2010 IEEE Intelligent Vehicles Symposium
University of California, San Diego, CA, USA
June 21-24, 2010

Online Visualization of Noisy 3D Point Clouds: From Monocular Image Sequences to Synthetic Views

Frank Pagel and Jochen Ring
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (IOSB)
Autonomous Systems and Machine Vision
76131 Karlsruhe, Germany
{frank.pagel, jochen.ring}@iosb.fraunhofer.de

Abstract— **This paper addresses the generation and visualization of noisy 3D point clouds. The goal is to extract 3D information from image sequences in real-time and to transform the resulting noisy point clouds into a representation with dense surfaces and an intuitive character. Therefore a bridge is built from image sequence analysis to computer graphics. A point cloud is calculated from a single camera by using state of the art algorithms for egomotion estimation and dense 3D reconstruction. Due to inaccuracies in the measurement process and hence of the resulting depth maps, a degree of uncertainty of the 3D data is defined. This additional information can be used to filter the point cloud in space and time. We present an approach to filter and polygonize weighted point clouds and visualize it by mapping image texture on the resulting surface.**

I. Introduction

In this paper we discuss the question, how a synthetic view can be generated in real-time from a camera (or a set of cameras) so that the resulting view can be used by human operators for navigation purposes. There is a tendency to capture the complete environment of vehicles, e. g. for precrash applications, pedestrians or moving objects detection. The big advantage of generating virtual views is that the operator is no longer bound to the original camera views. So, gaps between the field of views can be closed continously by panning the virtual camera. This is especially then of eminent importance if the cameras cover a field of view that is not visible for the client, for example the rear view. Synthetic views could also be used in the field of robotics for remote navigations. The more cameras are connected the more it gets relevant to handle the incoming amount of image information. To illustrate the problem, just imagine a screen split into eight pieces that display the views of the eight cameras that are placed all around the car. A human operator might be overstrained with this amount of information.

To generate synthetic views of a virtual camera with an arbitrary point of view from an image sequence it is necessary to know the 3D structure of the observed scene. As 3D reconstruction with one or more cameras became a popular research area in the last decade, these techniques can be used to extract 3D information from the image stream. Unfortunately, depth maps are noisy in practice and in most cases a simple online-display of the 3D points would lead to even more confusion. Continuity of the 3D structure is an important point and a big problem, because small errors in motion estimation can cause big changes in the calculated point cloud between two consecutive time frames. The resolution and hence the accuracy of the point cloud depends on the velocity of the camera, too. So, a way must be found that handles gaps in the 3D data, noise and outliers of 3D points as well as the blurred 3D structure. This blurring effect could be caused by the mentioned inaccuracies in motion estimation and it could occur if point clouds from several sources are merged.

A. Related Work

In the field of egomotion a lot of research has been done over the last few years. Pollefeys [1] presented a stratified approach to metric self calibration. Although the approach seems to perform well it is very difficult to implement this algorithm on a standard PC hardware to run in real-time.

Another solution was shown by Davison [2]. The core of this approach, called MonoSLAM, is the online creation of a sparse but persistent map of natural landmarks within a probabilistic framework. The algorithm runs very fast with about 30Hz on standard PC hardware.

One of the most recent work in this research area is the preemptive 5-point solution of Nister [3]. Roughly spoken, egomotion estimation is reformulated into a problem of solving a 10th degree polynomial. As the algorithm is based on an estimation of the essential matrix with only five point correspondenes the algorithm is predestinated for robust preprocessing steps. The algorithm runs in real-time.

There are many algorithms and approaches to calculate 3D depth information from two calibrated images with known epipolar geometry. Dang et al. [4] used a block matching algorithm. Block matching is one of the most popular algorithm in commercial stereo systems because of its simplicity. Since the disparities are searched sequentially for each pixel or feature, these algorithms are highly error-prone to periodic patterns.

Collins [5] presented an approach especially designed for parallel processing architectures and multi camera stereo applications. A plane is simply swept through 3D space. By back-projecting the points on the swept plane onto the image plane a pixelwise error can be calculated. Hence, the depth resolution is restricted by the number of planes in space.

Birchfield and Tomasi [6] calculated a minimal cost path through a matrix where each entry relates to a distance

measure between a pixel in the left image and a pixel in the right image via dynamic programming. Boykov et al. [7] minimized an energy function to find the globally optimal assignment for each pixel. Both dynamic programming and energy minimization result in dense depth maps and they consider the image information linewise depending on the epipolar geometry.
The work of Hirschmüller [8] seems to be the most promising approach. It approximates a 2D global optimization by performing an energy minimization along several 1D paths across the image. This is also the algorithm that is used in this paper.
In our approach handling noisy 3D data is important, too. Some approaches solved this problem by a 3D point registration. Merrell et al. [9], for example, propose a solution to register point clouds in a multi camera system with overlapping field of views. They perform a two stage approach by first calculating potentially noisy, overlapping depth maps and then fusing these maps to yield a higher accuracy and less redundancy.
Zhao et al. [10] present an approach to align continuous video to 3D sensor data. They adjust a point cloud computed from the video onto a point cloud directly obtained from a 3D sensor. The registration via Iterative Closest Point techniques (ICP) requires a highly accurate point cloud.

