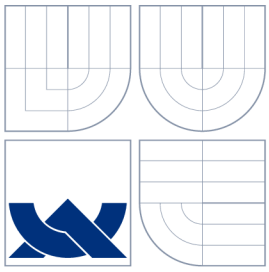


BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

POINT AND LINE PARAMETERIZATIONS USING PARALLEL COORDINATES FOR HOUGH TRANSFORM

PARAMETRIZACE BODŮ A ČAR POMOCÍ PARALELNÍCH SOUŘADNIC PRO POUŽITÍ V HOUGHOVĚ TRANSFORMACI

DOCTORAL THESIS
DISERTAČNÍ PRÁCE

AUTHOR
AUTOR PRÁCE

Ing. MARKÉTA DUBSKÁ

SUPERVISOR
VEDOUCÍ PRÁCE

Doc. ADAM HEROUT, Ph.D.

BRNO 2014

Contents

1 Introduction

The Hough transform is an important tool used in image processing and computer vision. It was used for detection of linear structures in raster images with great precision. Invented in 1962 by Hough [16], it was progressively improved in speed and accuracy, and adopted for detection of different objects. In general, the Hough transform is a parameter estimation method which detect objects in raster images. It is a voting technique where evidence found in an image votes for parameters of objects that could generate the evidence. Most voted combination of parameters is then the output of the method.

In the thesis, parallel coordinates are used for parameterization of the objects. Parallel coordinates – a coordinate system where all axes are mutually parallel – are nowadays mostly used in visualization and data mining [20]. The main contributions of this work are:

- point and line parameterization based on parallel coordinates;
- algorithms for line and grid detection;
- algorithms for vanishing point detection.

Above mentioned parameterizations exploit point-to-line duality between projective and parallel space. The challenge of these transformations is to find a set of mappings which transform infinite space into a

bounded space. The performance of the line parameterization is demonstrated on simple line detection and matrix code extraction. The point parameterization is validated on vanishing point detection. It utilizes the edges in images of the Manhattan world or trajectories of objects with straight movement trajectories like cars.

As it is further shown, such parameterizations are computationally very efficient because they require only line rasterization. Also, all calculations can be implemented without floating-point operations and can be easily accelerated by different hardware architectures. Side-by-side with high performance, they preserve or overcome the accuracy of state-of-the-art methods.

2 Hough Transform

The Hough Transform (HT) [16] is sometimes understood not as a specific algorithm for object detection but as a wide class of algorithms that share a common structure. Princen et al. [29] formalized HT as a *hypothesis testing* process. The structure of HT when described as generically as possible is:

I. Some *evidence* is extracted from the input.

II. For each piece of the evidence, *accumulators* corresponding to the *hypotheses* that are supported by that evidence are incremented. Possible hypotheses are represented by an N-dimensional *parameter space* of accumulators.

III. Probable hypotheses are detected as peaks in the parameter space.

The dimensionality of the parameter space is determined by the number of degrees of freedom of the hypotheses. The parameter space can be represented as an N-dimensional array.

2.1 Hough Transform for Line Detection

Line detection using the Hough transform requires a mapping between a line in an image and a point in the Hough space of parameters. The motivation for introducing new parameterizations is to find the optimal trade-off between requirements for the transformation. These requirements include preference of the bounding Hough space, discretization with

minimal aliasing errors, and uniform distribution of discretization error or intuitive mapping from the original system to the Hough space. In general, the image of a transformed point can be a curve of different shapes; for example circle, sinusoid curve, straight line, etc.

Slope–intercept Parameterization

The first Hough transform, introduced and patented by Paul Hough in 1962 [16], was based on the line equation in the slope–intercept form. Commonly, the slope–intercept line equation has this form:

$$\ell : y = xm + b. \tag{1}$$

However, the method used in the Hough's patent corresponds to:

$$\ell : x = ym + b. \tag{2}$$

Using parameters m and b, all lines passing through a single point form a line in the Hough space, so it is a *point-to-line mapping*. Using a bounded parameter space requires at least two complementary spaces of parameters. In the case of the slope–intercept line equation, these spaces are, for example, the two based on equations (1)(2).

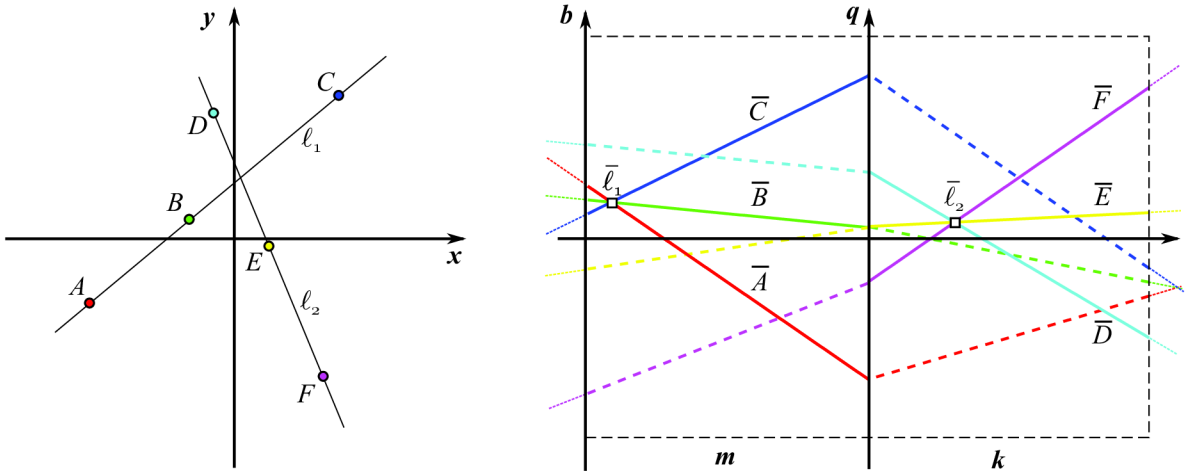


Figure 1: The Hough transform using m–b and k–q parameterizations of a line. (left) Input image and (right) parts of the corresponding Hough spaces attached together.

θ–ϱ Parameterization

In 1972, Duda and Hart [11] introduced a very popular parameterization denoted as θ–ϱ which is very important for its inherently bounded parameter space. It is based on the line equation in the normal form:

$$y \sin\theta + x \cos\theta = \varrho. \tag{3}$$

Parameter θ represents the angle of inclination and ϱ is the length of the shortest chord between the line and the origin of the image coordinate system, Figure 2 (left). In this case, images of all lines passing through a single point form a sinusoid curve in the parameter space, Figure 2 (right).

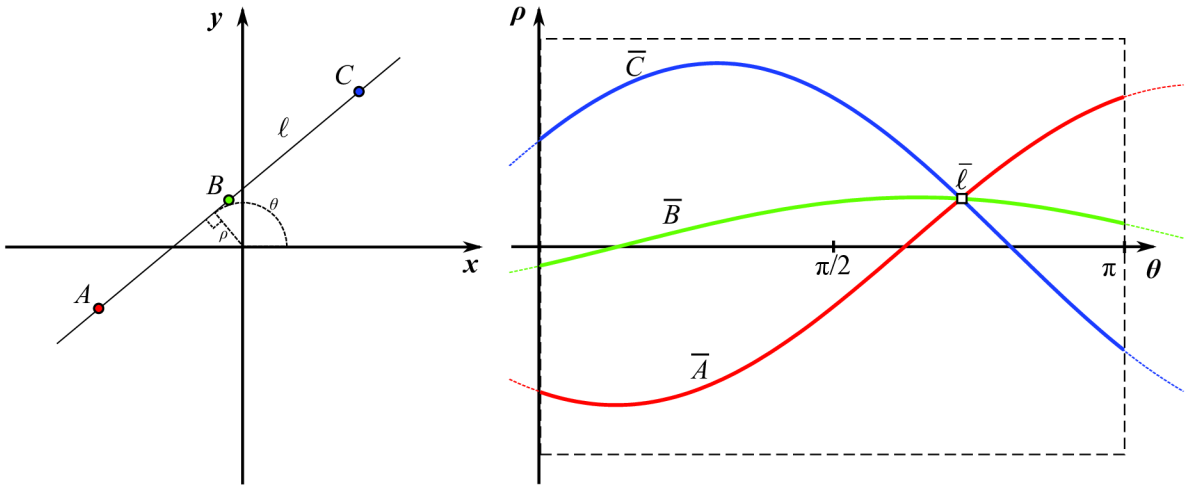


Figure 2: The Hough transform using the θ–ϱ parameterization of lines. (left) Input image and (right) corresponding Hough space.

3 Parallel Coordinates

Parallel coordinates were invented in 1885 by M. d'Ocagne [8] and further studied and popularized by Alfred Inselberg [19]. Currently, parallel coordinates are mostly used as a tool for visualization and data mining in high-dimensional data. Significant computational usage of parallel coordinates is in the air traffic control [18].

3.1 Representation of a Point

The parallel coordinate system represents the vector space by axes which are mutually parallel. The order and the distances between the axes are

arbitrary. Each N-dimensional vector is represented by $N-1$ lines connecting the axes and intersecting them at the corresponding coordinates, Figure 3.

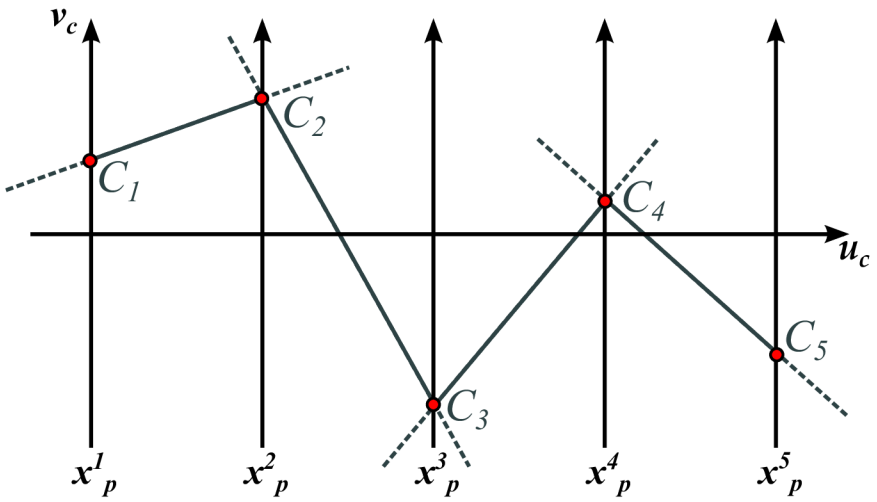


Figure 3: Representation of a 5-dimensional vector in parallel coordinates. The vector is represented by its coordinates $C_1, \ldots, C_5$ on axes $\boldsymbol{x^1_p}, \ldots, \boldsymbol{x^5_p}$, connected by a complete polyline (composed of 4 infinite lines).

3.2 Representation of a Line

In the two-dimensional case, points in the $\boldsymbol{x_c}$-$\boldsymbol{y_c}$ space are represented as lines in the space of parallel coordinates. All representations of collinear points intersect at a unique point – the representation of the common line, Figure 4.

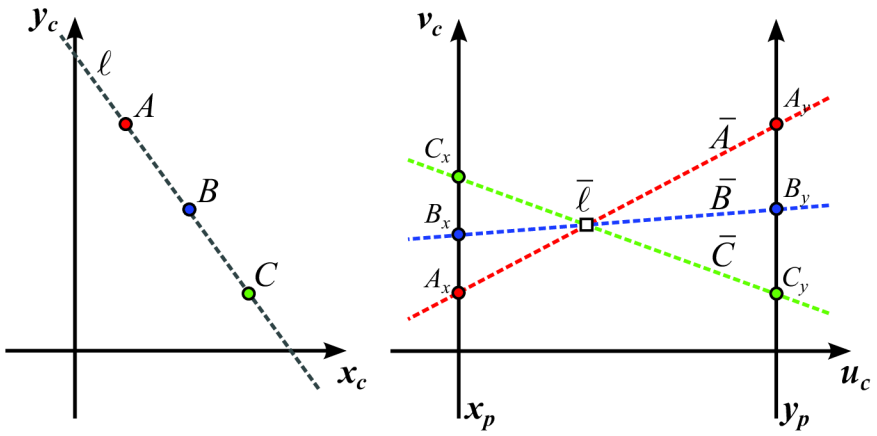


Figure 4: Three collinear points in parallel coordinates. (left) Cartesian coordinate system and (right) space of parallel coordinates. Line ℓ is represented by point $\overline{\ell}$ in parallel coordinates.

