Automatic Extraction of Parallel lines in Catadioptric Images

Jean-Charles Bazin, In So Kweon
Robotics and Computer Vision Laboratory, KAIST
Daejeon, KOREA
jcbazin@rcv.kaist.ac.kr *iskweon@kaist.ac.kr*

***Abstract* – Extracting parallel lines in images is an important step for many robotic tasks, such as road extraction, camera calibration, 3D reconstruction, etc… Moreover, nowadays, most of robots use catadioptric cameras since they provide a much wider field of view compared to traditional perspective cameras. Therefore, in this paper, we aim to develop an algorithm which automatically extracts parallel lines in single catadioptric images. We also provide many details concerning implementation issues so that our method can be easily used by researchers from different departments.**

I. INTRODUCTION

Parallel lines are often considered as an important tool for many robotic applications, such as road extraction [8], camera calibration [7], 3D reconstruction [7], etc… Unfortunately, automatically extracting parallel lines is not an easy task. Moreover, most of these works are based on traditional perspective cameras although today's robots are frequently equipped with a catadioptric camera. Contrary to conventional cameras, catadioptric cameras provide a much wider field of view which permits to obtain much more information from the environment and this is one of the main reasons why they are broadly used nowadays. In this paper, we first introduce catadioptric projection and we remind our previous work concerning catadioptric line extraction. Then we present our algorithm for automatic extraction of bundles of parallel lines in single catadioptric images and finally we show some results. This paper provides much information concerning implementation issues so that researchers whose major is not computer vision can easily implement the proposed algorithm.

II. CATADIOPTRIC VISION AND PROJECTION

A. General information

Intuitively, the wider the field of view is, the more information we can gather from the environment. Obviously, an imaging system that might see "in all direction" (cf Figure 1) could gather, for example, not only a higher number of parallel lines but also lines in different directions. Such sensors are simply called "omnidirectional systems". Catadioptric cameras are a specific kind of omnidirectional systems. They are devices which use both mirrors (catadioptric elements) and lenses (dioptric elements) to form images through a conventional camera. Such systems usually have a field of view greater than 180 degrees and are getting both cheaper and more effective. Whereas they have long been used in telescopes (to focus light from stars onto the eye of the observer), only recently they gained in popularity together with other omnidirectional vision systems based on fish-eye lenses or clusters of outwards-looking cameras.

Figure 1: compared to traditional cameras (left), catadioptric systems (right) can gather much more information from the environment, such as a higher number of parallel lines that also lie in different directions.

B. Sphere projection theorem

In [1], Geyer and Daniilidis have introduced the equivalent sphere theorem: a central catadioptric image formation is equivalent to a two step mapping via a sphere (cf Figure 2). First a point *Xw* in 3D world is projected to a unit sphere at *Xs* with respect to the single effective viewpoint *O*, then the point *Xs* on the sphere is perceptively projected to the image plane at *m* from a particular point *Oc*, obtained by calibration. This point *m* would have been the same if we had performed the

traditional catadioptric projection, meaning on the mirror and then on the image plane. This theorem not only permits to generalize all catadioptric projections (parabolic, hyperboloid) but also has strong implications in line detection, as show in the following sections.

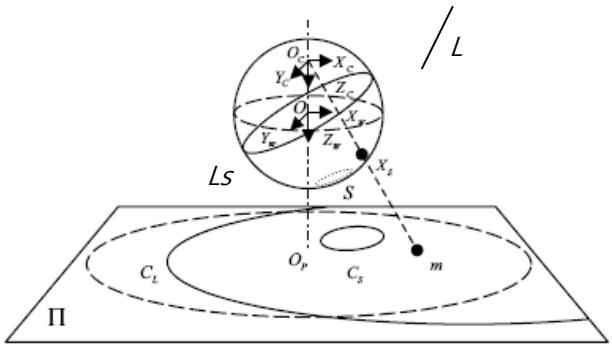

Fig.2: by sphere projection theorem, a line *L* in space is projected to a great circle *Ls* on the sphere, which then projected to a conic section C_L on the image plane from *Oc*, obtained by calibration (cf text for details).

