AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

KATEDRA TELEKOMUNIKACJI

PRACA DYPLOMOWA INŻYNIERSKA

Analysis of Algorithms for Geometric Distortion Correction of Camera Lens

Analiza algorytmów korekcji zniekształceń geometrycznych obiektywu kamery wideo

Autor: *Krzysztof Szczęsny*
Kierunek studiów: *Teleinformatyka*
Opiekun pracy: *dr inż. Jarosław Bułat*

Kraków, 2015

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim»”, oświadczam, że niniejszą pracę dyplomową wykonałem osobiście, samodzielnie i że nie korzystałem ze źródeł innych niż wymienione w pracy.

_____

Table of Contents

Abstract

Conventional, model-fitting based algorithms for lens distortion correction impose many assumptions upon modeled distortions, which may lead to inaccuracy. In this paper such method was compared to own implementation of a nonparametric correction algorithm. Lowest obtained RMSE in both cases was equal to 0.4 pixels. Usage of the nonparametric algorithm, however, allowed to avoid typical errors, such as invalid model extrapolation. Additionally the algorithm was verified for a broader grade of cameras.

Keywords — lens distortion, radial model, nonparametric, textured pattern

Abstrakt

Konwencjonalne algorytmy korekcji zniekształceń obiektywu oparte na dopasowaniu modelu nakładają na modelowane zniekształcenia wiele założeń, które mogą powodować niedokładność. W niniejszej pracy dokonano porównania metody konwencjonalnej z własną implementacją nieparametrycznego algorytmu korekcji. Najmniejszy błąd uzyskany podczas doświadczeń w obu przypadkach wynosił 0.4 piksela. Użycie algorytmu nieparametrycznego pozwoliło jednak na uniknięcie typowych błędów, takich jak niewłaściwa ekstrapolacja modelu. Dodatkowo prawidłowe działanie algorytmu nieparametrycznego zostało zweryfikowane dla szerszej grupy aparatów.

Słowa kluczowe — zniekształcenia obiektywu, model radialny, nieparametryczny, wzorzec teksturowy

Introduction

Computer vision is a field of computer science that is still becoming increasingly popular, particularly due to ubiquitous cameras: in computers, smartphones, game consoles or drones. Popularity entails high demands for accuracy and robustness.

For many tasks a camera should be first calibrated, i.e. both its position in relation the scene (extrinsic parameters) and optical properties such as focal length or pixel skew (intrinsic parameters) should be measured. Extrinsic parameters allow to change coordinate system from one associated with the world to one associated with the camera. Intrinsic parameters define how from such points an image is formed.

No camera is free of imperfections, however. They cause the camera's optical system to behave differently than expected, leading to distortions in images and videos alike. This greatly impedes many algorithms that require consistency of input images.

To mitigate this, a distortion correction can be carried out. Usually it is done during calibration, with distortion characteristics being treated as additional intrinsic parameters. "*The goal of the distortion calibration is to find the transformation (or undistortion) that maps the actual camera image plane onto an image following the perspective camera model*" [4]. Standard approach is to model this transformation using a function with adjustable coefficients.

This method had been used for decades with good results. In recent years, however, subpixel accuracy has become essential. Even small residual distortions can cause photogrammetry or 3D surface reconstruction from 2 images to fail.

Therefore other algorithms are studied, where the transformation can be any function within image domain. Such methods, for instance [9], are more complicated, but allow to model any distortion, regardless of its uniformity.

This paper aims to compare results of both methods, both quantitatively and qualitatively. Although this already has been researched in [25] by authors of [9], used parametric models had higher than usual number of degrees of freedom and tests have been performed using one DSLR camera. Therefore a standard model (Conrady-Brown) was selected for comparison and other camera types were chosen.

In order to perform this comparison, a nonparametric algorithm had to be implemented, along with adjuvant means to measure image quality before and after correction.

Paper is divided into 3 main chapters. First chapter contains theoretical introduction and state-of-the-art. Research methods and implementation details are elaborated in the second chapter, along with select initial results that illustrate many early development stage problems and their solutions. Final obtained results are given and commented in the concluding chapter.

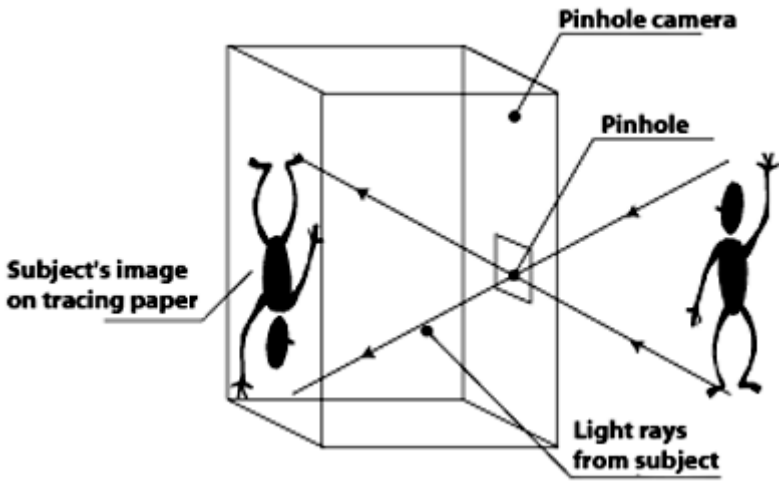
1. Subject of Lens Distortion

This chapter is intended to introduce mathematical elements of computer vision, in context of the phenomenon of geometric distortion. Theoretical relationships are compared with real camera design. Finally, methods for distortion correction that can be found in the literature are overviewed.

1.1 Projective geometry

Projective geometry [11] allows to map Euclidean space points $\mathbb{R}^3$ to $\mathbb{R}^2$ plane (projective plane), mimicking the functioning of human eye or a camera objective, as depicted in Figure 1.1.

The geometry is presented in Figure 1.2. Coordinate system *XYZ* is associated with camera, while *xy* describes the image. Image plane π is parallel to axes *OX* and *OY* and perpendicular to axis *OZ*. Distance between π and center point *O* is denoted as *f* (focal length). In order to avoid image inversion, it is assumed that image plane occupies the same half-space as Euclidean points that are being projected. Axis *OZ* intersects image plane in *P* (principal point). Every Euclidean point *Q* can be connected to the camera center point with ray *QO*. If ray intersects π, *Q*'s projection is defined by the point of intersection *Q'* [13, 27].



Figure[1] 1.1: Motivation for projective geometry

[1] source: http://www.hcc.commnet.edu/artmuseum/anseladams/graphics/pinhole.gif

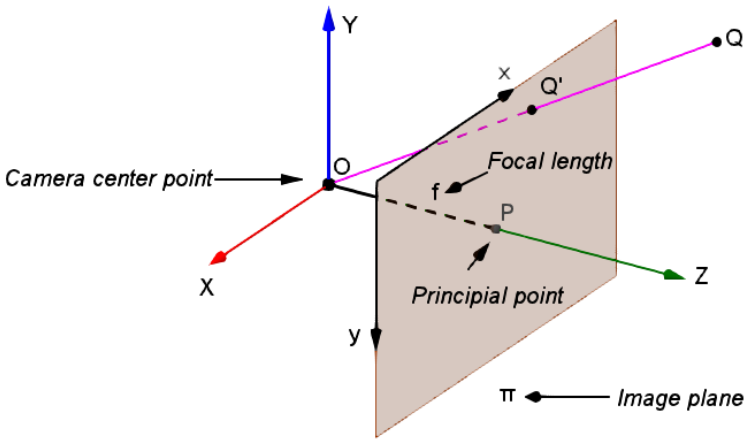


Figure 1.2: Image formation

1.1.1 Homogenous coordinate system

To perform transition between Euclidean and image coordinates, homogenous coordinate system is used. It can describe points at infinity, which are not a part of Euclidean space, and allows to express relationships between points with the means of matrix multiplication.

Following notation from Figure 1.2, one may observe that not only Q, but also every[2] other point lying on ray QO will be projected as Q'. On the other hand, ray QO is a line that passes through point $O = (0,0,0)$. Therefore it can be concluded that every point projected as Q' has form (1).

$$(x, y, z) = (x_Q, y_Q, z_Q) \cdot t, \qquad t \neq 0 \tag{1}$$

where: x, y, z – Euclidean coordinates of a point whose projection is Q';

x_Q, y_Q, z_Q – Euclidean coordinates of point Q;

t – real parameter

Homogenous system is based on the principle of (1). Transition from Euclidean to homogenous coordinates is done by adding a fourth coordinate equal to unity (2). Two homogenous points are equal (i.e. represent the same line consisting of Euclidean points mapped to one common point on projective plane) if and only if there exists $k \neq 0$ satisfying (3). Conversion back to the Euclidean system is described by (4).

$$(x, y, z) \rightarrow (x, y, z, 1) \tag{2}$$

[2] excluding point O, which is not a part of homogenous space

$$x_1 = kx_2, \quad y_1 = ky_2, \quad z_1 = kz_2, \quad t_1 = kt_2 \tag{3}$$

where: k – real parameter

$$(x, y, z, t) \rightarrow \left(\frac{x}{t}, \frac{y}{t}, \frac{z}{t}\right), \qquad t \neq 0 \tag{4}$$

It should be noted that if $t = 0$, then homogenous point $(x, y, z, 0)$ lies on the plane at infinity and cannot be expressed with Euclidean coordinates.

1.1.2 2D image transformations

If Euclidean points lie on a plane and are to be converted to homogenous coordinates, it can be assumed without loss of generality that $z = 0$ and thus z coefficient can be omitted. This is usually the case in the field of camera calibration [30].

If all homogenous image points are multiplied by a matrix H, a transformation (5) will be applied to the image.

$$\forall (x, y, t) \in \pi : \qquad (x', y', t') = \underbrace{\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}}_{H} (x, y, t) \tag{5}$$

where: x', y', t' – homogenous coordinates of transformed point x, y, t

Depending on form of H, following transformation types can be distinguished [11], ordering from less complex:

- isometry — rotation and translation,
- similarity — isometry with scaling,
- affine — non-isotropic scaling (skew),
- homography — projective transformation; an example is shown in Figure 1.3.

Figure 1.3: Example of projective transformation

When camera captures an image, in general case, a projective transformation is applied to the points of the photographed scene. Although neither lengths nor angles are preserved, point collinearity is an invariant property of this transformation. This trait is used to evaluate efficiency of algorithms that compensate geometric distortion[3] of camera lens. It is usually computed as RMS (Root Mean Square) of distance between points that are known to be collinear and their regression line.

1.1.3 Pinhole camera model

Basic principle of a pinhole camera is shown in (6). Camera matrix is solely a product of extrinsic and intrinsic parameters and has form (7) [9].

$$y \sim Cx \tag{6}$$

where: x – Euclidean point in homogenous coordinates (4-tuple);

y – homogenous image point corresponding to x (3-tuple);

C – 3x4 camera matrix [11];

$\sim$ – equality between homogenous points

$$C = \underbrace{\begin{bmatrix} fc_x & s & cc_x \\ 0 & fc_y & cc_y \\ 0 & 0 & 1 \end{bmatrix}}_{K} R\begin{bmatrix} I & | & -T \end{bmatrix} \tag{7}$$

where: K – 3x3 matrix composed of intrinsic parameters;

fc_k –focal length along axis k; cc_k – k-coordinate of principal point;

s – skew factor; R – 3x3 3D rotation matrix;

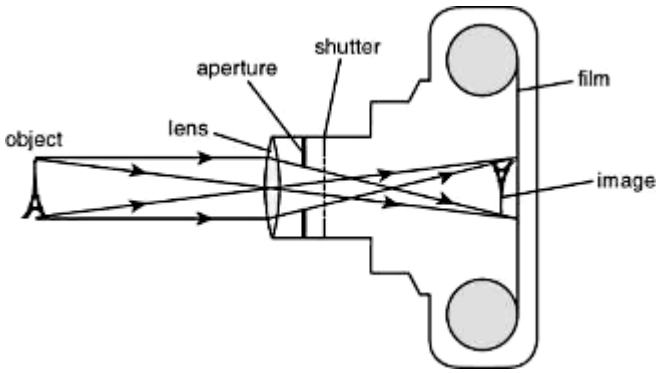
T – Euclidean point representing camera center (3-tuple)

1.2 Camera anatomy

Described camera model is linear [10, 16, 21], due to usage of homogenous coordinates [11]. However, since cameras consist of imperfect components, in reality they evince nonlinear behavior [29].

As one can observe in Figure 1.4, light rays must travel through lenses in order to reach image sensor, which can be either a photosensitive film in analogue cameras or a CCD (Charge Coupled Device) in digital ones. The lenses can be deformed as well as misaligned. Image sensor also is not perfectly perpendicular to optical axis. This causes ray trajectories to curve from expectations and thus introduces distortions.

[3] The phenomenon of geometric distortion is explained below in section 1.2.



Figure[4] **1.4**: Analogue camera outline; digital cameras feature a CCD in place of film

There are other causes for nonlinearities, such as quantization noise and color aberration [14]. Moreover, ultra wide angle cameras adhere to other (non-perspective) geometries [23, 24]. They are out of the scope of this paper.

1.3 Standard distortion model

There is no distortion model based strictly on a physical basis [25]. Widely used [14, 15] Conrady-Brown model [8, 29] has following assumptions:

1. There is a distortion center, which is neither principal point, nor geometrical image center. Although those points are usually situated in proximity, they cannot be assumed to be equal [12]. Every line passing through distortion center is not distorted [26].
2. Pixel positions are distorted by two independent factors: tangential distortion and radial distortion.
3. Tangential (decentering) distortion is modeled by (8) [19]. Due to this factor being usually smaller than radial component, it is sometimes omitted [4].

$$\begin{cases} x_{corrected} = x + \left[2p_1xy + p_2\left(r^2 + 2x^2\right)\right] \\ y_{corrected} = y + \left[p_1\left(r^2 + 2y^2\right) + 2p_2xy\right] \end{cases} \tag{8}$$

   where: x, y – pixel coordinates; r – distance from distortion center;

   p_i –tangential distortion coefficients

4. Radial distortion is represented by (9). The radial function $f(r)$ is frequently modeled with an even-order polynomial [12], such as (10).

[4] source: http://www.mauitroop22.org/merit_badges/images/camera.jpg

$$\begin{cases} x_{corrected} = x \cdot f(r) \\ y_{corrected} = y \cdot f(r) \end{cases} \tag{9}$$

$$f(r) = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 \tag{10}$$

where: x, y – pixel coordinates; r – distance from distortion center;

k_i – radial distortion coefficients

There also exists plethora [26, 25, 16] of models using other functions, notably: rational [3], FOV [4] (Field of View), division [7], cubic rational [10] and rational polynomial [17] model. The aim is to choose a function that is consistent with given camera type, but also uses as few coefficients as possible.

1.3.1 Distortion correction with standard model

In order to rectify image with this model, distortion coefficients must be calculated, as well as other camera intrinsic parameters. Level of RMSE (RMS Error) reported in literature commonly varies between 0.1 px and 1 px [4, 7, 30]. The algorithm [1, 13] can be summarized:

1. Make at least ten [19] photographs of a planar pattern, in different positions [30]. If photographs are too similar, computations will become numerically unstable [17]. Popular pattern is a checkerboard, depicted in Figure 1.5. Camera settings, particularly zoom and focal length, must not be changed, lest distortion is inconsistent between photographs.

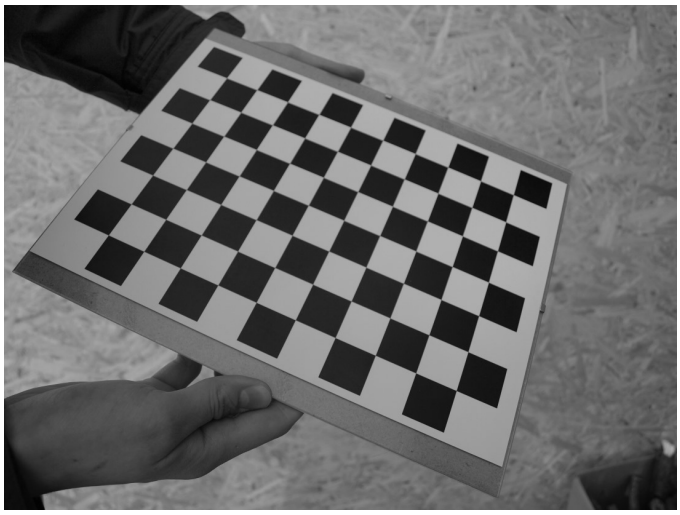


Figure 1.5: A popular checkerboard pattern

2. Extract characteristic points, such as intersection points or figure vertices, possibly with subpixel accuracy.

3. Use a closed-form solution, such as DLT [11] (Direct Linear Transformation), to calculate initial intrinsic camera parameter values. This step ignores distortions.
4. Perform iterative nonlinear optimization of an objective function. Because values obtained in step 3 are usually close to optimal, convergence to a local minimum is avoided [13]. In case distortions are extremely severe, complexity of this step can rise [20].

1.4 Nonparametric approach

Conrady-Brown model is very popular and has many variations tailored for different needs, e.g. for wide angle cameras [3]. Nevertheless, it contains flaws that in some cases may be critical. The assumption that distortion can be modeled with a simple radial function is not always correct. There are algorithms not making such assertions, or making them to a lesser extent.

In [12] a non-iterative radial method correction is proposed, where only assumption about the radial function is its monotonicity. Similarly to standard model, a calibration grid if required. Additionally authors provide a method of finding distortion center — crucial issue if only radial component is considered.

More complex solution is considered in [25], where distortion is compensated by 2D polynomials in image domain. It is argued that despite high order of used polynomials (highest tested being 15th), their coefficients can be computed linearly. Over-fitting can be avoided as long as there is a large number of control points.

