J Real-Time Image Proc
DOI 10.1007/s11554-013-0325-6

ORIGINAL RESEARCH PAPER

Real-time precise detection of regular grids and matrix codes

Markéta Dubská · Adam Herout · Jiří Havel

Received: 19 July 2012 / Accepted: 17 January 2013

Abstract The traditional approach in detecting sets of concurrent and/or parallel lines is to first detect lines in the image and then find such groups of them which meet the concurrence condition. The Hough Transform can be used for detecting the lines and variants of HT such as the Cascaded Hough Transform can be used to detect the vanishing points. However, these approaches disregard much of the information actually accumulated to the Hough space. This article proposes using the Hough space as a 2D signal instead of just detecting the local maxima and processing them. On the example of QRcode detection, it is shown that this approach is computationally cheap, robust, and accurate. The proposed algorithm can be used for efficient and accurate detection and localization of matrix codes (QRcode, Aztec, DataMatrix, etc.) and chessboard-like calibration patterns.

Keywords Line detection · Regular grids · Hough transform · Real-time detection · PClines

1 Introduction

Straight lines are an important class of objects to be detected in raster images. Based on detected lines, compound patterns can be constructed. Lines which are parallel in the original 3D space and are perspectively projected to a 2D image intersect in a vanishing point (or are projected to parallel lines). Detection of such vanishing points has various applications such as analysis of aerial photographs [13], localization of calibration chessboard patterns [15], detection of 2D and 1D barcodes [11], etc.

A common approach in the detection of vanishing points (and corresponding groups of mutually parallel lines) is to detect lines in the image and among them to look for groups of lines coincident with one point [12]. When the Hough transform is used for detecting lines in the image, the detected lines are represented by points in the transformed space. The positions of such points in the Hough space can be analyzed by RANSAC (RANdom SAmple Consensus [6]) or again by the Hough transform [3]. However, when the common θ-ϱ parameterization [5] is used, the representations of points involved in one vanishing point do not lie exactly on one straight line.

Tuytellaars et al. [14] introduced the Cascaded Hough Transform (CHT) which allows multiple transformations from the original image space to the Hough space and back. Their CHT allows for detecting vanishing points and groups of vanishing points coincident with one line, etc. Dubská et al. [4] recently introduced another parameterization which is a point-to-line-mapping [2] and thus allows for the accurate detection of vanishing points as lines interconnecting the detected maxima in the Hough space.

However, the Hough transform (just as any other method) is not perfect in detecting lines in noisy and real-life images. For a given threshold, a number of lines are missed while a number of false-positives are reported. The data used for subsequent runs of the CHT are, therefore, noisy and sparse and the detection of vanishing points is not reliable.

In this paper we propose to analyze the Hough space (based on the parameterization by Dubská et al. [4]) as a 2D signal and search there for patterns which characterize the vanishing points. This approach thus uses information contained in the Hough space which is disregarded by the

M. Dubská · A. Herout (✉) · J. Havel
Brno University of Technology, Brno, Czech Republic
e-mail: herout@fit.vutbr.cz

CHT and similar approaches—and which turns out to be useful for the detection of vanishing points. This idea is used in a new algorithm for the detection of regular (chessboard) grids and for matrix codes such as QRcode, DataMatrix, etc. The experimental evaluation shows that the algorithm is very stable and accurate and is tolerant to high degrees of perspective deformation. At the same time it is computationally very cheap. This allows for a range of applications, especially in the detection and recognition of matrix codes and camera calibration.

The algorithm's performance is evaluated on the task of detecting QR codes. Detection of QR codes is of practical interest and they constitute one particular case of chessboard grids. We collected a dataset with QR codes skewed by perspective projection and surrounded by text and other challenging backgrounds. We are making this dataset publicly available in order to allow for comparison with alternative approaches to QR code detection—such a dataset was missing.

Section 2 builds upon the existing approaches in the detection of lines, extends the detection to (originally) equidistant and parallel lines and derives the necessary mathematical background. Section 3 describes the proposed algorithm and discusses its properties. Section 4 contains the experimental results.

2 Lines and vanishing points

The Hough transform is an effective method for detecting lines especially in noisy images. In the input image, pixels of interest are extracted, typically by an edge detector. Each of these pixels "votes" for all possible lines coincident with it. The voting process is implemented by an accumulator array in the Hough space. The Hough space is a two dimensional space whose axes correspond to two parameters of a line.

