GPU-Accelerated Robotic Intra-operative Laparoscopic 3D Reconstruction

Markus Moll[1], Hsiao-Wei Tang[2], and Luc Van Gool[1,3]

[1] Katholieke Universiteit Leuven, ESAT-PSI/IBBT,
Department of Electrical Engineering, Heverlee, Belgium
{Markus.Moll,Luc.VanGool}@esat.kuleuven.be
[2] Katholieke Universiteit Leuven, PMA,
Department of Mechanical Engineering, Heverlee, Belgium
[3] Swiss Federal Institute of Technology (ETH), D-ITET/Computer Vision Laboratory, Zürich, Switzerland
VanGool@vision.ee.ethz.ch

Abstract. In this paper we present a real-time intra-operative reconstruction system for laparoscopic surgery. The system builds upon a surgical robot for laparoscopy that has previously been developed by us. Such a system is valuable for surgeons, who can get a three dimensional visualization of the scene online, without having to postprocess data. We gain a significant speed increase over existing such systems by carefully parallelizing tasks and using the GPU for computationally expensive subtasks, making real-time reconstruction and visualization possible. Our implementation is also robust with respect to outliers and can potentially be extended to be used with non-robotic surgery. We demonstrate the performance of our system on ex-vivo samples and compare it to alternative implementations.

1 Introduction

Minimally invasive surgery (MIS) has become an important tool in surgery, as it greatly reduces the risks for the patient and leads to faster over-all recovery. In MIS, the surgeon operates through small incisions only, where a view into the body is provided through an endoscope. We are specifically dealing with laparoscopy, which is MIS in the abdominal. Nowadays, such images are usually taken by a video camera mounted on the endoscope and are then displayed on a separate screen. Robot assisted surgery (RAS) has also become more and more popular in this field, where today Intuitive Surgery is dominating the market with their Da Vinci system.

The downside of MIS is that it requires a lot of training from the surgeon. This is mainly caused by the indirect interaction and feedback, with the surgical instruments and the video camera being the surgeons hands and eyes. A great part of the problem is caused by the unusual vision that an endoscope provides. Firstly, the perception of three dimensional structure is greatly reduced because the image is displayed on a screen. Secondly, the surgeon usually only sees a small part of the body cavity, which makes it harder to navigate.

N. Navab and P. Jannin (Eds.): IPCAI 2010, LNCS 6135, pp. 91–101, 2010.

For the former problem, stereo endoscopes have been introduced but have not become widely accepted. The reason for that is that the only reasonable way to display stereoscopic video is by using goggles, which isolate the surgeon from his surroundings.

The computer vision community has actively worked on solutions for these problems, notably by Koppel et al. [6,7] and Wengert [13]. The early work of Koppel et al. concentrates on estimating camera pose to correct the image orientation. To achieve this, only a small number (around 20) of features are tracked using a window search tracker. Full window search is relatively slow, so they use FFT to speed it up. They also describe how to create novel views by fitting a low order polynomial to these points. Wengert's work consists of an intra-operative system for 3D reconstruction relying on camera pose estimates from an external optical tracker. His system works at 2–8 frames per second for a resolution of 384x288 pixels, depending on the number of features[1]. The near real-time speed is achieved by using SURF features[2] and the efficient selection of candidate matches in the feature matching stage. He already noted that off-loading work to a graphics processor could yield a great speed up.

In recent years, graphics processors (GPUs) have become more and more powerful and at the same time more and more versatile. Graphics processors feature a large number of processor cores running with shared memory. They have virtually no caches when compared to general purpose processors, and the memory latency is instead hidden by the strong parallelism. In earlier GPUs, some of these processors could only perform special tasks, but in modern *unified shader architectures*, all these processors can share the same workload.

