# Point to Line Mappings and Other Line Parameterizations not only for Hough Transform

Jiří Havel

October 1, 2012

## Abstract

This works focuses on the Hough transform (HT). The HT is mostly used for the detection of lines or curves, but was also generalized for detection of arbitrary shapes.

The main theme of this work are line parameterizations, especially the Point-to-Line mappings. These parameterizations share the property, that a point in the image maps onto a line in the parameter space. This work presents proofs of some properties of PTLMs, notably the existence of a practical pair of PTLMs for line detection and the effect of a convolution in the image space on the contents of the parameter space.

Two realtime implementations of HT are presented in this work. Both accelerate HT using graphical hardware. One uses GPGPU API CUDA and the other the rendering API OpenGL. As an application of the line detection, this work

describes part of the detection of checkerboard marker usable for the augmented reality.

# 1 Introduction

This doctoral thesis focuses on line detection using HT, but it mostly deals with one aspect of the detection – the parameterization of a straight line. Although the mathematical desription of a straight line or a line segment is simple and straightforward, many radically different parameterizations with various properties exist.

This work examines in detail a subset of line parameterizations – the Point to Line Mappings (PTLMs). These parameterizations have an interesting property, that the set of lines that pass through a given point map to a set of points in the parameter space, that form a straight line. This property can significantly simplify the Hough transform implementation. Also lines are a common graphical primitive, so many fast rasterization algorithms exist and comodity GPUs can accelerate line rasterization.

This work introduces methods usable for fast line detection. The implementations presented in this thesis achieve realtime detection rates even for full HD input video.

This work can allow for uses of HT in atypical manners. It examines several corner cases of the PTLMs. These are probably not useful for some ordinary line detection. They however provide some insights and deeper understanding to the behavior of HT. An example may be the construction of a line parameterization that is most precise for a specific line orientation.

Also researchers, that want to detect perspectively distorted checkerboard-like patterns in picture may be interested in this work. The line detection is an important part of the detection of checkerboard patterns. This work also sketches out the basic principles of the detection of a projected chckerboard. With my colleagues, we are bulding on these principles and we are developing algorithms for fast and reliable detection of chessboard-like structures.[10, 18]

## 2 Objectives

The first objective of this work was to extend the theoretical knowledge about Point-to-Line Mappings. The chapter 4 of my dissertation deals with usability of various PTLMs for line detection and with its relationship to connvolution. This theoretical work provided an basis for the real-time line and marker detection.

The second objective was to provide a fast way to perform line detection. Because the line detection is only one part of some interresting machine vision algorithms, the detection must be performed in real time with a performance reserve. The new line parameterization PClines and the GPGPU implementations of Hough transform do fulfill this objective.

## 3 Hough Transform

The Hough Transform (HT) [11] 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. [17] 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.

HT is typically used for detecting curves with an analytical description. In that case, the evidence are edge points detected in the input raster image. Such edge points can typically be detected by gradient operators such as Sobel or Prewitt. The hypotheses are the possible curves of a given class in the image. For example, a line has two and a circle three degrees of freedom in a 2D space, but HT can be used for detection of objects such as hyperspheres or hyperplanes in spaces of arbitrary dimensionality. Algorithm 1 shows the detection of an implicit curve by the HT.

Shapes that do not have a simple analytical description can be detected by using the Generalized Hough Transform by Ballard [1]. In GHT, the object is not described by an equation but by a set of contour elements (edge points). Each contour element is described by its position with respect to the object reference point and the edge orientation. The parameter space has a dimension from two to four (object position, orientation and scale), but the representation of the detected object is complex even for simple shapes.

**Algorithm 1** Implicit curve detection by Hough Transform.

**Require:** Input image $I$, size of parameter space $H$
**Ensure:** Detected curves $C$

```
P_I = {(x, y) | (x, y) are coordinates of a pixel in I}
P_H = {(p_1, ··· , p_N) | (p_1, ··· , p_N) are coordinates in H}
H(x) ← 0, ∀x ∈ P_H
for all x ∈ P_I do
  if at x is an edge in I then
    for all {p ∈ P_H | f(x, p) = 0} do
      H(p) ← H(p) + 1
    end for
  end if
end for
C = {p ∈ P_H | at p is a high local maximum in H}
```

## PClines

In [5], together with Markéta Dubská and Adam Herout, I have used parallel coordinates as a line parameterization for the Hough transform. A similar parameterization was independently proposed by Mejdani et. al. [15].

Parallel coordinates [13] are mostly used for vizualization of multidimensional data in two dimensions without the projection of the data to the 2D space. The parallel coordinate system represents a given vector space by axes which are mutually parallel. Each $N$-dimensional vector is then represented by $N-1$ lines connecting the axes. To define the position of points in the space of parallel coordinates, this space will also have a 2D cartesian coordinate system $u$-$v$, and homogeneous cordinates $(u, v, w)$.