A popular way of parameterizing the lines is by using the $\theta - \varrho$ parameterization [5]. Several other parameterizations exist with different properties. Recently, Dubská et al. [4] presented PClines—a parameterization based on parallel coordinates [9] very similar to another recent parameterization PAT by Mejdani et al. [10]. PClines parameterize a point in the image space by a line in the Hough space, a point in the Hough space (naturally) represents a straight line in the image (Fig. 1). This parameterization is thus a Point-To-Line-Mapping (PTLM), studied and described by Bhattacharya et al. [2]. For more information on PClines and the related background, please refer to a short book by Herout et al. [8].

In the line parameterization based on parallel coordinates (PClines) the parameter space consists of two halves: straight $\mathcal{S}$ and twisted $\mathcal{T}$. Each line representation $\overline{\ell}$ lies in either of these halves. Each point $P = (P_x, P_y)$ in the original $x - y$ space can be coincident with a set of lines that is represented by two line segments in the $\mathcal{TS}$ space; the endpoints of the line segments in the $\mathcal{TS}$ space are $(-1, -P_x)$, $(0, P_y)$, and $(1, P_x)$—see Fig. 2. In the Hough transform, for each pixel of interest P, these two line segments need to be rasterized into the accumulator space.

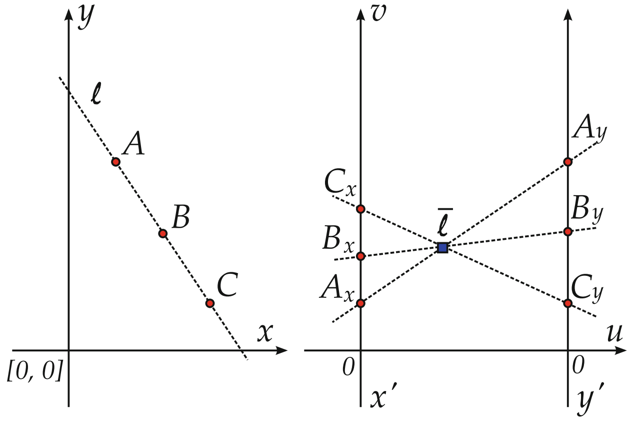

Fig. 1 Point-to-line mapping based on parallel coordinates. Representation of three collinear points in: (*left*) Cartesian space, (*right*) space of parallel coordinates. Line ℓ is represented by point $\overline{\ell}$ in parallel coordinates

2.1 Vanishing points in parallel coordinates

The PClines parameterization can be taken further, in order to allow for detection of regular grids. Let us consider a group of coplanar parallel lines. After perspective projection, lines from this group intersect at one point—the vanishing point. If the projected lines are still parallel after the projection, the vanishing point is an ideal point in infinity. Let us consider a 3D plane containing two or more groups of mutually parallel lines. Each of the groups of lines converge into a vanishing point and all these points lie on one line—the horizon [7].

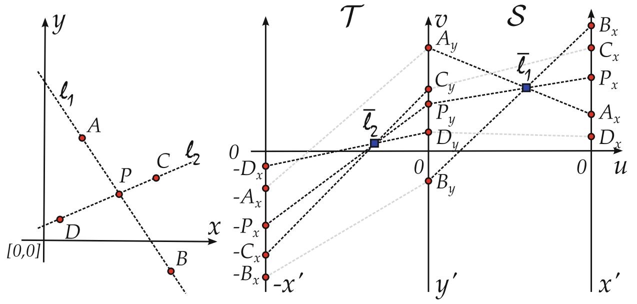

Fig. 2 *left:* Original image space $x - y$. *right:* $\mathcal{TS}$ space of the PClines. *Lines* ℓ_1 and ℓ_2 are transformed to points $\overline{\ell}_1$ and $\overline{\ell}_2$. Points P, A, B, C, D are transformed to couples of line segments. Representations of collinear points (A, P, B and C, P, D) intersect at the representations of lines $\overline{\ell}_1$ and $\overline{\ell}_2$

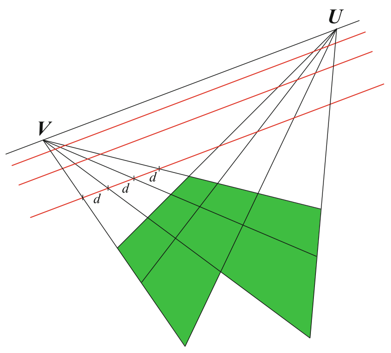

Fig. 3 Two groups of coplanar uniformly *spaced parallel lines* projected by perspective projection and their vanishing points *U*, *V*. *Red lines* are parallel to the *horizon*. Intersections of each *red line* with the *black lines* (from one pencil of lines) are uniformly spaced

In the case of planar grids or matrix codes projected perspectively from a 3D space to two dimensions, there are two groups of parallel lines, all coincident with a common plane. Two corresponding vanishing points *U*, *V* define the horizon line (Fig. 3).