When modeling the scene from single depth maps (e. g. from stereo cameras), depth can be considered as a function of the image coordinates $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ with $f(x, y) \mapsto Z$. Because the 3D structure is accumulated over time the function f cannot be used anymore to describe the scene. This is the reason why Delaunay-triangulation [11] is not applicable in this approach. As the point clouds are meant to be merged with 3D points generated from other cameras' data the influence of noise cannot be neglected. This makes the given task much more complicated.
Many visualization approaches perform upon structured grids. Marching Cubes (MC), as in [12], is a popular grid based visualization technique. MC visualizes 3D data in the sense of a divide and conquer algorithm using uniform and cubic grid cells. MC examines the values of the grid vertices and provides a lookup table which abstracts intersection cases of a surface and a grid cell. These vertex values are compared to a threshold called isolevel. Using the isolevel, 3D volumes could be separated with 3D isosurfaces by thresholding the volume's values.
The MC algorithm in [12] is only capable of producing sharp features like corners and edges if they conform to vertices and edges of the underlying grid. To overcome this drawback Schaefer et al. [13] generated a grid which is dual to an octree. The vertices of the dual grid are the real 3D features. This enables the modified MC to output surfaces containing sharp features. In [13] the dual grid is based on an octree quantization. Whereas volume quantization seems to be reasonable to our visualization algorithm, using an octree would be too expensive to meet real-time concern because of the noisy 3D data. So we decided to compute a grid which is dual to a homogeneous 3D space. We call this space *voxel space*. Unlike an octree, once constructed, the voxel space needs no further topology adjustment.

B. Proposed Solution

This paper only considers a monocular, calibrated image sequence (although the basic principles would work for calibrated multi-sensor applications, too). Our approach proceeds as follows:
The Euclidean egomotion of the camera is estimated using a robust iterated extended Kalman filter [14]. Therefore sparse optical flow fields must be extracted. Any other egomotion algorithm would basically work, too. Of course, the better the egomotion (which means the more precise the estimation over time) the more accurate are the point clouds in 3D space. One reason we chose this approach was the implicit calculation of a covariance of the motion parameters.
Once the transformation parameters between two frames are known, algorithms from the field of stereo vision are used and adapted to calculate a disparity respective a depth map. To quantize the uncertainty of the reconstructed 3D point cloud each pixel is weighted using a logarithmic function based on the costs for the disparities for that pixel. This step is important as even globally optimized dense depth maps remain uncertain due to the scene structure (homogeneous regions, periodic patterns, etc.) and noise.
Then the resulting weighted point cloud is quantized in a homogeneous grid in 3D space as voxels and temporarily filtered. This filter step is important to handle noise effects, to merge several point clouds and to enlarge the field of view by predicting the voxels, even when parts of the scene leave the field of view of the camera. Within the grid it is possible to apply the marching cubes algorithm that generates consistent surfaces. Finally, texture from the image stream is projected onto the 3D surface.

II. 3D Data Extraction

A. Egomotion Calculation

Assuming a static background the motion of the camera is considered as a Euclidean motion (fig 1). Therefore the motion between two time frames is modeled via a rotation matrix $\mathbf{R}(\alpha, \beta, \gamma)$ with $\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}$ and a translation vector $\mathbf{t} = (t_x, t_y, t_z)^T$, with

$$\mathbf{X}_{k+1} = \mathbf{R}^T\mathbf{X}_k - \mathbf{R}^T\mathbf{t}, \tag{1}$$

where $\mathbf{X}_k \in \mathbb{R}^3$ is a point in the coordinate system of camera $\mathbf{c}_k$ at time k.