Therefore, a line in 2D Cartesian space is represented by a point in parallel coordinates. For some cases, such as line $\ell : y = x$, the corresponding point $\overline{\ell}$ lies at infinity (it is an ideal point), Figure 5.

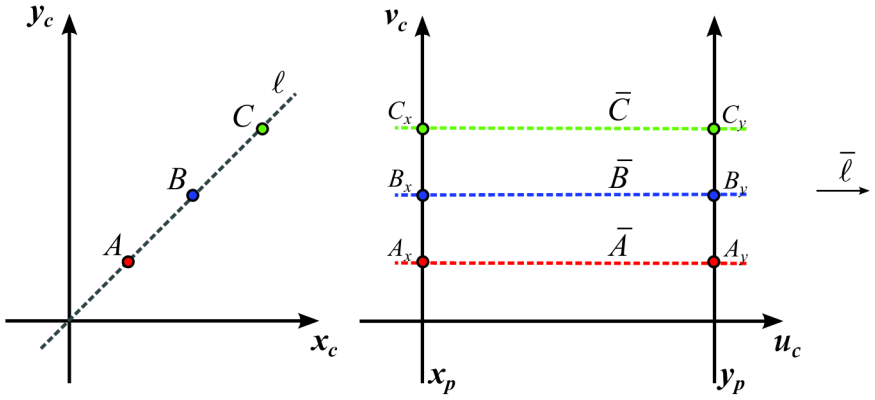


Figure 5: Line $y = x$ in (left) Cartesian coordinate system and (right) space of parallel coordinates. Line ℓ is represented by ideal point at infinity.

Using the homogeneous coordinates for the points and lines, it is possible to define mapping between an arbitrary line $\ell = (a, b, c)$ with equation $\ell : ax + by + c = 0$ and its image $\overline{\ell}$:

$$\ell = (a, b, c) \rightarrow \overline{\ell} = [db, -c, a + b], \tag{4}$$

where d is the distance between parallel axes $\boldsymbol{x_p}$ and $\boldsymbol{y_p}$ [19].

Inversely, each point $\wp = [u, v, w]$ in parallel coordinates, uniquely defines a line in Cartesian coordinates:

$$\overline{\wp} = [u, v, w] \rightarrow \wp = (-u + v, -wd, ud). \tag{5}$$

This point-to-line correspondence defines **duality** between projective plane $\mathbb{P}^2$ and space of parallel coordinates, which is crucial for the further tasks.

4 Parameterizations of Lines Using Parallel Coordinates

Due to the point-to-line duality of a two-dimensional parallel coordinates and projective plane $\mathbb{P}^2$, the parallel coordinates can be used as a line parameterization for the Hough transform. This section presents the properties of such a parameterization, called *PClines* [10] and its cascaded version [9] used for point transformation.

4.1 Point-to-Line Mapping

The transformations of the points and lines between Cartesian and parallel coordinate space can be defined by 3×3 matrices. These matrices are dependent on the arrangement of the parallel coordinate axes. Two following arrangements are considered here. The first one with the $\boldsymbol{x_p}$ axis passing through the origin and the $\boldsymbol{y_p}$ axis through point $[d, 0, 1]$, both with the same orientation, Figure 4. The second has the $\boldsymbol{y_p}$ axis inverted and passing through point $[-d, 0, 1]$, Figure 6.

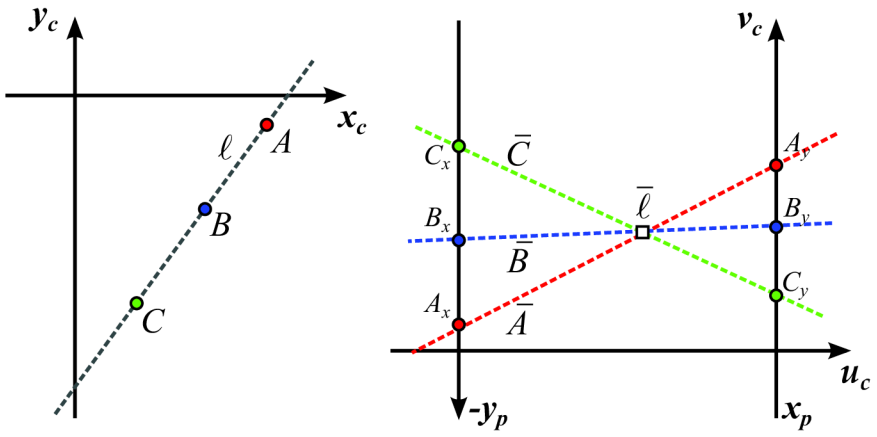


Figure 6: Three collinear points in parallel coordinates. (left) Cartesian coordinate system and (right) space of parallel coordinates. Line ℓ is represented by point $\bar{\ell}$ in parallel coordinates with inverted $\boldsymbol{y_p}$ axis.

The transformations and their corresponding spaces are denoted as $\mathcal{S}$ – straight – when both axes share the same orientation (6) and $\mathcal{T}$ – twisted – for coordinate system with flipped axis (7). The transformation matrices are $\mathbf{S}_\wp$ ($\mathbf{T}_\wp$) for points and $\mathbf{S}_\ell$ ($\mathbf{T}_\ell$) for lines.

$\mathcal{S}$ – Straight Transform

$$\mathbf{S}_\wp = \begin{pmatrix} -1 & 0 & d \\ 1 & 0 & 0 \\ 0 & -d & 0 \end{pmatrix} \qquad \mathbf{S}_\ell = \begin{pmatrix} 0 & 0 & 1 \\ d & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \tag{6}$$

$$\begin{aligned} \mathcal{S}_\wp : \ [x, y, w]\mathbf{S}_\wp &= (-x + y, -wd, xd) \\ \mathcal{S}_\ell : \ (a, b, c)]\mathbf{S}_\ell &= [bd, -c, a + b] \end{aligned}$$

$\mathcal{T}$ – Twisted Transform

$$\mathbf{T}_{\wp} = \begin{pmatrix} 1 & 0 & d \\ 1 & 0 & 0 \\ 0 & -d & 0 \end{pmatrix} \qquad \mathbf{T}_{\ell} = \begin{pmatrix} 0 & 0 & 1 \\ d & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{7}$$

$$\begin{aligned} \mathcal{T}_{\wp} : \ [x, y, w]\mathbf{T}_{\wp} &= (x + y, -wd, xd) \\ \mathcal{T}_{\ell} : \ (a, b, c)]\mathbf{T}_{\ell} &= [bd, -c, a - b] \end{aligned}$$

4.1.1 Subspaces

As Bhattacharya proved [4], an arbitrary point-to-line-mapping (PTLM) mapps a finite subspace to an infinite subspace. That also means that all possible lines from a bounded subspace have representations in an infinite subspace of parallel coordinates. However, two dual PTLM transformations are enough for a transformation into two finite subspaces. In parallel coordinates, this can be solved by using the $\mathcal{S}$ and $\mathcal{T}$ transformations.

From (6) and (7), a line $\ell = (a, b, c)$ has two possible representations; $\overline{\ell}_{\mathcal{S}} = \ell\mathcal{S}_{\ell}$ and $\overline{\ell}_{\mathcal{T}} = \ell\mathcal{T}_{\ell}$ with coordinates (8).

$$\begin{aligned} \overline{\ell}_{\mathcal{S}} &= [bd, -c, a + b] \\ \overline{\ell}_{\mathcal{T}} &= [bd, -c, a - b] \end{aligned} \tag{8}$$

If a line $\ell = (a, b, c)$ is be mapped to a point $\overline{\ell}$ with the following rules:

$$\boxed{\overline{\ell} = \begin{cases} [bd, -c, a + b] & \text{if } ab \geq 0 \\ [bd, -c, a - b] & \text{if } ab \leq 0, \end{cases}} \tag{9}$$

the image of the line always mapped between coordinate axes in $\mathcal{S}$ or $\mathcal{T}$ space. The subspaces of $\mathcal{T}$ and $\mathcal{S}$ spaces bordered with coordinate axes can be "attached" one to another. Figure 7 illustrates the spaces attached along the $\boldsymbol{x_p}$ axis. Attaching also the $\boldsymbol{y_p}$ and $-\boldsymbol{y_p}$ axes results in an enclosed Möbius strip. Figure 7 shows the original image with points and lines and their representations in $\mathcal{TS}$ space.

For an inverse mapping of the point $\overline{\ell} = [u, v, 1]$ from the $\mathcal{TS}$ space attached along $\boldsymbol{x_p}$ axis, the following formula is used:

$$\boxed{\ell = \begin{cases} (d - u, u, -vd) & \text{if } u \geq 0 \\ (d + u, u, -vd) & \text{if } u < 0. \end{cases}} \tag{10}$$

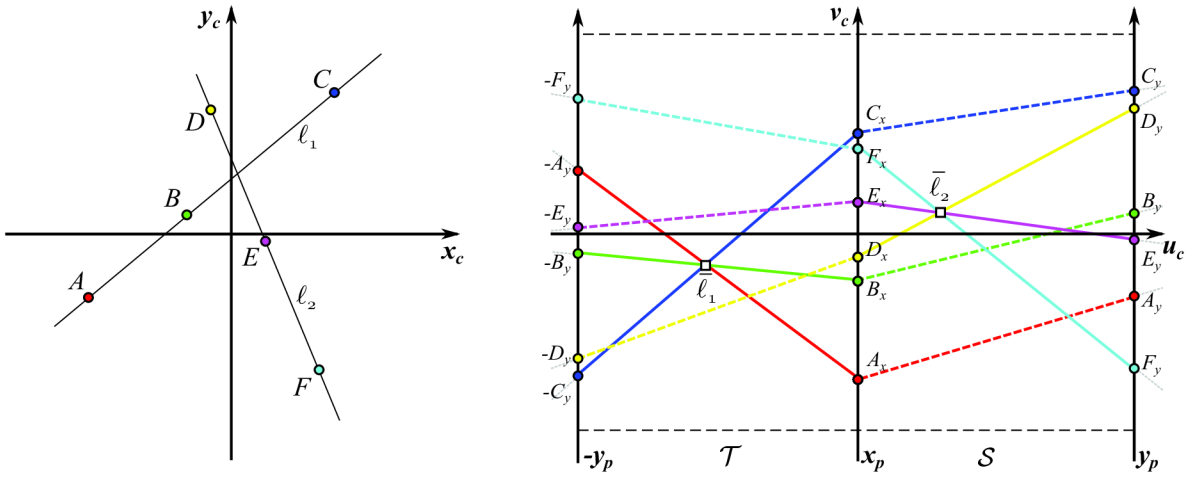


Figure 7: (left) Original x-y space and (right) its PClines representation – the corresponding $\mathcal{TS}$ space.

4.2 Concurrent Lines

Lines are considered to be concurrent if they intersect at a common point. When using parallel coordinates for representing these lines, they are projected to collinear points. These points lie on a line which corresponds to the intersection of the lines in Cartesian coordinates, Figure 8. And, as shown in the previous section, also the dual statement holds true in parallel coordinates: representations of collinear points intersect at a common point, Figure 4.

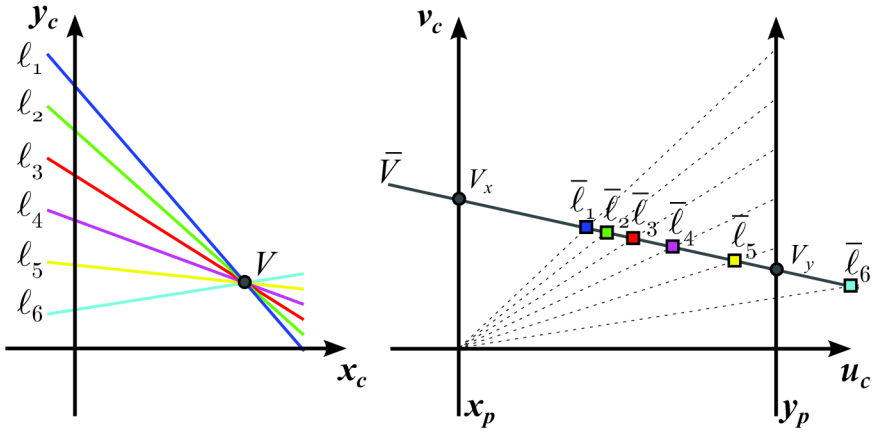


Figure 8: Six concurrent lines in (left) Cartesian coordinate system and (right) space of parallel coordinates. Concurrent lines ℓ_i are represented by collinear points $\overline{\ell}_i$ in parallel coordinates.