It can be proven that intersections of any line parallel to the horizon and the projections of equally spaced parallel lines are also equally spaced (Fig. 3). This can be used for deriving relations between points representing projected (equally spaced) lines in parallel coordinates.

Let us have a group of equally spaced coplanar parallel lines projected to a 2D image plane: $\ell_1, \ell_2, \ldots, \ell_k$. Their images in parallel coordinates are points $\overline{\ell}_1, \overline{\ell}_2, \ldots, \overline{\ell}_k$ that lie on line $\overline{V}$ which is the representation of their vanishing point (Fig. 4). This contrasts with the $\theta - \varrho$ parameterization, where the concurrent lines lie on a sinusoid curve. It should be noted that PClines handle vanishing points in infinity (ideal points) naturally and easily—such a vanishing point is represented by a vertical line (which indeed intersects the axes of parallel coordinate system x' and y' in infinity). For any different lines ℓ_a, ℓ_b, ℓ_c, ℓ_d, the cross ratio is:

$$\frac{|\overline{\ell}_c\overline{\ell}_a| : |\overline{\ell}_c\overline{\ell}_b|}{|\overline{\ell}_d\overline{\ell}_a| : |\overline{\ell}_d\overline{\ell}_b|} = \frac{(c-a):(c-b)}{(d-a):(d-b)}, \tag{1}$$

where *a*, *b*, *c*, *d* are the indices of the four lines within the sorted set of lines $\ell_1, \ell_2, \ldots, \ell_k$ and $|\overline{\ell}_i\overline{\ell}_j|$ is the distance of points representing lines in the parallel coordinates. Because these representations lie on a straight line, the equation also holds for separate *u* and *v* coordinates of the lines' representations. This means that for arbitrary ℓ_a, ℓ_b, ℓ_c with known $(b-a)$ and $(c-b)$ it is possible to derive the rest of the uniformly spaced parallel lines with index *x*:

$$\begin{aligned} \alpha &= (c-a)(x-b), \\ \beta &= (c-b)(x-a), \\ u_x &= \frac{\alpha u_c u_a - \beta u_b u_c + (\beta - \alpha) u_a u_b}{\beta u_a - \alpha u_b + (\alpha - \beta) u_c} \\ v_x &= \frac{\alpha v_c v_a - \beta v_b v_c + (\beta - \alpha) v_a v_b}{\beta v_a - \alpha v_b + (\alpha - \beta) v_c}, \end{aligned} \tag{2}$$

where *a*, *b*, *c* are indices of arbitrary three lines in the set, *x* is the index of the searched line, (u_a, v_a), (u_b, v_b), (u_c, v_c) are known positions of lines' representations and (u_x, v_x) are the coordinates of the searched line's representation.

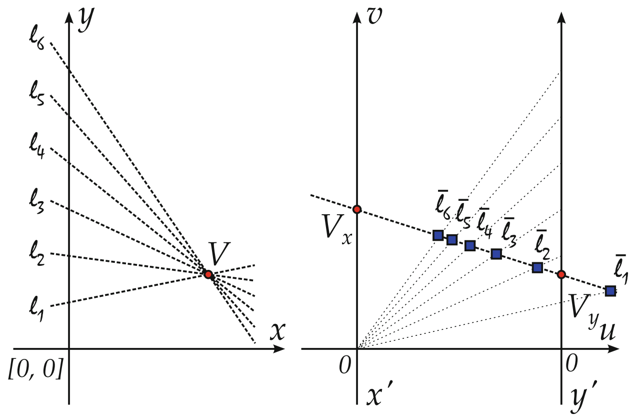

Fig. 4 *Left:* Set of equally spaced coplanar parallel lines projected to a 2D image plane. Their projections $\ell_1, \ell_2, \ldots, \ell_k$ and vanishing point *V*. *Right:* The same situation in parallel coordinates: $\overline{\ell}_1, \ldots, \overline{\ell}_k$ are points coincident with line $\overline{V}$ representing the vanishing point. Intersections of the *lines* with the *y* axis are marked with *dotted lines*

3 Detection of grids and matrix codes

The following algorithm is designed to detect a group of concurrent lines in the Hough space as one compound object. This contrasts to the common approach which is to first detect the lines and then based on them identify the vanishing point. The "standard" approach disregards much useful information contained in the Hough space which we intend to use for better detection of the vanishing points. The new algorithm consists of four main stages:

1. Extraction/selection of edges in the input image
2. Accumulation to the Hough space
3. Testing hypotheses about the vanishing points
4. Extraction of the matrix code or grid

These steps will be discussed in detail in the following subsections.

