_j))\, \hat{\mathsf{r}}_j, \tag{2.14}$$

where $\hat{\mathsf{r}}_j := \mathsf{r}_j / \|\mathsf{r}_j\|$. The rejection of data point w_i from line m_j is then given by

$$\mathcal{R}_j(\mathsf{w}_i) := (\mathsf{w}_i - \mathsf{a}_j) - \mathcal{P}_j(\mathsf{w}_i). \tag{2.15}$$

A distance measure d_{ij} between data point w_i and line segment m_j is then defined as

$$d_{ij} := \begin{cases} \|\mathcal{R}_j(\mathsf{w}_i)\| & : \ 0 \leq \|\mathcal{P}_j(\mathsf{w}_i)\| \leq \|\mathsf{r}_j\| \\ \infty & : \text{otherwise} \end{cases} \tag{2.16}$$

That is, we take as distance between a data point and a line segment the orthogonal separation of the point from the line segment, if the data point projects onto the line segment. If it does not, then the distance is taken as infinity. The latter condition implements the idea that a data point that does not project onto a line segment should not count at all towards the support of a line segment.

The support of the jth line segment is then given by

$$q_j^{sup} = \sum_{i=1}^{N_W} \exp\Big(-\frac{1}{2}\Big(\frac{d_{ij}}{\lambda\, d_{pix}}\Big)^2\Big), \tag{2.17}$$

where $d_{pix} \in \mathbb{R}$ gives the width of a pixel, and $\lambda \in \mathbb{R}$ is a scale factor. When $d_{ij} = 0$, then a data point lies directly on the line segment in question. This will then add unity to the support measure q_j^{sup}. The factor λ sets the support data points off the line segment add towards q_j^{sup}. If $d_{ij} \to \infty$, then this will add nothing to the q_j^{sup}, i.e. the corresponding data point adds no support to the respective line segment.

In order to decide whether an evaluated support measure q_j^{sup} represents good or bad support for a line segment, we have to evaluate the support that could ideally be expected for the line segment. Ideal support for a line segment means, that the maximum number of pixels possible along the line segment were present. If this is the case, the value of q_j^{sup} will be just this number of pixels. Since we only count those data points that appear between the end points of the line segment, the expectation value q_j^{exp} for q_j^{sup} can be evaluated as

$$q_j^{exp} := \frac{1}{d_{pix}} \max\Big\{ |r_j^1|,\, |r_j^2| \Big\} - 1, \tag{2.18}$$

where $\mathsf{r}_j := (r_j^1,\, r_j^2)$.

If $q_j^{sup} \geq q_j^{exp}$ we can be sure that the jth line segment has good support in the image. If, however, $q_j^{sup} < q_j^{exp}$ we should give the respective line segment a lower confidence value. The final quality measure for a line segment is therefore evaluated as

$$q_j := \begin{cases} \exp\Big(-\frac{1}{2}\Big(\frac{q_j^{sup} - q_j^{exp}}{\tau\, q_j^{exp}}\Big)^2\Big) & : \; q_j^{sup} < q_j^{exp} \\ 1 & : \; q_j^{sup} \geq q_j^{exp} \end{cases}, \tag{2.19}$$

where $\tau \in \mathbb{R}$ gives a measure of how close q_j^{sup} has to be to q_j^{exp} in order for it to give a high q_j value.

Every $q_j \in [0, 1]$ gives a measure of support for a line segment. The closer the value of q_j to unity, the more likely it is that the respective line segment is also present in the local image area under inspection. Which particular structure is present in the local image area depends on the combination of line segments with good support. It is therefore useful to collect the separate support measures in a support matrix. Let us denote the support value of the line segment between intersection points $\mathsf{s}_i \in \mathbb{S}_E$ and $\mathsf{s}_j \in \mathbb{S}_E$ by $q_{i,j} = q_{j,i}$. The support matrix Q is now defined as

$$\mathsf{Q} := \begin{pmatrix} 0 & q_{1,2} & q_{1,3} & q_{1,4} \\ q_{2,1} & 0 & q_{2,3} & q_{2,4} \\ q_{3,1} & q_{3,2} & 0 & q_{3,4} \\ q_{4,1} & q_{4,2} & q_{4,3} & 0 \end{pmatrix} \tag{2.20}$$

We can also regard Q as a weight matrix, giving the weights of the edges of a fully connected graph with four vertices. Note that if less than four conic intersection points are found, Q is reduced accordingly.

2.4.3 Analyzing the Line Segments

After the support for the set of possible line segments has been evaluated, we still have to analyze the set of lines and decide on the type of image structure that is present. This is actually not a trivial task. Figure 2.4 shows a set of typical structures that are encountered. In this collection of images the round points represent the conic intersection points found and the lines drawn show those lines for which sufficient support was found. The thicker a line, the higher its support value as evaluated in equation (2.19). In the following we will discuss the different structures separately.

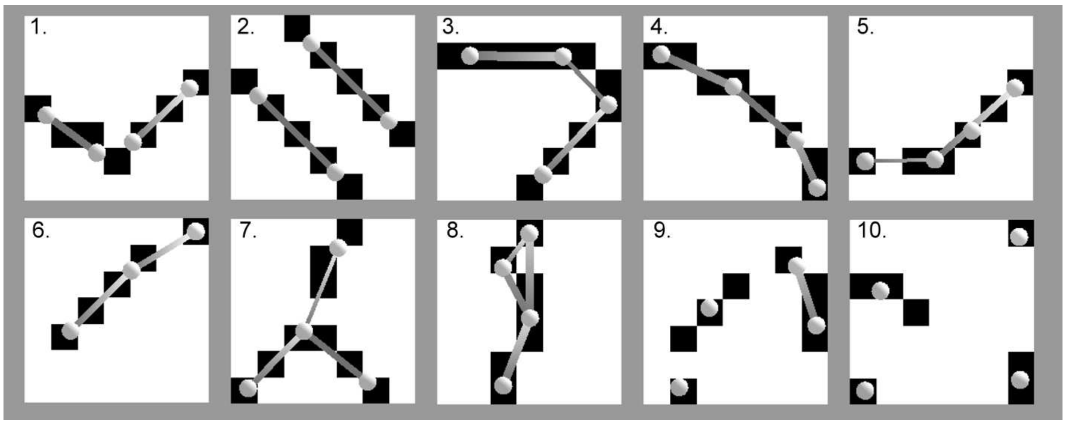

Figure 2.4: Examples of analyzed image structures.

1. **Line pair**. One may extrapolate these line segments and take their intersection point as a corner. In this case extrapolation would be warranted but in general it is unstable.

2. **Line pair**. This time the line segments are parallel. In this case it is clear that the two line segments do not describe a corner.

3. **4-Chain**. Here the line segments with good support in the image connect all four intersection points in a row. 4-chains can appear in many different forms, see for example images 4 and 5. If the two central points are close together we could regards this as a single corner. If they are further apart, as in image 4, the structure can be regarded as a double corner.

4. **4-Chain**. This time the 4-chain describes rather a curve than a corner. Nevertheless, one could still interpret it as a double corner.

5. **4-Chain**. In this case, two connected line segments have basically the same direction. This structure may therefore be regarded as a single corner, by combining the two similar line segments into one.

6. **3-Chain**. This constellation may immediately be interpreted as a corner. This structure also appears often on aliased lines, where the aliasing appears locally like a corner with a very wide opening angle.

7. **Star**. This can be regarded as a y-junction. If the angle between two legs is nearly 180 degrees, a star pattern represents a t-junction.

8. This structure has no clear interpretation.

9. This is actually a corner in the original image. However, the edge detection algorithm returned a disjunct contour, which makes it hard to analyze the structure at this scale.

10. Here a structure that cannot and should not return any output.

An example not shown here is that of a line. When a line is the only element in the local area that is analyzed, the four conic intersection points also lie almost on that line. In the following we will neglect such structures and concentrate on the detection of corners and junctions.

The structures we can interpret are thus a line pair, a 4-chain, a 3-chain and a star. Due to the way the support for lines is evaluated, these structures do not appear clearly in certain circumstances. It is therefore necessary to detect such situations and to adjust the support matrix Q accordingly. A

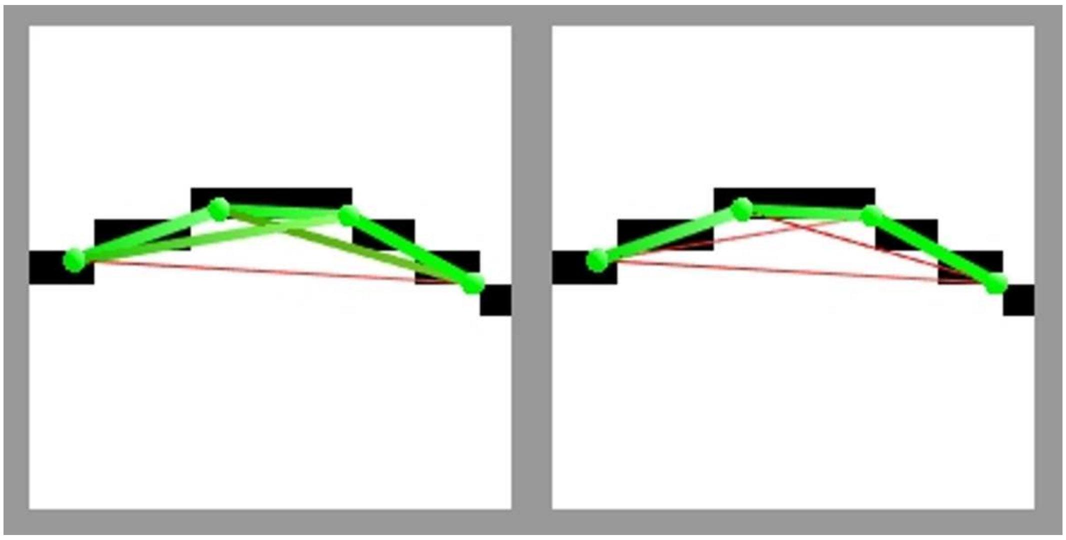

Figure 2.5: Examples of spurious lines with good support (left image) and the reduction of their support (right image).

typical example is shown in figure 2.5, where the points represent the intersection points and the thickness of the line segments reflects their support. What we would like to have would be the 4-chain that lies on the image structure. However, two additional lines have good support since they are near data points. One way to reduce the support value of these lines would be to reduce the factor λ from equation 2.17. That is, line segments do only get support when they pass very close to data points. However, this would also mean that the support evaluation would be very sensitive to incomplete structures. A better solution is to detect the problem structures explicitly.

The problem case shown in figure 2.5 is handled in the following way. We test every combination of three intersection points, whether their connecting line segments have all good support in the image. If this is case, we test whether the distance of any of the points in the triplet from their opposing line segment is below a certain threshold. If this is the case, the support value for the opposing line segment is reduced. The distance measure used for this purpose, is the one from equation 2.16. The effect of applying this method to the left image in figure 2.5 can be seen in the right image. Now only a 4-chain is left.

An example of another problem that may occur is shown in figure 2.6. Of the intersection points found initially, two points are very close. In fact, their distance is less than a pixel such that the line segment between them can have no support. Therefore, also the method for reducing the line segments of intersection point triplets with good support, as described above, fails.

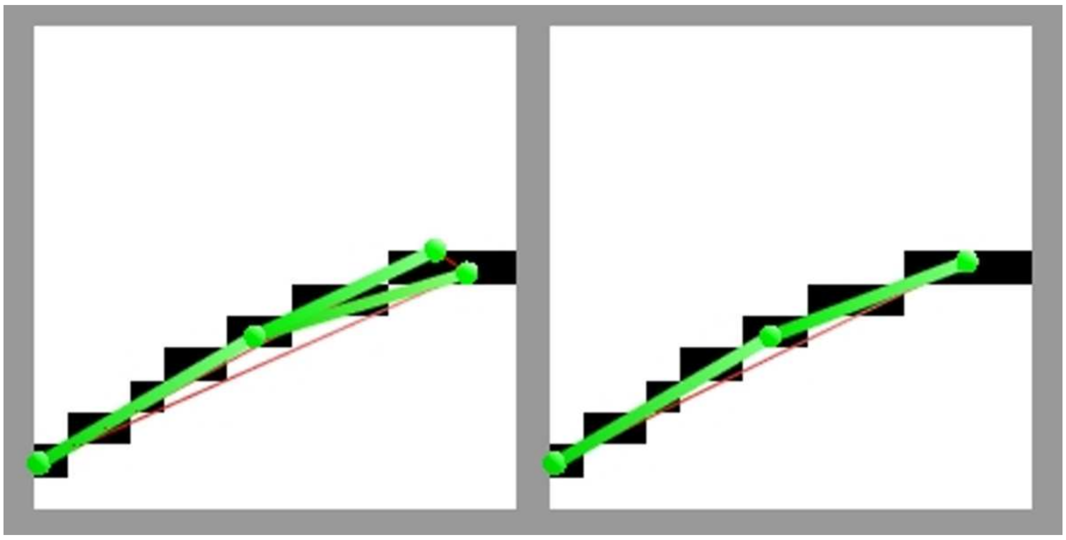

Figure 2.6: Examples of two close points which are preferably interpreted as a single point. While the structure found in the left image represents a spurious junction, the combination of the two close points results in the correct structure.

Therefore, we need to apply an additional correction method: if the distance between two intersection points is below a given threshold, the two points are replaced by a single point that lies half way between them. The result of this method when applied in the case presented in the left image of figure 2.6 can be seen in the figure's right image. The spurious structure is now reduced to a structure than can be readily interpreted.

Given an adjusted set of intersection points and line segments, the next step is to test the line segment structure for one of the different patterns described above. We will describe the method used with the help of an example. Figure 2.7 shows the intersection points and the line segments with their respective weights found for an image structure. Let Q denote the support quality matrix for this structure as defined in equation (2.20). In this case, the values $\mathsf{Q}_{1,2}$, $\mathsf{Q}_{1,3}$ and $\mathsf{Q}_{1,4}$ are close to unity and the values $\mathsf{Q}_{2,3}$, $\mathsf{Q}_{3,4}$ and $\mathsf{Q}_{4,2}$ are close to zero. We can therefore evaluate a measure of confidence that the present structure is a star as

$$C = (\mathsf{Q}_{1,2}\,\mathsf{Q}_{1,3}\,\mathsf{Q}_{1,4})\,(1 - \mathsf{Q}_{2,3}\,\mathsf{Q}_{3,4}\,\mathsf{Q}_{4,2}) \tag{2.21}$$

That is, we have to test for a positive and a negative pattern. Since the numbering of the intersection points is arbitrary, the above measure will in general have to be evaluated for all permutations of $\{1, 2, 3, 4\}$. In order to formulate this mathematically, let us denote by $\mathbf{i}$ and index vector defined as $\mathbf{i} := (i_1, i_2, i_3, i_4)$. We can then define a positive (p^+) and a negative

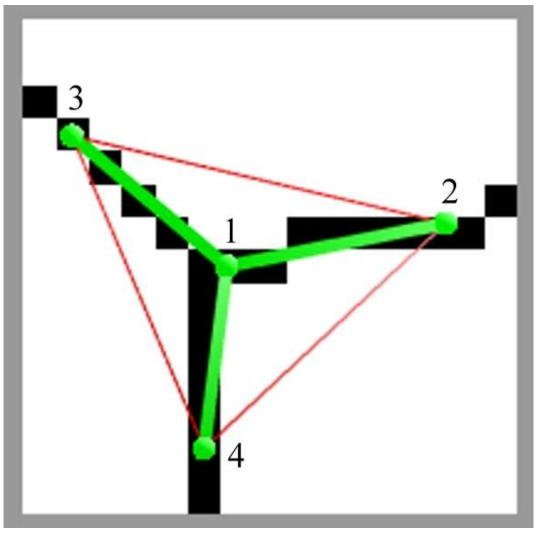

Figure 2.7: Examples of a junction structure.