The egomotion estimation is based on a robust iterative extended Kalman filter (RIEKF) as in [14]. The robust preprocessing is necessary because moving objects or outliers induce deviant motions and must therefore be rejected. The filter can be initialized e. g. by performing an estimation and a factorization of the essential matrix (as in [3]). Sparse optical flow vector fields are used to determine the rotation and translation of a calibrated camera. More precisely, we use point triples $\langle \mathbf{p}_i^{t-2}, \mathbf{p}_i^{t-1}, \mathbf{p}_i^t \rangle$ to minimize the epipolar and trifocal constraint. Additionally,

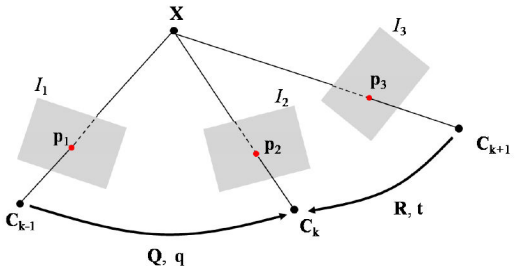

Fig. 1. Motion model for three consecutive frames. **R**, **t** are the Euclidean motion parameters to be estimated. **Q**, **q** are the motion parameters from the previous time step.

we integrated a constraint similar to bundle adjustment by minimizing the distance between the projection of the 3D point $\mathbf{X}_i(\mathbf{p}_i^{t-2}, \mathbf{p}_i^{t-1}, \mathbf{R}^{t-1}, \mathbf{t}^{t-1})$ and the image point $\mathbf{p}_i^t$. In contrast to bundle adjustment, the depth $Z(\mathbf{X}_i) = Z_i$ of $\mathbf{X}_i = (X_i, Y_i, Z_i)^T$ is added to the measurement vector $\mathbf{z}_i = (\mathbf{p}_i^{t-2}, \mathbf{p}_i^{t-1}, \mathbf{p}_i^t, Z(\mathbf{X}_i))$ of the RIEKF. As the filter does not only update the state variables $\mathbf{x}(\mathbf{R}, \mathbf{t})$ but also the measurements, we get an estimation of the depth values of the flow vectors without increasing the dimension of the state vector. This again helps stabilizing the estimation process over time.
Fig. 2 shows the top view of the scene that was used for data acquisition and evaluation and Fig. 3 shows the reconstructed motion path.

B. Reconstruction

In this paper, we use the semi-global matching (SGM) approach [8]. Compared to global approaches, it is one of the best competitive algorithms and is though computationally more efficient [15]. SGM approximates a global energy minimization for the whole 2D image by performing some kind of scanline optimization on several 1D paths across the image. As was shown by Ernst and Hirschmüller [16] SGM is applicable for real-time applications. SGM is not restricted to rectified images, but to images with known epipolar geometry. So we can use SGM also for images from a forward moving camera respective for sequences with the epipole close to the image center. Then, instead of determining the disparity along a horizontal line, the disparity is determined along the epipolar line.
Once the intrinsic camera matrix $\mathbf{K}$ and the motion parameters $\mathbf{R}$ and $\mathbf{t}$ are known, the fundamental matrix is given by

$$\mathbf{F} = \mathbf{K}^{-T}\mathbf{R}[\mathbf{t}]_\times\mathbf{K}^{-1} \tag{2}$$

with the skew symmetric matrix $[\mathbf{t}]_\times$. Then the epipolar line $\vec{l}_m$ in the match image I_m of a point $\mathbf{x}_b$ in the base image I_b is given by

$$\vec{l}_m = \mathbf{F} \cdot \mathbf{x}_b. \tag{3}$$

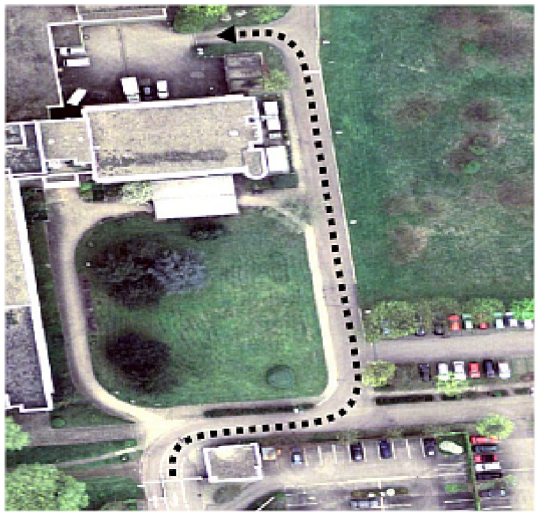

Fig. 2. Top view of the Fraunhofer test area and the covered path of the camera mounted vehicle.

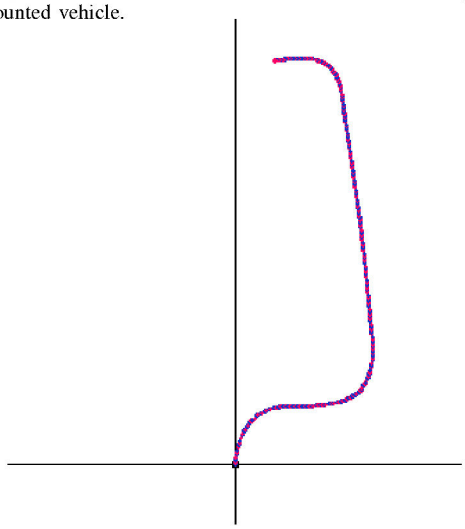

Fig. 3. Reconstructed camera path with the RIEKF-based on egomotion estimation.

