Image and Vision Computing 31 (2013) 969–981

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Corisco: Robust edgel-based orientation estimation for generic camera models ☆

Nicolau Leal Werneck *, Anna Helena Reali Costa

Intelligent Techniques Laboratory (LTI), CEP: 05508-900, University of São Paulo (USP), Av. Prof. Luciano Gualberto trav.3, n.158, São Paulo, SP, Brazil

ARTICLE INFO

Article history:
Received 17 October 2012
Received in revised form 21 September 2013
Accepted 10 October 2013

Keywords:
Grid mask
Edgels
Orientation estimation
Omnidirectional vision
Monocular vision
Quaternions

ABSTRACT

The estimation of camera orientation from image lines using the anthropic environment restriction is a well-known problem, but traditional methods to solve it depend on line extraction, a relatively complex procedure that is also incompatible with distorted images. We propose *Corisco*, a monocular orientation estimation method based on edgels instead of lines. Edgels are points sampled from image edges with their tangential directions, extracted in *Corisco* using a grid mask. The estimation aligns the measured edgel directions with the predicted directions calculated from the orientation, using a known camera model. *Corisco* uses the M-estimation technique to define an objective function that is optimized by two algorithms in sequence: RANSAC, which gives robustness and flexibility to *Corisco*, and FilterSQP, which performs a continuous optimization to refine the initial estimate, using closed formulas for the function derivatives. *Corisco* is the first edgel-based method able to analyze images with any camera model, and it also allows for a compromise between speed and accuracy, so that its performance can be tuned according to the application requirements. Our experiments demonstrate the effectiveness of *Corisco* with various camera models, and its performance surpasses similar edgel-based methods. The accuracy displayed a mean error below 2° for execution times above 8 s in a conventional computer, and above 3° for less than 2 s.

1. Introduction

This article describes *Corisco*, a monocular orientation estimation method. It explores the restriction of an *anthropic environment*, also called a *Manhattan world*, which means the environment must contain lines that are parallel to the axes of a *natural reference frame*, defined by three mutually orthogonal directions. The orientation estimated by *Corisco* is the three-dimensional rotation between this natural reference frame and the camera reference frame. This method can be used inside larger multiple-view systems for environment reconstruction and camera tracking [1,2], to provide initial estimates from individual images while also enforcing the anthropic environment restriction. *Corisco* can also be used in lightweight mobile and portable applications, for example, providing a heading reading for a mobile robot to drive along a corridor or street, thus dismissing a simultaneous position estimation or environment reconstruction.

The problem of estimating orientation from the lines of a single image using the anthropic environment restriction is well known [3–7], but the dependence of the traditional methods on the extraction of lines from the images creates a number of limitations. Line extraction cannot be applied to images obtained with a camera model other than perspective projection, because in that case the environment lines are projected into curves on the images. Line extraction also constitutes a relatively difficult problem to be solved, involving the estimation of the parameters of an unknown number of entities, while orientation estimation by itself depends only on the estimation of the few parameters that model a three-dimensional rotation. It is possible to deal with image distortions by fitting curves to the image [8–10], but this only solves the problem of analyzing distorted images while further increasing the difficulty of the extraction of those geometrical entities.

These limitations can be overcome by using edgels as the fundamental geometrical entity of the analysis. Edgels are points that are part of an image edge, either straight or curved, associated to the information of the tangential direction of the edge at that point. Edgels are easier to handle and more flexible than lines, and their use does not require the estimation of multiple intermediary numeric parameters. Edgels are closely related to the image gradient, and their extraction is based on edge detection techniques. Edgel-based orientation estimation methods [11–16] work by minimizing the angular errors between the measured directions of the extracted edgels and the directions predicted as a function of the orientation parameters. The method proposed here, *Corisco*, follows this same principle, but introduces many improvements.

The objective of this research is to explore the potential of the edgel-based analysis to overcome the limitations of line-based approaches, thus creating a method that offers a good performance while being

☆ This paper has been recommended for acceptance by Cornelia M Fermuller.

* Corresponding author. Tel.: +55 11 30915397; fax: +55 11 30915294.
E-mail addresses: nwerneck@gmail.com (N.L. Werneck), anna.reali@usp.br (A.H.R. Costa).
URL: http://nwerneck.sdf.org (N.L. Werneck).

0262-8856/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.imavis.2013.10.004

more simple, flexible, and robust. The main contributions of our research are in the following features, which make *Corisco* attractive for practical implementations:

- *Corisco* directly analyzes images created with any camera model, including radial distortions and omnidirectional models.
- *Corisco* allows for a compromise between speed and accuracy, so its performance can be tuned according to the application requirements.
- *Corisco* does not depend on strong restrictions of the solution, such as that the orientation should be approximately upright.
- *Corisco* uses a direct implementation of a constrained non-linear continuous optimization algorithm, with simple expressions for the objective function and its derivatives.

The possibility of working with different camera models using edgels was recognized in the past [14] but, to our knowledge, no previous research has effectively demonstrated this. Edgels have been used to estimate the radial distortion coefficient of a camera model, however using the plumb-line constraint. That research by Rosten [17] demonstrates the advantages of edgel-based techniques to deal with image distortions. *Corisco* differs from that proposal by estimating orientation and also working on the original space of the acquired image, which is more desirable because it usually improves the estimation accuracy [18], and also avoids the limitations of the perspective projection.

Fig. 1 illustrates the application of *Corisco* to estimate the orientation of an input image. Other necessary inputs are the parameters of the camera model, such as the focal length and projection center in the case of the perspective projection, and a few control parameters for the *Corisco* process. The output is the camera orientation relative to the natural reference frame, given by a set of parameters Ψ that model a three-dimensional rotation. *Corisco* works with quaternions, therefore the Ψ returned is a vector of four values restricted to $|\Psi| = 1$. These values can be used to calculate a three-dimensional rotation matrix if necessary. The bottom left part of Fig. 1 displays the edgels extracted from the input image. These geometrical entities are obtained in the first analysis stage, and the input image is not used again after that. The bottom right part of Fig. 1 displays the input image overlaid with a set of triplets of line segments. These line segments indicate the three possible edgel directions predicted at each different point from the correct orientation, also using the camera model parameters. The estimation procedure seeks an orientation that causes the predicted directions to be aligned to the edgels.

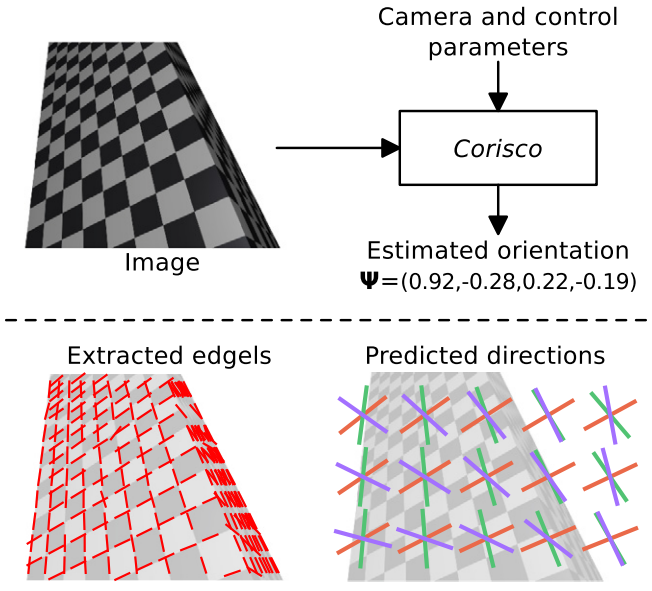

Fig. 1. Overview of how *Corisco* works. The inputs are the image to be analyzed (top left), and a set of parameters that define the camera model and control the process. *Corisco* (top right) outputs the numeric parameters Ψ of the estimated orientation of the camera relative to the natural reference frame. The image analysis performed by *Corisco* produces a set of edgels (bottom left) that are located over the image edges, and the estimation procedure is based on a calculation of predicted directions for these edgels. The directions predicted from the correct orientation are aligned to the image edges (bottom right).

Section 2 reviews existing edgel-based orientation estimation methods, and discusses the estimation techniques on which they are based. Section 3 describes *Corisco*. Section 4 discusses the experiments conducted with *Corisco*, and Section 5 contains conclusions about the research and some ideas for future developments.

2. Edgel-based orientation estimation

This section reviews existing methods for edgel-based orientation estimation [12–16]. All of these methods, including *Corisco*, share a common process structure with the following processing blocks:

- *Data extraction and sub-sampling* — This is the first analysis stage where a number of geometrical features are extracted from the image. The resulting dataset is the input to the second stage of the process, where the actual estimation is performed by an optimization procedure. This data is often sub-sampled in order to accelerate the process.
- *Objective function calculation procedure* — This is a procedure that depends on the extracted data, and also receives as argument the parameters of a hypothetical orientation. The output is the value of an objective function. During estimation, this calculation is performed many times for different orientation parameters.
- *Optimization procedure* — The optimization procedure constitutes the second processing stage. It interacts with the previous procedure, feeding hypothetical orientations and seeking to optimize its output value.

The following subsections discuss each of these blocks, but before such discussion, some definitions are necessary.

2.1. Definitions

An *edge* is an entity with length but no width, such as a straight line or a curve. An *edgel* is a point sampled from an edge, associated with the tangential direction of the edge at that point. *Edge detection* is the binary classification of an image point as being part of an edge or not. The recognized points are called *edge points*. Edge detection should not be confused with *edge extraction*, which is usually based on an edge detection followed by an algorithm such as flood-fill [6,7,10,2], RANSAC [9,1,19], J-linkage [20], RUDR [21] or variations of the Hough transform [22,23]. These algorithms fit geometrical models to clustered edge points.

Some authors use the name *edgel* for what we call *edge points*, but in the present article an edgel requires an associated direction. It is very easy to create an *edgel extraction* procedure based on an underlying edge detection. It should be noticed that none of the alternative edgel-based orientation estimation methods referred here used the term *edgel* to describe their observations; therefore, our discussion involves some reinterpretation of them.

One example of edgel extraction that is not directly based on edge detection is the procedure employed by Eade and Drummond [24]. These authors use the term *edgel* in their article, but they call *edgelet* the geometrical entity that is being tracked in space. We do not make this distinction, and we simply call *edgel* any point with associated direction, even in three-dimensions.

2.2. Data extraction and sub-sampling

All edgel-based orientation estimation methods start with calculating the image gradient. The edgels in early methods [12–14] were directly obtained from the gradient at the locations of the image pixels. The method proposed by Coughlan and Yuille [11,12] did not

employ any kind of sub-sampling, but later authors [13,14] used a random sub-sampling of the pixels in order to reduce the volume of data, and also to smooth the objective function. Schindler and Dellaert [14] also used a thresholding based on the magnitude of the gradient, in a first attempt to apply edge detection to this problem.

