Direct Fisheye Stereo Correspondence Using Enhanced Unified Camera Model and Semi-Global Matching Algorithm

Bogdan Khomutenko, Gaëtan Garcia, Philippe Martinet

Direct Fisheye Stereo Correspondence Using Enhanced Unified Camera Model and Semi-Global Matching Algorithm

Bogdan Khomutenko and Gaëtan Garcia and Philippe Martinet
IRCCYN, École Centrale de Nantes (ECN), Nantes, France
Email: firstname.lastname@irccyn.ec-nantes.fr

***Abstract*—In this paper, it is shown that the Enhanced Unified Camera Model, which is used to model fisheye cameras, projects straight lines into conic sections. Then a way to find equations of straight line projections is proposed. The method is applied to epipolar curves to adapt the Semi-Global Matching algorithm to fisheye stereo systems to compute a dense direct stereo correspondence without undistortion and rectification of fisheye images. It is done by efficient image sampling along an epipolar curve, using an algorithm for rasterization of implicit curves.**

A C++ implementation is available online.

I. Introduction

The fisheye camera is a powerful sensor for omnidirectional perception: it is cheap and robust, but it has one major inconvenience. Its projection is highly non-linear and images taken with them are distorted, which makes it difficult to deal with such problems as epipolar geometry and stereo correspondence.

A stereo correspondence algorithm called *Plane Sweep*, originally proposed in [1], was applied to fisheye cameras in [2] to get the direct stereo correspondence. In the same paper the advantage of the direct fisheye correspondence over rectified, pinhole-based version was discussed.

We propose an adaptation of the *Semi-Global Matching Algorithm* [3], which is, perhaps, the most well-known and most-used algorithm for stereo correspondence. It has an efficient and fast-to-compute regularizer. The main assumption of this algorithm is that it is possible to efficiently sample an image along an epipolar line. In order to do it, images are usually rectified and undistorted using the pinhole model, as mentioned above.

A recent approach to camera projection modeling and analysis, called *Projection Surface* and presented in [4], allowed us to find a convenient way to sample fisheye images along an epipolar line. Geyer and Daniilidis in [5] showed that the Unified Camera Model projects straight lines into conics. Here we show that the same is true for the Enhanced Unified Camera Model introduced in [4], and then suggest a method to compute an equation of epipolar curves. In the work on omnidirectional LSD-SLAM [6], the authors use the projection model and its Jacobian matrix at each step to sample the epipolar curve with a step of 1 pixel; then they mention that this way of curve sampling is much more computationally expensive than following a straight line. But if the curve equation is known, it is possible to efficiently follow it, using a curve rasterization algorithm. An algorithm for rasterizing implicit curves has been suggested in [7].

The structure of the paper is as follows. In the second section we review the Enhanced Unified Camera Model, the concept of the Projection surface, then show how to obtain an equation of a straight line projection. In the third section we describe how to compute the epipolar curve equation for a calibrated stereo system, as well as the algorithm for stereo correspondence. In the fourth section we discuss the experimental results. The last section is dedicated to the conclusions and the future work on the subject.

II. Enhanced Unified Camera Model

The basic projection relations are as follows [4]:

$$\boldsymbol{m} = \begin{pmatrix} \dfrac{x}{\alpha\rho + (1-\alpha)z} \\ \dfrac{y}{\alpha\rho + (1-\alpha)z} \\ 1 \end{pmatrix} \quad \rho = \sqrt{\beta(x^2+y^2)+z^2} \qquad (1)$$

$$\boldsymbol{p} = K\boldsymbol{m}$$

The two parameters are $\alpha \in [0,1]$ and $\beta > 0$. They allow us to better approximate the properties of lenses with strong distortion. This model assumes that the denominator $\alpha\rho+(1-\alpha)z > 0$. K is the projection matrix, which transforms normalized coordinates into image coordinates:

$$K = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \end{pmatrix} \qquad (2)$$

A. Projection Surfaces

To analyze the model, let us introduce the notion of *projection surface*. For the sake of simplicity let us consider only projection relations with radial distortion. That is, every projected point can be written as $\boldsymbol{m} = (x/\eta(\boldsymbol{X}) \;\; y/\eta(\boldsymbol{X}) \;\; 1)^T$ where x and y are the components of a spatial point $\boldsymbol{X}$, and $\eta : \mathbb{R}^3 \to \mathbb{R}_+$ is a function of $\boldsymbol{X}$, moreover $\forall \boldsymbol{X}_1, \boldsymbol{X}_2 \in \mathbb{R}^3$:

$$\begin{cases} x_1^2 + y_1^2 = x_2^2 + y_2^2 \\ z_1 = z_2 \end{cases} \implies \eta(\boldsymbol{X}_1) = \eta(\boldsymbol{X}_2) \qquad (3)$$

Also we require:

$$\forall \lambda \in \mathbb{R}_+ \quad \eta(\lambda \boldsymbol{X}) = \lambda \eta(\boldsymbol{X}) \qquad (4)$$

that is, η is a homogeneous function of degree 1.

