Real-Time Direct Dense Matching on Fisheye Images Using Plane-Sweeping Stereo

Christian Häne*, Lionel Heng, Gim Hee Lee†, Alexey Sizov, Marc Pollefeys

Department of Computer Science, ETH Zürich, Switzerland

Abstract

In this paper, we propose an adaptation of camera projection models for fisheye cameras into the plane-sweeping stereo matching algorithm. This adaptation allows us to do plane-sweeping stereo directly on fisheye images. Our approach also works for other non-pinhole cameras such as omnidirectional and catadioptric cameras when using the unified projection model. Despite the simplicity of our proposed approach, we are able to obtain full, good quality and high resolution depth maps from the fisheye images. To verify our approach, we show experimental results based on depth maps generated by our approach, and dense models produced from these depth maps.

1. Introduction

A camera is a low-cost and information-rich sensor that is suitable for many robotics applications such as obstacle detection and motion estimation. In these applications, a fisheye camera prevails over the conventional camera because it has a wider field-of-view that offers more coverage of the environment. Examples of successful use of fisheye cameras for robotics applications are Zingg $et.\ al$ [25] and Lee $et.\ al$ [14, 15], where fisheye cameras are used to perform optical flow for obstacle detection on a Micro-Aerial Vehicle (MAV) and motion estimation on a self-driving car. In these works, additional coverage provided by the fisheye cameras proved to be critical for the MAVs to gain maximum information of its surrounding for safe navigation, and increases the number of feature correspondences which improves motion estimation for the car. The use of fisheye cameras is not only limited to robotics applications. In recent years, we have seen an increasing number of mass-market cars such as the Nissan Qashqai and Honda Odyssey that are equipped with multiple fisheye cameras to provide drivers with a surround view for parking assistance. Another recent application of fisheye cameras is on hand-held mobile devices[1].

As compared to robotics and driving assistance applications, the use of fisheye cameras is limited in dense mapping applications probably due to the difficulty in handling the large distortion inherent in fisheye images for dense matching. Existing works in dense mapping such as the Urbanscape project [17] use multiple pinhole cameras to maximize coverage. The advantages of fisheye cameras for robotics applications, the convenience from the availability of these cameras on mass-market cars and the limited usage of fisheye cameras for dense mapping due to the large distortions motivate us to work on a simple yet effective model to produce depth maps from direct dense matching on fisheye images. We show that depth maps can be directly obtained from the fisheye camera by a simple adaptation of suitable fisheye camera models into the widely used plane-sweeping algorithm [4, 6]. The only requirement we need for the fisheye model is that both projection and unprojection can be done efficiently. In this work we use two different camera models, the unified projection model [8, 2, 16] and the field-of-view (FOV) model [5].

We choose to do plane-sweeping for dense matching because, in comparison to the rectification approach [18], the plane-sweeping approach can be done on more than two images. In addition, the plane-sweeping approach is particularly well-suited to Graphics Processing Unit (GPU) implementations for achieving real-time performance. We show experimentally that the alternative approach that first converts the fisheye images into pinhole images followed by plane-sweeping leads to a significant reduction in the coverage area. Our approach is only slightly slower than the standard plane sweep method, which uses planar homographies on the pinhole model, and thus still runs in real-time on a desktop computer. For mobile devices, high performance mobile GPUs have been released recently, for example the NVIDIA Tegra K1. Using devices equipped with such GPUs, there is a potential to reach interactive frame rates on mobile devices. Most importantly, we show that despite the simplicity of our proposed approach, we are able

*chaene@inf.ethz.ch

†Now at Mitsubishi Electric Research Laboratories

[1] www.google.com/atap/projecttango

to obtain full, good quality and high resolution depth maps from the fisheye images in arbitrary camera configurations without the need of prior rectification. It should be noted that the unified projection model [8, 2, 16], which we use for our approach also works for other non-pinhole cameras such as omnidirectional and catadioptric cameras.

In contrast to the existing works mentioned earlier in MAV navigation, motion estimation and sparse mapping for self-driving cars, which leveraged only sparse features from the fisheye images, our depth maps computed directly from the fisheye images provide more comprehensive information about the surrounding while retaining the advantages of fisheye cameras. We further demonstrate the potential of our approach by using the depth maps computed directly from the fisheye images to build dense maps of the environment. In comparison to the dense model built from the depth maps computed by the plane-sweeping algorithm on pinhole images [6] in the Urbanscape project [17], the dense model built from the depth maps which are computed directly from the fisheye images in our approach are shown to cover a larger part of the scene with fewer images and similar runtime performance.

1.1. Related Work

Rectified two-view and multi-view stereo matching are well established algorithms in computer vision. The main difference between these two methods is that the former works on only two rectified images which simplifies image matching along the aligned epiploar lines on the rectified images, while the latter works for any number of images. More details of the rectified two-view stereo can be found in [18]. In cases with more than two images, image matching along aligned epipolar lines is possible only in some special camera configurations. In addition, rectification fails when the epipoles fall within or too close to the borders of the images and this problem is exacerbated when wide field-of-view images are used as it is more probable that the motion direction lies inside the image boundaries. To overcome these problems, the plane-sweeping algorithm was introduced in [4]. This approach allows direct matching on multiple images without the need for rectification. Gallup $et.\ al$ [6] showed that the quality of the depth maps is improved by aligning the sweeping plane with the predominant orientations of the scene. In addition to the ease of matching images taken from arbitrary configurations, the plane-sweeping algorithm has the advantage of real-time performance which can be achieved by utilizing the texturing capabilities of GPUs [22, 6]. With such high performance stereo matching algorithms, dense reconstruction of whole cities becomes feasible [17].