We use the GPU to gain a significant performance increase over previous implementations. This allows us to process the image stream at video-rate, i.e. at 25 frames per second. To the best of our knowledge, our system is the only such system achieving real-time performance. We have implemented an affine feature drift prevention mechanism in Cg, Nvidia's shader language, as feature drift severly impacts the quality of the reconstruction. We use GPU and CPU in a multi-threaded fashion, so that GPU and CPU work can actually be done in parallel and very efficiently be pipelined (see Figure 1), allowing us to—whenever necessary—spend more computation time per frame at the expense of increased latency. We have also implemented a proper way to deal with outlying points in the bundle adjustment step, as outlying points tend to still occur even in the presence of drift prevention, e.g. through moving highlights on otherwise uniform surfaces.

2 Methods

We start by giving an overview of the individual units of our system. After an image from the video stream is captured with a frame grabber, we correct for lens distortion, which tends to be quite large in endoscopes. This simplifies the

[1] For example, a scene with on average 251 features resulted in a frame rate of 4.5 frames per second.

subsequent tracking and 3D reconstruction which assume that local image patch deformations are approximately affine. As this sampling step can be efficiently performed on the GPU, its cost is negligible. From the undistorted image, we extract image patches and subsequently track them. We have used our own surgical robot to move the endoscope. From this robot, we constantly obtain position estimates. With the information from the new image frame, we triangulate new points where this is possible and refine existing points non-linearly. In the end, we create a triangle mesh from these points for visualization. We project the current video frame onto that mesh to texture it, very much like a video beamer. In the following subsections we will give more detailed information on some of these steps.

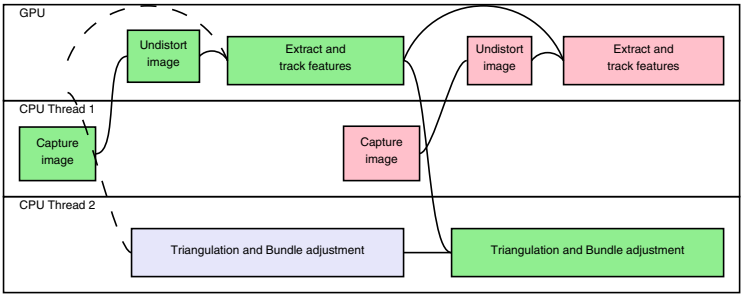


Fig. 1. Overview of the reconstruction part with data dependencies. GPU and CPU processing can be pipelined efficiently to increase throughput.

2.1 Image Undistortion

Typical endoscopic images show a fairly large amount of image distortion. Because this interferes with the remaining processing steps, the very first thing we do is to undistort these images on the GPU. Our distortion model consists of all three of radial distortion, tangential distortion and thin prism distortion. Let $\mathbf{x}_d = (x_d, y_d)^T$ be a point in the distorted input image and $\mathbf{x}_u = (x_u, y_u)^T$ be the corresponding point in the undistorted image, then the relation between these is commonly modeled as

$$\mathbf{x}_d = \mathbf{x}_u + \delta_{\text{radial}}(\mathbf{x}_u) + \delta_{\text{tangential}}(\mathbf{x}_u) + \delta_{\text{prism}}(\mathbf{x}_u) \tag{1}$$

$$\delta_{\text{radial}}(\mathbf{x}_u) = (k_1\|\mathbf{x}_u\|^2 + k_2\|\mathbf{x}_u\|^4 + k_3\|\mathbf{x}_u\|^6)\mathbf{x}_u \tag{2}$$

$$\delta_{\text{tangential}}(\mathbf{x}_u) = \begin{pmatrix} 2t_1x_uy_u + t_2(3x_u^2 + y_u^2) \\ t_1(x_u^2 + 3 * y_u^2) + 2t_2x_uy_u \end{pmatrix} \tag{3}$$

$$\delta_{\text{prism}}(\mathbf{x}_u) = \mathbf{p}\|\mathbf{x}_u\|^2 \tag{4}$$

Here, k_i, t_i and $\mathbf{p}$ are the distortion parameters which have to be estimated in advance.