C. Sphere projection equations

We suggest working in the sphere space, instead of the catadioptric image plane, for various reasons: taking the special distortions into account, generalizing the method for all catadioptric systems, simple neighborhood definition, etc… Therefore, the catadioptric image must be projected on this new space. Whereas it is an important step in catadioptric vision, the equation projections are either given with no derivations [5] or require specific knowledge in computer vision [9]. By using notations of figure 2, an image point *P* can be projected into a sphere point P_s as explained below (for derivations details, see appendix A1, based on [5]).

$$P = (x, y) \rightarrow P_s = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$$

First the image point *P* is converted to P_i in camera frame coordinates [7]:

$$x' = (x - u_0)/a_u$$

$$y' = (y - v_0)/a_v$$

Where (u_0,v_0) is the camera center and (a_u,a_v) the focal length in horizontal and vertical directions (details in [7]). Then we project this new point P_i into the sphere point *Ps*, based on spherical coordinates with parameters ϕ and θ:

$$[\phi, D] = cart2pol(x', y')$$

$$\theta = \text{acos}\left(\frac{1}{d^2}\left(\xi D^2 - (\xi+\varphi)\sqrt{(\xi+\varphi)^2 + D^2(1-\xi^2)}\right)\right)$$

Where $d^2 = \|O_p P_i\|^2 = D^2 + (\xi+\varphi)^2$ and parameters ξ and φ are depicted on figure 3. The calibration parameters (u_0, v_0, a_u, a_v, ξ and φ) can be easily obtained by the toolbox [6].

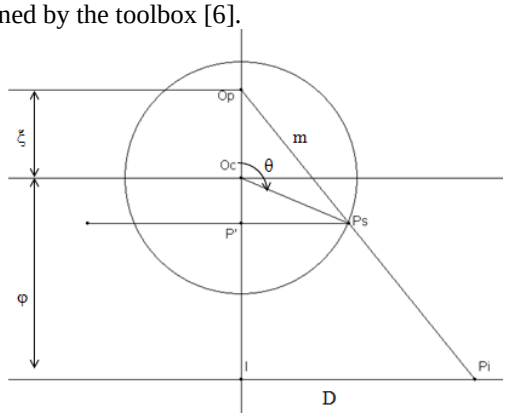

Fig 3: an image point Pi can be projected into the sphere at Ps and reciprocally

III. LINE EXTRACTION

In [3], we have introduced a method to extract lines in catadioptric images. We remind this algorithm for many reasons: convenience of the readers, explaining the parallel lines extraction method more clearly, and providing important information for an easy and efficient implementation.

A. General Framework

As explained, a line in space is projected as a great circle on the sphere. Let "associated plane" be the plane that goes through the sphere center O and the given great circle. Thus, each great circle can be characterized by the normal of its associated plane. The goal of the algorithm is to find the normal vectors. It is composed of five main steps as depicted in Figure 4 and is an extension of the polygonal approximation approach for perspective cameras towards catadioptric vision.

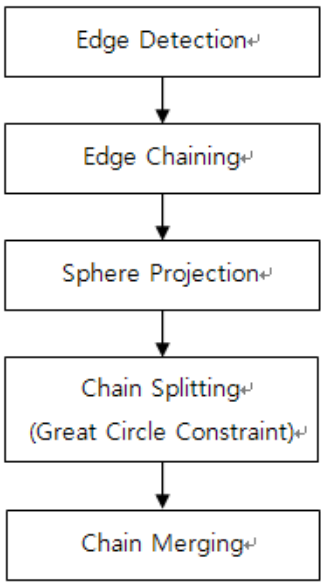

Figure 4: framework of our line extraction algorithm

Step 1 consists in detecting edges, typically by Canny Edge operator on the catadioptric images. Step 2, edge chaining, permits to link the connected edge points into chains by [10]. At step 3, each edge is projected on the sphere using projection equations. Instead of computing the projections at each frame, which is very time consuming, we suggest using a LookUp table that saves the correspondences between the image pixels and the sphere coordinates.