A set of arbitrary four concurrent lines can be characterized with a cross-ratio [14]. In general, the cross ratio is a numeric association between a 4-tuple of collinear points. What is very important characteristic of the

cross-ratio is its invariance under projective transformation. Because of the duality between $\mathbb{P}^2$ and parallel coordinates, the value of the cross-ratio is preserved also under transformation to parallel coordinates. The cross-ratio for four collinear points – images of the lines – stands the same.

The ability of the parallel coordinates to preserve the cross-ratio together with invariance under perspective projection leads to its usage for detection of perspectively projected group of parallel lines (e.g. a grid of a matrix code). If the lines were originally parallel and equidistant, three detected points in parallel coordinates with known relative indices are enough to define whole sequence. Suppose three detected points p_a, p_b, p_c with indices a, b, c respectively. From definition of cross ratio, the fourth point p_d with index d is calculated as:

$$\begin{aligned} &\frac{(p_a - p_c)(p_b - p_d)}{(p_a - p_d)(p_b - p_c)} = \frac{(a - c)(b - d)}{(a - d)(b - c)} \\ &\alpha = (a - c)(b - d) \\ &\beta = (a - d)(b - c) \\ &\boxed{p_d = \frac{\alpha p_a p_c - \beta p_b p_c + (\alpha - \beta) p_a p_b}{\alpha p_b - \beta p_a - (\alpha - \beta) p_c}}. \end{aligned} \tag{11}$$

The equations are evaluated separately for each coordinate and thus the position of point p_d can be easily found. The precise indices (a, b, c, d) in the sequence of the lines are not required, only the relative positions between them are necessary.

4.3 Line-to-line Mapping

Point-to-line duality of two-dimensional parallel coordinates and projective plane $\mathbb{P}^2$ can be used in a manner similar to the Cascaded Hough Transform [32] – lines are transformed to points and they are again transformed to lines. Dually, a point is mapped to a line and then again to a point. As shown below, transformations based on parallel coordinates, Figure 9, can be used to project ideal points to regular points. Therefore, the projective plane $\mathbb{P}^2$ can be projected to a finite space.

Transformation matrices which define point-to-line duality are presented in Section 4.1. When two of these mappings are applied sequentially in order to form a composition, a point is mapped again to a point and the same holds true for every line. The mappings vary in combination of the $\mathcal{S}$ and $\mathcal{T}$ transform. As a result, four different composite mappings

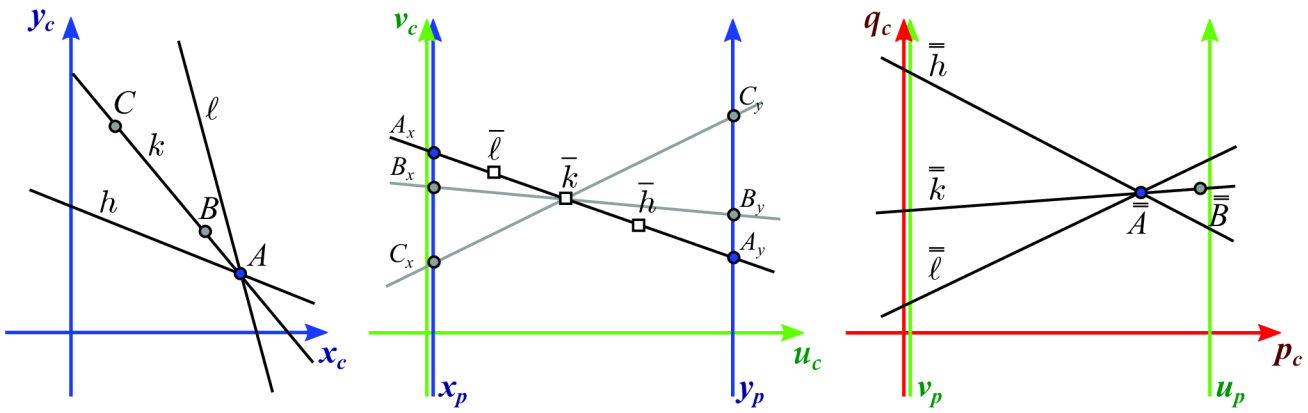


Figure 9: Three collinear points and three concurrent lines in cascaded parallel coordinates. (left) Cartesian coordinate system, (middle) first space of parallel coordinates and (right) second space of parallel coordinates. A line is mapped to a point and then to another line; and dually, a point is mapped to a line and to a point again.

can be constructed: $\mathcal{S} \circ \mathcal{S}$, $\mathcal{T} \circ \mathcal{S}$, $\mathcal{T} \circ \mathcal{T}$, $\mathcal{S} \circ \mathcal{T}$ (12 – 15). The distance of the parallel axes in the first space of parallel coordinates is d, the distance in the second one is D.

Composition of two $\mathcal{S}$ spaces

$$\mathbf{SS}_\wp = \mathbf{S}_\wp \mathbf{S}_\ell = \begin{pmatrix} 0 & -d & -1 \\ 0 & 0 & 1 \\ -dD & 0 & -d \end{pmatrix} \qquad \mathbf{SS}_\ell = \mathbf{S}_\ell \mathbf{S}_\wp = \begin{pmatrix} 0 & -D & 0 \\ -d & -D & dD \\ -1 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \mathcal{SS}_\wp : \ [x, y, w]\,\mathbf{SS}_\wp &= [-dDw, -dx, -x + y - dw] \\ \mathcal{SS}_\ell : \ (a, b, c)\mathbf{SS}_\ell &= (db + c, Da + Db, -dDb) \end{aligned} \tag{12}$$

Composition of $\mathcal{S}$ space and then $\mathcal{T}$ space

$$\mathbf{ST}_\wp = \mathbf{S}_\wp \mathbf{T}_\ell = \begin{pmatrix} 0 & -d & -1 \\ 0 & 0 & 1 \\ -dD & 0 & d \end{pmatrix} \qquad \mathbf{ST}_\ell = \mathbf{S}_\ell \mathbf{T}_\wp = \begin{pmatrix} 0 & -D & 0 \\ d & -D & dD \\ -1 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \mathcal{ST}_\wp : \ [x, y, w]\,\mathbf{ST}_\wp &= [-dDw, -dx, -x + y + dw] \\ \mathcal{ST}_\ell : \ (a, b, c)\mathbf{ST}_\ell &= (-db + c, Da + Db, -dDb) \end{aligned} \tag{13}$$

Composition of two $\mathcal{T}$ spaces

$$\mathbf{TT}_{\wp} = \mathbf{T}_{\wp}\mathbf{T}_{\ell} = \begin{pmatrix} 0 & -d & 1 \\ 0 & 0 & 1 \\ -dD & 0 & d \end{pmatrix} \qquad \mathbf{TT}_{\ell} = \mathbf{T}_{\ell}\mathbf{T}_{\wp} = \begin{pmatrix} 0 & -D & 0 \\ d & D & dD \\ -1 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \mathcal{TT}_{\wp}: \ [x,y,w]\,\mathbf{TT}_{\wp} &= [-dDw, -dx, x+y+dw] \\ \mathcal{TT}_{\ell}: \ (a,b,c)\mathbf{TT}_{\ell} &= (-db+c, Da-Db, -dDb) \end{aligned} \tag{14}$$

Composition of $\mathcal{T}$ space and then $\mathcal{S}$ space

$$\mathbf{TS}_{\wp} = \mathbf{T}_{\wp}\mathbf{S}_{\ell} = \begin{pmatrix} 0 & -d & 1 \\ 0 & 0 & 1 \\ -dD & 0 & -d \end{pmatrix} \qquad \mathbf{TS}_{\ell} = \mathbf{T}_{\ell}\mathbf{S}_{\wp} = \begin{pmatrix} 0 & -D & 0 \\ -d & D & dD \\ -1 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \mathcal{TS}_{\wp}: \ [x,y,w]\,\mathbf{TS}_{\wp} &= [-dDw, -dx, x+y-dw] \\ \mathcal{TS}_{\ell}: \ (a,b,c)\mathbf{TS}_{\ell} &= (db+c, Da-Db, -dDb) \end{aligned} \tag{15}$$

4.3.1 Subspaces

Each of the above mentioned mappings is a transformation of one infinite space to another [4]. The goal is to find an infinite subspace which is mapped to a finite subspace, similarly to the $\mathcal{TS}$ space for line transformation (Section 4.1.1) or Cascaded Hough Transform [31]. Application of above mentioned point mappings, each on one quadrant of the real projective plane, leads to finite domain of the transformation, Figure 10. Such a transformation maps both the ideal and the regular points to the regular points.

Point Transformation

It can be proved that all points from $\mathrm{I_c}$ quadrant transformed via two $\mathcal{T}$ transformation are mapped to a triangular subspace in $\mathrm{III_p}$ quadrant, $\mathrm{II_c}$ quadrant transformed with $\mathcal{ST}_{\wp}$ to $\mathrm{II_p}$ quadrant, $\mathrm{III_c}$ quadrant transformed with $\mathcal{TS}_{\wp}$ to $\mathrm{IV_p}$ quadrant and $\mathrm{IV_c}$ quadrant transformed with $\mathcal{SS}_{\wp}$ to $\mathrm{I_p}$ quadrant, Figure 10.

The coordinate axes $\boldsymbol{x_c}$, $\boldsymbol{y_c}$ and ideal line $\boldsymbol{\ell_\infty} = (0,0,1)$ lie on borders of the quadrants of the real projective plane. Their images also lie

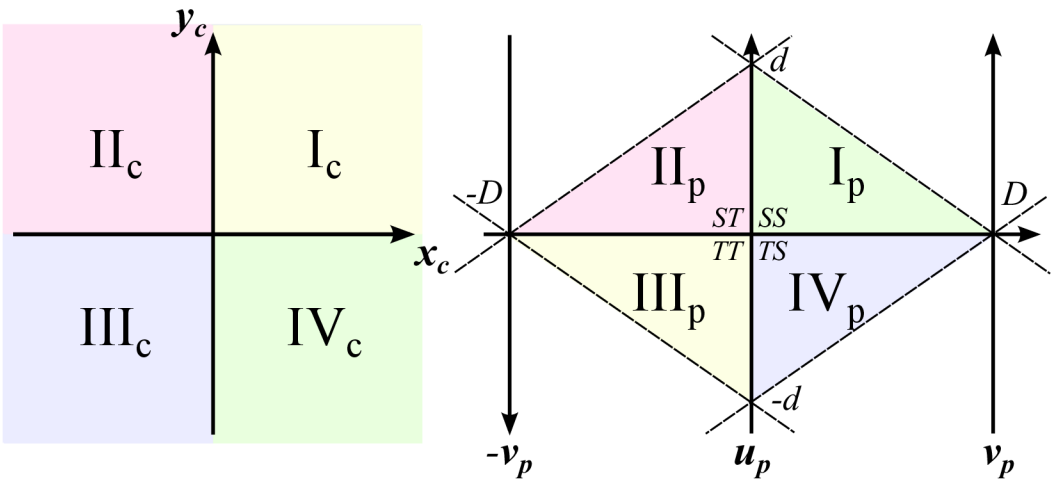


Figure 10: Correspondence between quadrants in Cartesian coordinates and triangle subspaces of the space of parallel coordinates. (left) Quadrants of the original infinite Cartesian space. (right) Quadrants of the parallel spaces attached to each other.

on the borders of the finite triangular subspaces. This enables the four parts, each mapped by a different mapping to be attached together. The attached space has a diamond shape and in the further text is referred to as the *diamond space* and the mapping denoted $\mathcal{D}$.