To find the point $\mathbf{x}_m^0$ in I_m that corresponds to $\mathbf{x}_b$ with zero disparity, the 3D point $\mathbf{X}_\infty = z_\infty \cdot \mathbf{K}^{-1}\mathbf{x}_b$ is projected into I_m:

$$\mathbf{x}_m^0 = \mathbf{K}[\mathbf{R}^T| - \mathbf{R}^T\mathbf{t}]\tilde{\mathbf{X}}_\infty. \tag{4}$$

Then the SGM cost $C(\mathbf{x}_b, \mathbf{x}_m^d)$ can be calculated along the epipolar line $\vec{l}_m$ for all disparities $d = 0, ..., d_{max} - 1$.
Additionally to the SGM cost paths along the image scanlines, an additional 1D cost path in the time axis is considered for each pixel. This means that only small disparity changes over time are preferred. Especially the robustness of the disparity estimation of background objects with a small but relatively constant disparity can be increased. On th eother hand, close and hence fast moving objects may occur blurred. This kind of 3D-blurring enforces the visual effect that human observers can experience when looking out of a window inside a fast moving vehicle.
Once the corresponding image points are determined, the 3D

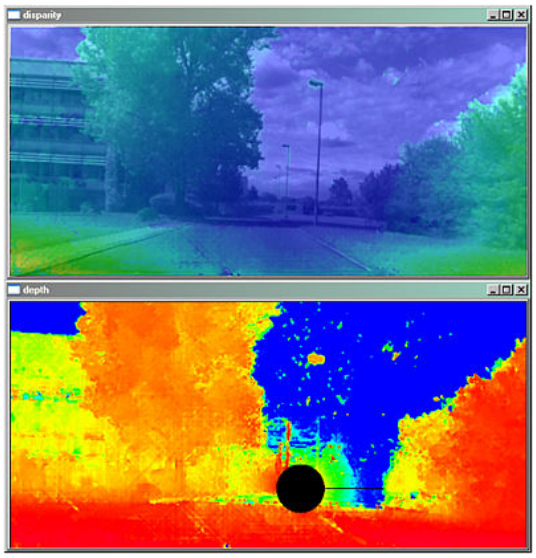

Fig. 4. Top: Estimated disparity map from a forward moving camera (blue=0, red=64). Bottom: Depth map (red=close, blue=far away). The region around the epipole is cut out.

scene point is calculated via triangulation with the known intrinsic and motion parameters (Fig. 4). Keep in mind that this reconstruction approach returns wrong results for independently moving objects because the assumed epipolar geometry is different for these objects.

C. Weighting the Data

Although SGM generally performs very well, there are scenes where stereo algorithms simply deliver bad disparity estimations (Fig. 6). This might be the case, if the epipole is inside the image or the image contains large homogeneous regions or periodic patterns. Then, we would like to know which pixels are reliable and which are not. As SGM delivers a cost for each pixel $\mathbf{p}$ and each disparity from 0 to $d_{max}-1$ we can use these costs to define a function (Fig. 5)

$$\omega\big(C_{min}(\mathbf{p})\big) = b \cdot log\Big(a \cdot \big(C_{max}(\mathbf{p}) - C_{min}(\mathbf{p})\big)\Big) \quad (5)$$

with

$$a := \frac{1}{C_{max}(\mathbf{p}) - C_{2nd}(\mathbf{p}) - \epsilon} \quad ,$$

$$b := \frac{1}{log(a \cdot C_{max}(\mathbf{p}))} \quad .$$

$C_{min}(\mathbf{p})$ is the minimum SGM cost at pixel $\mathbf{p}$ (which belongs to the current disparity d($\mathbf{p}$)). $C_{2nd}(\mathbf{p})$ and $C_{max}(\mathbf{p})$ are the second smallest and the biggest cost at $\mathbf{p}$ respectively. ω fulfills the constraints $\omega(0) = 1$, $\omega(C_{2nd}(\mathbf{p}) + \epsilon) = 0$ and $\omega(C_{max}(\mathbf{p})) \to -\infty$.

So, we have two effects: First, the smaller the distance between C_{min} and C_{2nd} the smaller is ω and therefore the weaker the certainty of $d(\mathbf{p})$. Second, on the other hand, if distance $C_{2nd} - C_{min}$ is still small but C_{max} gets bigger the function gets steeper and ω is bigger, too. Hence, when C_{max} is big relative to C_{min} this $\omega(C_{min})$ rewards minimum costs even if it is likely that they belong to a wrong disparity.