Dennis et al. [15] applied edge detection using the relatively sophisticated method of Elder and Zucker [25], but no further sub-sampling was performed. A subpixel refinement of the location of the edge points was also employed in that method. This actually no longer constitutes just detection, but an extraction procedure that takes in consideration the position of the detected point and the local direction of the gradient.

Werneck and Costa [16] introduced the sub-sampling by grid masking, but did not employ edge detection. Grid masking is deterministic, and similar to the tiling used by Deutscher et al. [13], resulting in a more homogeneous sampling than random techniques. Tiling is also employed by Eade and Drummond [24]. One important difference between grid masking and the procedure from Eade and Drummond [24] is that in that case edges may be ignored when they are located near the corners of a tile, so grid masking results in a more regular and reliable sampling.

The sub-sampling by image lines and columns resulting from grid masking is more natural than just selecting random image pixels because edges have no area and, therefore, have a peculiar relationship with the image resolution. The fraction of image pixels that are edge points increases with the inverse of the resolution, and the grid mask deals directly with this phenomenon. This form of sub-sampling can also be easily integrated with edge detection, as adopted in *Corisco* and described in Section 3.

2.3. Objective function

All the previously proposed methods [12–16] are based on probabilistic techniques for parameter estimation. That means that an observation model must be defined, starting with the definition of *probability density functions* (PDF) for the observed quantities. The estimation is carried out by the optimization of an objective function that is defined from these individual functions, usually by defining a *likelihood function* for the set of observations. One important point in this problem is that the observations can be produced by lines in each of the three possible directions in the environment, defining three different classes. Because the classes of observations are not known, it is necessary to use an estimation technique such as the *maximum a posteriori* (MAP) [12,13] or the Expectation-Maximization (EM) [14,15].

In most of the previous methods the observed quantities were both the magnitude $|\mathbf{g}|$ and direction $\angle\,\mathbf{g}$ of the image gradient at a set of points over the image. Early proposals [12,13] followed the idea that the edge detection should be integrated with the estimation process, as reflected in their objective functions. In more recent proposals [14,15] the analysis of $|\mathbf{g}|$ started sooner and separately, and the objective function became more clearly dependent on $\angle\,\mathbf{g}$ only.

Different distributions have been used in these methods to model the observation of the gradient or edgel directions, including the boxcar function [12], triangular [13,16], Gaussian [14] and generalized Laplacian [15]. In the case of the Gaussian distribution with the EM technique proposed by Schindler and Dellaert [14], the resulting objective function was

$$F(\Psi) = \sum_{n}^{N} c_n^x \left(\angle\mathbf{g}_n - h^x(\mathbf{p}_n, \Psi)\right)^2 + c_n^y \left(\angle\mathbf{g}_n - h^y(\mathbf{p}_n, \Psi)\right)^2 + c_n^z \left(\angle\mathbf{g}_n - h^z(\mathbf{p}_n, \Psi)\right)^2. \quad (1)$$

the h^k functions calculate the predicted direction of an edgel from class k at position $\mathbf{p}_n$ for the orientation Ψ. The c_n^k coefficients are the probabilities of each observation belonging to class $k = x$, y or z. They are calculated in each *E-step* of the estimation procedure, also taking into account the probabilities of the observation not being an edge, and of being a non-aligned edge — these re-calculations of c_n^k from the EM technique make the process a little more complex than the outline at the beginning of this section. In Eq. (1) the use of the Gaussian model and the application of the logarithm function and the Jensen inequality result in an expression that is a simple weighted sum of squared differences. In the proposal of Denis et al. [15] the different PDF does not result in such a simple error function, and their calculation of the c_n^k coefficients is also different because all the observations are assumed to be edge points.

2.4. Optimization

The optimization method employed by Coughlan and Yuille [11,12] was a simple and unsophisticated sweep through the parameter space. The orientation was parameterized using Euler angles, and the search was performed initially around the vertical axis, assuming an approximately upright camera. To speed the process up, a coarse-to-fine strategy was also employed. Deutscher et al. [13], on the other hand, employed stochastic search. The focal length was also estimated by that method, exploring the flexibility of this optimization technique.

Schindler and Dellaert [14] employed a continuous optimization algorithm, the Levenberg–Marquardt algorithm, exploring their quadratic objective function. For the first derivatives these authors employed automatic differentiation. In the case of Denis et al. [15] the objective function was not quadratic, so the more general BFGS optimization algorithm was employed, with numeric differentiation for the derivatives.

The latter two methods also rely on an initial brute force sweeping [12]. After the initial trials are tested the best hypotheses are refined by continuous optimization, and the best result selected. Schindler and Dellaert [14] note explicitly that their optimization presented great sensitivity to the initial estimates, and that a better and more general initialization procedure would be desirable. The method proposed by Werneck and Costa [16] also employs multiple initializations, and is based on the Powell optimization algorithm.

2.5. Discussion

This section reviewed existing edgel-based orientation estimation methods. The approach initially showed to be effective [12], but it was later shown that better estimation and optimization techniques could be applied to create more efficient methods [13,14]. While it was initially thought that analyzing the image gradient directly could be beneficial, the use of edge detection showed to enhance the performance with no detriment to accuracy [15].

The work by Denis et al. [15] guided the research on the problem to a new level by making their experimental data available. Some of the practical issues that were found in the previous studies to require further investigation are:

- How to employ sub-sampling with controlled effects on speed and accuracy.
- How to produce initial estimates in a more flexible and efficient way.
- How to deal with camera models other than the perspective projection.

These issues were targeted in the development of *Corisco*, as shown in Section 3.

3. Description of *Corisco*

Fig. 2 displays a block diagram of the process that constitutes *Corisco*, from the initial image analysis to the output of the estimated orientation parameters. The steps are:

1. The image is analyzed in the *Edgel extractor* block in order to produce a list of edgels.

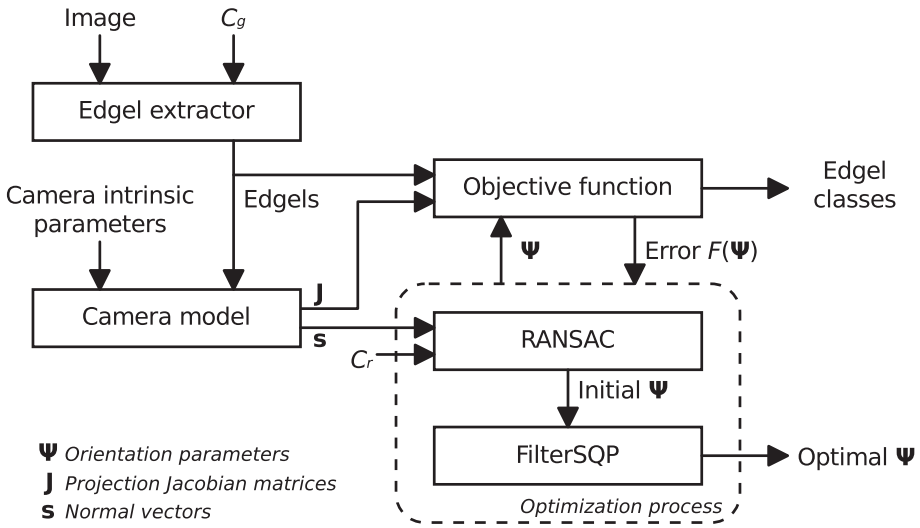

Fig. 2. Block diagram of *Corisco*.

2. A separate *Camera model* block produces more data complementing the edgels. The resulting dataset is the input to the second stage of the process.
3. This second stage is an optimization procedure that minimizes an objective function, calculated by the *Objective function* block. This block is executed once for each iteration of the optimization algorithms.

The optimization procedure has two steps, with a different iterative optimization algorithm in each of them. The first step produces an initial estimate used to initialize the second step. The *Objective function* block can also optionally return an estimated class k for each edgel, $k = x$, y or z.

This process is similar to the outline from Section 2, the largest difference being the separate *Camera model* block that performs an intermediary processing of the extracted data. In the other methods the camera model is often part of the calculation of the objective function, and this separation is one distinctive characteristic of *Corisco*. The remainder of this section was organized according to the blocks from Fig. 2.

3.1. Edgel extractor

The *Edgel extractor* block from *Corisco* is based on a well-known edge detection technique [26] coupled to the use of a grid mask [16]. The inputs to this block are the input image and the grid size C_g given in pixels. The output is a list of N edgels, each one defined by a pair of image coordinates $\mathbf{p}_n$ and a normalized vector $\mathbf{u}_n$ orthogonal to the direction of the line or curve over which the edgel was found, and thus parallel to the image gradient over $\mathbf{p}_n$.

The procedure starts with the calculation of the image gradient. A special direction normalization procedure is necessary during the analysis to handle color images. *Corisco* was implemented with the 5×5 maximally-consistent gradient filter developed by Ando [27]. An initial Gaussian smoothing is optional.

The analysis of one horizontal grid line is performed by the following steps:

1. For each image pixel, fetch its corresponding gradient vector in each channel. If the horizontal direction of the vector is negative, multiply the vector by -1. In the case of color images the vectors for this pixel are then added together. *Gradient* now refers to this vector.
2. Perform an edge detection, leaving only the pixels that are local maxima of gradient magnitude, and with a magnitude above a threshold value.
3. For each of the detected edge points, perform a new test of the gradient direction. If the angular distance between the gradient and the horizontal grid line is larger than 45°, the point is discarded.
4. For the remaining edge points, perform an edge extraction in the vicinity of the point, estimating the exact position where the detected edge crosses the grid line, with sub-pixel accuracy. This is similar to a usual position refinement after edge detection, except the edge is sought over the horizontal grid line instead of the gradient direction.
5. Once this more accurate position is found for each detected point, a new edgel is instantiated at that position, and with the same direction from the gradient.

The analysis of a vertical grid column works in the same way, as if the image had been transposed. The pixels that are not part of the grid lines or columns are simply ignored, acting as masking. As the edgels are extracted from the grid sweeps, they are simply accumulated as a set in a single list. This list is returned at the end of the process.

3.2. Camera model

The *Camera model* block from Fig. 2 receives as input the *intrinsic parameters* that define the camera model, and also the list of edgels from the *Edgel extractor* block. The outputs from this block are two lists containing for each edgel a corresponding Jacobian matrix $\mathbf{J}_n$ and a three-dimensional normal direction $\mathbf{s}_n$. This output along with the extracted edgels makes up all data necessary for the remainder of the process. We here discuss the mathematical characteristics of the camera model and how to calculate the $\mathbf{J}_n$ and $\mathbf{s}_n$. Their use in further calculations is detailed in Sections 3.3 and 3.4.

