IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Instrumentation Mesure Métrologie
Vol. 20, No. 2, April, 2021, pp. 73-78
Journal homepage: http://iieta.org/journals/i2m

Generation of Stereo Images from the Heterogeneous Cameras

SeongKi Kim

Division of SW Convergence, Sangmyung University, Seoul 03016, South Korea

Corresponding Author Email: skkim9226@gmail.com

https://doi.org/10.18280/i2m.200202

Received: 15 January 2021
Accepted: 25 March 2021

Keywords:
heterogeneous camera, stereo images, mobile devices, calibration

ABSTRACT

As mobile devices with multiple cameras have been widespread, the cameras can be used for generating stereo images for depth sensing. However, it is difficult to use the raw images from the cameras because the cameras have different characteristics such as resolutions, distortions, and field of view in many cases. In this paper, we suggest methods to generate the stereo images from the heterogeneous cameras within 0.329 seconds and verify the methods. Although we use the images from a smartphone to generate the stereo images, the suggested methods can be used for any other devices with heterogeneous cameras. To the best of our knowledge, the suggested methods are the world-first one that generates the stereo images from the images by real mobile devices.

1. INTRODUCTION

Over the last decade, mobile devices such as smartphones have been widespread, and the performance of hardware has been significantly improved. For example, the performance of mobile CPU has increased 193.61 times from 2010 to 2019 (Nexus One: 1,279, Pixel 4 XL: 247,622, CPUMark [1]), and the performance of mobile GPU has increased 18.17 times during the same period (Nexus One: 360, Pixel 4 XL: 6,540, 3D Graphics Mark [1]). Also, the resolutions of a display and a camera have respectively increased 11.4 times (Nexus One: WVGA 480 × 800, Pixel 4 XL: QHD 1,440 × 3,040), and 3.2 times (Nexus One: 5 MPixels, Pixel 4 XL: 16 MPixels). These improvements of mobile hardware have enabled real-time rendering of complex scenes with multiple light sources and many 3D meshes.

Besides the advancement of mobile hardware, mobile-based Head-Mounted Displays (HMD) such as Cardboard, Daydream, and Gear VR appeared at an affordable price. As a result, mobile Virtual Reality (VR) has been used in the fields such as education, healthcare, tourism, simulation, and games because it can be conveniently used without complex video tracking equipment and wires. In addition to these mobile VR, Augmented Reality (AR), which uses mobile GPU to render virtual objects on top of images from a camera, is also used in various fields such as education, marketing, and games.

When playing the VR/ AR contents, it is essential to render virtual objects on top of the surrounding environments, and the virtual objects need to respond appropriately to changes in the environments. For this purpose, the depth information should be obtained from the camera and the 3D models should be reconstructed in real-time. Besides the AR, the depth information is important for gesture recognition, autonomous drone, robot, and car systems.

To get the depth information, Time of Flight (ToF) [2], Structured Light (SL) [3], Stereo Vision [4], and Deep Neural Network (DNN) [5] can be used. ToF is used by Microsoft's Kinect and Intel's RealSense [6] and measures the speed at which signals are issues and returned to the camera. SL shoots infrared lights, which is invisible to the human eye, in a grid/ a horizon/ a point form and analyzes the observed light patterns to determine the depth of the scene. The stereo vision uses two or more pre-known cameras' locations for sensing the depth and extracts 3D information by matching the relative positions of objects in the images. DNN pre-trains various scenes and depth information in the network and uses the network to estimate depth maps with a single image.

All these methods have their advantages and disadvantages. For example, ToF and SL require special hardware for the purpose, but the stereo vision and the DNN only require the software and need additional processing times for the camera calibration and the estimation, respectively.

