Scene Flow from Stereo Fisheye Images

Arief Suryadi Satyawan, Hiroshi Watanabe
Department of Computer Science and Communications Engineering,
Waseda University,
3-14-9, Okubo, Shinjuku-ku, Tokyo, Japan
arief.suryadi@akane.waseda.jp, hiroshi.watanabe@waseda.jp

Junichi Hara
RICOH Company, Ltd.,
810 Shimo-Imaizumi, Ebina, Kanagawa, 243-0435, Japan
wave@nts.ricoh.co.jp

***Abstract*—Obtaining 3D scene flow from sequential images captured by fisheye camera is still challenging to be developed according to today's broad applications. In this research, we focus on generating 3D scene flow directly from sequential fisheye images captured by a binocular stereo camera setup. We adopt Lucas and Kanade's approach to calculate optical flow and stereo image disparity for obtaining the scene flow. The result shows that this method is very promising to produce the 3D motion vectors with respect to certain criteria of captured object movement.**

Keywords— scene flow, optical flow, disparity, fisheye, stereo, Lucas and Kanade

I. INTRODUCTION

Research on the omnidirectional vision has rapidly grown by the interest in many applications. Such application are robotics, autonomous navigation system, localization and mapping, surveillance system and video communication system. Wide field-of-view can be obtained efficiently by a single sensor, which is available today. The wide view used to be created from multiple limited views captured at the same time by using arranged general perspective camera [1],[2].

One of challenges to applying omnidirectional vision is to generate the 3D scene flow. This is usually used for the above applications especially for estimating motion in the real world. The state-of-the-art image processing along with sophisticated computer vision technique today has seemed to be achieving remarkable performance in estimating the 3D scene flow from the 2D general perspective images [3],[4]. However, this achievement might not benefit directly for serving omnidirectional images that usually suffer from the radial and tangential distortions due to the omnidirectional sensor characteristic. As a result, there is still a room for developing estimation of 3D scene flow from omnidirectional images according to many specific applications with certain constraints.

In this paper, we focus on the estimation of 3D scene flow directly from sequential fisheye images. These images are obtained from a binocular stereo fisheye camera setup. The proposed method for obtaining the 3D scene flow incorporates optical flow and image disparity calculations.

II. RELATED WORKS

The idea of obtaining 3D scene flow is likely to be started by [6]. This approach actually gives large opportunity for obtaining 3D scene with or without information continuously from real 3D surface. That last condition will make generating the 3D scene flow more efficient since we can depend on 2D information greatly, although in the same time we will face greater noise that makes the 3D scene flow seems to be unreliable. From then on, combining information from 2D optical flow and stereo disparity to obtain the scene flow is developed by [5]. This becomes a basic knowledge for developing the 3D scene flow in this research, although we use this for serving sequential fisheye images. Furthermore, we use a similar method for calculating either optical flow or disparity. We prefer to focus on making use of Lucas and Kanade's method for these calculations.

III. SCENE FLOW ESTIMATION

In this research, 3D scene flow is estimated by making utilization of optical flow and disparity calculations. Fig. 1, illustrates the 3D scene flow calculation framework. While the calculation of optical flow is obtained directly from two successive images captured by each fisheye camera, the calculation of disparity can be got straightly from two adjacent images captured by the stereo fisheye camera rig.

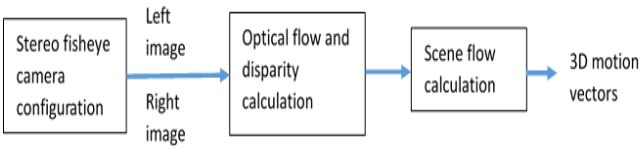

Fig. 1. The scene flow framework

Either the optical flow or disparity calculation is defined as problems of finding corresponding points in two images. This means that by assuming the pixel intensity remains at the same level, a pixel in the first image, $I(x,y,\beta)$, will be the same as the one captured in the second image, $I(x+u,y+v,\beta)$, although it could be located at difference location because of displacement (u,v). By using the first-order Tylor Series, therefore, linearization of the two pixels correspondence can result (1) that also valid for entire pixels in the image.