An arbitrary point in the real projective plane; including the ideal points; is mapped to a regular point in the diamond space and for some specific cases (such as the coordinate axes), the point is mapped to the couple of points on the borders (16).

$$\boxed{\begin{array}{ll} \mathcal{D}_{\wp}([x,y,w]) & = [-dDw, -dx, \operatorname{sgn}(xy)x + y + \operatorname{sgn}(y)dw] \\ \mathcal{D}_{\wp}^{-1}([p,q,1]) & = [Dq, \operatorname{sgn}(u)dp + \operatorname{sgn}(v)Dq - dD, p] \end{array}} \tag{16}$$

Line Transformation

An image of a line using transformations (12)–(15) is again a line (17).

$$\begin{array}{ll} (a,b,c)\mathbf{SS}_\ell & = \ell_{\mathcal{SS}} = (db + c, Da + Db, -dDb) \\ (a,b,c)\mathbf{TS}_\ell & = \ell_{\mathcal{TS}} = (db + c, Da - Db, -dDb) \\ (a,b,c)\mathbf{TT}_\ell & = \ell_{\mathcal{TT}} = (-db + c, Da - Db, -dDb) \\ (a,b,c)\mathbf{ST}_\ell & = \ell_{\mathcal{ST}} = (-db + c, Da + Db, -dDb) \end{array} \tag{17}$$

However, the image of a straight line in the joined diamond space is not a line anymore. The result of the mapping $\mathcal{D}_\ell$ is a polyline whose number

of segments depends on the number of quadrants the line passes through (each quadrant is mapped by a different transformation). Figure 11 shows a line after $\mathcal{SS}_\ell, \mathcal{ST}_\ell, \mathcal{TT}_\ell$ and $\mathcal{TS}_\ell$ transformation. From each transformation, only a line segment in the finite triangle subspace is included in the diamond space, Figure 12.

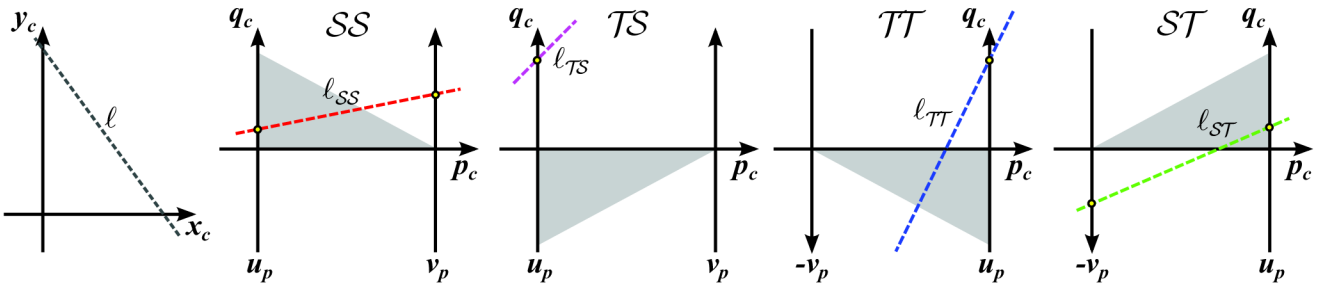


Figure 11: Representation of a line using different transformations. Only a darker triangular subspaces are parts of final diamond space.

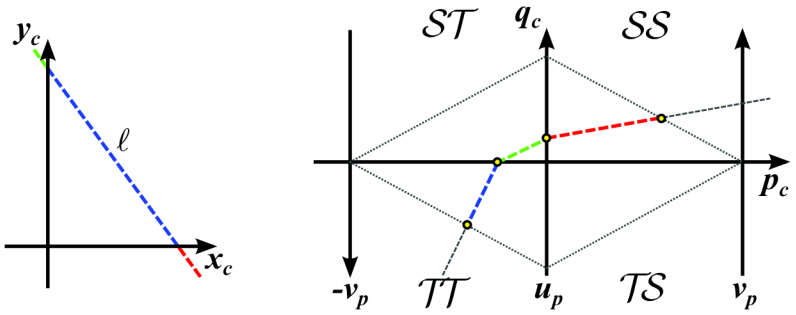


Figure 12: Representation of a line using diamond transformation. Polyline with three segments, each corresponding to one quadrant and one transformation.

The sequence of the endpoints defining the polyline depends on the intersections of the line images with coordinate axes $\boldsymbol{p_c}, \boldsymbol{q_c}$ and the diamond space borders. They can be defined using nonzero signum function (18). When a line passes through just two quadrants (vertical lines, horizontal lines, lines through the origin), one segment always degenerates to a point.

$\alpha = \mathrm{sgn}(ab);\ \beta = \mathrm{sgn}(bc);\ \gamma = \mathrm{sgn}(ac)$
$(a, b, c) \rightarrow$

$$\boxed{\left[\frac{\alpha dDa}{c+\gamma da}, \frac{-\alpha dc}{c+\gamma da}\right], \left[\frac{dDb}{c+\beta db}, 0\right], \left[0, \frac{db}{a+\alpha b}\right], \left[\frac{-\alpha dDa}{c+\gamma da}, \frac{\alpha c}{c+\gamma da}\right]} \tag{18}$$

Using the diamond space as an accumulator for the Hough transform requires discretization of the space. Reprojection of the discrete grid in the diamond space to image space is shown on the Figure 13. The shape of the back-projected accumulator's bins in the image space is different in the $\boldsymbol{x_c}$ and $\boldsymbol{y_c}$ axis because the diamond space was made with the vertical axis inverted ($\boldsymbol{y_p}$ and $\boldsymbol{v_p}$). As it is shown, the area covered by the accumulator's bin increases with the distance from the image origin. In Section 5.3, this behavior is used for an accurate vanishing point detection, where the bigger error in localization far from the origin has the same influence as the smaller error near to the origin.

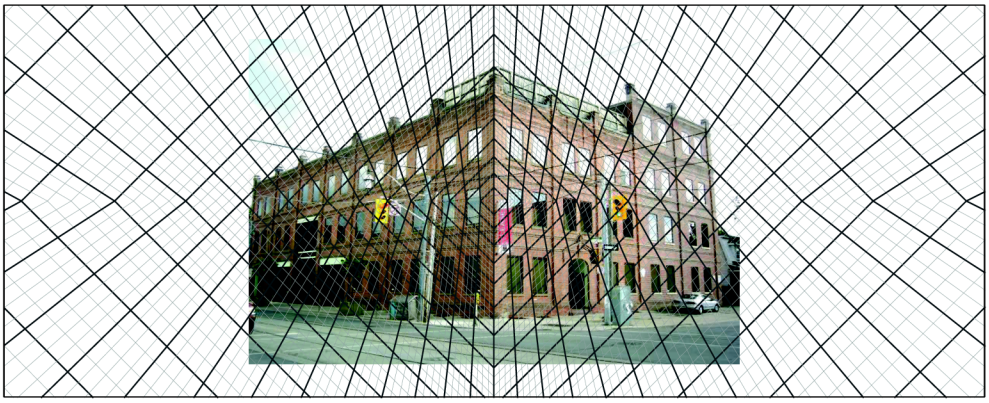


Figure 13: Visualization of the accumulator bins from the diamond space back-projected to the image plane.

5 Usage of Line Parameterizations based on Parallel Coordinates in Computer Vision

This section gives several examples how the parameterizations of points and lines based on parallel coordinated can be used in computer vision tasks in context of the Hough transform.

5.1 Line Detection

Line parameterization based on parallel coordinates used for line detection using the Hough transform, is presented in Algorithm 1.

As in the standard Hough transform [11], points of interest are firstly detected (line 2). For each point a two-segment polyline with vertices A, B, C is accumulated. One part is in the $\mathcal{T}$ space (line 7) and the second in the $\mathcal{S}$ space (line 8). The Hough space H is scanned for local maxima

Algorithm 1 Detection of lines using the Hough transform and parallel coordinates.

Input: Input image I with size $w_i \times h_i$, accumulation space H, axes distance d

Output: Detected lines $L = \{(a_1, b_1, c_1), \ldots\}$

1: Clear accumulator H
2: $E \subseteq \{1, \ldots, w_i\} \times \{1, \ldots, h_i\}$ is set of points of interest in I
3: **for all** $(i, j) \in E$ **do**
4: $\quad A = [-d, -j]$
5: $\quad B = [0, i]$
6: $\quad C = [d, j]$
7: $\quad$ Accumulate line $\overline{AB}$ to H
8: $\quad$ Accumulate line $\overline{BC}$ to H
9: **end for**
10: $M = \{(u, v); H(u, v) \text{ is a significant local maximum}\}$
11: $L = \{(d - \operatorname{sgn}(u)u, u, -vd); \forall (u, v) \in M\}$
12: **return** L

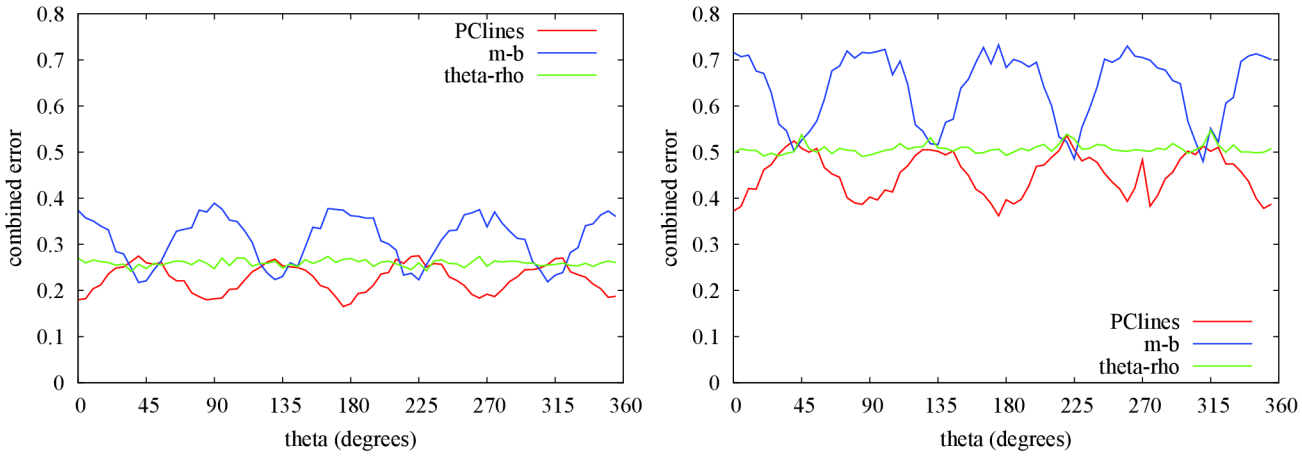
above a given threshold (line 10) and corresponding line parameters are calculated using Equation (10) (line 11).

For a real accuracy comparison between different parameterizations a dataset of automatically generated black-and-white images are used. In each image, one line is rasterized first. Then, 25 000 noise pixels are inverted. Figure 14 compares the errors of the three methods for lines with different θ generated within 5° wide intervals. For every interval, 100 images are generated. The computed average error as it depends on line's slope of all lines detected in the images are shown in Figure 14a and the average of the 5 least accurate lines out of all 100 lines on 14b.

The measurements confirm the theoretical considerations: the θ–ϱ parameterization discretized the space evenly; PClines are about as accurate as θ–ϱ for $\theta \in \{45°, 135°, 225°, 315°\}$ and more accurate at $\theta \in \{0°, 90°, 180°, 270°\}$; the m–b parameterization is the least accurate of the three evaluated methods.

Harnessing the Edge Orientation

O'Gorman and Clowes [28] improve the basic θ–ϱ parameterization with their idea of not accumulating values for all discretized values of θ but for



(a) Average error (b) Average of worst 5 detections

Figure 14: Line localization error as it depends on the lines' slope. For x on the horizontal scale, the lines' slope in degrees is at interval $\langle x, x+5)$. Red: PClines; Green: θ–ϱ; Blue: m–b. (a) Average error over all lines. (b) Average error of the 5 least accurate lines, i.e. a pessimistic error estimation.

a single value of θ, instead. The appropriate θ for a point can be obtained from the gradient of the detected edge present at this point [30].

One common way to calculate the local gradient direction of the image intensity is by using the Sobel operator. By Sobel convolution kernels, two derivative estimations G_x and G_y can be obtained for any discrete location in the input image. Using the response of the convolution kernels, the horizontal position of the $\overline{\ell}$ as it depends on the gradients is the following:

$$\overline{\ell}_u = d\frac{G_y}{G_x + \operatorname{sgn}(G_x G_y) G_y}. \tag{19}$$

In order to avoid errors caused by noise and the discrete nature of the input image, accumulators within a suitable interval $\langle \overline{\ell}_u - r, \overline{\ell}_u + r \rangle$ around the calculated $\overline{\ell}_u$ position are also incremented.