We denote the n-dimensional real coordinate space $\mathbf{R}^n$, and $\mathbf{S}^n$ the space of directions in $\mathbf{R}^{n+1}$, that corresponds to an n-sphere. A camera model is an injective mapping $\mathbf{S}^2 \to \mathbf{R}^2$, from directions in the three-dimensional space around the focal point of the camera to the plane of the image produced by the camera. The mapping from the image plane to three-dimensional points can be defined in different ways, as the norm of the output vector is not relevant. This mapping from the plane to points in space is useful when implementing the method, but it should be clear that the actual mapping is between points in a plane and directions in the three-dimensional space.

The most fundamental camera model is based on *perspective projection*. If we denote the three-dimensional vector in the camera

reference frame **q**, and its corresponding image point **p**, this projection is defined by the equations

$$p^x = (q^x/q^z)f + c^x \qquad p^y = (q^y/q^z)f + c^y. \tag{2}$$

this projection has two parameters that must be set, the focal length f and the projection center **c**. One example of an inverse mapping for this projection is

$$q^x = p^x - c^x \qquad q^y = p^y - c^y \qquad q^z = f. \tag{3}$$

There are other camera models outside the perspective projection. The Harris model [28] is one alternative to handle the radial distortion caused by some lenses. Many imaging systems also produce images with strong distortions, such as fisheye lenses, catadioptric systems [8,10,19] and camera rosettes that produce images with the equirectangular projection [29].

Table 1 displays the equations from four different camera models, all of which have been tested with *Corisco*. For a certain model choice and its parameters, the model calculates image point **p** that is the projection of a given three-dimensional point **q** in the camera reference frame. If point **q** is part of a line in space, this line will be projected on a curve over the image — a straight line in the case of the perspective projection — and point **p** will be part of this projected line or curve. The tangential direction of this curve is the predicted direction **v** for an edgel at **p**.

The calculation of the predicted direction of an edgel on the image depends on the camera model, the position of **q** in space — calculated from **p** — and the direction **r** in the camera reference frame of this hypothetical line that **q** is part of. The tangential direction **v** of the curve projected on point **p** can be found by the formula

$$\mathbf{v} \propto \mathbf{J}\mathbf{r}, \tag{4}$$

where **J** is the Jacobian matrix of the mapping calculated for that specific **q**, defined by

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{p}^x}{\partial \mathbf{q}^x} & \frac{\partial \mathbf{p}^x}{\partial \mathbf{q}^y} & \frac{\partial \mathbf{p}^x}{\partial \mathbf{q}^z} \\ \frac{\partial \mathbf{p}^y}{\partial \mathbf{q}^x} & \frac{\partial \mathbf{p}^y}{\partial \mathbf{q}^y} & \frac{\partial \mathbf{p}^y}{\partial \mathbf{q}^z} \end{bmatrix}. \tag{5}$$

the symbol ∝ in Eq. (4) indicates that the calculation yields only a vector in the desired direction, but with a generic norm. The *Camera model* block calculates all of the Jacobian matrices $\mathbf{J}_n$ for each extracted edgel in order to permit future calculations of Eq. (4).

Jacobian matrix **J** is also necessary to calculate the three-dimensional normal direction **s** from an edgel. This vector defines a plane that crosses the camera focal point and also the line that originated the corresponding edgel, and it can be calculated by

$$\mathbf{s}_n = u_n^x \mathbf{J}_n^x + u_n^y \mathbf{J}_n^y, \tag{6}$$

Table 1
Some examples of camera model equations.

Perspective	Equirectangular (lat-lon)
$p^x = (q^x/q^z)f + c^x$	$p^x = f\tan^{-1}(q^z, q^x)$
$p^y = (q^y/q^z)f + c^y$	$p^y = f\sin^{-1}(q^y/\lvert\mathbf{q}\rvert)$
Harris (radial distortion)	Polar equidistant (fisheye)
$g(\chi) = \frac{1}{\sqrt{1-2\kappa\chi^2}}$	$\varphi = \cos^{-1}(q^z/\lvert\mathbf{q}\rvert)$
$p^x = p'^x g(\lvert\mathbf{p}'\rvert) + c^x$	$p^x = \varphi \frac{q^x}{\sqrt{q^{x2}+q^{y2}}} f + c^x$
$p^y = p'^y g(\lvert\mathbf{p}'\rvert) + c^y$	$p^y = \varphi \frac{q^y}{\sqrt{q^{x2}+q^{y2}}} f + c^y$

where $\mathbf{J}_n^x$ and $\mathbf{J}_n^y$ are each line of the matrix $\mathbf{J}_n$, and **u** is the orthogonal direction of the edgel, defined by

$$u_n^x = v_n^y \qquad u_n^y = -v_n^x. \tag{7}$$

the direction $\mathbf{r}_k$ that defines class k is necessarily orthogonal to any **s** calculated from an edgel of that class. Therefore $\mathbf{r}_k$ can be found from two different **s** vectors by a cross product. These normal vectors are the basis of some orientation estimation methods [30], but they tend to be less accurate [7,15].

Once the Jacobian matrices and normal vectors are created in the *Camera model* block, the optimization procedure starts. Both the camera model and the input image are no longer needed for the remainder of the process, only the list of N edgels described by their positions $\mathbf{p}_n$ and orthogonal directions $\mathbf{u}_n$, provided by the *Edgel extractor* block, and the Jacobian matrices $\mathbf{J}_n$. The normal vectors $\mathbf{s}_n$ can be left to be calculated only when necessary. The use of **J** to calculate an edgel direction **v** as a function of the orientation Ψ will be detailed in Section 3.3, and the use of **s** to create hypothetical orientations in Section 3.4.

3.3. Objective function

The expression calculated by the *Objective function* block from Fig. 2 is the most distinctive characteristic of *Corisco*. The edgel extraction and the optimization procedure can be easily modified, but the objective function is what defines the solution, and it also guides the choice of the optimization algorithm. The estimation made in *Corisco* is based on a residue between the predicted edgel directions $\mathbf{v}_{nk}$ and the measured directions $\mathbf{v}_n$, or more specifically the *orthogonal* measured direction $\mathbf{u}_n$, resulting in

$$\mathbf{u}_n \mathbf{v}_{nk}. \tag{8}$$

vectors $\mathbf{u}_n$ come from the *Edgel extractor* block, and the calculation of $\mathbf{v}_{nk}$ will be explained next. Once these residues are calculated a total error function can be calculated, which will be explained later in this section.

The calculation of $\mathbf{v}_{nk}$ depends on the Jacobian matrices $\mathbf{J}_n$ calculated in the *Camera model* block, and on the line directions $\mathbf{r}_k$ that are calculated from a given orientation Ψ. *Corisco* uses quaternions to represent the orientation Ψ; therefore, we can write it as a four-dimensional vector $\Psi = (a,b,c,d)$. The directions of the three classes $\mathbf{r}_x$, $\mathbf{r}_y$ and $\mathbf{r}_z$ can now be obtained from the lines of a rotation matrix calculated from the components of Ψ

$$\begin{bmatrix} \mathbf{r}_x \\ \mathbf{r}_y \\ \mathbf{r}_z \end{bmatrix} = \mathbf{R}(\Psi). \tag{9}$$

this calculation is performed by the following quadratic expressions

$$\begin{aligned} \mathbf{r}_x &= \left(a^2 + b^2 - c^2 - d^2,\ 2bc + 2ad,\ 2bd - 2ac\right) \\ \mathbf{r}_y &= \left(2bc - 2ad,\ a^2 - b^2 + c^2 - d^2,\ 2cd + 2ab\right) \\ \mathbf{r}_z &= \left(2bd + 2ac,\ 2cd - 2ab,\ a^2 - b^2 - c^2 + d^2\right). \end{aligned} \tag{10}$$

a normalized quaternion would be necessary to produce normalized **r** vectors, but this is not necessary in *Corisco* because of the final normalization that happens when $\mathbf{v}_{nk}$ is calculated, as shown ahead. The objective function is therefore naturally defined for any quaternion in the four-dimensional space.

Fig. 3 illustrates an edgel that is part of the projection of a line over an image. The drawing shows the $\mathbf{r}_x$ and $\mathbf{r}_y$ directions from the natural reference frame, and the camera reference frame rotated relative to it. The focal point of the camera is the origin of the camera reference

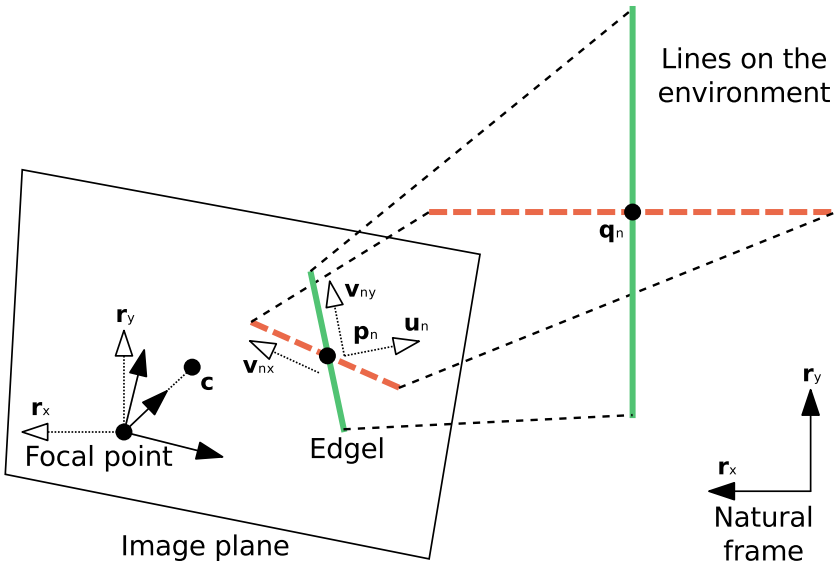

Fig. 3. Projection of an edgel on the image plane.

frame. The camera uses the perspective projection, and the drawing also shows its image plane and projection center $\mathbf{c}$. The environment contains a solid line from class $k = y$. This line contains a point $\mathbf{q}_n$, which is projected on the image at point $\mathbf{p}_n$, defining an edgel with direction $\mathbf{v}_{ny}$, and orthogonal direction $\mathbf{u}_n$.