Let us define the projection surface P by the following equation:

$$\eta(\boldsymbol{X}) = 1 \tag{5}$$

The projection surface is a surface of revolution. The geometric meaning of (5) is that all the points of the projection surface are projected orthogonally to the image plane. So, we can think of the projection process as scaling the point $\boldsymbol{X}$ by $\eta(\boldsymbol{X})$ and then projecting it orthogonally onto $\boldsymbol{m} \in M$ — the normal plane (see Fig. 1):

$$\begin{aligned} \boldsymbol{X}_p &= \frac{\boldsymbol{X}}{\eta(\boldsymbol{X})} \\ \boldsymbol{m} &= (x_p \ y_p \ 1) \end{aligned} \tag{6}$$

Using (4) we can deduce:

$$\eta(\boldsymbol{X}_p) = \eta\left(\frac{\boldsymbol{X}}{\eta(\boldsymbol{X})}\right) = \frac{\eta(\boldsymbol{X})}{\eta(\boldsymbol{X})} = 1 \tag{7}$$

Hence, all the points $\boldsymbol{X}_p$ belong to the projection surface. In fact $\boldsymbol{X}_p$ is the intersection between the projection surface P and the $O\boldsymbol{X}$ ray (Fig. 1).

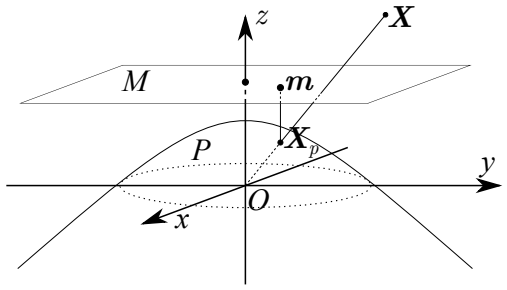

Fig. 1. Illustration of the notion of projection surface. z is the optical axis; O is the center of projection; $\boldsymbol{X}_p$ is obtained by projecting $\boldsymbol{X}$ to P along $O\boldsymbol{X}$ ray. Then this point is transformed into $\boldsymbol{m}$ by projecting it orthogonally onto an intermediate projection plane M which is defined as $z = 1$

One convenience of the notion is that it is relatively easy to see for which spatial points the projection is defined. For example, the classical pinhole camera model corresponds to $\eta(\boldsymbol{X}) = z$. So, the projection surface in this case is defined by $z = 1$. It is a plane. And all the points with $z \leq 0$ do not define rays that intersect the surface.

Let us apply the notion to the proposed model. In this case:

$$\eta(\boldsymbol{X}) = \alpha\sqrt{\beta(x^2 + y^2) + z^2} + (1 - \alpha)z \tag{8}$$

So, $\eta(\boldsymbol{X}) = 1$ leads to:

$$\alpha\sqrt{\beta(x^2 + y^2) + z^2} + (1 - \alpha)z = 1 \tag{9}$$

Let us replace $1 - \alpha$ by γ and $x^2 + y^2$ by r^2:

$$\alpha\sqrt{\beta r^2 + z^2} = 1 - \gamma z \tag{10}$$

By squaring both sides we get:

$$\alpha^2\beta r^2 + \alpha^2 z^2 = 1 - 2\gamma z + \gamma^2 z^2 \tag{11}$$

We should remember that as we have squared both sides, we may get some solutions that do not satisfy (10). We may notice that $\gamma^2 - \alpha^2 = \gamma - \alpha$, hence:

$$\alpha^2\beta r^2 = 1 - 2\gamma z + (\gamma - \alpha)z^2 \tag{12}$$

$$\alpha^2\beta r^2 + (\alpha - \gamma)z^2 + 2\gamma z = 1 \tag{13}$$

B. Straight Line Projection

Using the notion of projection surface, we can show that straight lines are projected as conic sections. Given straight line l let us define plane H which passes through l and O (Fig. 2). It is defined by the following equation:

$$Ax + By + Cz = 0 \tag{14}$$

To find the line projection first we need to project it onto the projection surface. To do it, we need to find intersection c between surface P and plane H. This intersection is defined by the following system of equations:

$$\begin{cases} Ax + By + Cz = 0 \\ \alpha^2\beta(x^2 + y^2) + (\alpha - \gamma)z^2 + 2\gamma z = 1 \end{cases} \tag{15}$$