$$\nabla I.\vec{\mathbf{v}} + I_{\beta} = 0 \qquad (1)$$

Note that ∇I is spatial gradient of the first image, which is composed of horizontal and vertical image gradient (I_x and I_y subsequently), I_β is change in image that can be induced either by time $(I_t=I_2-I_1)$ or view $(I_v=I_{left}-I_{right})$, and the last is velocity vectors ($\vec{\mathbf{v}}$) we want to know, which consists of horizontal (u) and vertical (v) direction respectively. Equation (1) is now an under-determined linear equation since there are two unknown parameters in a linear equation. To overcome this, we adopt Lucas and Kanade's approach in term of finding solution for

978-1-5386-2615-3/18/$31.00 ©2018 IEEE

optical flow case [7]. They assumed that optical flow in a small neighborhood of a pixel remains constantly, hence minimizing sum of quadratic (1) in respect to $\vec{\mathbf{v}}$ is able to do in a predefined neighborhood of each pixel. Moreover, the final solution can be obtained by applying the least square approach to yield (2). Practically, we incorporate *n x n* neighborhood pixels with additional Gaussian filter to obtain the velocity vectors of each pixel. In addition, (2) is also applicable for disparity calculation except I_t now come from left and right images.

$$\begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix} \tag{2}$$

In similar fashion to optical flow, which is defined as the two-dimensional motion of objects or pixels on an image plane, scene flow is defined as the three-dimensional motion field of objects or points in the real world. Simple projection of scene flow onto an image plane is optical flow [6]. This relation can be explained as follow. Let us consider a 3D point of **A** *(X,Y,Z)* is located in the real world. The projection of this point onto an image coordinate $\mathbf{a}_n$ (x_n,y_n) of camera *n* is given by (3) and (4), in which $[P_i]_j$ is j^{th} row of the projection matrix P_i.

$$x_n = \frac{[P_i]_1(X,Y,Z,1)^T}{[P_i]_3(X,Y,Z,1)^T} \tag{3}$$

$$y_n = \frac{[P_i]_2(X,Y,Z,1)^T}{[P_i]_3(X,Y,Z,1)^T} \tag{4}$$

By assuming that the cameras is not moving, the relation between the two-dimensional motion ($\vec{\mathbf{v}} = d\mathbf{a}_n/dt$) and the scene flow ($\vec{\mathbf{v}} = d\mathbf{A}/dt$) can be expressed as (5).

$$\frac{d\mathbf{a}_n}{dt} = \frac{\partial \mathbf{a}_n}{\partial \mathbf{A}} \frac{d\mathbf{A}}{dt} \tag{5}$$

According to (5), the scene flow then can be calculated by at least two cameras. Therefore, it should be composed as a linear equation *VB = U*, and it can be expressed completely by (6).

The 3D coordinate **A** in binocular stereo setup has a relation with disparity *(d)* and corresponding image coordinates (x_{Right} and y_{Right} for right view, while x_{Left} and y_{Left} for left view), and their relationship is defined by (7). T denotes as baseline (distance between the two cameras), while *f* denotes as focal length of the cameras (assume that the two cameras have similar focal length, *f = 1*).

$$B = \begin{bmatrix} \frac{\partial x_1}{\partial X} & \frac{\partial x_1}{\partial Y} & \frac{\partial x_1}{\partial Z} \\ \frac{\partial y_1}{\partial X} & \frac{\partial y_1}{\partial Y} & \frac{\partial y_1}{\partial Z} \\ \frac{\partial x_2}{\partial X} & \frac{\partial x_2}{\partial Y} & \frac{\partial x_2}{\partial Z} \\ \frac{\partial y_2}{\partial X} & \frac{\partial y_2}{\partial Y} & \frac{\partial y_2}{\partial Z} \end{bmatrix}, U = \begin{bmatrix} \frac{\partial x_1}{\partial t} \\ \frac{\partial y_1}{\partial t} \\ \frac{\partial x_2}{\partial t} \\ \frac{\partial y_2}{\partial t} \end{bmatrix} \tag{6}$$

$$X = \frac{T(x_{Right} + x_{Left})}{2d}, Y = \frac{T(y_{Right} + y_{Left})}{2d}, Z = \frac{fT}{d} \tag{7}$$

