Dense GPU-enhanced surface reconstruction from stereo endoscopic images for intraoperative registration

Sebastian Röhl, Sebastian Bodenstedt, Stefan Suwelack, Hannes Kenngott, Beat P. Müller-Stich, Rüdiger Dillmann, and Stefanie Speidel

Citation: Medical Physics **39**, 1632 (2012); doi: 10.1118/1.3681017
View online: http://dx.doi.org/10.1118/1.3681017
View Table of Contents: http://scitation.aip.org/content/aapm/journal/medphys/39/3?ver=pdfcov
Published by the American Association of Physicists in Medicine

FAST. SIMPLE. POWERFUL.
AUTOMATE YOUR TG-142 QA AND SAVE HOURS OF TIME
DoseLab
THE FIRST QA SOFTWARE TO GET TG-142 RIGHT
MOBIUS MEDICAL SYSTEMS
INNOVATIVE SOFTWARE FOR MODERN RADIATION ONCOLOGY
www.mobiusmed.com

Dense GPU-enhanced surface reconstruction from stereo endoscopic images for intraoperative registration

Sebastian Röhl,[a)] Sebastian Bodenstedt, and Stefan Suwelack
Humanoids and Intelligence Systems Lab, Karlsruhe Institute of Technology, Adenauerring 2, 76131 Karlsruhe, Germany

Hannes Kenngott and Beat P. Müller-Stich
General, Visceral and Transplantation Surgery, Heidelberg University Hospital, Im Neuenheimer Feld 110, 69120 Heidelberg, Germany

Rüdiger Dillmann and Stefanie Speidel
Humanoids and Intelligence Systems Lab, Karlsruhe Institute of Technology, Adenauerring 2, 76131 Karlsruhe, Germany

(Received 5 September 2011; revised 10 January 2012; accepted for publication 12 January 2012; published 1 March 2012)

Purpose: In laparoscopic surgery, soft tissue deformations substantially change the surgical site, thus impeding the use of preoperative planning during intraoperative navigation. Extracting depth information from endoscopic images and building a surface model of the surgical field-of-view is one way to represent this constantly deforming environment. The information can then be used for intraoperative registration. Stereo reconstruction is a typical problem within computer vision. However, most of the available methods do not fulfill the specific requirements in a minimally invasive setting such as the need of real-time performance, the problem of view-dependent specular reflections and large curved areas with partly homogeneous or periodic textures and occlusions.
Methods: In this paper, the authors present an approach toward intraoperative surface reconstruction based on stereo endoscopic images. The authors describe our answer to this problem through correspondence analysis, disparity correction and refinement, 3D reconstruction, point cloud smoothing and meshing. Real-time performance is achieved by implementing the algorithms on the GPU. The authors also present a new hybrid CPU-GPU algorithm that unifies the advantages of the CPU and the GPU version.
Results: In a comprehensive evaluation using *in vivo* data, *in silico* data from the literature and virtual data from a newly developed simulation environment, the CPU, the GPU, and the hybrid CPU-GPU versions of the surface reconstruction are compared to a CPU and a GPU algorithm from the literature. The recommended approach toward intraoperative surface reconstruction can be conducted in real-time depending on the image resolution (20 fps for the GPU and 14fps for the hybrid CPU-GPU version on resolution of 640×480). It is robust to homogeneous regions without texture, large image changes, noise or errors from camera calibration, and it reconstructs the surface down to sub millimeter accuracy. In all the experiments within the simulation environment, the mean distance to ground truth data is between 0.05 and 0.6 mm for the hybrid CPU-GPU version. The hybrid CPU-GPU algorithm shows a much more superior performance than its CPU and GPU counterpart (mean distance reduction 26% and 45%, respectively, for the experiments in the simulation environment).
Conclusions: The recommended approach for surface reconstruction is fast, robust, and accurate. It can represent changes in the intraoperative environment and can be used to adapt a preoperative model within the surgical site by registration of these two models. © *2012 American Association of Physicists in Medicine*. [DOI: 10.1118/1.3681017]

Key words: image-guided therapy, endoscopic procedures, surface reconstruction, intraoperative registration

I. INTRODUCTION

In the last decades, surgical procedures have changed substantially due to many technological innovations concerning diagnosis, treatment planning, and therapy. A good example is the rise of image-guided neurosurgery. It is now common practice to use intraoperative navigation systems based on pre-operative planning in tasks such as biopsy excision, tumor resection, or defect reconstruction.[1] Here, there is a definite increase in the quality of patient recovery. In other areas, such as laparoscopic surgery, the application of computer-assisted navigation has not yet been fully utilized in daily clinical routine. One reason for this are soft tissue deformations, which change the surgical site substantially, and therefore hamper the use of pre-operative planning. In order to adapt the planning, intraoperative sensor data have to be acquired on a continuous basis and incorporated into a patient-individual model of the surgical site. This model can then be registered to preoperative planning data.

In laparoscopic surgery, the endoscope is the typical intraoperative source of sensor data. Extracting depth information from stereo endoscopic images and building a surface model from the surgical field-of-view is one way to represent the constantly deforming environment (Fig. 1). Although the use of stereo endoscopes is still not common practice in traditional minimally invasive procedures, they are used in robotic systems such as the daVinci surgical system. With monocular endoscopes, depth information can only be acquired using structure from motion,[2] shape from shading,[3,4] simultaneous localization and mapping,[5] or structured light.[6,7] Structure from motion as well as simultaneous localization and mapping both have the disadvantage that the endoscope has to be constantly moved in order to get 3D information, that they need stable features to track and that it is difficult to recover deformations that happen during camera motion. Shape from shading approaches only provide relative depth information, have problems discriminating concave and convex surfaces and are sensitive to specular highlights. Structured light requires an additional projector, which makes miniaturization and calibration difficult. Recent hardware technologies such as time of flight endoscopes are still at the beginning of their development.[8]

When stereo images are used, depth information can be extracted in solving the correspondence problem by finding similar structures in both images. This is a typical problem in computer vision and many methods have been developed and evaluated.[9] However, most of them do not fulfill the specific requirements within a minimally invasive setting, such as the need for real-time performance, the problem of view-dependent specular reflections, large curved areas with partly homogeneous or periodic textures and occlusions due to surgical instruments.

Many 3D reconstruction approaches particularly developed for the minimally invasive scenario detect and track features.[10–12] They recover only a sparse surface mostly using structure from motion or deformation tracking. A semidense surface is reconstructed by Stoyanov *et al.*[13] At first, they reconstruct a sparse surface using detected features. Next, their 3D information is propagated into neighboring regions by using the matched features as seeds. Chen *et al.* also use feature detection to stabilize stereo correspondence calculation.[14] They combine a dynamic programming approach with Delaunay triangulation and planar homographies. These methods depend on the availability of stable features in order to work properly. Another select group of methods use the concept of parametric models to describe the surface.[15,16] These methods are restricted to surfaces with known geometry where a spatial deformation can be estimated precisely, such as the surface of a heart. One of the earliest works in dense surface reconstruction from stereo endoscopic images is by Devernay *et al.*[17] A standard correlation-based method is used and registered with a temporal 3D model of the heart. Another method for dense surface reconstruction has been developed by Stoyanov *et al.*[18] Here, images are rectified to facilitate the correspondence analysis. A hierarchical structure is combined with piecewise bilinear maps to propagate disparities from low-resolution images to higher resolutions. Temporal information is also included in order to track the surface over time. Recent work in dense surface reconstruction was done by Vagvolgyi *et al.*[19] They define a new minimization function and solve it with a global dynamic programming approach on rectified images. The disparity image is then read out recursively. The quality of all the described methods above is difficult to assess because a substantial evaluation of the accuracy, robustness and speed is not covered in the publications.

This publication presents an extended version of a robust and dense surface model from stereo endoscopic images with millimeter accuracy that is reconstructed in real-time.[20,21] We describe solutions to disparity calculation and correction, 3D reconstruction and point cloud smoothing and meshing. Real-time performance is achieved by implementing the algorithms on the GPU.

General purpose computation on GPUS has become more and more popular because of its computational power when using highly parallelized algorithms. Programming language extensions such as CUDA or OPENCL have made programming more and more user-friendly. In medical applications, the use of the GPU for parallel processing has also gained a lot of momentum in recent years.[22] GPUS are now being used, e.g., in tomographic image reconstruction,[23,24] dose calculation,[25] image segmentation,[26] image registration,[27] or image visualization.[28] However, when using GPUS, one has to carefully consider the application and its disadvantages such as its

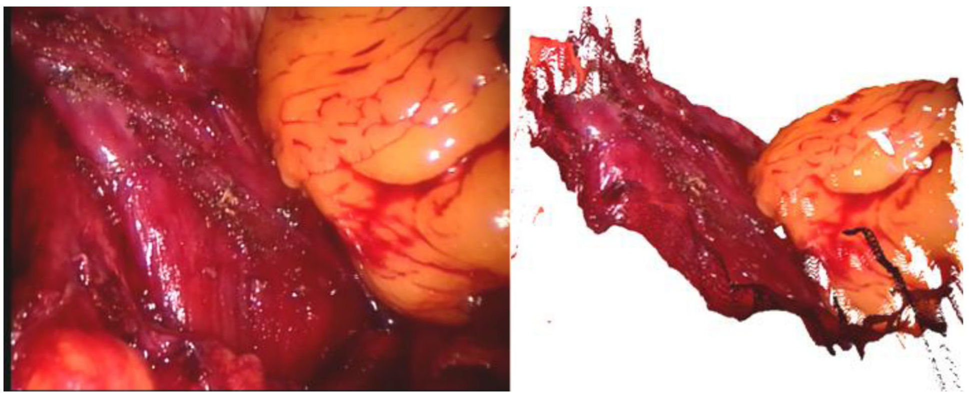
FIG. 1. Image from a daVinci intervention and reconstructed point cloud.

need for highly parallelizable problems, its limited memory, or the costly memory access.[24]

The main contributions of this publication are as follows. We describe in detail a complete redesign of the correspondence analysis method to work on the GPU. In addition, a sophisticated disparity post processing on the GPU with disparity correction and smoothing using bilateral filtering is introduced. The 3D reconstruction step is complemented by additional point cloud smoothing. The different algorithms are combined into a new hybrid CPU-GPU method which unifies the advantages of the CPU and the GPU versions of the surface reconstruction. In an extensive evaluation using *in vivo* data, *in silico* data from the literature and virtual data from a newly developed simulation environment, the CPU, the GPU, and the hybrid CPU-GPU versions of the surface reconstruction method are compared to a CPU and a GPU algorithm from the literature. The results reflect the quality of the approach through different stereo sequences and the superior performance of the hybrid CPU-GPU version.