A. Splitting Step

The splitting step is the most important one: for each edge chain, we test whether it verifies the great circle constraint. If it does not, the edge chain is split until the great circle constraint is verified or the chain length is too small.

Let n=(a,b,c) be the normal vector of a great circle. Thus its plane equation is $ax+by+cz=0$ A point (x_s, y_s, z_s) verifies the great circle constraint if $abs(ax_s+by_s+cz_s) < DistThresh$. Formally and in an implementation point of view, we apply the following function (Fig 5) for each edge chain:

```
function LineDetection(EdgePoints)
begin
  if length(EdgePoints)<LengthTresh
    return;
  else
    NormalVect=cross(EdgePoints(1),
                     EdgePoints(last));
    Dist= NormalVect.* EdgePoints;
    DistMean=mean(Dist)
    if DistMean <DistThresh
      Add(NormalVect, NormalVectList)
    else
      [val d]=max(Dist); %returns max value and index
      LineDetection(EdgePoints(1:d));
      LineDetection(EdgePoints(d+1:last));
    end
  end
end
```

Fig 5: pseudo-code for line splitting

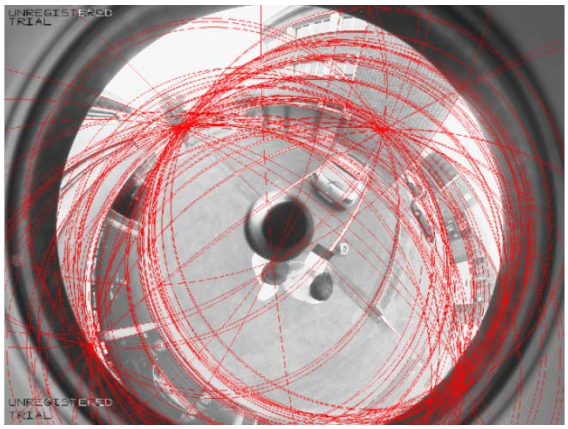

Fig 6: typical result after line splitting

The final output of the LineDetection function is a list of normal vectors whose associated edge chain points verify the great circle constraint, ie they belong to a line. For readers information, we have used LengthTresh=50 and DistThresh=0.004.

B. Merging Step

During the edge detection step, a 3D line might not be continuously detected as an edge. For example, a side of a building can be partially occluded by a tree. Thus this line may be divided into two or many edge parts during edge detection step and thus the splitting step will compute one great circle (ie normal vector) for each edge part. Thus the idea is to simply merge the similar normal vectors. Two normal vectors n_i and n_j are said similar if:

$$\text{acos}\left(\vec{n_i}.\vec{n_j}^{T}\right) < AngleThresh$$

Depending on the testing order, different results might be obtained. That is why we had also suggested using a graph approach and compute the new normal vectors using all the points of similar normals by Singular Value Decomposition.