In the two-dimensional case, points in the $x$-$y$ space are represented as lines in the space of parallel coordinates. Representations of collinear points intersect at one point – the representation of a line (see Fig. 1). Based on this relation,

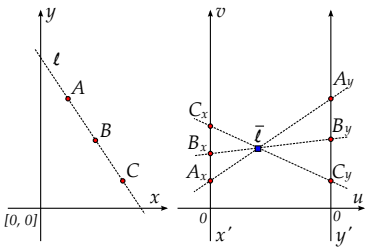


Figure 1: Three collinear points in parallel coordinates: (left) Cartesian space, (right) space of parallel coordinates. Line $\ell$ is represented by point $\overline{\ell}$ in parallel coordinates.

it is possible to define a point-to-line mapping between these spaces. For some cases, such as line $\ell : y = x$, the corresponding point $\overline{\ell}$ lies in infinity (it is an ideal point).

Because for some lines (e.g. $\ell : y = x$) the image in the space of parallel coordinates lies in infinity, it is necessary to construct a pair of parameter spaces to detect lines of all angles. Mejdani et. al. [15] and we in our paper [5] used slightly different approaches. In our approach, the PC based representation of line $\ell : y = mx+b$ in the $u$-$v$ space is $\overline{\mathbf{l}} = (d, b, 1-m)$, where $d$ is the distance between the parallel axes $x'$ and $y'$. The line's representation $\overline{\mathbf{l}}$ lies between the axes $x'$ and $y'$ if and only if $-\infty < m < 0$. For $m = 1$, $\overline{\mathbf{l}}$ is an ideal point (a point in infinity). For $m = 0$, $\overline{\mathbf{l}}$ lies on the $y'$ axis, for vertical lines ($m = \pm\infty$), $\overline{\mathbf{l}}$ lies on the $x'$ axis.

Besides this space of parallel coordinates $x', y'$ (further referred to as *straight*, $\mathcal{S}$), we proposed a *twisted* ($\mathcal{T}$) system $x', -y'$, which is identical to the straight space, except that the $y$ axis is inverted. In the twisted space, $\bar{\mathrm{l}}$ lies between the axes $x'$ and $-y'$ if and only if $0 < m < \infty$. By combining the *straight* and the *twisted* spaces, the whole $\mathcal{TS}$ plane can be constructed, as shown in Fig. 2.

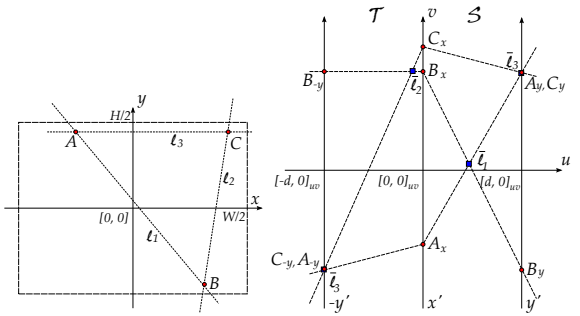


Figure 2: (left) Original $x$-$y$ space and (right) its PClines representation – the corresponding $\mathcal{TS}$ space.

Consequently, any line $\ell$ has exactly one image $\overline{\ell}$ in the $\mathcal{TS}$ space; except for cases that $m = 0$ and $m = \pm\infty$, when $\overline{\ell}$ lies in both spaces on $y'$ or $x'$, respectively. That allows the $\mathcal{T}$ and $\mathcal{S}$ spaces to be "attached" one to another. Figure 2 illustrates the spaces attached along the $x'$ axis. Attaching also the $y'$ and $-y'$ axes results in an enclosed Möbius strip.

## Comparison

In [5], we compared $\mathcal{TS}$ (with $(-y, x, y)$ arrangement), $\theta$-$\varrho$ and slope-intercept parameterizations in terms of precision.

In this evaluation, automatically generated data (similar to the data used in [12]) were used. The black-and-white image (sized $W \times H$) were genereated by first rasterizing $L$ lines directly from the line equation in its normal form; 8-connected neighborhood of pixels was used. Then, $P$ noise pixel positions were randomly generated $p_i \in \{0, \ldots, W-1\} \times \{0, \ldots, H-1\}$, and the corresponding pixels were inverted in the image.

Two errors of the detections were evaluated: $\varepsilon_\theta$ and $\varepsilon_\varrho$ which are the differences from the ground truth in degrees or pixels, respectively. To obtain one error metric, a combined error

$$\varepsilon = \sqrt{\omega_\theta \varepsilon_\theta^2 + \omega_\varrho \varepsilon_\varrho^2} \tag{1}$$

is used, with weights $\omega_\theta = \omega_\varrho = 1$. The accumulator space had the same dimensions for all three methods: for $512 \times 512$ images, accumulator space $768 \times 724$ was used.