Meanwhile, various vendors have released mobile devices containing multiple cameras on the front and back sides. For example, some devices with four cameras on the back [7] have been released recently [8]. As a result, it became possible to use the cameras to sense the depth and finally reconstruct 3D models. However, it is difficult to directly use the existing stereo matching algorithms among the multiple cameras and the raw images because the cameras have different characteristics such as lens, resolutions, and Field of View (FoV). However, it can significantly improve the AR experience that the smartphones utilize the recently widespread multiple cameras for depth sensing.

For the depth sensing from these cameras, we researched and developed methods to generate stereo images from any heterogeneous cameras. The contributions of this paper can be summarized as follows. First, we described our methods to generate the stereo image from the cameras that have different characteristics. Second, we verify our methods with the images from real mobile devices, and successfully the image within 0.329 seconds.

This paper is organized as follows. Section 2 introduces the camera calibration as related works. Section 3 describes our methods to generate stereo images from heterogeneous cameras, and Section 4 evaluates the methods. Section 5 summarizes the results of the algorithms.

2. RELATED WORKS

This section describes the related works to understand this paper.

2.1 Stereo vision

A person receives two-dimensional images from the two distant eyes and recognizes three-dimensional objects. Similarly to these process, stereo vision uses two cameras that are displaced horizontally/ vertically from one another. After two images are obtained from the separate cameras, the pixels in them are compared, matching points are found, and the relative depth information is obtained in the form of a disparity map.

During the process, stereo images are taken from two cameras. To use the stereo image for depth sensing, at least two images are required. Both images should be located at the positions with the different x positions and the same y position.

Due to the similarity with the human recognition, the stereo vision has been widely researched. For example, Fehrman and McGough [9] use 16 webcams to build a camera array and achieves 17 frames per second. Kaczmarek [10] uses 5 web cameras for the robotic arm. However, the existing research assumes that the same cameras are used.

2.2 Heterogeneous camera

Recently, multiple cameras are integrated into a smartphone, and the cameras have different characteristics in many cases because a camera can supplement each other. For example, one of the back cameras is normal, but another back camera may have a wide-angle lens to capture more areas. It may have a telephoto lens to capture an image from a long distance. The cameras can also have different resolutions (e.g., 16 million pixels with a wide-angle lens, 12 million pixels with a normal angle) or different FoV. Due to these different characteristics, it is difficult to use the existing stereo-matching algorithm to match pixels within two images.

In this paper, we suggest methods to overcome these problems.

2.3 Camera calibration

The camera obtains images by projecting the lights from 3D space into a 2D plane. This transformation from a space to a plane can be modeled through the below Eq. (1):

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = ART \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (1)$$

In Eq. (1), (X, Y, Z) is the coordinate of a 3D point in the world coordinate system and (X', Y') is the coordinate of a 2D point on a 2D plane in the camera coordinate system. R and T are the rotation/ translation matrices to transform the world coordinate into the camera coordinate and are the extrinsic parameters to a camera. On the other hand, A represents the intrinsic parameters of a camera. The combination of A and RT is called the camera matrix or projection matrix and finally converts a 3D point into a 2D point.

The extrinsic parameters are related to the geometric relationship between the camera and the world, such as the camera's position, orientation (fan, tilt), and intrinsic parameters are related to the camera's characteristics such as focal length, principal point, and skew coefficient. Camera calibration is the process of finding the intrinsic parameter, A, in Eq. (1).

Because each camera has a different lens, the captured image from a camera can have a distortion. Two major kinds of distortion are radial and tangential distortions [11]. Radial distortion happens because the lens is spherical, and makes straight lines appear curved. Tangential distortion occurs because the image-taking lens is not aligned perfectly parallel to the imaging plane, so some areas in the image may look nearer than expected. These distortions can be modelled by focal length, principal point, and skew coefficient, which can be expressed by the matrix A in Eq. (1).

3. GENERATION OF THE STEREO IMAGES

This section describes the suggested methods to generate stereo images from the heterogeneous camera.

3.1 Undistorting image

Because each camera has its own distortion, the obtained images should be normalized first, and matrix A is calculated through the below Figure 1.