Although both vectors refer to the same entity, we always use $\mathbf{u}_n$ to denote the *measured* orthogonal direction of an edgel, and $\mathbf{v}_{nk}$ to denote a *predicted* direction, calculated during the estimation process. An edgel direction $\mathbf{v}_{ny}$ can be calculated from $\mathbf{r}_y$ and the corresponding Jacobian $\mathbf{J}_n$ using Eq. (4). If the calculation assumed class $k = x$, it would produce direction $\mathbf{v}_{nx}$, which is not orthogonal to $\mathbf{u}_n$, and it would mean that the edgel was produced from the hypothetical dashed line in the environment, which also crosses $\mathbf{q}_n$. It is worth noticing that although $\mathbf{r}_y$ and $\mathbf{r}_x$ are necessarily orthogonal, $\mathbf{v}_{ny}$ and $\mathbf{v}_{nx}$ are not.

Vector $\mathbf{v}_{nk}$ is therefore a function of the edgel position $\mathbf{p}_n$, the class k and the orientation Ψ that defines directions $\mathbf{r}_k$. If we define the auxiliary vector

$$\mathbf{w}_{nk} = \mathbf{J}_n \mathbf{r}_k, \tag{11}$$

we can then define

$$\mathbf{v}_{nk} = \frac{\mathbf{w}_{nk}}{|\mathbf{w}_{nk}|}. \tag{12}$$

The Jacobian matrices used in Eq. (14) are the ones calculated in the *Camera model* block.

The objective function used in *Corisco* is defined from the $\mathbf{u}_n \mathbf{v}_{nk}$ residue using *M-estimation*. This *Robust Statistics* technique replaces in *Corisco* the MAP [14,13,16] or EM [14,15] techniques that were used in the previous methods. The result is the quite simple objective function expression

$$F(\Psi) = \sum_n \min_k \rho(\mathbf{u}_n \mathbf{v}_{nk}), \tag{15}$$

where $\min_k$ is a function that takes the smallest calculated value found among classes k. The function ρ is an error function that is chosen during the design of the objective function.

In the estimation techniques discussed in Section 2, the objective function is created in a bottom-up fashion, from the modeling of the observation errors to the definition of the actual objective function. In the case of the M-estimation technique the approach is top-down. The error function ρ in Eq. (16) is selected directly, although still in the context of a traditional probabilistic analysis. This analysis shows that this error function should be a *redescending* function, which is constant after a certain threshold. In the previous methods this characteristic is caused indirectly by the chosen models for the likelihood functions. In the case of the methods based on EM [14,15], this can also be seen as an effect of this estimation technique interpreted in the light of the IRLS algorithm [31], in which the use of the Gaussian model results in the minimization of a redescending error function. This characteristic is also justified because it makes the estimation robust against the great errors caused by incorrect class assignments.

In *Corisco* the Tukey bisquare function was selected for ρ. This is a smooth and continuous piecewise-polynomial redescending function that is widely used with M-estimation. It is defined by

$$\rho(x) = \begin{cases} 1 - \left[1 - (x/s)^2\right]^3 & \text{if} \quad |x| \le s \\ 1 & \text{if} \quad |x| > s \end{cases}. \tag{14}$$

the coefficient s controls the scale of the function, and it should be chosen to reflect the intensity of noise in the observations, or the variance from its PDF. The values used for s in *Corisco* during the tests were typically from 0.1 to 0.15. That means a range of directional errors from approximately 10° to 17°.

The use of the Tukey bisquare explains only part of Eq. (16). That expression also includes the choice of the smallest error from the three classes as the contribution of each observation to the summation. If the three error values were just added together the result would be similar to the application of the MAP technique [12,16]. In that case, the observations would not be classified, and this tends to reduce the accuracy of the results. The EM technique, on the other hand, causes a classification of the observations near the solution, and this tends to benefit the accuracy of the estimation.

In *Corisco*, selecting the smallest error for each observation is proposed as a means to perform this classification, directly implementing another effect of the EM technique. This is not a common procedure in the application of M-estimation, but it can be justified as approximating what happens in the EM technique when the c_n^k weight for one of the classes approaches 1 and the others 0. This objective function from *Corisco* can also be seen as a kind of Generalized EM technique, or GEM [31].

In short, the *Objective function* block in Fig. 2 receives as input the measured edgel directions $\mathbf{u}_n$ from the *Edgel extractor* block and the Jacobian matrices $\mathbf{J}_n$ from the *Camera model* block, and also a hypothetical orientation Ψ given by the optimization procedure. The orientation and the Jacobian matrices are used to calculate the predicted directions for each edgel and to produce the residues $\mathbf{u}_n\mathbf{v}_{nk}$, which are used to calculate a total error, or objective function $F(\Psi)$ using Eq. (16). This block can also optionally output a list of classes of observations, obtained as a result of the $\min_k$ operation in the calculations. This block also calculates the gradient and the Hessian matrix of the objective function $F(\Psi)$, relative to the four parameters of Ψ. These derivatives are transmitted along with the objective function value; thus, they would belong to the same arrow as $F(\Psi)$ in the diagram.

3.4. Optimization

This sub-section discusses the *RANSAC* and *FilterSQP* blocks from Fig. 2, which constitute the complete *Optimization process* used in *Corisco*. As the diagram illustrates, the *RANSAC* block does not depend on an initial orientation estimate, but it produces one to be used by the *FilterSQP* block. On the other hand, the *RANSAC* block needs to have access to a list of the normal vectors $\mathbf{s}_n$ calculated from the image edgels. Both optimization algorithms are iterative, and at each iteration a new hypothetical orientation Ψ is produced and its corresponding value $F(\Psi)$ is calculated by the *Objective function* block. These returned values are used by the optimization algorithms in their control logic that seeks to minimize this value. In the case of calculations requested by the *FilterSQP* block the derivatives of the objective function are also calculated and returned.

The RANSAC algorithm [32] is a Robust Statistics estimation technique employed by many different Computer Vision methods ([18], Section 4.7), and variations of it are also being developed seeking, for example, to connect it to probabilistic methods [33,34]. RANSAC is a random search algorithm that uses samples of small sets of observations to produce solution hypotheses to be tested. The use of RANSAC in orientation estimation methods is not new [1,5,9,10,24], but *Corisco* is the first edgel-based method to use it. The role of RANSAC in *Corisco* is to produce an initial estimate of the solution. Unlike similar methods, RANSAC does not make assumptions about the initial estimates of the solution. The exception is the method by Deutscher et al. [13], which uses a random search, but it is not guided by the data available.

The application of RANSAC in *Corisco* is very basic. It is an iterative algorithm, and each iteration is constituted by the following steps:

1. Pick the normals $\mathbf{s}_n$ from two edgels, and assume that they are from the same class $k = x$.
2. Calculate $\mathbf{r}_x$ using a cross product of these vectors.
3. Define a full orientation with the x axis aligned to this direction, and an arbitrary rotation around it.
4. Pick the normal $\mathbf{s}_n$ from a third edgel, assuming its class to be $k = y$.
5. Rotate the initial orientation around the x axis so its y axis is orthogonal to the third $\mathbf{s}_n$. This produces a hypothetical orientation Ψ.
6. Calculate the objective function value $F(\Psi)$ for this Ψ.
7. If $F(\Psi)$ is smaller than a previously stored value, store these new Ψ and $F(\Psi)$ as the new best estimate found.

These steps are carried out for a predetermined number C_r of iterations. This number, along with the grid size C_g, are the two main control parameters for *Corisco*, and both can be used to regulate the compromise between speed and accuracy.

Once all the C_r RANSAC iterations are over, its final estimate is used to initialize a non-linear continuous optimization algorithm, FilterSQP. This algorithm is not a conventional continuous optimization algorithm such as Levenberg–Marquadt [14] and BFGS [15], because FilterSQP performs a constrained optimization. Even though the *Corisco* objective function can be calculated for a generic Ψ anywhere in the four-dimensional quaternion space, the returned solution still needs to be normalized. The optimization problem solved in *Corisco* is therefore

$$\begin{aligned} \underset{\Psi}{\text{argmin}} \quad & F(\Psi) = \sum_n \min_k \rho(\mathbf{u}_n\mathbf{v}_{nk}) \\ \text{subject to} \quad & |\Psi| = 1. \end{aligned}$$

this objective function minimized is Eq. (16), and the constraint is that the quaternion Ψ must have unit norm, or $\sqrt{a^2+b^2+c^2+d^2} = 1$. Another way to write this constraint is to define the constraint function $G(\Psi) = \sqrt{a^2+b^2+c^2+d^2}$ and state that $G(\Psi) = 1$. FilterSQP [35] is an algorithm that can solve this kind of problem using the *sequential quadratic programming* (SQP) technique, avoiding the use of the so-called barrier functions.

This concludes the discussion of the optimization procedure used in *Corisco*. The Ψ found by FilterSQP is returned by *Corisco*. The extracted edgels can also be returned, optionally classified according to their direction in the environment at the optimal orientation Ψ found. The value of the objective function can also be returned, and it might be useful to larger systems built on top of *Corisco*.

4. Experiments

This section describes four experiments conducted with *Corisco*. The method was applied to estimate the orientation of images from different datasets, and compared with reference orientations obtained in different ways. In the first two experiments, *Corisco* performance was compared to those of similar alternative methods.

The implementation of *Corisco* used in these tests was based on the languages Python, Cython and C. Vector normalizations were performed with a fast but precise numeric procedure to calculate $x^{-1/2}$. The computers used in the experiments were Amazon Web Services `c1.xlarge` instances, with processor clocks of at least 2 GHz.

4.1. YorkUrbanDB dataset

The first dataset analyzed was the YorkUrbanDB [15]. It contains 102 images of indoor and outdoor anthropic environments. An ideal perspective projection camera model was assumed, and the intrinsic parameters of the camera and reference orientations Ψ_{ref} were provided by the dataset creators, obtained from image lines extracted automatically and labeled manually.

Corisco was executed to find the orientations from each image with different combinations of C_g and C_r, to study the compromise between speed and accuracy. The accuracy was evaluated by observing the distribution of estimation errors over all the 102 images. These errors were found by applying the reverse rotation matrix corresponding to the estimated orientation to the reference rotation matrix. The estimation error is the absolute value in degrees of the residual rotation obtained.

Fig. 4 shows one example of the application of *Corisco* to a picture from the YorkUrbanDB set. The left graph shows the extracted edgels as small line segments. The right graph shows the predicted edgel directions calculated at selected points over the image and for the orientation estimated by *Corisco*. It is possible to see that these predicted directions are aligned to the objects on the image.

Fig. 5 shows the performance of *Corisco* for the YorkUrbanDB image set. Each graph refers to a different number of iterations C_r, and the vertical axis of the graphs indicate the spacing of the grid lines and columns C_g varying from 1 to 128 pixels. The left part of each graph shows the error distributions, and the right part shows the mean processing time. The solid curve shows the total time, and the dashed curve shows the time spent only during the RANSAC step.