II. MATERIALS AND METHODS

On the following pages, we present the necessary methods for reconstructing a surface from endoscopic images which can be registered with a preoperative model. An outline of the workflow can be seen in Fig. 2.

II.A. Preprocessing

The stereo endoscopic images are preprocessed in order to facilitate the correspondence analysis.[29] For this purpose and for an accurate 3D reconstruction, the intrinsic and extrinsic camera parameters are needed. We calibrate the camera using methods from Röhl *et al.* and the OPENCV library.[19,30] As endoscopic images are highly distorted, image undistortion has to be performed. We also rectify the images to reduce the search space during correspondence analysis. In a final step, the images are optionally smoothed with a 3×3 Gauss filter in order to reduce noise in the images.

II.B. Correspondence analysis

The object of the correspondence analysis in stereo images is to find points in both images which are the projection of the same world point into the image planes. The relationship between corresponding image pixels is described by the disparity, which is the relative distance between these pixels. The 3D reconstruction utilizes the detected correspondences and the geometry of the cameras to generate the three-dimensional structure of the observed scene.

Correspondence analysis methods can be classified into three groups. Feature-based methods detect and match features in both images using their descriptors. Feature detection is robust and very accurate, but the result is in general a sparse disparity image and the features are often distributed unevenly. Correlation- or pixel-based methods try to find correspondences for each pixel in the image using a similarity measure which is defined by a certain region around this pixel. These methods are simple and fast, but they often produce mismatches because each pixel has to be examined independently. Also, the similarity measures react sensitively to regions with no or periodic textures, occlusions and image artefacts. Global methods iteratively find a disparity function which describes the transformation of the whole image into the other. The problem presents itself as an energy minimization problem. These methods produce very accurate results, even in regions without texture, and lead to a dense disparity image, but are complex, slow and difficult to parallelize.

II.B.1. *Hybrid recursive matching (CPU version)*

The method we use is the Hybrid Recursive Matching (HRM) algorithm.[31] This algorithm, originally developed for video conferencing systems, is adapted and extended to be used with stereo endoscopic images.[20] The HRM is a combination of correlation-based and global methods. It recursively propagates the disparity information through the

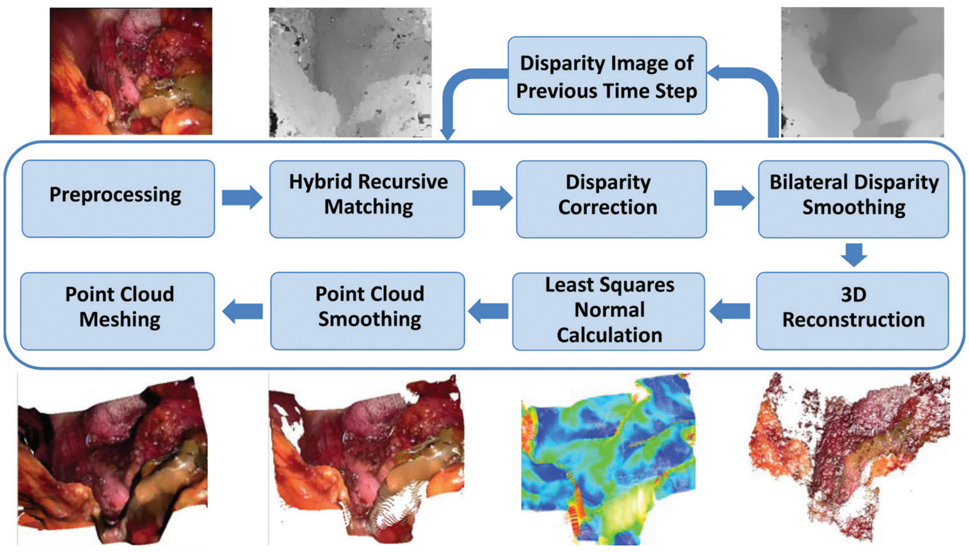
Disparity Image of Previous Time Step
Preprocessing
Hybrid Recursive Matching
Disparity Correction
Bilateral Disparity Smoothing
3D Reconstruction
Least Squares Normal Calculation
Point Cloud Smoothing
Point Cloud Meshing

FIG. 2. Outline of the surface reconstruction workflow.

image by generating potential correspondence candidates while using already determined correspondences from neighboring pixels. Additionally, one candidate is obtained from the disparity image of the previous time step. These candidates are then evaluated with a correlation-based similarity measure and the best is chosen as the correspondence. Hereby, a spatial and temporal dependency between disparities is introduced. This reduces the number of mismatches in homogeneous regions or regions which remain static over time. As input, the two rectified and undistorted grey images from the stereo endoscope, the disparity image from the previous time step and the already calculated disparities from the current time step are required [Figs. 3(a) and 3(b)]. As output, a dense disparity image is produced. The robustness of the algorithm is increased by iterating in a meandering pattern through the images and alternating the iteration starting point and direction with each new image pair.

In a two-stage process, consisting of the block recursion and the pixel recursion, a new disparity value is calculated for the current pixel by choosing between four different candidates in the other image. Three disparity candidates are generated in the block recursion step. Two originate from the vertically and horizontally neighboring pixel, the third from the previous time step. They ensure a smooth disparity distribution especially in low-textured regions. The fourth is calculated in the pixel recursion step. Using optical flow techniques, this step introduces new disparities in regions of discontinuity or at the edge of the image where no spatial candidates are known. This candidate is only computed if either the similarity error of the best block recursion candidate exceeds a certain threshold, which indicates such a region, or if there is no block recursion candidate, e.g., in the first image pair where no temporal information is given.

For each candidate, a similarity error is calculated by using block matching and a simplified census similarity measure.[31] This measure only considers relative intensity changes between pixels. This is beneficial for regions with global or local intensity changes such as areas near specularities or the image border. Here, comparing absolute intensities would lead to large errors in the similarity measure, whereas the relative intensity difference often is the same in both images. The candidate with the smallest error (most similar neighborhood) is set as the correspondence.

The algorithm has been extended to the level of subpixel precision. This is due to our camera setup. Because of the small distance between the cameras in the endoscope, the disparity interval is very narrow. When only using pixel precision, the reconstructed point cloud would consist of

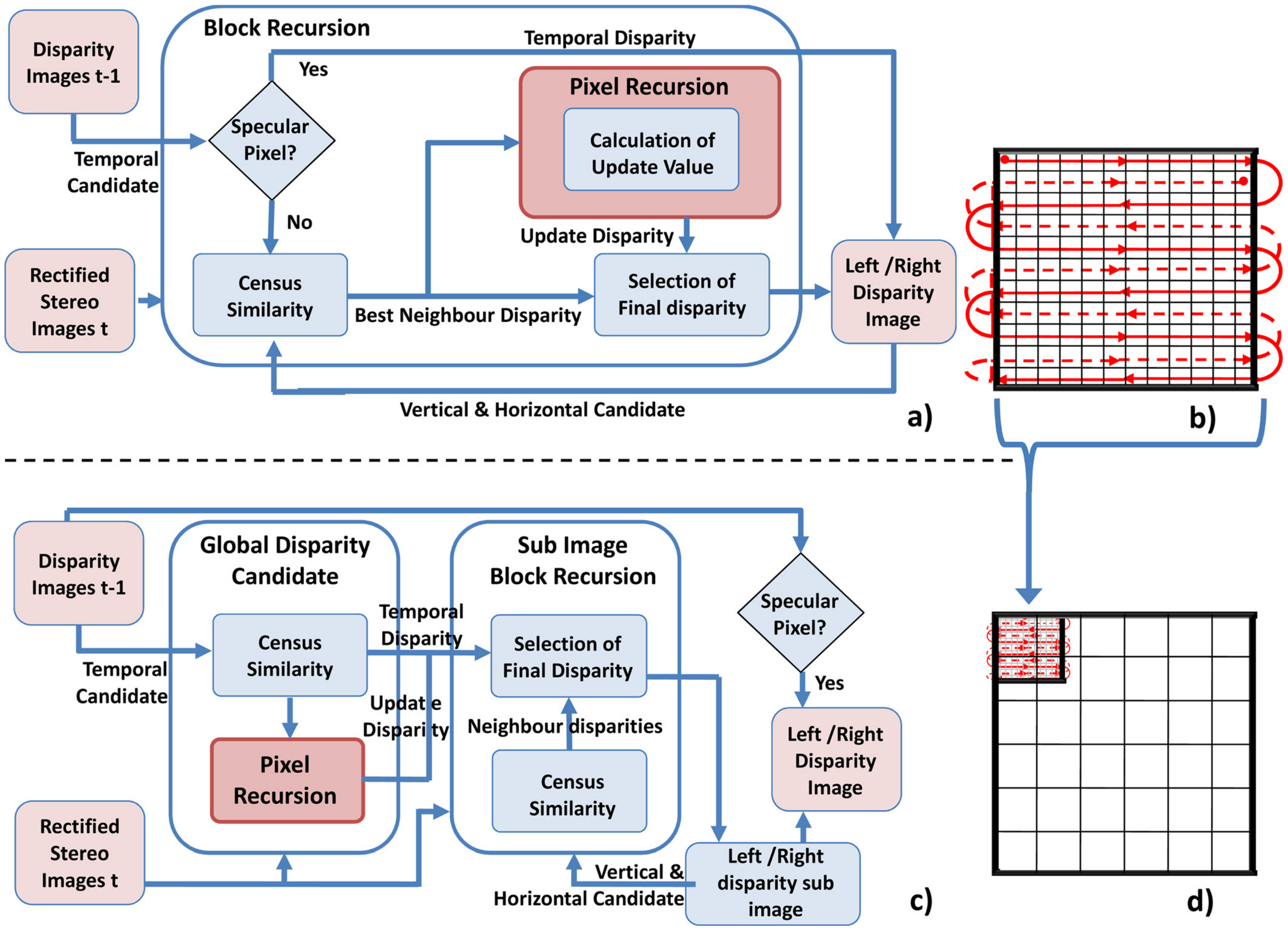
Disparity Images t-1
Block Recursion
Temporal Disparity
Yes
Specular Pixel?
Temporal Candidate
Pixel Recursion
Calculation of Update Value
No
Update Disparity
Rectified Stereo Images t
Census Similarity
Best Neighbour Disparity
Selection of Final disparity
Left /Right Disparity Image
Vertical & Horizontal Candidate
a)
b)
Disparity Images t-1
Global Disparity Candidate
Sub Image Block Recursion
Temporal Disparity
Census Similarity
Selection of Final Disparity
Specular Pixel?
Temporal Candidate
Update Disparity
Neighbour disparities
Yes
Pixel Recursion
Census Similarity
Left /Right Disparity Image
Rectified Stereo Images t
Vertical & Horizontal Candidate
Left /Right disparity sub image
c)
d)