IV. EXPERIMENTAL

In this experiment, we make sequential fisheye images synthetically by using Blender [8], as it is shown by Fig. 2. We firstly construct a binocular stereo camera located in front of a cube. The distance between the two cameras (baseline) is set to 4 cm. While the edge of cube is set to 10 cm, the distance between the center of the cube and the baseline is set to 20 cm.

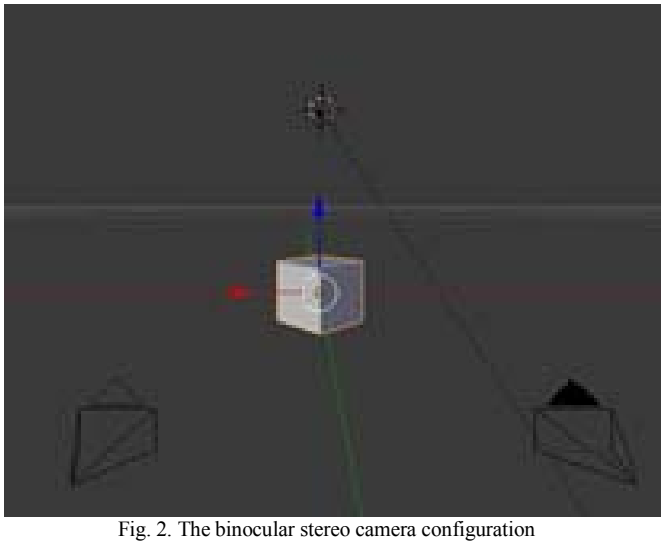

Fig. 2. The binocular stereo camera configuration

The cube is then moved 1, 2 or 3 cm to the left as an assumption that it is a small uniform movement. Each camera followed this condition, therefore we have sequential images showing the cube starting from the original position to the following one cm movements (e.g., first image presents the cube at original position, the second image presents the cube at one cm movement, third image presents the cube at 2 cm movement, etc.).

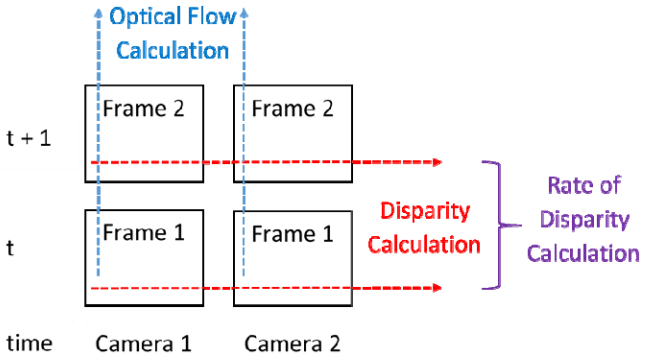

Fig. 3. Scheme of optical flow and disparity calculations

The scheme of calculation starts from finding optical flow and disparity as soon as the images are captured. At least we need two successive images from each camera for the optical flow calculation and two stereo pairs of images for the disparity and rate of disparity calculation, as they can be seen in Fig. 3. All of the calculation process are written into a design of algorithm, as we can see in TABLE I, and it is developed by using Matlab [9].

Having been calculated, the optical flow, disparity and rate of disparity are then finally used for obtaining scene flow through (5). As a result, from the sequential images captured by the stereo camera at *t* and *t+1* respectively, which can be seen from Fig. 4, we can generate the optical flow, as it is

presented by Fig. 5, the disparity, which is displayed by Fig. 6, and the scene flow, which is displayed by Fig.7. These figures are presented for 2 cm movement.