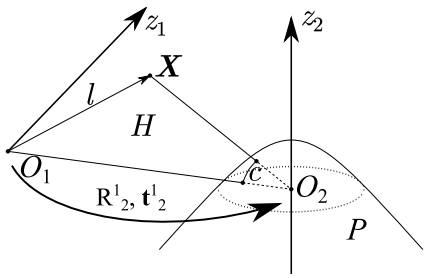

Fig. 2. A calibrated stereo system. R_2^1 and $\boldsymbol{t}_2^1$ define the transformation between the camera frames (rotation matrix and translation vector); $\boldsymbol{X}$ is a reconstructed point; straight line l passes through it and the center of projection of the first camera O_1. Plane H passes through O_2 and l. Curve c is the intersection between the plane and projection surface P. To get an epipolar curve we need to exclude z coordinate from the equation of c.

The next step in the projection process is to project the curve orthogonally onto the intermediate plane M. It means that we need to exclude z from the equations. If $C = 0$ then the projection is a straight line, defined by

$$Ax + By = 0 \tag{16}$$

This line passes through the center of projection. It is logical because the system has only radial distortions and the only straight lines that are projected into straight lines are the ones which pass through the optical axis. If $C \neq 0$ then we can find z from the first equation:

$$z = \frac{Ax + By}{C} \tag{17}$$

and substitute it into the second one:

$$\alpha^2\beta(x^2 + y^2) + (\alpha - \gamma)\left(\frac{Ax + By}{C}\right)^2 + 2\gamma\frac{Ax + By}{C} = 1 \tag{18}$$

As you can see, we have a second degree polynomial of x and y. This polynomial defines a conic section.

III. Stereo Matching Using the Enhanced Model

Using the epipolar geometry described above we can efficiently sample the right image of a stereo image pair along an epipolar line. It allows us to apply the Semi-Global Block Matching algorithm to the problem of stereo reconstruction using the proposed model without image rectification.

A. Epipolar Geometry

We can get closed-form expressions for coefficients of epipolar curve equations for a calibrated stereo system. Consider two cameras with calibrated projection models f_1 and f_2. The transformation between them is known and represented by a rotation matrix R_2^1 and a translation vector $\boldsymbol{t}_2^1$ (see Fig. 2). Hereafter the superscript defines the projection frame. For a reconstructed point $\boldsymbol{X}^1 = f_1^{-1}(\boldsymbol{p}_1)$, we can define a plane that passes through it and both centers of projection. It will be defined by the following equation:

$$\boldsymbol{X}^2 \cdot \left(R_1^2(\boldsymbol{t}_2^1 \times \boldsymbol{X}^1)\right) = 0 \tag{19}$$

This is a classical equation, which introduces the Essential matrix $R_1^2[\boldsymbol{t}_2^1]_\times$. But what we are interested in is the fact that we have an equation of type (14). Hence we can find the equation of projection of the intersection of this plane with the projection surface of the second camera using (16) or (18).

The final thing to do is to replace x and y by their expressions as functions of u and v:

$$x = \frac{u - u_0}{f_u} \quad y = \frac{v - v_0}{f_v} \tag{20}$$

to get a polynomial of the following kind:

$$k_{uu}u^2 + k_{uv}uv + k_{vv}v^2 + k_u u + k_v v + k_1 = 0 \tag{21}$$

Let us denote this polynomial by $h(u, v)$ or $h(\boldsymbol{p})$. Except for the epipolar lines that pass very close to the projection center (in which case we can apply (16)), the expressions of the polynomial coefficients in this equation are cumbersome, especially k_1:

$$\begin{aligned}
k_{uu} &= \tfrac{A^2\kappa + C^2\alpha^2\beta}{C^2 f_u^2} \\
k_{uv} &= \tfrac{2AB\kappa}{C^2 f_u f_v} \\
k_{vv} &= \tfrac{B^2\kappa + C^2\alpha^2\beta}{C^2 f_v^2} \\
k_u &= \tfrac{-A^2 f_v u_0 \kappa - AB f_u v_0 \kappa - AC f_u f_v \gamma - C^2\alpha^2\beta f_v u_0}{C^2 f_u^2 f_v} \\
k_v &= \tfrac{-AB f_v u_0 \kappa - B^2 f_u v_0 \kappa - BC f_u f_v \gamma - C^2\alpha^2\beta f_u v_0}{C^2 f_u f_v^2} \\
\\
k_1 &= \tfrac{h_{A^2}A^2 + h_{AB}AB + h_{AC}AC + h_{B^2}B^2 + h_{BC}BC + h_{C^2}C^2}{C^2 f_u^2 f_v^2} \\
h_{A^2} &= f_v^2 u_0^2 \kappa \\
h_{AB} &= 2 f_u f_v u_0 v_0 \kappa \\
h_{AC} &= 2 f_u f_v^2 \gamma u_0 \\
h_{B^2} &= f_u^2 v_0^2 \kappa \\
h_{BC} &= 2 f_u^2 f_v \gamma v_0 \\
h_{C^2} &= \alpha^2\beta v_0^2 (f_u^2 + f_v^2) - f_u^2 f_v^2 \\
\\
\kappa &= \alpha - \gamma
\end{aligned} \tag{22}$$

Fortunately, we don't have to evaluate all of them. If we know the projection of the epipole $\boldsymbol{e}$, we can calculate k_1 using the fact that $h(\boldsymbol{e}) = 0$ (since the epipolar curves pass through it). If we denote the first five terms of $h(u, v)$ by $h'(u, v)$ then:

$$k_1 = -h'(\boldsymbol{e}) \tag{23}$$

B. Curve Rasterization

To follow the curve in an efficient manner we have to rasterize it first. The following algorithm requires that the curve be defined by an equation as follows:

$$f(u, v) = 0 \tag{24}$$

Here $f : \mathbb{R}^2 \to \mathbb{R}$ with $\nabla f = (f_u, f_v)^T$ defined; $u, v \in \mathbb{R}$ are the image coordinates. Then we need a starting pixel $\boldsymbol{p}_0 = (u_0, v_0)^T$ and a goal $\boldsymbol{p}_1 = (u_1, v_1)^T$ (see Fig. 3). Since the image is discrete, we cannot require that $f(\boldsymbol{p}_0) = 0$. But at least we can assume that the starting point and the goal are close to the curve. That is the distance from the nearest solution of (24) to either of these points is less than 1. We assume that $\boldsymbol{p}_0 \in \mathbb{Z}^2$ and it might be the case that $\boldsymbol{p}_1 \in \mathbb{Z}^2$, but it is not necessary. Among the two possible directions of the curve starting at $\boldsymbol{p}_0$, the algorithm chooses the one which leads towards the goal $\boldsymbol{p}_1$. More formally, if the chosen direction is defined by a tangent vector $\boldsymbol{t}$, then:

$$\boldsymbol{t} \cdot (\boldsymbol{p}_1 - \boldsymbol{p}_0) > 0 \tag{25}$$

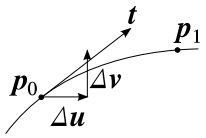

Fig. 3. A curve rasterization process. We start from $\boldsymbol{p}_0$ and make step $\Delta\boldsymbol{u}$ towards $\boldsymbol{p}_1$, because tangential vector $\boldsymbol{t}$ is closer to u-axis, than to v-axis. Then we use the perpendicular direction to compensate the error, that is make step $\Delta\boldsymbol{v}$.

We assume that the angle between $\boldsymbol{t}$ and the desired direction is small, so the singular configuration when $\boldsymbol{t} \cdot (\boldsymbol{p}_1 - \boldsymbol{p}_0) = 0$ is supposed to never happen. $\boldsymbol{t}$ is calculated by rotating gradient vector ∇f by $\pi/2$ clockwise or counterclockwise depending on the goal condition. Then we retain the rotation direction and use it until end of rasterization process. One may notice that

$$\boldsymbol{t} \cdot \boldsymbol{q} = \varepsilon \det[\nabla f \ \boldsymbol{q}] \tag{26}$$

Here ε defines the rotation direction; $\varepsilon = 1$ means counterclockwise, while $\varepsilon = -1$ means clockwise.