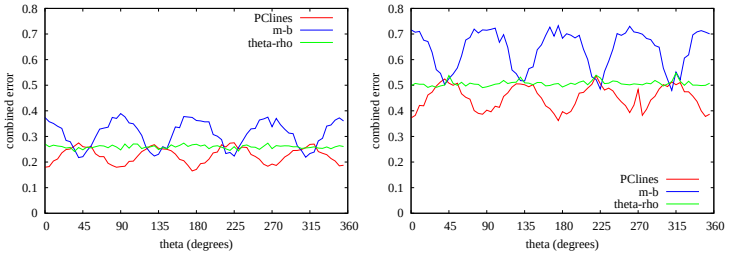


Figure 3: Line localization error as it depends on the lines' slope. For $x$ on the horizontal scale, the lines' slope in degrees is at interval $[x, x+5[$. Red: PClines; Green: $\theta$-$\varrho$; Blue: $m$-$b$. Left: average error over all lines; right: average error of the 5 least accurate lines, i.e. a pessimistic error estimation.

In this measurement, PClines seem the most accurate, while

the $m$-$b$ the least. I am convinced, this is caused by the fact that both $\theta$-$\varrho$ and $m$-$b$ parameterizations do not use some portions of the parameter space, while the PClines completely utilize the assigned memory.

# 4 Point to Line Mappings

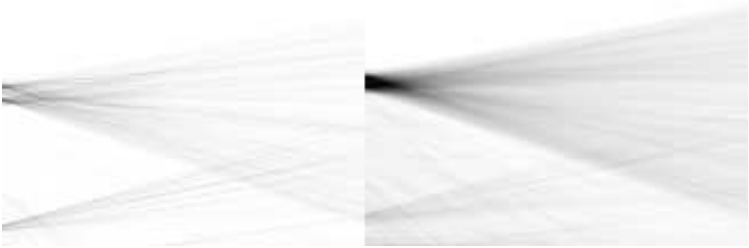

Figure 4: Detail of a parameter space before (left) and after (right) smoothing by vertical convolution. The maxima is no longer broken to separate intersections.

Point-to-line mappings (PTLMs) are a special case of line parameterizations that map points (or precisely all lines that pass though this point) in the $x$-$y$ space to a *line* in the parameter space. Parametrizations that fall into this class are the slope-intercept, $\gamma$-$\omega$ and parallel coordinates based parametrization. Some properties of these mappings are discussed and proven by Bhattacharya et al. [2]. My work [9] extends it. Figure 4 illustrates smoothing of the parameter space by a technique described in this section.

To be useful in Hough transform, the PTLMs must be bijective and map collinear points (points that lie on a line) to concurrent lines (lines that intersect at a single point).

**Theorem 4.1** (Bhattacharya 1)**.** *A 1-1 PTLM takes collinear points into concurrent lines iff it is linear.*

The proof comes from the Fundamental Theorem of Projective Geometry as shown in [2]. In other words, this theorem means, that PTLMs can be represented by matrix multiplication.

The second Bhattacharya theorem generalizes the issue with the original Hough's parameterization $m$-$b$, $y = mx + b$ that the parameter $m$ of vertical lines is infinite.

**Theorem 4.2** (Bhattacharya 2)**.** *A 1-1 PTLM cannot map all the sets of collinear points that lie in a bounded region into sets of concurrent lines whose intersections lie in a bounded region.*

Bhattacharya et al. [2] mention a solution of this problem by using a parameter space that is composed of two finite parts. All commonly used PTLMs such as $m$-$b$ or the PClines form such pairs. However, Bhattacharya's paper does not present any proof that a second mapping exists for every linear 1-1 PTLM or if some additional conditions must be fulfilled for existence of such a pair.

A pair of PTLMs $f$ and $g$ that can be used for line detections will be called a *Complementary PTLM Pair*. Also, $g$ is the complementary mapping for $f$ and vice versa. It should be noted in advance, that some PTLMs have complementary mappings only for some bounded regions in the image space.

The necessary condition for an PTLM pair $f$ and $g$ to be usable for line detection is that for any bounded region $R$ in the image space, there must exist a bounded region for both of the parameter spaces, that images of all lines through $R$ lie at least in one of those bounded regions. It is not necessary for those regions to be identical.

Since every two PTLMs share a common problematic line, this line must not intersect the bounded region $R$. Ideally, this line does not intersect any bounded region, which means it is the ideal line in infinity.

**Theorem 4.3.** *A linear 1-1 PTLM $f(\mathbf{x}) = F\mathbf{x}^T$ has at least one complementary mapping $g$ for a circular region with radius $r$ around the origin iff the third row $(u, v, w)$ of $F^{-T}$ has the property that*

$$\left| \frac{\sqrt{u^2 + v^2}}{w} \right| > r. \tag{2}$$

Two special cases of $F'_{3,*}$ exist :

$(u, v, 0)$ This PTLM has a complementary mapping for every $r$. When the ($G'_{3,*}$ also lies in the $ab$ plane, their common problematic line intersects the cylinder at infinity. Any pair of such PTLMs has the problematic line outside of every bounded region.

$(0, 0, w)$ This mapping maps lines passing through the origin to the ideal line. It does not have a complementary mapping because no single matrix can map all lines passing through the origin to a bounded region. At least two additional mappings are required.

**Corollary 4.4.** *For a linear 1-1 PTLM $f$ with $(F^{-T})_{3,3} = 0$ exists a complementary PTLM for every finite size of the image space region.*

All commonly used PTLMs such as *m-b* [11, 15], Tuytelars CHT [19] or the PAT/PClines [5, 15] do have this property. The third mapping in the Tuytelaars parameterization is the second special case ($F'_{3,*} = (0, 0, 1)$).