Very elaborate nonparametric algorithm is introduced in [24]. In order to better suit ultra wide angle cameras (140°–220°), stereographic projection model is used instead of perspective projection. Calibration rig consist of numerous black balls of varying sizes.

Amidst these methods *Grompone von Gioi et al.* [9] algorithm was selected for implementation and analysis. Its main advantage over other nonparametric algorithms is simplicity of calibration rig, especially in comparison to [24].

1.4.1 *Grompone von Gioi et al.* algorithm outline

The algorithm requires an image of a textured pattern, such as the one presented in Figure 1.6. The following steps should be taken:

1. Take 2 slightly different images of the planar textured pattern. Camera settings must not be changed while images are being taken.

2. Extract keypoints from the pattern and its photographs using SIFT [18] (Scale-Invariant Feature Transform) feature detection algorithm.
3. Find correspondences between pattern and photographs' keypoints with a descriptor matching algorithm. Match pairs constitute a sparse reverse distortion field (i.e. for each match pair there exists an affine transformation that locally maps pattern pixels to photography pixels).
4. Filter outlier matches using loop validation:
    a. Project first photograph match points through second photograph's sparse reverse distortion field into pattern domain.
    b. Estimate homography between projected points and first photograph's pattern match points using RANSAC [6] (RANdom SAmple Consensus) algorithm.
    c. Discard matches marked as outliers by RANSAC.
    d. Discard second photograph.
5. Find triangulation of pattern match points using Delaunay algorithm.
6. Calculate affine transformation from pattern match points to scene match points for every triangle.
7. Compute displacement vector for every pixel inside triangles to get dense reverse distortion field. Discard border pixels lying outside the triangulation's perimeter.
8. Apply resulting displacement field to the photography in order to rectify it.

Theoretical formulas that allow successful image rectification are as follows. Since there are distortions, camera matrix has form (11) [9] instead of (7). Field D does not change as long as camera settings are constant. Homography H is a result of camera position in regard to pattern plane (any 3D translation and rotation) when calibration photograph for the algorithm was taken. The algorithm allows to estimate only DH; D and H are unknown individually. Application of $(DH)^{-1}$ to a distorted image cancels out distortions, but introduces a homography H^{1}. The homography transforms camera into a *virtual pinhole camera*, whose parameters no longer have physical meaning.

$$\hat{C} = DHKR[I \mid -T] \tag{11}$$

where: D – 2D vector field in image domain representing distortions;

H – 3x3 homography matrix

After applying feature matching there is relationship (12) [9] between match points.

$$DH_v I_v = v_I, \quad DH_u I_u = u_I \tag{12}$$

where: v – 1st photograph;

u – 2nd photograph;

H_k – homography matrix H for photograph k;

I_k – match point on the pattern;

k_I – corresponding match point on photograph k

Principle of the loop validation is shown in (13) [9]. Points v_I, having form (12), are projected into pattern by applying $(DH_u)^{-1}$. The relationship between such projections and I_v is a homography $H_u^{-1}H_v$. No additional distortions are present.

$$I_{uv} = (DH_u)^{-1} v_I = (DH_u)^{-1} DH_v I_v = H_u^{-1} H_v I_v \tag{13}$$

where: I_{uv} – projection of v_I through v's distortion field

The algorithm was examined by its authors — RMSE after rectification was reported to be equal to 0.08 pixels. This method was also used as a benchmark for model-based algorithm comparison in [25]. However, in the papers only one camera is calibrated and in both cases it is a high-end DSLR camera. Thus it is unknown how well does the algorithm correct images captured by inexpensive cameras.

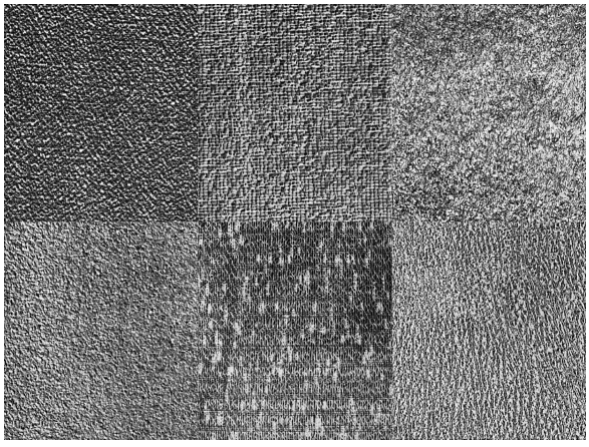


Figure 1.6: Textured pattern used in [9]

2. Used Distortion Correction Methods

This section describes test rig along with reasons for particular choices. Cameras were designated for particular figures so that presented issues would be most noticeable. Photographs used to generate figures were taken during early stages of research and are not included on the CD-ROM. Details of the two analyzed algorithms are elaborated:

1. The parametric correction algorithm [19].
2. The nonparametric correction algorithm based on [9].

2.1 Software environment

Python 2.7.10 was chosen as programming language for both algorithms. Key features that lead to this choice were: availability of the OpenCV library as well as a MATLAB-like numeric library (NumPy) and Python's rapid development capabilities. OpenCV version 2.4.11 was used, due to absence of SIFT in version 3.0. The work resulted in package containing over 1600 lines of code. It features following classes:

- `Radial` — a wrapper for OpenCV correction,
- `NonparametricCorrection` — implementation of the nonparametric correction algorithm,
- `RMS` — a module for line straightness evaluation. Two types of patterns can be photographed and analyzed to calculate RMSE:
    - Checkerboard — same as for OpenCV calibration. Corners are extracted and fitted into vertical and horizontal lines. Whole board must be visible.
    - Straight line — a single continuous line (a photograph of this pattern is depicted in Figure 2.1). Using Canny [2] edge detector, both edges of the

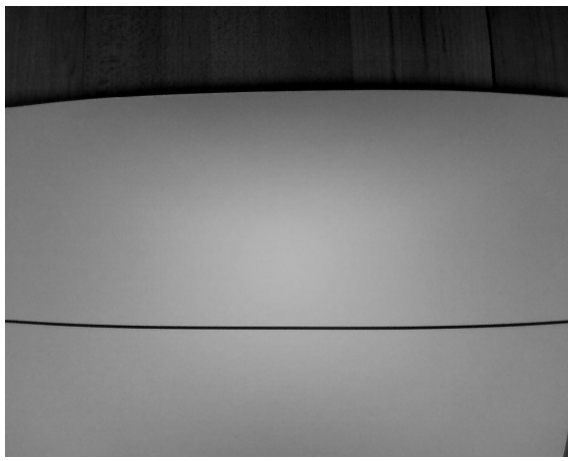


Figure 2.1: Sample photograph of the straight line pattern

line are extracted. Outliers are filtered with hit-or-miss transform, using 8 binary signatures. Left points are compared with their regression line. For reliable error estimation, multiple images should be used, each containing the line in different region of the field of view. It should be noted that robust line detection was not the main focus of this paper, therefore it is not foolproof and each result requires manual countercheck.

2.2 Experimental setup

Although distortion correction can be used in photography solely for visual enhancement, it is mainly used in computer vision. Among applications of computer vision there are, for instance, surveillance video interpretation or endoscopic imaging [23]. Hence it cannot be presupposed that camera is of high quality. Therefore both algorithms were tested on a wide range of photographic devices, as shown in Table 2.1.

Setups #1–#5 were thoroughly analyzed. In setup #6 the camera was unable to focus on single line pattern and therefore RMSE could not be measured. If the pattern had been situated further from the camera, it would have been focused, but the line would not have passed through whole image as required. Setup #7 was available only for early tests of the second algorithm. The pattern was photographed on monitor, with camera situated on a tripod. Nonetheless, these initial results are noteworthy.

Lanczos interpolation over 8x8 pixel neighborhood was used for rectifications.

2.2.1 Decisions based on initial results

Initially patterns were displayed on a monitor, ensuring pattern flatness. This approach was used in [26] with excellent outcomes. However, most cameras, particularly setup #5, performed poorly. Photograph of monitor displaying the pattern is presented in Figure 2.2 — extreme vignetting can be observed in image corners. Therefore another approach was taken and patterns were printed. Offset A4 paper was selected for its stiffness. Then prints were fixed onto clip frame glass with spray adhesive.

For parametric model a 10x7 checkerboard (i.e. with 10 horizontal and 7 vertical intersection points) was printed. It was already featured in Figure 1.5. More dense checkerboards were used initially, but they were not reliable in conjunction with available cameras. The shorter square edge, the less precise corner search, as shown in Figure 2.3, where a denser board (49x32) was used. Four corners found by OpenCV are marked as red pixels. It can be observed that their positions are inconsistent.

Table 2.1: Tested cameras

Setup	Device name	Camera type	Resolution [px]
#1	Canon Legria FS200	camcorder	1024x768
#2	Creative VF0420	low-end webcam	800x600
#3	Konica Minolta DiMAGE A200	bridge camera	3264x2448
#4	Nokia E90	cell phone	2048x1536
#5	Genius WideCam 1050	wide angle webcam	1280x1024
#6	Nokia E9	smartphone	3264x2448
#7	Olympus E520	DSLR camera	3648x2736

Figure 2.2: Image of monitor, with imaging software watermark; setup #5

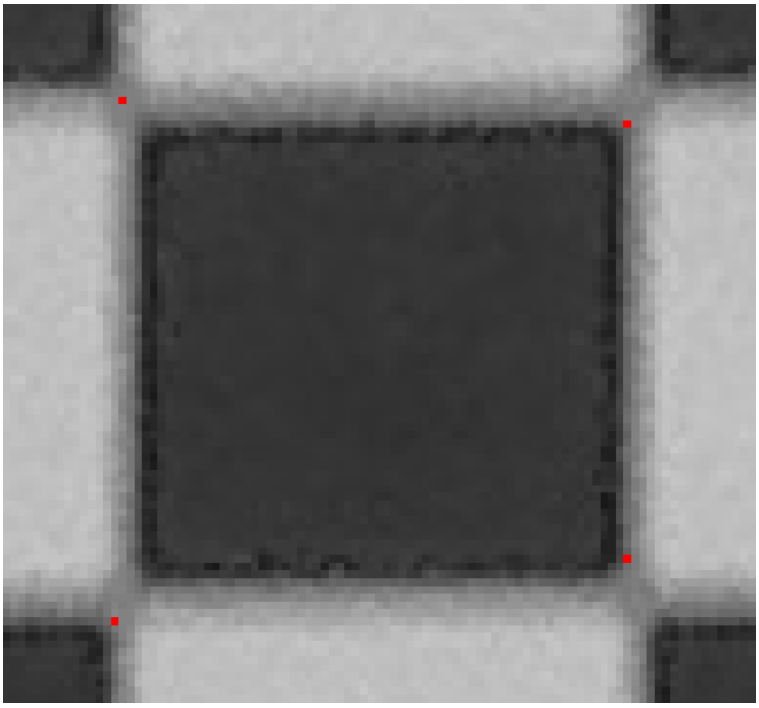
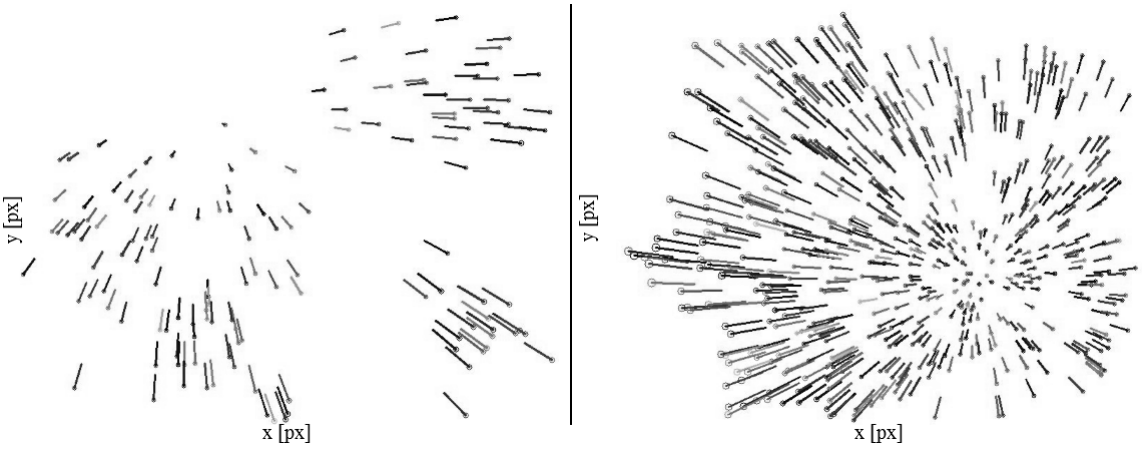


Figure 2.3: Magnification of one checkerboard square; setup #3

RMSE measurements were carried out with two patterns featuring a single line. Lines varied in thickness, because there was a possibility that either: a line too thick would be detected as 4 edges (due to effects of camera processing), or a line too thin would not be detected at all.

Initial test have shown that worse cameras did not perform well with a highly detailed textured pattern. There were too few feature matches, which resulted in discontinuous reverse distortion field. Most notable results are depicted in Figure 2.4. Hence three textured patterns were chosen for nonparametric algorithm, each with different detail level. During final measurements usage of the most detailed one gave satisfactory results only in setup #5.



(a) detailed pattern (138 matches) (b) simpler pattern (565 matches)

Figure 2.4:Matches between pattern and image points, defining the reverse distortion field; setup #2

Apart from setup #7, all photographs were taken by the window on a foggy day, when sunlight was scattered, so as to provide illumination as uniform as possible. Patterns were laid down and cameras were moved around them. For each fully analyzed setup there was taken following number of photographs: 14–23 of chessboard, 19–40 of line pattern and 4–6 of textured pattern.

2.3 Parametric algorithm

Parametric correction was done by using following OpenCV functions: `findChessboardCorners()`, `cornerSubPix()`, `initCameraMatrix2D()`, `calibrateCamera()` and `initUndistortRectifyMap()`. Then resulting distortion field was used to rectify images with `remap()` function, which transforms the image according to (9). The mapping can be visualized as in Figure 2.5, similarly to [1].

$$dst(x, y) = src(map_x(x, y),\ map_y(x, y)) \tag{14}$$

where: x, y – pixels; map_k – displacement map for coefficient k;

src – input (distorted) image; dst – output (rectified) image

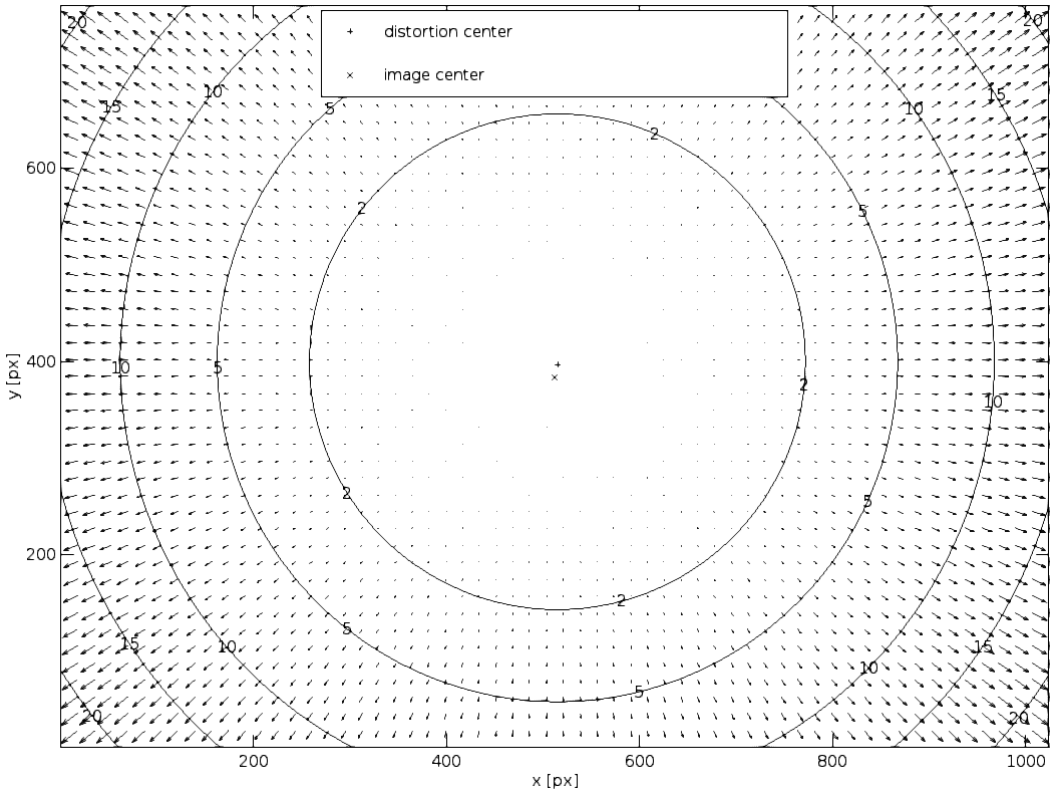


Figure 2.5: Mapping resulting from parametric correction; setup #1;
vectors connect distorted pixel positions with their undistorted counterparts;
contours mark areas with constant vector length

2.3.1 Verification

Since aforementioned OpenCV functions are an implementation of [1], *Camera Calibration Toolbox for Matlab* was used to verify select setups. Because of tedious manual grid corner extraction procedure, not every setup was verified. Outcomes for setup #1 were collected in the Table 2.2. The parameters are: focal length (fc_x, fc_y), principal point location (cc_x, cc_y), radial (k_1, k_2, k_3) and tangential (p_1, p_2) distortion coefficients. Uncertainties were computed as triples of the standard deviations (99.73% confidence level).