Now we can describe algorithm 1. The idea is that at every step we check which direction, u or v, is closer to the actual tangential direction. To do it, we compare components of ∇f by absolute value. Let us say, $|f_v| > |f_u|$. It means that $\boldsymbol{t}$ is closer to u-direction. So we change u by ± 1 depending on the sign of f_v and the rotation direction that was defined before. Then we check what is the actual error of (24) and apply a simple control law to make it smaller:

$$\Delta v = -\mathrm{round}\left(\frac{f(\boldsymbol{p})}{f_v(\boldsymbol{p})}\right) \tag{27}$$

Algorithm 1 Algebraic curve rasterization

$\boldsymbol{p} = (u, v)^T$ ▷ We use u and v to refer to different components of $\boldsymbol{p}$
$\nabla f = (f_u, f_v)^T$ ▷ The same for ∇f, f_u and f_v
$\boldsymbol{p} \leftarrow \boldsymbol{p}_0$
if $\det [\nabla f(\boldsymbol{p}_0) \quad (\boldsymbol{p}_1 - \boldsymbol{p}_0)] > 0$ **then** ▷ The rotation is counterclockwise
    $\varepsilon \leftarrow 1$
else ▷ The rotation is clockwise
    $\varepsilon \leftarrow -1$
end if
for $n = 1..N_{max}$ **do**
    if $|f_u(\boldsymbol{p})| > |f_v(\boldsymbol{p})|$ **then** ▷ The curve tangent is closer to v-direction
        $v \leftarrow v + \varepsilon \text{sign}(f_u(\boldsymbol{p}))$ ▷ First go along the tangent
        $u \leftarrow u - \text{round}(f(\boldsymbol{p})/f_u(\boldsymbol{p}))$ ▷ Compensate the total error
    else ▷ The curve tangent is closer to u-direction
        $u \leftarrow u - \varepsilon \text{sign}(f_v(\boldsymbol{p}))$
        $v \leftarrow v - \text{round}(f(\boldsymbol{p})/f_v(\boldsymbol{p}))$
    end if
    PROCESS($\boldsymbol{p}$) ▷ Either draw a point at the pixel or add it to the list of points
end for

The described algorithm is general and works for any f given that it does not contain features too fine to be properly depicted with the image's sampling step.

C. Semi-Global Matching Algorithm Using the Model

Now we are ready to present the Semi-Global Matching algorithm [3] for two fisheye cameras. The actual change in the algorithm appears only in the first step — error accumulation. The second part stays intact. The global idea of the algorithm is to use an efficient dynamic programming algorithm to regularize the distance map.

a) Error accumulation: The second part of the algorithm requires a table of photometric error for all pixels and all possible disparity values. It means that for a 480p video frame (which is not incredibly high) and maximum allowed disparity of 64 we need $480 \times 720 \times 64 \approx 22M$ bytes of memory which is a lot for a single data buffer. Moreover for the second step we'll need at least 4 times that much for the dynamic programming algorithm. If we want to have better reconstruction quality, this number will be even higher (in the original paper they suggest to perform the dynamic programming in 16 directions). There is a modification of this algorithm, called Semi-Global Block Matching, which significantly reduces the required buffer size. The idea is to divide the image into square blocks and compute the photometric error for them, not for individual pixels. If the block size is 3×3 then it reduces the buffer size by a factor of 9 (as well as computation time of the second step).

We need an assumption that the two cameras are oriented more-less in the same direction and the second camera is on the right of the first one. It allows us to say that the epipolar curves are mostly horizontal and the points move left along them from the first frame to the second.

The final error accumulation step works as follows:

1) For every image block, compute the equation of the epipolar line which corresponds to the block center. This step is required only once as long as the cameras' relative position remains the same.
2) For every epipolar curve we take a narrow band along it (Fig. 4) and save it into a separate rectangular buffer of size $N \times (D_{\max} + N - 1)$, where N is the block size, $D_{\max}$ is the maximum disparity size.
3) Moving the image block along this band, we compute the average absolute photometric error and store it into the corresponding error buffer line.