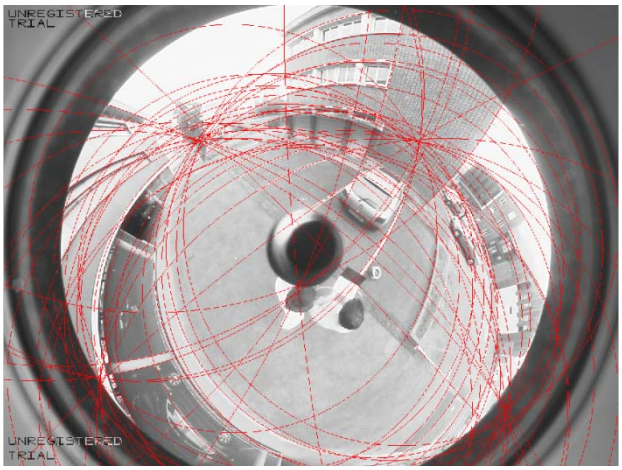

Fig 7: typical result after line merging

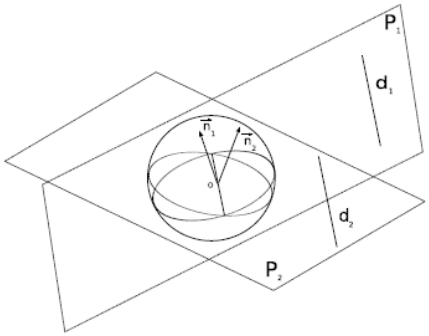

Fig 8: the direction of two parallel lines corresponds to the cross product of the normal vectors of their associated planes ([2],[4]).

IV. PARALLEL LINES EXTRACTION

Now that we have detected lines, the goal is to extract the parallel lines. As explained in [2], a set of world parallel lines intersect in two antipodal points in the sphere space. These two points are actually the vanishing points and can be characterized by a unit vector u. Our idea consists in detecting the set of lines that correspond to a same vector u. Given two catadioptric lines defined by n_1 and n_2, their vanishing direction can be computed by $u = n_1 \times n_2$, as depicted in Figure 8. It means that for each pair of lines, a direction u can be calculated. Therefore the goal is to detect the lines that have a similar direction. In [4], the authors argued that "if at least three lines have the same intersection points, they consider them as a bundle". When we know the normals of two great circles, we can draw their corresponding conics in the image and then detect their two intersection points. However, their detections might be done independently (one intersecting point and then the other one) and thus if we project these two intersecting points in the sphere, there might not be antipodal points. Therefore, instead of computing the intersection in the image, we detect the intersection directly in the sphere which will impose the antipodal constraint. Concretely, noting N the number of detected lines after the merging step, we compute

$\vec{u}_k = \vec{n}_i \times \vec{n}_j$, where i=1..N, j>i

We impose up-vector ($\vec{u}_k(3) \geq 0$) and unitary vector $\|\vec{u}_k\| = 1$. Therefore each u_k belongs to a semi-sphere (cf figure 9-a). This operation cost N(N-1)/2

If at least K u_k are similar then we consider that its associate normals, ie conics, are parallel. The similarity measure can be the same than in merging step.

We have also worked on extracting dominant directions. Intuitively, dominant directions are those having a high number of similar directions. However, this technique cannot be directly applied. Indeed, let *u* be a direction which has many similar directions which correspond to a set *S* composed of these similar directions. Then any element of *S* might also be similar with many elements of the same set *S*. It means that we could find many dominant directions in a similar set and thus it will be hard to detect other main directions. Therefore we suggest the following idea. Let note {S} the set of all directions, u the dominant direction (still unknown) u_i the direction which has the highest number of similar directions, *l_d* the list of these corresponding directions including u_i, *l_n* the list of all normal vectors composing *l_d*.

Thus, we easily compute the main direction $\vec{u}$ as $u = \text{mean}(l\_d)$. Then, in order to detect the second dominant direction, we perform the same method on the set {S} ={S-*l_n*}, ie the set of all directions except those whose any of the normal vectors belong to *l_n*. (cf Figure 9 - b). It means that we remove the direction vectors whose corresponding line has already been detected as a dominant direction, since a line cannot have different orientations.

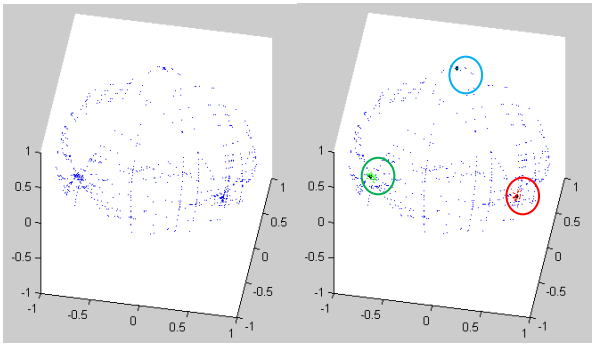

Fig 9: Left (a): the set of all directions are projected on a semi-sphere. We can notice at least two dense regions, each one corresponding to a dominant direction. Right (b): detection of the 3 densest regions (red, green, blue).