(p^-) pattern that we expect for a particular structure. In the case of the star structure, these patterns are

$$\begin{aligned} p^+(\mathbf{i}) &= \Big((i_1, i_2),\, (i_1, i_2),\, (i_1, i_4)\Big), \\ p^-(\mathbf{i}) &= \Big((i_2, i_3),\, (i_3, i_4),\, (i_2, i_4)\Big). \end{aligned} \tag{2.22}$$

In the following let p_k^+ denote the kth index pair of p^+, and analogously for p^-. In order to improve the readability of the following formulas, we will also write $\mathsf{Q}[i_1,\, i_2]$ in order to denote the element Q_{i_1,i_2}. The confidence value for the star pattern for a particular $\mathbf{i}$ may then be written as

$$C\Big(p^+(\mathbf{i}),\, p^-(\mathbf{i})\Big) = \Big(\prod_k Q[p_k^+(\mathbf{i})]\Big)\Big(1 - \prod_l Q[p_l^-(\mathbf{i})]\Big) \tag{2.23}$$

The permutation of $\mathbf{i}$ that gives the largest value of $C(p^+(\mathbf{i}),\, p^-(\mathbf{i}))$ then allows us to evaluate the central point of the star (i_1) and the three end points (i_2, i_3, i_4). We will denote this value of $\mathbf{i}$ as $\hat{\mathbf{i}}$, with

$$\hat{\mathbf{i}} = \arg \max_{\mathbf{i} \in \mathrm{perm}\{(1,2,3,4)\}} C\Big(p^+(\mathbf{i}),\, p^-(\mathbf{i})\Big), \tag{2.24}$$

where $\mathrm{perm}\{(1,2,3,4)\}$ denotes the set of index vectors of permutations of $(1,2,3,4)$.

For each structure that we would like to test for, we can define a positive (p^+) and a negative (p^-) pattern and then evaluate the confidence value

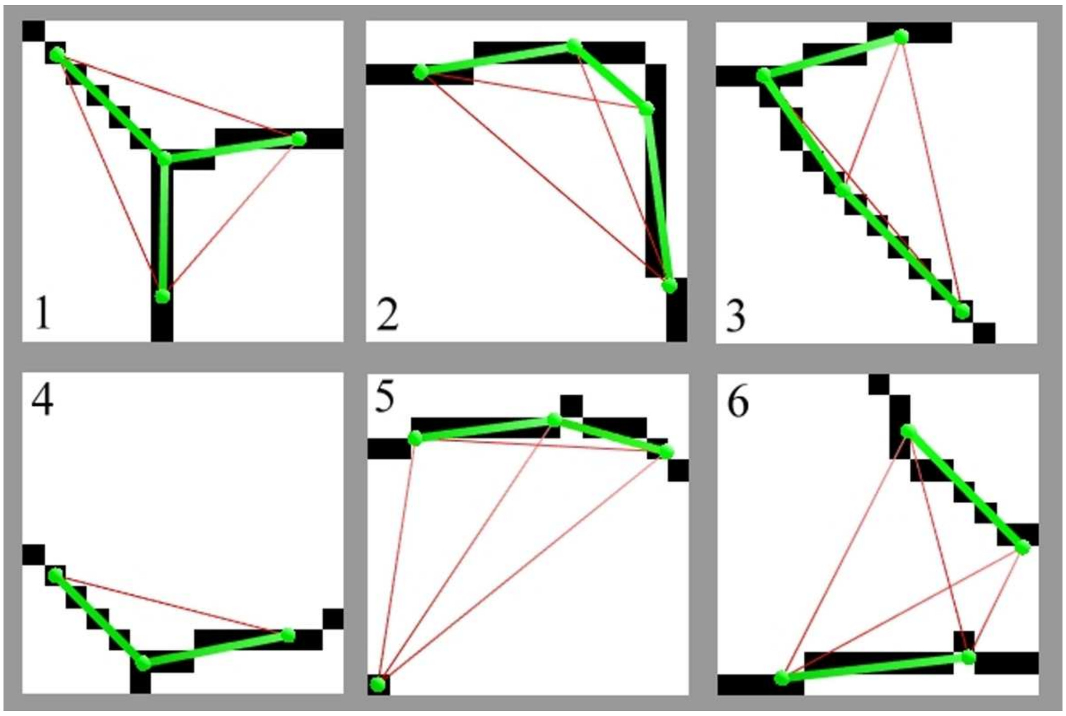

Figure 2.8: Examples of structures tested for.

$C(p^+(\hat{\mathbf{i}}),\, p^-(\hat{\mathbf{i}}))$ on a given Q matrix. In our implementation of the algorithm we test for the star, the 4-chain, the 3-chain, the 3-chain with a disjoint point and the line pair. Typical examples of these structures are shown in figure 2.8. Note that image 5 shows a 3-chain with a disjoint point and images 2 and 3 both show 4-chains. The latter two structures should be interpreted in different ways. While image 2 can be interpreted as a double corner, image 3 should be interpreted as a single corner. This shows that by finding the best matching structure to a local image area, we still cannot make a final decision on what the structure represents.

2.4.4 Analyzing Line Structures

For each of the structures we test for, we obtain a confidence value $C(p^+(\hat{\mathbf{i}}),\, p^-(\hat{\mathbf{i}}))$. The structure with the highest confidence value is then analyzed further to decide whether it represents a corner, a double corner or a junction. One could also test for a curve, a line or a line pair, but in this text we are mainly interested in finding corners and junctions.

The Star

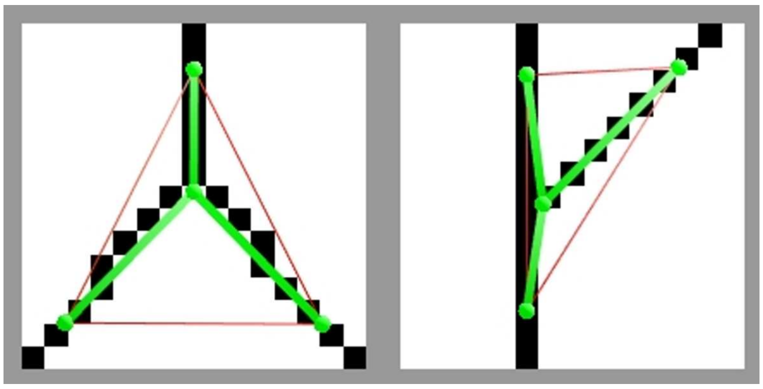

Figure 2.9: Examples of a y-junction (left) and a t-junction (right).

The star can be interpreted immediately. Since we have $\hat{\mathbf{i}}$ we know which of the intersection points is the central point and which are the three edge points. From this the position of the junction in the image and the angles between the legs can be readily evaluated. If the angle between two legs is nearly 180 degrees one may also call the junction a t-junction and otherwise a y-junction. See figure 2.9 for an example.

The 4-Chain

A 4-chain can represent a number of different entities: a corner, a double corner, a curve and a line. The difference between these entities cannot be defined strictly in general. Which structure is present depends on the angles between the legs of the 4-chain. Here thresholds have to be set that are most appropriate for the current application. In this text we will concentrate on distinguishing between corners and junctions. Therefore, a curve will also be interpreted as a corner with large opening angles. See figure 2.10 for examples of these structures.

Two angles (α_1, α_2) can be evaluated between the three legs of a 4-chain. Since we always take the smaller angle between two line segments, we have to make sure that the present 4-chain does not have a form as in the bottom-right image of figure 2.10, which we will call a "snake". This can be checked by evaluating the cross products of the directions of the line segment pairs from which the angles are evaluated. If the resultant vectors point in opposite directions, then the 4-chain describes a snake.

The other structures are distinguished using α_1 and α_2 as follows.

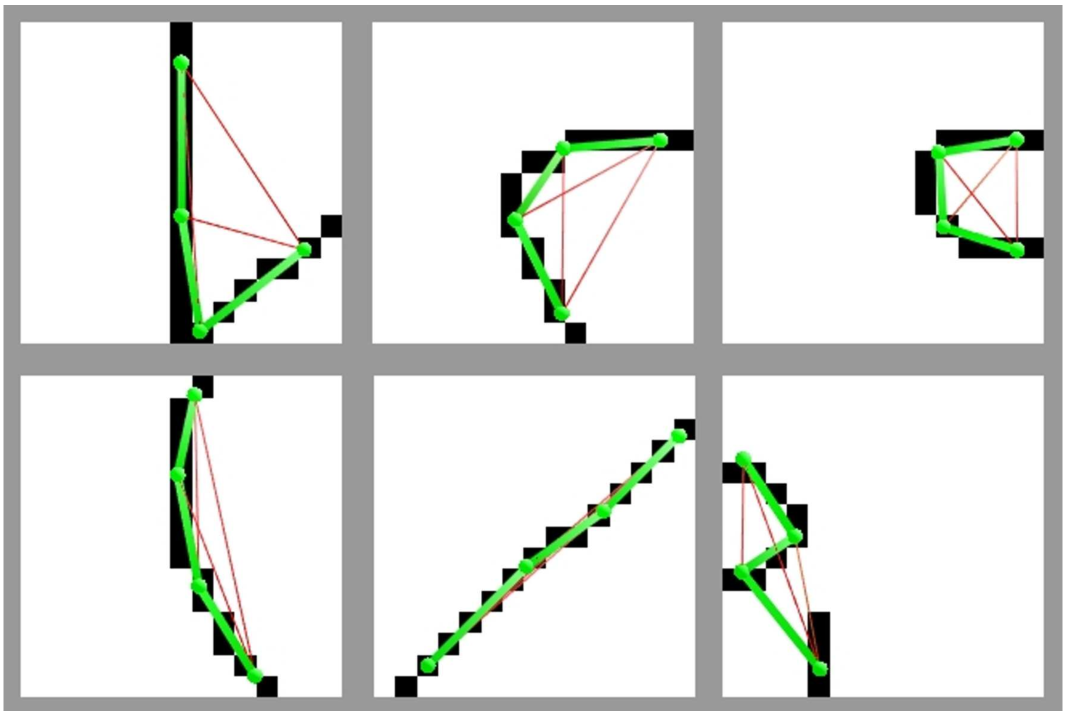

Figure 2.10: Examples of entities that can be described by a 4-chain. From top-left to bottom-right: corner, double corner, double corner, curve, line, "snake".

- **Line**, $\alpha_1 > 170°$ and $\alpha_2 > 170°$.
- **Double Corner**, $\alpha_1 < 150°$ and $\alpha_2 < 150°$.
- **Corner**, in all other cases. The corner is given by the two line segments with the larger angle between them.

The 3-Chain

A 3-chain either describes a corner or a line. It usually appears if one of the intersection points of the conics lies outside the local image area under investigation and is thus neglected. A 3-chain can also appear if two intersection points are so close to each other that they are combined into a single point. Examples of these cases can be seen in figure 2.11. The left most image in this figure shows a 3-chain with a disjunct intersection point. These cases occur when within the local image area there is a corner and some pixels from a neighboring structure.

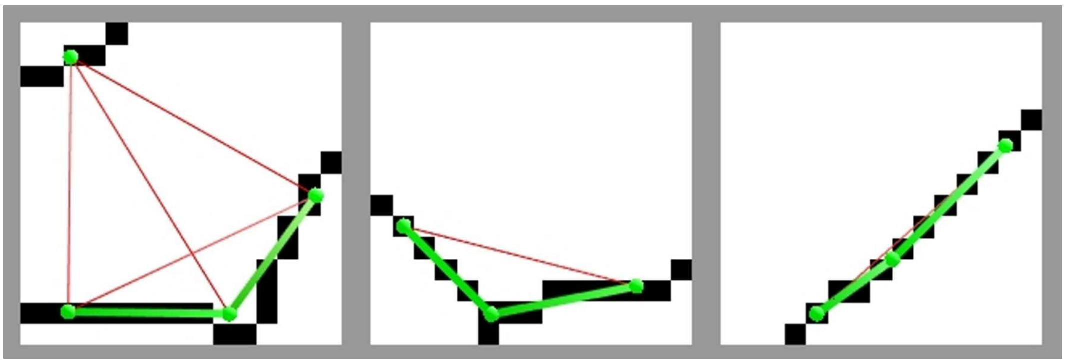

Figure 2.11: Examples of 3-chains. Both describe a corner.

The Line Pair

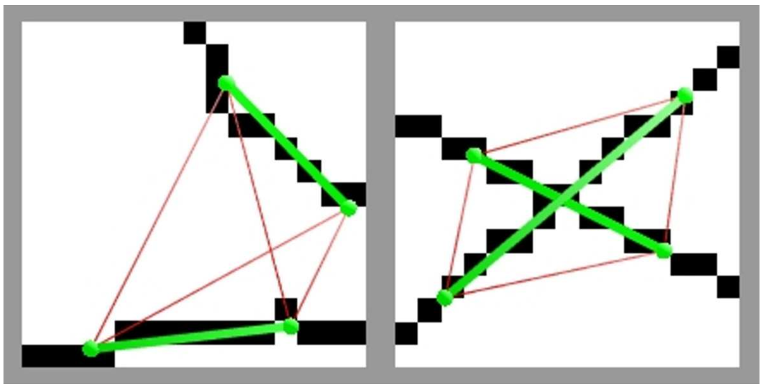

Figure 2.12: Examples of line pairs. A crossing is described by a line pair as shown in the right image.

A line pair either describes two disjunct lines which we will not interpret further, or a crossing of two lines, as shown in the right image of figure 2.12. These two cases can be distinguished quite easily by evaluating the intersection point of the two lines given by the extension of the line segments. If the intersection point lies on both line segments, then we found a crossing.

2.4.5 Translation Invariance of Structure Analysis

Using the analysis described in the previous sections, we can obtain for each local area in an image an interpretation of the area's structure, where we distinguish between corners and junctions. For every corner we obtain its location, its opening angle and its orientation. Junctions may be separated into y-junctions, t-junctions and crossings. For each of these we also obtain their location, orientation and angles.

The same structure found in one local area is also likely to be found in neighboring areas, whereby each time the structure has the same absolute position. This follows, since the method described here is translation and rotation invariant for one data set. In a real image, however, translating a local area will remove some edge points from the local area and others will appear.

Figure 2.13 and 2.14 show two examples of translation invariance of the structure analysis. In both cases a selection window is moved over an image region from left to right in steps of two pixels. Therefore, the present image structure translates from right to left relative to the window. In the image series in figure 2.13, a structure appears on the right hand side of the window and the intersection points and the support for the line segments change as long as new pixels enter the window or pixels leave the window. From image 4 to 7, no pixels are added or removed and the structure analysis always finds the same structure at the same position in terms of absolute image coordinates.