2.2 Feature Extraction and Tracking

For feature tracking we use the classic Lucas-Kanade tracking method based on Shi and Tomasi's interest point detector (KLT)[12]. This has turned out to be robust with respect to the relatively high image noise, and can also be computed very efficiently. Shi and Tomasi define key points as points which are reliably trackable, i.e. in whose neighborhood there is considerable gradient variation. If $\Omega(\mathbf{x})$ denotes a small window around the point $\mathbf{x}$ in the image I, then key points are extracted at local minima of the smaller eigenvalue of

$$\sum_{\mathbf{z}\in\Omega(\mathbf{x})} \left(\nabla I|_{\mathbf{z}}\right)^T \nabla I|_{\mathbf{z}}$$

In contrast to e.g. SURF[2] or SIFT[9] features which come with discriminative descriptors, KLT features cannot reliably be redetected once they are lost, as the Shi and Tomasi detector has relatively low repeatability and the descriptors (the local image patches) are not invariant to any geometric or photometric changes.

However, when camera poses are externally measured, a feature loss can be dealt with by simply starting with a new set of tracks, assuming that the external camera pose correctly aligns the two otherwise independent reconstructions. It is also worth noting that in that case the matching cost is typically quadratic and thus rather costly. If the only goal is to collect tracks to build a 3D model, KLT features promise to be faster and most robust, as has also been shown for example in [5].

We have chosen to perform both feature tracking and interest point detection on the GPU, as prior experiments with a CPU based implementation on top of OpenCV have shown that these steps are the most time consuming. Tracking naturally lends itself to parallelization because all points can be tracked independently. This fits modern GPUs perfectly, because they are designed for data parallelism, where the same task has to be performed on multiple data. As one of the most time consuming steps is the accumulation of gradients over a local window, a GPU implementation further benefits from the GPU's texture units, which are designed for image sampling tasks. We have implemented a GPU algorithm of the drift-resistant KLT tracker from Zinsser et al.[15], based on the GPU tracker by Zach[14]. We refer to the respective authors for a more detailed discussion of the GPU KLT implementation.

Feature drift not only occurs in very long sequences, but also during lighting changes, especially in the presence of specularities, which are quite common in endoscopic surgery. The tracking method of Zinsser et al. continuously estimates an affinely distorted registration of the feature's very first image patch and the patch in the current image, and discards tracks if the difference becomes too large, thus avoiding feature drift. To this end, they use an affine lighting model such that if $I(\mathbf{x})$ are the points in the first patch and $J(\mathbf{x})$ are those in the most recent patch an iterative update

$$\underset{\Delta A,\Delta\mathbf{b},\alpha,\beta}{\arg\min} \sum_{\mathbf{x}} \left(\alpha I(\Delta A\,\mathbf{x} + \Delta\mathbf{b}) + \beta - J(\mathbf{x})\right)^2$$

is found. ΔA and $\Delta \mathbf{b}$ are small updates to an estimated affine distortion $A\mathbf{x} + \mathbf{b}$ of the patch and α and β are used to adjust for intensity changes. The resulting linear least squares problem is solved by a Gauss-Newton iteration. We have implemented the above algorithm in the Cg shader language, taking care of the data layout to benefit from the GPU SIMD instructions.

2.3 Bundle Adjustment

While the robot provides the vision system with pose estimates, it is not stiff enough for these estimates to be correct. The problem is magnified by the relatively large leverage from the endoscope, in that small angular errors in the robot readings can have a huge impact on the camera position. Furthermore, initially point positions are chosen so as to minimize errors in the views that contribute to their creation. If the point track continues, errors in additional views will not be taken into account. Therefore, it is usually advisable to optimize both structure and motion through bundle adjustment. We employ local bundle adjustment, which has proven to be useful in online problems [10,4]. In local bundle adjustment, only a small subset of the most recent views is optimized after every iteration. This reduces the problem size considerably and thus allows to find a solution in real-time.