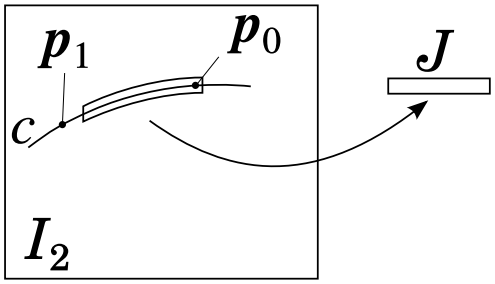

Fig. 4. An illustration of the error accumulation process. Curve c is the epipolar curve of a certain point from the first image. $\boldsymbol{p}_1$ is the epipole; $\boldsymbol{p}_0$ corresponds to the case where the point's depth is infinite. Starting in $\boldsymbol{p}_0$ we go along c for a certain number of steps, then we copy the band of image I_2 around the curve to separate buffer J. Then, sliding with a patch from I_1 along J we compute the cost for different values of disparity. Notice that one pixel step along J is equivalent to one pixel step along c on I_2.

Once we have done this, the error buffer is filled up and we can go to the second step

b) Dynamic programming: Let us denote the error buffer $E : \mathbb{Z}^3 \to \mathbb{R}$. It has two spacial coordinates u, v and one disparity coordinate d. To regularize the disparity map we compute the final cost of a pixel (u, v) having a disparity value d as a sum of costs of cheapest paths through uvd space, whose footprint in uv is a straight line and passing through (u, v, d). A path cost is defined as a sum of all errors along the path plus λ_S times the number of disparity changes by 1 (number of steps) plus λ_J times the number of disparity changes by more than 1 (number of jumps); λ_S and λ_J are the algorithm's parameters, and generally the algorithm's output is quite stable with respect to them. The number of paths defines how regular we want the result to be. The tests show that only two directions is enough to get decent results. But considering the possibility of a GPU implementation we can increase this number to improve the result.

The main interest of this approach is that we can compute the optimal cost of all the paths in a certain direction in linear time with respect to the error buffer size. Assume that we have chosen the number of regularization directions. Suppose that we treat the horizontal direction first. For other directions everything is the same, but for the sake of clarity we explain the algorithm for the horizontal one. Now to know the optimal

path's cost for a given point we need to know the cost of optimal paths on either side. Hence we have to execute the dynamic programming algorithm twice. Once for all the semi-paths from the left side of the image till the point, and the second time from the right side. For each of these tasks we need a cost buffer. Let's call it $C : \mathbb{Z}^3 \to \mathbb{R}$. Then we notice that we will treat every line in uv independently. For a line defined by $v = v_0$ let us denote $C(u, v_0, d)$ by $C_{u,d}$. If we suppose that for a certain u we have the optimal cost in $C_{u,d}$, then the following recursive relation is true [3]:

$$C_{u+1,d} = E_{u+1,d} + \min(C_{u,d}, C_{u,d-1} + \lambda_S, C_{u,d+1} + \lambda_S, C_{u,\min} + \lambda_J) \quad (28)$$

It means that it takes $O(1)$ time to compute one value of C. The complete procedure is described in alg. 2.

Algorithm 2 Dynamic programming algorithm to compute a left-to-right path cost

for $v \in 1..v_{\max}$ **do**
  $C_{u,d} \triangleq C(u, v, d)$
  $E_{u,d} \triangleq E(u, v, d)$
  for $d \in 1..D_{\max}$ **do**
    $C(1, v, d) \leftarrow E(1, v, d)$
  end for
  for $u \in 1..u_{\max} - 1$ **do**
    $C_{\min} \leftarrow \min_d C_{u,d}$
    for $d \in 1..D_{\max}$ **do**
      $C_{u+1,d} \leftarrow E_{u+1,d} + \min(C_{u,d}, C_{u,d-1} + \lambda_S, C_{u,d+1} + \lambda_S, C_{u,\min} + \lambda_J)$
    end for
  end for
end for

IV. Experimental Results

The algorithm was tested using a calibrated camera attached to a robotic arm. This setup allows us to know precisely the transformation between two camera poses. The implementation is available at `https://github.com/BKhomutenko/visgeom`. Since we implemented the most basic version of the algorithm, we had to use textured objects for the reconstruction. In Fig. 5 you can see an example of the algorithm's result an image and its disparity map.

The block size in our case is 3×3; the maximum disparity value is 64; the image size is 1024×768; the time to compute disparity for a pair of images is about 0.3 s for the given parameters (can be improved using parallelization).