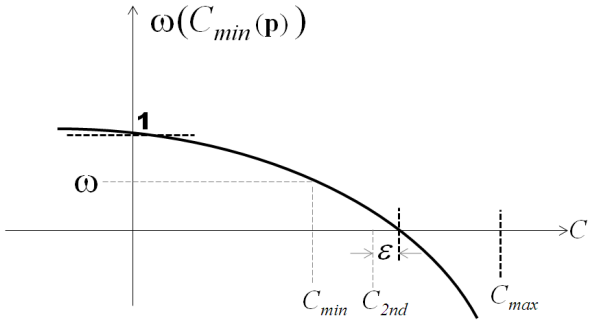

Fig. 5. Weight function $\omega\left(C_{min}(\mathbf{p})\right) \in (0, 1]$.

$\epsilon > 0$ is simply a value to avoid $\omega = 0$. One should keep in mind that although ω is used as a weight for the 3D points it basically says nothing about the quality of the 3D point. Even more it is a degree of certainty of the disparity in the image domain.

III. 3D Data Processing

A. The Voxelfilter

Errors in depth calculation are hardly avoidable when the 3D point cloud is estimated from a monocular image with an online processing algorithm. Here, online means that the data is processed continously. The amount of data from the past, that can be considered to refine the measurement, is strongly restricted by computational resources. This is a big disadvantage compared to approaches like global bundle adjustment optimizations where a whole image sequence could be considered to find an optimal solution for both camera motion and the 3D scene structure. As the motion estimation itself is defective because of the noisy measurement process, the resulting point cloud will also be erroneous. This leads to a fuzzy 3D structure over time. This effect gets even worse when the depth maps are fused with point clouds from other cameras where calibration errors additionally complicate the data fusion. Furthermore, the different base lines between two consecutive frames due to different camera motion velocities lead to different depth resolutions which can also be interpreted as threedimensional noise.

Instead of registering all points between two time steps or different sources, we chose a solution that is quite simple and takes account of the weights of the point cloud. The space is quantized by homogeneous 3D grid cells, so called voxels. As our goal is an intuitive online visualization rather than the generation of exact surfaces, this loss of accuracy is acceptable. Even more, this quantization significantly reduces the amount of data and hence computation time and can be used to polygonize the surfaces.

Initially, the 3D points within a single cell are averaged. So each cells has its own mean position and weight. At the next time step, this mean can be predicted using the known motion information of the camera.

At each prediction step the weight of the voxel is multiplied by a "history factor" $\rho \in [0, 1)$. This factor effects that

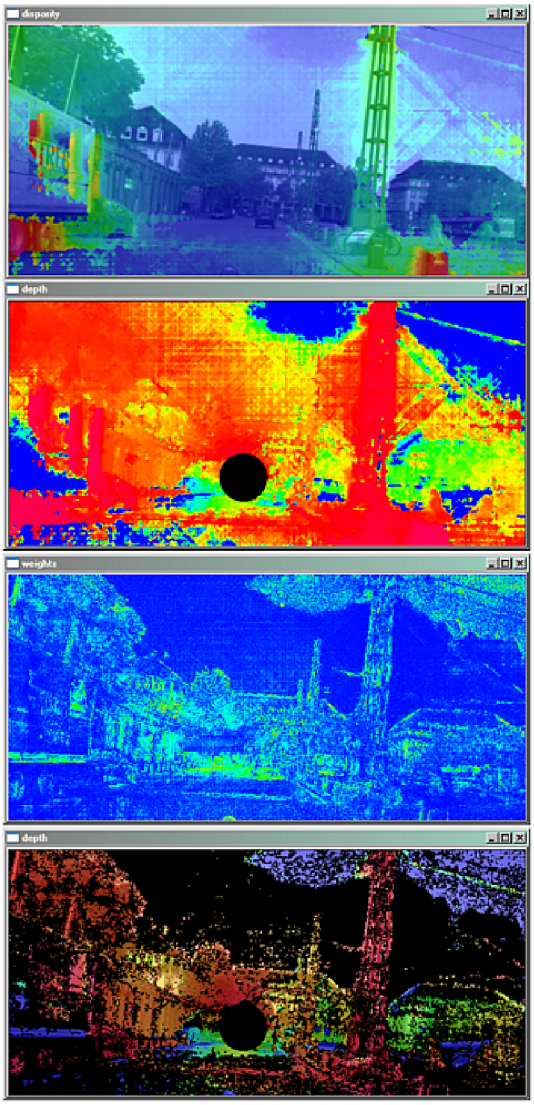

Fig. 6. From top to bottom: Scene with estimated disparity using SGM (blue=0, red=d_{max}); calculated depth map (blue=far, red=close, the epipole in the center is cut off); ω-weights with $\epsilon = 1.1$ (blue=0, red=1); adjusted depth map after thresholding with 0.1 (image texture is deposited)

voxels, that are not fed with new 3D points during the update step, lose weight over time and finally distinguish. The stronger the weight of a voxel is, the longer is its lifetime. ρ should be linked with the velocity of the camera. The slower the motion becomes, the closer ρ approaches to 1. If the egomotion is quite fast (and hence the scene changes very quickly), ρ could be chosen to be small. By predicting 3D points and weights, the 3D structure of the scene is tracked although there might be no more image information available. This is especially then reasonable when point clouds from different sources should be merged.