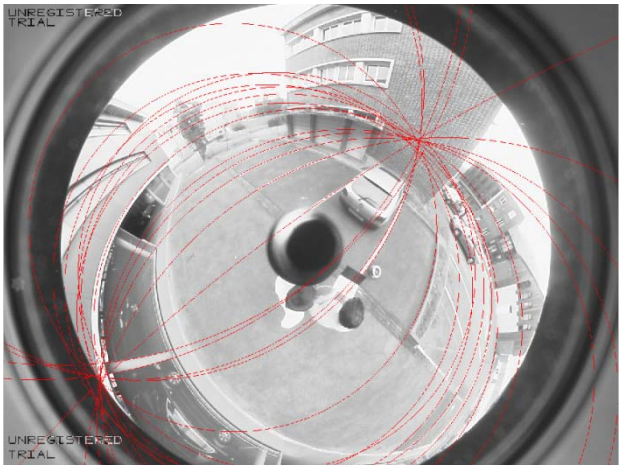

Figure 10: the set of parallel lines corresponding to the most dominant direction

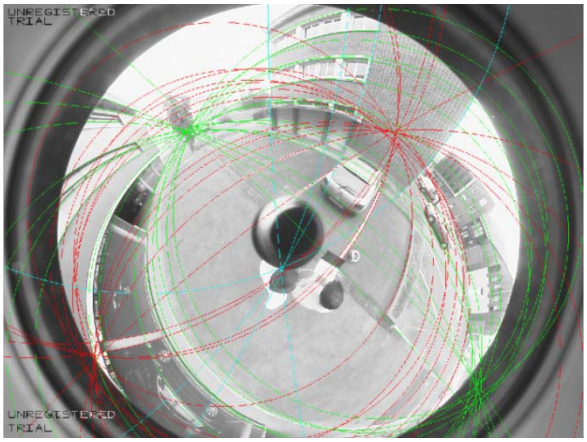

Figure 11: the set of parallel lines corresponding to the three most dominant directions. The line color corresponds to the region color in figure 9-b.

V. CONCLUSION

This paper deals with the problem of parallel line extraction in catadioptric images. We have presented an algorithm that runs fully automatically with even one single image and that does not require any knowledge about the scene. We have also explained how to extract dominant directions. Finally, we have enhanced many implementation issues so that even non-experts in computer vision could implement the proposed algorithm easily. Near future works will be dedicated to the application of parallel lines extraction for various tasks, especially road extraction and 3D reconstruction.

ACKNOWLEDGMENT

This work has been possible thanks to our international collaboration with Pascal Vasseur and Cedric Demonceaux, CREA – France

ANNEXE

Annexe 1: Projection on the sphere

OpPsPi is obviously a line. Thus $\overrightarrow{O_pP_s} = \lambda\overrightarrow{O_pP_i}$

In distance point of view: $\left\|\overrightarrow{O_pP_s}\right\| = \lambda\left\|\overrightarrow{O_pP_i}\right\|$, ie

$$m = \lambda P = \lambda\sqrt{D^2 + (\xi+\varphi)^2}$$

The only variable that we do not know is *m* so let's compute it.

In triangle OpPcPs, by using Al-Kashi's theorem:

$$R^2 = m^2 + \xi^2 - 2m\xi\cos\varphi$$

In triangle OpIPi, we have $\cos\varphi = \frac{\xi+\varphi}{P}$ where $P = \sqrt{D^2 + (\xi+\varphi)^2}$ by Pythagoras

$$R^2 = m^2 + \xi^2 - 2m\xi\frac{\xi+\varphi}{P}$$

$$Pm^2 - 2\xi(\xi+\varphi)m + P(\xi^2 - R^2) = 0$$

This is a second order equation

First coefficient is positive, second is negative, third is negative (since $\xi < R$)

$$\Delta = \xi^2(\xi+\varphi)^2 - P^2(\xi^2 - R^2)$$

$$m = \frac{\xi(\xi+\varphi) + \sqrt{\xi^2(\xi+\varphi)^2 - P^2(\xi^2 - R^2)}}{P}$$