For every two PTLMs exists at least one line, that is mapped to infinity by both of them. To represent all lines in $\mathbb{R}\mathrm{P}^2$, three PTLMs are necessary. CHT by Tuytelaars et. al. is an example.

**Corollary 4.5.** *Three linearly-independent 1-1 PTLMs map every set of collinear points onto a set of concurrent lines whose intersection lies inside a bounded region for at least one of those mappings.*

## Convolution

The projection-slice theorem is easily applicable on Hough transform using $\theta$-$\varrho$ parametrization as

$$\mathcal{H}_\theta[f * g](\varrho) = (\mathcal{H}_\theta[f] * \mathcal{H}_\theta[g])(\varrho). \tag{3}$$

In the case of PTLMs, the necessary conditions are more stricter than mapping collinear points onto concurrent lines.

**Lemma 4.6.** *The convolution in the image space can be transformed to the convolution in the parameter space if PTLM $h$ has*

$$H = \begin{pmatrix} a & b & 0 \\ 0 & 0 & c \\ d & e & 0 \end{pmatrix}, \tag{4}$$

*with*

$$c \neq 0, \ \textit{and} \tag{5a}$$
$$ae \neq bd, \tag{5b}$$

*because of the invertibility of* $H$*.*

These conditions are sufficient but might not be necessary. The conditions (5) come directly from the determinant of the matrix $H$.

**Theorem 4.7.** *For an convolvable PTLM* $h$*, the 2D convolution in the image space can be expressed as a 1D convolution in the parameter space as*

$$\mathcal{H}_u[f*g](v) = \left| \frac{c}{\sqrt{(e+ub)^2 + (d+ua)^2}} \right| \int_{-\infty}^{\infty} \mathcal{H}_u[f](s)\mathcal{H}_u[g](v-s)ds. \tag{6}$$

*where* $a \ldots, e$ *are values from* $H$ *according to Lemma 4.6.*

# 5 Grids and Markers

This section presents an application of the various line parameterizations, mostly PClines. It was used for the detection of checkerboard-like pattern for the camera localization in augmented reality. This section describes a relevant subset of papers [10, 18]. The papers introduce uniform and fractal marker fields that are usable for camera localization and augmented reality. Figure 5 illustrates these marker fields.

Due to problems such as nonuniform lighting, edges are easier to detect, than the uniformly colored squares. The edges

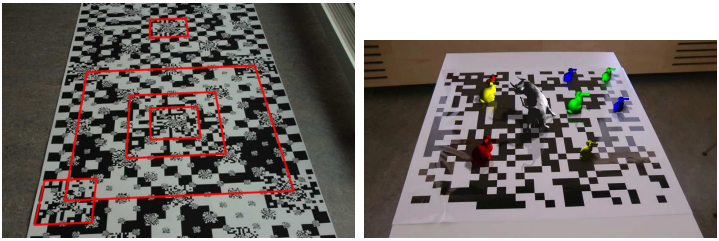

Figure 5: Illustration of checkerboard-like marker fields. Left: Fractal Marker Field; Right: Uniform Marker Field.

of the checkerboard squares form two perpendicular sets of equidistantly spaced parallel lines. The conditions for perpendicularity and equidistant spacing are not necessary.

Points on a set of parallel lines in $\mathbb{R}^3$ can be expressed as

$$\mathbf{p} = \mathbf{p}_0 + s\mathbf{u} + t\mathbf{v}. \tag{7}$$

Both $\mathbf{u}$ and $\mathbf{v}$ can be used as the line direction. One of $s$ and $t$ does belong to $\mathbb{R}$ and the other is generates by some step function

$$\text{step}_{\{u,v\}} : \mathbb{Z} \to \mathbb{R}. \tag{8}$$

For description of checkerboards QR codes and similar patterns or our uniform markers [18], two types of the step function are usefull.

The first has the form

$$\text{step}_{\{u,v\}}(i) = k_{\{u,v\}} i, \text{ for some } k_{\{u,v\}} \in \mathbb{R}^+. \tag{9}$$

This is THE step function for checkerboards and all other uniform grids. The size $s_{\{u,v\}}$ of the grid square in $\mathbb{R}^3$ is

$$s_u = k_u \|\mathbf{u}\| \quad s_v = k_v \|\mathbf{v}\|. \tag{10}$$

Even more useful is a slight modification of the step function (9) to the form

$$\text{step}_{\{u,v\}}(i) = k_{\{u,v\}}i + q_{\{u,v\}}, \text{ for some } k_{\{u,v\}} \in \mathbb{R}^+, q_{\{u,v\}} \in \mathbb{R}. \tag{11}$$

The step function (11) allows to base the grid on an arbitrary point or line. The function (9) requires that $\mathbf{p}_0$ and $\mathbf{l}^{(0)}_{\{u,v\}}$ are members of the grid. Using the function (11), it is no longer necessary.