While most of the works on stereo matching made use of pinhole images, limited works are shown on non-pinhole images such as omnidirectional, catadioptric or fisheye cameras. In [12, 24], Ishiguro $et.\ al$ and Zhu $et.\ al$ showed the first approaches on omnidirectional stereo vision that created a horizontal 360° panorama stereo pair by rotating a camera around a center point. In a later work [9], multiple catadioptric cameras were used to produce similar horizontal omnidirectional stereo pairs. These approaches, however, share the limitation that the field-of-view is large in only one direction. Catadioptric and fisheye cameras are able to capture images with an opening angle of 180° and beyond. One straightforward way to match the whole field-of-view of omnidirectional images would be to extract a set of pinhole images that cover the whole field of view. However, this leads to a significant increase in running time as many more images have to be matched. For specific tasks such as obstacle detection in automotive applications, [7] proposed to extract multiple pinhole images out of a single omnidirectional image for dense stereo matching only in important directions for obstacle detection. In [1], Abraham $et.\ al$ proposed an approach that rectifies fisheye images and preserves most of the field-of-view. However, such approaches do not generalize to arbitrary camera configurations.

2. Stereo Matching on Fisheye Images

In this section, we first introduce the generic notation for the camera projections that are suitable for our method. The details for the specific models can be found in [8, 2, 16] for the unified projection model and in [5] for the FOV model. Next, we describe in detail how the fisheye projection model is adapted into the plane-sweeping stereo matching algorithm.

2.1. Camera Projection Model

The camera projection model describes how a point is mapped to the image plane and its inverse function computes the corresponding ray in 3D space for a given image pixel. For our application, it is important that both of the functions can be evaluated efficiently.

Given a 3D point $\mathbf{X} = [X_x, X_y, X_z]^T$ in the camera frame F_C, the function $\mathbf{x}_u = \hbar(\xi, \mathbf{X})$ maps the 3D point $\mathbf{X}$ to an undistorted image point $\mathbf{x}$ on the normalized image plane. ξ is a single scalar parameter that models the fisheye lens. Imperfections of the camera lens cause additional radial and tangential distortions to the image. This transforms the normalized undistorted image coordinate $\mathbf{x}$ to the normalized distorted image coordinate $\mathbf{x}'$ according to the plumb blob model [3], which we denote as $\mathbf{x}' = D(\mathbf{x})$. Applying the camera projection matrix $\mathbf{K}$ on $\mathbf{x}'$, we get the image coordinate $(\tilde{x}', \tilde{y}')$ that we observed from an image captured with the fisheye camera. The intrinsic parameters $\boldsymbol{\phi} = [\xi,\ k_1,\ k_2,\ k_3,\ k_4,\ k_5,\ f_x,\ f_y,\ C_x,\ C_y]$ are obtained from the camera calibration. $[k_1,\ k_2,\ k_3,\ k_4,\ k_5]$ and $[f_x,\ f_y,\ C_x,\ C_y]$ are from the distortion model and

camera projection matrix respectively. We denote the unprojection, which maps a point $\mathbf{x}$ on the undistorted image plane to a ray by $\mathbf{X} = \hbar^{-1}(\xi, \mathbf{x})$.

The details of the individual camera models can be found in [16] and [5] for the unified projection model and the FOV model, respectively. For our work the important nature of these two camera models is that both the projection $\hbar$ and unprojection $\hbar^{-1}$ can be stated in closed form and hence be computed efficiently on the GPU. This is important as on the GPU a look up table is slower than the direct computation.

2.2. Plane-Sweeping for Fisheye Images

Plane-sweeping stereo matching [4] allows for dense stereo matching between multiple images without prior stereo rectification. Existing implementations such as [6] work for cameras that follow the standard pinhole model. We propose to extend the plane-sweeping stereo matching for fisheye cameras by incorporating the fisheye projection model directly into the plane-sweeping stereo matching algorithm. We closely follow [6] in the description of our algorithm.