3.1 Extraction/selection of edges

Firstly, edges are detected in the input image by the Sobel edge detector. A small threshold $G_{\min}$ is set for their gradient magnitude *G* in order to suppress noise and very weak edges. A histogram with a low number of bins (e.g. 12) of the edge orientations θ is computed—see Fig. 5. Two maxima roughly 90° apart are detected in the histogram as the *dominant edge orientations*. These dominant orientations will be denoted as θ_h, θ_v, where h stands for horizontal and v for vertical; the naming is arbitrary and has no connection to the actual matrix code orientation.

In further processing, only the edges from the dominant histogram bins and their direct neighbors are used. Out of these edges, up to E best edges (with the highest gradient magnitude G) are selected for each dominant direction. This is done to avoid fine tuning of the $G_{\min}$ threshold. The parameter E is selected to be roughly proportional to the image area and its actual value does not influence the process notably, regardless of the image lighting and other factors.

3.2 Accumulation to the Hough space

For the dominant directions θ_h, θ_v two corresponding columns in the $\mathcal{TS}$ space u_h, u_v are determined [4]. Only two vertical columns (wide 2δ) of the $\mathcal{TS}$ space are accumulated for the ranges $(u_h - \delta, u_h + \delta)$ and $(u_v - \delta, u_v + \delta)$. Each of these parts is processed as if it lied completely in either the $\mathcal{T}$ or the $\mathcal{S}$ space (note that each of these spaces is infinite and it is only a convention that a finite part is used in the PClines). This means that even if u_h or u_v is close to any of the y', x', or $-x'$ axes (Fig. 2), only one line segment is always accumulated to the appropriate part.

See Fig. 6a for the two parts of the $\mathcal{TS}$ space accumulated for the two dominant directions from Fig. 5. In this figure, 16 highest local maxima are marked with red. These would be normally processed by the Cascaded Hough Transform or another algorithm so as to obtain the vanishing points. Note that this set of maxima is very small and noisy, which makes it difficult to process.

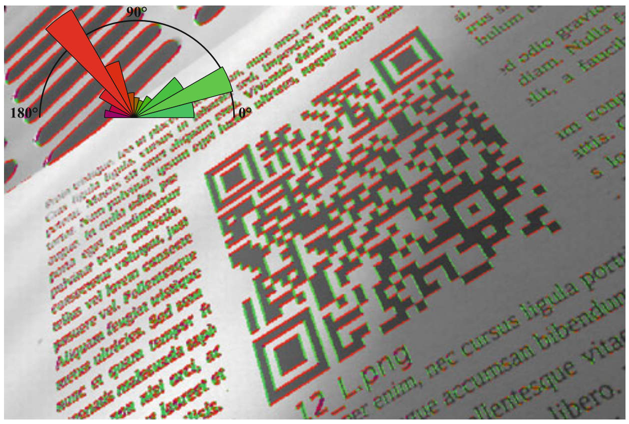

Fig. 5 Edge extraction and selection. Original QRcode image with detected edges. In the *left-top corner* histogram of edge orientations—two peaks can be found and only the relevant edges are selected for subsequent use

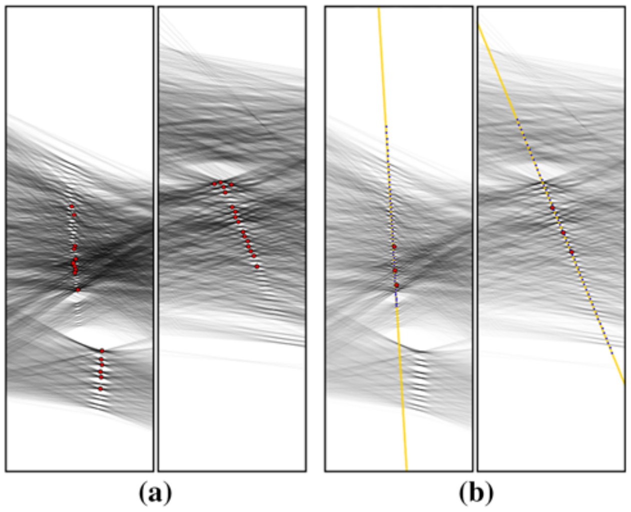

Fig. 6 **a** Two accumulation areas with 16 best found maxima. These 16 maxima are used for generating the hypotheses. **b** One generated hypothesis for each half of the $\mathcal{TS}$ space. A line is approximated through 3 random maxima. The $\mathcal{TS}$ space is searched along the *line* and a score of the hypothesis is computed

3.3 Testing hypotheses about vanishing points

Each of the two parts of the $\mathcal{TS}$ space is searched for the vanishing point and the corresponding group of projected parallel lines. First, N highest local maxima are found in the accumulator space (Fig. 6a). These maxima are used for generating hypotheses about the vanishing points and the corresponding groups of concurrent lines. One hypothesis is defined by 3 local maxima (i.e. triplets are drawn out of the N points). Since N can be a small number, typically 10 or 16, all combinations can be considered and the algorithm is deterministic, avoiding any randomized processing.