Since the images input consists of simple contour and brightness, it is easy to identify changes caused by movement around edge areas of the cube. Hence, either the optical flow or the disparity can be generated very well. This condition is shown visually by Fig. 5 and 6. However, it is difficult to generate them at an edge that have similar brightness like at the front side of the cube. As a result, the 3D scene flow can be obtained only at the areas that the optical flow and disparity occur. This can be seen in Fig. 7. In addition, Fig. 8 presents once again the 3D scene flow when the object movement is 1 cm.

The quantitative measurements shown by TABLE II, inform that the average angular error (AAE) and the standard deviation average angular error (SAE) are almost lower than 0.4^{o} and 4.1^{o} respectively, although the movement of the cube varies from 1 cm to 3 cm. This condition is also happened for every changes of distance between the cube and the baseline. However, these numbers can be obtained since the calculation only includes the existing 3D vectors with the corresponding 3D vectors reference.

I. CONCLUSION

Generating 3D scene flow directly from sequential fisheye images can be done by knowing optical flow and disparity. The system can present scene flow vectors around areas of object that have different brightness such as the edge of cube. In the areas that have almost similar brightness, however, scene flow seems to be difficult to be generated. The error of the estimated 3D scene flow is relatively low. This means that the use of Lucas and Kanade's approach for obtaining the 3D scene flow has produced a potential result. Nevertheless, the next problem to be solved is to increase the number of 3D vectors, and all of the system should be applied for serving real fisheye images.