In figure 2.14 pixels enter and leave the window at each step. Nonetheless, the center of the junction is evaluated approximately at the same position in images 2 to 6, in terms of absolute image coordinates. In the other images a corner is detected instead of a junction. However, the center of the corner is more or less at the same position as the center of the junction.

These two examples show, that a particular corner or junction may not only be found at one particular test position. Instead, strong structures are likely to appear for a set of neighboring test positions. This offers the possibility of applying a clustering procedure to the corners and junctions found, in order to stabilize the output of the algorithm. However, this has so far not been implemented.

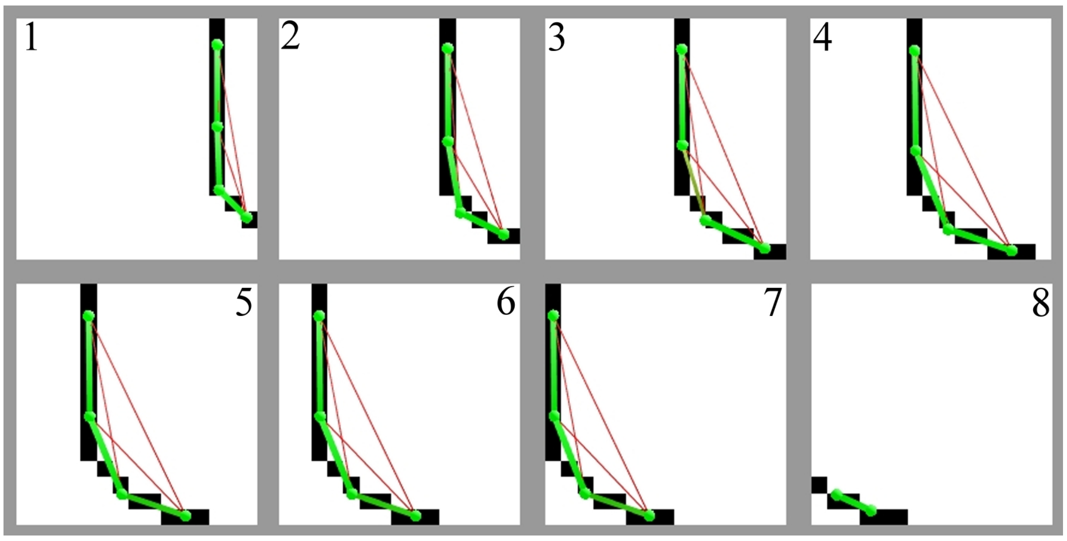

Figure 2.13: Example of translation invariance of structure analysis, if relative positioning of pixel data does not change.

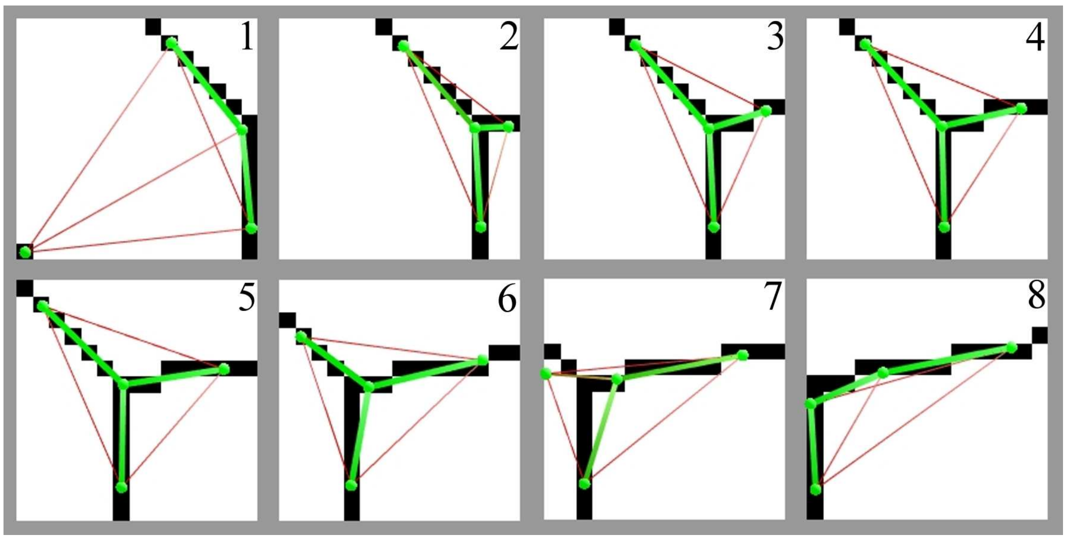

Figure 2.14: Example of approximate translation invariance of structure analysis, if pixels change but still describe the same structure.

Chapter 3

Experiments & Conclusions

In order to test the corner and junction detection quality of the algorithm, we applied it to the standard "blox" and laboratory example images from [6]. The extracted line segments were analyzed and only those constellations that could be regarded as corners or junctions were stored. Note that in this chapter we only present the results of the "blox" image. The results for the laboratory example image can be found in appendix A.

3.1 Experiments

Before the structure analysis algorithm can be applied, the edges of an image have to be extracted. This was done using the Canny edge detector [13]. The initial image and the result of the edge detection can be seen in figure 3.1. The algorithm was applied to this edge image, whereby a test window of 15×15 pixels was moved over the image in steps of two pixels. The factor λ from equation (2.17) was set to 0.1 and the factor τ from equation (2.19) to 0.2.

Recall that equation (2.24) gives a confidence value for a structure. This confidence can be used to measure the confidence we can have in a corner or junction found. The junctions found are shown in figure 3.2. Here the left image shows all junctions and the right image only those junctions with a confidence value of 0.90 or higher.

Both images in figure 3.3 show the corners found with an opening angle between 0 and 180 degrees. However, while the left image shows all corners, the right image shows only those with a confidence value of 0.99 or higher. The images in figure 3.4 show those corners with a confidence value of 0.99 or higher and an opening angle between 0 and 150 degrees, and 0 and 110 degrees, for the left and right image, respectively.

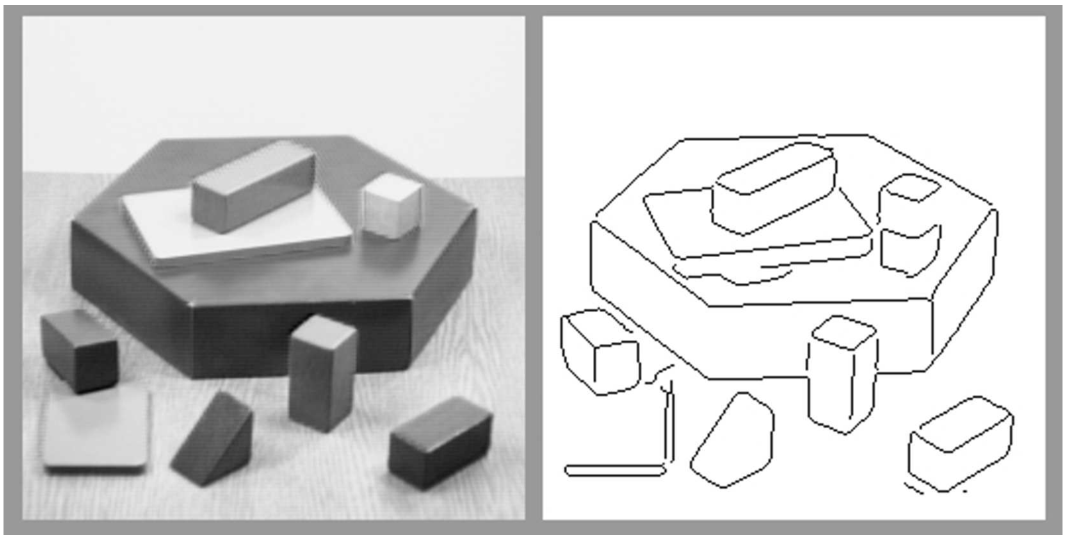
Figure 3.1: Example image "blox" (left), and the extracted edges (right).

From the images shown here it can be seen that the algorithm finds all important corners and also gives a good measure of their opening angle. Furthermore, almost all junctions were found. Junctions that were not detected have fairly large gaps in their contour with respect to the size of the test window. Three spurious junction were found. These false positives occurred at places where the gap between two separate structures became so small that they appear locally as one structure with some missing pixels. The problem that manifests itself here is, that within a small test window structures can only be interpreted locally. Global relationships are not taken into account which leads to false positives and false negatives.

The two main problems the algorithm faces are the following:

- Corners and junctions only become apparent at a particular scale. If the scale is chosen too small, many spurious corners may be found. If it chosen too large, too much structure may be present in a test window such that the algorithm fails.

- Edges may be incomplete. If there are only small gaps in the edges, the algorithm may still give good results. However, if the gaps become too large with respect to the test window, structures will not be detected correctly. Here the balance has to be found between bridging gaps and detecting corners and junction where there are none.

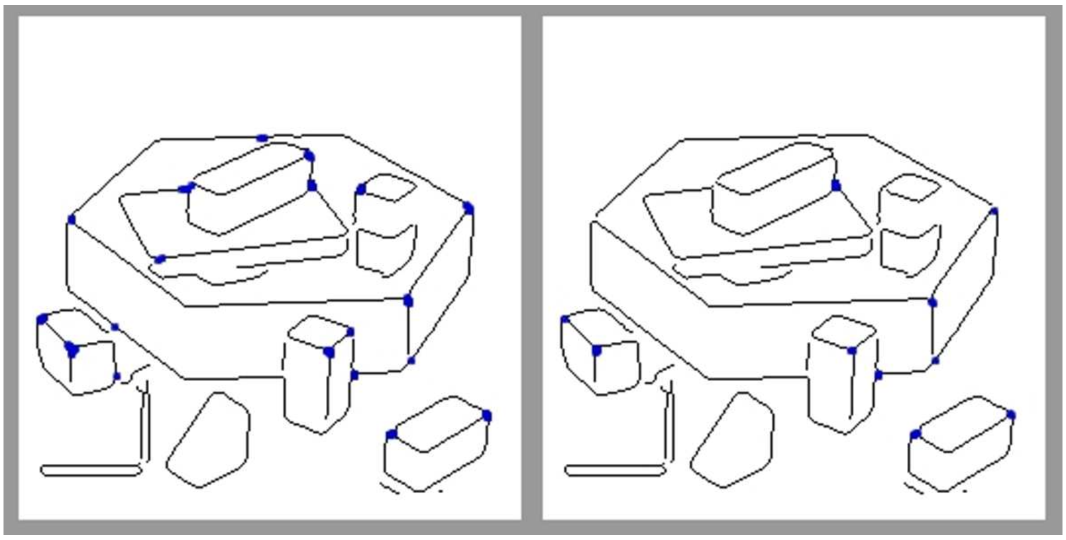

Figure 3.2: Detected junctions in example image "blox", with confidence value ≥ 0.50 (left) and ≥ 0.90 (right).

3.2 Conclusions

We have presented an algorithm that uses conics to analyze local image structure. The main idea is to fit intersections of conics to the data. It was found that these intersections can represent local image structures in a very useful way: the line segments that make up the local image structure lie between intersection point pairs. This basically reduces the search space of possible line segments to at most six specific ones. It was then shown that through additional analysis it is possible to extract corners and junctions from an image and to evaluate their parameters. A potential advantage of the presented algorithm over standard corner detectors is that it can distinguish between different types of i2D structures. Furthermore, it can be used to extract parameters of the local image structures, like the opening angle of a corner.

Acknowledgment

This work has been supported by DFG Graduiertenkolleg No. 357 and by EC Grant IST-2001-3422 (VISATEC).

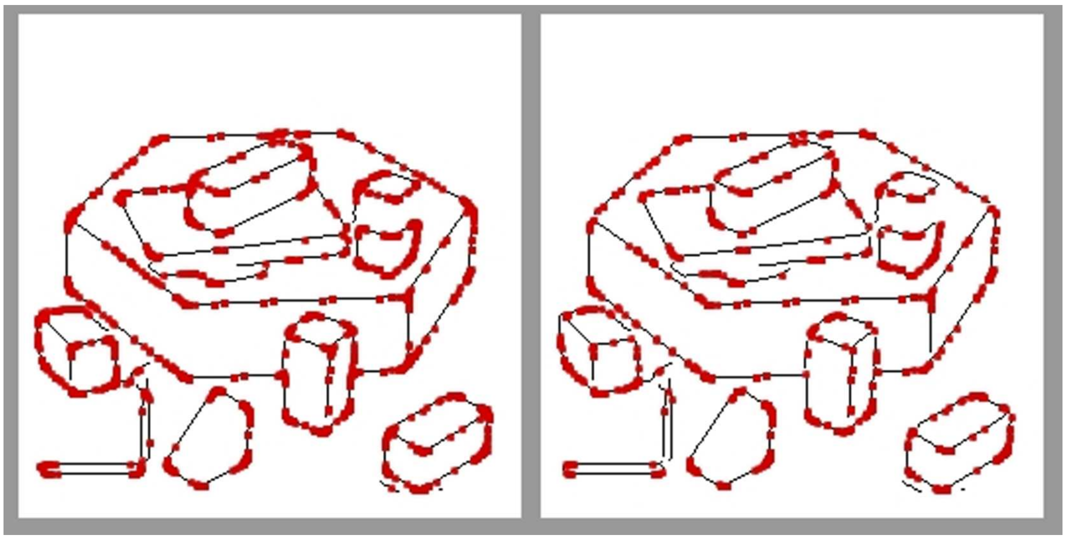

Figure 3.3: Detected corners in example image "blox", with opening angles between 0 and 180 degrees and confidence value ≥ 0.50 (left) and ≥ 0.90 (right).

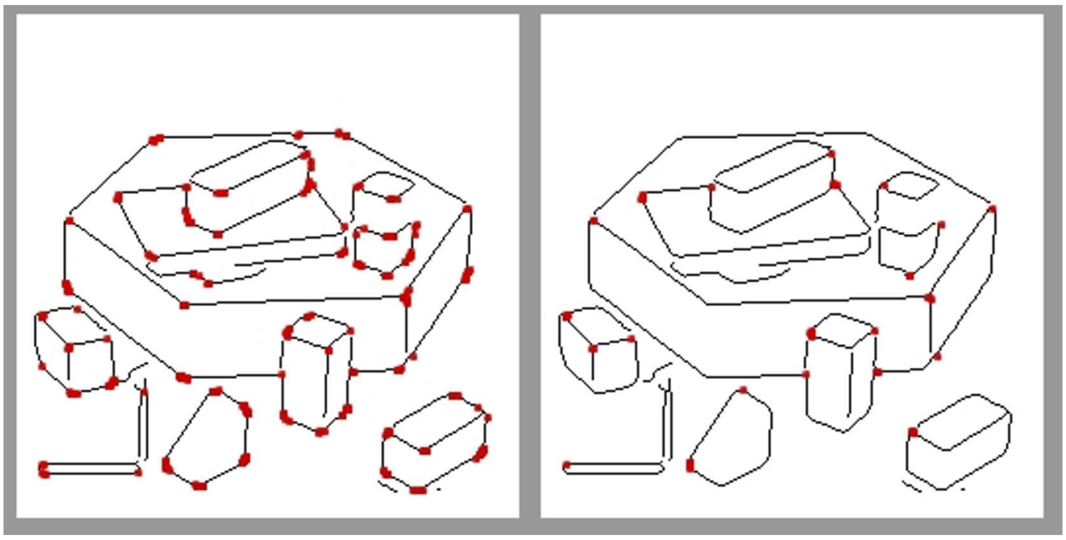

Figure 3.4: Detected corners in example image "blox", with confidence value ≥ 0.90 and opening angles between 0 and 150 degrees (left), and 0 and 110 degrees (right).