The error distributions in Fig. 5 are represented by box-plots. The left whisker indicates the smallest error, and the right whisker is the sixth

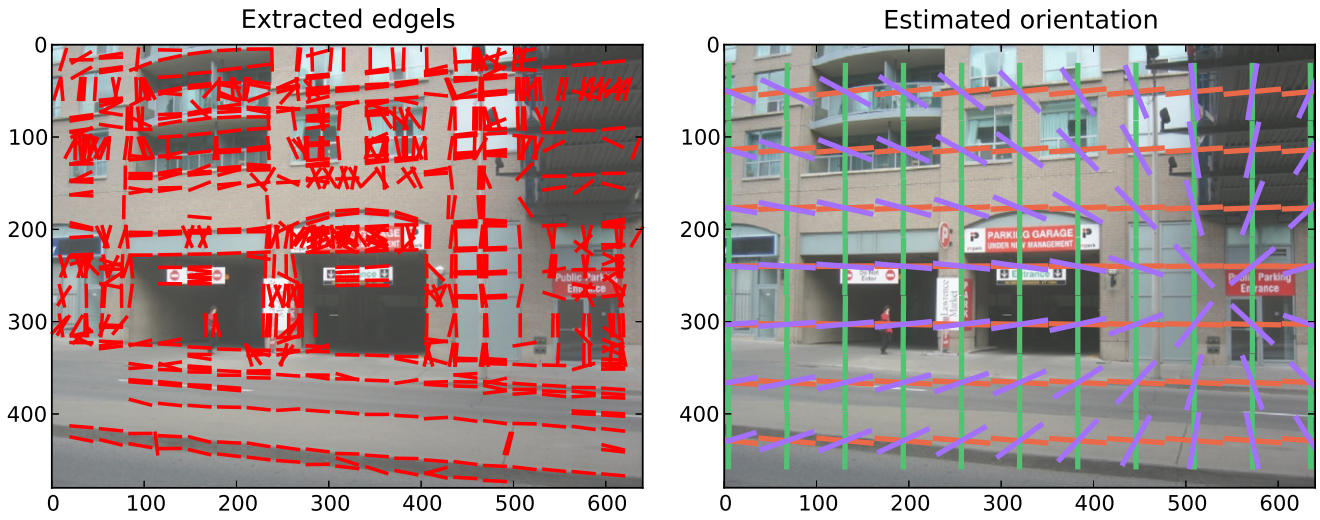

Fig. 4. Example of an image from the YorkUrbanDB set. The extracted edgels (left), and edgel directions predicted from the orientation estimated by *Corisco* (right) ($C_g = 32$ $C_r = 10,000$).

largest error from the 102 images. The box limits are given by the first and third quartile of the error distribution. The vertical lines that cross the boxes indicate the median distribution.

The graphs show that the error distribution increases with the grid size C_g, and that the reduction of C_r also increases errors. On the other hand, increasing C_g or decreasing C_r reduces the calculation time, as expected. It is possible to see how the time spent on the RANSAC algorithm is directly proportional to C_r, and for large values of C_r this also becomes the stage that takes the longest in the process.

Table 2 shows some numerical performance results. The first three lines from the table refer to three alternative methods tested by Denis et al. [15,36], and all these values were reported by these authors. Their experiments used a CPU with 1.83 GHz, while processors with 2.0 to 2.33 GHz were used with *Corisco*; hence, a factor of 0.8 could be applied to these time values to try to compensate for the difference. On the other hand, these authors did not clearly state whether the measured times refer just to the optimization procedure or whether they also include the time for the image processing; this is indicated by the question marks in the table.

The fourth line from Table 2 refers to the method by Tardif [7], based on line extraction and the J-linkage algorithm. We obtained these results executing the source code kindly provided by that author on a machine slightly slower than that used for *Corisco*.

The first three alternative methods in Table 2 display a compromise between time and accuracy, similar to that of *Corisco*. The J-linkage method exhibited good performance and accuracy when compared to the other three alternative methods. The table shows that *Corisco* was able to deliver both faster and more accurate results than the alternative methods. The combination of $C_r = 1000$ iterations and $C_g = 32$ pixels was especially interesting, and rivals all of the four alternative methods both in time and in accuracy.

4.2. ApaSt dataset

We developed the ApaSt image set during this research. It consists of 48 images from the same environment. Half of these images were obtained with a Nokia N8 smartphone, and the other half with a Sony α230 camera. All the images were scaled to a width of 1000 pixels. Half of the images captured had the camera in an approximately upright orientation, and the other half were rotated by 90° around the z direction. This increases the diversity of orientations in the set, and should also benefit the estimation of intrinsic parameters. Fig. 6 displays the application of *Corisco* to one image from the ApaSt set, similar to Fig. 4.

The reference orientations and the intrinsic parameters for the two cameras were found using *Bundler*, a point-based multi-view shape-from-motion method by Snavely et al. [37]. *Bundler* uses a simpler quadratic distortion model instead of the Harris model used in *Corisco*, but for such weak distortions they are very similar.

Bundler ignores the existence of a natural reference frame; thus, the reference orientations found are off the desired values by an unknown rotation that had to be estimated in order to evaluate *Corisco* and the alternative method. This rotation between the *Bundler* reference and the natural reference frame was estimated by solving a linear system in which the unknowns are the coefficients of the quaternion from the rotation. The right-side constants from the system are the orientations estimated with the analyzed methods, and the reference orientations found by *Bundler*, represented as quaternions, are used to assemble the matrix with the left-side coefficients from the system, following the rules of the Hamilton product. The rotation for the alternative method was estimated separately in order to ensure the best possible fit.

Similar to the previous experiment, *Corisco* was applied to this dataset with multiple combinations of its control parameters, and the alternative method by Tardif [7] was also run. The results from the alternative method and from some *Corisco* cases are shown in Table 3. The accuracies observed were generally better than those observed in the first experiment, for both *Corisco* and the alternative method. However, in this case it was not possible to observe a better overall *Corisco* performance than in the alternative method, although it was still possible to obtain better accuracy for a larger execution time.

It should be noticed that the alternative method was applied in this experiment disregarding the radial distortion model. An attempt was made to rectify the images before applying the alternative method, but the result was only a small increase in execution time (30 ms), and also a small reduction of accuracy. Because accuracy was not improved as expected, these results were not used. On the other hand, ignoring the radial distortion model in the application of *Corisco* did reduce accuracy as expected, and the results with the distortion model were kept.

Fig. 7 shows the *Corisco* performance in the ApaSt dataset for more parameter combinations. The results are similar to those from the YorkUrbanDB experiment. However, in the case of the ApaSt dataset, *Corisco* reached smaller errors in some tests. Some reasons for the better accuracy obtained both with *Corisco* and with the alternative method might be the size and quality of the images, or better accuracy from the reference orientations.

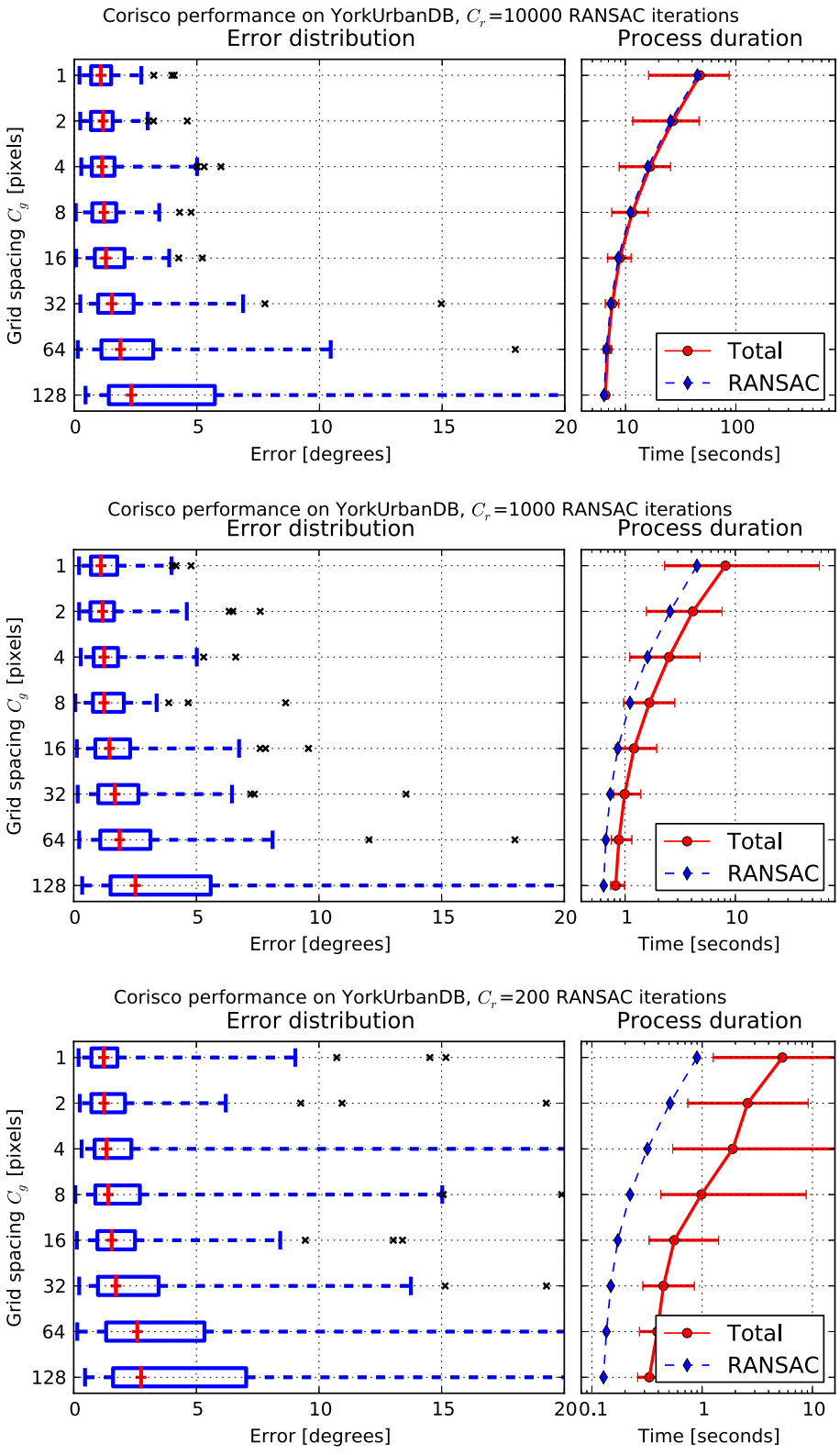

Fig. 5. *Corisco* performance for the YorkUrbanDB dataset. Each graph corresponds to a different number of C_r, and each line in the graph represents a different C_g. The error distributions are represented by box-plots. The compromises between speed and accuracy can be noticed by comparing the different curves.