To update the filter, the currently measured point cloud is inserted into the grid with the predicted voxels. Then, new weighted means are calculated for each voxel. To resample the voxels, the voxel grid can simply be smoothed with a threedimensional gaussian kernel.

B. Dual Marching Cubes

To visualize the filtered 3D point clouds generated by the voxelfilter our algorithm calculates a grid which is dual to the voxel space connecting the weighted means within the voxels. These dual grid cells can be assessed individually by MC to create isosurfaces consisting of triangle meshes.

Starting from voxels containing points of the point cloud, the dual grid connects weighted means of eight voxels having one vertex in common (Fig. 7). Voxels which weight is > 0 are called "active", empty voxels are called "dead". This results in dual grid cells which are topologically equivalent to cubes consisting of eight weighted vertices. We are aligning the dual grid construction only to the active voxels. In this context, active voxels have to be enclosed by dual grid cells. So each active voxel is the center of eight dual grid cells. Furthermore, border vertices of the dual grid have to correspond to weighted means of dead voxels. Sign changes occur, if the weight of a vertex that is spanning an edge is above or equal to the isolevel while the weight of the other vertex is below the isolevel. Using MC to polygonize a dual grid cell, its edges have to be examined for sign changes. In the presence of sign changes a dual grid cell affords triangles of the isosurface of the scene. So, constructing the dual grid is restricted to active voxels, since connecting the weighted means of dead voxels does not affect sign changes. Once a sign change is present, the intersection of isosurface and edge is calculated by linear interpolation.

Due to applying MC to the dual grid, sharp features can be modeled in the resulting visualization. As the cells of the dual grid connect weighted means within a voxel of the voxel space, its vertices correspond to 3D features.

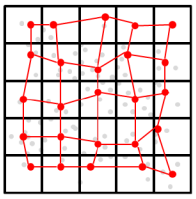

Fig. 7. Dual grid of the means of the measured points in a 2D homogeneous grid.

C. Texture Mapping

Once MC has put out triangle meshes modeling objects of the scene each of the triangles is defined by the coordinates of its corners. Using the properties of projective geometry of the pinhole camera paradigm one can project the 3D coordinates of a triangle corner $\mathbf{U}$ into the image plane

$$\mathbf{u} = \mathbf{K}\mathbf{U}, \tag{6}$$

where $\mathbf{K}$ is the intrinsic camera matrix. Once the 2D coordinates of the three corners of a triangle are available a triangular aperture of the 2D image can be assigned to the 3D triangle. In this manner, the triangle meshes generated by MC can be annotated with texture from the online captured camera images.

IV. Results

Fig. 8 shows some results from a sequence (scene in Fig. 2) with filtered voxels. The weighted mean of each homogenous grid cell (which is a vertex of a dual grid cell) is connected with the means of the neighboring cells. As one can see, the rough scene structure is alreaday visible.
Fig. 9 shows results after MC and texture mapping was applied. As can be seen, there is still space for improving the quality of the online generated 3D model. Although there is still clutter around the objects, the scene structure is well observable. An attached smoothing step could help to design the scene more intuitive. The voxel filter tracks the 3D structure even when no data is available from the reconstruction step. In the textured views, an additional light source and render step could help to enforce the visual 3D effect.
The bottleneck in our application was the 3D reconstruction. The computation time of voxel filtering and dual grid computation as well as marching cubes and texture mapping is altogether less than 250 ms.

V. Conclusions

A. Summary

We presented a framework for an online generation of synthetic views from monocular image sequences. The process chain starts with an egomotion estimation based on sparse image features and flow triples. These image features are processed within a robust iterative extended Kalman filter to estimate the Euclidean motion of the camera. Once the motion is known, a SGM stereo approach is used to generate weighted depth maps. These weights are important for multi-sensor fusion and temporal accumulation of point clouds respective the scene structure. The resulting point cloud is quantized and filtered in a homogenous grid. Hence, a simple but effective method could be used to temporally filter 3D data. Based on the weighted means of the grid a dual grid is created which is used to run a marching cubes algorithm. As a result polygonized surfaces are available to map the image texture. The whole chain is designed to run in real-time and hence delivers an online 3D visualization of the scene.