FIG. 3. Comparison between the structure of the CPU and GPU version of the HRM: (a) CPU HRM structure; (b) CPU HRM recursion over whole image; (c) GPU HRM structure; and (d) GPU HRM recursion over overlapping sub images.

separated layers which have a distance of several millimetres between them.

In addition, simple specular highlight detection with intensity and saturation thresholds is being performed. For specular highlight pixels, a new disparity value is chosen by taking the disparity of the previous time step and weighting it in relation to the difference between the disparity of the previous and the current time step from the horizontal and vertical neighbor.

II.B.2. Hybrid recursive matching (GPU version)

We also implemented the correspondence analysis as well as the other steps of the approach on the GPU in order to benefit from the high degree of parallelism. When porting the HRM algorithm to the GPU, the recursive structure makes parallelization difficult. In order to be efficient, an algorithm on the GPU should consist of at least several hundred independent threads. Therefore, the HRM algorithm has to be restructured. The idea is to divide the image into a large number of overlapping sub images and to recursively generate a disparity image for each sub image before fusing them [Fig. 3(d)]. The overlapping does not result in memory conflicts, in which threads try to write to the same memory location, because the calculation in each sub image is synchronized to the other sub images. Some calculations for the whole image can still be performed independently. In this way, the algorithm is separated into a global disparity candidate step that processes the whole image in parallel and the sub image block recursion step, in which each sub image is processed independently [Fig. 3(c)].

With the given disparity images of the previous time step, we first calculate the similarity errors of the temporal disparity candidates. This can be done for each pixel independently using a large amount of threads (153 600 threads for two images with image resolution of 320×240). In order to improve the performance, the pixel recursion step is also included in the global disparity candidate step. As opposed to the CPU version which uses the winner of the block recursion step, it uses the temporal candidate to calculate a second disparity candidate and its similarity error for each pixel.

The sub image block recursion step uses either the temporal candidate or the pixel recursion candidate, depending on their similarity errors, as input. For each pixel in a sub image, a vertical and horizontal candidate is now calculated as it is done in the CPU version and is then compared to the input value. In contrast to the CPU version, there is no pixel recursion in the sub image block recursion step by default. Only if there is no temporal candidate, e.g., in the first image of the sequence, or if the temporal candidates are unreliable due to large image changes, pixel recursion can be called upon in this step to compute another candidate. The overlapping of the sub images ensures a smooth fusion between them. Disparities are propagated into other image parts, which leads to a reduction in mismatches at the edges of the sub images where none or only one spatial candidate is known. This step is the bottle neck of the GPU HRM. The number of threads is strongly reduced (768 threads with two 320×240 images as input). Due to the increase in the number of usable threads when using higher image resolutions, it is still faster than the CPU version. The loss of the global recursive structure can also lead to the effect that in large regions with homogeneous textures, the disparity is not well distributed, which in turn reduces the quality of the disparity image.

Specular pixels are detected and corrected as they are in the CPU version. The difference is that this happens after the HRM step whereas in the CPU version, they are corrected during the block recursion step. In this way, the correction is fully parallelized.

In both versions, interpolation can optionally be performed to speed up the process. In the sub image block recursion step of the GPU version, every other row can be interpolated using the calculated neighboring disparities. In the CPU version, every other row and every other second column of the image can be interpolated.

II.C. Disparity correction

The correspondence analysis step still produces mismatches in regions without texture or near occluded areas, which have to be corrected. The detection of these mismatches is performed via a double consistency check. An outline of the CPU disparity correction is described by Röhl *et al.*[20]

The GPU consistency check uses the same correction methods as its CPU counterpart, however, it combines them differently (Fig. 4). At first, the right disparity image is corrected. The result is used to correct the left disparity image. Each correction is separated into two steps. In the first step, information from the other image of the stereo pair is used to

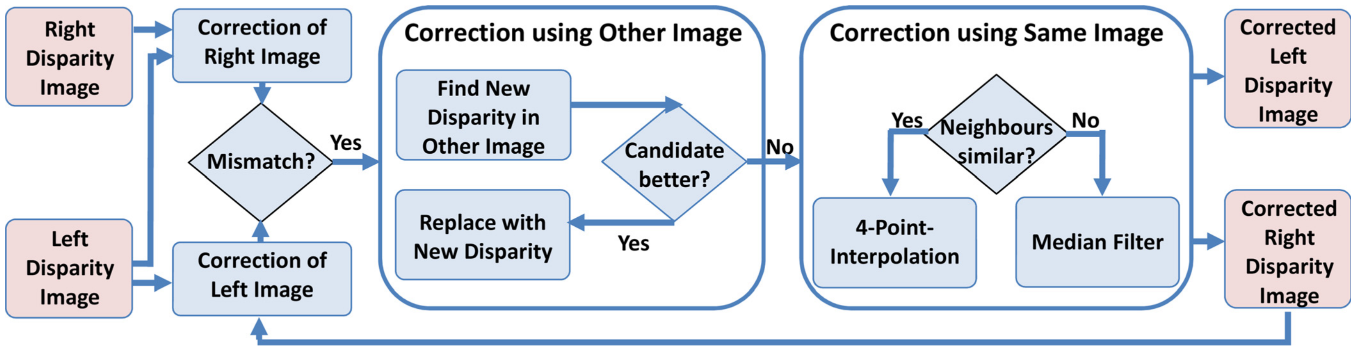
Right Disparity Image
Correction of Right Image
Mismatch?
Yes
Left Disparity Image
Correction of Left Image
Correction using Other Image
Find New Disparity in Other Image
Candidate better?
Replace with New Disparity
Yes
No
Correction using Same Image
Neighbours similar?
Yes
No
4-Point-Interpolation
Median Filter
Corrected Left Disparity Image
Corrected Right Disparity Image

FIG. 4. GPU consistency check and disparity correction procedure.

correct mismatches. The idea is to examine, whether the disparities in the other image are smooth and stable in that region and if their similarity error from the HRM step is smaller. In that case, they are regarded as dependable. If this is not the case, information from the same image is used. A simple interpolation using the four neighboring disparities is applied if their values are similar. The procedure removes single outliers in the disparity image. If the neighboring disparities differ too much, more complex median filtering over a larger neighborhood is used to obtain a new value.

Porting the median filter to the GPU is challenging. Although the median can be calculated independently for each pixel by utilizing a defined neighborhood, standard variants of the filter that find the median by sorting this neighborhood are inefficient due to the high amount of memory accesses. In order to reduce them, the idea is to approximate the median without explicitly sorting the neighborhood by using a histogram search that works for floating point numbers. Given a neighborhood N, we first calculate the width of the histogram bins by using a predefined histogram size and determining the smallest and largest value of N. For each bin, the number of values of N which would fall into this bin are computed. We use this information to build a cumulative histogram of N. The median is located in the first bin of the cumulative histogram whose value equals or exceeds half the size of N. We search through N again to determine all values of N which would fall into this bin. If it is only one value, the exact median is found. Otherwise, we compute their arithmetic average to get an approximated median.

II.D. Bilateral disparity smoothing

Without smoothing, the obtained disparity image is still noisy. This is especially critical when disparity values with subpixel precision are needed. Traditional smoothing filters do not preserve edges, which reduces the level of detail of the surface. To prevent that, we use the bilateral filter.[21,32] This filter calculates a new value for each pixel depending on the weighted average of neighboring pixel values. The idea is to weight the influence of a neighboring pixel not only by its spatial distance but also by its pixel value similarity in order to reduce the influence of pixels which are separated by a disparity discontinuity.