4.3. StreetView dataset

The third experiment used a dataset provided by the Google company, collected for their StreetView project [38]. Only 251 images were used in this experiment. The images use the equirectangular projection, and were captured with a camera rosette installed on top of a car driving in an urban environment. The reference orientations in this dataset were obtained with sensors such as accelerometers and magnetometers. This dataset also ignores the existence of a natural reference frame; thus, a transformation had to be estimated as previously.

Fig. 8 shows an example of the analysis of one of these images using *Corisco*. Although the distortions can be quite intense, *Corisco* works

Table 2
Comparative table of *Corisco* performance in the YorkUrbanDB dataset for a few selected combinations of parameters, and the performance of some alternative methods.

Method	Time	Error				
	[s]	Mean	σ	1/4	Median	3/4
EM Newton	27+?	1.00°	1.15°	2.61°	4.10°	4.00°
MAP Quasi-Newton	6 + ?	4.00°	1.00°	1.32°	2.39°	4.07°
EM Quasi-Newton	1 + ?	9.00°	1.00°	4.04°	6.21°	10.33°
J-linkage	1.13	8.23°	13.76°	1.14°	2.36°	4.44°
Corisco $C_r = 10^4$ $C_g = 1$	47.20	1.51°	3.26°	0.69°	1.09°	1.51°
Corisco $C_r = 10^4$ $C_g = 4$	16.68	1.71°	3.35°	0.72°	1.14°	1.64°
Corisco $C_r = 10^4$ $C_g = 32$	7.57	2.43°	4.03°	0.97°	1.54°	2.42°
Corisco $C_r = 10^3$ $C_g = 1$	8.12	1.70°	3.22°	0.70°	1.11°	1.77°
Corisco $C_r = 10^3$ $C_g = 4$	2.50	2.02°	3.86°	0.81°	1.24°	1.80°
Corisco $C_r = 10^3$ $C_g = 32$	0.99	2.44°	3.54°	1.00°	1.68°	2.64°
Corisco $C_r = 200$ $C_g = 1$	5.34	2.08°	3.38°	0.72°	1.22°	1.78°
Corisco $C_r = 200$ $C_g = 4$	1.89	3.27°	6.38°	0.85°	1.34°	2.35°
Corisco $C_r = 200$ $C_g = 32$	0.45	3.29°	4.99°	0.99°	1.72°	3.46°

Table 3
Comparative table of the *Corisco* performance in the ApaSt dataset for a few selected combinations of parameters, and the performance of an alternative method.

Method	Time	Error				
	[s]	Mean	σ	1/4	Median	3/4
J-linkage	0.74	2.61°	3.28°	1.32°	1.87°	2.77°
Corisco $C_r = 10^4$ $C_g = 1$	91.69	0.65°	1.37°	0.29°	0.47°	0.60°
Corisco $C_r = 10^4$ $C_g = 4$	28.05	0.58°	0.27°	0.39°	0.56°	0.78°
Corisco $C_r = 10^4$ $C_g = 32$	9.20	2.51°	4.60°	0.79°	1.39°	2.33°
Corisco $C_r = 10^3$ $C_g = 1$	17.65	2.02°	4.93°	0.31°	0.47°	0.68°
Corisco $C_r = 10^3$ $C_g = 4$	5.05	1.47°	4.34°	0.39°	0.56°	0.78°
Corisco $C_r = 10^3$ $C_g = 32$	1.51	3.45°	5.36°	0.79°	1.46°	2.63°
Corisco $C_r = 200$ $C_g = 1$	14.44	3.90°	7.31°	0.36°	0.56°	0.80°
Corisco $C_r = 200$ $C_g = 4$	3.90	4.43°	7.81°	0.42°	0.67°	0.96°
Corisco $C_r = 200$ $C_g = 32$	0.81	8.84°	9.82°	0.89°	3.55°	16.45°

with this projection in the same way as in the previous ones, and is equally effective.

The images were analyzed using two different combinations of parameters, first with $C_r = 10000$ and $C_g = 1$, and later with $C_r = 1000$ and $C_g = 16$. The median and the third quartile found in the error distribution for the first pair of parameters were located at 0.37° and 0.53° respectively, and the maximum observed error was 2.28°. The mean process duration in this case was 62.11 s. With the second pair of parameters the median and third quartile were 0.73° and 1.07°, the maximum error was 4.31° and the mean time was 1.55 s. This performance was relatively good, and comparable to what was observed in the previous experiments.

4.4. Fisheye lens

The last experiment performed with *Corisco* was the analysis of images captured with a Fujinon FE185C046HA fisheye lens attached to a Basler Ace acA1300-30gc camera. The intrinsic parameters from this camera, assumed to follow the Polar Equidistant projection from Table 1, were obtained by *bundle adjustment* [39,18] with a known calibration target.

Fig. 9 shows one example image from this set. The *Corisco* performance was not evaluated with reference orientations, but it is possible to see in Fig. 9 that the method was effective. These images were also re-mapped into perspective projection images, taking the estimated orientation into account so that the image planes were aligned to the environment directions. The resulting image edges were determined to be aligned to the image frame in a visual inspection.

5. Conclusions

This article presented *Corisco*, an edgel-based orientation estimation method that extended previous works [12–16], and was also influenced by other edgel-based vision approaches [17,24]. *Corisco* starts with an edgel extraction procedure based on edge detection and a grid mask. The estimation happens by a minimization of an error function, defined using M-estimation, comparing the measured direction from the edgels to predicted directions calculated with a camera model. The proposed optimization process starts with the flexible and robust RANSAC algorithm, and finishes with a more accurate continuous optimization performed by FilterSQP. The objective function proposed is relatively simple, and allows for closed formulas for its derivatives in FilterSQP. The grid mask spacing C_g and the number of RANSAC iterations C_r are two control parameters that can be used to regulate the compromise between speed and accuracy.

Corisco managed to create a method that is simpler, more flexible and more robust than the alternatives. The experiments, conducted with a variety of camera models and reference orientation sources, also demonstrated its accuracy, flexibility and competitive performance. The experiments also validated the proposed mechanisms to compromise between speed and accuracy. In some cases a speedup of 10 times was achieved while causing little accuracy loss. Because *Corisco* has excellent performance, works with any camera model and allows tuning the

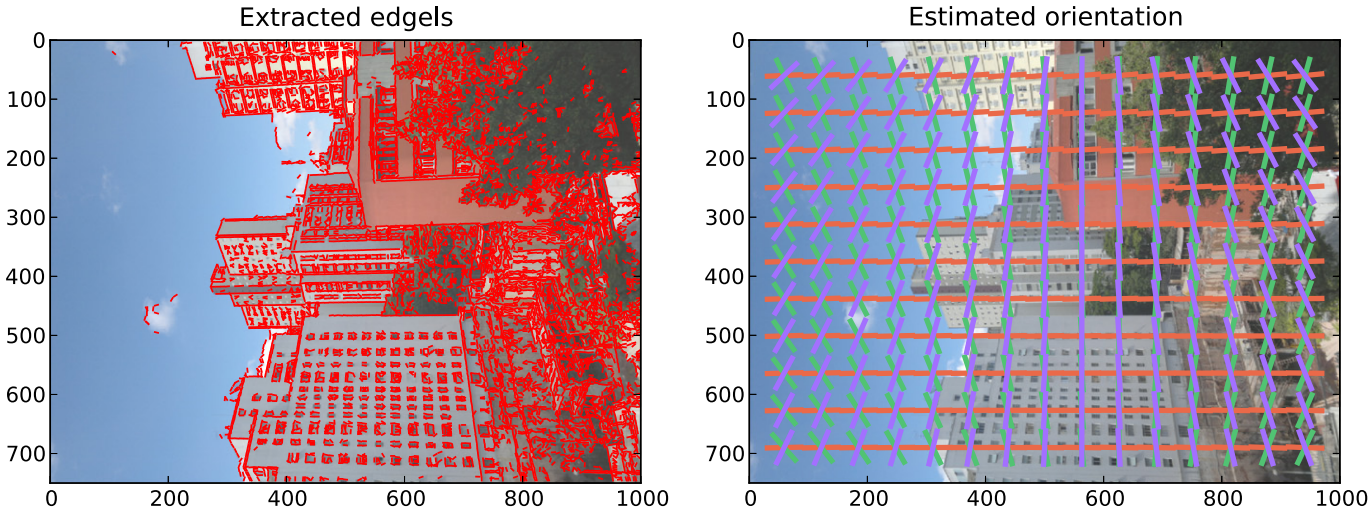

Fig. 6. Example image from the ApaSt dataset ($C_g = 8$ $C_r = 10{,}000$). The left graph shows the extracted edgels, and the right shows the predicted direction from the orientation estimated by *Corisco* using the camera models provided.