B. Future Works

The contributed framework is generally designed for multi camera applications. Hence, we are going to build up a calibrated multi sensor rig. By fusing the monocular generated and filtered point clouds of at least four cameras around a vehicle, we hope for an almost omnidirectional field of view. This offers new possibilities in driver's navigation and online scene exploration.
Further algorithms can be attached to reduce clutter and to smooth surfaces. Another important step to increase the quality of the visualization is the integration of a specific ground floor model. There are several possibilities, e. g. via an offline calibration or a dynamic model. Finally, some work could be invested to render the scene or to generate artificial scene parameters, e. g. lighting conditions, to increase the visual 3D effect.
To come closer to the goal of real-time processing, current development is done to speed up the algorithms and to implement parts of it on hardware like GPUs and DSPs. Currently, the bottleneck is the stereo processing. But as real-time capable hardware implementations have already been realized, we are confident to yield significant speedups. The visualization and voxel filtering itself is quite fast and predestinated for parallel processing.

References

[1] M. Pollefeys, "Self-calibration and metric 3d reconstruction from uncalibrated image sequences," Dissertation, 1999.
[2] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, 2007, pp. 1052–1067.
[3] D. Nister, "An efficient solution to the five-point relative pose problem," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.
[4] T. Dang, C. Hoffmann, and C. Stiller, "Fusing optical flow and stereodisparity for object tracking," in *Proc. of the 5th IEEE International Conference on Intelligent Transportation Systems*, 2002, pp. 112–117.
[5] R. T. Collins, "A space-sweep approach to true multi-image matching," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 1996, pp. 358–363.
[6] S. Birchfield and C. Tomasi, "Depth discontinuities by pixel-to-pixel stereo," *International Journal of Computer Vision*, vol. 35, pp. 1073–1080, 1999.
[7] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate eenergy minimization via graph cuts," Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, Tech. Rep., 2003.
[8] H. Hirschmüller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 807–814.
[9] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, R. Yang, D. Nister, and M. Pollefeys, "Real-time visibility-based fusion of depth maps," in *IEEE International Conference on Computer Vision*, 2007.
[10] W. Zhao and S. H. D. Nister, "Alignment of continuous video onto 3d point clouds," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.
[11] S. Fortune, *Voronoi diagrams and Delaunay triangulation*. Du, D.-Z.; Hwang, F.: World Scientific, Computing in Euclidean Geometry (Lecture Notes Series on Computing 1), 1992.
[12] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *Proceedings of ACM SIGGRAPH*, 1987.
[13] S. Schaefer and J. Warren, "Dual marching cubes: Primal contouring of dual grids," in *Proceedings of Pacific Graphics*, 2004.
[14] F. Pagel, "Robust monocular egomotion estimation based on an iekf," in *Canadian Conference on Computer and Robot Vision*, 2009.
[15] P. Steingrube, S. K. Gehrig, and U. Franke, "Performance evaluation of stereo algorithms for automotive applications," in *Proceedings of the 7th International Conference on Computer Vision Systems*, 2009.
[16] I. Ernst and H. Hirschmüller, "Mutual information based semi-global stereo matching on the gpu," in *Int. Symposium on Visual Computing*, 2008.