It is notable that for almost all coefficients their absolute values are smaller than uncertainties, especially for sixth order radial coefficient k_3. Had been only the *Toolbox* used, this coefficient would have been set to 0 and calculation would have been redone,

most likely resulting in better calibration. The OpenCV implementation, however, does not return any information about parameter uncertainties. So the goal of verification was just to check whether OpenCV functions were used adequately and not whether their default behavior produced best results possible.

Results from OpenCV are within reported uncertainties, so it can be concluded that the library has been used correctly.

The *Toolbox* allows additional visualizations. For example, Figure 2.6 depicts camera extrinsic parameters for each of the supplied photographs.

Table 2.2: Verification of calibration results; setup #1

parameter	OpenCV	Camera Calibration Toolbox	
	value	value	uncertainty
fc_x	1313.47	1313.55	1.86
fc_y	1311.39	1311.42	1.85
cc_x	515.52	515.58	2.32
cc_y	396.43	396.12	1.80
k_1	-2.083e-1	-2.100e-1	2.259e-2
k_2	1.125e-1	1.372e-1	3.836e-1
k_3	6.461e-1	5.705e-1	1.939e+0
p_1	3.690e-4	3.265e-4	2.893e-4
p_2	-2.117e-4	-2.069e-4	3.202e-4

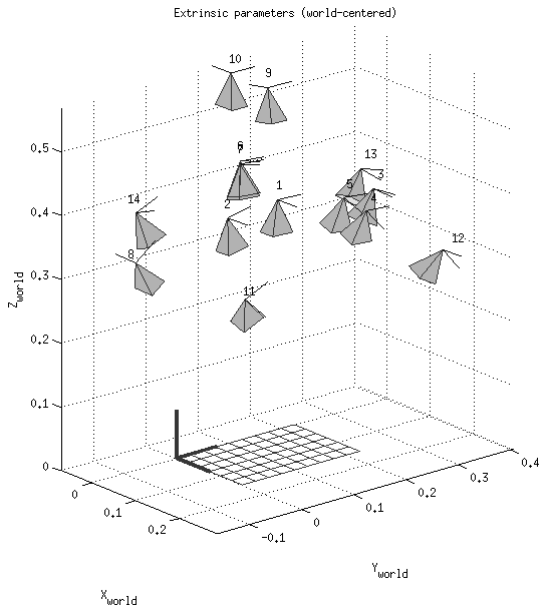


Figure 2.6: Camera positions (piramids) in relation to the checkerboard; setup #1

2.4 Nonparametric algorithm

Because *Grompone von Gioi et al.* algorithm description in [9] is terse, analyzed implementation differs in details from the paper. This was an opportunity to inspect additional input options and scenarios.

First of all, it was observed that for some photographs the feature match distribution was unbalanced between pattern regions. For instance in Figure 2.7 there are 3 regions with hardly any keypoints. While SIFT algorithm can be set to higher sensitivity to features, its time complexity[1] and memory consumption are high (see section 3.3.3). Additionally, it would be impractical to search for more features in areas that already are replete. Thus after first feature detection, images were partitioned into rectangular chunks and a more sensitive SIFT was repeated only on select chunks. This step was named **chunking**.

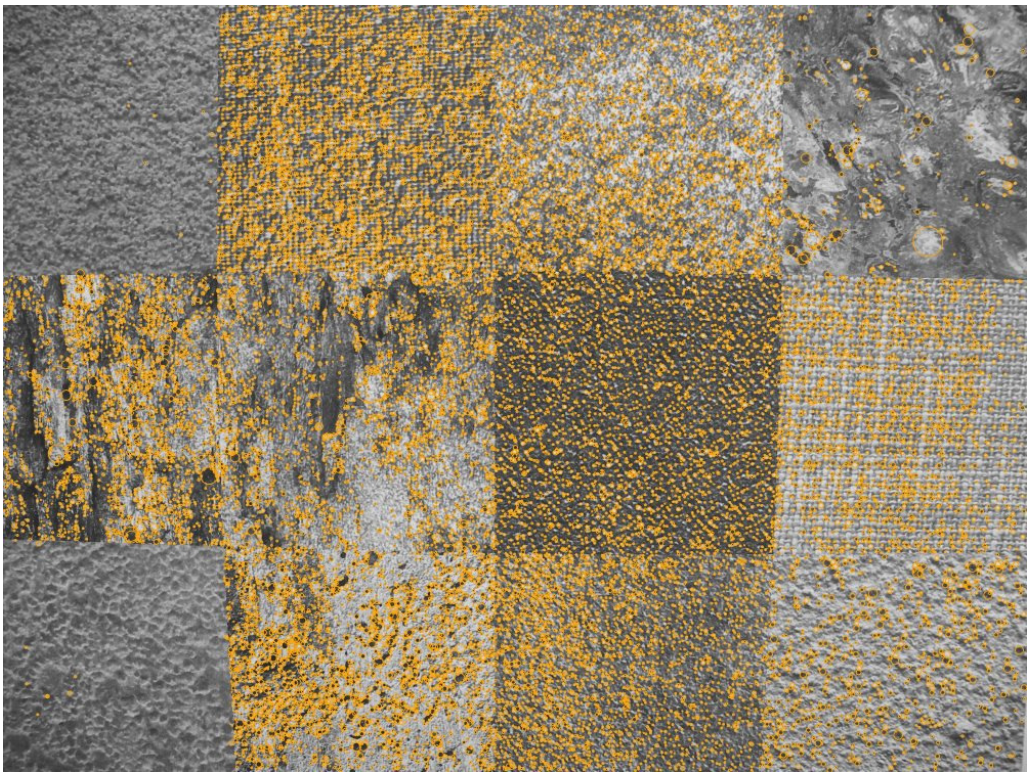


Figure 2.7: Keypoints (orange) found in photograph[2]; setup #3

[1] More precisely, time complexity of the descriptor brute-force matcher is high.
[2] Photographed pattern consists of images whose copyrights belong to: R. Grompone von Gioi, P. Monasse, J.-M. Morel, Z. Tang, P. Kratochvil, B. Reynolds, Spiral Graphics Inc., *plaintextures.com*, *textureofnewyork.com*, A. Kuzniatsou and *textureshot.com*.

Secondly, there was the issue of outlier removal. After feature matching, there are regularly some matches that do not correlate with real correspondences. They can be observed in Figure 2.8 as (mostly) long lines that do not follow the tendency. Basic method used to filter out bad matches was a Lowe ratio test, proposed in [18]. It was not enough, however.

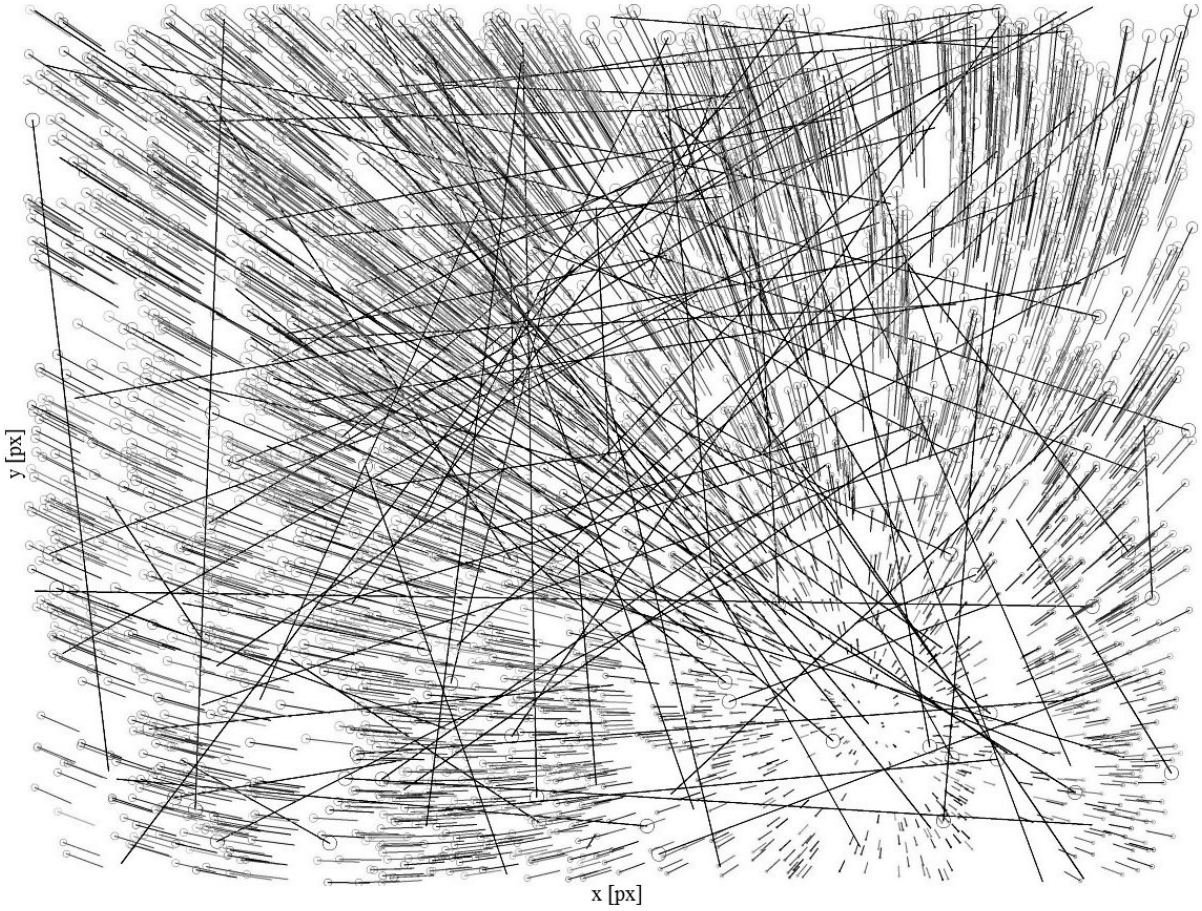


Figure 2.8: Matches between pattern points and image points, with outliers; setup #3

If there existed a homography between the pattern and photograph (i.e. if there were no distortions), this task could be accomplished with RANSAC (RANdom SAmple Consensus) algorithm [6]. However, because distortions are present, RANSAC would also discard not only outliers, but also many valid matches from highly distorted image regions (usually borders).

Solution to this problem was presented by algorithm authors and denoted as **loop validation** [9]. A second photograph of the same pattern, slightly shifted, is also analyzed. Every match point on the pattern can be projected back into pattern domain by reverse distortion field of the second image. Such projection introduces additional arbi-

trary homography, but nullifies distortions, as shown in section 2.4. Thus RANSAC can finally be applied.

The projection was implemented with bivariate B-spline interpolation [22]. For every match of a pattern point, its 18 closest neighbors on second image were selected. To avoid the influence of outliers present in second image, in total 8 neighbors with most extreme displacement vector values were rejected. Remaining 10 were used for interpolation. After applying loop validation to both images, one with more remaining matches was singled out for further processing.

Finally it transpired that reverse distortion field after piecewise affine interpolation was still visibly discontinuous between triangles. Moreover, every outlier not eliminated by loop validation produced an unacceptable artifact. Modulus of such field is depicted in Figure 2.9. Hence three **smoothing** strategies were examined:

1. Least squares low-order 2D polynomial fitting.
2. Incremental Gauss filtering.
3. Single Gauss filtering preceded by nearest-neighbor interpolation.

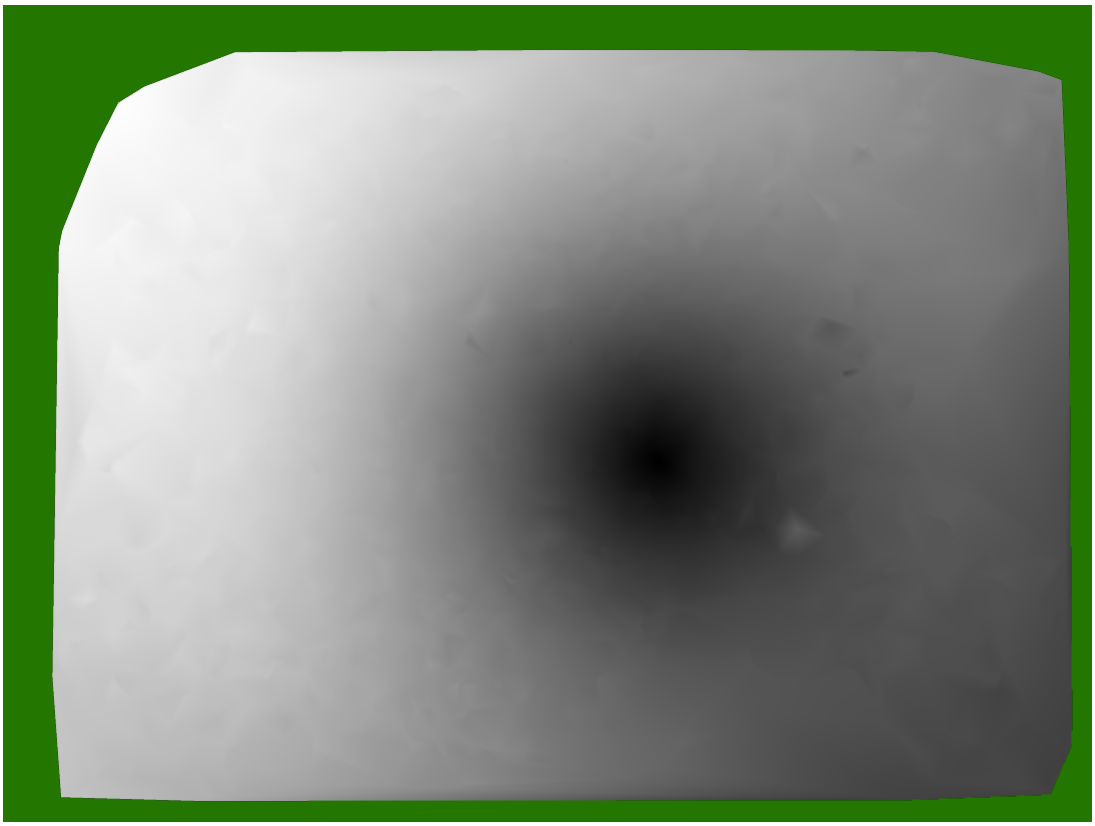


Figure 2.9: Modulus of a jagged reverse distortion field; setup 4;
lowest values are drawn with black, NaNs — with green

Gauss filtering replaces each matrix value with a weighted mean of its neighbors values. If any of neighbors is equal to NaN, this value will propagate. Thus had been reverse distortion field simply filtered, its area would have shrunk. Therefore in 2. a set of gradually smaller filters was used to fill out areas left by larger filters. In 3. the border area was filled by nearest neighbor interpolation, one Gaussian filter was applied and then original border was reset to NaN.

After smoothing and image rectification there were observed additional strong distortions on borders. This effect was attributed to triangulation. A product of triangulation is shown in Figure 2.10. Triangles near the perimeter tend to be unusually oblong. Since local affine transformation is constant within each triangle and should differ minimally from neighboring transformations, lengthy triangles introduce deformations. To resolve this issue, border triangles of side-to-height ratio exceeding a threshold were rejected.

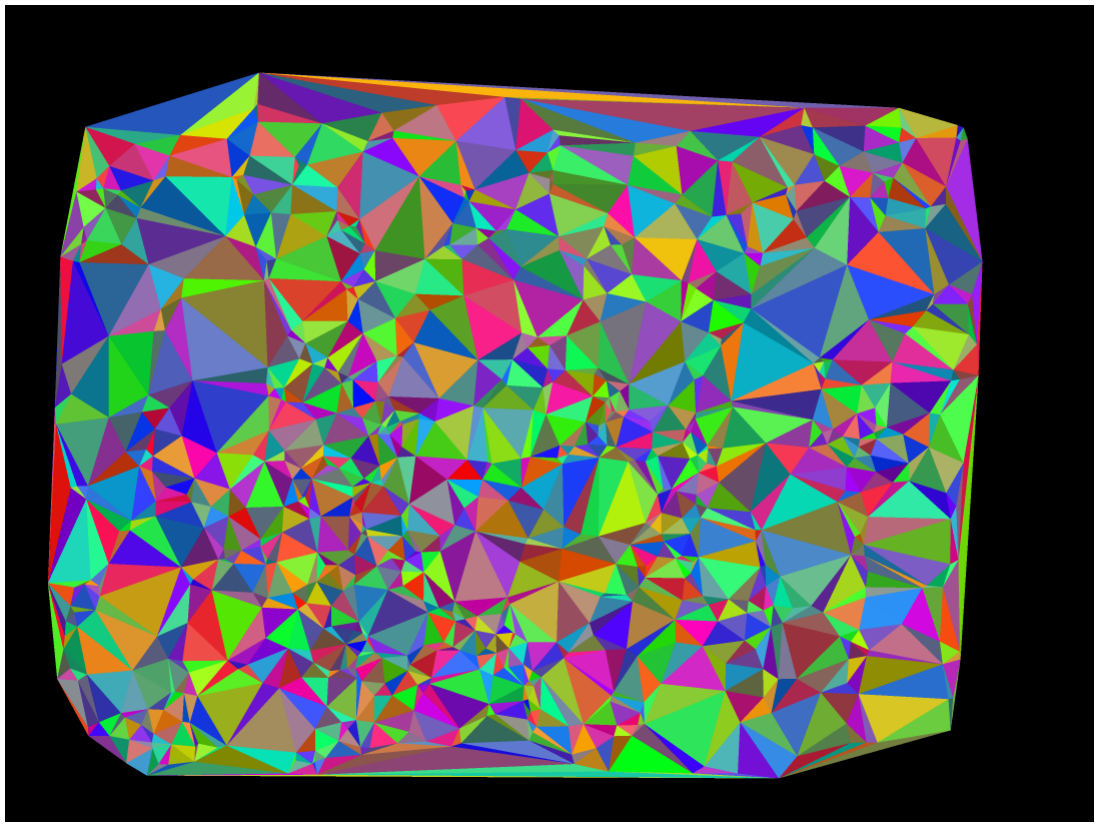


Figure 2.10: Pattern match points triangulation; setup #6;
each triangle is denoted by a random color

2.4.1 Verification

Because no publicly available implementation of the algorithm was found, precise result verification was impossible. A verbose approach was taken instead and every algorithm step was documented. For instance, in Figure 2.11 randomly chosen 64 matches are presented. This form is more visually appealing than form of matches shown in Figure 2.4 and Figure 2.8, as it can be plainly seen that match points do correspond with each other. It confirms that feature matching was implemented correctly. All output images are described in detail on the CD-ROM (see Appendix B).