In $\mathbb{R}^3$ it may be usefull to place an origin of the grid in the center of one square and use the origin also for $\mathbf{p}_0$, but it does not simplify the situation much. However it significantly simplifies the situation in $\mathbb{R}\text{P}^2$. It is possible to use completely arbitrary point such as center of the screen for $\mathbf{p}_0$.

The homogeneous coordinates of the lines of the grid are described by equation (12). The line is a weighted sum of two lines. The "first" line $\mathbf{l}_u(0)$ that connects $\mathbf{p}_0$ and the vanishing point $\mathbf{v}$ and the line $\mathbf{h}$, that connects both vanishing points.

$$\mathbf{l}_u(i) = \mathbf{l}^{(0)}_u + \text{step}_u(i)\mathbf{h}, \tag{12a}$$

$$\mathbf{l}_v(i) = \mathbf{l}^{(0)}_v + \text{step}_v(i)\mathbf{h}. \tag{12b}$$

Line $\mathbf{h}$ is common for both directions of the grid edges and is called the *horizon*. It is the intersection of the plane in which the lines lie with the image plane. The lines of both the sets approach the horizon as the value of the step function approaches infinity.

# 6 Grid Detection

The main problem in the grid detection is finding of the parameters of the whole grid from a set of imprecisely detected

lines. The line detection step can be done by many methods, not only by the Hough transform. The grid parameters can be found in the parameter space of the HT or from the parameters of the detected lines.

The grid has 8 degrees of freedom plus the number of DoF from the step function, whether it is described by equation (7) or (12). This is caused by the fact that the resulting points or lines can be scaled arbitrarily, but only as a whole equation.

Although it is possible to solve the whole system of equations for all unknowns, the problem can be split to much simpler parts. First of all, the two vanishing points and the corresponding fans (or more precisely pencils) of lines can be detected almost independently. Second, it is possible to find the vanishing points first and then search for the $\mathbf{l}(0)$.

From the fact that all lines of the fan pass through the vanishing point, it is clear that the vanishing point can be detected directly from the line parameters. Theoretically only two lines are required for the localization of the vanishing point, but due to the errors and imprecisions (imprecisely detected lines and false detections) the lines of the fan do not intersect in one point. Aditionally, small error in the angle of the detected lines can lead to a large error in the vanishing point location, because when the projective distortion is small, the vanishing points lie almost in infinity.

It is possible to use RANSAC to find the best pair of lines, or at least to remove the most significant outliers. The vanishing point can be found from all lines as a solution of an overspecified system of equations. Given a set of lines $\{\mathbf{l}_1, \ldots, \mathbf{l}_N\}$,

the vanishing point $\mathbf{v}$ can be found by solving equation

$$L\mathbf{v}^T = \begin{pmatrix} \mathbf{l}_1 \\ \vdots \\ \mathbf{l}_N \end{pmatrix} \mathbf{v}^T = \mathbf{0}^T, \tag{13}$$

which just means that every line passes through the point $\mathbf{v}$. When there are more than two lines, the system is overspecified (has more equations than unknowns). Of course, because of imprecisely detected lines, no accurate solution exists. The vanishing point must be found as a least square error solution or in a similar manner.

The vanishing points correspond to the hyperplanes through the origin. This is the geometrical meaning of equation (13). The vanishing point can therefore be found by hyperplane fitting, for example by uncentered PCA. By eigendecomposition of the correlation matrix

$$C = \begin{pmatrix} \mathbf{l}_1^T \dots \mathbf{l}_N^T \end{pmatrix} \begin{pmatrix} \mathbf{l}_1 \\ \vdots \\ \mathbf{l}_N \end{pmatrix}, \tag{14}$$

three principal components are found. The component with the least variance (eigenvalue) is the hyperplane normal and a good approximation of the desired vanishing point.

If the step function has the form

$$\text{step}(i) = f(i) + q, \tag{15}$$

such as (11) it is possible to use $\mathbf{l}^{(0)}$ that is not actually a line of the fan. Almost any line through the vanishng point

is usable. A line through the center of the image is a possible choice. Using step function (11), a fan of an uniform grid is

$$\mathbf{l}_{\{u,v\}}(i) = \text{normalize}(\mathbf{o} \times \{\mathbf{v}, \mathbf{u}\}) + (ki + q)\text{normalize}(\mathbf{u} \times \mathbf{v}) \tag{16}$$

The normalization again uses an $\ell_2$ norm of the underlying vector space.

For each imprecisely detected line $\mathbf{l}(i)$ it is possible to solve this overspecified system to find the value of $(ki + q)$. The values $k$ and $q$ can be then found by linear regression after clustering of the detected lines to get its indices $i$.

# 7 HW Implementation

The classical $\theta$-$\varrho$ Hough transform was implemented on graphical hardware by Diard [3] and Fung et. al. [7, 6], but the common graphical APIs can not directly rasterize sinusoid curves. Fung's implementation rasterizes the sinusoid curve as a polyline and Diards implementation rasterizes several quads, that span larger portion of the parameter space.