The inputs to our plane-sweeping algorithm for fisheye cameras are a set of fisheye images, $\mathcal{I}' = \{I'_1, \cdots, I'_n, \cdots, I'_N\}$, and the camera parameters, $\mathcal{P} = \{P_1, \cdots, P_n, \cdots, P_N\}$ associated with the respective images. Each camera parameter $P_n = \{\boldsymbol{\phi}_n, \mathbf{R}_n, \mathbf{C}_n\}$ is made up of the camera intrinsics $\boldsymbol{\phi}_n$ and pose $[\mathbf{R}_n, \mathbf{C}_n]$. To keep our plane-sweeping algorithm for fisheye cameras simple and avoid a costly look up table on the GPU, we do a pre-processing step to remove the tangential and radial distortions from the fisheye images. Each pixel $[\tilde{x}, \tilde{y}, 1]^T$ on the set of undistorted images $\mathcal{I} = \{I_1, \cdots, I_n, \cdots, I_N\}$ is mapped onto $\mathcal{I}'$ by

$$[\tilde{x}', \tilde{y}', 1]^T = \mathbf{K} D\left(\mathbf{K}^{-1}[\tilde{x}, \tilde{y}, 1]^T\right) \tag{1}$$

This allows us to efficiently do a look-up for the corresponding pixel $[\tilde{x}', \tilde{y}', 1]^T$ in the fisheye image for each pixel $[\tilde{x}, \tilde{y}, 1]^T$ in the undistorted image $\mathcal{I}$.

Let us choose $I_{\text{ref}} \in \mathcal{I}$ as a reference view for the computation of a depth map. Additionally, we define a set of plane hypotheses $\Pi = \{\boldsymbol{\Pi}_1, \cdots, \boldsymbol{\Pi}_m, \cdots, \boldsymbol{\Pi}_M\}$. The main idea of the plane-sweeping algorithm is to use these planes Π as the hypothetical local reconstructions of the scene, where we choose the best plane for each pixel by evaluating an image dissimilarity measure. Each plane hypothesis is defined as $\boldsymbol{\Pi}_m = [\mathbf{n}_m^T, d_m]$, where $\mathbf{n}_m$ is the unit length normal vector of the plane pointing towards the camera center and d_m the distance of the plane to the camera center. To evaluate a plane hypothesis $\boldsymbol{\Pi}_m$, the pixels of the image I_n are mapped onto the reference view by projecting the image I_n onto plane $\boldsymbol{\Pi}_m$ followed by rendering it onto the reference view P_{ref}, we call this $I^{\text{ref}}_{n,m}$. In projective geometry, such a planar mapping is described by a homography. Using the fisheye projection model described in the previous section, the mapping that directly uses the pixels $[\tilde{x}_n, \tilde{y}_n, 1]^T$ from the undistorted fisheye image is given by

$$\begin{aligned}\mathbf{H}^{\text{ref}}_{n,m} &= \mathbf{R}^T_n \mathbf{R}_{\text{ref}} + \frac{1}{d_m}(\mathbf{R}^T_n \mathbf{C}_n - \mathbf{R}^T_n \mathbf{C}_{\text{ref}})\mathbf{n}^T_m \\ [\tilde{x}_n, \tilde{y}_n, 1]^T &= \mathbf{K}_n \hbar(\mathbf{H}^{\text{ref}}_{n,m} \hbar^{-1}(\mathbf{K}^{-1}_{\text{ref}}[\tilde{x}_{\text{ref}}, \tilde{y}_{\text{ref}}, 1]^T))\end{aligned} \tag{2}$$

We now warp image I_n to an image $I^{\text{ref}}_{n,m}$ with Equation 2, which we compare against the reference image I_{ref}. In particular, the image dissimilarity is evaluated between all the warped $I^{\text{ref}}_{n,m}$ and reference I_{ref} image, $\forall n \neq \text{ref}$ and for all planes $\Pi = \{\boldsymbol{\Pi}_1, \cdots, \boldsymbol{\Pi}_m, \cdots, \boldsymbol{\Pi}_M\}$. In our experiments, we use the negative zero mean normalized cross correlation (ZNCC) over a correlation window $\mathcal{W}$ for the evaluation of image dissimilarity (see Equation 3).

Evaluating the ZNCC matching scores can be done in constant time independent of the window size by using integral images [21]. As memory access on GPUs is often slower than computing the required values, Stam [19] proposes the use of a box filter which maintains a running sum in the vertical direction and sums up the horizontal direction by summing up the values in shared memory. We utilize a sequence of five times box filtering to compute the ZNCC scores according to [21] leading to a running time that increases linearly with the matching window width.

2.3. Cost Aggregation

We use the aggregated cost $\mathcal{D}^{\text{ref}}_m(\tilde{x}, \tilde{y})$ in cases where more than one image gets matched to the reference image. There are different options to compute the aggregated cost, with the choice of taking occlusions into account [13]. For this work we used two of them:

1. All matching costs are averaged. For example, with $2k$ images, k images before and k after the reference image,

$$\mathcal{D}^{\text{ref}}_m(\tilde{x}, \tilde{y}) = \frac{1}{2k} \sum_{n=\text{ref}-k}^{n=\text{ref}+k} \mathcal{D}^{\text{ref}}_{n,m}(\tilde{x}, \tilde{y}), \quad n \neq ref \tag{4}$$

2. Average cost from best half of the sequence. This assumes that the images are taken in a sequence with a predominant motion direction. First, we take the average cost over the half sequences on both sides of the reference view. Next, the final matching cost is the minimum of the two averages.