For each hypothesis defined by 3 points A, B, C, sorted by their vertical v coordinate, the following steps are taken (see Fig. 6b):

1. Calculate a hypothetical vanishing point line p best fitting points A, B, C using the least squares regression.
2. Count local maxima between A_p, B_p and B_p, C_p, where A_p, B_p, and C_p are projections of points A, B, C onto line p.
3. Determine positions of other maxima and minima on line p and compute the hypothesis' confidence score.

Let m be the number of maxima between A_p and B_p and n the number of maxima between B_p and C_p. Let us consider A_p as the 0th maximum (which makes B_p the $(m+1)$th maximum and C_p the $(m+n+2)$th maximum). The predicted coordinates for ith maximum or minimum can now be determined using Eq. (2). It should be noted that the indices i can be both negative and positive; positive values of i mean maxima on the half-line from A toward B and C, while negative values lay on the other half of the line and point A is the 0th maximum.

Each ith minimum represents a line between two neighboring edge lines in the input image. The confidence score for each hypothesis is designed to prefer high ith maxima and low ith minima along line p. Let $\max_i$ be the value of the accumulation space at the position of the ith maximum and $\min_i$ the value at the position of the ith minimum. The confidence score can be defined as:

$$S = \sum_{i=-b}^{t} (\max_i - \min_i). \tag{3}$$

Different borders $-b$ and t (bottom, top) define more hypotheses for one triplet A, B, C. Not all pairs (b, t) are tested as distinct hypotheses. Instead, a greedy growing algorithm is used, starting at $b = 0$ and $t = m + n + 2$ and increasing b and t until a suitable stopping criterion is reached.

The hypothesis with the best confidence score is selected for each of the two dominant directions (Sect. 3.1). Parameters of line p and positions of minima in the accumulation space define both the vanishing point and the set of lines coincident with this vanishing point which lie in the centers of the grid cells

It should be noted that the only configuration parameters of the algorithm are counts E and N. The algorithm is not very sensitive to their setting and they are completely independent of most properties of the input image. This contrasts with, for example, RANSAC which is in this case very sensitive to the threshold for the inclusion of inliers and other settings.

3.4 Extraction of the matrix code

The previous step outputs two groups of minima corresponding to two pencils of lines in the input image. These lines pass right through the middles of the matrix code black and white modules. By sampling the image at the intersections of such lines, a bitmap of the data matrix code can be extracted (Fig. 7). The resolution of this bitmap directly corresponds to the resolution of the matrix code: one pixel of this extracted bitmap directly represents one matrix code module. Processing of this bitmap is thus very efficient and its processing cost is negligible. The intersections are computed directly in the space of parallel coordinates in order to avoid the use of goniometric functions and similar precision sensitive operations.

The input image is sampled at the intersections of the lines corresponding to the minima—either the pixels directly (nearest neighbor) or a small neighborhood to

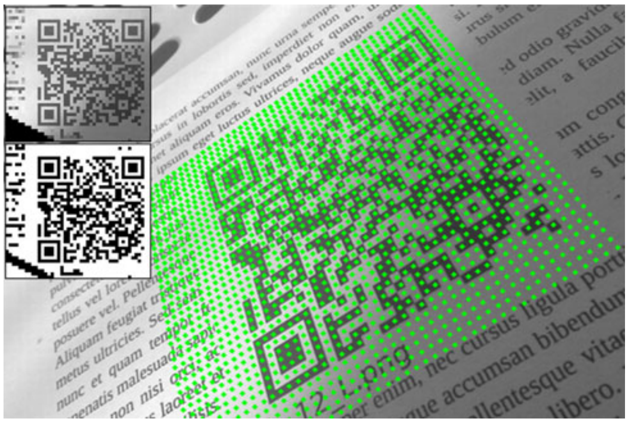

Fig. 7 Extraction of the matrix code. The *points shown in green* are sampled and form a bitmap which is then adaptively thresholded

achieve better stability. After extracting the grayscale image, adaptive thresholding is used for creating the binary bitmap of the matrix code (Fig. 7). A larger part than the matrix code is extracted to make sure that all modules of the matrix code are intact. The code is then searched for in this binary bitmap. This step is straightforward and computationally cheap.