If additive zero-mean Gaussian noise is assumed on the image position measurements, then the solution that minimizes the sum of squared image errors is the maximum likelihood estimate. However, in real settings this assumption is usually violated, as features can drift or jump even when using drift-resistant trackers, and single outlying measurement can have arbitrary influence on the estimate. We found it necessary to deal with outliers, as ignoring them resulted in substantial pose estimation errors and inferior structure estimates. A commonly chosen approach is to remove outliers prior to running the bundle adjustment. However, besides being inherently arbitrary because of thresholding, this also suffers from outlier masking problems if the initial estimate is too far from the optimum. Similar to [4], we chose to re-weight the individual image errors such that their squared norm follows a more robust error measure, i.e. for a set of 3D points $\mathbf{X}_i$ with corresponding image measurements $\mathbf{x}_{i,j}$ we solve the least squares problem

$$\min \sum_{i,j} \left\| \sqrt{\rho\left(\|\mathbf{x}_{i,j} - \Pi_j(\mathbf{X}_i)\|^2\right)} \frac{\mathbf{x}_{i,j} - \Pi_j(\mathbf{X}_i)}{\|\mathbf{x}_{i,j} - \Pi_j(\mathbf{X}_i)\|} \right\|^2 \tag{5}$$

which is equivalent to minimizing

$$\sum_{i,j} \rho\left(\|\mathbf{x}_{i,j} - \Pi_j(\mathbf{X}_i)\|^2\right) \tag{6}$$

Here, $\Pi_j(\mathbf{X}_i)$ denotes the projection of the i-th 3D point into the j-th image. We actually use two different functions ρ, the first is the Cauchy distribution likelihood, the second is Tukey's bi-weight function. The Cauchy distribution looks

similar to the Gaussian distribution at first glance, but its tail is heavier, so that gross outliers do not have as large an influence. Tukey's bi-weight function features a so-called re-descending influence function, i.e. the influence of gross outliers is actually approaching zero. It is highly efficient at Gaussian distributions but, because it completely ignores outliers above a certain threshold, it actually does not lead to a unique solution and its performance heavily depends on the initial estimate. Therefore we first use the Cauchy likelihood to converge to a better guess and only then use the bi-weight function. The remaining problem is that of finding a good scale of the data, i.e. a good outlier threshold. We use the *median absolute deviation*, which is a highly robust scale estimator.

We have added the robustness functionality and some speed improvements to the SBA package by Lourakis [8] which we use in our system.

2.4 Experiments

In order to evaluate how our system performs, we have done experiments with an apple for the sake of simplicity, and have furthermore conducted experiments on both an ex-vivo uterus and a skull.

We have used a Storz laser laparoscope on the Storz telecam SL pal system for the apple sequences, and the same endoscope in conjunction with Storz HD equipment for the other two sequences. However, in all cases only half frame PAL resolution of 384×288 pixels was recorded with a generic PAL video grabber from S-Video inputs in order to avoid interlacing.

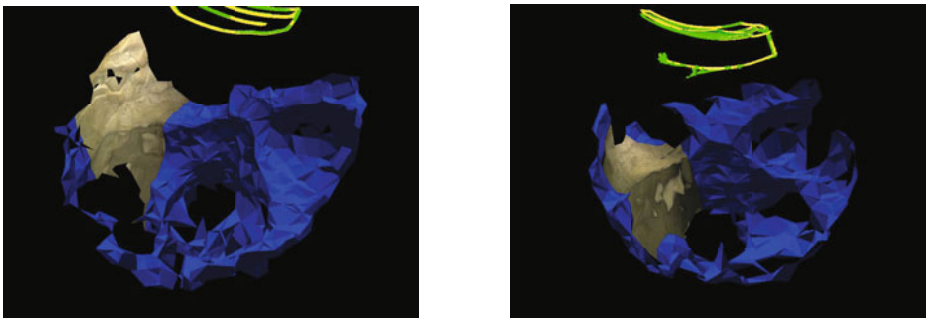
The apple experiments were run in our lab on an Intel Q9300 2.5GHz equipped with a GeForce 8800 GTS graphics board. For the other experiments, we used an Intel core i7 2.8GHz with a GeForce 8800 GTX card.