$$\mathcal{D}^{\text{ref}}_m(\tilde{x}, \tilde{y}) = \frac{1}{k} \min\Big\{ \sum_{n=\text{ref}-k}^{n=\text{ref}-1} \mathcal{D}^{\text{ref}}_{n,m}(\tilde{x}, \tilde{y}), \sum_{n=\text{ref}+1}^{n=\text{ref}+k} \mathcal{D}^{\text{ref}}_{n,m}(\tilde{x}, \tilde{y}) \Big\} \tag{5}$$

$$\mathcal{D}^{\text{ref}}_{n,m}(\tilde{x},\tilde{y}) = \frac{-\sum_{(i,j)\in\mathcal{W}} \{I_{\text{ref}}(\tilde{x}+i,\tilde{y}+j) - \overline{I}_{\text{ref}}(\tilde{x},\tilde{y})\}\{I^{\text{ref}}_{n,m}(\tilde{x}+i,\tilde{y}+j) - \overline{I}^{\text{ref}}_{n,m}(\tilde{x},\tilde{y})\}}{\sqrt{\sum_{(i,j)\in\mathcal{W}} \{I_{\text{ref}}(\tilde{x}+i,\tilde{y}+j) - \overline{I}_{\text{ref}}(\tilde{x},\tilde{y})\}^2 \sum_{(i,j)\in\mathcal{W}} \{I^{\text{ref}}_{n,m}(\tilde{x}+i,\tilde{y}+j) - \overline{I}^{\text{ref}}_{n,m}(\tilde{x},\tilde{y})\}^2}} \tag{3}$$

$$\overline{I}(\tilde{x},\tilde{y}) = \frac{1}{|\mathcal{W}|}\sum_{(i,j)\in\mathcal{W}} I(\tilde{x}+i,\tilde{y}+j)$$

This option handles occlusion with the assumption that each half of the sequence sees a different view of the occlusion.

2.4. Depth Extraction and Sub-Pixel Interpolation

We adopt the winner-takes-all strategy to decide the final depth value for fast processing. In particular, the plane $\mathbf{\Pi}_m$ with the lowest aggregated matching cost for each pixel in the reference view is chosen.

$$\hat{m}(\tilde{x},\tilde{y}) = \underset{m}{\text{argmin}}\ \mathcal{D}^{\text{ref}}_m(\tilde{x},\tilde{y}) \tag{6}$$

Finally, the final depth Z is computed as

$$Z_m(\tilde{x},\tilde{y}) = -d_m\left(\mathbf{n}_m^T \hbar^{-1}\left(\mathbf{K}^{-1}_{\text{ref}}[\tilde{x},\tilde{y},1]^T\right)\right)^{-1} \tag{7}$$

with the fisheye model taken into account.

Depths are extracted at discrete plane positions using Equation 7 and this leads to non-smooth surfaces. We circumvent this problem with sub-pixel interpolation [20]. The idea is to look at the matching scores of the neighboring planes, fit an interpolation function through the matching scores and extract the depth at the extremal point of the interpolated matching cost. We fit a parabola using the scores of the two neighboring planes in our implementation.

3. Experimental Evaluation

3.1. Camera Setups

We use three different camera setups to experimentally validate the dense matching on fisheye images. The first camera setup consists of four CCD cameras with fisheye lenses that each capture a 185° field-of-view. These cameras provide 1280×800 images at 12.5 Hz, and are integrated into a Volkswagen Golf car in such a way that the camera setup provides a surround view with a minimal overlapping field of view between any two cameras. Similarly, the second camera setup comprises four CMOS cameras with 185° fisheye lenses. These cameras provide 754×480 images at 15 Hz, and are installed on an AscTec Firefly hexacopter. The first and second cameras are arranged in a forward-looking stereo configuration while the third and fourth cameras are arranged in a rear-looking stereo configuration. The third camera setup consists of a single fisheye lens which is installed on a mobile hand-held platform. For all camera setups, the cameras are hardware-synchronized. Where necessary, we use SLAM-based calibration to obtain both the camera intrinsics and extrinsics [11, 10].

The camera poses that are given to our plane-sweeping algorithm are computed using visual odometry combined with wheel odometry and inertial meaturements where available.

3.2. Run-Time Performance

In this section, we evaluate the performance of the fisheye plane sweep. We implemented both the standard pinhole plane sweep and our proposed fisheye extension using Nvidia CUDA. Our test platform is equipped with a GeForce GTX 680 GPU with 4GB of memory.

For all the experiments, we precomputed the poses and stored them to disk as a preprocessing step. In the actual experiment, we load the poses and the images from disk. As for many applications, the images do not need to be loaded from disk because they are directly captured by a camera. Hence, we do not include in our measurements the time spent on loading the images from disk. The reported times include the time to upload the images to the GPU, undistort them or extract pinhole images respectively, run the plane sweep, and download the depth map. As our images are taken in sequence, we only upload each image to the GPU once, and use the same image to compute multiple depth maps.

In the first experiment, we use images taken in a sequence from a moving car. The native resolution of the images is 1280×800. In the process of undistortion or pinhole image extraction, we downsample them by half such that we obtain an image resolution of 640×400. We match two consecutive images using 64 planes, ZNCC matching scores, no occlusion handling, and subpixel interpolation. Some example depth maps and the running times are shown in Fig. 1. For a 9×9 ZNCC matching window, the fisheye and pinhole versions generate depth maps at 59.8 Hz and 65.1 Hz respectively. The fisheye version has a 8.96% longer running time but we obtain depth maps that cover the entire field of view of the camera.