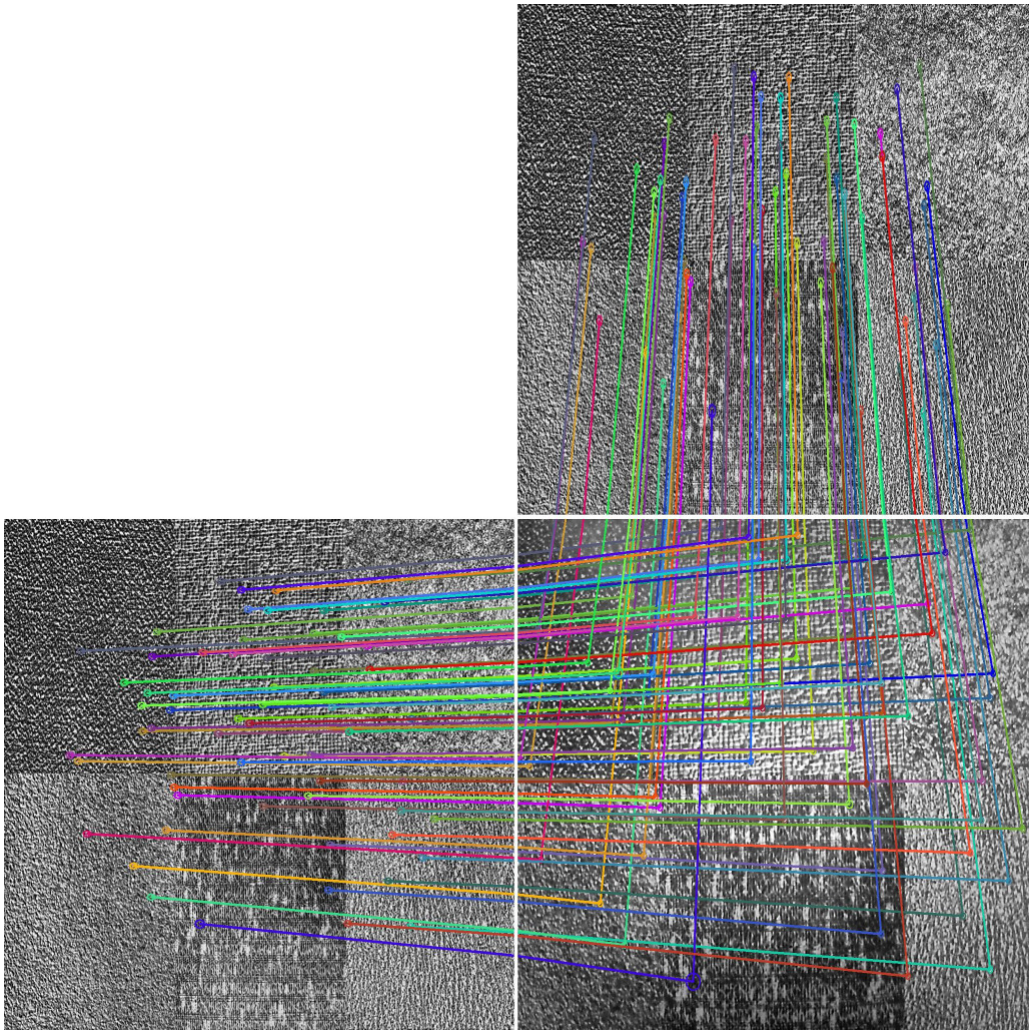


Figure 2.11: Matches between pattern (left bottom, right top) and photograph (right bottom); images are laid out so that horizontal and vertical components of correspondences are visible separately; setup #5

3. Discussion of Results

In this chapter results of both analyzed algorithms are presented and interpreted. They are based on final photographs, if not stated otherwise.

As already explained in Chapter 2, RMSE assessment was not flawless. There were cases of line edges fusion or faulty corner detection, the former due to edges being situated too tightly and the latter due to corners being placed too close to image border. Such images were manually excluded from being processed. Hence number of photographs with detected features varies between algorithms for each setup. For other images ROIs (Regions Of Interest) had to be specified, so that paper edges or background features would not be detected instead of intended pattern line.

3.1 Straightness of original images

Before any algorithm can be appraised, original photographs were examined to estimate room for improvement. This step seems to be frequently omitted in the literature.

Results are presented in Figure 3.1. Both checkerboard (*checker horizontal*, *checker vertical*) and single line (*lines horizontal*, *lines vertical*) patterns were used for plots depicted in the left column. Same checkerboard images that were applied later for parametric calibration were used. As far as single line pattern is concerned, the pattern itself was situated roughly perpendicularly to optical axis and only lines passing horizontally or vertically through image were photographed. It was observed that this way quantization error was lower and outlier (line "appendices") elimination was easier.

For each control image, the *RMS error* value is calculated as RMS distance from respective regression lines of all points detected in the image. In case of checkerboard images a distance is calculated twice for each corner point: once for horizontal regression line and once for vertical. In single line images points of both edges are taken into account, each edge compared with its own regression line. *Image index* refers to image processing order and can be used, in conjunction with right log file, to identify detailed, per-image results.

In the right column a course of one line from a single line pattern photograph is shown, with both edges visible. Longer edge is colored with darker color. Edge points are ordered by their distance from image border corresponding to their orientation. Each line selected for the right column had highest RMSE among setup's control images and also was properly detected in later steps (after rectifications).

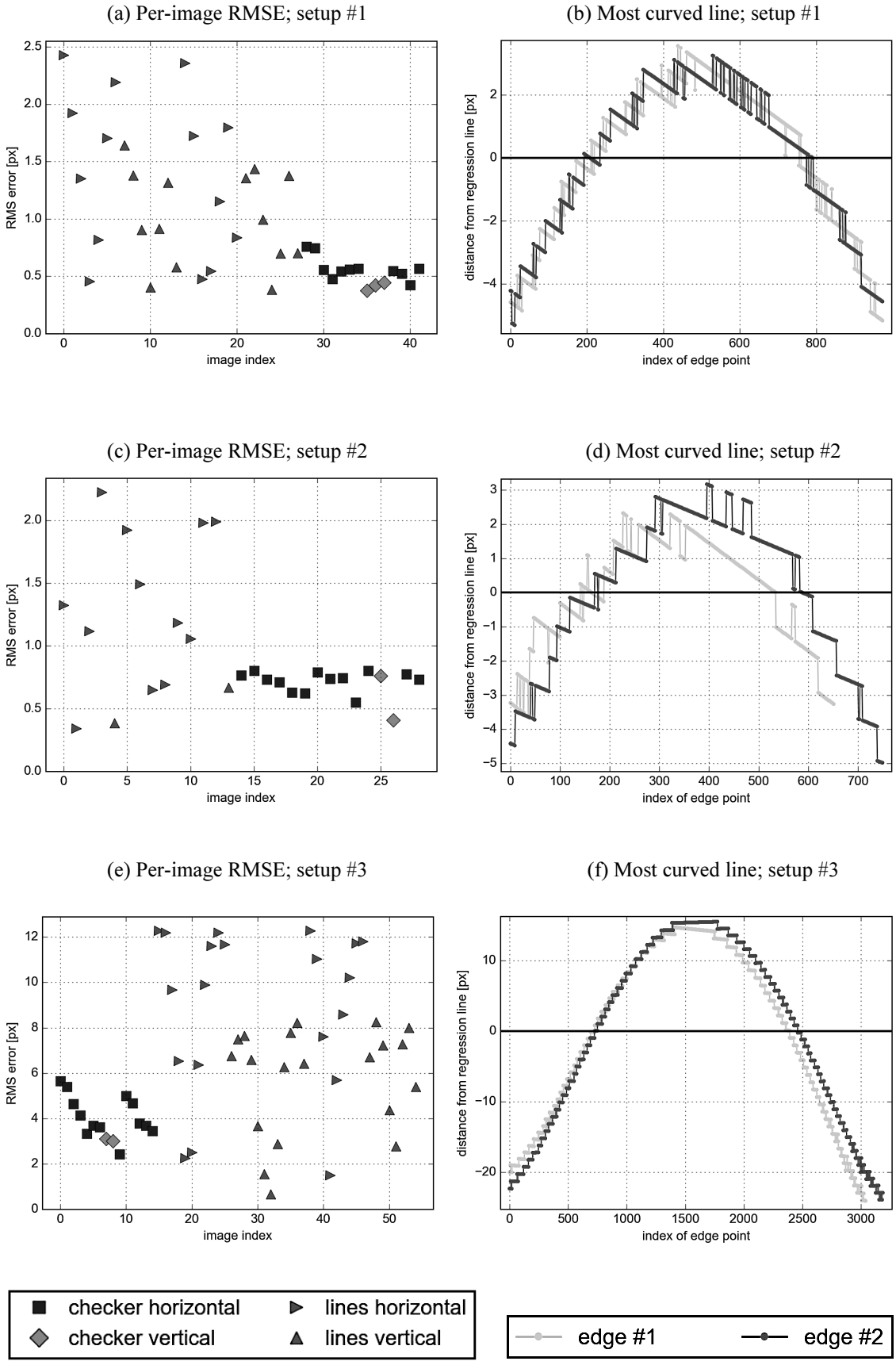


Figure 3.1 a)–f): RMSE of original images

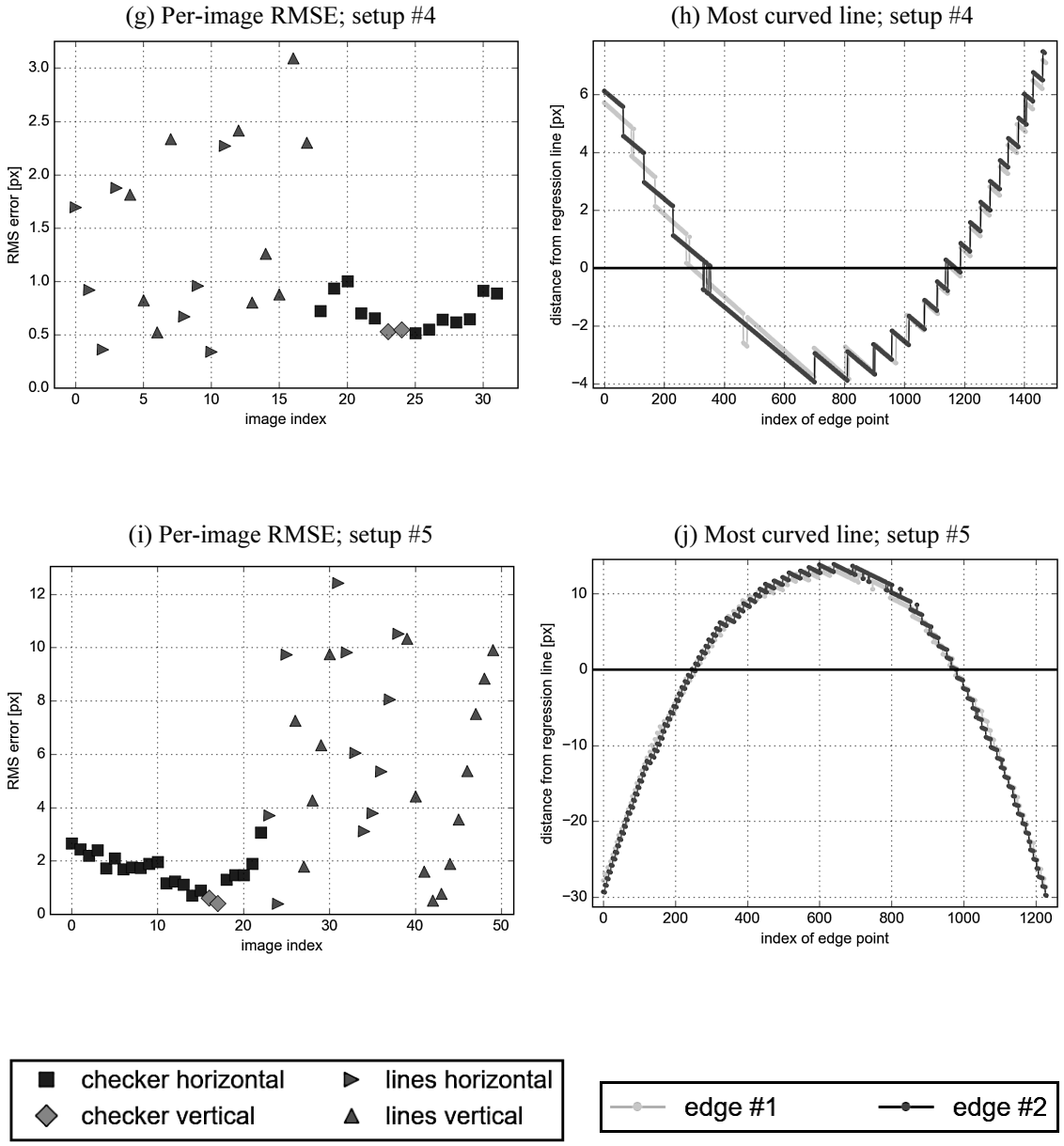


Figure 3.1 g)–j): RMSE of original images

Same layout is used later in Figure 3.6 and Figure 3.15. The *image index* values, however, do not represent same control images between those figures, because after rectifications some control images were not properly processed and had to be discarded. On the other hand, for every setup the same single line pattern image was used in the right column, so that they could be directly compared.

It is noticeable that detected edges do not constitute smooth curves in most cases. This phenomenon is discussed later.

In case of setups #1, #2 and #4 initial distortions were fairly low, with maximal absolute error not exceeding 7 px and most per-image RMSE not exceeding 2.5 px.

Setup #5 is a wide angle camera, therefore large error was foreseeable. High values reported for setup #4 were puzzling, however, especially considering that distortions were not evident on photographs (see Figure 3.2). In order to explain this phenomenon, image resolution should be checked. Setup #3 photographs are approximately 2.55 times wider and 2.4 times higher than setup #5 photographs, so their calculated RMSE, especially before rectification, should not be directly compared.

A better metric, relative to image dimensions, is proposed in (15). Final RMSE (calculated as RMSE of edges and checkerboard corners detected in all control images) values are collected in Table 3.1 and their corresponding ρ values.

$$\rho = \frac{RMSE\,[px]}{d\,[px]} \cdot 10^3 \tag{15}$$

where: $RMSE$ – total RMSE computed for all control images;

d – maximal image dimension

Table 3.1: Total RMSE in absolute and relative values

Setup	Before rectification		Parametric algorithm		Nonparametric algorithm	
	RMSE [px]	ρ	RMSE [px]	ρ	RMSE [px]	ρ
#1	1.38	1.35	0.38	0.37	0.39	0.38
#2	1.34	1.68	0.43	0.54	0.40	0.50
#3	8.29	2.54	0.41	0.13	0.46	0.14
#4	1.62	0.79	1.73	0.84	0.92	0.45
#5	6.68	5.22	1.25	0.98	0.38	0.30

(a) Setup #5 (b) Setup #3

Figure 3.2: Photographs of the single line pattern, featuring most curved line

It can be observed in Figure 3.1 that level of errors in checkerboard images was rather constant in all setups and was noticeably lower than in line images. This is be-

cause a line can be situated arbitrarily close to image border (where distortions are usually highest), while nearly entire checkerboard is always in the image center. For this reason checkerboard alone is not a good tool for straightness evaluation (hence setup #6 was not included in the figure).

It is, however, a source of other valuable information, apart from its obvious role in the parametric algorithm calibration. Compelling checkerboard-based plots were found when the board was parallel to the image plane and was not rotated. Parabolic, smooth curve shapes observed in the Figure 3.3 (a) might suggest that distortion has a uniform nature and therefore could be modeled parametrically. This phenomenon was observed to a moderate degree in setup #1, #2, #3 and #5. In setup #4 and #6 no such shapes were visible. This indicates that either distortion is irregular, or corner extraction was not precise.