Typical applications of this filter are noise reduction in images, tone mapping or mesh smoothing. It is straightforward to use this filter on disparity images, since the pixel values encode the distance of a 3D point to the camera reconstructed from the corresponding points. Hereby, the disparity image is implicitly separated into regions which have either the same or a slightly changing distance to the camera. The filter only smoothes in these regions; discontinuities are preserved. The smoothing is also beneficial to the correspondence analysis in the following time step, as it improves the quality of the temporal candidate in these regions. The bilateral filter is implemented as an approximate version on the CPU (Ref. 32) and as an exact version on the GPU.

II.E. Point cloud reconstruction, smoothing and meshing

The camera calibration and the corrected and smoothed disparity image are used to calculate the 3D coordinates of the image points. This is realized by a standard triangulation method, which calculates the 3D point for each pixel independently. A GPU implementation is straightforward. The same mismatch detection as in the disparity correction step is used to identify any remaining mismatches. These points are not reconstructed. In addition, 3D points that violate the ordering constraint from the disparity image are also deleted from the point cloud. In this way, points from faulty disparities are effectively filtered.

In order to further smooth the point cloud, we applied the Total Least Square method presented by Hoppe *et al.*[34] and implemented it on the CPU and the GPU.[21] Here, for each point, a tangent plane is calculated from its neighborhood using the least squares method. The tangent plane normal vector serves as an approximation the real surface normal and can be used to describe the shape of the surface. It can serve as additional feature that can be used in surface registration.[35] In order to further smooth the point cloud, the normal of each point is used to project this point into the calculated tangent plane.

It is possible to quickly generate a surface mesh from the point cloud without searching for the nearest neighbors of a point. For meshing, we exploit the implicit alignment of the pixels in the image. The neighbors of a pixel can be identified precisely, without explicitly calculating them. When connecting points to triangles, we only have to ensure that the 3D points do not violate the ordering constraint, meaning that the neighbors in the image and the reconstructed point cloud have to be the same. If there are holes in the point cloud, we identify the neighboring points by searching from each missing point in horizontal direction, until after a valid point is found. Finally, the resulting point cloud or mesh is rendered using standard methods from VTK.[36]

II.F. Hybrid CPU-GPU algorithm

While the CPU implementation of the approach is too slow when using higher image resolutions than 320×240, the GPU implementation is not as robust because disparity information is only propagated within the sub images. Because of the modular structure of the workflow and the short time for copy instructions between CPU and GPU, GPU, and CPU parts can be combined arbitrarily. Thus, we developed a hybrid CPU-GPU algorithm, which is faster than the CPU implementation and more robust both implementations (Fig. 5). To achieve this, the idea is to take advantage of the temporal character of the HRM. We can improve the robustness by iterating multiple times over the same image. In addition, we want to combine the advantages of both versions. The CPU HRM works better on low-textured images because it can propagate disparities over larger parts of the images. Its GPU counterpart can only propagate disparities over one sub image, but it runs faster, especially on higher image resolutions.

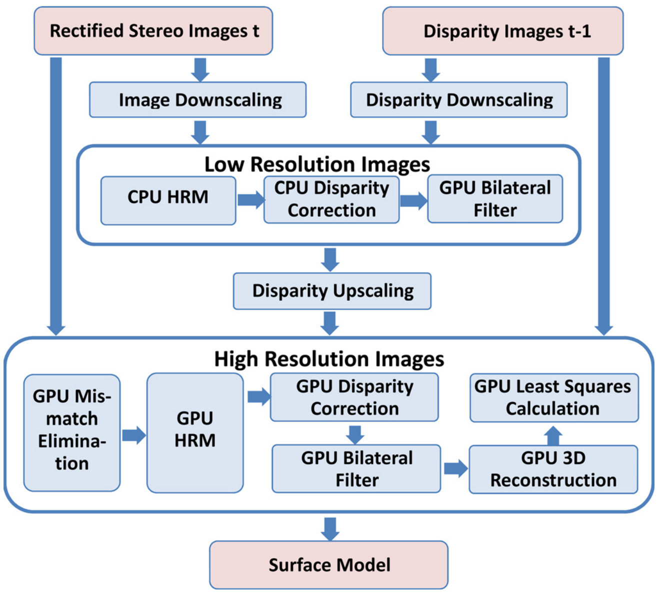

FIG. 5. Hybrid CPU-GPU algorithm workflow.

We interconnect CPU and GPU by first calculating the HRM and consistency check on the CPU, using half resolution images. To speed up the process, full interpolation is used. We also use the fact that the output of the HRM algorithm slightly differs depending on the direction of the iteration of the image. We therefore calculate two left and two right disparity images and merge the disparities by choosing the disparity value with the smaller similarity error. Afterward, GPU bilateral filtering is called upon to smooth the disparity images.

The resulting low-resolution disparity images are scaled up by using bilinear interpolation. Then, mismatches are filtered using a consistency check. The remaining disparities serve as first input for the GPU workflow. Furthermore, we use the disparity images from the previous time step as second input. In this way, the global disparity calculation step is extended to calculate a similarity error for both candidates. The candidate with the smaller error is chosen.

Subsequently, correspondence correction, bilateral filtering, 3D reconstruction and point cloud smoothing are performed on the GPU. The output of the GPU bilateral filter is scaled down and used as input during the following time step. The hybrid CPU-GPU version has the advantage that it produces more stable point clouds due to the repeated processing of the images and the noise reduction in lower image resolutions. It is also faster than the CPU workflow.

III. RESULTS

We evaluated the CPU, the GPU, and the hybrid CPU-GPU workflow on virtual and on *in vivo* images (see Video 1). The main focus of the evaluation was on run-time, accuracy and robustness against low-textured areas, large view point changes, noise and errors due to a poor camera calibration. It extends previous evaluations concerning the impact of temporal information and the use of bilateral post processing.[20,21] We decided against an evaluation on standard data sets, such as the Middlebury evaluation, because they only use single image pairs instead of image sequences and the images have no medical background with typical characteristics of endoscopic images.

The algorithms were implemented in C/C++ using Qt,[37] VTK from Kitware and IVT, a vision library.[29] The GPU implementation was done using CUDA from NVIDIA. The workflow was integrated into the MediAssist system, a computer assistance system for minimally invasive surgery that builds upon the IGSTK framework.[38,39]

For our experiments, we developed a simulation environment, where we capture images from a textured 3D liver model with a simulated stereo camera and compare the reconstructed point cloud with the known 3D surface (Fig. 6). We also evaluated stereo endoscopic images of a silicone phantom with ground truth point clouds.[13,40] In addition, we used images from three stereo endoscopes, a PAL endoscope (max. resolution 720×576) and a prototype HD-endoscope (1920×1080) from FA WOLF as well as the endoscope from the daVinci system.

We also compared the results of our three algorithms to a variant of a CPU semiglobal block matching stereo reconstruction algorithm[41] and a GPU constant-space belief propagation algorithm.[42] Both algorithms are implemented in the OPENCV library.[43] The CPU and the GPU OPENCV algorithm parameters are taken from the sample file in the library. Both OPENCV algorithms have their own disparity correction. They only reconstruct one disparity image, so there is no additional validity check in the 3D reconstruction step.

In the following paragraphs, we use as abbreviations CPU for our CPU version of the workflow, GPU for the GPU version and CPU-GPU for our hybrid workflow. The CPU semiglobal block matching algorithm is defined as CPU-OPENCV, the GPU constant-space belief propagation algorithm as GPU-OPENCV.

III.A. Evaluation on virtual image sequences

To evaluate the accuracy and robustness of the surface reconstruction algorithms, we created a test setting where we

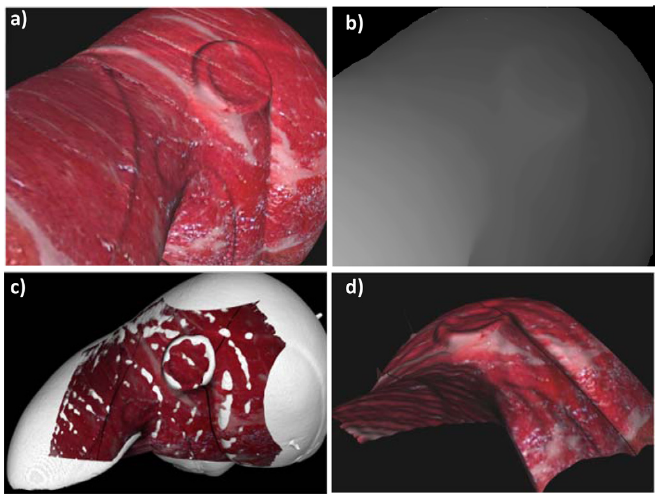

FIG. 6. Simulation environment: (a) image from virtual camera; (b) disparity image; (c) overlay of virtual liver with reconstructed surface; and (d) side view of reconstructed surface. [URL: http://dx.doi.org/10.1118/1.3681017.1]

capture images from a virtual liver model using a simulated stereo endoscope with a known calibration. Thus, we can compare a reconstructed point cloud directly with the liver model without requiring a registration. We are also able to reconstruct ground truth disparity images, which can be compared to ours. Errors in accuracy due to poor camera calibration, an inaccurate registration or image noise can be controlled or totally omitted.