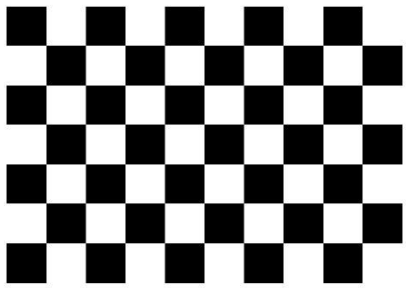

Figure 1. Image for the camera calibration

Figure 1 is used to get the intrinsic matrix A and extrinsic matrix of a camera for the normalization. Then, the calculated matrix is stored for future usage. Later, the obtained images from the camera are undistorted by the pre-stored matrix.

3.2 Matching feature points

In Subsection 3.1, our methods first calculate the intrinsic matrix, A, and the extrinsic of each camera to decrease the difference among cameras, then undistort the images from the cameras. After undistorting, the feature points need to be found from the camera and matched to find the magnification ratio of width and height. Figure 2 illustrates the examples of found points and matches.

In Figure 2, the feature points in both images are found/ matched and connected to each other by the green lines after undistorting by the intrinsic matrix, A.

These processes consist of finding feature points and finding matches between the found feature points. To find feature points, SIFT (Scale Invariant Feature Transformation) [12], SURF (Speeded Up Robust Features) [13] and ORB (Oriented and Rotated BRIEF) [14] algorithms can be used. Among these algorithms, we used the SIFT algorithm because it shows the best performance on mobile devices [15]. To match the feature points, the FLANN (Fast Library for

Approximate Nearest Neighbors) is used due to its performance [16].

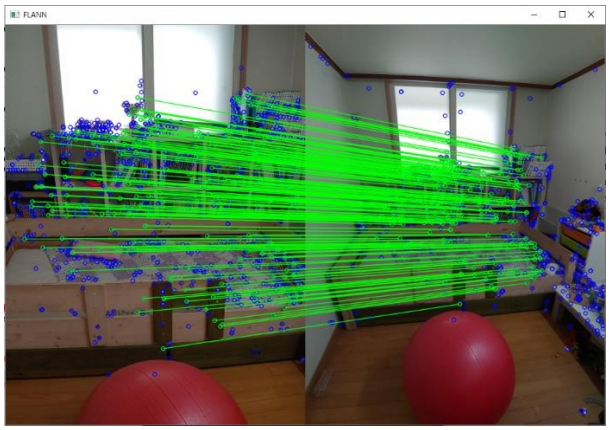

Figure 2. Found/ matched feature points from the calibrated images

3.3 Scaling image

Our methods try to match the FoVs of images through the obtained feature points. To do this, the more FoV image is scaled up to the less FoV image because more FoV image includes the point within less FoV image. To determine the upscale ratio, we find left/ right/ top/ bottom-most feature points as shown in Figure 3.

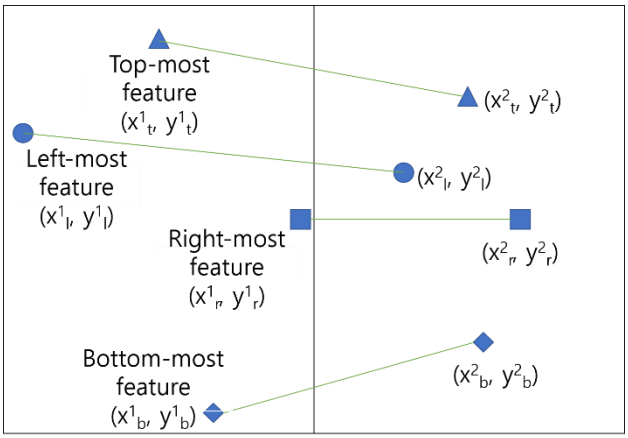

Figure 3. Matched feature points

In Figure 3, the left figure has less FoV, and the right figure has more one. Because the right figure includes the left one, our algorithm tries to scale up the right figure. To calculate the upscale ratio, the below Eq. (2) and (3) are used.