Appendix A

Laboratory Example Image

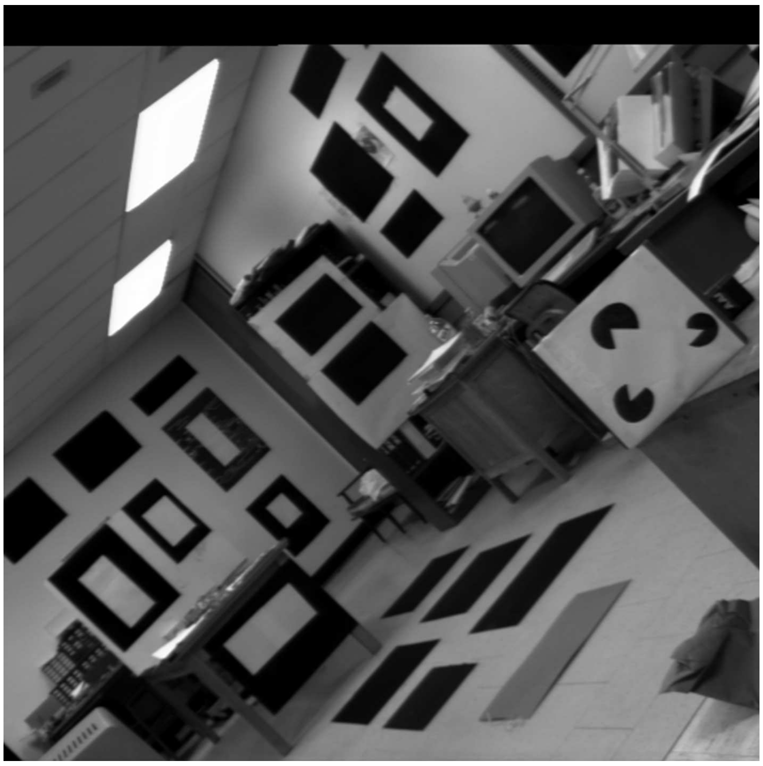

Figure A.1: Example image "lab".

In this appendix the results of applying the structure analysis algorithm to the image shown in figure A.1 are presented. The image in figure A.1 is 512×512 pixels. The result of applying the Canny edge detector to this image can be seen in figure A.2. The structure analysis algorithm was applied to this edge image. A test window of 15×15 pixels was moved over the edge image in steps of 2 pixels. The factor λ from equation (2.17) was set to 0.1 and the factor τ from equation (2.19) to 0.2.

The other images in this appendix show the location of the corners and junctions found for different confidence values and angle ranges. From equation (2.24) we can evaluate a confidence value for a structure. This is the confidence value that will be referred to in the following. Figure A.3 shows all corners found. That is, corners of all confidences and opening angles are drawn here. Figure A.4 shows all corners with a confidence value of 0.99 or above and an opening angle between 0 and 180 degrees. Figure A.5 shows all corners with the same confidence but an opening angle between 0 and 150 degrees. In figure A.6 all corners with the same confidence but an opening angle between 0 and 110 degrees are drawn.

Figure A.7 shows all junctions found, while figure A.8 shows those junctions with a confidence above 0.90.

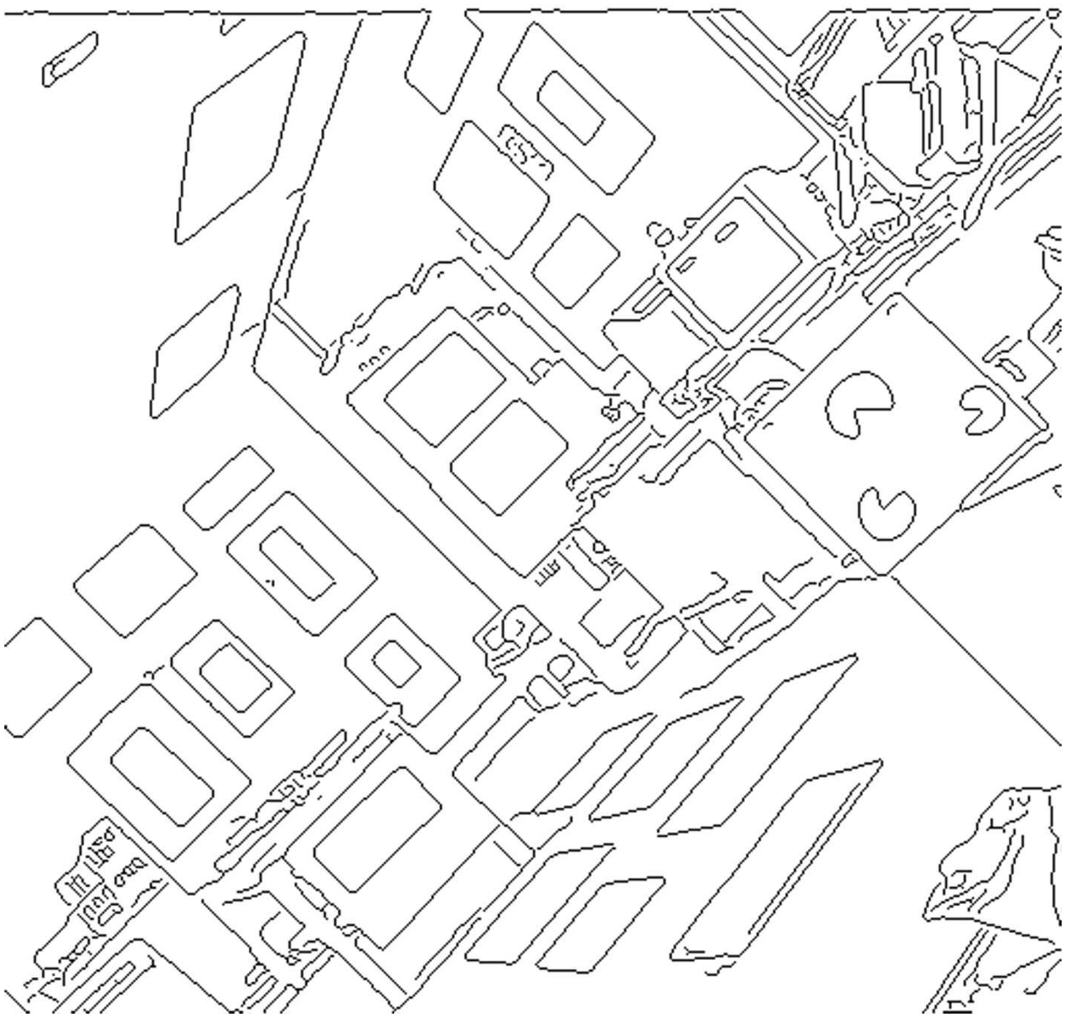

Figure A.2: Extracted edges of example image "lab".

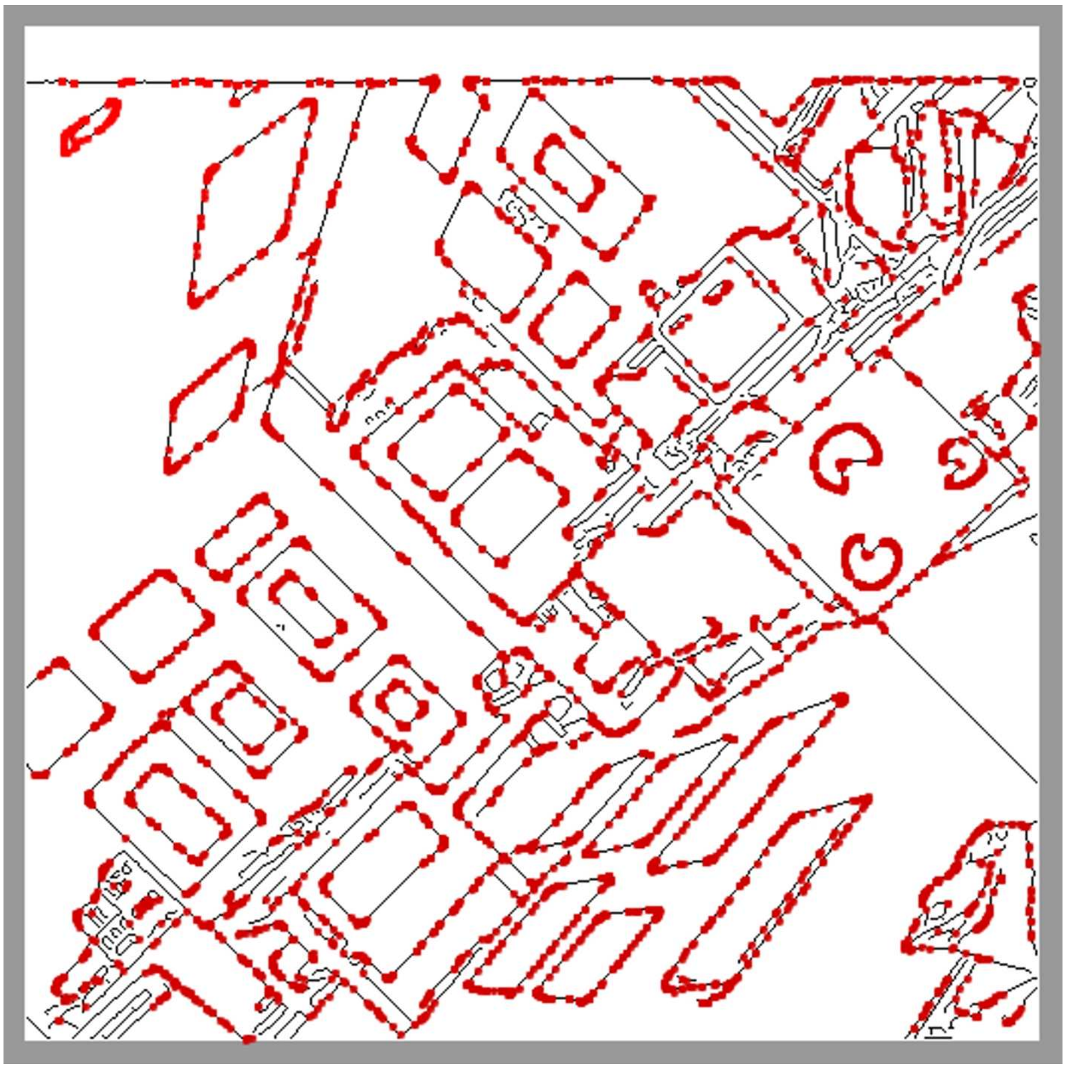

Figure A.3: All corners found in example image "lab", with confidence value ≥ 0.50 and opening angle between 0 and 180 degrees.

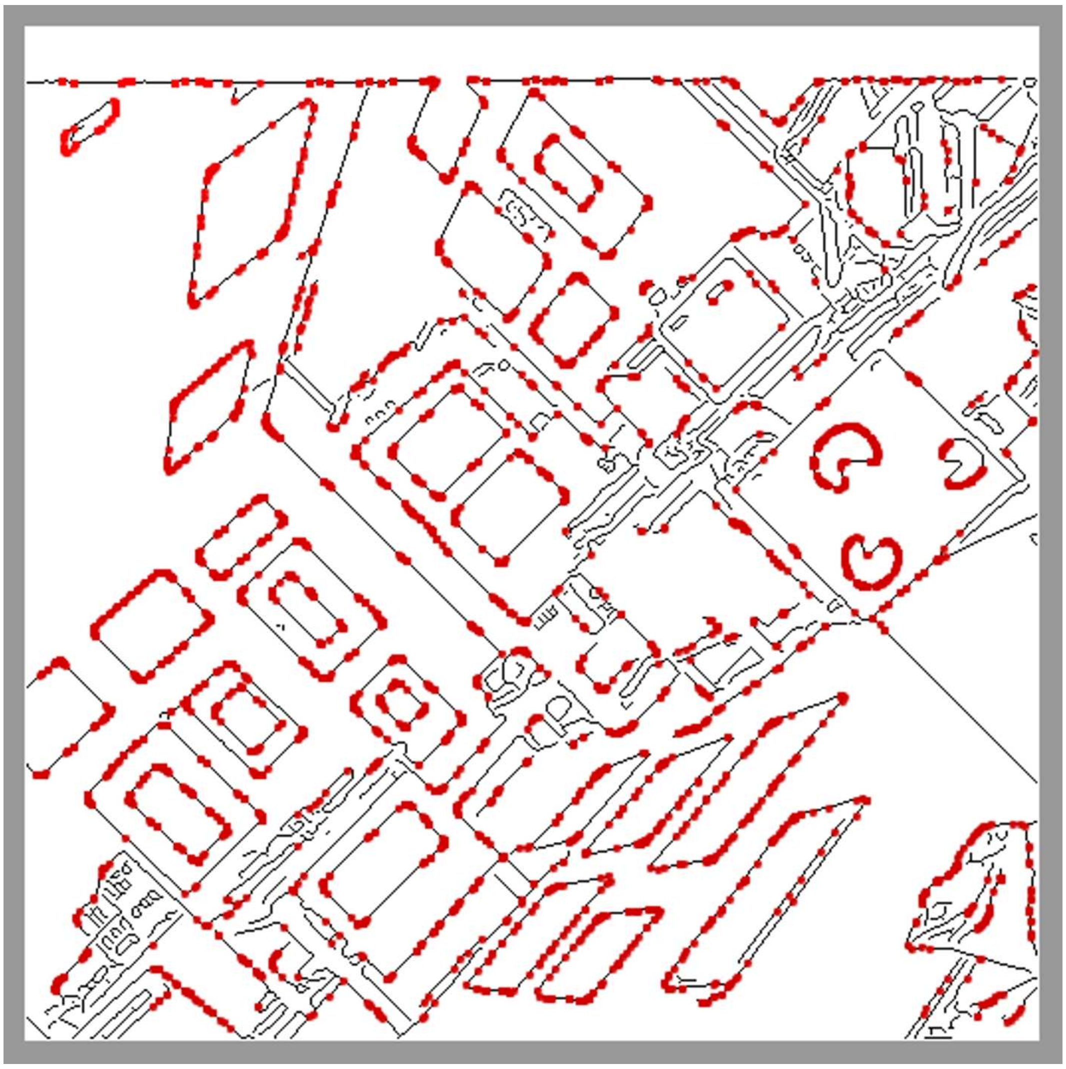

Figure A.4: All corners found in example image "lab", with confidence value ≥ 0.99 and opening angle between 0 and 180 degrees.