To quantify the result, 3D reconstruction of a planar object was performed and compared with ground truth (see Fig. 6). To obtain the ground truth, the transformation between the robot's base and the plane was measured. Knowing the robot's pose it is possible to compute the transformation between the plane and the camera. Then, using the calibrated camera model, the ground truth for distance maps was generated for 6 different camera poses. The planar object does not cover the image entirely, so multiple datasets were acquired to test the approach in different parts of the images. To get the numeric results, disparity maps generated by the algorithm were transformed into distance maps; then the difference between the obtained maps and the ground truth was computed and analyzed. The pixels outside the planar object are ignored. All pixels, whose distance value is off by more than 100 mm, are considered as outliers and rejected.

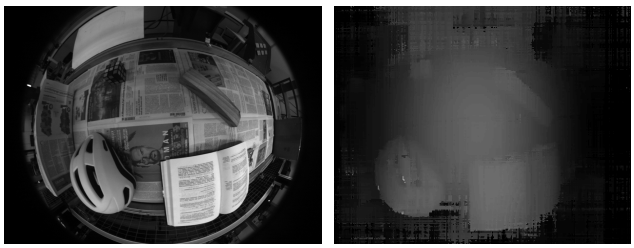

Fig. 5. An example of running the implemented stereo correspondence algorithm. One of the original images (left) and its disparity map (right) for 2 cm stereo base.

Table I represents the error distribution of the distance reconstruction as a function of the stereo base. You can see that increasing the stereo base consistently leads to a decrease in the mean and standard deviation of the error distribution. Table II shows the results for different datasets. Average distances from the camera to the plane are given to give a better idea about the reconstruction precision The error distribution is relatively narrow with respect to the actual distance values. Also the inlier rate is high, which means that overall this algorithm is efficient in plane reconstruction.

TABLE I
Plane reconstruction error as a function of the stereo base.

Stereo Base, mm	Mean Error, mm	σ_{error}, mm	Inliers, %
10	-7.50	23.4	97.8
15	-4.41	18.1	98.8
20	-4.07	15.4	98.7
25	-2.40	13.6	99.0
30	-1.79	12.4	99.1
35	-1.70	11.7	99.1
40	-0.98	11.2	99.0
45	0.09	10.1	98.7

TABLE II
Plane reconstruction error for different datasets. The stereo base is 35mm.

Dataset	Avg Distance, mm	Mean Error, mm	σ_{error}, mm	Inliers, %
1	552	1.00	15.0	99.3
2	381	-1.70	11.7	99.1
3	490	-0.83	15.1	99.7
4	423	-1.29	13.4	98.7
5	356	-2.42	14.1	99.5
6	433	1.61	22.0	98.4

V. Conclusion

The enhanced unified camera model offers nice analytic properties. The fact that straight lines become conic sections