In our evaluation, we used nine image sequences which consist of three test settings with three differently textured virtual liver models (Fig. 7). In each test setting, the distance between the virtual endoscope and the liver is approximately 5 cm. In setting 1 (circle), a 360° rotation around the liver with a step size of 5° is performed. Here, we can assess how our three versions for surface reconstruction react when the endoscope is moving sideways. In this case, temporal disparities can still be reliable in some parts of the image if the distance of the camera to the surface changes only slightly. In setting 2 (zoom), the camera zooms in and out on the same spot at the liver to see how the versions work with images where the distance and, hence, the disparity values rapidly change. The step size is 2 mm; the maximum zoom is 40 mm. Here, we investigate how the versions perform without any valid temporal information. In setting 3 (deform), only a certain area of the image is deformed whereas the endoscope has a fixed position. This shows how the three versions can cope with local deformations. In this case, temporal information is valid for a large part of the image. Disparity changes arise only in the deformation area. The first two textures are taken off images from different soft tissue. The third variant is an untextured liver. The different lighting of the pixels is the only viable information in order to distinguish between them. The untextured variant is the most challenging one. Here, the impact of homogeneous or surfaces without texture on the algorithm is being investigated.

We measured the accuracy and robustness of our five reconstruction algorithms using these image sequences. We also evaluated the robustness to image noise and camera calibration errors.

During the evaluation we used the parameters from Table I. We looked at the disparity difference between the ground truth disparities and the disparities reconstructed in our versions. Here, pixels at the image margin (10 pixels wide) and pixels that cannot be seen in the other image are not considered. We compared the mean error using all remaining disparities or, alternatively, only values that result

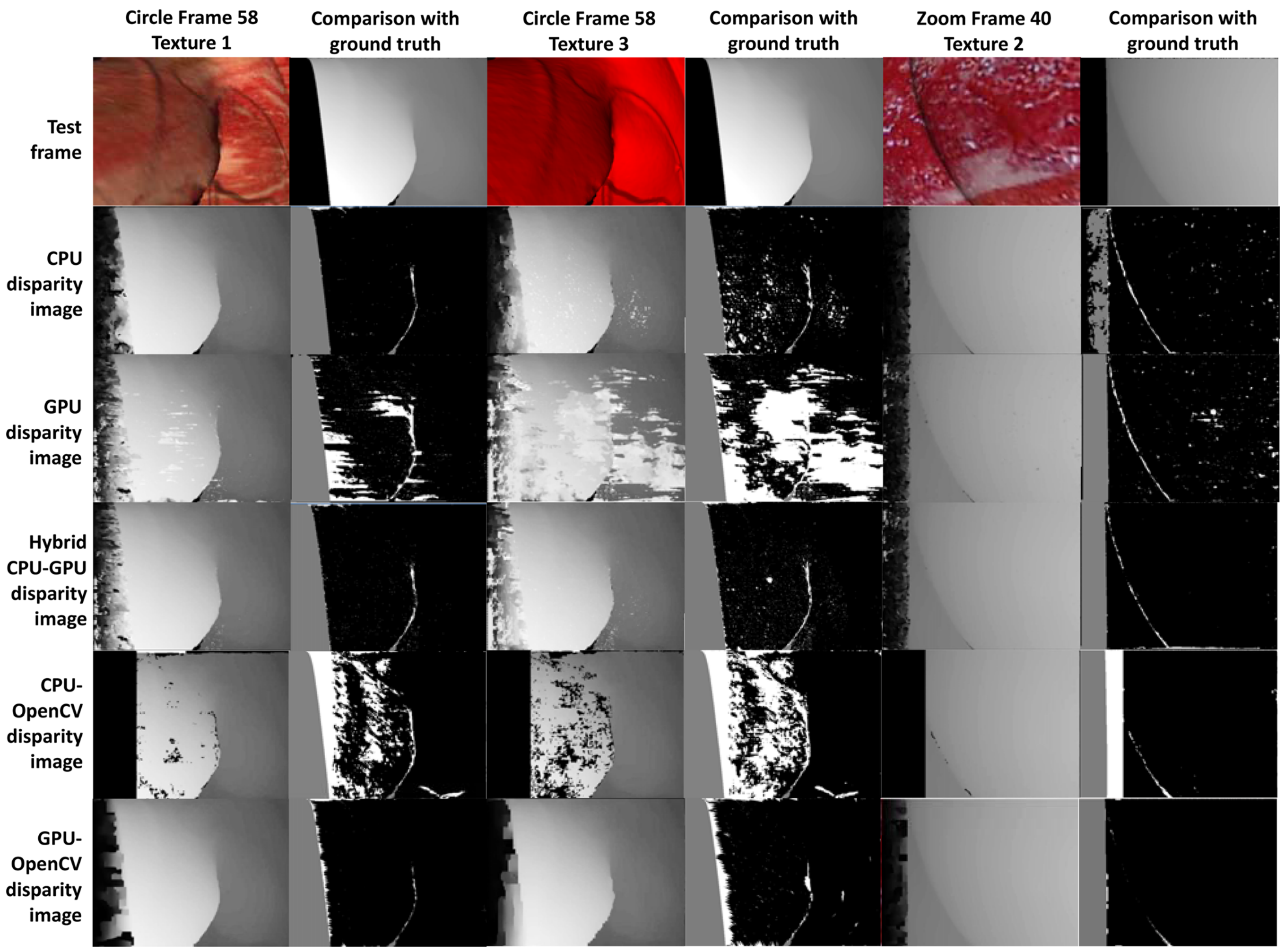
Circle Frame 58 Texture 1
Comparison with ground truth
Circle Frame 58 Texture 3
Comparison with ground truth
Zoom Frame 40 Texture 2
Comparison with ground truth
Test frame
CPU disparity image
GPU disparity image
Hybrid CPU-GPU disparity image
CPU-OpenCV disparity image
GPU-OpenCV disparity image

FIG. 7. Selected frames with different textures: CPU, GPU, CPU-GPU, and both OPENCV surface reconstruction disparity images and disparity difference (>1 pixel) to ground truth disparities.

TABLE I. Parameterization of the evaluation with virtual image sequences.

Parameter	Value
HRM block size	7×7
Correspondence correction threshold	0.5
GPU HRM sub image size	20×16
GPU HRM overlapping in x and y	4 Pixel
Bilateral filter neighborhood size	17×17
Bilateral filter spatial parameter σ_s	8
Bilateral filter range parameter σ_r	1.5
Least Squares neighborhood size	7×7
3D reconstruction image margin	10 pixel

in a reconstructed 3D point. We also compared the mean distance between the reconstructed and the ground truth point cloud. Therefore, we calculated the distance between a reconstructed 3D point and a plane described by the ground truth 3D point and its ground truth normal. Finally, we evaluated the number of reconstructed points for each version and image sequence. In order to calculate the percentage, we divided the number of reconstructed points by the width and height of the image.

The results in Table II show that the CPU-GPU version performs best in most of the cases. The GPU version performs significantly poorer on sequences with large areas without texture and large image changes such as the circle sequence. This is due to the limited disparity propagation and the separation of the pixel recursion step from the recursive parts of the algorithm which hampers the introduction of new disparities in large areas. But with enough texture or small image changes, it works as well as the other versions.

While the difference between the CPU-GPU and the CPU version while using the two textured livers is not significant, it outperforms the CPU version particularly on the untextured liver for the circle and zoom sequence. The mean distance between the reconstructed point cloud and the liver surface is down from 0.72 to 0.28 mm (a reduction by 61%) for the zoom sequence and down from 2.81 to 0.39 mm (86%) for the circle sequence. It also reconstructs more points (6% for the circle and 9% for the zoom sequence) which implies that more disparity mismatches are corrected.

In the zoom sequence using an untextured liver, both OPENCV algorithms outperform the CPU-GPU version. The mean distance for the CPU-OPENCV algorithm is reduced from 0.39 to 0.22 mm (reduction by 44%). The main reason for this is that here temporal information is completely unreliable and can even impair the results. In all other sequences, the CPU-GPU version is significantly more accurate, with a mean distance reduction of about 51% compared to the CPU-OPENCV and 60% compared to the GPU-OPENCV algorithm. It also reconstructs about 12% more points than the CPU-OPENCV algorithm, which in turn produces holes in the disparity images (Fig. 7). However, the GPU-OPENCV algorithm always reconstructs more points than the other algorithms.

Figure 7 also shows that the CPU-GPU version can cope with disparity discontinuities. In the "circle" frame, the rear part of the virtual liver is partially occluded. The approach produces only some mismatches directly at the edge of the occlusion. While using the CPU-GPU version, the mean distance is always below 0.6 mm and below 0.2 mm for sequences with a textured liver.

In addition, we investigated the influence of noise and calibration errors using stereo endoscopic images (Table III). In the first experiment, the images were altered by adding zero-centred Gaussian noise with variance $\sigma = 10$ to all colour channels. To generate the noise, we used the polar form of the Box-Muller transform.[44] The CPU-GPU version always outperforms the CPU and GPU version (at least 39% more accurate), but its mean disparity error and the mean distance are worse compared to the CPU-OPENCV algorithm. In the "circle" sequence, the mean distance of the CPU-OPENCV algorithm is about 35% better whereas the "deform" sequence shows an increase in accuracy of about 8%. An easy way to improve the results is to apply a 3×3 Gaussian filter to the images. Here, the impact of the noise is strongly reduced. This leads to a performance comparable to the CPU-OPENCV algorithm which does not benefit from the smoothing step. Also the number of reconstructed points is about 10% higher. The GPU-OPENCV algorithm always performs much poorer than the CPU-GPU algorithm, however, it also strongly profits from Gaussian smoothing. Its advantage is a significantly larger number of reconstructed points (8% and 10%, respectively).