The b and t limits for the sampling are selected conservatively so that a few lines outside the actual matrix code are sampled. The bitmap is searched for evidence of the code and the code is decoded—this operation is inexpensive and straightforward.

4 Experimental evaluation

The algorithm is evaluated on a dataset of challenging photos of QRcodes.[1] This dataset was collected and made public in order to allow for proper evaluation of perspective-invariant QR code detectors; the detailed results of our detector on each entry of this dataset are also made public. To our knowledge, this is the first QR code public dataset of this kind.

As a reference, we used the ZXing Project[2]—a popular open source barcode recognizer, having the position of an industry standard. The ZXing implementation is using the finder patterns of the QRcode to localize the code, which is the common and originally intended way of detecting the QRcodes. The ZXing seems to be the best publicly available comprehensive QR code detector; it can serve as a good benchmark for comparison with alternative approaches.

The comparison is targeted on QR codes that (1) cover a significant part of the input image (approx. 20 % or more)

[1] Downloadable from http://medusa.fit.vutbr.cz/pclines/.

[2] http://zxing.appspot.com/.

and (2) are planar. The dataset is constructed in this limited way for two reasons. Firstly, we are targeting on low-resolution cameras, such as the web cameras or cameras available in the mobile phones. Secondly, one important application of real-time (at least 25 fps) QR code detection is augmented reality with fiduciary markers. In the case of AR, planarity of the matrix code is necessary for accurate camera calibration. At the same time, the requirement for code planarity is easy to achieve. These limitations can be relaxed, as we mention in the following Sect. 5.

It should be noted that our algorithm is not using the QR codes' finder patterns for detection, but detects for a chessboard instead. Our algorithm is thus usable for a wider range of tasks, still outperforming the specialized QR code detector.

Section 4.1 contains an analysis of the computational complexity of different parts of the algorithm. Section 4.2 compares the detection rate of our algorithm and ZXing on the dataset of images. Section 4.3 compares our algorithm with ZXing on videos shot by a handycam—both the detection rate and the execution time.

4.1 Computational requirements of different parts of the algorithm

Table 1 shows the time breakdown into different parts of the algorithm. The times are average over 430 of the images used in the previous test. Different realistic values are used for the algorithm parameters N, E (Sect. 3.3). The times were measured on a single core of a Intel Core 2 DUO E8300, 2 × 1 GB DDR 400 MHz RAM. The programs were compiled by GCC 4.4.5 with speed optimizations.

The most time-consuming part by far is the edge detection by the Sobel operator performed together with computation of the histogram of edge orientations (Sect. 3.1). This part of the algorithm is not dependent on parameters E and N. All parts of the algorithm can be executed in

Table 1 Average computational time at different stages of the algorithm in milliseconds

N	10		10		16		16	
E	6,000		10,000		6,000		10,000	
E:	6.7	67 %	6.7	57 %	6.7	54 %	6.7	46 %
A:	1.8	18 %	3.1	27 %	1.8	15 %	3.1	21%
M:	0.6	6 %	0.8	7 %	0.6	5 %	0.8	6 %
H:	0.7	7 %	0.8	7 %	3.0	25 %	3.7	25 %
B:	0.2	2 %	0.3	2 %	0.2	2 %	0.3	2 %
$\sum$	10.0		11.7		12.4		14.6	

E edge Extraction, *A* hough space Accumulation, *M* Maxima detection, *H* Hypothesis testing, *B* Bitmap extraction

parallel, but this part of the algorithm is massively parallelizable by graphics chips and other technologies.

4.2 Detection rate on QRcodes

The images used in the evaluation are photos of QRcodes of different sizes. Since the QRcode decoders are differently sensitive to the rotation of the code and perspective deformation, the images are thus sorted into four categories:

plain the code is upright
rot the code is rotated
pers the code is upright but skewed by perspective
rot+pers general orientation of the code

Both our algorithm and the ZXing project are used for detection/localization of the code. If the code is detected, the binary bitmap of the code's blocks is extracted and compared to the ground-truth bitmap. Since the QRcode is capable of error repair and a few percent of the code can be missed, correct detection is counted when at least 95 % (99, 100 %) of the code is correctly extracted. We report two algorithm's settings: $E = 10{,}000$ and $N = 16$ (the "thorough one") and $E = 6{,}000$, $N = 10$ (the "fast one"). The images' resolutions were 600 × 400 or 640 × 360 (some are taken by a quality camera, some by a smartphone).