The family of point-to-line mappings seems therefore to be suitable for hardware accelerated Hough transform, because rasterization of lines is a widely supported graphical operation. Another way could be through the GPGPU capabilities of modern graphical hardware that allows acceleration of almost any parallel algorithm.

With my colleagues, I tried and compared both ways. We used OpenGL to accelerate our parallel coordinate based Hough transform [4]. Using GPGPU API CUDA, we accelerated the Hough transform with PClines and classical $\theta$-$\varrho$ parameterization and compared the results [14, 8]. The implementation

details can be found in my thesis or papers, here I will shortly summarize the measured performance.

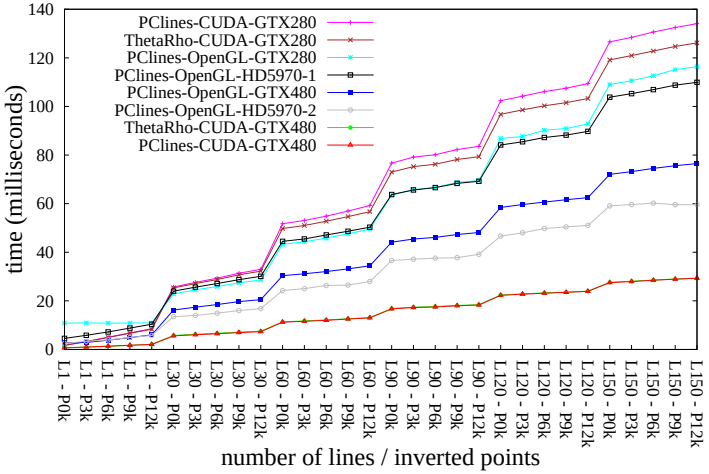


Figure 6: Performance evaluation on generated data.

Figure 6 shows the performance os sythetic images. These images were generated by rasterization of several random lines and the images were then distorted by a random noise.

Figure 7 shows the performance on real-life images.

On current graphics chips the performance of the sliding window algorithm perform equally fast for both $\theta - \varrho$ and the PClines line parameterization (it should be noted that in Figures 7 and 6 their curves totally overlap). On special, embedded, and low-power architectures the PClines-based version may perform much better or can be the only feasible one. That is because it requires no floating-point computations and

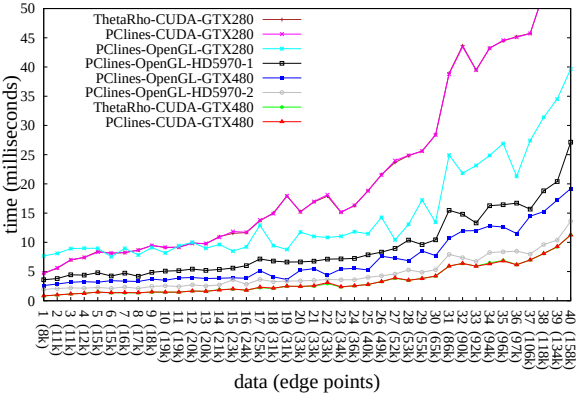


Figure 7: Performance evaluation on real-world images.

no goniometric functions (which are cheaply available on the GPUs). The only advantages of the PClines-based algorithm on GPU is, therefore, its better accuracy [5] and its ability to directly detect parallel lines and sets of lines coincident with one point.

Figures 7 and 6 show that on the pre-Fermi NVIDIA card (GTX280), the OpenGL version of the PClines-based Hough transform performs better than CUDA. That is because the atomic increment operation (`atomicInc`) in the shared memory is not optimized on this generation of the graphics chips. Very good results also come from recent Radeon graphics chips (with the OpenGL version).

The Fermi architecture (compared to the previous generation) speeded up the algorithm in the OpenGL version just the amount which can be expected from the increase in the number of the streaming multiprocessors. However, the CUDA ver-

sion presented in this paper speeded up notably more (about 4 times) on the Fermi architecture. This can be explained by the improved atomic operations in the shared memory, involving the new design of the L2 cache on the GTX480 [16]. Attribution of the performance boost between the GTX280 and GTX480 to the atomic instructions was verified by running the algorithm with the non-atomic equivalents of the increment/add instructions. Such a modified program achieved a speedup corresponding to the number of processing cores on the graphics boards. The atomic instructions are used in both the edge extraction and sorting phase and in the phase of accumulation into the Hough space. Therefore, the implementation of our algorithm uses the atomic instructions heavily and the improvement present in the Fermi architecture is beneficial.

# 8 Conclusion

In my thesis, I summarized the Hough transform and its usage for detection of straight lines. The main theme were the line parameterizations with focus on their subset – the Point to Line Mappings. A new parameterization (PClines), that belongs to this group was introduced.

The hough transform was described as a modified integral transform and the relationships with other common integral transforms were analyzed. The most important integral transforms, that relate to the Hough transform are the Radon and Fourier transform.

I summarized most of the line parameterizations used with the Hough transform. Selected parameterizations were compared with respect to the precision of the detection. In this

comparison, the new PClines parameterization seems to be the most precise one. For some line slopes, it is similarly precise to the most commonly used $\theta$-$\varrho$ parameterization and outperforms it for the rest.