(a) Setup #3 (b) Setup #4

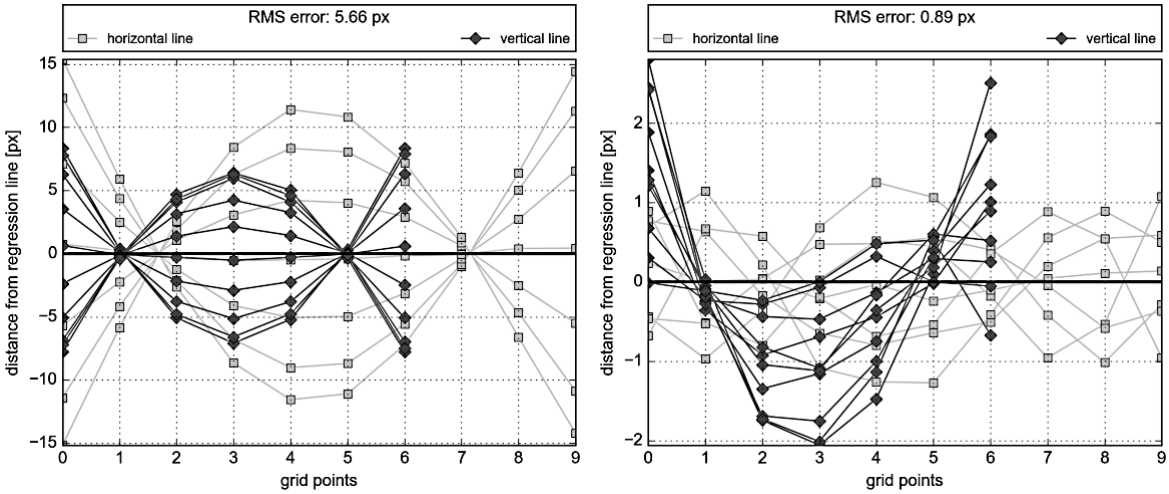
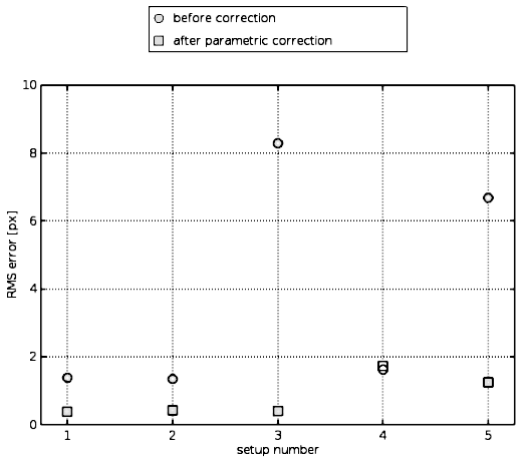


Figure 3.3: Plots of corner distance for board parallel to the image plane and not rotated; grid points are ordered by their distance from relevant image border; RMSE measurement is based on all grid points

3.2 Parametric algorithm

General results of the parametric correction are depicted in Figure 3.4, where RMSE was calculated basing on all edge and corner points found in all control images. Setup #1, #2 and #3 have been corrected by the algorithm to acceptable values (comparing to the state-of-the-art), with setup #3 having impressive relative improvement of 95%. While setup #5 has improved, final outcome is still distorted. Setup #4 has actually worsened.

(a) Full scale plot



(a) Plot scaled down in *y* axis

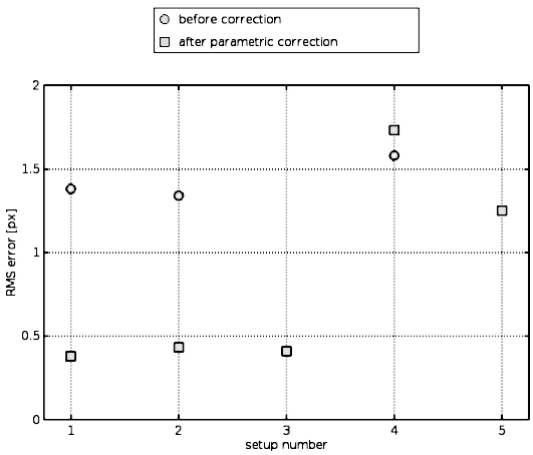
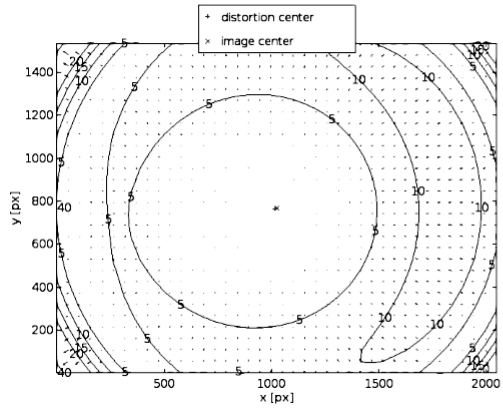


Figure 3.4: Total RMSE before and after parametric correction;
scaled down in (b) for better depiction

It can be stated that calibration has failed for setup #4. Not only are rectified images worse, but also tangential distortion component dominates over radial, which is unusual. The rectification field has been depicted in Figure 3.5 (a); same relationship can be recognized for setup #6 in Figure 3.5 (b). It is notable that they are both compact phone cameras.

In case of setup #4 there was also autofocus, which could not be switched off, and might have contaminated measurements, as noted in the Chapter 1. To verify this conjuncture, *Camera Calibration Toolbox for Matlab* was used. Both *fc* and *cc* parameter uncertainties were lower than 1%, however.

(a) Setup #4



(b) Setup #6

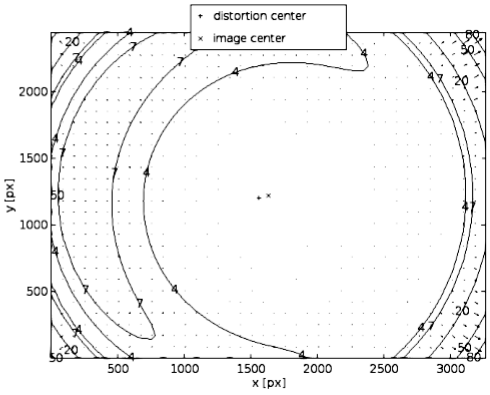


Figure 3.5: Mapping resulting from parametric correction in miscalibrated setups

3.2.1 Per-image results

First of all, it can be observed in Figure 3.6 that in setup #1, #2 and #5 checkerboard corners were very straight after rectification, with per-image RMSE oscillating on 0.1 px and 0.2 px levels. This was to be expected — corner positions are used by the algorithm, therefore they are optimized.

Large RMSE of checkerboard corner positions in setup #3 can be attributed to blur resulting from narrow depth of field. A plot of corner distances is presented in Figure 3.7, along with the photograph. Corners present in the lower half of image (low grid point indexes) exhibit larger distances from their respective regression lines than focused corners found in upper half.

RMSE measurements based on line patterns allow more comprehensive analysis. In setup #1 all rectified lines are curved to an equal degree, so it can be concluded that this camera follows the adopted model well. Errors of single points are usually lower than 2 px, and those above 1.5 px seem to stem from overall low image quality rather than residual distortion.

In setup #2 there are 3 lines curved averagely 52% more than others. One of such lines is depicted both in Figure 3.1 (d) and Figure 3.6 (d). Close inspection shows that line's concavity has shifted during rectification. It has happened only to lines lying on image borders, where parametric model is less accurate. Similar observations have been made for setup #3 on 4 vertical lines visible in Figure 3.6 (e), whose distortion level exceeds 0.5 px.

Another argument supporting presumption that calibration in setup #4 was invalid can be gathered from Figure 3.6 (g). — 3 lines have been deformed to higher extent than any line was in original photographs. Moreover, resulting line shape presented in Figure 3.6 (h) is no longer parabolic.

Plot for setup #5 shown in Figure 3.6 (i) is very characteristic. While all lines are more straight than in the beginning and most have been acceptably rectified, many of them are still visibly distorted (see Figure 3.8). Their curvature functions have preserved same concavity and similar shape, which means that distortions in those areas were larger than model could compensate, while simultaneously maintaining good rectification in image center.

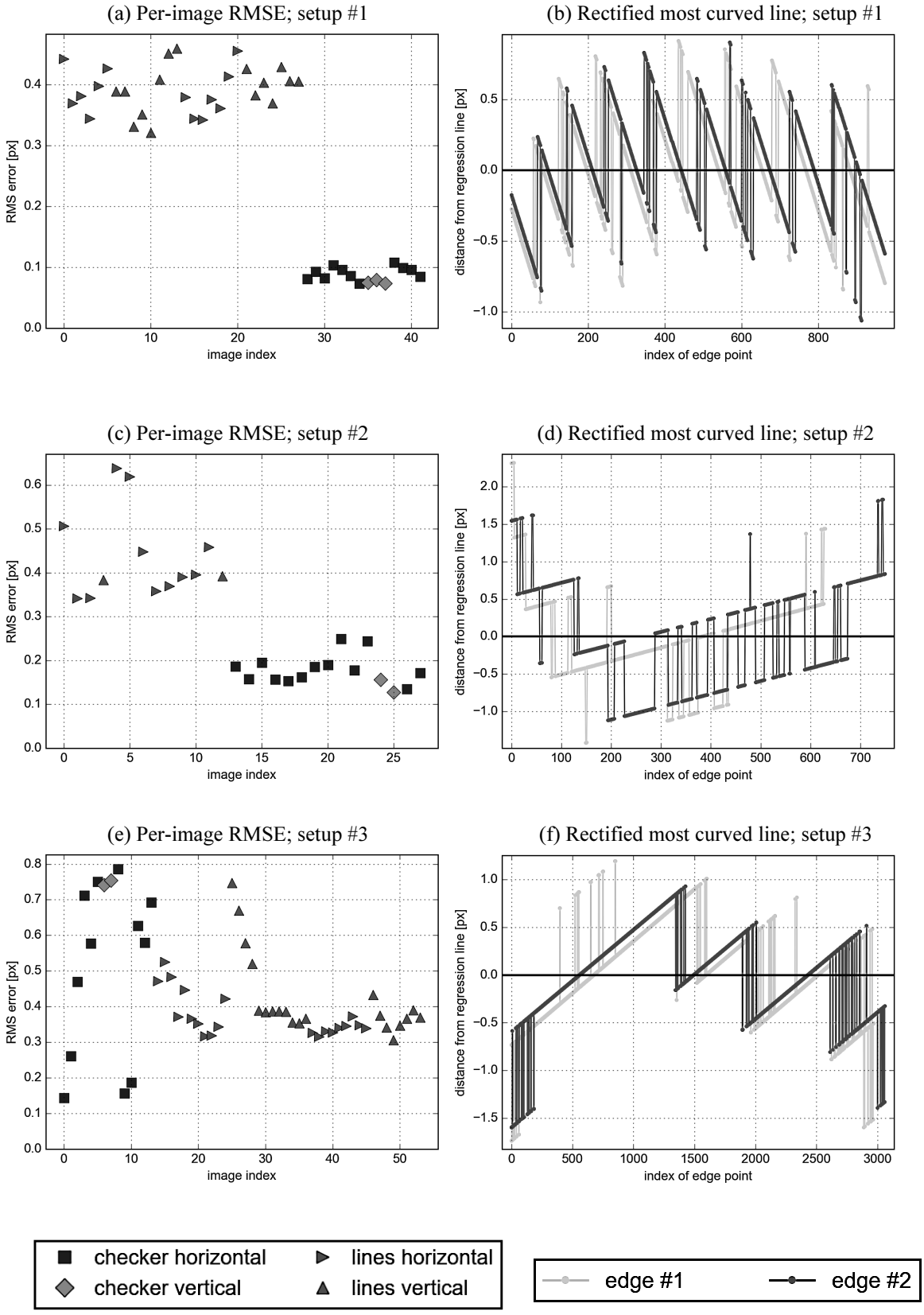


Figure 3.6 a)–f): RMSE of images corrected by the parametric algorithm

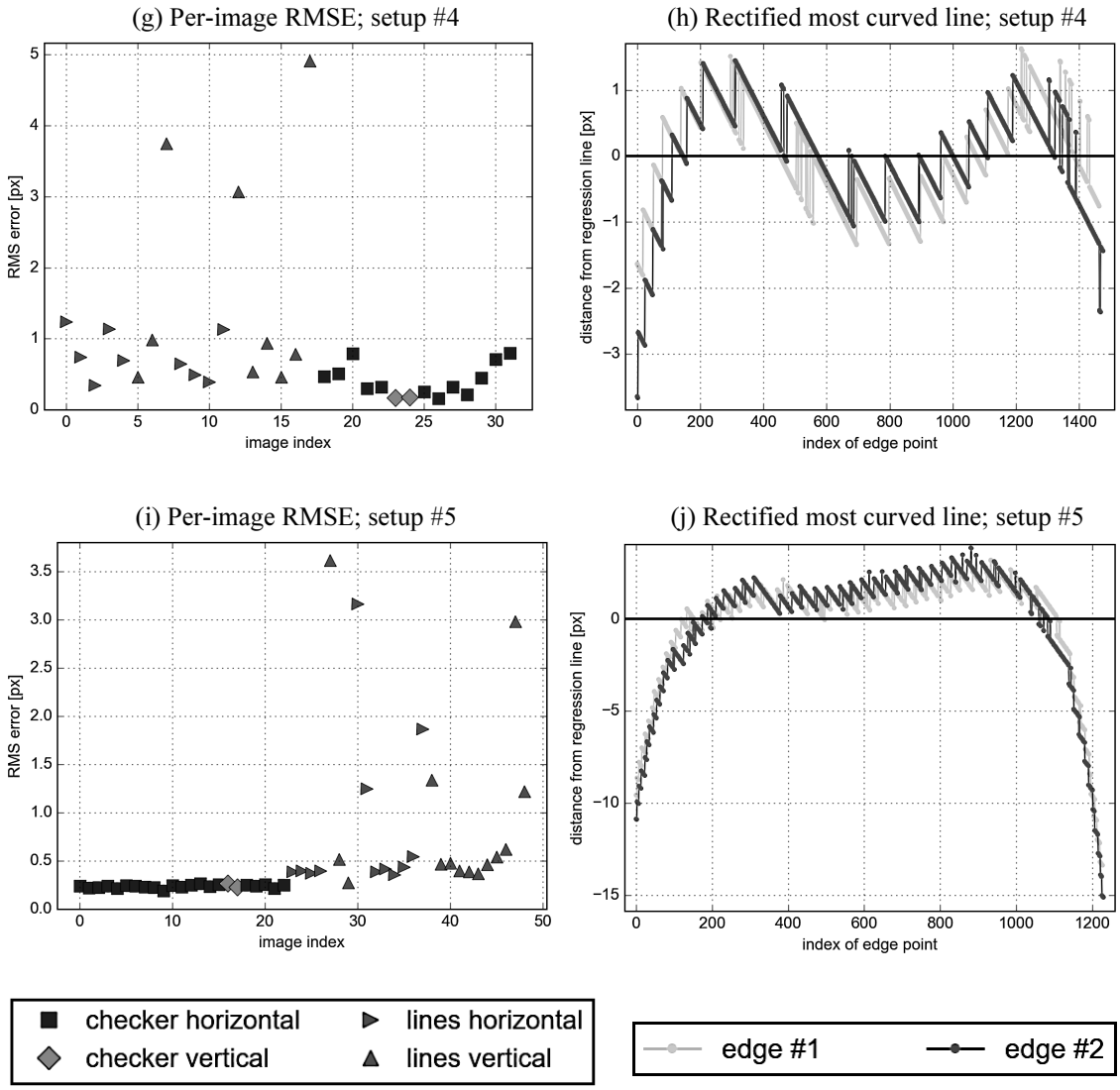


Figure 3.6 g)-j): RMSE of images corrected by the parametric algorithm

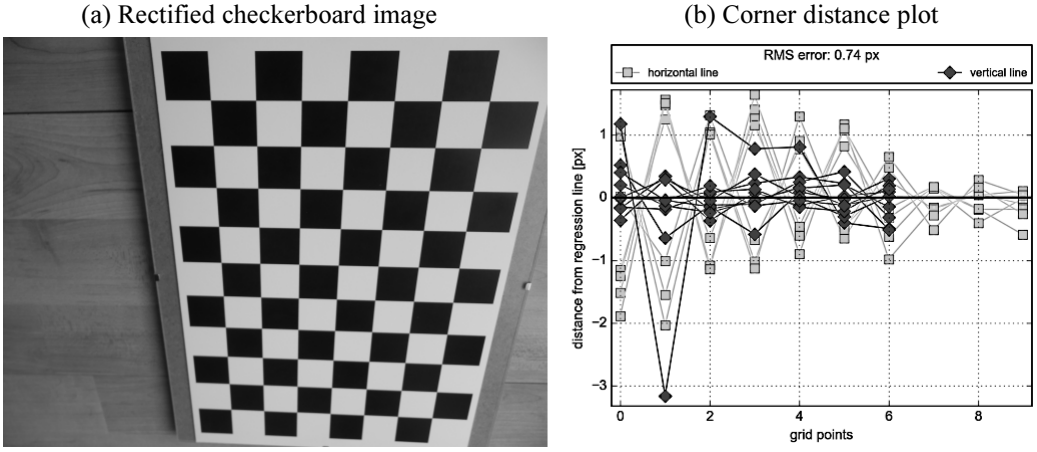


Figure 3.7: Results of the parametric algorithm for a blurred checkerboard image; setup #3;
horizontal and *vertical* refer to line alignment relatively to the pattern;
RMSE measurement is based on all grid points

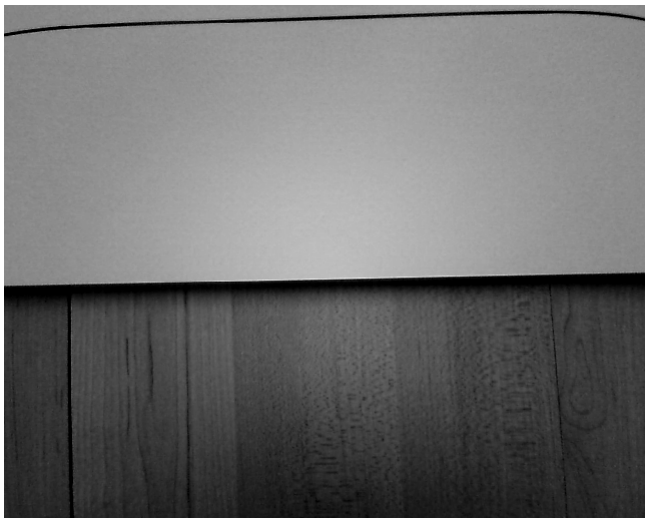


Figure 3.8: Image featuring most curved line, rectified by the parametric algorithm, but still not straight; setup #5

It should be stressed that although RMSE of 0.4 px is larger than most values reported in the literature, the measurement itself was not very accurate. Canny edge detection works only on pixel level, resulting in high quantization noise. For instance, rectified line studied in Figure 3.6 (b) is very straight, oscillating usually not further than 0.5 px from regression line. Because of pixel size, the edge points could not be closer to the regression line. Yet, RMSE in this image is equal to 0.38 px. Image of line examined in (b), with marked edges and regression lines, is depicted in Figure 3.9.