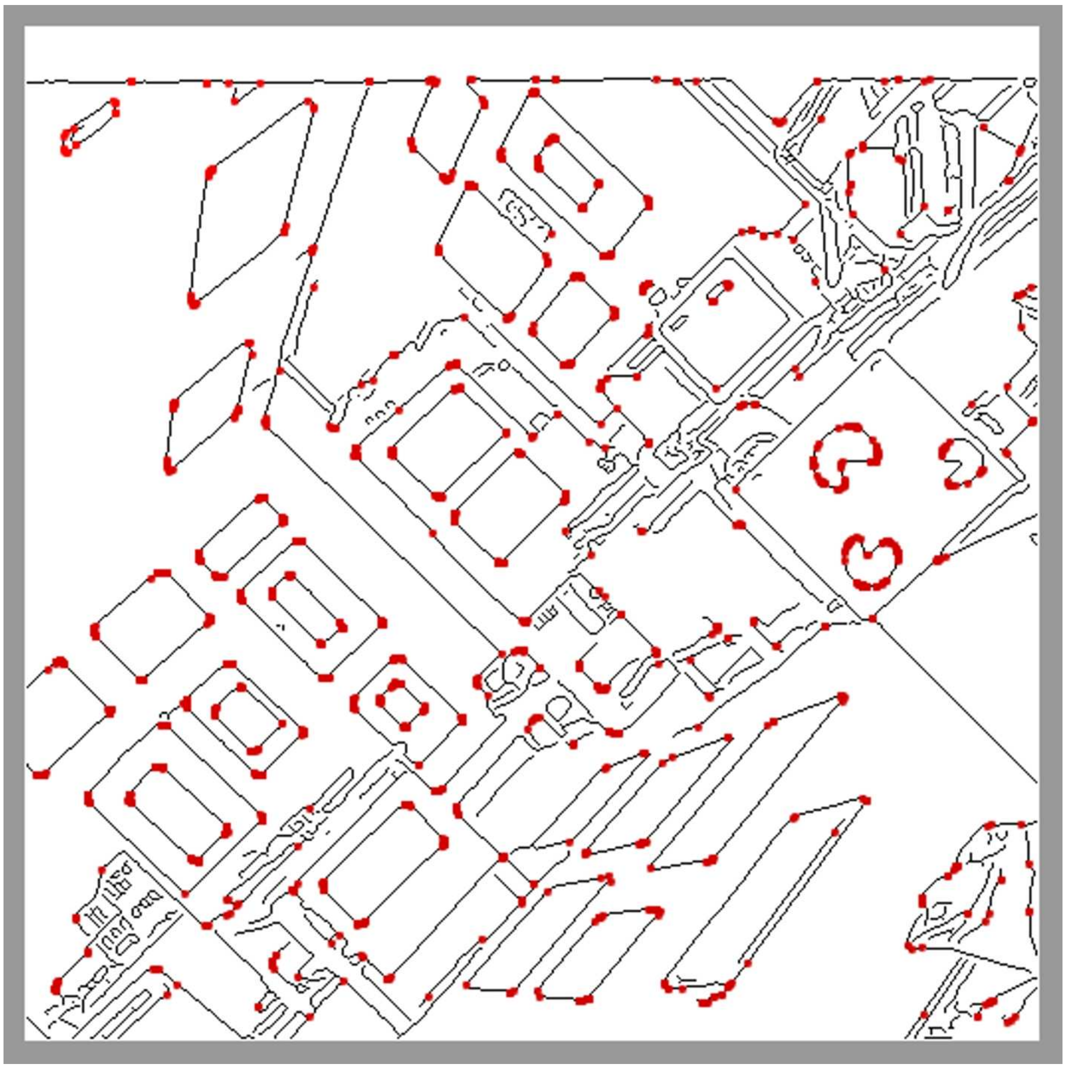

Figure A.5: All corners found in example image "lab", with confidence value ≥ 0.99 and opening angle between 0 and 150 degrees.

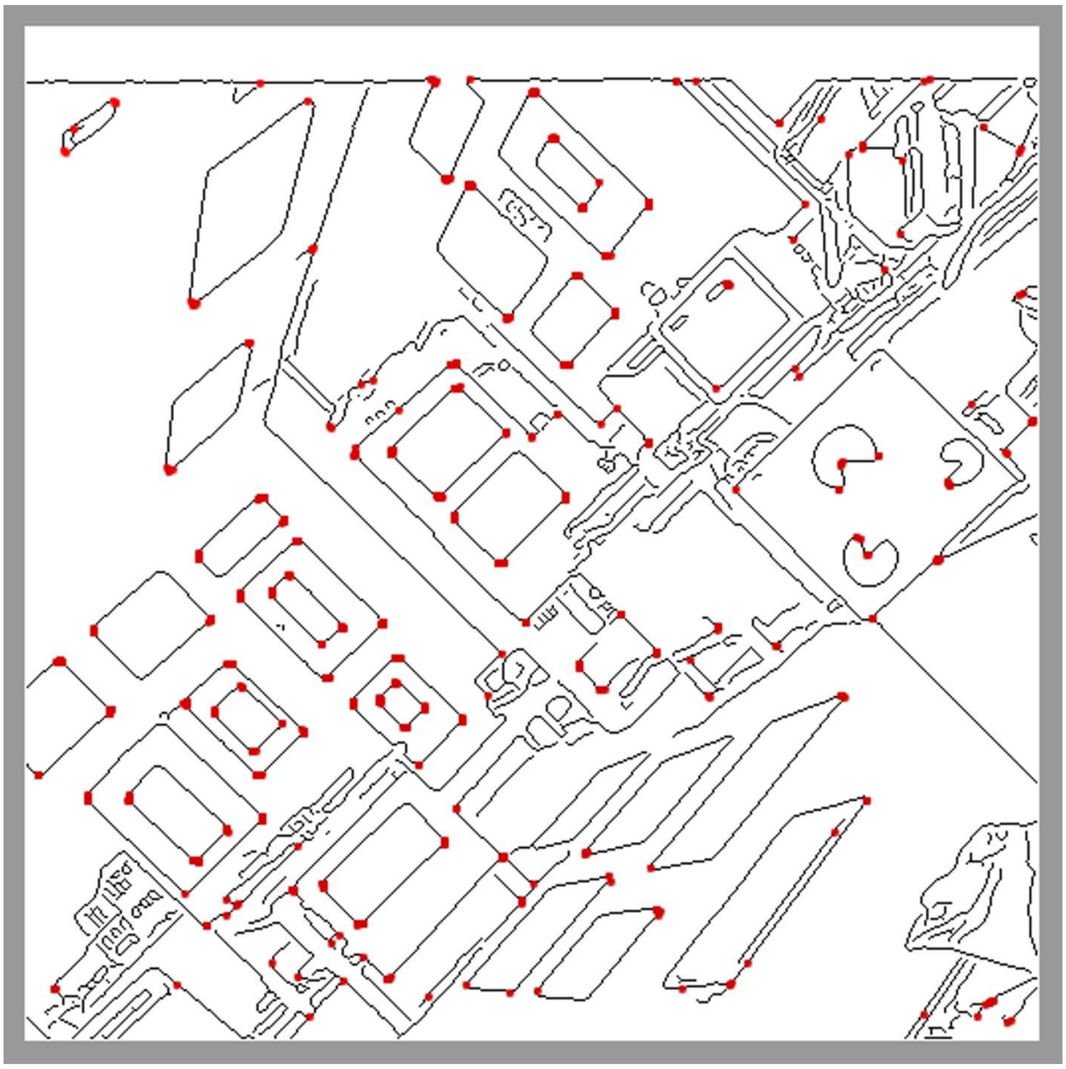
Figure A.6: All corners found in example image "lab", with confidence value ≥ 0.99 and opening angle between 0 and 110 degrees.

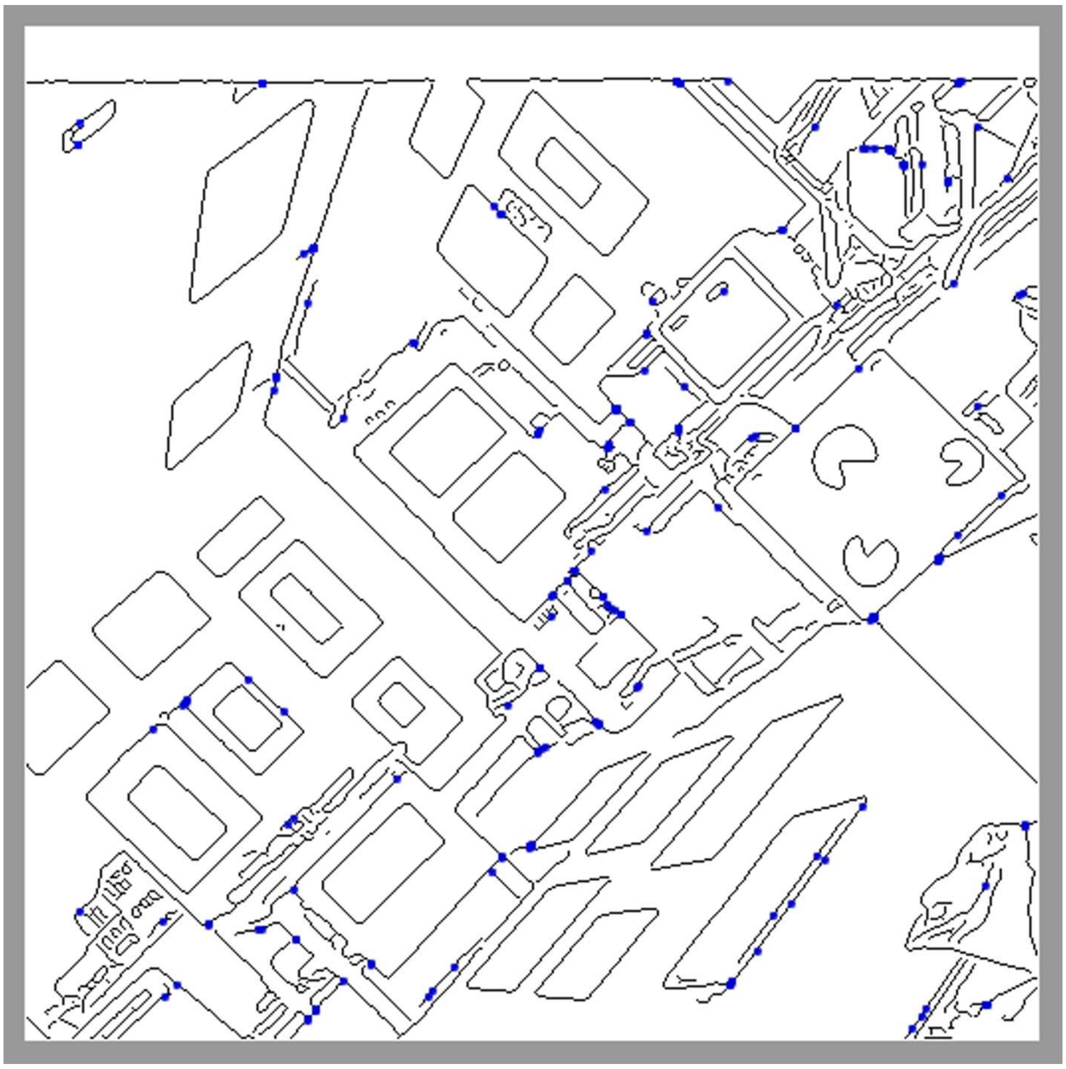

Figure A.7: All junctions found in example image "lab", with confidence value ≥ 0.50.

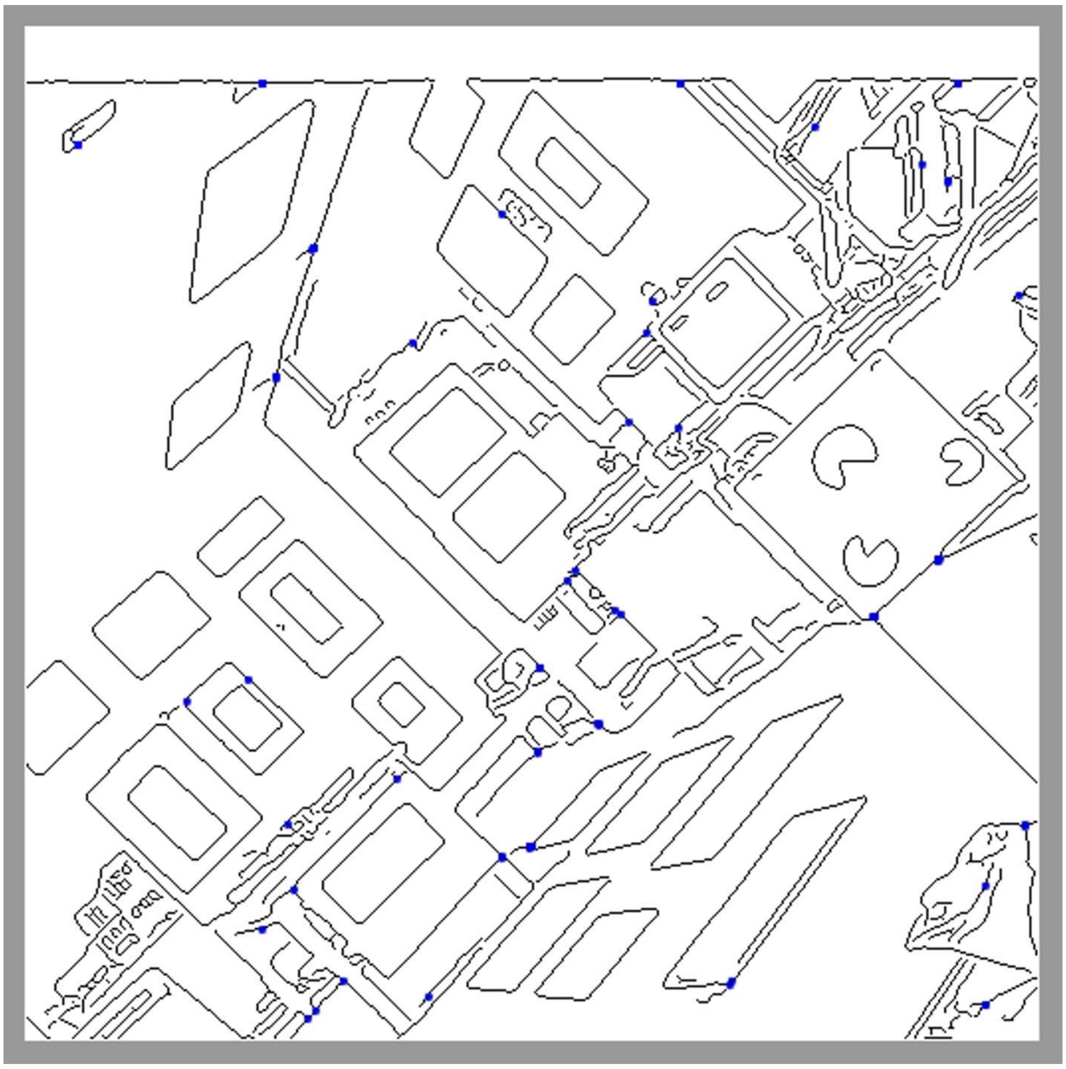

Figure A.8: All junctions found in example image "lab", with confidence value ≥ 0.90.