$$S_x = \frac{x_r^1 - x_l^1}{x_r^2 - x_l^2} \tag{2}$$

$$S_y = \frac{y_r^1 - y_l^1}{y_r^2 - y_l^2} \tag{3}$$

Eq. (2) calculates the ratio of the horizontal difference of a left figure between the left-most feature and the right-most feature to a right figure. Eq. (3) calculates the vertical ratio. Then, our methods scale up the right figure in Figure 2. As the result, the width/ height of the right figure in Figure 2 has the same ratio as the left figure.

3.4 Cropping image

In Subsection 3.3, the more FoV image is scaled up to the less FoV image, thus both images have the same ratio as shown in Figure 4. However, the magnified figure has a different width/ height from the left figure as shown in Figure 4.

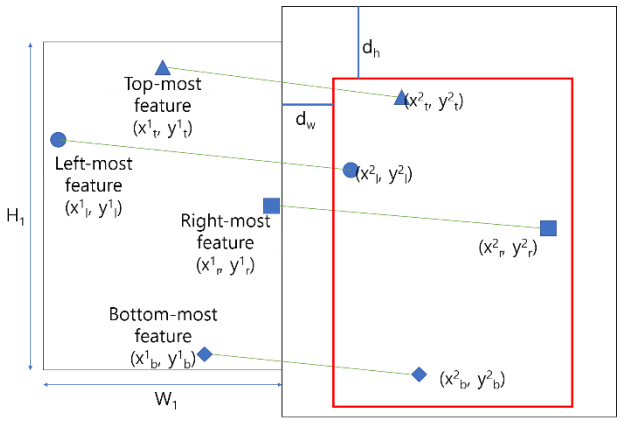

Figure 4. Scaled figures with different sizes

In Figure 4, the right figure shows the upscaled results by Eq. (2) and (3). The difference between the right most-feature and the left-most feature in the left figure has the same width as the one in the right figure because the right figure is magnified. The difference between the bottom-most feature and the top-most feature also has the same height. However, two images have different the width/ height.

Our goal is to generate the stereo images from heterogeneous cameras thus their sizes need to be the same. So, we calculate the d_w and d_h to crop the image in Figure 4 through the below Eq. (4) and (5).

$$d_w = x_l^2 * S_x - x_l^1 + diff_x \tag{4}$$

$$d_h = y_t^2 * S_y - y_t^1 + diff_y \tag{5}$$

In Eq. (4) and (5), the d_w and the d_h are starting positions of the magnified figure, and the S_x and the S_y are the upscaling ratios calculated by Eq. (2) and (3). The $diff_x$ and the $diff_y$ are the difference between the two cameras. Eq. (4) means that it moves the origin point of the *x*-axis within the up-scaled image to $diff_x$, and that of the *y*-axis to $diff_y$. After calculating the d_w and the d_h by Eq. (4) and (5), we crop the rectangle from (d_w, d_h) to (W_1, H_1).

As a result, both images have the same ratio and the same size, but the only difference is that the right figure moves to the camera difference.

4. RESULTS

To verify our methods described in Section 3, we captured images through LG's LM-V500N that has 3 different cameras at the backside. The cameras have 12 million-pixel resolutions with a telescopic lens, 16 million-pixel resolutions with an ultrawide-angle lens and 12 million-pixel resolutions with a normal lens. Among them, we captured 36 images from normal/ ultra-wide-angle cameras. We used 34 captured images for the camera calibration and used 4 images to verify our results. When running all of the processes in Section 3, we

used OpenCV 4.4.0 and Python 3.85 on a laptop with Intel's i5-8250U and 24 GB memory.

Figure 5 illustrates 2 images from the captured 34 ones as examples.