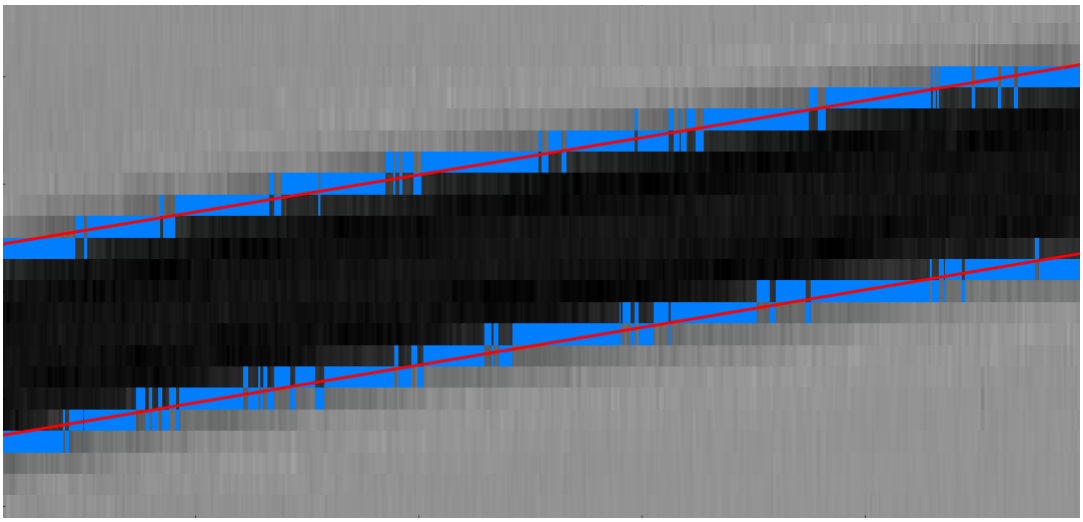


Figure 3.9: Edge pixels (blue) and regression lines (red) in a rectified image; pixel aspect ratio is not equal to unity; setup #1

Moreover, on some images line edges have oscillated not due to nonlinearity, but because of image quality being not high enough. There are such oscillations in

Figure 3.6 (f). It is clear that in reality the edge lies between alternating points, thus its distance from regression line would be shorter.

3.2.2 Susceptibility to errors

During one of initial tests, calibration was performed with setup #5 using photographs depicted in Figure 3.10. The aim was to check algorithm outcome when no corners were located near borders.

Results are shown in Figure 3.11. As expected, anomalies can be identified in corners of the mapping (b). Specifically there is a change of displacement vector length's monotonicity, which leads to worse rectification in image corner, as visible in (d). This is due to model being extrapolated to domain of whole image. The algorithm does not indicate which parts of the mapping are thus unreliable.

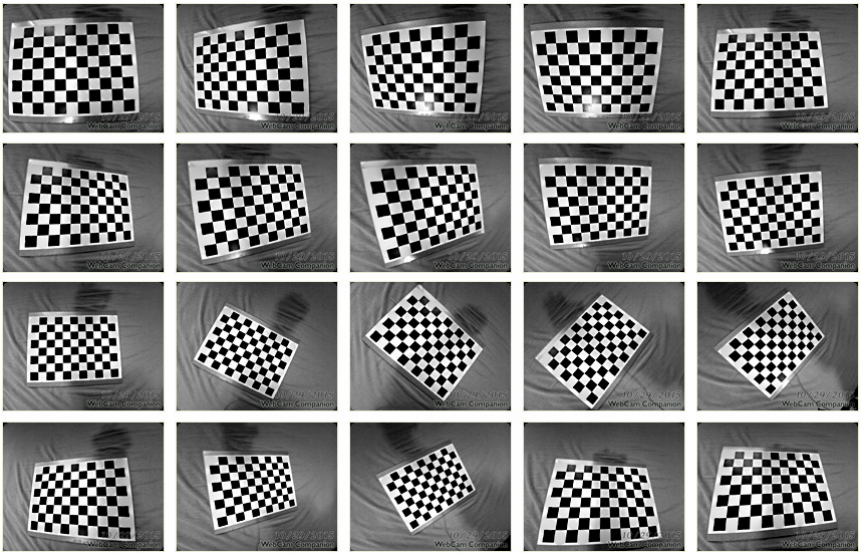
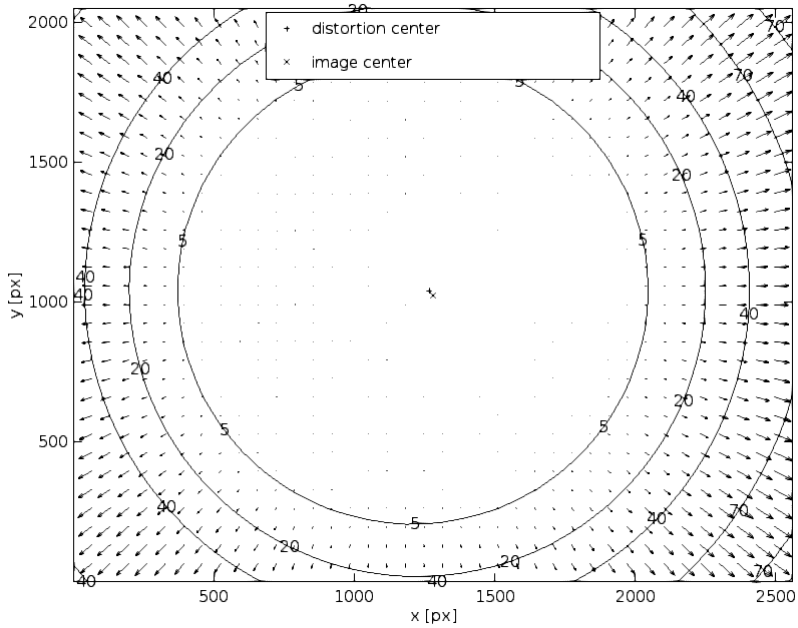


Figure 3.10: Calibration photographs used for test; setup #5

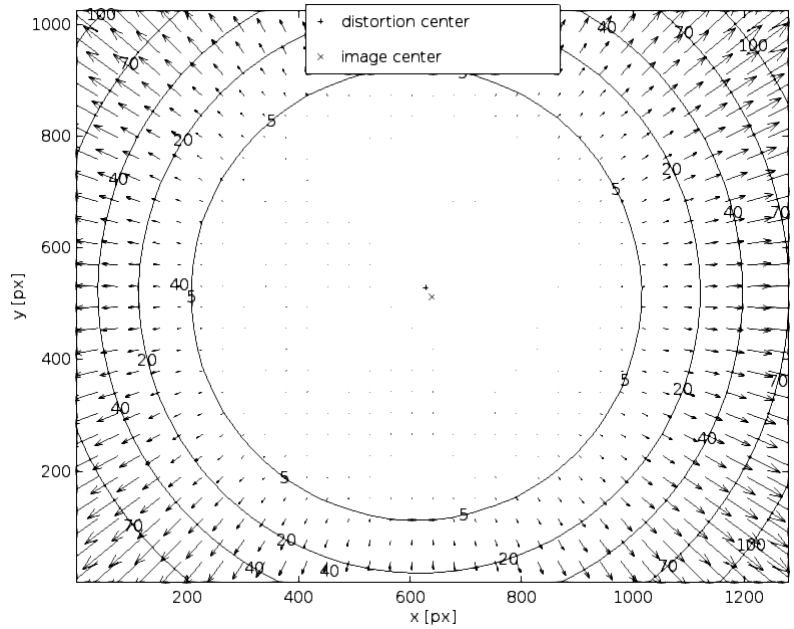
3.2.3 Time complexity

Time measurement were carried out on a PC equipped with Intel Core 2 Quad Q3000 processor and included: image reading, corner detection and complete intrinsic parameter calculation. `CALIB_CB_FAST_CHECK` flag was used to speed up checkerboard search ("*can drastically speed up the call in the degenerate condition when no chessboard is observed*" [19], but does not affect precision if corners are found). Results are presented in Table 3.2. Elapsed time depended both on image quality (sharpness) and resolution. During initial test, calibration based on 17 sharp images obtained from setup #7 (largest tested, in terms of photograph dimensions) took 32 seconds.

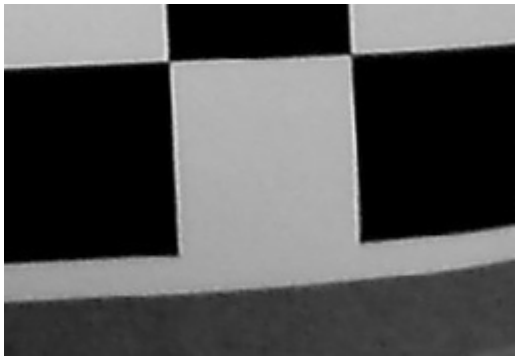
(a) Mapping resulting from good parametric correction (final photographs)



(b) Mapping resulting from bad parametric correction (photographs depicted in Figure 3.10)



(c) Corner of photograph rectified by (a)



(d) Corner of photograph rectified by (b)

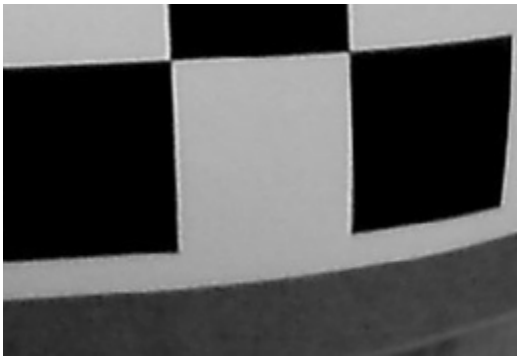


Figure 3.11: Differences in image corner caused by different mappings; setup #5

Table 3.2: Time performance of the parametric algorithm

Setup	Time elapsed [s]	Number of input images	Discarded images (due to not detected corners)
#1	2	14	0
#2	1	15	0
#3	17	15	0
#4	8	15	1
#5	4	23	2
#6	58	16	2

3.3 Nonparametric algorithm

For each setup, 2–3 pairs of pattern photographs were taken, each pair featuring different texture. Then best pairs were selected, with criterion being percentage of preserved photograph pixels. It ought to be not confused with size of valid rectified images (i.e. number of interior, nonblack pixels). An example is presented in Figure 3.12. Photograph has been transformed by inverse of estimated homography between itself and the pattern in (c). Although result is smaller, no pixels were lost. They have been sampled down instead. On the other hand, some pixels have been discarded during formation of (d), due to being located outside of triangulation's perimeter.

Results are depicted in Figure 3.13. Total RMSE (all edges and corner points in all control images) has improved for all setups, albeit it is still high in setup #4. However, as result of the algorithm's approach, other factors must be taken into account in order to rate the outcomes. They have been collected in Table 3.3. Column *Pattern* contains index of used pattern, with 2 meaning the most detailed one.

First of all, it is noticeable that the **chunking** strategy provided very few additional matches. It is insignificant next to the number of initial matches.

The ratio of number of inliers left after loop validation to image resolution varies among setups. While it could influence resolution (and thus quality) of the reverse distortion field, no such relation was observed (as setup #3 performed better, in terms of RMSE, than setup #4, despite difference in the ratio). For reference, in [25] number of inliers was reported to be equal to about 8000. Supposing maximum camera resolution was used, the ratio was equal to 0.98 matches per kpx.

As already stated, percentage of preserved pixels is an important quality metric. RMSE of 0.4 px in setup #2 is an insignificant improvement, considering that a third of

every image was lost in process (see Figure 3.14). In other setups, where camera quality was better, this percentage was notably higher.

(a) Pattern

(b) Photograph of the pattern

(c) Photograph warped by homography

(d) Rectified photograph

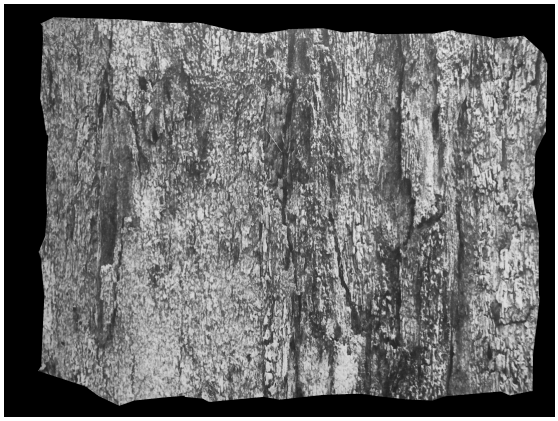
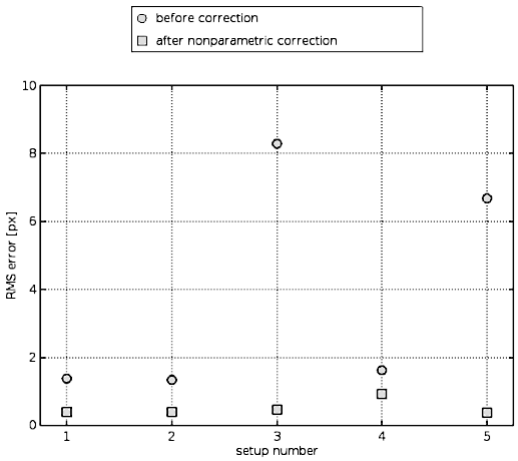


Figure 3.12: Illustration of border pixel loss and photograph[1] resizing; setup #3

(a) Full scale plot



(b) Plot scaled down in *y* axis

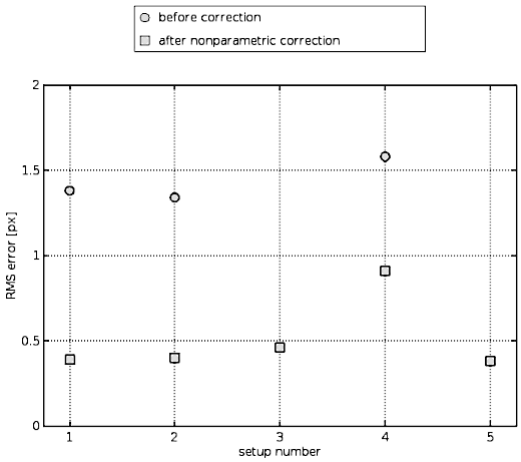
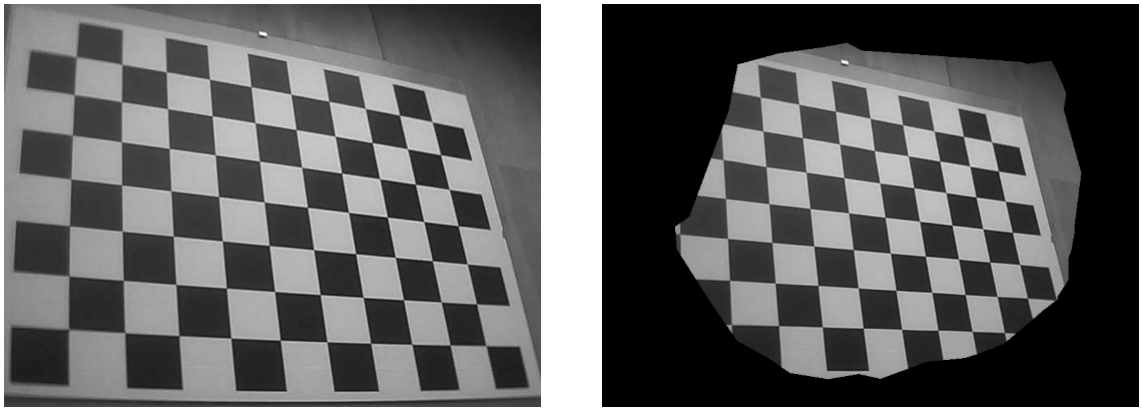


Figure 3.13: Total RMSE before and after nonparametric correction; scaled down in (b) for better depiction

[1] Copyrights to the photographed pattern belong to P. Kratochvil.

Table 3.3: Nonparametric algorithm results (regarding matches)

Setup	Pattern	Feature matches	Matches gained through *chunking*	Inliers	Inliers per image kilopixel	Percentage of discarded pixels
#1	1	1526	2	1493	1.90	8.1%
#2	0	944	6	639	1.33	33.6%
#3	0	3953	0	3176	0.40	7.3%
#4	1	1727	21	1603	0.51	6.6%
#5	2	3093	13	2876	2.19	10.2%
#6	0	1318	0	811	0.10	14.5%
#7	1	6951		6728	0.67	10.9%



(a) Original photograph (b) Rectified photograph

Figure 3.14: Worst encountered case of field of view depletion; setup #2

3.3.1 Per-image results

In case of the nonparametric algorithm, RMSE measurements based on checkerboard have frequently failed (i.e. the board could not be found, because field of view has been diminished). When they did succeed, RMSE no longer was on the extremely low level encountered in the parametric algorithm.

In Figure 3.15 (a) two lines stand out as badly corrected. Closer inspection of Figure 3.16 shows that left one is indeed still slightly curved. Right one, on the other hand, regularly alternates between 3 pixels, meaning that it is not extracted entirely correctly.

Results for setup #2 are less reliable, because remaining field of view is small. Moreover, less lines were found (only 10).