For the second experiment, we added a rectification error of 0.5 and 1 pixels to the images. This means that the second image is shifted vertically. Rectification errors are common in endoscopic images and are caused by either a poor camera calibration or by a not fully synchronized capturing of the left and right image. As result, the disparity images are noisier, but the overall mean distance error remains in an acceptable range of well under 0.5 mm for all algorithms. The CPU-GPU algorithm outperforms the CPU algorithm here as well. The CPU-OPENCV algorithm performs slightly better for a rectification error of 1 pixel, but fewer points are reconstructed. Most points are reconstructed by the GPU-OPENCV algorithm, yet its mean distance error is the worst of all.

To summarize, most errors arise with the untextured liver and for noisy sequences. Here, the GPU version produces unacceptable noisy point clouds. In sequences with a well textured liver and no or small disturbances, its surface reconstruction is always as good as in the other versions. The CPU-GPU version performs better than the GPU and CPU version, especially in experiments with the untextured liver or image disturbances. The overall mean distance error for the CPU-GPU version remains permanently under 1 mm. It is more accurate than both OPENCV algorithms and almost as robust as the CPU-OPENCV algorithm toward image disturbances. Only the zoom sequence with an untextured model delivers significantly poorer results due to the lack of valid temporal information.

III.B. Evaluation on stereo endoscopic images

III.B.1. Image sequences of a silicone phantom

In this evaluation, we used two sequences (Heart 1 with 2426 and Heart 2 with 3388 image pairs) of a silicone phantom with an image resolution of 320×288. They can be

TABLE II. Evaluation of the CPU, GPU, CPU-GPU, the CPU- and GPU-OPENCV surface reconstruction on nine image sequences from the simulation environment; best results in bold print.

			Circle			Zoom			Deform		
Evaluation criteria	Algorithm	Pixel used	Tex 1	Tex 2	Tex 3	Tex 1	Tex 2	Tex 3	Tex 1	Tex 2	Tex 3
% of pixel with disparity difference >1 pixel	CPU	all	1.1	1.1	23.6	0.8	0.7	25.5	0.4	0.4	2.9
	GPU		4.9	2.5	49.7	1.3	0.7	56.0	**0.3**	**0.3**	16.0
	CPU-GPU		**0.6**	**0.7**	**8.7**	**0.5**	**0.4**	**10.4**	**0.3**	**0.2**	**1.4**
Mean disparity error between reconstructed and ground truth disparity image (pixels)	CPU	all	0.24	0.25	1.77	**0.22**	0.21	2.05	0.21	0.21	0.33
	GPU		0.84	0.53	12.00	0.25	0.21	15.1	0.20	0.21	5.93
	CPU-GPU		**0.18**	**0.19**	**0.53**	**0.18**	**0.17**	**1.03**	**0.14**	**0.13**	**0.29**
	CPU	from reconstructed points	0.11	0.10	1.24	**0.09**	**0.07**	2.04	0.11	0.10	0.23
	GPU		0.37	0.17	8.17	0.10	0.08	13.9	0.12	0.13	3.50
	CPU-GPU		**0.10**	**0.09**	**0.37**	**0.09**	0.08	0.56	**0.08**	**0.08**	**0.15**
	CPU-OPENCV		0.40	0.31	0.54	0.18	0.14	**0.30**	0.21	0.16	0.34
	GPU-OPENCV		0.29	0.29	0.42	0.26	0.27	0.34	0.26	0.26	0.34
Mean distance of reconstructed points to virtual liver model (mm)	CPU	from reconstructed points	0.08	0.08	0.72	**0.06**	**0.06**	2.81	0.08	0.07	0.16
	GPU		0.16	0.10	4.08	0.08	**0.06**	5.70	0.08	0.08	1.33
	CPU-GPU		**0.07**	**0.07**	**0.28**	0.07	**0.06**	0.39	**0.06**	**0.05**	**0.11**
	CPU-OPENCV		0.23	0.18	0.45	0.12	0.09	**0.22**	0.14	0.11	0.23
	GPU-OPENCV		0.21	0.23	0.45	0.18	0.19	0.27	0.14	0.16	0.21
Number of reconstructed points (%)	CPU		81.7	81.9	75.8	85.1	85.2	73.6	85.8	85.8	85.7
	GPU		78.6	80.3	64.1	84.5	84.8	58.0	85.8	85.6	74.4
	CPU-GPU		82.4	82.6	81.7	85.3	85.3	82.5	85.8	85.7	85.8
	CPU-OPENCV		69.7	70.0	58.7	73.6	73.7	61.1	73.6	73.7	70.0
	GPU-OPENCV		**84.0**	**84.1**	**83.2**	**87.6**	**87.6**	**86.5**	**88.2**	**88.4**	**87.6**

TABLE III. Errors due to two error sources: Gaussian noise and poorly rectified images.

Error source		Evaluation criteria	Algorithm	Deform Tex 1	Circle Tex 2
Noise variance	Noise variance $\sigma = 10$	Mean distance (mm)	CPU	0.66	1.39
			GPU	0.83	1.94
			CPU-GPU	0.40	0.46
			CPU-OPENCV	**0.37**	**0.30**
			GPU-OPENCV	0.65	1.80
	Noise variance $\sigma = 10$ (with Gaussian smoothing)	Mean distance (mm)	CPU	0.37	0.51
			GPU	0.46	0.77
			CPU-GPU	**0.31**	**0.32**
			CPU-OPENCV	0.34	0.33
			GPU-OPENCV	0.38	0.58
		Reconstructed points (%)	CPU	80.4	74.0
			GPU	75.9	67.1
			CPU-GPU	79.0	78.1
			CPU-OPENCV	69.9	65.4
			GPU-OPENCV	**87.5**	**83.5**
Rectification error	0.5 pixel	Mean distance (mm)	CPU	0.19	0.19
			GPU	0.20	0.22
			CPU-GPU	**0.14**	**0.17**
			CPU-OPENCV	**0.14**	0.21
			GPU-OPENCV	0.22	0.30
	1 pixel	Mean distance (mm)	CPU	0.30	0.39
			GPU	0.32	0.46
			CPU-GPU	0.27	0.34
			CPU-OPENCV	**0.20**	**0.29**
			GPU-OPENCV	0.38	0.48
		Reconstructed points (%)	CPU	85.6	80.9
			GPU	85.1	76.3
			CPU-GPU	85.5	81.9
			CPU-OPENCV	73.6	69.6
			GPU-OPENCV	**88.1**	**83.5**

acquired on the VIP Laparoscopic/Endoscopic Video Dataset page.[13,40,45] Ground truth data are obtained by using CT data which are manually registered to the endoscopic images. Each sequence has 20 ground truth point clouds associated with it. Each point cloud is used for multiple image pairs. We used the CPU, the CPU-GPU, and both OPENCV workflows to reconstruct point clouds from the images. For each sequence, we measured the mean distance error between the reconstructed and the ground truth point cloud. We also calculated the mean distance error separately for each ground truth data set (Fig. 8). Here, the overall error is influenced by our disparity calculation, but also by errors in rectification and 3D reconstruction, due to inaccuracies in the camera calibration. Additional errors arise due to the manual registration.

Results in Table IV show that the mean distance is significantly larger than the error from the evaluation using the virtual simulation environment. This is explained by the additional error sources introduced here. The large difference in error between the best and the worst ground truth data set also implies that especially the manual registration causes a significant part of the total error. Also, the rectification error changes from image pair to image pair. Interestingly, the CPU-GPU version performs better than the CPU-OPENCV algorithm on sequence "Heart 1" (mean distance error about 29% reduced), but worse on sequence "Heart 2" (mean distance error about 7% increased). The best data sets are also not the same for the algorithms, which indicates that they perform differently depending on the images. Again, the CPU-OPENCV algorithm reconstructs fewer points than the

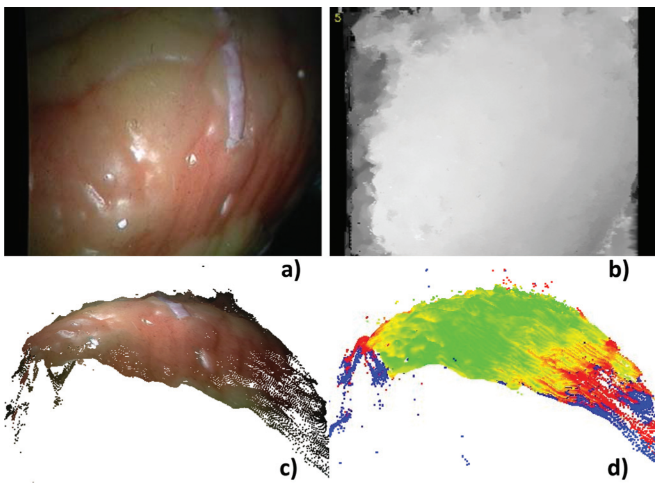

FIG. 8. Heart 1 sequence: (a) frame 5; (b) disparity image; (c) Reconstructed point cloud; and (d) Color-coded distance map (from green (error = 0 mm) to red (error = 2 mm)) between reconstructed surface and ground truth: blue = no comparable points found.

TABLE IV. Mean distance between reconstructed and ground truth point cloud for two image sequences of the silicone phantom and the CPU version, the CPU-GPU version and both OPENCV algorithms.