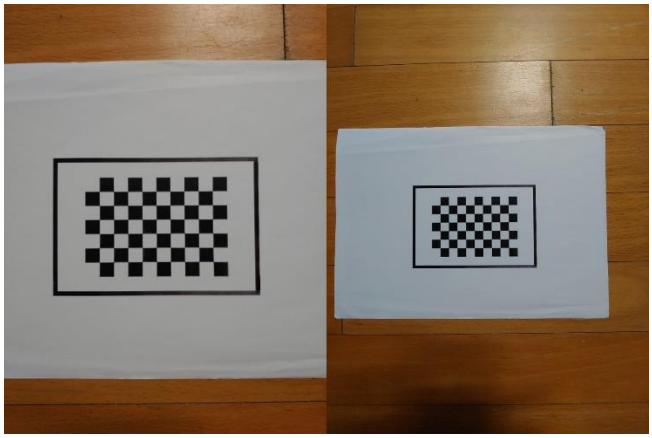

Figure 5. Capture images with different sizes by the heterogeneous cameras

Figure 5 illustrates the examples that are captured from normal/ ultra-wide-angle cameras, respectively.

After getting the calibration matrices of the normal/ ultra-wide-angle cameras from 34 captured images like the left/ right figures in Figure 5, we used 2 captured images as shown in Figure 6 to verify our methods.

Figure 6. Figures from different cameras

The left figure is from the normal camera, and the right figure is from the ultra-wide-angle camera. After applying the calibration matrix, the figures are undistorted, and we can get the figures in Figure 7.

Figure 7 looks like Figure 6, but a slight difference can be found when the images are enlarged. Especially, the bookshelf in Figure 6 is tilted, but that in Figure 7 is less tilted.

After undistorting the images through the intrinsic matrix as shown in Figure 7, our methods tried to find the feature points with the SIFT algorithm. Then, our methods find the match between the normal and the ultra-wide-angle images with the FLANN library [16]. After matching the feature points, our methods enlarge the image from the ultra-wide-angle camera by Eq. (2) and (3) as described in Subsection 3.3. Then, the enlarged image is clipped by Eq. 4 and 5. We heuristically use 20 as the $diff_x$ and 0 as the $diff_y$. After all these procedures, we could successfully get the images shown in Figure 8.

Figure 7. Calibrated figures from the obtained matrix

Figure 8(a) shows the left/ right images captured from the normal/ ultra-wide-angle cameras, and Figure 8(b) illustrates the modified images by the suggested methods. The left image in Figure 8(b) is the undistorted result from the normal lens, and the right image in Figure 8(b) is the result that passed all the methods described in Section 3. At the right images within the Figure 8(b), the aliasing and the blurring effects are observed differently from the right image within the Figure 8(a). Also, the second/ fourth right images within the Figure 8(b) also shows the light effects differently from the left ones because two lenses are different. If the SIFT algorithm fails to find the feature points or the FLANN fails to find the matches between the feature points, our methods can fail to generate the stereo image.

(a) (b)

Figure 8. Generated stereo images from the heterogeneous cameras: (a) are original ones from the normal/ ultra-wide-angle lens (b) are from our methods

We also measured the performance of these processes 10 times and averaged the results in Table 1.

Table 1. Elapsed time in Seconds

Process	Time in Seconds
Undistorting image	69.096
Finding feature points	0.246
Matching	0.068
Resizing	0.005
Cropping	0.010

Table 1 illustrates that most of the times are consumed when undistorting 34 images. When the undistorting time is excluded, 0.329 seconds are required to generate the right image from the ultra-wide-angle image. When we consider that the calibration matrix can be stored and reused, the required time is only 0.329 seconds.

5. CONCLUSION

As the recent smartphones include multiple cameras, they can be used for depth-sensing which is important to many applications such as augmented reality and gesture recognition. However, most of the cameras have heterogeneous characteristics, thus it is difficult to directly use the images from them. In this paper, we suggest methods that generate stereo images from any heterogeneous cameras. Our methods first generate an intrinsic matrix, find the feature points, matches them between 2 different images, resize the more FoV image, and finally clip the image. From these processes, we can successfully generate the right image within 0.329 seconds on average.