Appendix B

The Clifford Algebra Interpretation

In the following we discuss the use of Clifford algebra in the description of the intersections of conics. Note that to the best of our knowledge the Clifford algebra over the space of symmetric matrices (i.e. the vector space of conics) has not yet been discussed in the literature. We believe that it offers an intuitive way to deal with 2D-conic.

B.1 The Clifford Algebra over $\mathbb{D}^2$

Recall that we denote the 6D-vector space in which 2D-conics may be represented by $\mathbb{D}^2 \equiv \mathbb{R}^6$. The Clifford algebra over this vector space will be denoted by $\mathcal{C}\ell(\mathbb{D}^2)$. A 2D-vector $(x, y) \in \mathbb{R}^2$ is transformed to $\mathbb{D}^2$ by the function

$$\mathcal{D}: \quad (x, y) \in \mathbb{R}^2 \mapsto (x,\, y,\, \tfrac{1}{\sqrt{2}},\, \tfrac{1}{\sqrt{2}}\,x^2,\, \tfrac{1}{\sqrt{2}}\,y^2,\, xy) \in \mathbb{D}^2. \tag{B.1}$$

The Clifford Algebra $\mathcal{C}\ell(\mathbb{D}^2)$ has (algebra) dimension $2^6 = 64$. We define the *inner product null space* (IPNS) of a vector $\mathbf{A} \in \mathbb{D}^2$ to be the set of all those vectors $\mathbf{X} \in \mathbb{D}^2$ that satisfy $\mathbf{X} \cdot \mathbf{A} = 0$, whereby $\mathbf{X} = \mathcal{D}(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^2$. As was shown before, this null space is a (possibly degenerate) conic. Furthermore, the IPNS of the outer product of two vectors $\mathbf{A}, \mathbf{B} \in \mathbb{D}^2$, $\mathbf{A} \wedge \mathbf{B}$, now has to represent the intersection of the conics represented by $\mathbf{A}$ and $\mathbf{B}$. This can be seen quite easily, by using the algebraic rules for the inner and outer product.

$$\mathbf{X} \cdot (\mathbf{A} \wedge \mathbf{B}) = (\mathbf{X} \cdot \mathbf{A})\,\mathbf{B} - (\mathbf{X} \cdot \mathbf{B})\,\mathbf{A}. \tag{B.2}$$

The inner products $(\mathbf{X} \cdot \mathbf{A})$ and $(\mathbf{X} \cdot \mathbf{B})$ are zero if and only if $\mathbf{X}$ represents a point that lies on the conic represented by $\mathbf{A}$ and $\mathbf{B}$, respectively.

If $\mathbf{A}$ and $\mathbf{B}$ represent different conics, then they are linearly independent. Therefore, the expression $\mathbf{X} \cdot (\mathbf{A} \wedge \mathbf{B})$ can only be zero if and only if $\mathbf{X}$ lies on both conics. Hence, in terms of the IPNS, the *bivector* $\mathbf{A} \wedge \mathbf{B}$ represents the intersection of the conics represented by $\mathbf{A}$ and $\mathbf{B}$.

Let $\mathbf{A}_i \in \mathbb{D}^2$ be a set of linearly independent vectors that represent conics in $\mathbb{R}^2$. Then the **IPNS** of the outer products of these vectors in $\mathcal{C}\ell(\mathbb{D}^2)$ represent intersection of the respective conics.

$$\begin{aligned}
\mathbf{A}_1 &: \text{Conic} \\
\mathbf{A}_1 \wedge \mathbf{A}_2 &: \text{At most point quadruplet} \\
\mathbf{A}_1 \wedge \mathbf{A}_2 \wedge \mathbf{A}_3 &: \text{At most point triplet} \\
\mathbf{A}_1 \wedge \mathbf{A}_2 \wedge \mathbf{A}_3 \wedge \mathbf{A}_4 &: \text{At most point pair} \\
\mathbf{A}_1 \wedge \mathbf{A}_2 \wedge \mathbf{A}_3 \wedge \mathbf{A}_4 \wedge \mathbf{A}_5 &: \text{At most a point}
\end{aligned} \tag{B.3}$$

We can also define the *outer product null space* (OPNS) of a vector $\mathbf{A} \in \mathbb{D}^2$ as the set of vectors $\mathbf{X} = \mathcal{D}(\mathbf{x}) \in \mathbb{D}^2$, $\mathbf{x} \in \mathbb{R}^2$, that satisfy $\mathbf{X} \wedge \mathbf{A} = 0$. Let $\mathbf{x}_i \in \mathbb{R}^2$ denote a number of different points and let $\mathbf{X}_i \in \mathbb{D}^2$ be defined by $\mathbf{X}_i = \mathcal{D}(\mathbf{x}_i)\, \forall\, i$. Then the outer product null space of blades in $\mathcal{C}\ell(\mathbb{D}^2)$ may be shown to represent the following objects.

$$\begin{aligned}
\mathbf{X}_1 &: \text{Point } \mathbf{x}_1 \\
\mathbf{X}_1 \wedge \mathbf{X}_2 &: \text{Point pair } (\mathbf{x}_1,\, \mathbf{x}_2) \\
\mathbf{X}_1 \wedge \mathbf{X}_2 \wedge \mathbf{X}_3 &: \text{Point triplet } (\mathbf{x}_1,\, \mathbf{x}_2,\, \mathbf{x}_3) \\
\mathbf{X}_1 \wedge \mathbf{X}_2 \wedge \mathbf{X}_3 \wedge \mathbf{X}_4 &: \text{Point quadruplet } (\mathbf{x}_1,\, \mathbf{x}_2,\, \mathbf{x}_3, \mathbf{x}_4) \\
\mathbf{X}_1 \wedge \mathbf{X}_2 \wedge \mathbf{X}_3 \wedge \mathbf{X}_4 \wedge \mathbf{X}_5 &: \text{The conic through } \mathbf{x}_1,\, \mathbf{x}_2,\, \mathbf{x}_3,\, \mathbf{x}_4,\, \mathbf{x}_5.
\end{aligned} \tag{B.4}$$

In particular, it can be shown that the outer product null space of $\mathbf{X}_1 \wedge \mathbf{X}_2 \wedge \mathbf{X}_3 \wedge \mathbf{X}_4 \wedge \mathbf{X}_5$ is the same as the inner product null space of its dual, which is a vector. Hence, this is also a simple way to construct the symmetric matrix that represents a conic through five points.

Next we show how a Clifford algebra can be represented in a standard vector space. This will then allow us to fit also bivectors to data, in just the same way as vectors. From this the intersection of two conics that best fit some data points can be evaluated directly, instead of evaluating the best fitting conics and then intersecting them.

B.2 Representing $\mathcal{C}\ell(\mathbb{R}^n)$ in $\mathbb{R}^{2^n}$

A Clifford algebra $\mathcal{C}\ell_n$ over a vector space $\mathbb{R}^n$ has dimension 2^n. An algebraic basis of $\mathcal{C}\ell_n$ may therefore be denoted by a set $\{E_i\}_{i=1}^{2^n}$ of so called *basis blades*. It may be shown that these basis blades satisfy a number of constraints with respect to the *geometric* or *Clifford* product. This product will simply be denoted by juxtaposition, i.e. the geometric product of two elements $A, B \in \mathcal{C}\ell_n$ is written as AB. The basis blades of $\mathcal{C}\ell_n$ have the following properties:

1. There exists a unit element denoted by E_1 such that
$$E_i E_1 = E_1 E_i = E_i,\ \forall i \in \{1, \dots, 2^n\}. \tag{B.5}$$

2. The geometric product of a basis blades with itself results in the unit element, i.e.
$$E_i E_i = \lambda_i\, E_1,\ \lambda_i \in \{-1, 1\},\ \forall i \in \{1, \dots, 2^n\}. \tag{B.6}$$

3. The geometric product of any two basis blades gives a third basis blade times a scalar factor. More precisely,
$$E_i E_j = \sum_{k=1}^{2^n} g^k{}_{ij}\, E_k,\ \forall i, j \in \{1, \dots, 2^n\}, \tag{B.7}$$
where $g^k{}_{ij} \in \{-1, 0, 1\}$ and is only non-zero for exactly one value of k given a pair i and j.

The last condition basically says that the geometric product is invertible. For example, given indices (i, j, k) such that $E_i E_j = E_k$, we find that
$$E_i E_j = E_k \iff E_i E_j E_j = E_k E_j \iff E_k E_j = \lambda_j\, E_i,$$
and thus $g^i{}_{kj} = \lambda_j$.

A general element of $\mathcal{C}\ell_n$ is called *multivector*. In terms of basis blades a general multivector $A \in \mathcal{C}\ell_n$ may be given by
$$A = \sum_{i=1}^{2^n} \alpha^i\, E_i. \tag{B.8}$$

In the following we will use the Einstein summation convention, that a superscript index repeated within a product as a subscript index is implicitly summed over its range. That is, equation (B.8) may be written as
$$A = \alpha^i\, E_i,$$
if it is clear that $i \in \{1, \dots, 2^n\}$. The geometric product of two multivectors $A, B \in \mathcal{C}\ell_n$, with $A = \alpha^i E_i$ and $B = \beta^i E_i$, is then given by
$$\begin{aligned} AB = &\ (\alpha^i\, E_i)\,(\beta^j\, E_j) \\ = &\ \alpha^i\, \beta^j\, E_i E_j \\ = &\ \alpha^i\, \beta^j\, g^k{}_{ij}\, E_k. \end{aligned} \tag{B.9}$$

Writing the result multivector $M \in \mathcal{C}\ell_n$ of $M = AB$ as $M = \mu^i E_i$ then gives

$$M = AB \iff \mu^k E_k = \alpha^i \, \beta^j \, g^k{}_{ij} \; E_k \iff \mu^k = \alpha^i \, \beta^j \, g^k{}_{ij}. \tag{B.10}$$

This shows that if multivectors are expressed as vectors in $\mathbb{R}^{2^n}$, the geometric product between them becomes a bilinear function. Note that the other products available in Clifford algebra like the inner and outer product, and the commutator and anti-commutator product may also be expressed in this way.

B.3 An Isomorphism

If we use again $\{E_i\}_{i=1}^{2^n}$ to denote the algebra basis of $\mathcal{C}\ell(\mathbb{R}^n)$ and write the three multivectors $A, B, M \in \mathcal{C}\ell(\mathbb{R}^n)$ as $A = \alpha^i E_i$, $B = \beta^i E_i$ and $M = \mu^i E_i$, we may regard them as vectors in some $\mathbb{R}^m$, with orthonormal basis $\{\mathsf{e}_i\}_{i=1}^m$, where $m = 2^n$. In this vector space the multivectors may be written as column vectors $\mathsf{a} = [\alpha^1, \dots, \alpha^m]^\mathsf{T}$, $\mathsf{b} = [\beta^1, \dots, \beta^m]^\mathsf{T}$ and $\mathsf{m} = [\mu^1, \dots, \mu^m]^\mathsf{T}$, respectively. We use here sans serif letters to denote vectors in $\mathbb{R}^m$ in order to distinguish them from (multi-)vectors in $\mathcal{C}\ell(\mathbb{R}^n)$. The relation between multivectors in $\mathcal{C}\ell(\mathbb{R}^n)$ and their representation in $\mathbb{R}^m$ may be regarded as an isomorphism ϕ between these two spaces, whereby $\phi(A \in \mathcal{C}\ell_n) = \mathsf{a} \in \mathbb{R}^m$ and $\phi^{-1}(\mathsf{a}) = A$. This isomorphism also transforms Clifford algebra products to matrix products with special matrices. For example, if $M = A \wedge B$ then

$$\mathsf{m} = \phi(M) = \phi(A \wedge B) = \mathsf{U}(\phi(A))\, \phi(B) = \mathsf{U}(\mathsf{a})\, \mathsf{b},$$

where $\mathsf{U}(\mathsf{a})$ is a matrix whose entries depend on a. In the following all matrices will be written as capital sans-serif letters. The form of matrix U is derived through the following considerations. A product in $\mathcal{C}\ell(\mathbb{R}^n)$ between two multivectors can be expressed as a bilinear function g which is a map $\mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ and may be written as

$$\mathsf{g}(\mathsf{a},\, \mathsf{b}) := \alpha^i \, \beta^j \, g^k{}_{ij} \; \mathsf{e}_k, \tag{B.11}$$

where we have implicit sums over i, j and k. The object $g^k{}_{ij}$ is again the 3-valence tensor from equation (B.7). It encodes the relation between the basis blades of $\mathcal{C}\ell_n$ for a particular product. For example, if $g^k{}_{ij}$ encodes the outer product, then the equation $M = A \wedge B$ may be written in $\mathbb{R}^m$ as

$$\mathsf{m} = \mathsf{g}(\mathsf{a},\, \mathsf{b}) \iff \mu^k = \alpha^i \beta^j \, g^k{}_{ij} \; \forall k. \tag{B.12}$$

If we now denote the matrix of derivatives of $\mathsf{g}(\mathsf{a}, \mathsf{b})$ with respect to the $\{\beta^j\}$ as $\mathsf{U}(\mathsf{a})$, and with respect to the $\{\alpha^i\}$ as $\mathsf{V}(\mathsf{b})$, we can write $M = A \wedge B$ equivalently in $\mathbb{R}^m$ as

$$\mathsf{m} = \mathsf{U}(\mathsf{a})\, \mathsf{b} = \mathsf{V}(\mathsf{b})\, \mathsf{a}. \tag{B.13}$$

That is, U and V are the Jacobi matrices of g.

B.4 Solving for a Multivector

In section 2.2 we were given a set of 2D-points $\mathbf{x}_i \in \mathbb{R}^2$ embedded in $\mathbb{D}^2$ as $\mathbf{X}_i := \mathcal{D}(\mathbf{x}_i)$, where N is the number of data vectors. What we were then looking for was the vector $\mathbf{A} \in \mathbb{D}^2$ that minimizes

$$\sum_{i=1}^{N} |\mathbf{X}_i \cdot \mathbf{A}|^2. \tag{B.14}$$

Let $\mathsf{a} := \phi(\mathbf{A})$ and $\mathsf{x}_i := \phi(\mathbf{X}_i)$, then

$$\phi(\mathbf{X}_i \cdot \mathbf{A}) = \mathsf{U}(\mathsf{x}_i)\,\mathsf{a}, \tag{B.15}$$

where $\mathsf{U}(\mathsf{x}_i)$ encodes the inner product. Since the inner product of two vectors is identical to their scalar product, this simply becomes

$$\phi(\mathbf{X}_i \cdot \mathbf{A}) = \mathsf{U}(\mathsf{x}_i)\,\mathsf{a} = \mathsf{x}_i^{\mathsf{T}}\,\mathsf{a}. \tag{B.16}$$

Now we construct a matrix W_1 from the $\{\mathsf{x}_i\}$ as

$$\mathsf{W}_1 = \begin{pmatrix} \mathsf{x}_1^{\mathsf{T}} \\ \vdots \\ \mathsf{x}_N^{\mathsf{T}} \end{pmatrix}. \tag{B.17}$$

Clearly, the solution vector a now has to lie in the null space of W_1, i.e. $\mathsf{W}_1\,\mathsf{a} = 0$. If we evaluate the SVD of W_1, we obtain the basis of the null space and the range of W_1. Note that the singular vectors found are the eigenvectors of $\mathsf{W}_1^{\mathsf{T}}\,\mathsf{W}_1$. Hence, the singular vector with the smallest singular value is the best fit to the $\{\mathsf{x}_i\}$ in a least squares sense.