Ground truth data sets used	Heart 1 mean distance error (mm)				Heart 2 mean distance error (mm)			
	CPU	CPU-GPU	CPU-OPENCV	GPU-OPENCV	CPU	CPU-GPU	CPU-OPENCV	GPU-OPENCV
All ground truth data sets	1.55	**1.45**	1.91	3.29	1.61	1.64	**1.53**	2.70
Best data set (ground truth data set number)	0.80 (12)	**0.77** (12)	1.09 (11)	2.40 (15)	0.84 (12)	**0.75** (15)	0.82 (16)	1.97(15)
Worst data set (ground truth number)	2.91 (5)	**2.62** (5)	3.50 (5)	4.33 (5)	2.86 (6)	2.96 (6)	**2.70** (6)	3.60(8)
Reconstructed points (%)	66.2	65.7	60.8	**77.3**	66.5	**69.8**	58.0	69.6

CPU-GPU algorithm. The difference between the CPU and the CPU-GPU version is not significant, which indicates that when using a low image resolution, the robustness of the two versions is comparable. The mean distance error from the GPU-OPENCV algorithm is significantly poorer in both sequences.

Overall, the mean distance stays in a range of well under 2 mm for 16 of the 20 data sets in sequence "Heart 1" and 14 of 20 data sets in sequence "Heart 2". Figure 8 also shows that the approach can cope with image artefacts, such as the specular reflections in the middle of the image.

III.B.2. Image sequences from daVinci interventions

Due to the lack of intraoperative ground truth data, we could only qualitatively evaluate *in vivo* images. In the image sequences, we especially looked at the impact of image disturbances caused by specular reflections, smoke or interlacing artefacts, as well as the overall noisiness of the disparity images and the reconstructed point clouds.

When using images from daVinci interventions (Fig. 1), the correspondence analysis of the CPU-GPU algorithm produces smooth disparity images. Most mismatches arise at the image margin, an area that is less important for surface registration. Most areas that are not reconstructed are small. Artefacts mainly arise when the images are filled with smoke, which is caused by tissue ablation, or suffer from interlacing effects, due to fast movements of objects. However, they are corrected in the next time steps, as the algorithm optimizes its disparity image over time. The disparity of specular pixels can be reconstructed quite well from the spatial and temporal neighborhood, as long as the specular reflection remains sufficiently small. Only large specularities lead to holes in the reconstructed surface. Occluded areas can be reconstructed by propagating the disparity values from neighboring nonoccluded areas into them. Although the baseline between the cameras is small, the 3D points in the point cloud are smoothly distributed.

The CPU-OPENCV algorithm also produces smooth disparity images, but there are more large holes or areas with smooth, but inaccurate disparity values in the images. When smoke is present, the images of the CPU-OPENCV algorithm are noisier than the images resulting from the CPU-GPU algorithm. It also has problems reconstructing regions with very homogeneous texture, e.g., regions covered with blood. In these cases, the temporal candidate of the HRM and its recursive structure is especially beneficial. The CPU-OPENCV algorithm also has more problems reconstructing dark areas, but it deals with specularities quite similarly to the CPU-GPU algorithm. Edges at object boundaries, e.g., instruments, are reconstructed more precisely by the CPU-OPENCV algorithm. Here, the reconstruction of the CPU-GPU version displays a larger number of frayed edges.

The GPU-OPENCV algorithm is less robust than the CPU-GPU algorithm. Homogeneous areas and the border area of the images are especially problematic. It is also more sensitive to smoke or blood. Like its CPU counterpart, it is more accurate near object boundaries.

III.C. Runtime evaluation

The runtime was evaluated using sequences with resolutions of 320×240, 640×480, and 960×540 (Table V). We only used the first field of the images of the HD-endoscope to avoid interlacing, which has a more severe impact on the surface reconstruction with a higher image resolution. We tested the performance on an Intel i7 930 CPU with four cores and 12 GB RAM and an NVIDIA Tesla C 2070 graphic card. The CPU version of the algorithm is parallelized using OPENMP to capitalize on the multicore architecture. All versions use the same parameters. Preprocessing and visualization time is not taken into account. As both OPENCV algorithms only perform disparity calculation, we use the GPU implementation of the 3D reconstruction methods for surface reconstruction.

The GPU implementation always outperforms the CPU implementation of the algorithms. Especially the GPU bilateral filter calculation is improved upon compared to the CPU version (about 30 times faster when using a resolution of 640×480). The GPU HRM shows the least acceleration due to its recursive elements, but is still faster than its CPU counterpart. The disparity correction time varies a great deal. The speed depends on the number of mismatches found after the HRM step. This also explains why the disparity correction of the hybrid CPU-GPU algorithm is faster than its GPU counterpart. Here, some mismatches are already corrected in the CPU step. The disparity refinement step, which includes bilateral filtering, is significantly faster on the GPU than on the CPU. The hybrid CPU-GPU algorithm speed was evaluated on 640×480 and 960×540 images. For lower image resolutions, the CPU and GPU versions are directly combined without reducing the image resolution for the CPU step. The additional calculation of the bisected images on the CPU costs a bit of performance, but it is still much faster than the CPU version. The correspondence analysis, disparity correction and

TABLE V. Run time comparison in ms between the CPU, the GPU, and the CPU-GPU version of the algorithms as well as both OPENCV correspondence analysis algorithms for left and right image with resolution of 320 × 240, 640 × 480, and 960 × 540.

	320 × 240				640 × 480					960 × 540				
Algorithm	CPU	GPU	CPU-OPEN-CV	GPU-OPEN-CV	CPU	GPU	CPU-GPU	CPU-OPEN-CV	GPU-OPEN-CV	CPU	GPU	CPU-GPU	CPU-OPEN-CV	GPU-OPEN-CV
Correspond. analysis (ms)	$\sim$17	9	$\sim$30	$\sim$41	$\sim$85	25	25	$\sim$225	$\sim$125	$\sim$140	34	34	$\sim$340	$\sim$200
Disp. correction (ms)	5-10	2			$\sim$50	7	5			$\sim$90	10	7		
Disp. refinement (ms)	14	2			$\sim$170	6	6			$\sim$200	10	10		
3D reconstr. (ms)	2	1	1	1	7	1	1	1	1	12	2	2	2	2
Least squares (ms)	12	1	1	4	$\sim$35	4	4	4	4	$\sim$75	14	14	14	14
Copy from and to GPU (ms)	x	1	1	1	x	2	3	2	1	x	3	4	3	1
CPU part of CPU-GPU (ms)	x	x	x	x	x	x	26	x	x	x	x	37	x	x
Overall (fps)	19	60	30	22	3	21	14	4.5	7.5	1.5	13	9	2.5	4.5

refinement steps of the GPU and the hybrid CPU-GPU algorithms are always faster than both OPENCV algorithms.

IV. DISCUSSION

In this paper, we presented an approach toward reconstruction of a dense surface from stereo endoscopic image sequences in real-time. It depicts accurately the constantly deforming environment and can be used to build an intraoperative model of the surgical site, which can be registered to preoperative planning data. The correspondence analysis is able to produce a dense disparity image by finding corresponding point pairs in a recursive process, while using information from the spatial and temporal neighborhood. The disparity correction detects mismatches in both disparity images and corrects them using several correction methods depending upon the characteristic properties of their surroundings. Bilateral filtering is used to refine the disparity images. The filter smoothes the disparity images efficiently, while preserving edges. In the 3D reconstruction step, remaining outliers are filtered to get a more robust point cloud. This point cloud is then smoothed and normal vectors for each point are extracted using the Total Least Squares normal calculation method. The result is a filtered and smoothed point cloud with surface normals for each point, which closely describes the original surface and can be used for surface registration.

We focused on the GPU implementation of the algorithms, in order to get a system that works in real-time when using higher image resolutions. Thus, we presented a solution to transfer the recursive correspondence analysis method to the GPU. We also gave a detailed explanation of our GPU version of the disparity correction. Finally, we presented a new hybrid CPU-GPU algorithm to perform surface reconstruction, which benefits from the modular structure of our workflow.

We compared the CPU, the GPU, and the hybrid CPU-GPU version of the surface reconstruction with algorithms from the literature, which are implemented in the OPENCV library. The GPU version works in real-time for an image resolution of 640 × 480, but is sensitive to large areas without texture, image noise or large movements in the images. The CPU version is more robust, but is considerably slower. This fact also reduces its performance quality because as the update rate slows, temporal information will become more and more unreliable. The hybrid CPU-GPU version combines their advantages. It is more robust than the CPU version and only slightly slower than the GPU version. It shows significantly better results especially for low-textured areas. The reason for this is that each image is now processed twice, which improves upon the outcome due to its temporal character. Its robustness is comparable to the CPU -based OPENCV algorithm, but it has a higher accuracy, reconstructs more points and is significantly faster. The GPU-based OPENCV algorithm is less robust and less accurate. Overall, the hybrid CPU-GPU version reconstructs the surface down to submillimeter accuracy in real-time. It is robust toward regions without texture, fast and large movements in the images, specular reflections and errors due to a poor image rectification.

Future work includes the improvement of the approach near discontinuities and occlusions where disparities tend to leak out in neighboring regions without texture. While this effect can be reduced by repeating the method on the same images, it is still not always satisfactory. We also have to be able to separate objects from the background, e.g., instruments, in order to remove them from the point cloud. For this, we need robust instrument detection. In addition, image artifacts caused by smoke or blood have to be detected. We also have to implement efficient solutions to merge surface models from different time steps.

The main goal for the future is to successfully register the reconstructed surface with a preoperative model from the surgical site. Therefore, further examination on nonrigid surface registration methods is required.

ACKNOWLEDGMENTS

The present research was conducted in correlation with the "Research training group 1126: Intelligent Surgery—Development of new computer-based methods for the future workplace in surgery" founded by the German Research Foundation. Furthermore, it is sponsored by the European Social Fund through the state of Baden-Württemberg. The authors thank the Visual Information Processing Group at the Imperial College in London for their help with evaluating data sets. The authors also thank FA Richard Wolf and the NVidia Academic Partnership program for their support.