In the next experiment, we measure the performance using half-sequence occlusion handling. Again, we use

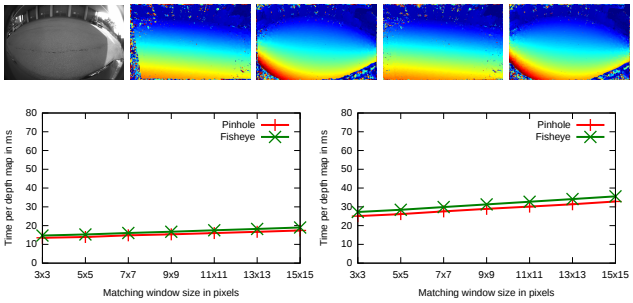

Figure 1. First row (from left to right): An input image, example pinhole and fisheye depth maps with 9×9 ZNCC for the two-view and three-view case. Second row (from left to right): Running times for the two-view and three-view cases averaged over 1667 and 1666 depth maps respectively.

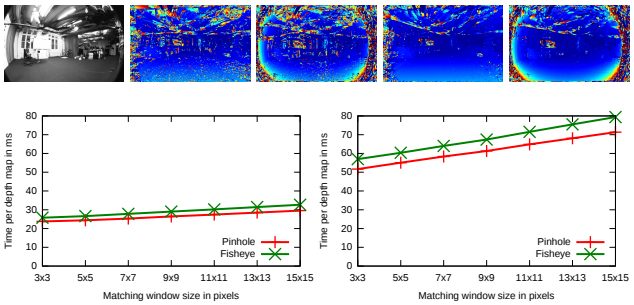

Figure 2. First row (from left to right): An input image, example pinhole and fisheye depth maps with 9x9 ZNCC for the two-view and four-view case. Second row (from left to right) Running times for the two-view and four-view cases averaged over 569 depth maps.

the half-resolution images with 64 planes, ZNCC matching scores, and subpixel interpolation. We match three images, choose the middle image as the reference image, and do half-sequence occlusion handling. The results can be seen in Fig. 1. For a 9×9 window, the fisheye and pinhole versions generate depth maps at 32.0 Hz and 34.6 Hz respectively. The fisheye version has a 8.15% longer running time.

We also use images from a MAV. The native resolution of the images is 754×480. We undistort the images or extract pinhole images with the same resolution. Each pair of cameras is set up in a stereo configuration. For each stereo camera, we use two images captured at the same time and match them. We use 96 planes, ZNCC matching scores, no occlusion handling, and subpixel interpolation. An example image and depth maps can be seen in Fig. 2. We notice that some noise in the depth map occurs due to some parts of the MAV being visible in the images. With a 9×9 ZNCC window, depth maps are generated at 37.8 Hz and 34.4 Hz for the pinhole and fisheye versions respectively. In this case, the running time for the fisheye version is longer by 9.8%.

In the last experiment, we use two stereo image pairs captured consecutively. Again, we use the full 754×480 resolution with 96 planes. We use ZNCC matching scores, no occlusion handling, and subpixel interpolation. Example depth maps and a plot of time measurements can be found in Fig 2. The performance for a 9×9 ZNCC window is 16.3 Hz for the pinhole version and 14.8 Hz for the fisheye version. Here, the running time for the fisheye version is longer by 9.9%.

All our measurements show that the increase in running time from using our proposed plane sweep with the unified projection model on fish eye images is below 10%. However, from running one plane sweep, we obtain a depth map that covers the entire field of view of the fisheye camera, and this advantage far outweighs the performance penalty.

3.3. Coverage

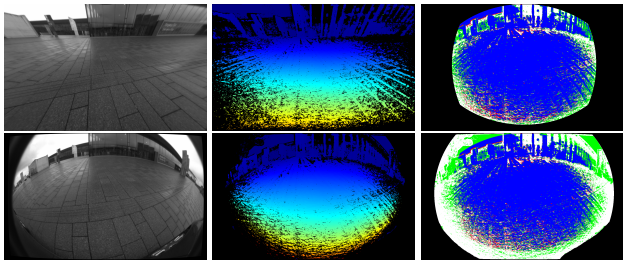

Figure 3. Coverage comparison of locally fused and filtered pinhole depth maps and fisheye depth maps. The first and second rows correspond to the pinhole and fisheye images respectively. The three columns show example input images, depth maps, and coverage images. In each coverage image, a blue pixel indicates the presence of a depth value for both depth maps, a red pixel indicates that a depth value is only present in the pinhole depth map, and a green pixel indicates that a depth value is only present in the fisheye depth map. The non-black area indicates the field of view of the camera. For the fisheye images, a mask was applied to mask out parts of the car visible in the images.