However, our methods have future works as follows. We do not implement it on real mobile devices in real-time. In the future, we plan to implement them on real mobile devices and verify the methods.

ACKNOWLEDGMENT

This research was funded by a 2020 research Grant (2020-A000-0329) from Sangmyung University. SeongKi Kim is the corresponding author.

REFERENCES

[1] Android Benchmarks. https://www.androidbenchmark.net/, accessed on Feb. 15, 2021.

[2] Kolb, A., Barth, E., Koch, R. (2008). ToF-sensors: New dimensions for realism and interactivity. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops, pp. 1-6. https://doi.org/10.1109/CVPRW.2008.4563159

[3] Fanello, S.R., Rhemann, C., Tankovich, V., Kowdle, A., Escolano, S.O., Kim, D., Izadi, S. (2016). HyperDepth: Learning depth from structured light without matching. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 5441-5450. https://doi.org/10.1109/CVPR.2016.587

[4] Hamzah, R.A., Ibrahim, H. (2015). Literature survey on stereo vision disparity map algorithms. Journal of Sensors, 2016: 8742920. https://doi.org/10.1155/2016/8742920

[5] Zhao, C., Sun, Q., Zhang, C., Tang, Y., Qian, F. (2020). Monocular depth estimation based on deep learning: An overview. Sci. China Technol. Sci., 63: 1612-1627. https://doi.org/10.1007/s11431-020-1582-8

[6] Intel RealSense. https://www.intelrealsense.com/, accessed on Feb. 15, 2021.

[7] MobileScout. Best smartphones with dual rear cameras. https://www.mobilescout.com/android/news/n69960/best-smartphones-dual-rear-cameras.html, accessed on Feb. 15, 2021.

[8] Galaxy Note10. https://www.samsung.com/sec/smartphones/galaxy-note10/, accessed on Feb. 15, 2021.

[9] Fehrman, B., McGough, J. (2014). Depth mapping using a low-cost camera array. 2014 Southwest Symposium on Image Analysis and Interpretation, San Diego, CA, USA, pp. 101-104. https://doi.org/10.1109/SSIAI.2014.6806039

[10] Kaczmarek, A.L. (2020). 3D vision system for a robotic arm based on equal baseline camera array. J Intell Robot Syst., 99: 13-28. https://doi.org/10.1007/s10846-019-01117-8

[11] Pierre, D., Julien, L. (2016). An exact formula for calculating inverse radial lens distortions. Sensors (Basel), 16(6): 807. https://doi.org/10.3390/s16060807

[12] Lowe, D.G. (1999). Object recognition from local scale-invariant features. Proceedings of the International Conference on Computer Vision, Kerkyra, Greece, pp. 1150-1157. https://doi.org/10.1109/ICCV.1999.790410

[13] Bay, H., Tuytelaars, T., Van Gool, L. (2006). SURF: Speeded Up Robust Features. In: Leonardis A., Bischof H., Pinz A. (eds) Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3951. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11744023_32

[14] Rublee, E., Rabaud, V., Konolige, K., Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. Proceedings of the International Conference on Computer Vision, Barcelona, Barcelona, Spain, pp. 2564-2571. https://doi.org/10.1109/ICCV.2011.6126544

[15] Jiwan, L., Jihoo, A., KiYong, L. (2018). Development of a raspberry Pi-based banknote recognition system for the visually impaired. The Journal of Society for e-Business Studies, 23(2): 21-31. https://doi.org/10.7838/jsebs.2018.23.2.021

[16] FLANN - Fast Library for Approximate Nearest Neighbors. https://web.archive.org/web/20150422010219/http://www.cs.ubc.ca/research/flann/, accessed on Feb. 15, 2021.