[a)]Author to whom correspondence should be addressed. Electronic mail: roehl@kit.edu; Telephone: + 49 (0)721 60848487; Fax: + 49 (0)721 60848270.

[1]R. Maciunas, "Computer-assisted neurosurgery," Clin. Neurosurg. **53**, 267 (2006).

[2]A. Malti, A. Bartoli, and T. Collins, "Template-based conformal shape-from-motion from registered laparoscopic images," *Proceedings of Medical Image Understanding and Analysis Conference* (IEEE Computer Society, Washington, DC, 2011).

[3]S. Lee, M. Lerotic, V. Vitiello, S. Giannarou, K. Kwok, M. Visentini-Scarzanella, and G. Yang, "From medical images to minimally invasive intervention: Computer assistance for robotic surgery," Comput. Med. Imaging Graph. **34**(1), 33–45 (2010).

[4]P. Sánchez-González, A. Cano, I. Oropesa, F. Sánchez-Margallo, F. Del Pozo, P. Lamata, and E. Gómez, "Laparoscopic video analysis for training and image-guided surgery," Minimally Invasive Ther. Allied Technol. **20**(6), 311–320 (2011).

[5]P. Mountney and G. Yang, "Motion compensated slam for image guided surgery," *Proceedings of MICCAI* (MICCAI Society, Minnesota, 2010), Vol. 6362, pp. 496–504.

[6]J. Ackerman, K. Keller, and H. Fuchs, "Surface reconstruction of abdominal organs using laparoscopic structured light for augmented reality," Proc. SPIE Med. Imaging **4661**, 39–46 (2002).

[7]N. Clancy, D. Stoyanov, G. Yang, and D. Elson, "An endoscopic structured lighting probe using spectral encoding," Proc. SPIE Med. Imaging **8090**, 809002 (2011).

[8]J. Penne, K. Höller, M. Stürmer, T. Schrauder, A. Schneider, R. Engelbrecht, H. Feußner, B. Schmauss, and J. Hornegger, "Time-of-flight 3-d endoscopy," *Proceedings of MICCAI* (MICCAI Society, Minnesota, 2009), Vol. 5761, pp. 467–474.

[9]http://vision.middlebury.edu/stereo/

[10]C. Wu, Y. Sun, and C. Chang, "Three-dimensional modeling from endoscopic video using geometric constraints via feature positioning," IEEE Trans. Biomed. Eng. **54**(7), 1199–1211 (2007).

[11]M. Hu, G. Penney, M. Figl, P. Edwards, F. Bello, R. Casula, D. Rueckert, and D. Hawkes, "Reconstruction of a 3D surface from video that is robust to missing data and outliers: Application to minimally invasive surgery using stereo and mono endoscopes," Med. Image Anal. **15** (2011).

[12]P. Mountney and G. Yang, "Soft tissue tracking for minimally invasive surgery: Learning local deformation online," *Proceedings of MICCAI* (MICCAI Society, Minnesota, 2008), Vol. 5242, pp. 364–372.

[13]D. Stoyanov, M. Scarzanella, P. Pratt, and G. Yang, "Real-time stereo reconstruction in robotically assisted minimally invasive surgery," *Proceedings of MICCAI* (MICCAI Society, Beijing, China, 2010), Vol. 6361, pp. 275–282.

[14]C. Chen, D. Sargent, C. Tsai, Y. Wang, and D. Koppel, "Stabilizing stereo correspondence computation using delaunay triangulation and planar homography," LECT NOTES COMPUT SC 5358/2008, 836–845 (2008).

[15]R. Richa, A. Bo, and P. Poignet, "Towards robust 3d visual tracking for motion compensation in beating heart surgery," Med. Image Anal. **15**(3), 302–315 (2011).

[16]W. Lau, N. Ramey, J. Corso, N. Thakor, and G. Hager, "Stereo-based endoscopic tracking of cardiac surface deformation," *Proceedings of MICCAI* (MICCAI Society, Minnesota, 2004), Vol. 3217, pp. 494–501.

[17]F. Devernay, F. Mourgues, and E. Coste-Maniere, "Towards endoscopic augmented reality for robotically assisted minimally invasive cardiac surgery," *Proceedings of Medical Imaging and Augmented Reality (MIAR)* (The University of Tokio, Japan, 2001), pp. 16–20.

[18]D. Stoyanov, A. Darzi, and G. Yang, "A practical approach towards accurate dense 3d depth recovery for robotic laparoscopic surgery," Comput. Aided Surg. **4**, 199–208 (2005).

[19]B. Vagvolgyi, L. Su, R. Taylor, and G. Hager, "Video to CT registration for image overlay on solid organs," *Proceedings of Augmented Reality in Medical Imaging and Augmented Reality in Computer-Aided Surgery (AMIARCS)* (Imperial College, London, UK, 2008), pp. 78–86.

[20]S. Röhl, S. Bodenstedt, S. Suwelack, H. Kenngott, B. Müller-Stich, R. Dillmann, and S. Speidel, "Real-time surface reconstruction from stereo endoscopic images for intraoperative registration," Proc. SPIE Med. Imaging **7964**, 796414 (2011).

[21]S. Röhl, S. Speidel, D. Gonzalez-Aguirre, S. Suwelack, H. Kenngott, T. Asfour, B. Müller-Stich, and R. Dillmann, "From stereo image sequences to smooth and robust surface models using temporal information and bilateral postprocessing," *IEEE International conference on Robotics and Biomimetics (ROBIO)* (IEEE Computer Society, Washington, DC, 2011).

[22]G. Pratx and L. Xing, "GPU computing in medical physics: A review," Med. Phys. **38**, 2685 (2011).

[23]C. Chou, Y. Chuo, Y. Hung, and W. Wang, "A fast forward projection using multithreads for multirays on GPUs in medical image reconstruction," Med. Phys. **38**, 4052 (2011).

[24]H. Hofmann, B. Keck, C. Rohkohl, and J. Hornegger, "Comparing performance of many-core CPUs and GPUs for static and motion compensated reconstruction of c-arm CT data," Med. Phys. **38**, 468 (2011).

[25]L. Persoon, M. Podesta, W. van Elmpt, S. Nijsten, and F. Verhaegen, "A fast three-dimensional gamma evaluation using a GPU utilizing texture memory for on-the-fly interpolations," Med. Phys. **38**, 4032 (2011).

[26]Y. Zhuge, Y. Cao, J. Udupa, and R. Miller, "Parallel fuzzy connected image segmentation on GPU," Med. Phys. **38**, 4365 (2011).

[27]J. Wu, M. Kim, J. Peters, H. Chung, and S. Samant, "Evaluation of similarity measures for use in the intensity-based rigid 2D-3D registration for patient positioning in radiotherapy," Med. Phys. **36**, 5391 (2009).

[28]J. Spoerk, H. Bergmann, F. Wanschitz, S. Dong, and W. Birkfellner, "Fast drr splat rendering using common consumer graphics hardware," Med. Phys. **34**, 4302 (2007).

[29]P. Azad, T. Gockel, and R. Dillmann, *Computer Vision: Principles and Practice* (Elektor-Verlag, Aachen, Germany, 2007).

[30]opencv.willowgarage.com

[31]N. Atzpadin, P. Kauff, and O. Schreer, "Stereo analysis by hybrid recursive matching for real-time immersive video conferencing," *IEEE Transactions on Circuits and Systems for Video Technology* (IEEE Computer Society, Washington, DC, 2004) Vol. **14**.

[32]S. Paris, P. Kornprobst, J. Tumblin, and F. Durand, "A gentle introduction to bilateral filtering and its applications," *ACM SIGGRAPH 2007 courses* (Association for Computing Machinery, Inc., New York City, 2007), p. 1.

[33]S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," Int. J. Comput. Vis. **81**(1), 24–52 (2009).

[34]H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," Proc. SIGGRAPH **26**, 71–71 (1992).

[35]N. Mitra, N. Gelfand, H. Pottmann, and L. Guibas, "Registration of point cloud data from a geometric optimization perspective," *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (43 Association for Computing Machinery, Inc., New York City, USA 2004), pp. 22–31.

[36]www.vtk.org

[37]qt.nokia.com

[38]http://www.igstk.org/

[39]S. Bodenstedt, S. Röhl, S. Suwelack, D. Katic, H. Kenngott, B. Müller-Stich, R. Dillmann, and S. Speidel, "A flexible framework for multiple sensor integration in to a context-aware cas-system," *Computer Assisted Radiology and Surgery (CARS)* (CARS Society, Kuessaberg, Germany, 2011).

[40]P. Pratt, D. Stoyanov, M. Visentini-Scarzanella, and G. Yang, "Dynamic guidance for robotic surgery using image-constrained biomechanical models," *Proceedings of MICCAI* (MICCAI Society, Minnesota, 2010), Vol. 6361, 77–85.

[41]H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information," IEEE Trans. Pattern Anal. Mach. Intell. **30**(2), 328–341 (2008).

[42]Q. Yang, L. Wang, and N. Ahuja, "A constant-space belief propagation algorithm for stereo matching," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE Computer Society, Washington, DC, 2010), 1458–1465.

[43]G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library* (O'Reilly Media, Sebastopol, 2008).

[44]W. Press *et al.*, *Numerical Recipes* (Cambridge University Press, Cambridge, UK, 2007), Vol. 3.

[45]http://hamlyn.doc.ic.ac.uk/vision/