For the PTLMs, I extended Bhattacharya's work by finding conditions under which the PTLMs can be used for line detection, i.e. the conditions under which a pair of PTLMs can describe all lines in an image. I also found the subset of PTLMs for which a 2D convolution in an image space can be transformed to a 1D convolution in the parameter space.

With colleagues, I provided a realtime line detector that uses GPU implementation of the HT. Two variations were implemented. One uses the renderig API OpenGL and its geometry shaders. This implementation is very simple and straightforward, due to the convenience of PTLMs. The other uses GPGPU API CUDA. While it is more complex, it allows the use of non-PTLM parameterizations. The performance of both implementations was tested on synthetic and real-life images. Both implementations allow realtime detection in full HD images on common graphical hardware. The CUDA implementation is slightly faster on the same hardware, but more complex.

As an application of the line detection, this work describes the detection of checkerboard-like patterns. Two aspects of this detection are examined in detail. First, the mathematical description of a perspectively projected checkerboard or more specifically its edge lines, was derived. Second, the behavior of the checkerboard lines was examined, when the lines were detecter using HT with PTLM line parameterization. This work is a part of a marker detection code that will be used for

the applications of augmented reality.

## References


[1] D. H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[2] Prabir Bhattacharya, Azriel Rosenfeld, and Isaac Weiss. Point-to-line mappings as hough transforms. *Pattern Recognition Letters*, 23(14):1705 – 1710, 2002.

[3] Franck Diard. *Using the Geometry Shader for Compact and Variable-Length GPU Feedback*, chapter 41. Addison-Wesley, 2008.

[4] Markéta Dubská, Jiří Havel, and Adam Herout. Real-Time Detection of Lines using Parallel Coordinates and OpenGL. In *Proceedings of SCCG 2011*, page 7. Comenius University in Bratislava, 2011.

[5] Markéta Dubská, Adam Herout, and Jiří Havel. PClines - Line Detection Using Parallel Coordinates. In *Proceedings of CVPR 2011*, pages 1489–1494. IEEE Computer Society, 2011.

[6] James Fung. *Computer Vision on the GPU*, chapter 40. Adison-Wesley, 2005.

[7] James Fung, Steve Mann, and Chris Aimone. Openvidia: parallel gpu computer vision. In *Proceedings of*

*the 13th annual ACM international conference on Multimedia*, MULTIMEDIA '05, pages 849–852, New York, NY, USA, 2005. ACM.

[8] Jiří Havel, Adam Herout, Markéta Dubská, and Radovan Jošth. Real-Time Detection of Lines using Parallel Coordinates and CUDA. *Submitted to Journal of Real-Time Image Processing*, 2012.

[9] Jiří Havel, Adam Herout, and Markéta Dubská. Vanishing Points in Point-to-Line Mappings and Other Line Parameterizations. *Submitted to Pattern Recognition Letters*, 2012.

[10] Adam Herout, Michal Zachariáš, Markéta Dubská, and Jiří Havel. Fractal Marker Fields: No More Scale Limitations for Fiduciary Markers. *Accepted to ISMAR*, 2012.

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

[12] J. Illingworth and J. Kittler. The adaptive hough transform. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9:690–698, May 1987.

[13] Alfred Inselberg. *Parallel Coordinates; Visual Multidimensional Geometry and Its Applications*. Springer, 2009. ISBN: 978-0-387-21507-5.

[14] Radovan Jošth, Markéta Dubská, Adam Herout, and Jiří Havel. Real-Time Line Detection Using Accelerated High-Resolution Hough Transform. In *Proceedings of SCIA 2011, LNCS*, pages 784–793. Springer Verlag, 2011.

[15] S. El Mejdani, R. Egli, and F. Dubeau. Old and new straight-line detectors: Description and comparison. *Pattern Recognition*, 41(6):1845 – 1866, 2008.

[16] NVIDIA Corporation. *Fermi Compute Architecture Whitepaper*, 2009.

[17] J. Princen, J. Illingowrth, and J. Kittler. Hypothesis testing: A framework for analyzing and optimizing Hough transform performance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(4):329–341, 1994.

[18] István Szentandrási, Michal Zachariáš, Jiří Havel, Adam Herout, Markéta Dubská, and Rudolf Kajan. Uniform Marker Fields: Camera Localization By Orientable De Bruijn Tori. *Accepted to ISMAR*, 2012.

[19] Tinne Tuytelaars, Marc Proesmans, Luc Van Gool, and Esat Mi. The cascaded hough transform. In *In Proceedings of ICIP*, pages 736–739, 1997.

# 9 Curiculum Vitae

Jiří Havel
*Phone:* +420 54114-1048
*Fax:* +420 54114-1270
*Mobile:* +420 776084859
*E-mail:* ihavel@fit.vutbr.cz, JSH@seznam.cz
*WWW:* www.fit.vutbr.cz/~ihavel
*Address:* Podvihovská 42, 747 70 Opava 9, Czech Republic

## Current Study

**Brno University of Technology**
Faculty of Information Technology

Ph.D. **2008 – present day**
Graph@FIT Research Group
*Topic:* Line Parameterizations not only for Hough Transform
*Supervisor:* Adam Herout

## Awards

**TOP 10 Excelence VUT 2012**

3rd place among Ph.D. students, category "Publications".