To evaluate the detection rate, 500 blurred images are used, each with 20 random half-planes and 25 000 noisy pixels. Local maxima higher then threshold $T = \max(H)/4$ are considered as detected lines. These lines are compared with ground truth lines. Results in Figure 15 show that for small r, the detection rate is low, which is caused by inaccurate estimation of edge gradient. The detection rate can be increased by: better estimation of the gradient or bigger value of r. With bigger Sobel kernel, the estimation of the gradient is more accurate, i.e. closer to the

correct position. With higher r it becomes more probable, that even with inaccurate estimate of gradient, the accumulated bins include the correct position. However with very high value of r the space is more "messy" because noisy pixels start to form lines (i.e. local maxima) which are not real lines.

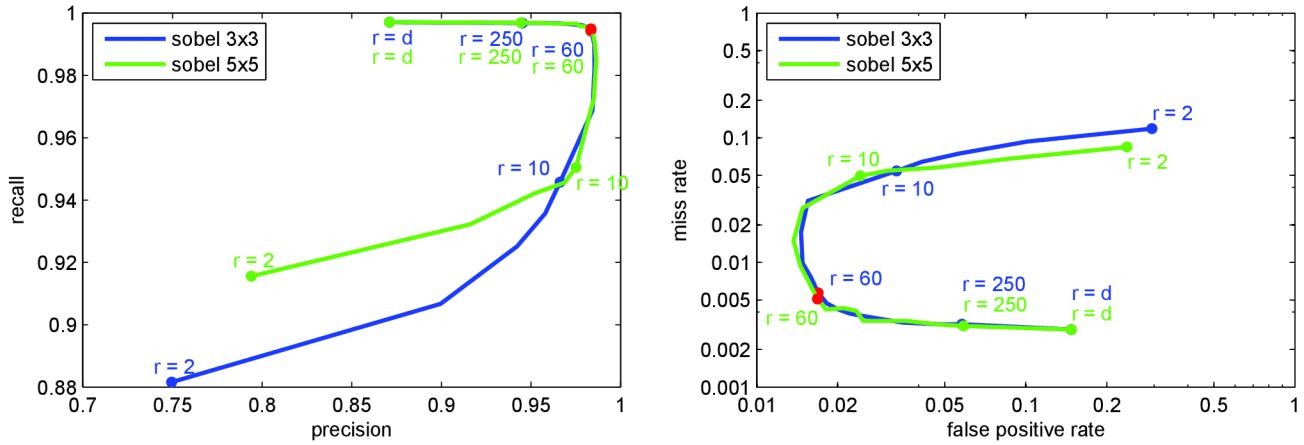


Figure 15: PRC and ROC characteristics for line detection using edge orientation. Blue lines stand for values using the Sobel kernel with size 3×3. Green lines are for kernel with size 5×5. Red dots are the optimal r; i.e. r with maximal F1 score.

5.2 Grid Detection

Section 4.2 discusses the invariance of the cross ratio under projection and in parallel coordinates. This property can be used for detection of parallel lines under perspective projection. The detection is demonstrated on a projected grid. Two methods for the detection are introduced here. The common part involves: firstly, the edge points are detected and their gradients are estimated; a histogram of oriented gradients with a small number of bins is built together with a list of edges that voted for the bins. Since a grid is expected to be present in the image, two main peaks can be detected in the histogram, Figure 16. These two peaks, roughly 90° degrees apart, represent two main orientations in the image. Because the two main orientations in the image are known, only two narrow stripe parts of the $\mathcal{TS}$ space need to be accumulated, instead of the whole space.

After accumulation, each stripe of the Hough space contains collinear maxima with a known distribution which have to be detected.

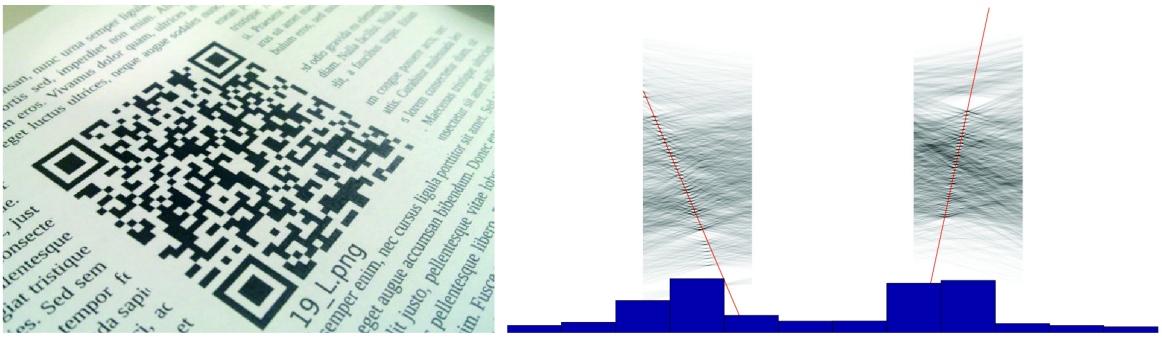


Figure 16: Input QRcodes and corresponding accumulated stripes in $\mathcal{TS}$ space which corresponds to two dominant edge orientations.

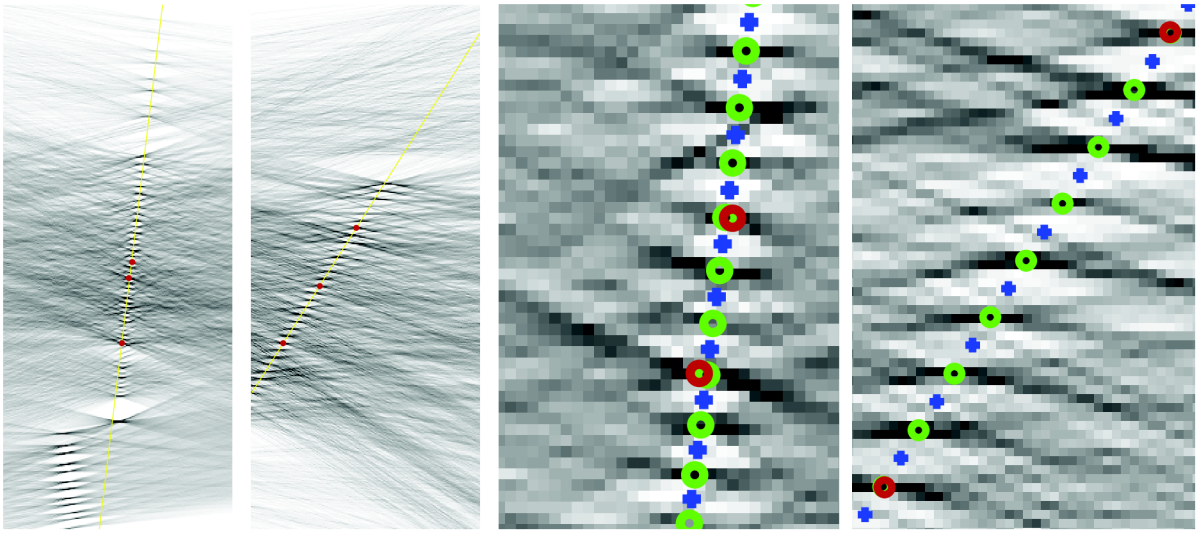
5.2.1 Detection by Using the Cross Ratio

The value of the cross-ratio for a quadruplet of the points in the parallel coordinates is equal to the cross-ratio of the lines represented by these points. This can be used for the detection of a pencil of lines with a known distribution, for example equidistant lines under a perspective projection.

The detection algorithm requires the filled Hough space as its input. Firstly, few highest local maxima are detected. The maxima, in an ideal case, correspond to the most significant lines from the grid. For uniquely determining the perspective projection and rest of the lines, three lines with known relative indices are required. Therefore, triplets $\{p_a, p_b, p_c\}$ are selected from the set. Then, linear regression with least squares approach is used to calculate line ℓ best fitting points p_a, p_b, p_c, yellow line in Figure 17a. The line is a hypothetical vanishing point of one group of the grid lines. The relative indices of the lines are found and used in equation (11). The confidence score of the hypothesis about the vanishing point and the distribution of the lines depends on the values in the Hough space at the locations of the estimated maxima, green circles in Figure 17b. The algorithm returns the hypothesis with the highest score.

Detection of QR Codes

By finding the point between two consecutive maxima and sampling the image at the intersections of lines represented by these points, a bitmap of the data matrix code can by extracted, Figure 18a. The input image is sampled at the intersections and after extracting the grayscale image, adaptive thresholding is used for creating the binary bitmap of the matrix code, Figure 18b.



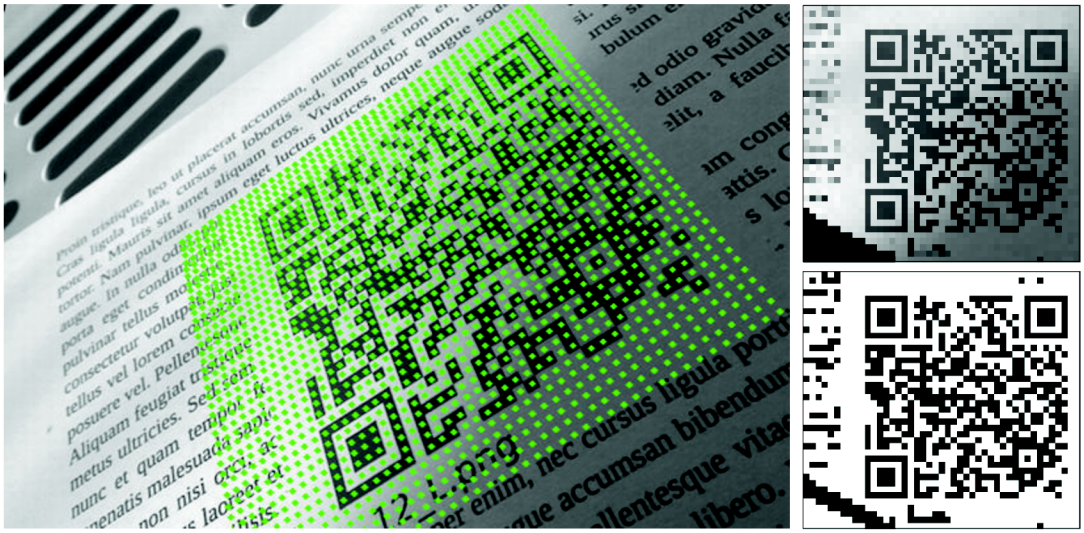
(a) Stripes of the accumulator (b) Zoomed parts

Figure 17: (a) The whole accumulator stripes: triplets of maxima (red dots) with the corresponding line used for vanishing point estimation (yellow line). (b) Zoomed parts of the stripes: blue crosses represent the estimated centers, where low values are expected and the green circles stand for the estimated maxima.

The performance of the algorithm is evaluated on a dataset of challenging photos of QRcodes and as a reference is used the ZXing Project. Table 1 shows that the algorithm is very well resistant to rotation and perspective deformation of the matrix code. The main issue of the algorithm based on PClines exhibits when the code is surrounded by intensive parallel edges.

	presented			ZXing		
(count)	100%	99%	95%	100%	99 %	95%
plain (113)	88.5	97.3	98.2	88.5	97.3	97.3
rot (95)	68.4	94.7	98.9	53.7	58.9	62.1
pers (110)	89.1	95.5	95.5	69.1	74.5	74.5
rot+pers (112)	78.6	96.4	97.3	9.8	9.8	9.8
all (430)	81.6	96.0	97.4	55.3	60.2	60.9

Table 1: Detection rate achieved by ZXing and the algorithm based on PClines. The detection rates are reported for different notions of correct detection: 100% / 99% / 95% of code pixels detected correctly.



(a) QR code cells sampling (b) Extracted data

Figure 18: Extraction of the matrix code. The points shown in green are sampled and form a bitmap which is then adaptively thresholded.

5.2.2 Detection of Grids by Inverse Projection

A regular grid under perspective projection can be transformed with a projective transformation so that the lines in each direction are again parallel and aligned with coordinate axes. In parallel coordinates, it means a transformation of the space after which the points representing the lines are equidistant and lie on a line parallel to the axes.