In order to quantitatively determine the increase in coverage obtained from fisheye images, that is, the additional information we gain with a bigger field of view, we conduct an experiment using images from the four cameras mounted on the car. In the first step, we run three-view stereo matching using half-sequence occlusion handling on full-resolution 1280×800 images, 128 planes, 11×11 ZNCC matching costs, and subpixel interpolation. We cannot directly compare the resulting depth maps because they contain a significant number of outliers. Therefore, we compute locally fused depth maps by using 11 raw depth maps and warping them into the middle view. To extract a high-quality depth map, we only accept depths in which at least 3 out of the 11 raw depth values per pixel agree. For each of the pinhole

	Avg coverage (pinhole)	Avg coverage (fisheye)
Pinhole area (front)	36.15%	45.38%
Pinhole area (left)	65.24%	73.07%
Pinhole area (rear)	32.21%	45.38%
Pinhole area (right)	73.22%	80.80 %
Fisheye area (front)	25.91%	37.48%
Fisheye area (left)	49.56%	61.56%
Fisheye area (rear)	24.17%	39.55%
Fisheye area (right)	54.38%	67.79%

Table 1. Average coverage of the depth maps over 200 frames.

and fisheye depth maps, we compute how many pixels have a depth estimate. To make the comparison consistent, we compare both depth maps in two ways; we look at both the portion of the depth maps visible in a pinhole image, and the portion of the depth maps visible in the fisheye image. Example depth maps are shown in Fig. 3, and the average coverage is given in Table 1. Here, we define the coverage to be the number of pixels associated with a depth value divided by the total number of pixels in the portion of the depth map to be compared (pinhole or fisheye area).

In a second experiment, we compute a dense point cloud reconstructed from 31 depth maps for both the fisheye and pinhole models. The resolution is 640×480 and each depth map is computed by locally fusing five depth maps and keeping the points that are consistent in at least three depth maps. Example input images and top-view renderings can be seen in Fig. 4. With the fisheye model, the ground is more complete, and the point cloud includes overhanging parts of buildings which are not visible in the point cloud corresponding to the pinhole model. Note that the camera is pointing roughly $45°$ towards the ground which makes reconstructing the elevated structures difficult. Close-up renderings of the point clouds are depicted in Fig. 5. We observe that when compared to the pinhole model, the truck on the left and the building facade on the right look more complete with the fisheye model.

3.4. Dense Modelling

One important use case of depth maps is to build dense 3D models. In this section, we compare the dense models we obtain from both pinhole depth maps and fisheye depth maps. To fuse the depth maps, we use in both cases the volumetric TV-Flux fusion of [23].

We use images from the MAV, and compute 20 depth maps using two consecutive stereo image pairs. From each fisheye image, we extract two different pinhole images; the first image has a horizontal field of view of $65°$ which corresponds to the field of view of standard pinhole cameras, and the second image has a wide field of view of $114.8°$. The depth maps are computed with a resolution of 754×480 pixels using 96 planes, 9×9 ZNCC matching scores, no occlusion handling, and subpixel interpolation. The resulting dense 3D models are visualized in Fig. 6. All the dense models are computed with the same settings. We observe that the $65°$ pinhole images are not able to see the ground or the ceiling, and thus, the reconstruction covers a small volume. Using the wide-angle $114.8°$ pinhole images, the reconstruction becomes more complete. The reconstruction is most complete with the fisheye images as we directly do dense matching on the fisheye images using our proposed matching algorithm.

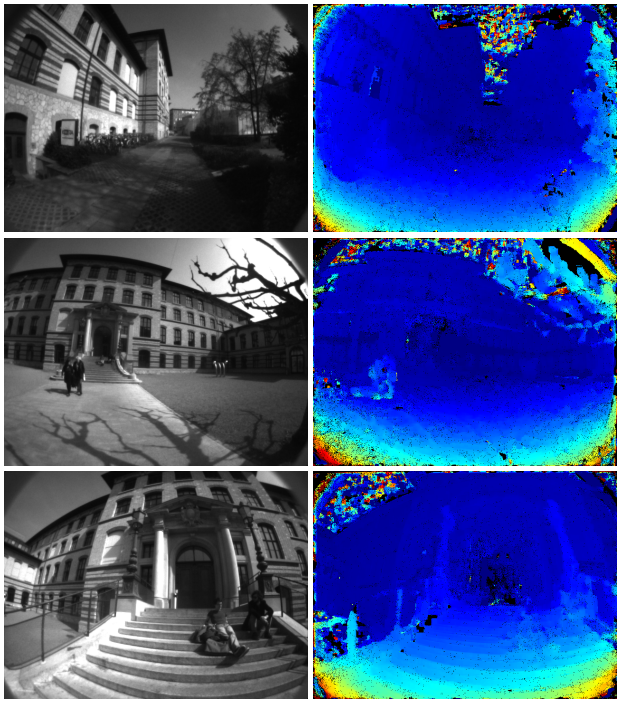

Figure 7. Results from the mobile device.

3.5. Mobile Devices

Another application of our algorithm is on hand-held mobile devices. We use the Google Tango phone[2]. This device contains a depth sensor. This sensor has a standard pinhole camera field-of-view and works indoors only. Our algorithm makes use of the fisheye camera from the device. It is able to compute depth information with a large field-of-view based on the motion of the device and the fisheye images. The camera poses are directly computed on the device in real-time. Our depth map computation is run offline on a desktop computer as a post processing step.