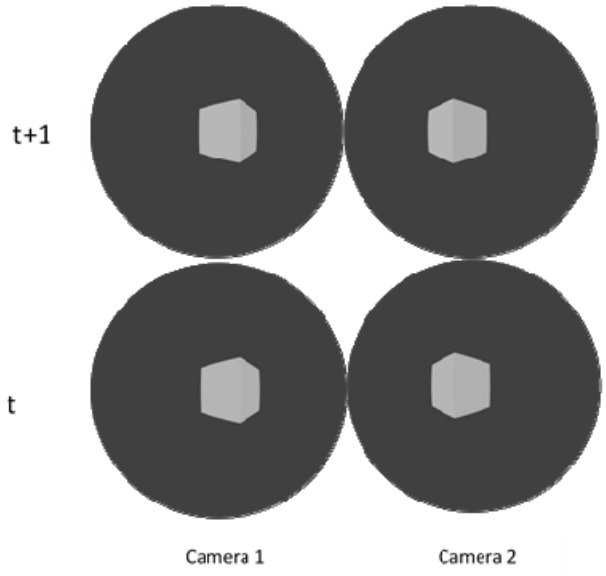

Fig. 4. Sequential images captured by the stereo camera at *t* and *t+1*

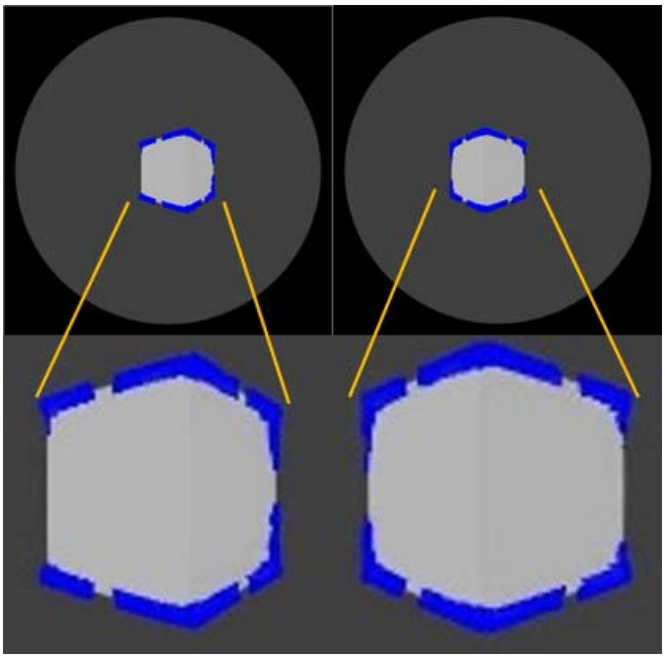

Fig. 5. Optical flow for left (from camera 1) and right (from camera 2) image

TABLE I. Design of Algorithm

Algorithm for calculating 3D scene flow
Input: two successive fish-eye images from either left or right camera
Calculating optical flow
1: ***for*** *each two successive images,* ***do***
2: $im1_{st}$ & $im2_{nd}$ ← *converting the two images to grayscale*
3: $im1_{st}$ & $im2_{nd}$ ← *smoothing the two grayscale images by using*
4: *Gaussian smooth filter*
5: I_x, I_y & I_t ← *obtaining spatial and temporal gradient of the images*
6: *u & v ← calculating vector flows by using matrix equation 2.*
7: ***end for***
8: ***Return***
Input: two stereo fish-eye images from either t=1 or t=2
Calculating disparity
1: ***for*** *each two successive images,* ***do***
2: im_{Left} & im_{Right} ← *converting the two images to grayscale*
3: im_{Left} & im_{Right} ← *smoothing the two grayscale images by using*
4: *Gaussian smooth filter*
5: I_x, I_y & I_t (im_{Left} & im_{Right}) ← *obtaining spatial and stereo gradient of the images*
6: *d ← calculating disparity by using equation 2 and calculating rate of disparity*
7: ***end for***
8: ***Return***
Input : optical flow and disparity
1: *Calculating equation 5*
2: *Displaying 3D scene flow*
3: *Calculating AAE and SAE*
4: ***Return***

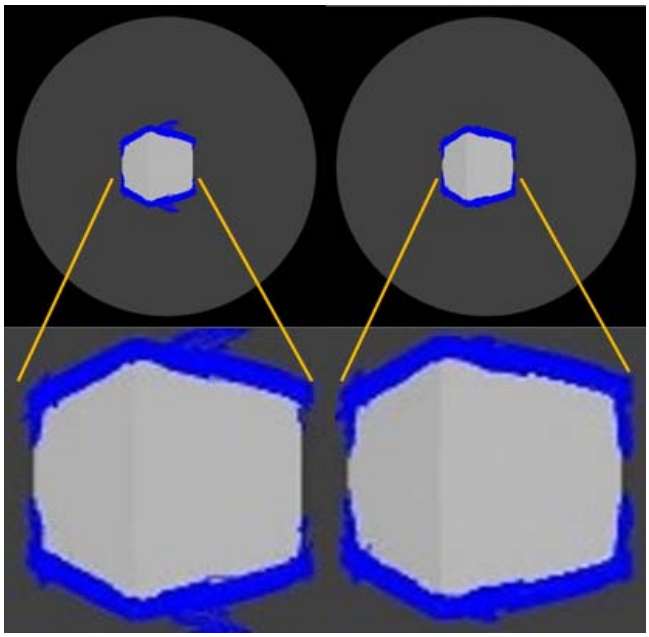

Fig. 6. Disparity image at t (left) and $t + 1$ (right)