The algorithm based on inverse projection expects the accumulated stripe of the Hough space using PClines. Firstly, a set of maxima is detected in each stripe and lines are fitted to each group of the points. The lines represent the vanishing point $\overline{U}$ and $\overline{V}$. Further, they are processed separately for each part. The goal is to obtain a uniform distribution of the maxima on the line using a nonuniform sampling along the line. To get the n_i sample, the algorithm uses the remapping equation depending on the orientation of the horizon.

$$t = \frac{\begin{vmatrix} m_u & m_v & 0 \\ 1 & 1 & 1 \\ \sigma(\overline{U})b_u & \sigma(\overline{U})b_v & d \end{vmatrix}}{\begin{vmatrix} m_u & m_v \\ b_u & b_v \end{vmatrix}}, \quad \sigma(x) = \begin{cases} 1 & x \in \mathcal{S} \\ -1 & x \in \mathcal{T} \end{cases} \tag{20}$$

where d is the distance between parallel axes. To define the scale, two

points are fixed to have the same distance in the original Hough space and in the new sampling (n_0, n_1). The resampling function maps each point from 1D vector to a corresponding value in the Hough space on position $[u_i, v_i, 1]$, (21).

$$\begin{aligned} u_i &= \frac{(-t-\sigma(\ell))u_1u_0 + d(iu_1 - (i-1)u_0)}{(-t-\sigma(\ell))(iu_0 - (i-1)u_1) + d} \\ v_i &= \frac{(-t-\sigma(\ell))(iu_0v_1 - (i-1)u_1v_0) + d(iv_1 - (i-1)v_0)}{(-t-\sigma(\ell))(iu_0 - (i-1)u_1) + d} \\ n_i &= H([u_i, v_i, 1]) \end{aligned} \tag{21}$$

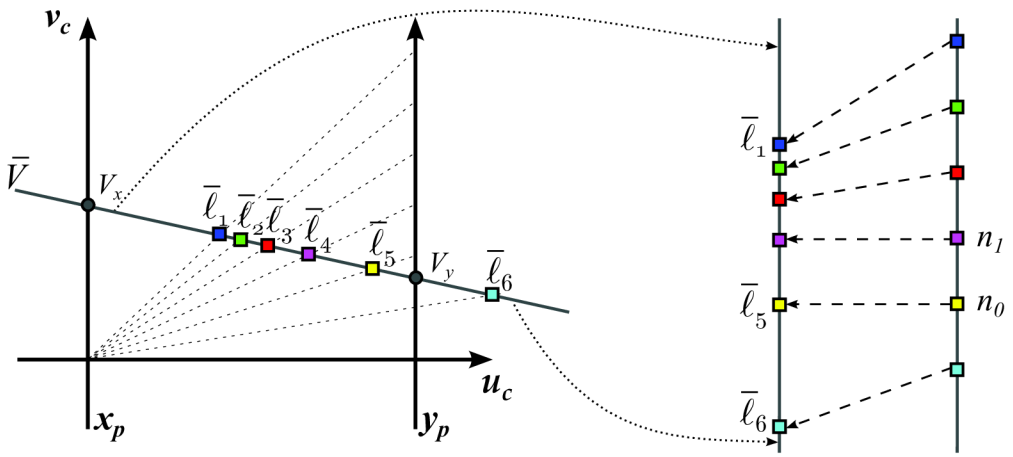


Figure 19: Sampling of the values along the vanishing point. Dashed arrows show the couples of input image points.

After sampling along the line ℓ, the 1D vector $\{n_i\}$ has maxima distributed uniformly. In order to find the frequency of the maxima, autocorrelation of the sequence is used. Using the uniform distance and one maximum, the positions of the rest of the maxima are derived. These indices are again used in (21) and the resulting $[u_j, v_j, 1]$ are the positions of the maxima in the Hough space.

Detection of Fractal Marker Field

Detection based on the inverse projection was used for Fractal Marker Field (FMF) extraction. FMF [15] is a fiduciary marker with a fractal structure. Because FMF consists of more then one frequency of the grid lines, the algorithm is modified to find more then one dominant frequency of maxima. Similar to QRcode detection the input image is sampled in the center of each module.

Table 2 shows the detection rates on this set of images. When the FMF is viewed from a skew angle without rotation (*pers*), the detector is confused by the different frequencies of the two groups of (originally perpendicular) lines and in each direction, different levels are sampled. Figure 20 shows examples of successfully detected and localized FMF under different conditions.

	plain	rot	pers	rot + pers	all
480 × 320	100	90.3	80.3	87.7	88.5
600 × 400	100	99.2	87.9	97.7	96.7
1200 × 800	100	100	93.9	100	98.9

Table 2: Detection rate (%) achieved by the detector based on parallel coordinates.

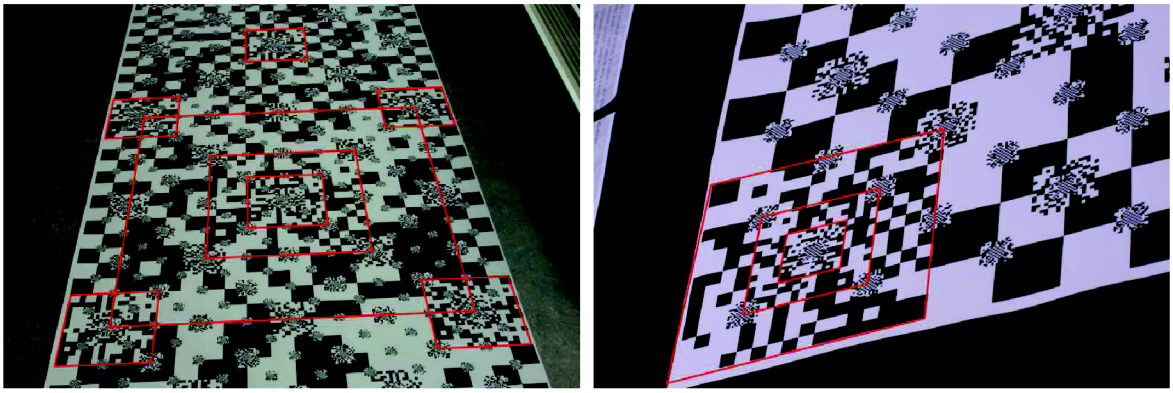


Figure 20: Successfully detected and localized Fractal Marker Fields on real-life images.

5.3 Detection of Vanishing Points

In man-made architecture, parallel and perpendicular structures often occur. When captured by a camcorder or a camera, these regular structures can be used for estimation of camera orientation or calibration and many other tasks. At the same time, the vanishing points tend to be very stable and supported by various parts of the scene and their detection is thus robust against various distortions. Having the vanishing points reliably and efficiently detected facilitates many other computer vision tasks.

Algorithm 2 shows the detection of the vanishing point by using the diamond space. Firstly, the input image is normalized. The edgelets with

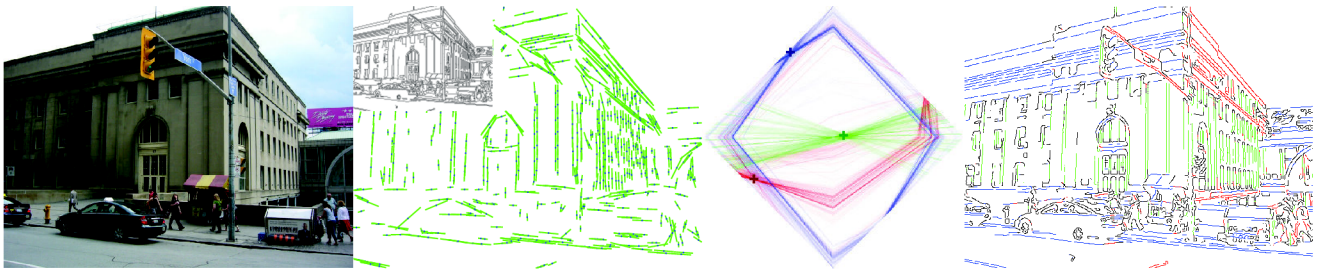
a known position and orientation are detected (line 2). These information are used to calculate parameters of the corresponding fitted line (line 4). Lines are transformed to polylines (18) (line 5) and accumulated into the diamond space (line 6). The highest peak in the diamond space is the most voted vanishing point (line 8) and by using Equation (16) the detected diamond space maximum is projected to the image plane (line 9).

Algorithm 2 Vanishing point detection using diamond space.

Input: Normalized image I, accumulation space H
Output: Vanishing point v_1

1: Clear accumlator H
2: $E \subseteq \{1, \ldots, w_i\} \times \{1, \ldots, h_i\} \times \langle 0, \pi \rangle$ is set of points of interest in I
3: **for all** $(i, j, \theta) \in E$ **do**
4: $\quad \ell = (\cos\theta, \sin\theta, -(i\cos\theta + j\sin\theta))$
5: $\quad$ Get polyline vertices P for line ℓ
6: $\quad$ Accumulate polyline to H
7: **end for**
8: $M = (p, q); H(p, q) \geq H(i, j), \forall (i, j) \in I$
9: $v_1 = [q, \mathrm{sgn}(p)p + \mathrm{sgn}(q)q - 1, p]$
10: **return** v_1

If there are more then one vanishing point to be found the accumulator is cleared and only edgelets which do not contribute to the already found vanishing points are accumulated and the highest response is sought for.



(a) Input image (b) Extracted edges (c) Accumulator (d) Color-coded edges

Figure 21: Detection of orthogonal vanishing points. Right images depict the affiliation of edges/contributions to different orthogonal VPs by color.

In a Manhattan world scenario, three projections of *orthogonal* vanishing points are of interest. The new three maxima are found to be orthogonal, have maximal response in the accumulator and be close to

the originally detected triplet.

The detection accuracy is evaluated on the York Urban Database [7], consisting of 102 images, each with three orthogonal ground truth vanishing points. The parameters (space resolution, image normalization) were obtained from the training set and the evaluation was done on the complete set in order to be comparable with previous works. Two means of evaluation are used: detection rate with 10° angular error tolerance, Table 3, and cumulative histogram of the count of correctly recognized vanishing points based on the angular error tolerance, Figure 22.

method	[27]	[12]	[25]	[24]	[26]	[13]	**ours**
correct [%]	94.35	100⋆	84.6	99.3⋆⋆	90.03	93	88.04/**98.04**
avg. err [°]	3.5	1.63	-	-	<3	-	1.87/**1.41**

Table 3: Detection rate at $< 10°$ angular error tolerance. For the presented method, the value for directly detected VPs are reported, followed by the orthogonalized VPs in bold. ⋆ Column should be omitted because the authors seem to apply the tolerance on individual angles. ⋆⋆ Column should also be treated lightly because the authors feed the detector only with edges user-annotated as belonging to one of the VPs.

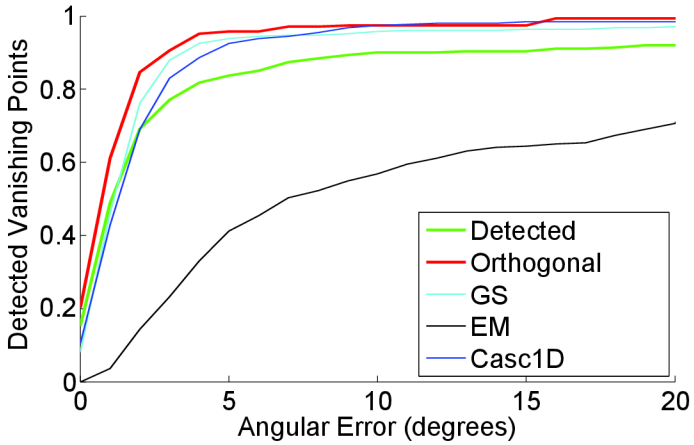


Figure 22: Cumulative histogram of the number of correctly detected vanishing points. *x-axis:* average angular error of the detected vanishing points from the ground truth. *y-axis:* fraction of vanishing points detected with the given error tolerance. **green:** Presented algorithm without the orthogonalization. **red:** Presented after search for orthogonal triplet of vanishing points. **GS**, **EM** and **Casc1D** are algorithms used in [23, 22, 2].