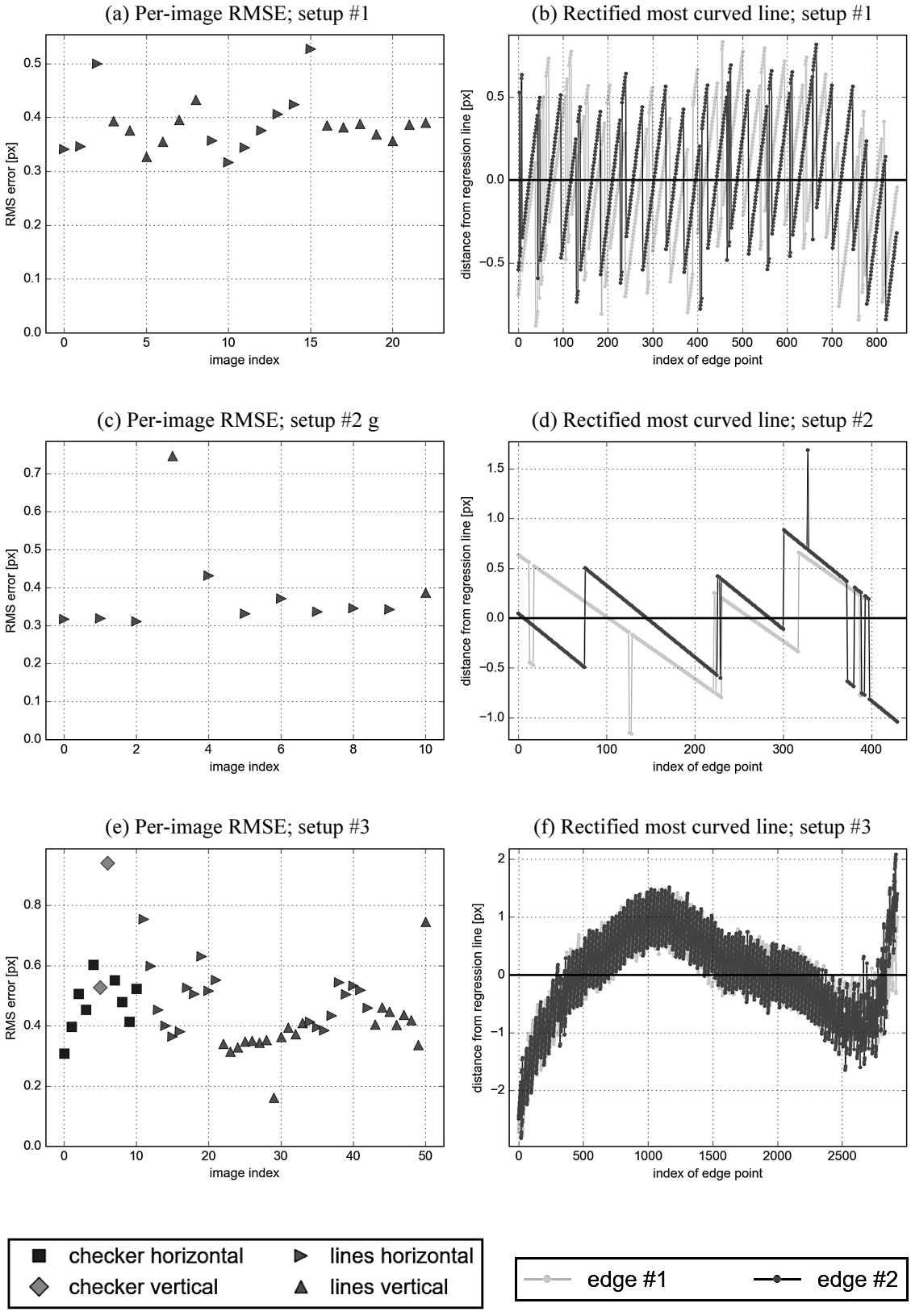


Figure 3.15 a)–f): RMSE of images corrected by the nonparametric algorithm

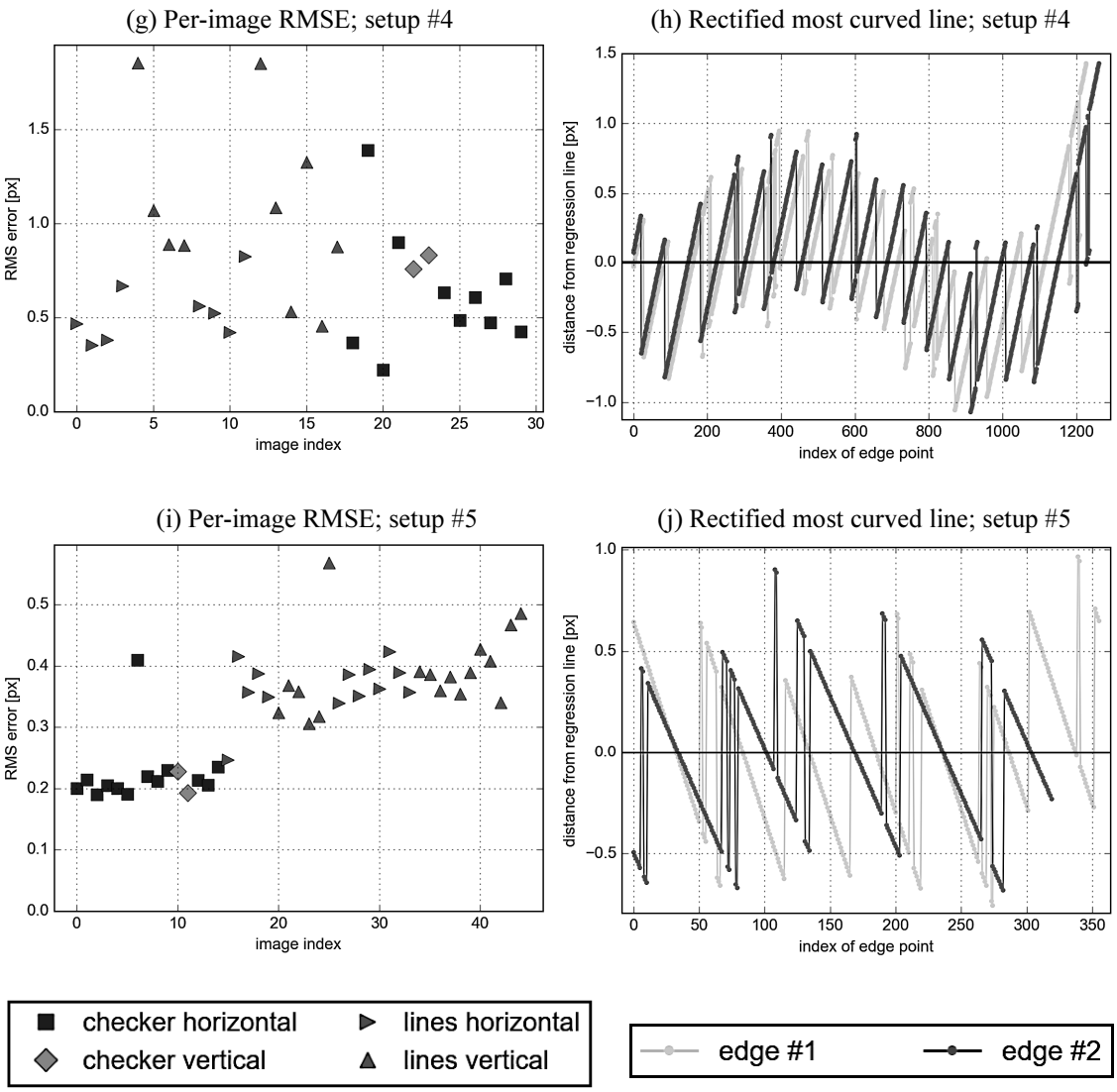


Figure 3.15 g)-j): RMSE of images corrected by the nonparametric algo-

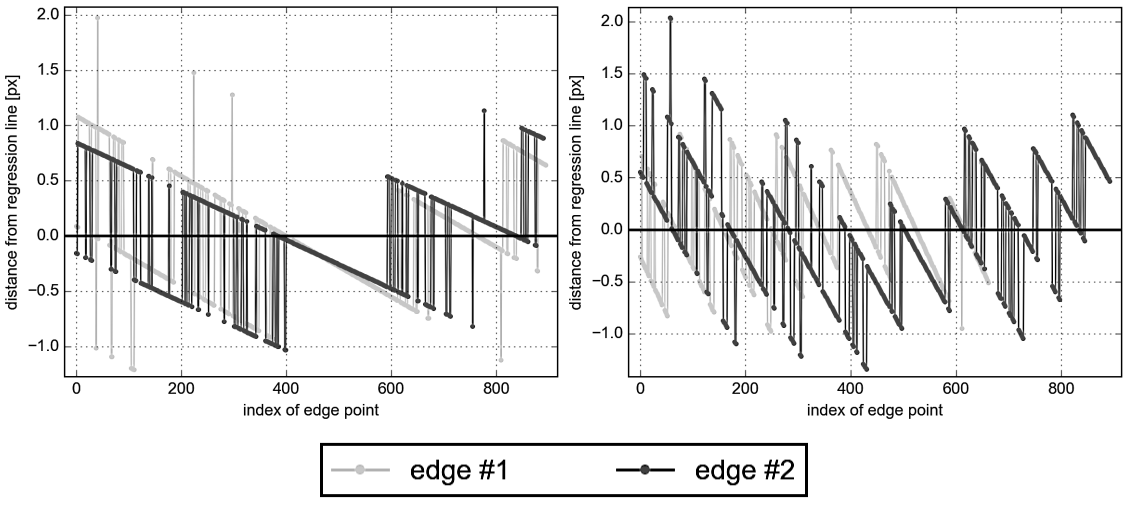


Figure 3.16: Curvature of 2 most distorted lines (after rectification); setup #1

Three remarks can be made on correction of setup #3. Firstly there are 3 horizontal lines in Figure 3.15 (e) with per-image RMSE exceeding 0.6 px. One of them is depicted in Figure 3.15 (f). They are not completely straight, but the parts most distant from regression line are edge beginning and end. Secondly, there is a vertical line with 0.75 px RMSE. After rectification it is situated right beside image border and is still clearly curved (see Figure 3.17). Both this phenomena are explained in section 3.3.2. Finally there is one vertical line with very low RMSE. This line happens by chance to be well fit in the pixel grid.

(a) Rectified line photograph

(b) Line curvature

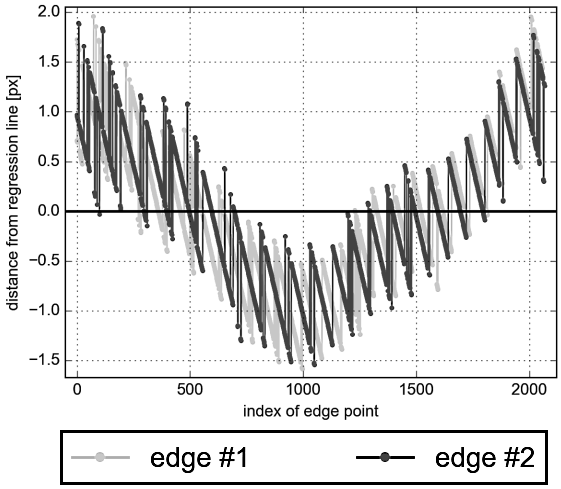


Figure 3.17: Vertical line rectified by the nonparametric algorithm, but still curved; setup #3

There is no straightforward explanation for result diversification in setup #4. Some line were rectified well, as in case of the one depicted in Figure 3.15 (h). Others that were not previously severely distorted, have retained their shape (see Figure 3.18). Moreover, when correction was calculated basing on the other pair of texture images, there was an impressive amount of inliers (7231), but resulting RMSE was even worse (1.32 px). However disappointing these results are, they are both better than parametric correction outcomes.

As far as setup #5 is concerned, rectification results are uniform, with the exception of one vertical line. This line happens to be somewhat diagonal after rectification, resulting in high quantization noise (see Figure 3.19). It is also curved in its end, similarly to setup #3 lines.

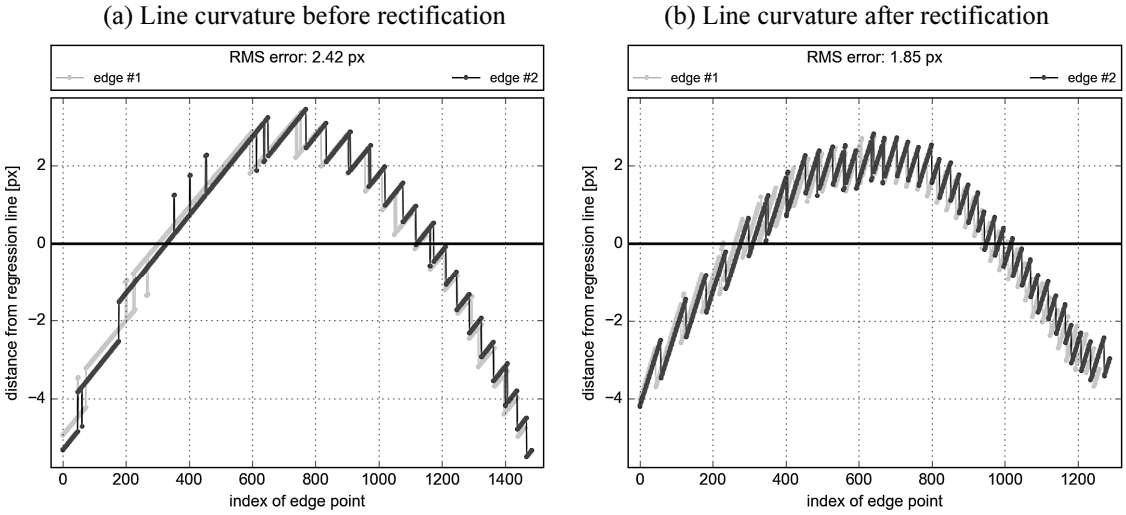


Figure 3.18: Same line before and after nonparametric correction; setup #4; RMSE measurement is based on all grid points

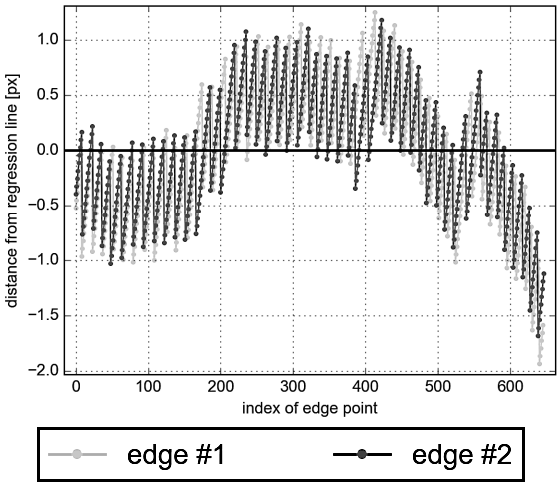
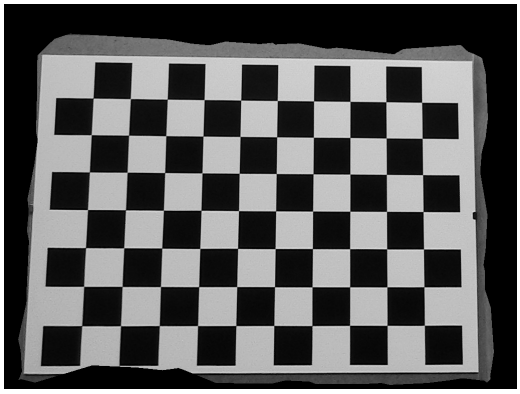


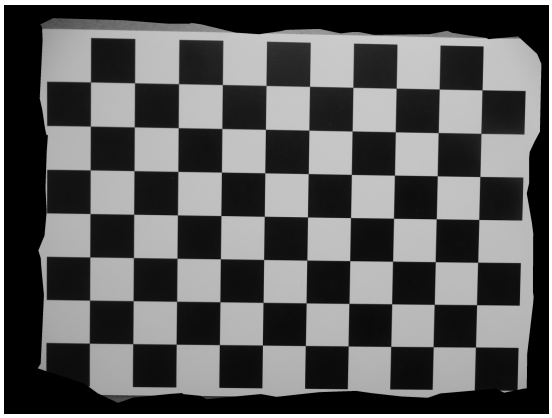
Figure 3.19: Diagonal line curvature after rectification; setup #5

A rectified image is presented in Figure 3.20 for every camera, excluding setup #2 (due to high pixel loss). Checkerboard was selected for visual purposes. Modulus of final reverse distortion field from setup #7 is depicted in Figure 3.21. Despite jagged edges (resulting from implementation details), it is comparable to [9].