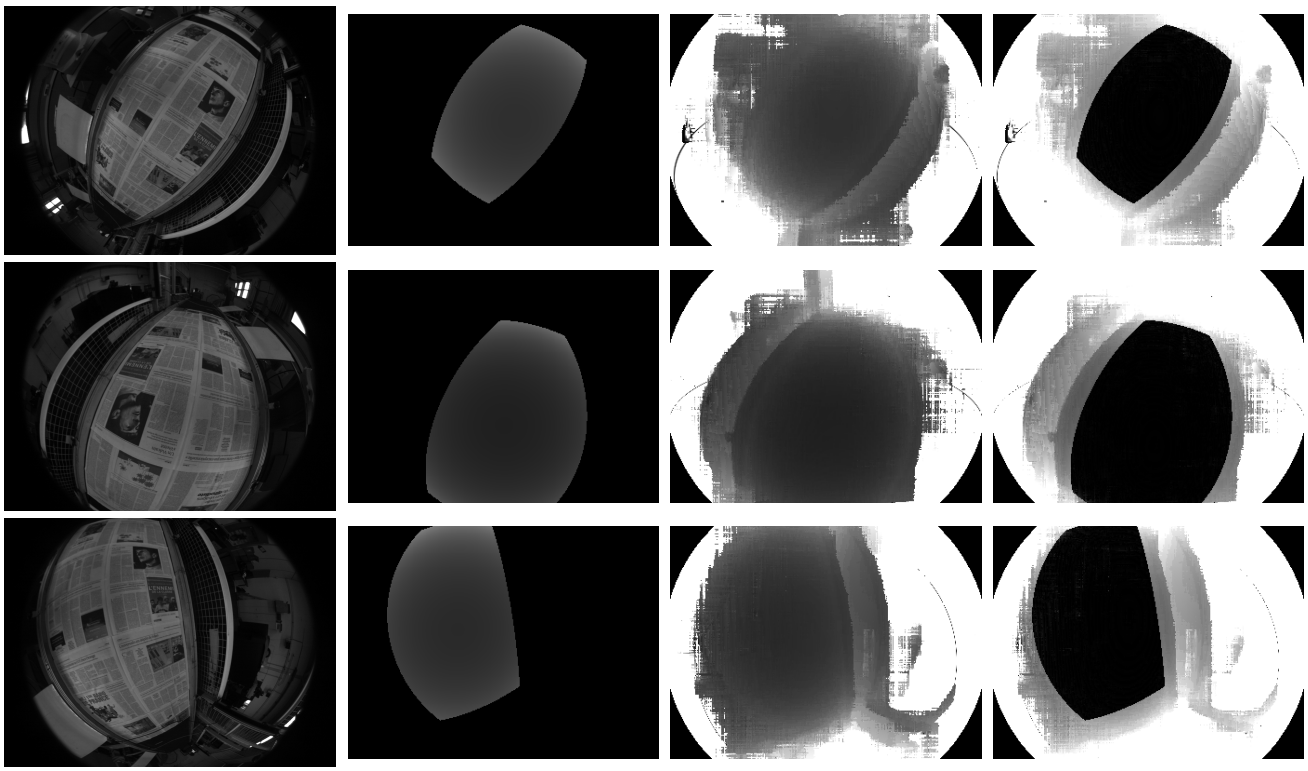

Fig. 6. An illustration of the evaluation methodology. Each row corresponds to a different dataset (from the top: 3, 5, 6); a — the reference image; b — the ground truth of the newspaper plane distance map; c — the reconstruction result using disparity maps, which were computed using the camera's intrinsics and the robot poses as extrinsic parameters of stereo; d — the absolute difference between the ground truth and the reconstruction using the stereo.

after being projected offers us new ways to treat problems of visual geometry in case of fisheye camera. The proposed approach to stereo correspondence problem, which does not require rectification and undistortion, may be applied in mobile robotics as an omnidirectional depth sensor. In fact, any other stereo correspondence algorithm based on search along epipolar lines can be applied to fisheye cameras using the proposed approach or its slight modification.

Other possible applications of the geometric results above are visual servoing using straight lines, straight line reconstruction, and straight-line-based SLAM.

There are multiple ways to improve the algorithm. We can improve the reconstruction quality by using better metrics for patch comparison, which may take into account local affine transformations along epipolar lines. Right now the regularization coefficients are fixed and equal across the image. But they can be computed based on the image content (for example, lower jump cost across edges). By adding more constraint directions the quality of disparity maps can be improved as well (currently only two directions are used).

Another important thing to do is to create a dataset with ground truth and camera intrinsic parameters for testing and comparing direct stereo correspondence algorithms for fisheye cameras.

References

[1] R. T. Collins, "A space-sweep approach to true multi-image matching," in *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, ser. CVPR '96. Washington, DC, USA: IEEE Computer Society, 1996, pp. 358–. [Online]. Available: http://dl.acm.org/citation.cfm?id=794190.794659

[2] C. Häne, L. Heng, G. H. Lee, A. Sizov, and M. Pollefeys, "Real-time direct dense matching on fisheye images using plane-sweeping stereo," in *Proceedings of the 2014 2Nd International Conference on 3D Vision - Volume 01*, ser. 3DV '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 57–64. [Online]. Available: http://dx.doi.org/10.1109/3DV.2014.77

[3] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, June 2005, pp. 807–814 vol. 2.

[4] B. Khomutenko, G. Garcia, and P. Martinet, "An Enhanced Unified Camera Model," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 137–144, Jan 2016.

[5] C. Geyer and K. Daniilidis, "A unifying theory for central panoramic systems and practical applications," in *Proceedings of the 6th European Conference on Computer Vision-Part II*, ser. ECCV '00. London, UK, UK: Springer-Verlag, 2000, pp. 445–461. [Online]. Available: http://dl.acm.org/citation.cfm?id=645314.649434

[6] D. Caruso, J. Engel, and D. Cremers, "Large-scale direct slam for omnidirectional cameras," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, Sept 2015, pp. 141–148.

[7] J. D. Hobby, "Rasterization of nonparametric curves," *ACM Trans. Graph.*, vol. 9, no. 3, pp. 262–277, Jul. 1990. [Online]. Available: http://doi.acm.org/10.1145/78964.78966