In order to compare the performance of alternative tracking methods, we have recorded video data together with pose estimates. These were replayed afterwards so that all algorithms ran on the exact same data.

3 Results

Figure 4 shows reconstructions of a skull as well as an ex-vivo uterus. The skull has also been scanned in a structured light scanner to obtain high quality 3D reconstructions that our results can be compared to. To this end, we created a point cloud model from the points that were reconstructed during the course of the whole sequence. We then manually aligned the unstructured point cloud obtained from our system to the structured light scan and found the best similarity transform of the point cloud model by the iterative closest point method (ICP). We have recorded the root mean square residual errors as a measure of goodness of these fits and thereby of the reconstructions that our system created. The results can be found in Figure 6(b).

For evaluation, we have replaced the GPU tracker in our system by a number of other tracking methods. We have used the same GPU tracker, but without the drift detection and correction as well as the OpenCV KLT implementation with and without drift correction.



(a) skull reconstruction (b) skull reconstruction

Fig. 2. Results of our system on a sequence of the base of a skull. The pyramids indicate camera position, green pyramids signify refined poses.

We attempted to compare the tracker with SIFT or SURF by mimicking a "tracker" that would only match the strongest keypoints from image to image. However, the resulting tracker failed to produce any reasonable results in our test setup. Timing and accuracy results of these alternative implementations are given in Figure 6. It is also interesting to look at the number of inlying reconstructed points at the end of the sequences in Figure 3. The affine drift correction increases the number of inliers for both GPU and CPU implementations. The relatively high number of inliers in the OpenCV implementation is due to our not restricting the proximity of newly detected features to existing ones in the OpenCV implementation. The GPU tracker prevents clustered feature points.

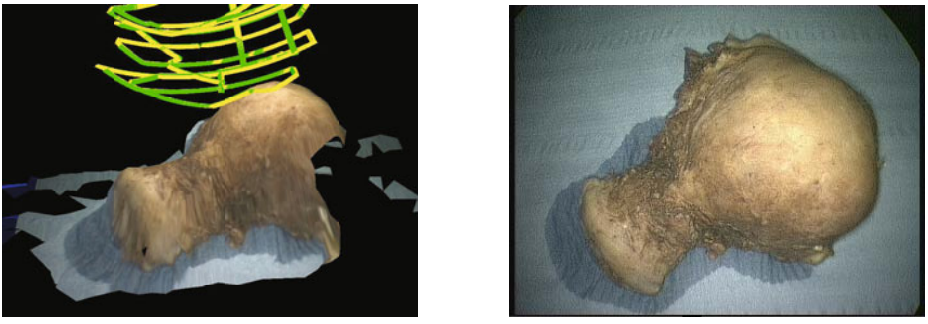
Algorithm	Uterus Sequence	Skull Sequence
GPU	7865	4945
GPU drift corr.	8658	5138
OpenCV	17542	7309
OpenCV drift corr.	21894	16217

Fig. 3. Number of inliers after reconstruction of the complete sequence

We have also artificially increased the image size to 768x576 by upsampling the image. The system still only dropped 120 of 1770 frames, achieving a framerate of on average 23–24 frames per second at full PAL resolution. However, we performed the upsampling on the graphics card so this number does not account for the larger data transfer that would normally occur.

4 Discussion

The timing results presented in the previous section show that while the GPU implementations are indeed faster, with multi-core processors the gap has become relatively small. The question might arise whether a GPU implementation



(a) uterus reconstruction (b) uterus video frame

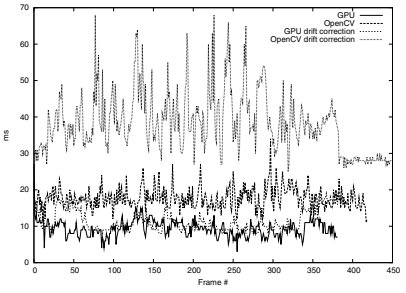
Fig. 4. Results of our system on a sequence of an ex-vivo uterus. The pyramids indicate camera position, green pyramids signify refined poses. Meshing artifacts can be seen near the ground in the uterus.

is beneficial, given that the CPU implementations do not perform much worse. However, it is important to notice that the tracking algorithms in the CPU implementations are also parallelized and made use of all available processor cores, leaving no further computing power to other processes. To us, using the GPU for highly parallelizable tasks seems very attractive as it leaves the general purpose CPU free for other purposes. Also, we could not further investigate how the runtime increases with increasing image sizes or increasing feature count and whether or not there is a significant constant overhead.