By Thales in triangle OpP'Ps or trigonometry,

$$\frac{m}{P} = \frac{\xi+l}{\xi+\varphi} \text{ ie } l = \frac{m(\xi+\varphi)}{P} - \xi$$

$$\cos(\pi-\theta) = \frac{l}{R}, \text{ ie } \cos\theta = -\frac{l}{R}$$

Thus $\cos\theta = -\frac{m(\xi+\varphi)}{PR} + \frac{\xi}{R}$

Where

$$m = \frac{\xi(\xi+\varphi) + \sqrt{\xi^2(\xi+\varphi)^2 - P^2(\xi^2 - R^2)}}{P}$$

We can use this previous relation or manipulate it in order to obtain the paper's version.

$$\cos\theta = -\frac{m(\xi+\varphi)}{PR} + \frac{\xi}{R} = \frac{1}{P^2}\left(\frac{\xi P^2}{R} - \frac{mP(\xi+\varphi)}{PR}\right)$$

Let $R = 1$ and use the expression of *m*.

$$\cos\theta = \frac{1}{P^2}\left(\xi P^2 - \xi(\xi+\varphi)^2 + (\xi+\varphi)\sqrt{\xi^2(\xi+\varphi)^2 - P^2(\xi^2 - 1^2)}\right)$$

$$\cos\theta = \frac{1}{P^2}\left(\xi P^2 - \xi(\xi+\varphi)^2 + (\xi+\varphi)\sqrt{\xi^2(\xi+\varphi)^2 - P^2(\xi^2 - 1)}\right)$$

By noting that $P^2 = D^2 + (\xi+\varphi)^2$, we finally obtain

$$\cos\theta = \frac{1}{P^2}\left(\xi D^2 - (\xi+\varphi)\sqrt{(\xi+\varphi)^2 + D^2(1-\xi^2)}\right)$$

REFERENCES

[1] C. Geyer and K. Daniilidis, "a unifying theory for central panoramic systems and practical implications", ECCV, pp.445-462, 2000.

[2] C. Geyer and K. Daniilidis, "Catadioptric camera calibration", Proceedings of 7th International Conference on Computer Vision (ICCV), Vol. 1, pp. 398-404, 1999.

[3] J.C. Bazin, C. Demonceaux, P. Vasseur, "Fast Central Catadioptric Line Extraction", Iberian Pattern Recognition and Image Analysis (IbPRIA), 2007.

[4] C. Demonceaux and P. Vasseur "UAV Attitude Computation by Omnidirectional Vision in Urban Environment", International Conference on Robotics and Automation (ICRA), 2007.

[5] C. Demonceaux, P. Vasseur, "UAV Attitude Computation by Central Catadioptric Vision ",15ème Congrès de Reconnaissance des Formes et Intelligence Artificielle (RFIA), 2006.

[6] C. Mei, open-source calibration toolbox for catadioptric cameras: http://www-sop.inria.fr/icare/

personnel/Christopher.Mei/Toolbox.html
[7] R. Hartley and A. Zisserman. “Multiple View Geometry in Computer Vision”, Cambridge University Press, second edition, 2004.
[8] Massimo Bertozzi and Alberto Broggi, GOLD: a Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection, IEEE Transactions on Image Processing, 7(1):62-81, 1998.
[9] J.P. Barreto, “General Central Projection Systems: Modelling, Calibration and Visual Servoing”, PhD Thesis, University of Coimbra, 2003.
[10] P. D. Kovesi. MATLAB and Octave Functions for Computer Vision and Image Processing, School of Computer Science & Software Engineering, The University of Western Australia. Available from: http://www.csse.uwa.edu.au/~pk/research/matlabfns