The settings of the plane sweep where as follows. We match three images where the last one is used as reference view. Occlusion handling is turned off and the image dissimilarity measure is an 11x11 ZNCC score. Some example depth maps are shown in figure 7.

[2]www.google.com/atap/projecttango

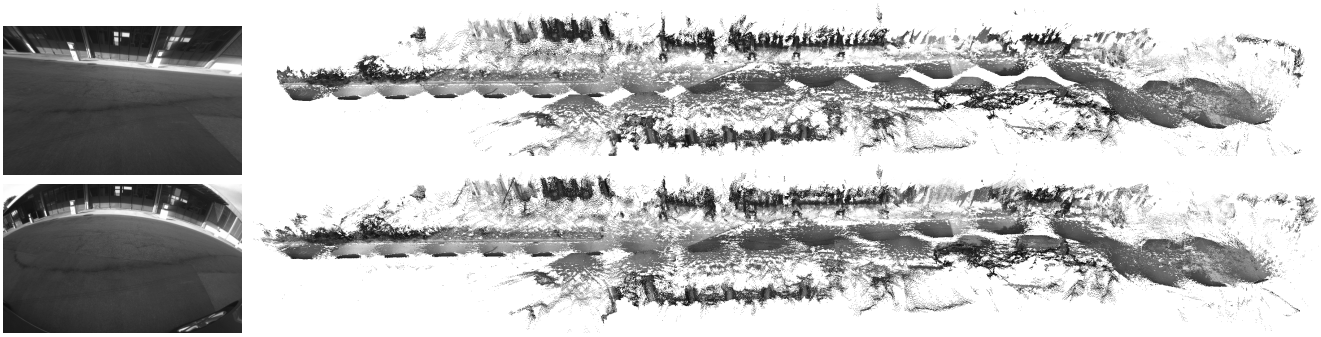

Figure 4. Top: An example pinhole image is shown on the left, and a top-view rendering of the generated point cloud corresponding to the pinhole model is shown on the right. Bottom: An example fisheye image is shown on the left, and a top-view rendering of the generated point cloud corresponding to the fisheye model is shown on the right.

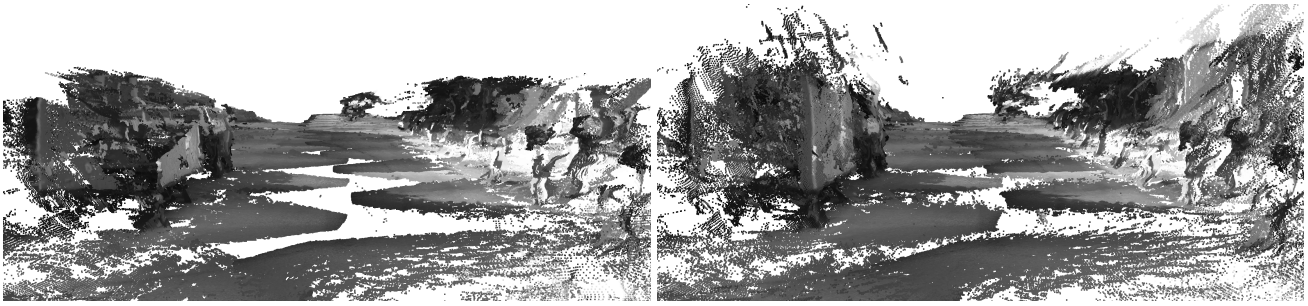

Figure 5. Close-up renderings of the generated point clouds. The left and right images correspond to the pinhole and fisheye models respectively.

4. Conclusions and Future Work

We have shown that our plane-sweeping stereo directly applied to fisheye images is capable of producing high-quality depth maps that cover the entire field-of-view of the fisheye camera. We make a simple adaptation to the widely-used plane-sweeping stereo algorithm for pinhole images by including the fisheye model such that our plane-sweeping stereo algorithm directly works with fisheye images without losing coverage. Our plane-sweeping stereo algorithm is well-suited for GPU implementations. Furthermore, our GPU implementation realizes real-time applications based on fisheye depth maps. In future work, we will implement our plane-sweeping stereo in a car equipped with a surround-view camera system, and generate a textured dense 3D model in real-time via volumetric fusion. In contrast to top-view composite 2D images that surround-view camera systems typically produce, a textured dense 3D model significantly improves the driver's situational awareness. Similarly, we will apply our plane- sweeping stereo to a robot platform equipped with a surround-view camera system such that the robot has an instantaneous 360° view of obstacles in the vicinity. Another future work is to implement the algorithm on a mobile GPU such as the NVIDIA Tegra K1.

Acknowledgements: We thank Torsten Sattler for providing the data sets from the mobile device. Furthermore we acknowledge the support of the V-Charge grant #269916 under the EC's FP7/2007-2013 and Google's Project Tango. Lionel Heng is funded by the DSO National Laboratories Postgraduate Scholarship.