Table 2 shows that our algorithm is very well resistant to rotation and perspective deformation of the matrix code and outperforms the ZXing project. The ZXing implementation (representing the dominant approach to matrix code detection based on the finding patterns) fails for rotated and skewed images. The PClines parameterization is less accurate for rotated lines then for the vertical or horizontal lines [4], which partly explains why the rotated images have lower success rate in our algorithm. However, the lower performance of our algorithm in this case is mostly caused by blurred edges, where Sobel edge and gradient detector failed. Hence the Hough space is not accumulated perfectly and the probability of finding the vanishing point with corresponding group of lines is lower. The other issue of our algorithm manifests itself when the code is surrounded by intensive parallel edges. In such a case, the dominant edge direction includes these edges instead of the matrix edges and a wrong part of the Hough space is accumulated.

4.3 Experimental videos

We recorded five videos of different QRcodes by a handycam.[3] Table 3 shows the detection rates (in the same sense as in Sect. 4.2) and the total processing time of our

[3] Downloadable from http://medusa.fit.vutbr.cz/pclines/.

Table 2 Detection rate achieved by ZXing and our algorithm based on PClines

Data type (Count)	Our			Our			ZXing		
	$E = 6{,}000$, $N = 10$			$E = 10{,}000$, $N = 16$					
	100 %	99 %	95 %	100 %	99 %	95 %	100 %	99 %	95 %
Plain (113)	84.9	96.5	96.5	88.5	97.3	98.2	88.5	97.3	97.3
Rot (95)	72.6	91.6	95.8	68.4	94.7	98.9	53.7	58.9	62.1
Pers (110)	80.9	90	90.9	89.1	95.5	95.5	69.1	74.5	74.5
Rot+pers (112)	73.2	95.5	97.3	78.6	96.4	97.3	9.8	9.8	9.8
All (430)	77.9	93.5	95.1	81.6	96.0	97.4	55.3	60.2	60.9

The detection rates (percent of correctly detected images from the set) are reported for different notions of correct detection: 100%/99%/95% of code pixels detected correctly. As expected, for faster settings ($E = 6{,}000$, $N = 10$), our algorithm performs slightly worse than in the case of recommended settings ($E = 10{,}000$, $N = 16$). This degradation of results can be explained by the unability of the hypotheses to hit the actual code location. The computational performance for these cases is discussed in Table 1

Table 3 Results of five videos (resolution 960 × 540)

Video	Frames	Success rate Our 100/99/95%	Success rate ZXing 100/99/95%	Proc. time Our/ZXing
A	3,072	98/99/99	72/73/73	59/44
B	3,048	96/98/98	89/90/90	61/50
C	3,072	53/55/56	90/90/90	52/52
D	3,120	95/97/97	89/89/89	62/53
E	3,096	98/100/100	28/28/29	61/58

Detection rates are defined equally as in Table 2. Processing time is in seconds and includes video decoding. The settings are $N = 16$, $E = 10{,}000$ (the slowest ones)

algorithm based on PClines and of the ZXing implementation.

Our algorithm based on PClines outperforms the ZXing implementation notably in the detection rate. The main issue of our algorithm exhibits when the code is surrounded by intensive parallel edges (video C). We intend to solve this problem in future modifications of the algorithm (see Sect. 5) On the other hand, the ZXing implementation is very prone to fast changes of illumination (video E) where our algorithm works fine.

5 Future work

The algorithm described in this paper supposes that the matrix pattern spans over a significant portion of the image. If the matrix was much smaller, the histogram used in Sect. 3.1 could fail in detecting the dominant directions and the rest of the algorithm would fail as well. The observation made by Alfthan [1] can be used: for localization of the barcode, a lower image quality and simpler processing can be used, compared to its decoding.

However, since a majority of the computational time is spent on edge extraction (by the Sobel operator or a similar one), the extraction can be done once and the edges processed repeatedly on properly selected windows of the input image. Preliminary experiments show that this way even multiple matrix patterns (QRcodes) of arbitrary sizes could be detected in a high-resolution image. The computational time should not dramatically exceed the time necessary for the edge extraction step. This whole process is very suitable for parallelization using GPU chips. We used GPU for line detection and achieved a speedup over an order of magnitude compared to a CPU. Here, the edge extraction and accumulation is slightly different, but we expect that we can speed up two first parts of this algorithm roughly ten times, too. Also the following RANSAC steps can evaluate multiple hypotheses at the same time in multiple blocks. Our future research will, therefore, be targeted on a real-time detector/recognizer of matrix codes (such as the QRcode) in high-resolution images. Such a detector should be capable of finding codes of arbitrary sizes and rotations.

One important limitation of our algorithm is that it requires the matrix pattern to be perfectly planar—we are looking for lines that meet the concurrence condition. We intend to relax these requirements to some extent—both the concurrence requirement and the requirement for the edges within the matrix to be totally straight lines. The space of parallel coordinates allows for detection of such deformations and operations for their correction.