This method can be easily extended to any elements of the Clifford algebra $\mathcal{C}\ell(\mathbb{D}^2)$. For example, if instead of a vector $\mathbf{A} \in \mathbb{D}^2$ we are looking for a bivector $\mathbf{A} \wedge \mathbf{B} \in \mathcal{C}\ell(\mathbb{D}^2)$ with $\mathbf{B} \in \mathbb{D}^2$, that best fits the data, we can minimize the following expression

$$\sum_{i=1}^{N} \|\mathbf{X}_i \cdot (\mathbf{A} \wedge \mathbf{B})\|_2^2. \tag{B.18}$$

Let $\mathsf{p} = \phi(\mathbf{A} \wedge \mathbf{B})$, then

$$\phi(\mathbf{X}_i \cdot (\mathbf{A} \wedge \mathbf{B})) = \mathsf{U}(\mathsf{x}_i)\,\mathsf{p}, \tag{B.19}$$

where $\mathsf{U}(\mathsf{x}_i)$ again encodes the inner product. However, since the components of p relate to a bivector and the inner product of a bivector with a vector results in a vector, U is this time a 6×6 matrix, whose components depend on p. Nevertheless, we may still construct a matrix W_2 as

$$\mathsf{W}_2 = \begin{pmatrix} \mathsf{U}(\mathsf{x}_1) \\ \vdots \\ \mathsf{U}(\mathsf{x}_N) \end{pmatrix}. \tag{B.20}$$

The null space of W_2 now gives the best *bivector* that fits the data in a least squares sense.

Suppose that the matrix W_1 constructed from the given data $\{\mathbf{X}_i\}$ has a 2D-null space, whose orthonormal basis is given by $\mathbf{A}, \mathbf{B} \in \mathbb{D}^2$. This means that also any linear combination of $\mathbf{A}$ and $\mathbf{B}$ lies in the null space of W_1. Let $\mathbf{U}_1 := \alpha_1\,\mathbf{A} + \beta_1\,\mathbf{B}$ and $\mathbf{U}_2 := \alpha_2\,\mathbf{A} + \beta_2\,\mathbf{B}$. Then

$$\mathbf{X}_i \cdot \mathbf{U}_1 = \mathbf{X}_i \cdot \mathbf{U}_2 = 0\ \forall\, i.$$

If $\mathbf{A}$ and $\mathbf{B}$ represent two conics, then also $\mathbf{U}_1$ and $\mathbf{U}_2$ represent conics. We will now show that all linear combinations of $\mathbf{A}$ and $\mathbf{B}$ intersect in the same points. For this purpose consider the outer product of $\mathbf{U}_1$ and $\mathbf{U}_2$.

$$\mathbf{U}_1 \wedge \mathbf{U}_2 = (\alpha_1\,\beta_2 - \alpha_2\,\beta_1)\ (\mathbf{A} \wedge \mathbf{B}). \tag{B.21}$$

We have already shown that the outer product of two vectors in $\mathbb{D}^2$ represents the intersection of the two objects represented by the vectors. Since an overall scalar factor does not change the object represented by a vector, this equation shows that the intersection of $\mathbf{U}_1$ and $\mathbf{U}_2$ represents the same object as the intersection of $\mathbf{A}$ and $\mathbf{B}$.

From this analysis it follows that the null space of W_2 represents the intersection of the objects represented by the basis of the null space of W_1.

Appendix C

Evaluating the Intersection of Conics

The problem we discuss here in some detail is how to evaluate the intersection of two conics. Recall that a projective 2D-conic may be represented by a symmetric 3×3 matrix A. All vectors $(x, y)^{\mathsf{T}} \in \mathbb{E}^2$ that satisfy

$$(x, y, 1)\ \mathsf{A} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0, \tag{C.1}$$

lie on the conic represented by A. In section 2.1 we defined a 6D-vector space $\mathbb{D}^2$ in which projective conics may be represented in an equivalent way. In order to achieve this we defined two transformations $\mathcal{T} : \mathbb{R}^{3\times 3} \rightarrow \mathbb{D}^2$ and $\mathcal{D} : \mathbb{E}^2 \rightarrow \mathbb{D}^2$ as

$$\mathcal{T}\ :\ \mathsf{A} \in \mathbb{R}^{3\times 3} \mapsto (a_{13},\, a_{23},\, \tfrac{1}{\sqrt{2}}\, a_{33},\, \tfrac{1}{\sqrt{2}}\, a_{11},\, \tfrac{1}{\sqrt{2}}\, a_{22},\, a_{12})^{\mathsf{T}} \in \mathbb{D}^2, \tag{C.2}$$

and

$$\mathcal{D}\ :\quad (x, y) \in \mathbb{E}^2\ \mapsto\ (x,\, y,\, \tfrac{1}{\sqrt{2}},\, \tfrac{1}{\sqrt{2}}\, x^2,\, \tfrac{1}{\sqrt{2}}\, y^2,\, xy) \in \mathbb{D}^2. \tag{C.3}$$

A vector $\mathbf{x} \in \mathbb{E}^2$ lies on a conic represented by A if $\mathcal{D}(\mathbf{x}) \cdot \mathcal{T}(\mathsf{A}) = 0$, where $'\cdot'$ represents the scalar product in $\mathbb{D}^2$.

Suppose now we are given two conics represented by $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{D}^2$ and would like to find their intersection. Note that two conics can intersect in at most four points. We will denote the space of intersection points by $\mathbb{S}$, which may be defined as

$$\mathbb{S} := \{\, \mathbf{X} \in \mathbb{D}^2\ :\ \mathbf{C}_1 \cdot \mathbf{X} = 0,\, \mathbf{C}_2 \cdot \mathbf{X} = 0 \,\}. \tag{C.4}$$

Note that $\mathbb{S}$ is in fact a vector space, since any linear combination of elements of $\mathbb{S}$ lies again in $\mathbb{S}$. That is, if $\mathbf{X}, \mathbf{Y} \in \mathbb{S}$, then $\mathbf{C}_1 \cdot (\alpha\mathbf{X} + \beta\mathbf{Y}) = 0$,

$\forall\, \alpha, \beta \in \mathbb{R}$. It is also useful to define the set of Euclidean points in $\mathbb{E}^2$ that, when embedded in $\mathbb{D}^2$, lie in $\mathbb{S}$. We will denote this set by $\mathbb{S}_E$. It is defined as

$$\mathbb{S}_E := \{\ \mathbf{x} \in \mathbb{R}^2\ :\ \mathbf{C}_1 \cdot \mathcal{D}(\mathbf{x}) = 0,\ \mathbf{C}_2 \cdot \mathcal{D}(\mathbf{x}) = 0\ \}. \tag{C.5}$$

The expressions $\mathbf{C}_1 \cdot \mathbf{X}$ and $\mathbf{C}_2 \cdot \mathbf{X}$ are both polynomials of order two in the components of $\mathbf{X}$. Combining both expressions it is possible to obtain a polynomial of order four whose roots are the intersection points of the two conics. The roots of a polynomial of order four can be found by finding the roots of a polynomial of order three and one of order two (see e.g. [14]). However, note that we have two coupled polynomials of order four, so the polynomial components will be rather complex. In any case, see [15] for a discussion of this type of method.

We were looking for a method of evaluating the intersection of two conics that uses standard matrix methods which can be applied directly to the matrices representing the conics. In the following we will present this method. To the best of our knowledge this is a novel method for the evaluation of the intersection of two conics.

C.1 Finding Degenerate Conics

We start with a short lemma.

Lemma C.1 *Let* $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{D}^2$ *be two linearly independent vectors representing two conics that intersect in four points. That is, their Euclidean intersection set* $\mathbb{S}_E$*, as defined above, contains four elements and the intersection space* $\mathbb{S}$ *has dimension four. A conic* $\mathbf{C} \in \mathbb{D}^2$ *passes through these four intersection points if and only if it is a linear combination of* $\mathbf{C}_1$ *and* $\mathbf{C}_2$*. This is not necessarily the case if* $|\mathbb{S}_E| < 4$.

Proof. First of all, if $\mathbf{C}$ is a linear combination of $\mathbf{C}_1$ and $\mathbf{C}_2$, i.e. $\mathbf{C} = \alpha\,\mathbf{C}_1 + \beta\,\mathbf{C}_2$, $\alpha, \beta \in \mathbb{R}$, then for any $\mathbf{X} \in \mathbb{S}$ we have

$$\mathbf{C} \cdot \mathbf{X} = (\alpha\,\mathbf{C}_1 + \beta\,\mathbf{C}_2) \cdot \mathbf{X} = \alpha\,(\mathbf{C}_1 \cdot \mathbf{X}) + \beta\,(\mathbf{C}_2 \cdot \mathbf{X}) = 0. \tag{C.6}$$

Now for the other direction. That is, if $\mathbf{C} \cdot \mathbf{X} = 0$, we have to show that $\mathbf{C}$ is a linear combination of $\mathbf{C}_1$ and $\mathbf{C}_2$. We will do this via a dimensional argument. Recall that $\mathbb{D}^2$ is a 6-dimensional vector space. Since $\mathbf{C}_1$ and $\mathbf{C}_2$ are linearly independent they span a 2D-subspace of $\mathbb{D}^2$. If $\mathbf{C}_1$ and $\mathbf{C}_2$ intersect in four points then $\dim(\mathbb{S}) = 4$, $\mathbb{S} \subset \mathbb{D}^2$ and $\mathrm{span}\{\mathbf{C}_1,\, \mathbf{C}_2\} \perp \mathbb{S}$. Since $\mathbf{C} \perp \mathbb{S}$, $\mathbf{C}$ has to lie in $\mathrm{span}\{\mathbf{C}_1,\, \mathbf{C}_2\}$. Hence, $\mathbf{C}$ has to be a linear combination of $\mathbf{C}_1$ and $\mathbf{C}_2$.

If $\dim(\mathbb{S}) < 4$ this argument does not hold anymore. That is, there do exist conics that pass through the same intersection points as $\mathbf{C}_1$ and $\mathbf{C}_2$, but cannot be written as a linear combination of $\mathbf{C}_1$ and $\mathbf{C}_2$. □

This lemma gives the motivation for the following idea. If two conics $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{D}^2$ intersect in four points, then all conics that pass through these four points can be represented as a linear combination of $\mathbf{C}_1$ and $\mathbf{C}_2$. This also has to include degenerate conics, in particular those representing line pairs (2D-cones). Given four points there are three unique line pairs that contain these four points. If we are able to find the particular linear combinations of $\mathbf{C}_1$ and $\mathbf{C}_2$ that generate these degenerate conics, we can reduce the evaluation of the intersection of two conics to the intersection of line pairs. As we will see later, if the two conics only intersect in two or three points, at least one degenerate conic can still be found and we can then find the intersection points by intersecting a conics with a degenerate one.

Lemma C.2 *Let $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{D}^2$ denote two conics and let $\mathsf{A} = \mathcal{T}^{-1}(\mathbf{C}_1)$ and $\mathsf{B} = \mathcal{T}^{-1}(\mathbf{C}_2)$ be their matrix representations. The 3×3 matrices A and B are of full rank if the conics are non-degenerate. Let B be of full rank, then $\mathsf{M} := \mathsf{B}^{-1}\,\mathsf{A}$ exists. If λ is a real eigenvalue of M, then $\mathsf{A} - \lambda\,\mathsf{B}$ represents a degenerate conic.*

Proof. Let $\mathsf{C} = \alpha\,\mathsf{A} + \beta\,\mathsf{B}$, $\alpha, \beta \in \mathbb{R}$. C represents a degenerate conic if and only if $\det(\mathsf{C}) = 0$. We have to find those α and β for which this is the case.

$$\begin{aligned} & \det(\alpha\,\mathsf{A} + \beta\,\mathsf{B}) = 0 \\ \iff\quad & \alpha^3\ \det\left(\mathsf{B}\,\mathsf{B}^{-1}\ (\mathsf{A} + \tfrac{\beta}{\alpha}\,\mathsf{B})\right) = 0 \\ \iff\quad & \det(\mathsf{B})\ \det(\mathsf{B}^{-1}\mathsf{A} + \tfrac{\beta}{\alpha}\,\mathsf{I}) = 0 \\ \iff\quad & \det(\mathsf{M} - \lambda\,\mathsf{I}) = 0, \end{aligned} \tag{C.7}$$

where $\mathsf{M} := \mathsf{B}^{-1}\mathsf{A}$ and $\lambda = -\beta/\alpha$. The values of λ that satisfy the last equation are just the eigenvalues of M. If λ is a real eigenvalue of M, then $\mathsf{C} = \mathsf{A} - \lambda\,\mathsf{B}$ represents a real, degenerate conic. □

Clearly, the degenerate conics found in this way pass through the intersection points of $\mathbf{C}_1$ and $\mathbf{C}_2$, independently of how many intersection points these conics have.

Corollary C.1 *Let $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{D}^2$ be two non-degenerate conics. Then there exists at least one linear combination of $\mathbf{C}_1$ and $\mathbf{C}_2$ which represents a degenerate conic.*

Proof. Let $\mathsf{A} = \mathcal{T}^{-1}(\mathbf{C}_1)$ and $\mathsf{B} = \mathcal{T}^{-1}(\mathbf{C}_2)$ be the matrix representations of $\mathbf{C}_1$ and $\mathbf{C}_2$. Since the conics are non-degenerate, A and B are of full rank. Hence, also $\mathsf{M} = \mathsf{B}^{-1}\,\mathsf{A}$ has to be of full rank. Therefore, M has three non-zero eigenvalues. Furthermore, since complex eigenvalues of real matrices always appear in conjugate pairs, M must always have at least one real, non-zero eigenvalue. It thus follows from lemma C.2 that there always exists a linear combination of $\mathbf{C}_1$ and $\mathbf{C}_2$ that represents a degenerate conic. $\square$

We can follow from corollary C.1 that if two conics intersect in at least two points, the degenerate conic will have to represent a line pair, or at least a line. Intersecting a line with a conic is quite simple, since this comes down to finding the roots of a quadratic equation. We should therefore extract the parameters of the lines represented by a degenerate conic.