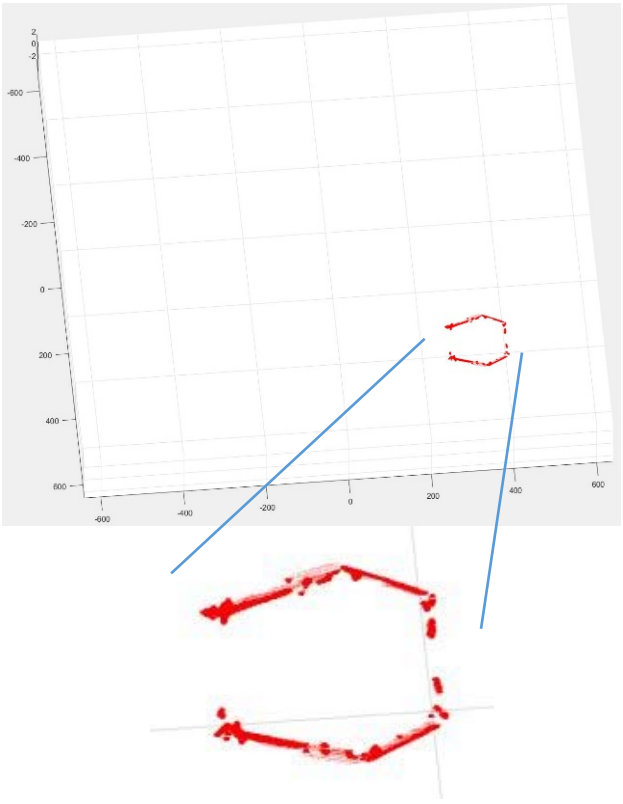

Fig. 7. The 3D scene flow for 2 cm movement

TABLE II. Quantitative measurement (AAE and SAE)

Object Movements in cm	AAE (Average Angular Error in degree)	SAE (Standard Deviation of Angular Error in degree)
The distance between the cube and the baseline 15 cm		
1	0.228	2.5774
2	0.3434	3.5325
3	0.3919	4.0677
The distance between the cube and the baseline 20 cm		
1	0.1913	2.3515
2	0.2738	3.2232
3	0.3196	3.7414
The distance between the cube and the baseline 25 cm		
1	0.1514	2.1622
2	0.193	2.5129
3	0.2176	2.8568

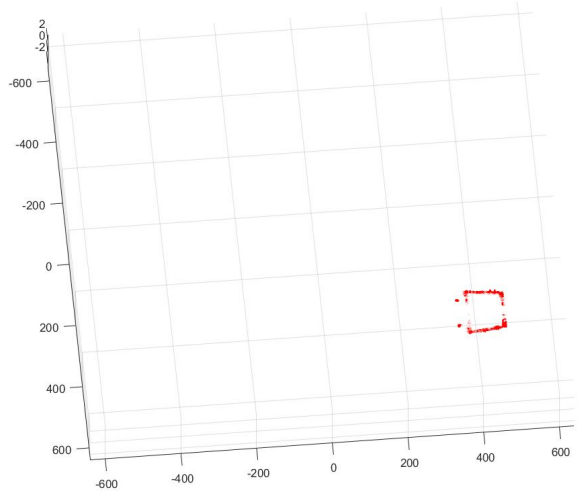

Fig. 8. The 3D Scene flow for 1 cm movement

REFERENCES

[1] http://www.cis.upenn.edu/~kostas/omni.html, accessed on 1 April 2017
[2] https://theta360.com, accessed on 1 April 2017
[3] Z. Yan and X. Xiang, "Scene Flow Estimation: A Survey", arXiv preprint arXiv:1612.02590 (2016).
[4] A.Wedel and D. Cremers, "Stereo Scene Flow for 3D Motion Analysis", DOI 10.1007/978-0-85729-965-9_2, Springer-Verlag 2011.
[5] R. Li and S. Sclaroff, "Multi-scale 3D scene flow from binocular stereo sequences", Computer Vision and Image Understanding, Vol.110, No.1, pp.75-90, Apr. 2008.
[6] S. Vedula, P. Rander, R. Collins, and T. Kanade, "Three-dimensional scene flow", IEEE Transactions on Pattern Analysis and Machine Intelligence,Vol. 27, No.3, Mar. 2005.
[7] S. Baker and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework", International Journal of Computer Vision, Vol.56, No.3, pp.221–255, 2004.
[8] Blender, https://www.blender.org/, accessed on 1 April 2017
[9] Matlab, https://www.mathworks, accessed on 1 April 2017