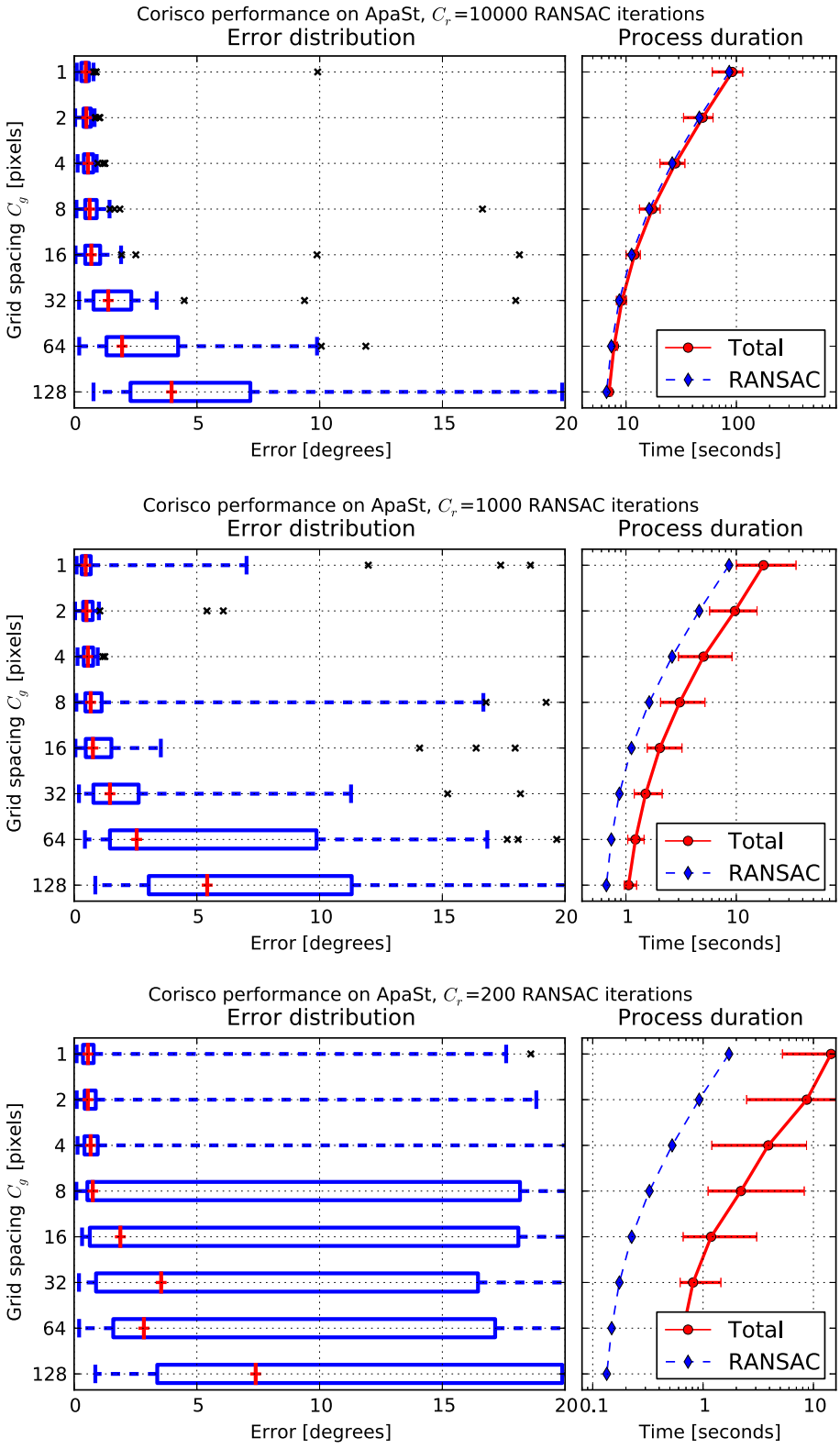

Fig. 7. *Corisco* performance for the ApaSt dataset, similar to Fig. 5.

process to achieve better performances, it encourages wider use of edgel-based orientation estimation techniques.

This research also illustrates the benefits from investigating techniques such as M-estimation and non-linear optimization algorithms such as FilterSQP. We agree with Meer [40] in that investigating these techniques is important to the development of Computer Vision.

Despite the good performance that *Corisco* achieves, it might prove interesting to investigate further enhancements of its optimization procedure, starting by replacing the proposed RANSAC implementation with alternative algorithms [33,21,41,20]. It should be particularly interesting to consider techniques that were successfully applied in line-based orientation estimation methods, such as J-linkage employed in the alternative method prominently featured in this article [7], and

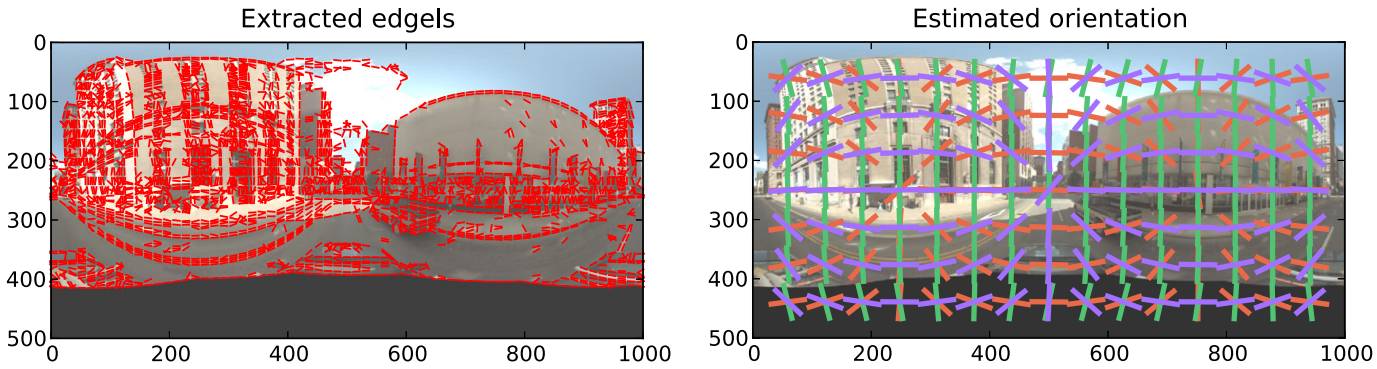

Fig. 8. Example image from the StreetView dataset ($C_g = 16$ $C_r = 10000$).

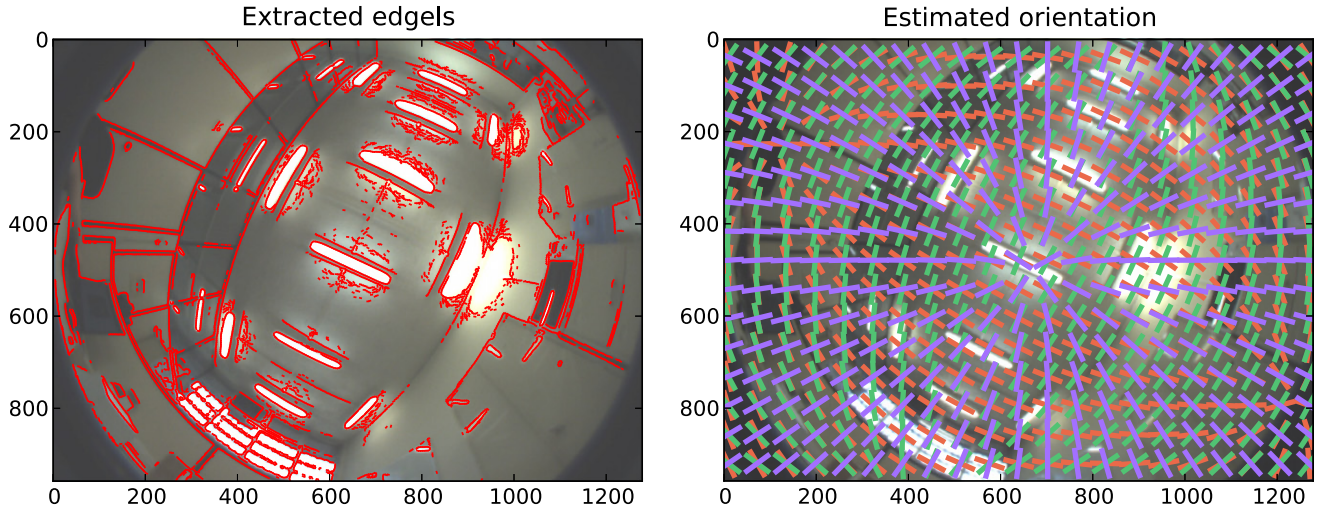

Fig. 9. Example image from the fisheye lens experiment ($C_g = 4$ $C_r = 1000$).

which exhibited a better execution time than the proposed method in one of the experiments. Another alternative is to employ recently proposed techniques such as the branch-and-bound procedure by Bazin et al. [42], or the Facility Location Problem techniques by Antunes and Barreto [43].

It is also possible to investigate the application of other Robust Statistics techniques [31] and other error functions to this problem, replacing the choice of M-estimation and the Tukey bisquare function. And finally, it might be interesting to investigate alternative ways to extract the edgels [44,45,24] and to calculate the residues [46]. The interplay of the grid-masking procedure with image smoothing and down-scaling should also be better studied.

One interesting way to extend *Corisco* might be to cluster the edgels into straight lines or curves, performing a line extraction along with the orientation estimation, possibly creating a feedback process similar to the one proposed by Ji and Fermüller [47]. The existence of estimation biases such as discussed in that article should also be studied.

There are two immediate applications for *Corisco* under study right now. The first is to use it as a visual compass with data fusion techniques to enhance the localization of a robot estimated from odometry data. The other is to perform a preliminary orientation estimation with environment reconstruction systems [1,2,48]. *Corisco* can also be adapted to analyze plane normals extracted from 3D point clouds obtained by sensors such as laser rangers or structured light rigs.

Acknowledgments

We acknowledge the support provided by CAPES, FAPESP (process no. 2011/19280-8), and CNPq (process no. 311058/2011-6).

References

[1] S. Sinha, D. Steedly, R. Szeliski, A multi-stage linear approach to structure from motion, ECCV Workshop on Reconstruction and Modeling of Large Scale 3D Virtual Environments, vol. 51, Springer, Hersonissos, Crete, Greece, 2010, pp. 1–14, (URL http://research.microsoft.com/pubs/138039/Sinha-RMLE10.pdf).

[2] A. Flint, C. Mei, I. Reid, D. Murray, Growing semantically meaningful models for visual SLAM, Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 467–474, http://dx.doi.org/10.1109/CVPR.2010.5540176, (URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnum ber=5540176).

[3] B. Caprile, V. Torre, Using vanishing points for camera calibration, Int. J. Comput. Vis. 4 (2) (1990) 127–139, http://dx.doi.org/10.1007/BF00127813(URL http://www.springerlink.com/content/k75077108473tm15/).

[4] R. Cipolla, T. Drummond, D. Robertson, Camera calibration from vanishing points in images of architectural scenes, British Machine Vision Conference, vol. 2, BMVA, Nottingham, England, 1999, pp. 382–391.

[5] C. Rother, A new approach to vanishing point detection in architectural environments, Image Vis. Comput. 20 (9–10) (2002) 647–655, http://dx.doi.org/10.1016/S0262-8856(02)00054-9(URL http://dx.doi.org/10.1016/S0262-8856(02)00054-9).

[6] L. Grammatikopoulos, G. Karras, E. Petsa, An automatic approach for camera calibration from vanishing points, ISPRS J. Photogramm. Remote Sens. 62 (1) (2007) 64–76, http://dx.doi.org/10.1016/j.isprsjprs.2007.02.002(URL http://dx.doi.org/10.1016/j.isprsjprs.2007.02.002).

[7] J.-P. Tardif, Non-iterative approach for fast and accurate vanishing point detection, International Conference on Computer Vision, IEEE, Kyoto, Japan, 2009, http://dx.doi.org/10.1109/ICCV.2009.5459328, (URL http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber= 5459328).

[8] J.P. Barreto, H. Araujo, Geometric properties of central catadioptric line images and their application in calibration, IEEE Trans. Pattern. Anal. Mach. Intell. 27 (8) (2005) 1327–1333, http://dx.doi.org/10.1109/TPAMI.2005.163(URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnum ber=1453519).