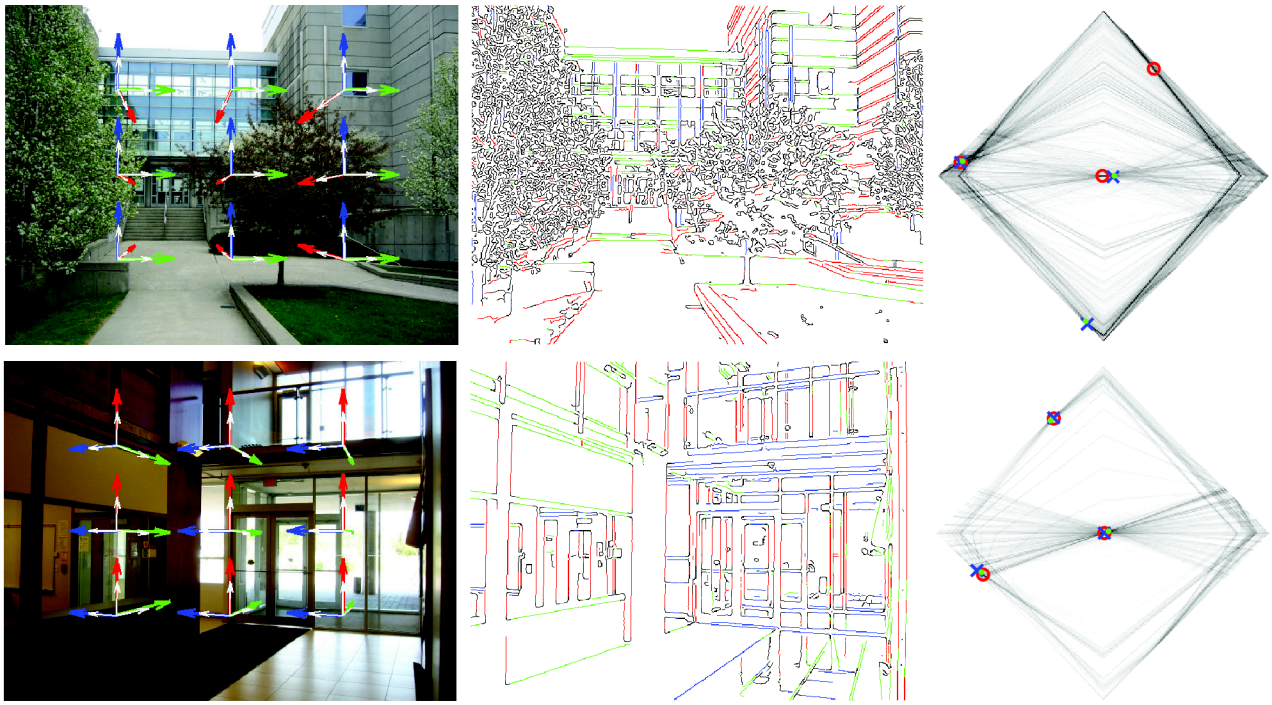


Figure 23: Examples of the diamond accumulation space. White arrows represent ground truth directions and the red-green-blue are the detected ones. In accumulated diamond space: **red circles** Non-orthogonal vanishing points $(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \bar{\mathbf{v}}_3)$; **blue crosses** Orthogonal triplet of vanishing points; **green dots** Ground truth vanishing points.

5.4 Roadside Camera Calibration

The number of internet-connected cameras is quickly increasing and a notable amount of them are used in traffic monitoring. The goal is to provide fully automatic traffic processing algorithms – leading towards vehicle classification and counting, speed measurement, congestion detection, etc. In the presented method, the calibration of internal camera parameters as well as external parameters (camera orientation and position up to scale with respect to the dominant motion of the vehicles) and radial distortion compensation parameters are automatically determined.

Figure 24 shows the vanishing points with the color notation and VP visualization to be used throughout this section: *red* 1^{st} vanishing point in the direction of the car motion; *green* 2^{nd} vanishing point in the ground plane, perpendicular to the vehicle motion; *blue* 3^{rd}vanishing point perpendicular to the ground plane.

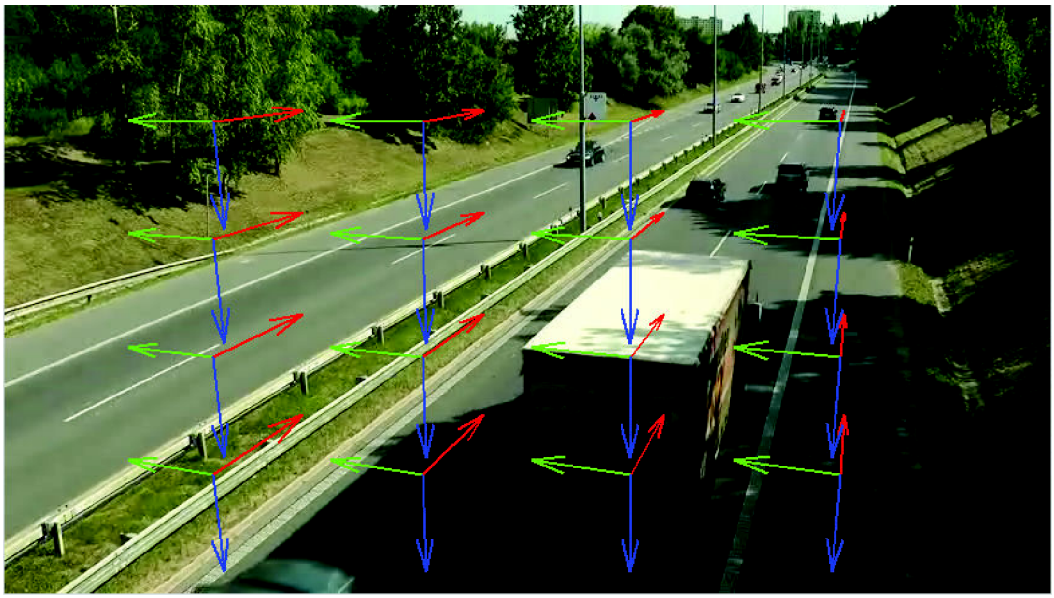


Figure 24: Automatic approach to camera calibration by recognizing the three dominant vanishing points which characterize the vehicles and their motion.

First Vanishing Point Extraction

For the detection of 1^{st} VP, the Hough transform with the *diamond space* accumulator is used. In each video frame, feature points are detected and tracked in the subsequent frame. Successfully detected and tracked points exhibiting a significant movement are treated as fragments of vehicle trajectories. These fragments of trajectories are extended to infinite lines, assuming that they pass through the first vanishing point. All these lines vote in the diamond space accumulator. The most voted point is considered to be the first vanishing point. Figure 25 shows the tracked points accumulated to the diamond space. Once the first VP is determined, moving points can be discerned whether they move towards the VP or from it, or whether they are moving in a completely different direction.

Second Vanishing Point

The 2^{nd} vanishing point is the direction parallel to the road and perpendicular to the first direction. Again, the diamond space is used for its detection. An edge background model is used to select only edges on moving objects (probable vehicles). The model is updated each frame to deal with shadows and other slow changes. Figure 26 shows the edge background model, omitted and accumulated edges and the diamond space.

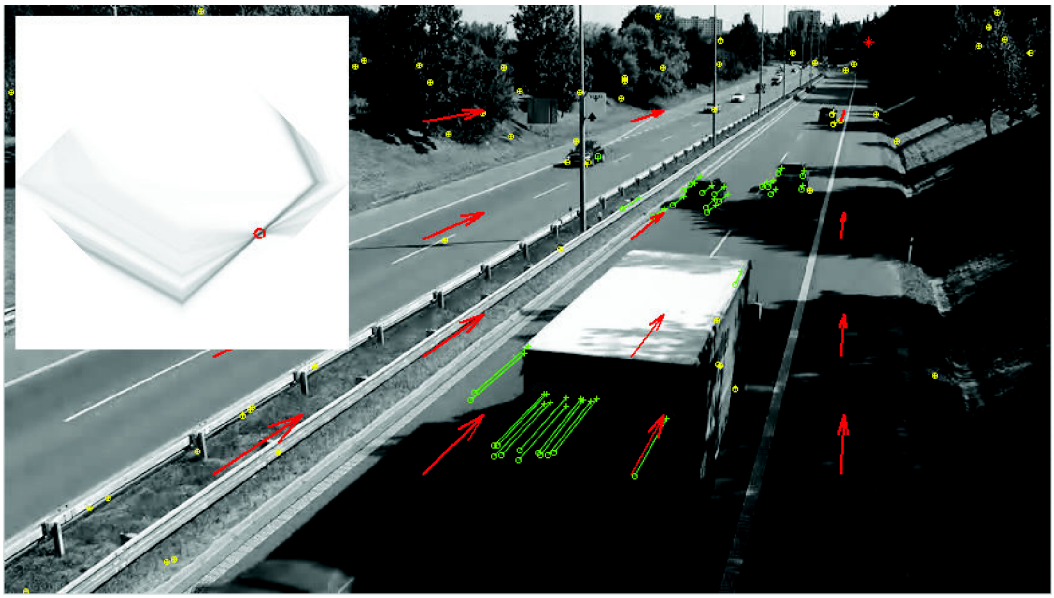


Figure 25: Illustration of the tracked points. Points marked by green exhibit a significant movement and they are accumulated. Points marked by yellow are stable points and do not vote. The accumulation diamond space is in the top left corner.

Third Vanishing Point, Principal Point and Focal Length

The third vanishing point corresponds to the direction perpendicular to the ground plane. Unfortunately, in majority of the roadside views, there seems to be minimal amount of edges supporting the third VP. Instead of finding the third VP, its position is calculated using the first two VPs and the assumption that the principal point is in the middle of the image.

Radial Distortion Compensation

In practice, some real-life cameras exhibit a large degree of radial distortion. Provided the assumption of the road being straight, the tracked trajectories can be used to compensate for the camera's radial distortion. The corrected position of input points can be modeled by the polynomial radial distortion model [5]. In order to find distortion parameters the extracted trajectories are used. Optimal parameters are found by minimization of the sum of square differences of all points in all trajectories to their best fitting lines. The evolutionary strategy is used to search for the first two coefficients. The optimization is done on-line. When new trajectories are tracked, one iteration of the optimization is executed. The radial distortion compensation process is shown in Figure 27.

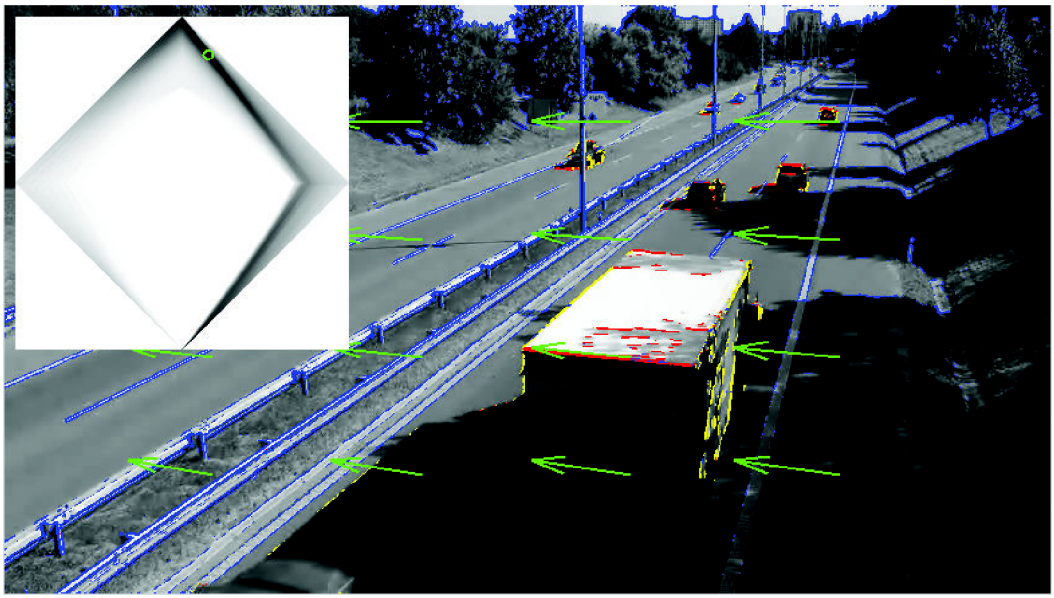
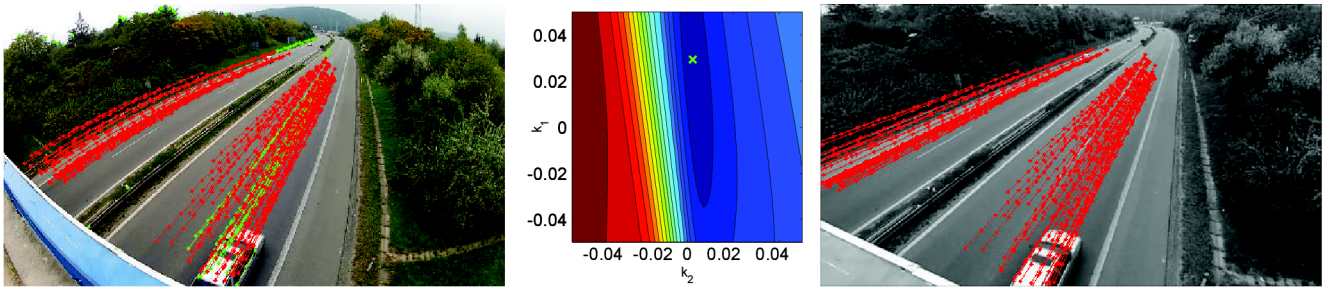


Figure 26: Accumulation of the 2nd vanishing point. Blue edges belong to the background. Yellow edges are omitted from voting because of their vertical direction or direction towards the first VP. Red edges are accumulated to the diamond space (in the corner; green circle marks the maximum).