References

[1] S. Abraham and W. Förstner. Fish-eye-stereo calibration and epipolar rectification. *Journal of photogrammetry and remote sensing (ISPRS)*, 2005. 2

[2] J. P. Barreto and H. Araujo. Issues on the geometry of central catadioptric image formation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. 1, 2

[3] D. C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 1971. 2

[4] R. T. Collins. A space-sweep approach to true multi-image matching. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1996. 1, 2, 3

[5] F. Devernay and O. Faugeras. Straight lines have to be straight. *Machine vision and applications*, 2001. 1, 2, 3

[6] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 1, 2, 3

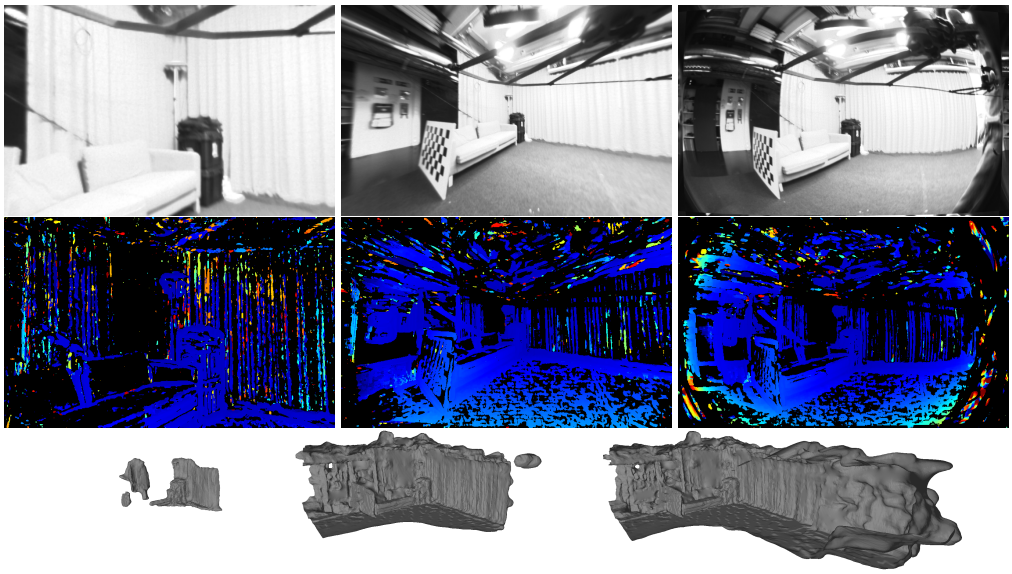

Figure 6. The first, second, and third columns correspond to the 65° pinhole model, 114.8° pinhole model, and the fisheye model respectively. The first row shows the input images, the second row shows example depth maps, and the third row shows the resulting dense 3D models.

[7] S. Gehrig. Large-field-of-view stereo for automotive applications. In *Proceedings of the 6th Workshop on Omnidirectional Vision (OMNIVIS)*, 2005. 2

[8] C. Geyer and K. Daniilidis. A unifying theory for central panoramic systems and practical implications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2000. 1, 2

[9] J. Gluckman, S. K. Nayar, and K. J. Thoresz. Real-time omnidirectional and panoramic stereo. In *Proceedings of the Image Understanding Workshop*, 1998. 2

[10] L. Heng, P. Furgale, and M. Pollefeys. Leveraging image-based localization for infrastructure-based calibration of a multi-camera rig. *Journal of Field Robotics (JFR)*, 2014. 4

[11] L. Heng, G. H. Lee, and M. Pollefeys. Self-calibration and visual slam with a multi-camera system on a micro aerial vehicle. In *Proceedings of Robotics: Science and Systems (RSS)*, 2014. 4

[12] H. Ishiguro, M. Yamamoto, and S. Tsuji. Omni-directional stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1992. 2

[13] S. B. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. 3

[14] G. H. Lee, F. Fraundorfer, and M. Pollefeys. Motion estimation for a self-driving car with a generalized camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 1

[15] G. H. Lee, F. Fraundorfer, and M. Pollefeys. Structureless pose-graph loop-closure with a multi-camera system on a self-driving car. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013. 1

[16] C. Mei and P. Rives. Single view point omnidirectional camera calibration from planar grids. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007. 1, 2, 3

[17] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, C. Salami, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision (IJCV)*, 2008. 1, 2

[18] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision (IJCV)*, 2002. 1, 2

[19] J. Stam. Stereo imaging with cuda. Technical report, NVIDIA Corporation, 2008. 3

[20] Q. Tian and M. N. Huhns. Algorithms for subpixel registration. *Computer Vision, Graphics and Image Processing*, 1986. 4

[21] D. Tsai and C. Lin. Fast normalized cross correlation for defect detection. *Pattern Recognition Letters*, 2003. 3

[22] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003. 2

[23] C. Zach. Fast and high quality fusion of depth maps. In *Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2008. 6

[24] Z. Zhu. Omnidirectional stereo vision. In *Proceedings of the Workshop on Omnidirectinal Vision (OMNIVIS)*, 2001. 2

[25] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart. Mav navigation through indoor corridors using optical flow. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010. 1