## Education

**Brno University of Technology**
Faculty of Information Technology

Ing. (master degree) **2006 – 2008**
*Programme:* Information Technology
*Specialization:* Computer Graphics and Multimedia
*Thesis Topic:* Fast Ray-Triangle Intersection

Bc. (bachelor degree) **2003 – 2006**
*Programme:* Information Technology
*Thesis Topic:* Toolkit for 3D Realtime Applications

## Notable Publications

- **Herout, A., Dubská, M., Havel, J.**: Real-Time Detection of Lines and Grids By PClines and Other Approaches, Springer, 2013, in print, ISBN 978-1447144137

- **Dubská, M., Herout, A., Havel, J.**: PClines - Line Detection Using Parallel Coordinates, In: Proceedings of CVPR 2011, Colorado Springs, US, IEEE CS, 2011, p. 1489-1494, ISBN 978-1-4577-0393-5

- **Antikainen, J., Havel, J., Jošth, R., Herout, A., Zemčík, P., Hauta-Kasari, M.**: Non-Negative Tensor Factorization Accelerated Using GPGPU, In: IEEE Transactions on Parallel and Distributed Systems (TPDS), Vol. 2011, No. 1111, US, p. 7, ISSN 1045-9219

- **Jošth, R., Antikainen, J., Havel, J., Herout, A., Zemčík, P., Hauta-Kasari, M.**: Real-Time PCA Calculation for Spectral Imaging (using SIMD and GP-GPU), In: Journal of Real-Time Image Processing , Vol. 2011, No. 1111, DE, p. 8, ISSN 1861-8200

- **Vlček, A., Havel, J., Herout, A.**: Front-to-Back Blending with Early Fragment Discarding, In: Proceedings of Spring Conference on Computer Graphics, Bratislava, SK, UNIBA, 2010, p. 91-97, ISBN 978-80-223-2843-2

- **Havel ,J., Herout, A.**: Yet Faster Ray-Triangle Intersection (Using SSE4), In: IEEE Transactions on Visualization and Computer Graphics, Vol. 2010, No. 3, US, p. 434-438, ISSN 1077-2626

- **Herout, A., Zemčík, P., Hradiš, M., Juránek, R., Havel, J., Jošth, R., Žádník, M.**: Low-Level Image Features for Real-Time Object Detection, Pattern Recognition, Recent Advances, Vienna, AT, IN-TECH, 2010, p. 111-136, ISBN 978-953-7619-90-9

- **Havel, J.**: Functional Programming of Geometry Shaders, In: WSCG 2010 Communication Papers Proceedings, Plzeň, CZ, ZČU v Plzni, 2010, s. 9-13, ISBN 978-80-86943-87-9

- **Havel, J., Herout, A.**: Rendering Pipeline Modelled by Category Theory, In: GraVisMa 2010 workshop proceedings, Plzeň, CZ, ZČU v Plzni, 2010, s. 101-105, ISBN 978-80-86943-85-5

- **Havel, J.**: Accelerated Object Detection, In: Proceedings of the 15th Conference Student EEICT 2009, Brno, CZ, FEKT VUT, 2009, s. 456-460, ISBN 978-80-214-3870-5

- **Havel, J.**: Rychlý výpočet průsečíku paprsku s trojúhelníkem, In: Proceedings of the 14th Conference Student EEICT

2008, Brno, CZ, FEKT VUT, 2008, s. 196-198, ISBN 978-80-214-3615-2

## Teaching Experience

**Computer Graphics**
Lectures **2010 – 2011**
Laboratory exercises **2008 – 2011**
The course contains basics of OpenGL programming. I have innovated the course contents to better match the modern OpenGL usage i.e. the core profile and shaders.

**Advanced Computer Graphics**
Lectures **2010 – 2011**
One lecture on advanced OpenGL topics such as Geometry Shaders, Transform Feedback and Tesselation Shaders

## Internships

9 - 27.5 2011
Infotonics Center, University of Eastern Finland, Joensuu, Finland

16.9. - 15.12.2012
IRISA, University of South Brittany, Vannes, France

## Hardware and Software Skills

GP-GPU computing using OpenCL and CUDA.

Algorithm parallelization and acceleration using OpenMP and SIMD.

C, C++ (excellent knowledge, including advanced template metaprogramming), Haskell, Assembler (x86, 8051)

Winapi, OpenCV, Eigen, OpenGL, C++ stl+boost, SDL

8051 microcontrollers and TI C6000 DSPs

TEX, Vim, Make, Subversion, Maple

## Theoretical Knowledge

Mathematics - Geometric Algebra, Linear Algebra and Geometry

Computer Vision - Hough Transform, Boosting algorithms

Computer Graphics - Raytracing and its acceleration, Rasterization

## Language Skills

**English** advanced knowledge

**German** basics