(a) Original trajectories (b) Parameter space (c) Undistorted trajectories

Figure 27: Radial distortion compensation. (a) Original image with trajectories. (b) Parameter space with value calculated from trajectories. Green cross stands for the optimal parameter combination found by the evolution algorithm. The color gradient shows the error for each combination of the parameters k_1, k_2. (c) Undistorted image using the optimal coefficients.

Experimental Results

The presented approach is evaluated on 5 groups of videos, each containing 5–8 videos. Videos in a common group share the same camera intrinsic parameters but have different extrinsic parameters and capture different scenes or scenes from a different view.

In order to evaluate the accuracy of the detection of the vanishing points, the precision of length measurements in videos similarly to Zhang et al. [33] is computed. From each video, 15–25 pairs of feature points are tracked in 21 subsequent frames. These points are projected with the matrix obtained from the vanishing points and the stability of their distance d is evaluated. Error of i^{th} pair in j^{th} sequence is calculated as

$$e_{ji} = \left| 1 - \frac{d_{ij}}{\overline{d}_j} \right|, \tag{22}$$

where $\overline{d}_j$ is the mean distance in the j^{th} sequence. For each video, two errors are computed from e_{ji} – the worst (e_w^v) and the mean error (e_m^v).

group	g1	g2	g3	g4	g5
e_w^g (%)	6.5	1.8	10.1	5.3	4.0
e_m^g (%)	1.2	0.2	1.3	0.8	0.7
f	705.7	7163.7	674.6	769.6	2465.1

Table 4: Mean and worst length-measurement error for groups of videos in % and the computed focal lengths.

Table 4 shows the worst and mean error for the groups and the computed focal lengths. The focal length f is taken from the video with the lowest e_m^v in the group. It is mentioned here in order to illustrate the differences in the camera settings. Larger f leads to smaller length-measurement error due to smaller perspective distortion and consequent smaller dependence on the point tracker accuracy. Zhang et al. [33] report similar measurements (single scene, 28 point pairs, 6 sequences), their mean error appears to be 6 %, the worst 19 %, the second worst 13 %.

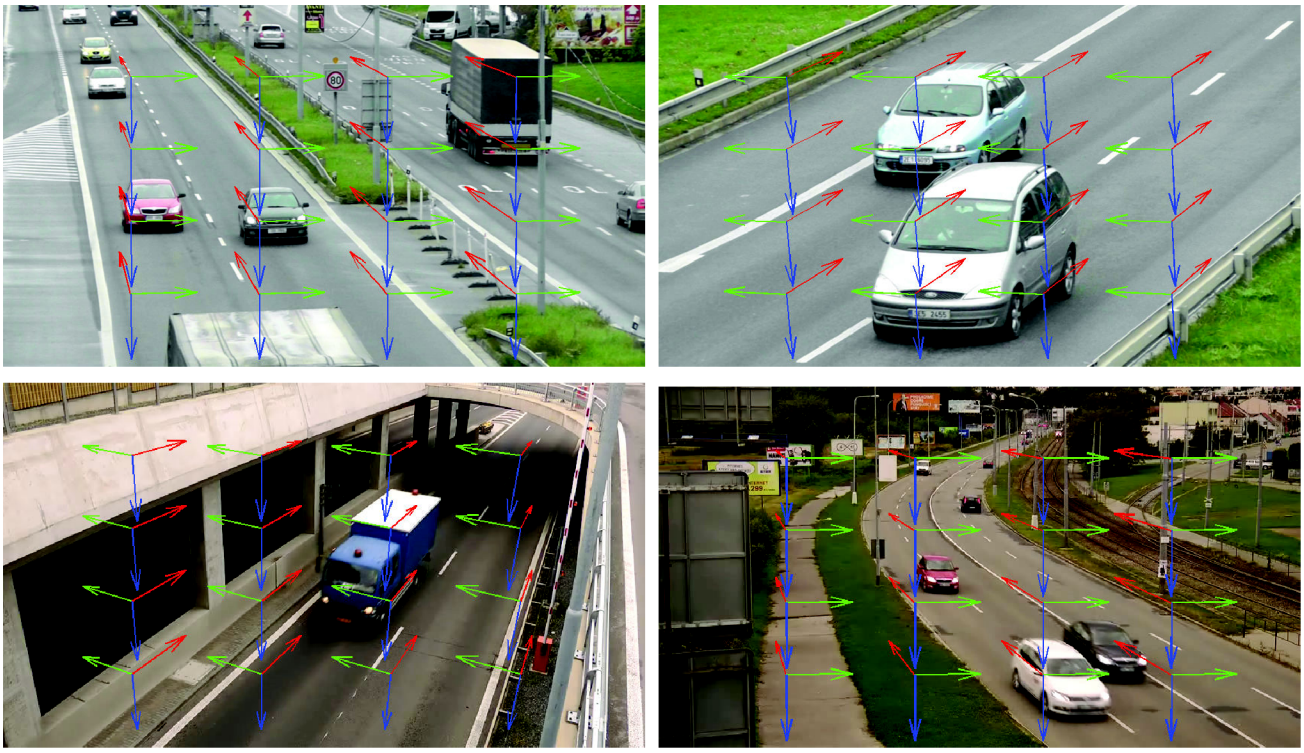


Figure 28: Examples of real-life videos: Automatic detection of three vanishing points.

Figure 29: Examples of real-life videos: Radial distortion estimation and compensation.

6 Conclusion

This short report presents the parameterizations of lines and points using parallel coordinates. These parameterizations are based on point-to-line duality between projective plane $\mathbb{P}^2$ and the space of parallel coordinates. By composing two point-to-line mappings, final transformation has bounded range for an arbitrary bounded space. Using the mapping two times consecutively leads to a cascaded version which defines point-to-point transformation. By this mapping, all points, including the ideal points at infinity are mapped to regular points.

Both transformations can be used in the Hough transform manner for detection of line structures and vanishing points. The accumulation can be done very fast, since it requires only line rasterization and can be implemented without using floating point operations. The line parameterization allows convenient detection of parallel and concurrent lines. This was demonstrated on the detection and extraction of matrix code and fiduciary markers.

Point-to-point mapping was used for detection of orthogonal and non-orthogonal vanishing points in real-world images or videos. In these algorithms, accumulated lines are obtained from oriented edges or straight trajectories. The results show that the presented algorithms outperforms existing methods in terms of accuracy and at the same time, they are computationally very efficient.

References

[1] M. E. Antone and S. Teller. Automatic recovery of relative camera rotations for urban scenes. In *Proceedings of CVPR*, 2000.

[2] S. T. Barnard. Interpreting perspective images. *Artificial Intelligence*, 1983.

[3] H.-G. Beyer and H.-P. Schwefel. Evolution strategies – a comprehensive introduction. *Natural computing*, 2002.

[4] P. Bhattacharya, A. Rosenfeld, and I. Weiss. Point-to-line mappings as Hough transforms. *Pattern Recognition Letters*, 2002.

[5] D. C. Brown. Close-range camera calibration. *Photogrammetric engineering*, 1971.

[6] R. Cipolla, T. Drummond, and D. Robertson. Camera calibration from vanishing points in images of architectural scenes. In *Proceedings of BMVC*, 1999.

[7] P. Denis, J. H. Elder, and F. J. Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *Proceedings of ECCV*, 2008.

[8] M. d'Ocagne. *Coordonnées parallèles et axiales. Méthode de transformation géométrique et procédé nouveau de calcul graphique déduits de la considération des coordonnées parallèlles.* Gauthier-Villars, 1885.

[9] M. Dubská and A. Herout. Real projective plane mapping for detection of orthogonal vanishing points. In *Proceedings of BMVC*, 2013.

[10] M. Dubská, A. Herout, and J. Havel. PClines - line detection using parallel coordinates. In *Proceedings of CVPR*, 2011.

[11] R. O. Duda and P. E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 1972.

[12] W. Elloumi, S. Treuillet, and R. Leconge. Tracking orthogonal vanishing points in video sequences for a reliable camera orientation in Manhattan world. In *Proceedings of CISP*, 2012.

[13] W. Förstner. Optimal vanishing point detection and rotation estimation of single images from a legoland scene. In *Proceedings of ISPRS*, 2010.

[14] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision.* 2000.

[15] A. Herout, M. Zachariáš, M. Dubská, and J. Havel. Fractal marker fields: No more scale limitations for fiduciary markers. In *Proceedings of ISMAR*, 2012.

[16] P. V. C. Hough. Method and means for recognizing complex patterns, 1962. U.S. Patent 3,069,654.

[17] C. Hughes, P. Denny, M. E. Glavin, and E. Jones. Equidistant fish-eye calibration and rectification by vanishing point extraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[18] A. Inselberg. N-Dimensional information display method for air traffic control, 1989. US Patent 4823272.

[19] A. Inselberg. *Parallel Coordinates; Visual Multidimensional Geometry and Its Applications.* Springer, 2009.

[20] A. Inselberg and B. Dimsdale. Parallel coordinates. In *Human-Machine Interactive Systems.* Springer, 1991.

[21] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 2002.

[22] J. Košecká and W. Zhang. Video compass. In *Proceedings of ECCV*. 2002.

[23] B. Li, K. Peng, X. Ying, and H. Zha. Vanishing point detection using cascaded 1D Hough transform from single images. *Pattern Recognition Letters*, 2012.

[24] F. M. Mirzaei and S. I. Roumeliotis. Optimal estimation of vanishing points in a Manhattan world. In *Proceedings og ICCV*, 2011.

[25] M. Nieto and L. Salgado. Non-linear optimization for robust estimation of vanishing points. In *Proceedings of ICIP*, 2010.

[26] M. Nieto and L. Salgado. Real-time robust estimation of vanishing points through nonlinear optimization. In *Proceedings of RTIVP*, 2010.

[27] M. Nieto and L. Salgado. Simultaneous estimation of vanishing points and their converging lines using the EM algorithm. *Pattern Recognition Letters*, 2011.

[28] F. O'Gorman and M. B. Clowes. Finding picture edges through collinearity of feature points. *IEEE Transactions on Computers*, 1976.

[29] J. Princen, J. Illingworth, and J. Kittler. Hypothesis testing: a framework for analyzing and optimizing Hough transform performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1994.

[30] L. G. Shapiro and G. C. Stockman. *Computer Vision*. Tom Robbins, 2001.

[31] T. Tuytelaars, L. V. Gool, M. Proesmans, and T. Moons. The cascaded Hough transform as an aid in aerial image interpretation. In *Proceedings of ICCV*, 1998.

[32] T. Tuytelaars, M. Proesmans, L. V. Gool, and E. Mi. The cascaded Hough transform. In *Proceedings of ICIP*, 1997.

[33] Z. Zhang, T. Tan, K. Huang, and Y. Wang. Practical camera calibration from moving objects for traffic scene surveillance. *IEEE Transactions on Circuits and Systems for Video Technology*, 2013.