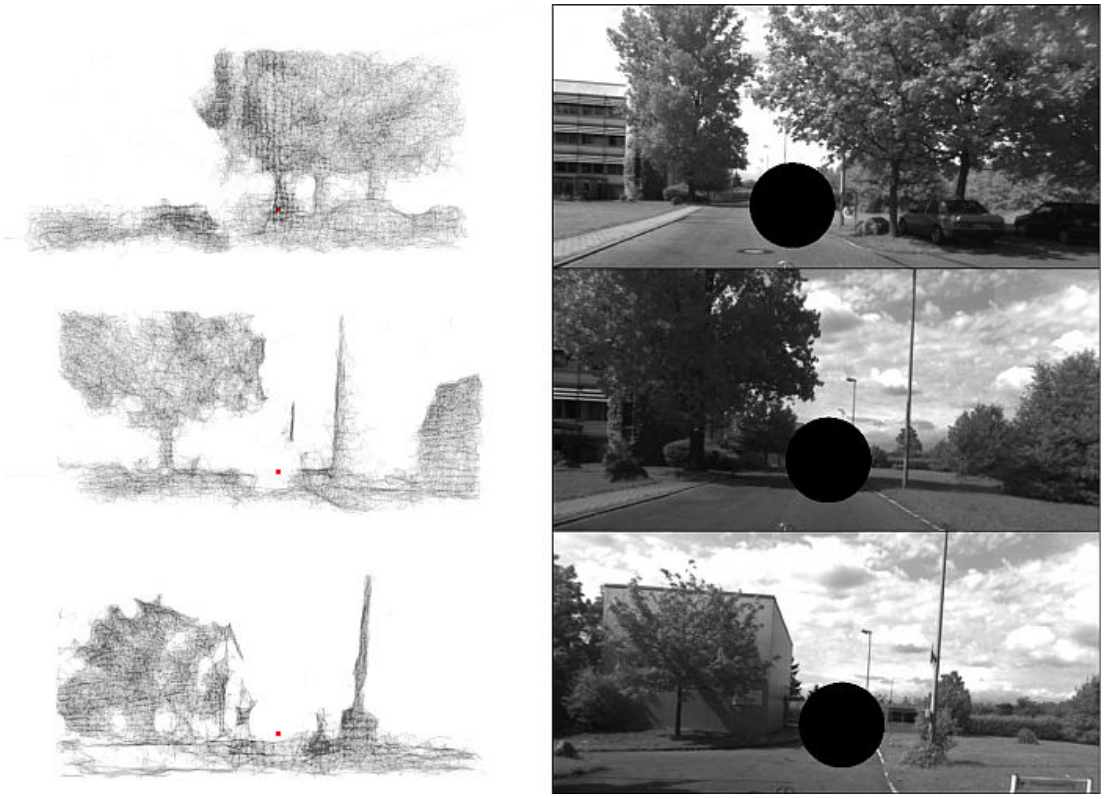

Fig. 8. Filtered dual grid structures from the scene in Fig. 2. The epipole in the original views (right column) is cut out.

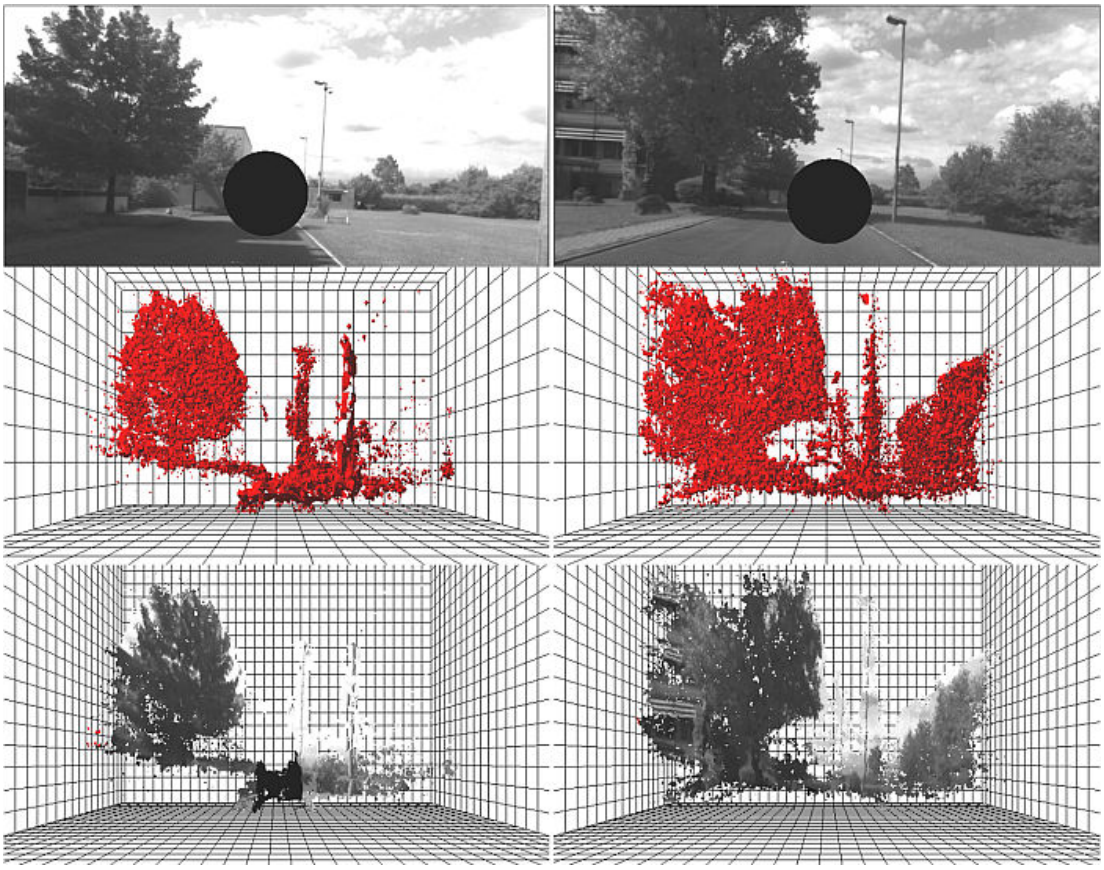

Fig. 9. Two stills from the test sequence (Fig. 2). Top: Original views (the epipole is cut out). Middle: Polygonized reconstruction after Marching Cubes. Bottom: Textured scene.