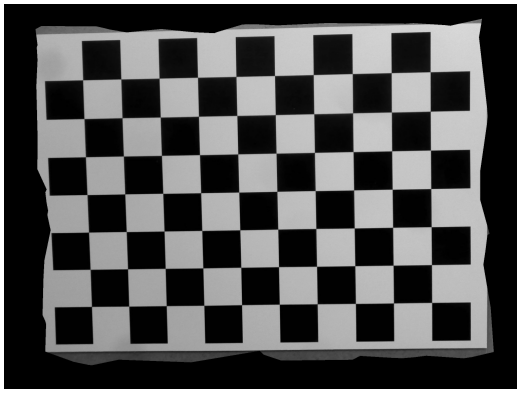
(a) Setup #1



(b) Setup #3



(c) Setup #4



(d) Setup #5

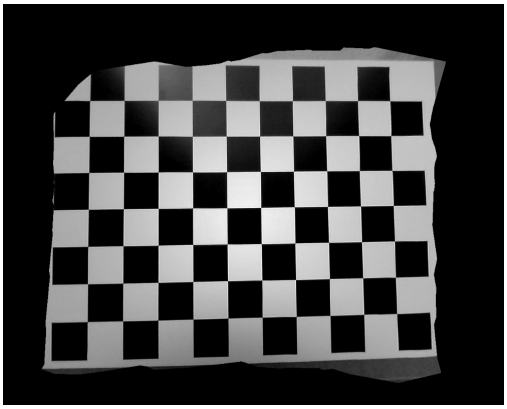


Figure 3.20: Images rectified by the nonparametric algorithm

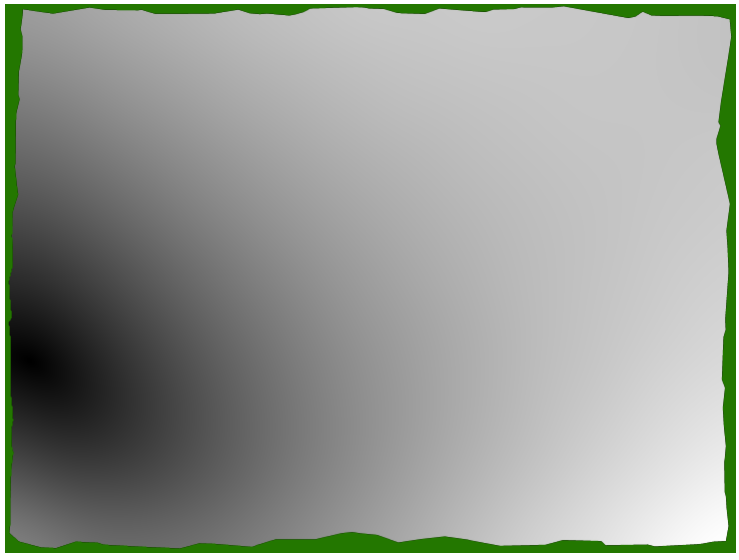


Figure 3.21: Modulus of the reverse distortion field; setup #7

3.3.2 Implementation-specific details

The nonparametric algorithm avoids its parametric counterpart's extrapolation flaw (explained in section 3.2.2), by not including into calculations pixels lying outside reverse distortion field. In the studied cases this led to field of view diminishment. While this stems from algorithm design, observed results were worse than anticipated.

Although **loop validation** was implemented, good matches on the field's boundary were still marked as outliers, while some bad matches remained, introducing obtrusive artifacts. As explained in section 2.4, three **smoothing** algorithms were tested. First one was least-squares 2D polynomial approximation. Polynomials of order 2, 3 and 4 were used, but it transpired that neither was able to fit well into the field's dynamic range (see Table 3.4).

Then Gauss filter was tested. Because the field's area had shrunk after simple filtering, NaN pixels had to be given a value first (using nearest neighbor interpolation). This worked well, as long as values used for interpolation were correct. Otherwise filtering lead to propagation of incorrect displacement values, which resulted in high distortions near borders. It was the case in setup #3, where lines tended to be curved at both ends. In Figure 3.22 a gleam can be observed near edges, indicating that value propagation took place.

Other approach was to use filters with different sizes; to fill out areas left by bigger filters. The drawback was that the field was less smooth near edges and thus some artifacts were still present. Out of these 2 methods each was chosen individually for each photograph pair, so that results would be best (although total RMSE difference was insignificant).

3.3.3 Time complexity and memory consumption

Time performance for one photograph pair is collected in Table 3.5. Tests were carried out on the same PC as for parametric algorithm and didn't include smoothing, because many methods were evaluated, but only one contributed to final results. Much intermediate data was saved to storage during time measurements, but it didn't influence performance visibly, as most lengthy task was usually the feature matching.

While memory consumption was not directly measured, it should be noted that during feature detection 32-bit address space assigned to Python process (effectively 2 GB) had not been enough. Hence algorithm was developed for 64-bit systems.

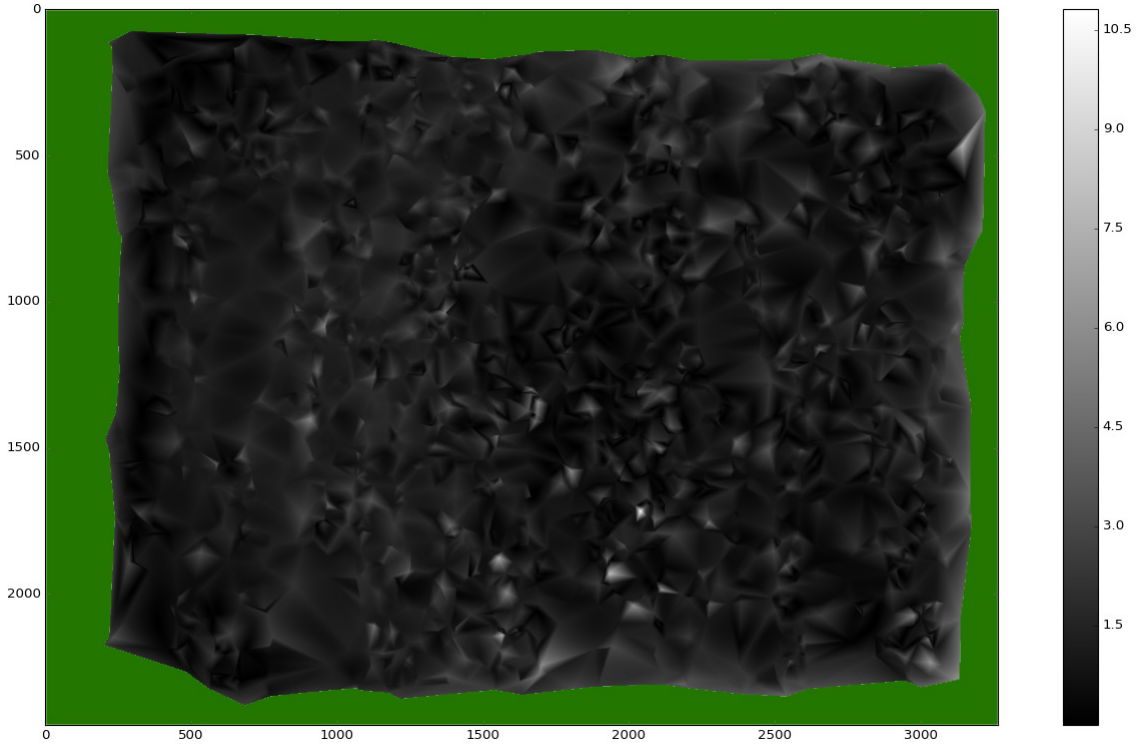


Figure 3.22: Absolute difference between initial and Gauss-smoothed reverse distortion field; setup #3

Table 3.4: Mean absolute difference between initial reverse distortion field and its smoothed versions

Setup	2nd order polynomial	3rd order polynomial	4th order polynomial	Gauss filter
#1	2,15	4,44	13,81	0,39
#2	1,55	1,04	3,34	0,47
#3	13,29	12,86	61,45	1,57
#4	2,26	10,39	23,45	1,00
#5	8,54	4,46	18,08	1,03
#6	2,22	11,95	57,20	2,15

Table 3.5: Time performance of the parametric algorithm

Setup	Time elapsed [min]	Time relative to product of image and pattern keypoint counts [µs]
#1	1.4	2.15
#2	1.1	0.40
#3	11.5	1.05
#4	4.1	0.89
#5	7.2	0.67
#6	3.6	0.56
#7	28.6	1.66

Conclusions

Many algorithms were considered for analysis in the paper. Finally two algorithms were singled out: a popular implementation of the Conrady-Brown model and own implementation of a promising algorithm proposed by [9]. Results were generally worse than state-of-the-art, but even so have allowed to perform deductions.

Three cameras were corrected to similar degree by both algorithms. Phone camera correction performed by the nonparametric algorithm, while not as good as anticipated, has successfully lowered RMSE. It was impossible to fully correct high distortions introduced by the wide angle camera with the parametric algorithm. The sixth order polynomial model is not sufficient.

The parametric algorithm applies rectification blindly to whole image, even if in some parts there were not enough (or were not any) points to deduce about distortion. Thus calibration images should be carefully planned to fill whole image space. Nonetheless model fitting is prioritized in image center. The algorithm is quick (usually taking few seconds). The results are highly portable — rectification maps can be easily recalculated using only a handful of parameters (distortion coefficients and distortion center position), so the algorithm can be easily used in conjunction with a camera database.

Nonparametric algorithm estimates its area of validity. This can, however, dramatically diminish field of view when camera quality is very low. As noted in [9], one of advantages of the nonparametric algorithm is that its results are not final and could be processed further to attain better rectification. This particular implementation was shown to introduce small distortions near borders due to piecewise interpolation scheme. The algorithm needs only 2 photographs, but introduces a homography. Data processing is very demanding in terms of computing power, even for images as small as SXGA. Since algorithm is nonparametric, full rectification maps have to be stored (2 floating point arrays of same size as images).

Obtained results have shown that Canny edge detector is not suitable for RMSE estimation when subpixel rectification is expected. In most cases 0.4 px mean errors were caused by quantization noise, while distortion residual was imperceptible. However, there were also cases when distortions far surpassed the noise.

Implementation of the *Grompone von Gioi et al.* algorithm is far from perfect. Following improvements could be considered in future:

- better piecewise affine interpolation, so that there would be no discontinuities on triangle boundaries and separate smoothing would not be necessary,
- triangulation enhancement — oblong border triangles are currently simply discarded, perhaps better approach is achievable,
- mitigation of photograph scale change due to introduced homography,
- more robust outlier elimination,
- other feature detection algorithms can be incorporated, as SIFT was designed to detect features on different scales and this is not necessary (and is patented for commercial use),
- a faster feature matching algorithm can be chosen,
- support for regular patterns (e.g. a very dense checkerboard); characteristic points could be extracted by centroid search instead of corner detection, which is faulty in such cases.

Additionally precision of edge detection in RMSE evaluation module might be enhanced. A more precise variation [5] of Canny edge detector or entirely different technique such as [28] could be used to achieve subpixel accuracy.

Other line pattern should also be considered. Line printed on A4 paper was in some cases too short and it was impossible to capture it passing through whole image and focus at the same time.

References

[1] Bouguet, Jean-Yves. *Camera Calibration Toolbox for Matlab*. Computer software. Web. 23 Dec. 2015. <http://www.vision.caltech.edu/bouguetj/calib_doc/>.

[2] Canny, John. "A computational approach to edge detection." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 6 (1986): 679-698.

[3] Claus, David, and Andrew W. Fitzgibbon. "A rational function lens distortion model for general cameras." *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE, 2005.

[4] Devernay, Frédéric, and Olivier Faugeras. "Straight lines have to be straight." *Machine vision and applications* 13.1 (2001): 14-24.

[5] Devernay, Frédéric. "A non-maxima suppression method for edge detection with sub-pixel accuracy." (1995).

[6] Fischler, Martin A., and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." *Communications of the ACM* 24.6 (1981): 381-395.

[7] Fitzgibbon, Andrew W. "Simultaneous linear estimation of multiple view geometry and lens distortion." *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE, 2001.

[8] Fryer, John G., and Duane C. Brown. "Lens distortion for close-range photogrammetry." *Photogrammetric engineering and remote sensing* 52 (1986): 51-58.

[9] Grompone von Gioi, et al. "Towards high-precision lens distortion correction." *Image Processing (ICIP), 2010 17th IEEE International Conference on*. IEEE, 2010.

[10] Hartley, Richard, and Tushar Saxena. "The cubic rational polynomial camera model." *Image Understanding Workshop*. Vol. 649. 1997.

[11] Hartley, Richard, and Andrew Zisserman. "Multiple view geometry in computer vision." (2003).

[12] Hartley, Richard, and Sing Bing Kang. "Parameter-free radial distortion correction with center of distortion estimation." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 29.8 (2007): 1309-1321.

[13] Heikkilä, Janne, and Olli Silvén. "A four-step camera calibration procedure with implicit image correction." *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. IEEE, 1997.

[14] Heikkilä, Janne, and Olli Silvén. "Calibration procedure for short focal length off-the-shelf CCD cameras." *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*. Vol. 1. IEEE, 1996.

[15] Hugemann, Wolfgang. "Correcting lens distortions in digital photographs." *Ingenieurbüro Morawski + Hugemann: Leverkusen, Germany* (2010).

[16] Hughes, Ciarán, et al. "Review of geometric distortion compensation in fish-eye cameras." (2008): 162-167.

[17] Li, Hongdong, and Richard Hartley. "A non-iterative method for correcting lens distortion from nine point correspondences." *OMNIVIS 2005* 2 (2005): 7.

[18] Lowe, David G. "Distinctive image features from scale-invariant keypoints." *International journal of computer vision* 60.2 (2004): 91-110.

[19] OpenCV. Program documentation. Vers. 2.4.11. Web. 23 Dec. 2015. <http://docs.opencv.org/2.4.11/>.

[20] Park, Dae Hyuck, et al. "Distortion Center Estimation Technique Using the FOV Model and 2D Patterns." *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2014.

[21] Perš, Janez, and Stanislav Kovacic. "Nonparametric, model-based radial lens distortion correction using tilted camera assumption." *Proceedings of the Computer Vision Winter Workshop*. Vol. 1. 2002.

[22] SciPy. Program documentation. Vers. 0.16.1. Web. 23 Dec. 2015. <http://docs.scipy.org/doc/scipy-0.16.1/reference/>.

[23] Stehle, Thomas, et al. "Camera calibration for fish-eye lenses in endoscopy with an application to 3D reconstruction." *Biomedical Imaging: From Nano to Macro, 2007. ISBI 2007. 4th IEEE International Symposium on*. IEEE, 2007.

[24] Stevenson, Daniel E., and Margaret M. Fleck. "Nonparametric correction of distortion." *Applications of Computer Vision, 1996. WACV'96., Proceedings 3rd IEEE Workshop on*. IEEE, 1996.

[25] Tang, Zhongwei, et al. "Self-consistency and universality of camera lens distortion models." (2012).

[26] Tavakoli, Hamed Rezazadegan, and Hamid Reza Pourreza. "Automated center of radial distortion estimation, using active targets." *Computer Vision–ACCV 2009* (2010): 325-334.

[27] Tsai, Roger Y. "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses." *Robotics and Automation, IEEE Journal of* 3.4 (1987): 323-344.

[28] Wei, Guang, et al. "A Two-Dimensional Sub-Pixel Edge Detection Algorithm Based on Hyperbolic Tangent." *Information Engineering and Computer Science (ICIECS), 2010 2nd International Conference on*. IEEE, 2010.

[29] Weng, Juyang, Paul Cohen, and Marc Herniou. "Camera calibration with distortion models and accuracy evaluation." *IEEE Transactions on Pattern Analysis & Machine Intelligence* 10 (1992): 965-980.

[30] Zhang, Zhengyou. "Flexible camera calibration by viewing a plane from unknown orientations." *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Vol. 1. IEEE, 1999.

Appendix A — nonparametric package usage

The package was verified to be working with Python version 2.7.10 for Microsoft Windows. While not required, 64-bit release is recommended, because of high memory consumption during feature detection stage of the nonparametric algorithm. Additionally, following Python modules are required:

- OpenCv, version 2.4.11,
- NumPy, version 1.9.3,
- SciPy, version 0.16.1,
- matplotlib, version 1.5.0,
- PyQt4, version 4.11.4.

The `nonparametric` package should be placed in Python search path (`C:\Python27\Lib\site-packages` by default), so that it can be imported with statement: `import nonparametric`.

For parametric correction, one instance of `Radial` class is created. Basic constructor parameters are: `pattern_size`, `input` and `output`. Despite its name, `Radial` class corrects both radial and tangential distortions.

Afterwards the object should be called. Call arguments control RMSE evaluation. The call performs full calibration, image rectification and RMSE evaluation (only for rectified photographs).

Nonparametric correction differs programmatically from the parametric one. An instance of `NonparametricCorrection` class should be created for every photograph pair. This way there is more control over each pair options. Usually at least `scene1`, `scene2`, `output_folder` and `pattern_config` constructor parameters should be specified. It is essential that every object has a unique `output_folder` parameter passed to the constructor.

Then constructed objects should be called, similarly to the `Radial` objects. Call arguments likewise regulate RMSE evaluation, but also some optional correction steps. It should be stressed that in case any checkerboard control photographs are used, `rms_pattern_size` argument should be specified. The call performs full calibration, image rectification and RMSE assessment, including original images if it was not conducted yet. The RMSE results are stored in static class members. They can be cleared afterwards, using static methods: `reset()` or `report_and_reset()`.

All arguments and internal methods are described in documentation located in docstrings inside the source files.

A simple case of package usage is presented in Listing 1. Default input and output directories are assumed. Two predefined patterns are supplied to the nonparametric algorithm.

Listing 1.: Illustrative usage of the package

```
import nonparametric
import glob

radial = nonparametric.Radial( pattern_size=(10,7) )
radial()

names = glob.glob('images/pattern/*.JPG')
pattern_configs = [1, 0]
for i in range(len(names)/2):
      output_folder = 'pair_{0:d}'.format(i)
      correction = nonparametric.NonparametricCorrection(
            scene1 = names[2*i],
            scene2 = names[2*i+1],
            output_folder = output_folder,
            pattern_config = pattern_configs[i]
      )
      correction()

nonparametric.NonparametricCorrection.report_and_reset()
```

Appendix B — CD-ROM contents

CD-ROM supplied with the paper contains:

- `thesis.pdf` — digital version of this paper,
- `data/` — input images, correction results and scripts; structure of this folder is elaborated in the `Readme.txt` file inside the directory,
- `nonparametric/` — source code of the developed Python package along with three sample pattern images; authors of images used for pattern creation are listed in the `__init__.py` file,
- `win_x86-64/` — installation files. Installation instructions are included in the `Readme.txt` file.