6 Conclusions

We presented a new algorithm for detecting regular grids and matrix codes. It is based on the Hough Transform, but in contrast to existing approaches it does not detect lines in the transformed space and then find concurrent groups among them. The paper introduces the calculations necessary to understand the representation of a projected regular grid in a point-to-line-mapping. Our approach treats the accumulated Hough space as a 2D signal and looks for evidence of the matrix as one compound object. That allows us to use information present in the Hough space which was disregarded by previous approaches. This information boosts the accuracy and stability of our algorithm.

We collected an annotated dataset of planar QR codes with different level of rotation/perspective distortion. This dataset should allow for evaluation of invariant QR code detectors—such a dataset was missing.

The presented solution is computationally very cheap—the majority of the computation time is consumed by the edge detection stage. The algorithm is suitable for hardware acceleration by graphics chips—both powerful desktop GPUs and the recent mobile ones. Even without acceleration, the algorithm performs in real time using moderate computational resources.

In near future, the algorithm will be extended to detect grids (matrix codes) of arbitrary sizes in high-resolution images. Such an algorithm is still expected to perform in real time. The algorithm's structure should match the requirements of contemporary graphics chips used in the GPGPU fashion.

Acknowledgments This research was supported by the CEZMSMT project IT4I—CZ 1.05/1.1.00/02.0070, by the TACR grant V3C TE01010415 and by MV CR grant VG20102015006.

References

1. Alfthan, J.: Robust detection of two-dimensional barcodes in blurry images. Master's thesis, KTH Computer Science and Communication, Stockholm (2008)
2. Bhattacharya, P., Rosenfeld, A., Weiss, I.: Point-to-line mappings as Hough transforms. Pattern Recognit. Lett. **23**(14), 1705–1710 (2002)
3. la Escalera, A., Armingol, J.M.: Automatic chessboard detection for intrinsic and extrinsic camera parameter calibration. Sensors **10**(3), 2027–2044 (2010)
4. Dubská, M., Herout, A., Havel, J.: PClines—line detection using parallel coordinates. In: Proceedings of CVPR (2011)
5. Duda, R.O., Hart, P.E.: Use of the Hough transformation to detect lines and curves in pictures. Commun. ACM **15**(1), 11–15 (1972)
6. Fischler, M.A., Bolles R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
7. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision, second edition. Cambridge University Press, Cambridge (2004). ISBN:0521540518
8. Adam H., Markéta D., Jiří H.: Real-Time Detection of Lines and Grids By PClines and Other Approaches. Springer Briefs in Computer Science. Springer, Berlin (2012). ISBN:978-1-4471-4413-7
9. Inselberg, A.: Parallel Coordinates; Visual Multidimensional Geometry and Its Applications. Springer, Berlin (2009)
10. El Mejdani, S., Egli, R., Dubeau, F.: Old and new straight-line detectors: Description and comparison. Pattern Recognit. **41**, 1845–1866 (2008)
11. Muniz, R., Junco, L., Otero, A.: A robust software barcode reader using the hough transform. In: International Conference on Information Intelligence and Systems, 1999. Proceedings, pp 313–319 (1999)
12. Schaffalitzky, F., Zisserman, A: Planar grouping for automatic detection of vanishing lines and points. Image Vis. Comput. **18**, 647–658 (2000)
13. Tuytelaars, T., Van Gool, L., Proesmans, M., Moons, T.: The cascaded hough transform as an aid in aerial image interpretation. In: Sixth International Conference on Computer Vision. pp 67–72 (1998)
14. Tinne, T., Marc, P., Luc Van, G., Esat, M.: The cascaded hough transform. In: Proceedings of ICIP (1998)
15. Zhongshi, W., Zhongge, W., Yingbin, W.: Recognition of corners of planar checkboard calibration pattern image. In: Control and Decision Conference (CCDC), Chinese, pp 3224–3228 (2010)

Author Biographies

Markéta Dubská received the MS degree at Faculty of Information Technology, Brno University of Technology, Czech Republic. She is currently a PhD student at Department of Computer Graphics and Multimedia at FIT Brno University of Technology. Her research interests include computer vision, geometry and computation using parallel coordinates.

Adam Herout received his PhD from Faculty of Information Technology, Brno University of Technology, Czech Republic, where he works as an associate professor and leads the Graph@FIT research group. His research interests include fast algorithms and hardware acceleration in computer vision and graphics.

Jiří Havel received the MS degree from the Faculty of Information Technology, Brno University of Technology, Czech Republic. He is currently a finishing PhD student at Graph@FIT research group at Brno University of Technology. His research interests include computer graphics, image processing, and functional programming.