C.2 Analysis of Matrices representing Conics

A 2D-conic centered at the origin can be represented by a 2×2 matrix A as follows.

$$\mathsf{x}^\mathsf{T}\,\mathsf{A}\,\mathsf{x} = \rho, \tag{C.8}$$

where $\mathsf{x} := (x_1,\, x_2)^\mathsf{T} \in \mathbb{R}^2$ and $\rho \in \mathbb{R}$ is a scale (radius). The set of points x that satisfy this equation, lie on the conic represented by A. The equation can be made homogeneous by embedding x and A in a projective space. For this purpose we make the definitions

$$\mathsf{x}_H := (x_1,\, x_2,\, 1)^\mathsf{T}, \qquad \mathsf{A}_H := \begin{pmatrix} \mathsf{A} & 0 \\ 0 & -\rho \end{pmatrix}, \tag{C.9}$$

such that

$$\mathsf{x}^\mathsf{T}\,\mathsf{A}\,\mathsf{x} = \rho \quad \Longleftrightarrow \quad \mathsf{x}_H^\mathsf{T}\,\mathsf{A}_H\,\mathsf{x}_H = 0. \tag{C.10}$$

Similarly, for a given 2×2 rotation matrix R we define a homogeneous counterpart as

$$\mathsf{R}_H := \begin{pmatrix} \mathsf{R} & 0 \\ 0 & 1 \end{pmatrix}. \tag{C.11}$$

A conic rotated about the origin by R can be represented by rotating the vector that are multiplied from left and right with the conic matrix in the opposite direction. That is, a conic rotated by R is represented by those vectors x_H that satisfy

$$\mathsf{x}_H^\mathsf{T}\,\mathsf{R}_H\,\mathsf{A}_H\,\mathsf{R}_H^\mathsf{T}\,\mathsf{x}_H = \mathsf{x}_H^\mathsf{T}\begin{pmatrix} \mathsf{R}\,\mathsf{A}\,\mathsf{R}^\mathsf{T} & 0 \\ 0 & -\rho \end{pmatrix} = 0. \tag{C.12}$$

If we diagonalize A we obtain

$$\mathsf{A} = \mathsf{U}\,\mathsf{\Lambda}\,\mathsf{U}^\mathsf{T}, \tag{C.13}$$

where U is a unitary matrix containing the eigenvectors of A in its columns and Λ is a diagonal matrix with the eigenvalues of A on its diagonal. Since A is a real, symmetric matrix, the eigenvalues are real and the eigenvectors are orthogonal, hence U is unitary. Therefore, U gives the rotation matrix by which the conic has been rotated and Λ describes what type of conic A represents. We will denote the eigenvectors of A by u_1 and u_2 such that $\mathsf{U} = (\mathsf{u}_1, \mathsf{u}_2)$. If we define

$$\mathsf{U}_H := \begin{pmatrix} \mathsf{U} & \mathsf{0} \\ \mathsf{0} & 1 \end{pmatrix}, \tag{C.14}$$

then the following relation holds

$$\mathsf{U}_H^\mathsf{T}\, \mathsf{A}_H\, \mathsf{U}_H = \Lambda_H := \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & -\rho \end{pmatrix}, \tag{C.15}$$

where λ_1 and λ_2 are the eigenvalues of A.

However, a conic need not be centered on the origin. Its origin may be translated to a point $\mathsf{t} \in \mathbb{R}^2$, by applying the inverse translation to the points that are multiplied with the conic. If we define a homogeneous translation matrix T_H as

$$\mathsf{T}_H := \begin{pmatrix} \mathsf{I} & -\mathsf{t} \\ -\mathsf{t}^\mathsf{T} & 1 \end{pmatrix}, \tag{C.16}$$

then those points x_H that satisfy

$$\mathsf{x}_H^\mathsf{T}\, \mathsf{T}_H^\mathsf{T}\, \mathsf{A}_H\, \mathsf{T}_H\, \mathsf{x}_H = 0, \tag{C.17}$$

lie on the conic represented by A_H, translated by the vector t.

Suppose we are given an arbitrary, symmetric 3×3 matrix Q_H and we would like to know what type of conics this matrix represents. Using the above definitions we know that Q_H may be written as

$$\mathsf{Q}_H = \mathsf{T}_H^\mathsf{T}\, \mathsf{A}_H\, \mathsf{T}_H. \tag{C.18}$$

Since the top left 2×2 submatrix of T_H is the identity matrix, we can evaluate the eigenvector matrix U of A from that part of Q_H. That is, Q_H has the form

$$\mathsf{Q}_H = \begin{pmatrix} \mathsf{A} & \mathsf{q} \\ \mathsf{q}^\mathsf{T} & p \end{pmatrix}, \tag{C.19}$$

where $\mathsf{q} \in \mathbb{R}^2$ is some vector and $p \in \mathbb{R}$ is a scalar. In some way we need to extract Λ_H and the translation vector t from Q_H. We can write

$$\mathsf{L}_H := \mathsf{U}_H^\mathsf{T}\, \mathsf{Q}_H\, \mathsf{U}_H = \mathsf{U}_H^\mathsf{T}\, \mathsf{T}_H^\mathsf{T}\, \mathsf{U}_H\, \Lambda\, \mathsf{U}_H^\mathsf{T}\, \mathsf{T}_H\, \mathsf{U}_H. \tag{C.20}$$

We find that

$$\mathsf{S}_H := \mathsf{U}_H^\mathsf{T}\,\mathsf{T}_H\,\mathsf{U}_H = \begin{pmatrix} 1 & 0 & -\mathsf{t}^\mathsf{T}\,\mathsf{u}_1 \\ 0 & 1 & -\mathsf{t}^\mathsf{T}\,\mathsf{u}_2 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & s_1 \\ 0 & 1 & s_2 \\ 0 & 0 & 1 \end{pmatrix}, \tag{C.21}$$

where $s_1 := -\mathsf{t}^\mathsf{T}\,\mathsf{u}_1$ and $s_2 := -\mathsf{t}^\mathsf{T}\,\mathsf{u}_2$. Hence,

$$\mathsf{L}_H = \mathsf{S}_H^\mathsf{T}\,\Lambda_H\,\mathsf{S}_H = \begin{pmatrix} \lambda_1 & 0 & \lambda_1\,s_1 \\ 0 & \lambda_2 & \lambda_2\,s_2 \\ \lambda_1\,s_1 & \lambda_2\,s_2 & \lambda_1\,s_1^2 + \lambda_2\,s_2^2 - \rho \end{pmatrix}. \tag{C.22}$$

U and thus U_H can be evaluated from the upper-left 2×2 submatrix of Q_H. The translation vector t can be evaluated from L_H via

$$\mathsf{t} = -s_1\,\mathsf{u}_1 - s_2\,\mathsf{u}_2. \tag{C.23}$$

Furthermore, ρ can be evaluated from L_H by first evaluating λ_1, λ_2, s_1 and s_2. Also note that if $\rho = 0$, the matrix Q_H is at most of rank two. This can be seen from the form of the matrix L_H since the two matrices are related via a similarity transformation. Simply multiply the first row of L_H with s_1 and the second row with s_2. The sum of these two rows is then equal to the third row if $\rho = 0$.

The type of conic represented by Q_H can now be deduced from λ_1, λ_2 and ρ, that can all be evaluated from L_H. We can distinguish between the following types of conics. Note that any scalar multiple of Q_H represents the same conic as Q_H. In particular, $-\mathsf{Q}_H$ represents the same conic as Q_H. Therefore, the following signatures of λ_1, λ_2 and ρ may also be inverted.

- **Point**. $\lambda_1, \lambda_2 > 0$ and $\rho = 0$.
- **Ellipse**. $\lambda_1, \lambda_2 > 0$ and $\rho > 0$.
- **Hyperbola**. $\lambda_1 > 0$ and $\lambda_2 < 0$ or vice versa, and $\rho > 0$.
- **Two intersecting lines**. $\lambda_1 > 0$ and $\lambda_2 < 0$ or vice versa, and $\rho = 0$.
- **Two parallel lines**. $\lambda_1 > 0$ and $\lambda_2 = 0$ or vice versa, and $\rho > 0$.
- **Line**. $\lambda_1 > 0$ and $\lambda_2 = 0$ or vice versa, and $\rho = 0$.

The axes or directions of the various entities are given through the eigenvectors u_1 and u_2, and the scales of the axes by the corresponding eigenvalues.

For example, for the case of two intersecting lines with $\lambda_1 > 0$, $\lambda_2 < 0$ and $\rho = 0$, the set of vectors x that satisfy the following equation lie on the conic.

$$\begin{aligned} & \mathsf{x}^{\mathsf{T}} (\mathsf{u}_1,\, \mathsf{u}_2) \begin{pmatrix} |\lambda_1| & 0 \\ 0 & -|\lambda_2| \end{pmatrix} (\mathsf{u}_1,\, \mathsf{u}_2)^{\mathsf{T}} \mathsf{x} = 0 \\ \Longleftrightarrow \quad & |\lambda_1|\, (\mathsf{x}^{\mathsf{T}}\, \mathsf{u}_1\, \mathsf{u}_1^{\mathsf{T}}\, \mathsf{x}) - |\lambda_2|\, (\mathsf{x}^{\mathsf{T}}\, \mathsf{u}_2\, \mathsf{u}_2^{\mathsf{T}}\, \mathsf{x}) = 0 \end{aligned} \tag{C.24}$$

Since u_1 and u_2 are normalized and orthogonal, the solutions for x to the above equation are simply

$$\mathsf{x} = \pm \frac{1}{\sqrt{|\lambda_1|}}\, \mathsf{u}_1 \pm \frac{1}{\sqrt{|\lambda_2|}}\, \mathsf{u}_2. \tag{C.25}$$

These solutions give the directions of the two lines. Their intersection point is given by t which can be evaluated from the corresponding L_H.

C.3 Intersecting Lines with Conics

Intersecting a line with an arbitrary conic is quite simple. Suppose a line is given in parametric form as

$$\mathsf{x}(\alpha) := \mathsf{p} + \alpha\, \mathsf{r}, \tag{C.26}$$

where p gives the line's offset from the origin and r is the line's direction in homogeneous coordinates. We neglect here the subscript 'H' for brevity. Note that the third component of p is unity while the third component of r is zero. Let A represent a conic in homogeneous coordinates as defined in the previous sections. Now, $\mathsf{x}(\alpha)$ lies on the conic if $\mathsf{x}^{\mathsf{T}}(\alpha)\, \mathsf{A}\, \mathsf{x}(\alpha) = 0$. By expanding this equation we find

$$\begin{aligned} \mathsf{x}^{\mathsf{T}}(\alpha)\, \mathsf{A}\, \mathsf{x} = {} & (\mathsf{p}^{\mathsf{T}} + \alpha\, \mathsf{r}^{\mathsf{T}})\; \mathsf{A}\; (\mathsf{p} + \alpha\, \mathsf{r}) \\ = {} & \mathsf{r}^{\mathsf{T}}\, \mathsf{A}\, \mathsf{r}\, \alpha^2 + (\mathsf{p}^{\mathsf{T}}\, \mathsf{A}\, \mathsf{r} + \mathsf{r}^{\mathsf{T}}\, \mathsf{A}\, \mathsf{p})\; \alpha + \mathsf{p}^{\mathsf{T}}\, \mathsf{A}\, \mathsf{p} \\ = {} & 0. \end{aligned} \tag{C.27}$$

If we define

$$a := \mathsf{r}^{\mathsf{T}}\, \mathsf{A}\, \mathsf{r}, \quad b := \mathsf{p}^{\mathsf{T}}\, \mathsf{A}\, \mathsf{r} + \mathsf{r}^{\mathsf{T}}\, \mathsf{A}\, \mathsf{p}, \quad c := \mathsf{p}^{\mathsf{T}}\, \mathsf{A}\, \mathsf{p}, \tag{C.28}$$

Then we can write the above equation as

$$a\, \alpha^2 + b\, \alpha + c = 0, \tag{C.29}$$

which has the well known solutions

$$\alpha_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{C.30}$$

If the term in the square root is negative, then the line does not intersect the conic, if it is zero it intersects the conic in a single point and if it is positive in two points.

C.4 Summary

The method we use for evaluating the intersection of two non-degenerate conics represented by two symmetric 3×3 matrices A and B can be summarized as follows.

1. Find a degenerate conic as a linear combination of A and B that represents two lines by evaluating the eigenvalues of $\mathsf{M} = \mathsf{B}^{-1}\mathsf{A}$. If λ is a real eigenvalue of M, then $\mathsf{C} = \mathsf{A} - \lambda \mathsf{B}$ is a degenerate conic passing through the intersection points of A and B.
2. Analyze the degenerate conic C. If it represents two lines, extract the line parameters.
3. Intersect the lines found in the previous step with either conic represented by A or B.

Good features of this method are that the only numerically sensitive calculation is the evaluation of eigenvectors and eigenvalues. However, for these many stable numerical routines do already exist. Furthermore, we can work directly with the matrices representing the conics, which makes the method fairly simple to apply.

Bibliography

[1] C. Fuchs, *Extraktion polymorpher Bildstrukturen und ihre topologische und geometrische Gruppierung*. PhD thesis, Akademie der Wissenshaften, Reihe C, Heft 502, 1998.

[2] D. G. Lowe, "Local feature view clustering for 3d object recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 682–688, 2001.

[3] G. H. Granlund and A. Moe, "Unrestricted recognition of 3-D objects using multi-level triplet invariants," in *Proceedings of the Cognitive Vision Workshop*, (Zürich, Switzerland), September 2002. URL: http://www.vision.ethz.ch/cogvis02/.

[4] C. G. Harris and M. J. Stevens, "A combines corner and edge detector," in *Proc. of 4th Alvey Vision Conference*, 1988.

[5] W. Förstner, "A framework for low level feature extraction," in *Computer Vision - ECCV'94* (J. O. Eklundh, ed.), vol. 2 of *LNCS 801*, pp. 383–394, Springer-Verlag, 1994.

[6] F. Mokhtarian and R. Suomela, "Curvature scale space for robust image corner detection," in *Proc. International Conference on Pattern Recognition*, pp. 1819–1821, 1998.

[7] M. Felsberg and G. Sommer, "Image features based on a new approach to 2D rotation invariant quadrature filters," in *Computer Vision, ECCV02, Kopenhagen, 2002* (A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, eds.), vol. 2350 of *LNCS*, pp. 369–383, Springer, 2002.

[8] S. Baker, S. K. Nayar, and H. Murase, "Parametric feature detection," *IJCV*, vol. 27, no. 1, pp. 27–50, 1998.

[9] M. Cazorla, F. Escolano, R. Rizo, and D. Gallardo, "Bayesian models for finding and grouping junctions," in *Second International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 1999. York.

[10] U. Köthe, "Edge and junction detection with an improved structure tensor," in *Pattern Recognition* (B. Michaelis and G. Krell, eds.), LNCS 2781, pp. 25–32, Springer-Verlag, 2003.

[11] M. Shpitalni and H. Lipson, "Classification of sketch strokes and corner detection using conic sections and adaptive clustering," *Trans. of ASME J. of Mechanical Design*, vol. 119, no. 2, pp. 131–135, 1997.

[12] F. Bookstein, "Fitting conic sections to scattered data," *Comp. Graph. Image Proc.*, vol. 9, pp. 56–71, 1979.

[13] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, November 1986.

[14] I. Bronstein, K. Semendjajew, G. Musiol, and H. Mühlig, *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 4 ed., 1999.

[15] E. Berberich, A. Eigenwillig, M. Hemmer, S. Hert, K. Mehlhorn, and E. Schömer, "A computational basis for conic arcs and boolean operations on conic polygons," in *10th European Symposium on Algorithms*, no. 2461 in LNCS, pp. 174–186, Springer, 2002.