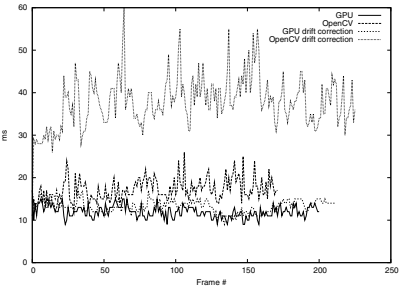
The KLT tracking performance was generally good. As can be seen in the results section, adding the drift correction step increased the number of inliers and decreased the residual error at the same time, leading to larger reconstructions with smaller errors. There is of course a certain trade-off between this improvement and the additional run-time cost one has to pay, but that has to be considered in the actual case at hand individually. It needs to be noted that we have implemented the drift correction in OpenCV ourselves and have tried to support as many border cases as the original OpenCV tracking code. Because of that, our implementation sacrifices speed for generality, so timings for that method have to be interpreted with some care.

We have not tried to evaluate a full SIFT or SURF based reconstruction method, as this would necessarily have quite a different design. Matching only temporally close, strong features was unsuccessful, so a feature database comprising a large part of the system's history would have to be built. However, as the size of that database increases, the number of candidate matches becomes larger and matching therefore takes more time.

Wengert used SURF features, but basically degraded the matching to a simple tracking by imposing strong geometric constraints on the position of candidate matches. This is however not general enough and will only work for triangulated features whose 3D positions are already known. Wengert extends this to two and three-view cases using epipolar constraints and trifocal point transfer.

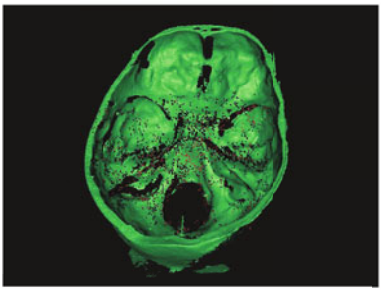


(a) Tracking time for the uterus sequence



(b) Tracking time for the skull sequence

Fig. 5. Experimental timing results



(a) The structured light scan of the skull with aligned point cloud

Algorithm	ICP error
GPU	4.790656
GPU drift corr.	2.204764
OpenCV	5.516799
OpenCV drift corr.	1.961356

(b) Residual RMS ICP errors in mm

Fig. 6. The high quality skull model used for evaluation and ICP registration errors

But the worst case number of fundamental matrices that need to be considered is quadratic in the number of views. Likewise, it is our understanding that trifocal point transfer amounts to a simple triangulation method that strongly underestimates errors in one of the two views.

Under these circumstances, KLT achieves a far higher match count. However, when it is necessary to do pose estimation as in hand-held laparoscopic surgery without any external tracking, SURF or similar features are needed for the case that all tracks are lost, for example due to blood or water in front of the lens. Wengert's system offers the advantage of the theoretical capability to be used for pose estimation, although the speed would suffer tremendously.

We feel that in these circumstances it is better to only add a few landmark SURF features that are sufficient to estimate the pose but use KLT tracking on many more features for structure estimation. This way, real-time performance can be maintained while the pose can be re-established after feature loss.

Also, currently we have to build the triangle mesh in a separate step, as the construction thereof takes too much time for integration into the system. We currently use the alpha shape reconstruction suggested by Wengert, and

have taken the implementation from CGAL[1]. However, on its own the time required for meshing is negligible and can thus be done intra-operatively as well. All meshes were created in significantly less than one second. Although the triangle mesh creation needs time, mapping the current video image to it as a live texture is fast and is integrated into the processing loop. It seems interesting to experiment with point-based visualization methods (textured splatting) in order to achieve better live visualization than displaying the point cloud.