[9] R. Strand, E. Hayman, Correcting radial distortion by circle fitting, British Machine Vision Conference, British Machine Vision Association, Oxford, England, 2005, http://dx.doi.org/10.5244/C.19.9, (URL http://www.bmva.org/bmvc/2005/papers/paper-138.html).

[10] C. Hughes, R. McFeely, P. Denny, M. Glavin, E. Jones, Equidistant fish-eye perspective with application in distortion centre estimation, Image Vis. Comput. 28 (3) (2010) 538–551, http://dx.doi.org/10.1016/j.imavis.2009.09.001(URL http://linkinghub.elsevier.com/retrieve/pii/S0262885609 001863).

[11] J.M. Coughlan, A.L. Yuille, The Manhattan World Assumption: regularities in scene statistics which enable Bayesian inference, Neural Information Processing Systems, 2000.

[12] J.M. Coughlan, A.L. Yuille, Manhattan World: orientation and outlier detection by Bayesian inference, Neural Comput. 15 (5) (2003) 1063–1088(URL http://www.mitpressjournals.org/doi/abs/10.1162/0899766 03765202668).

[13] J. Deutscher, M. Isard, J. MacCormick, Automatic camera calibration from a single Manhattan image, in: A. Heyden, G. Sparr, M. Nielsen, P. Johansen (Eds.), European Conference on Computer Vision, Vol. 2353 of Lecture Notes in Computer Science, vol. 2353, Springer, 2002, pp. 373–377, http://dx.doi.org/10.1007/3-540-47979-1, (URL http://www.springerlink.com/content/pxm2t0e8y3t65q4m/).

[14] G. Schindler, F. Dellaert, Atlanta World: an expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments, Conference on Computer Vision and Pattern Recognition, IEEE, Washington, DC, USA, 2004, pp. 203–209, http://dx.doi.org/10.1109/CVPR.2004.1315033, (URL http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumbe r=1315033).

[15] P. Denis, J.H. Elder, F.J. Estrada, Efficient edge-based methods for estimating Manhattan frames in urban imagery, European Conference on Computer Vision, Springer, Marselha, França, 2008, pp. 197–210, http://dx.doi.org/10.1007/978-3-540-88688-4_15.

[16] N.L. Werneck, A.H.R. Costa, Speeding up probabilistic inference of camera orientation by function approximation and grid masking, in: G. Baranoski, V. Skala (Eds.), WSCG 2011 Communication Papers, UNION Agency, Plzen, Czech Republic, 2011, pp. 127–134, (URL http://nwerneck.sdf.org/almoxarifado/nic-wscg2011.pdf).

[17] E. Rosten, R. Loveland, Camera distortion self-calibration using the plumb-line constraint and minimal Hough entropy, Mach. Vis. Appl. 22 (1) (2009) 77–85, http://dx.doi.org/10.1007/s00138-009-0196-9(URL http://www.springerlink.com/content/dl72369q405n8327/).

[18] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, 2nd edition Cambridge University Press, 2003., http://dx.doi.org/10.1016/S0143-8166(01)00145-2 (URL http://www.robots.ox.ac.uk/ vgg/hzbook/).

[19] C. Geyer, K. Daniilidis, Paracatadioptric camera calibration, IEEE Trans. Pattern. Anal. Mach. Intell. 24 (5) (2002) 687–695, http://dx.doi.org/10.1109/34.1000241(URL http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=1000241&contentType=Journals+&+Magazines).

[20] R. Toldo, A. Fusiello, Robust multiple structures estimation with J-Linkage, European Conference on Computer Vision, vol. 5302, Springer, Marseille, France, 2008, pp. 537–547, http://dx.doi.org/10.1007/978-3-540-88682-2_41, (URL http://www.springerlink.com/content/f0310906x67884qv/).

[21] C.F. Olson, A general method for geometric feature matching and model extraction, Int. J. Comput. Vis. 45 (1) (2001) 39–54, http://dx.doi.org/10.1023/A:1012317923177(URL http://www.springerlink.com/content/h410214304031u38/).

[22] R. Stephens, Probabilistic approach to the Hough transform, Image Vis. Comput. 9 (1) (1991) 66–71, http://dx.doi.org/10.1016/0262-8856(91)90051-P(URL http://dx.doi.org/10.1016/0262-8856(91)90051-P).

[23] J. Matas, C. Galambos, J. Kittler, Robust detection of lines using the progressive probabilistic Hough transform, Comput. Vis. Image Underst. 78 (1) (2000) 119–137, http://dx.doi.org/10.1006/cviu.1999.0831(URL http://linkinghub.elsevier.com/retrieve/pii/S1077314299 908317).

[24] E. Eade, T. Drummond, Edge landmarks in monocular SLAM, Image Vis. Comput. 27 (5) (2009) 588–596, http://dx.doi.org/10.1016/j.imavis.2008.04.012(URL http://linkinghub.elsevier.com/retrieve/pii/S0262885608 000978).

[25] J. Elder, S. Zucker, Local scale control for edge detection and blur estimation, IEEE Trans. Pattern. Anal. Mach. Intell. 20 (7) (1998) 699–716, http://dx.doi.org/10.1109/34.689301(URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnum ber=689301).

[26] J. Canny, A computational approach to edge detection, IEEE Trans. Pattern. Anal. Mach. Intell. (6) (1986) 679–698(URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4 767851).

[27] S. Ando, Consistent gradient operators, IEEE Trans. Pattern. Anal. Mach. Intell. 22 (3) (2000) 252–265, http://dx.doi.org/10.1109/34.841757(URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=& arnumber=841757).

[28] B. Tordoff, D. Murray, The impact of radial distortion on the self-calibration of rotating cameras, Comput. Vis. Image Underst. 96 (1) (2004) 17–34, http://dx.doi.org/10.1016/j.cviu.2004.06.002(URL http://www.sciencedirect.com/science/article/pii/S10773 14204000906).

[29] D. Swart, B. Torrence, The viewable sphere, Math. Horiz. (2011) 14–17, http://dx.doi.org/10.4169/mathhorizons.19.1.14(September).

[30] R. Collins, R. Weiss, Vanishing point calculation as a statistical inference on the unit sphere, International Conference on Computer Vision, IEEE Comput. Soc. Press, 1990, pp. 400–403, http://dx.doi.org/10.1109/ICCV.1990.139560, (URL http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumbe r=139560).

[31] R.A. Maronna, D.R. Martin, V.J. Yohai, Robust Statistics: Theory and Methods, John Wiley & Sons, Hoboken, NJ, USA, 2006.

[32] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (6) (1981) 381–395, http://dx.doi.org/10.1145/358669.358692(URL http://dl.acm.org/citation.cfm?id=358669.358692).

[33] P. Torr, MLESAC: a new robust estimator with application to estimating image geometry, Comput. Vis. Image Underst. 78 (1) (2000) 138–156, http://dx.doi.org/10.1006/cviu.1999.0832(URL http://dx.doi.org/10.1006/cviu.1999.0832).

[34] S. Choi, T. Kim, W. Yu, Performance evaluation of RANSAC family, British Machine Vision Conference, vol. 24, BMVA, London, England, 2009, (URL http://www.bmva.org/bmvc/2009/Papers/Paper355/Paper355. pdf).

[35] R. Fletcher, S. Leyffer, Nonlinear programming without a penalty function, Math. Program. 91 (2) (2002) 239–269, http://dx.doi.org/10.1007/s101070100244(URL http://www.springerlink.com/content/qqj37x00y79ygdl8/).

[36] P.H. Denis, Efficient Edge-based Methods for Estimating Manhattan Frames in Urban Imagery, York University, 2008. (Ph.D. thesis).

[37] N. Snavely, S.M. Seitz, R. Szeliski, Photo Tourism: Exploring Photo Collections in 3D, SIGGRAPH, ACM, Boston, MA, USA, 2006, pp. 835–846, (URL http://dl.acm.org/citation.cfm?id=1141964).

[38] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, J. Weaver, Google Street View: capturing the world at street level, IEEE Comput. 43 (6) (2010) 32–38, http://dx.doi.org/10.1109/MC.2010.170(URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=& arnumber=5481932).

[39] B. Triggs, P. McLauchlan, R. Hartley, A.W. Fitzgibbon, Bundle adjustment — A modern synthesis, ICCV Workshop on Vision Algorithms: Theory and Practice, vol. 34099, Springer, 2000, pp. 298–372, (URL http://www.springerlink.com/index/PLVCRQ5BX753A2TN.pdf).

[40] P. Meer, Are we making real progress in computer vision today? Image Vis. Comput. 30 (2012) 472–473(URL http://dx.doi.org/10.1016/j.imavis.2011.10.004).

[41] O. Chum, J. Matas, S. Obdrzalek, Enhancing RANSAC by generalized model optimization, Asian Conference on Computer Vision, vol. 2, Asian Federation of Computer Vision Societies, Jeju, Korea, 2004, pp. 812–817, (URL ftp://cmp.felk.cvut.cz/pub/cmp/articles/chum/chum-accv0 4.pdf).

[42] J.-C. Bazin, Y. Seo, C. Demonceaux, P. Vasseur, K. Ikeuchi, I. Kweon, M. Pollefeys, Globally optimal line clustering and vanishing point estimation in Manhattan world, Conference on Computer Vision and Pattern Recognition, No. 1, IEEE, Providence, RI, USA, 2012, http://dx.doi.org/10.1109/CVPR.2012.6247731, (URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=& arnumber=6247731).

[43] M. Antunes, J.P. Barreto, A global approach for the detection of vanishing points and mutually orthogonal vanishing directions, Conference on Computer Vision and Pattern Recognition, IEEE, Portland, OR, USA, 2013, pp. 1336–1343.

[44] E. Lyvers, O. Mitchell, Precision edge contrast and orientation estimation, IEEE Trans. Pattern. Anal. Mach. Intell. 10 (6) (1988) 927–937, http://dx.doi.org/10.1109/34.9114(URL http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumbe r=9114).

[45] S. Ghosal, R. Mehrotra, A moment-based unified approach to image feature detection, IEEE Trans. Image Process. 6 (6) (1997) 781–793, http://dx.doi.org/10.1109/83.585230(URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5 85230).

[46] P. Perona, Deformable kernels for early vision, IEEE Trans. Pattern. Anal. Mach. Intell. 17 (5) (1995) 488–499(URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=3 91394).

[47] H. Ji, C. Fermüller, Noise causes slant underestimation in stereo and motion, Vis. Res. 46 (19) (2006) 3105–3120, http://dx.doi.org/10.1016/j.visres.2006.04.010(URL http://www.ncbi.nlm.nih.gov/pubmed/16750551).

[48] N.L. Werneck, A.H.R. Costa, Mapping with monocular vision in two dimensions, Int. J. Nat. Comput. Res. 1 (4) (2010) 56–65.