At the current stage, the 3D visualizations seem to be a promising addition to the information provided to the surgeon, while measurements based on it should be treated with care.

5 Conclusion and Future Work

We have demonstrated a real-time intra-operative 3D reconstruction system that can provide the surgeon with additional three-dimensional information for approximately rigid structures. We are convinced that the results concerning both timing and reconstruction quality are satisfactory for the target application. While we could not test our robotic system during real surgery, we have good reason to believe that performance will be similar.

We are planning to use the extracted 3D information for robot guidance, for example to keep a fixed distance to the surface for laser ablation, or to detect and avoid collisions. We hope to further improve the reconstruction quality significantly by using higher definition video, but because of hardware limitations in the grabber device, no such videos were taken for our tests. Earlier experiments on higher resolution material lead us to think that structure estimates will greatly benefit from that. Finally we would like to extend the work beyond rigid structures by using the presented method to create a model for tracking non-rigid deformations, similar to e.g. [11].

References

1. CGAL, Computational Geometry Algorithms Library, `http://www.cgal.org`
2. Bay, H., Tuytelaars, T., Gool, L.J.V.: Surf: Speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
3. Bouguet, J.Y.: Pyramidal implementation of the lucas kanade feature tracker, description of the algorithm. Tech. rep., Intel Corporation Microprocessor Research Labs (2000) (from the OpenCV documentation)
4. Engels, C., Stewénius, H., Nistér, D.: Bundle adjustment rules. In: Proceedings of Photogrammetric Computer Vision. The International Archives of The Photogrammetry, Remote Sensing and Spatial Information Sciences (2006)
5. Klippenstein, J., Zhang, H.: Quantitative evaluation of feature extractors for visual slam. In: Proceedings of the Fourth Canadian Conference on Computer and Robot Vision (2007)
6. Koppel, D., Wang, Y.F., Lee, H.: Image-based rendering and modeling in video-endoscopy. In: Proceedings of the 2004 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, April 2004, pp. 269–272. IEEE, Arlington (2004)

7. Koppel, D., Wang, Y.F., Lee, H.: Robust and real-time image stabilization and rectification. In: Proceedings of the Seventh IEEE Workshop on Applications of Computer Vision/IEEE Workshop on Motion and Video Computing, vol. 1, pp. 320–355. IEEE Computer Society, Los Alamitos (2005)
8. Lourakis, M.I.A., Argyros, A.A.: The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Tech. Rep. 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece (August 2004), http://www.ics.forth.gr/~lourakis/sba
9. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
10. Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., Sayd, P.: 3d reconstruction of complex structures with bundle adjustment: an incremental approach. In: Proceedings of the 2006 IEEE International Conference on Robotics and Automation, May 2006, pp. 3055–3061. Orlando, Florida (2006)
11. Salzmann, M., Hartley, R., Fua, P.: Convex optimization for deformable surface 3-d tracking. In: Proceedings of the 2007 IEEE International Conference on Computer Vision (2007)
12. Shi, J., Tomasi, C.: Good features to track. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 1994), June 1994, pp. 593–600 (1994)
13. Wengert, C.: Quantitative Endoscopy. Ph.D. thesis, ETH Zürich (2008)
14. Zach, C., Gallup, D., Frahm, J.M.: Fast gain-adaptive klt tracking on the gpu. In: Proceedings of Computer Vision and Pattern Recognition Workshops, pp. 1–7 (2008)
15. Zinßer, T., Gräßl, C., Niemann, H.: Efficient feature tracking for long video sequences. In: Rasmussen, C.E., Bülthoff, H.H., Schölkopf, B., Giese, M.A. (eds.) DAGM 2004. LNCS, vol. 3175, pp. 326–333. Springer, Heidelberg (2004)