# Real-Time 3D Reconstruction in Minimally Invasive Surgery with Quasi-Dense Matching*

Georgios Zampokas, Konstantinos Tsiolis, Georgia Peleka, Ioannis Mariolis, Sotiris Malasiotis, Dimitrios Tzovaras
*Information Technologies Institute*
*Centre for Research & Technology - Hellas*
Thessaloniki, Greece
{gzampokas, ktsiolis, gepe, ymariolis, malasiot, tzovaras}@iti.gr

***Abstract*—In this work, a method for 3D reconstruction of Minimally Invasive Surgery data in real-time is presented. It is formulated on top of the already established framework of Quasi-Dense Matching, optimizing its components for speed. First, it recovers a set of sparse features, which are matched robustly. Then, 3D information is propagated in a spatial neighbourhood, until similarity reaches a predefined threshold, to cover a semi-dense portion of operating field domain. Matching on dense level is achieved with Zero Mean Normalized Cross Correlation metric to establish correspondences. The algorithm is able to recover disparity maps with relatively small error, while maintaining real-time performance.**

***Index Terms*—3D Reconstruction, Stereo Matching, Disparity Estimation, MIS, CUDA**

## I. Introduction

The problem of reconstructing the 3D geometry from arbitrary scenes or videos is a well-studied field where several algorithms have been developed. However, formulating the problem in the context of Minimally Invasive Surgery (MIS), introduces important limitations and constraints. Most of them originate from the environment of MIS, such as the presence of smoke, blood, occlusion, challenging lighting conditions and deformation of tissues caused by surgical instruments or other factors. Additionally, real-time performance, which is a prerequisite for the adoption of 3D reconstruction in real MIS, introduces an important constraint regarding execution time. As a consequence, for 3D reconstruction to be efficient and useful, a compromise between accuracy and execution time must be reached.

Considering the aforementioned constraints, we propose a massively parallel GPU adaptation of a Quasi Dense 3D reconstruction propagation algorithm, to enforce real-time performance. By tailoring the algorithm to take advantage of modern GPU capabilities and features, significant speed up is achieved. First, costly correlation computations are performed in parallel kernels. At the same time data transfers are overlapped with kernels, while being executed in parallel GPU streams. Quality of reconstruction is assured with the use of a propagation queue, which is partially sorted in every iteration of the algorithm based on the confidence of matches, in each seed's neighbourhood. This queue is explicitly stored in pinned memory, to ensure fast data transfer between the CPU and GPU.

## II. Related Work

Given the availability of a stereo endoscopes and the popularity of binocular stereo methods in various applications, 3D reconstruction problem in MIS is mainly investigated with methods belonging in the stereoscopic category. Stereoscopic methods try to estimate the 3D structure from a pair of images, which are produced from two camera sensors attached in a single setup. The most critical component in the pipeline of Stereoscopy is establishing stereo correspondences between the images. Once those correspondences have been established, the depth of the 3D points can be estimated [1], [2]. Such correspondences are found by matching pixels or higher level features between the two images so that those matches describe the same points or features in 3D space. Most feature detection and tracking methods take advantage of texture variations of the target surfaces in order to detect their location. If the variation is high enough, features can be detected and matched robustly.

Several approaches have been reported to apply stereoscopic methods to MIS data in the literature. Stoyanov [3] proposed to first establish a sparse set of correspondences of salient features and then propagate the disparity information of those salient features to nearby pixels, assuming small disparity changes between neighboring pixels. Based on this, Bernhardt [4] suggested a similar method, including three stereo matching criteria, in order to remove outliers. Penza [5] suggested two methods, based on block matching algorithm and a non-parametric modified census transform, respectively. Holes are filled and disparity is refined using a simple linear iterative clustering (SLIC). Computing descriptors and correspondences in images is often time consuming. Thus, several implementations have been introduced, which rely on executing heavy computational loads on the GPU. More specifically, Roel [6] proposed a hybrid recursive matching approach, performing a non-parametric transformation on the images. Outside of MIS context, Hernandez-Juarez [7] proposed a Semi-Global Matching approach, fully adapting the algorithm to the GPU, taking advantage of special features of modern GPUs, achieving 3D

*The research leading to these results has received funding from the European Communitys Framework Programme Horizon 2020 under grant agreement No 732515 SMARTsurg.

reconstruction with very fast real-time performance. Geiger [8] in Efficient Large Scale Stereo Matching (ELAS), suggested the use of a probabilistic Generative Model approach for stereo matching, along with a maximum-a-posteriori estimation for disparity estimation. To increase speed and robustness, the disparity search space is reduced, by building a prior over the disparity space by forming a triangulation on a set of robustly matched correspondences, named support points. As a result, accurate disparity maps of high resolution images can be computed at high frame rates.

In order to increase the accuracy of 3D reconstruction, machine learning methods have also been explored. Convolutional Neural Netowrk frameworks for feature matching and disparity estimation have been discussed in the literature [9], demonstrating promising results. However, the lack of availability of MIS data for training such networks poses a serious challenge for the adoption of such methods, which is often addressed with unsupervised learning approaches [10].

## III. Quasi-Dense 3D reconstruction

Quasi Dense Stereo Matching method is built for the reconstruction of 3D information from stereo-laparoscopic images during robotic assisted surgery [3]. It is a novel stereo semi-dense reconstruction algorithm that propagates disparity around a set of candidate feature matches. In this way, problems with specular highlights and occlusions from instruments can be eliminated. Furthermore, the algorithm can be used with any feature matching strategy allowing the propagation of depth in very disparate views. Disparity is estimated in two phases, namely Sparse Matching and Dense Matching, presented below.

### A. Sparse Matching

Sparse matching consists of a sparse 3D reconstruction base on feature matching across the stereo pair. The initial step of this method is to recover a sparse set of robust matches across the stereo-laparoscopic image pair using a feature based technique. This step includes two more sub processes. Firstly, it detects strong feature points in the left image. Detection can be achieved with several feature detection methods, namely ORB [11], SURF /citeb12 and G-SURF /citeb13. After experimentation, it is concluded that the most efficient approach is Good Features to track proposed by Shi-Tomasi [12]. It recovers corners or features in an image based on image intensity gradient. Secondly, in order to find the corresponding points in the right image, optical flow is estimated using Lucas-Kanade method [14]. This method assumes that the flow is essentially constant in a local neighbourhood of the pixel under consideration and solves the basic optical flow equations for all the pixels in that neighbourhood, by the least squares criterion. By combining information from several nearby pixels, the Lucas-Kanade method can often resolve the inherent ambiguity of the optical flow equation.

### B. Dense Matching

As a sparse set of 3D points has been established in the surgical field of view, it is possible to propagate 3D information to cover a semi-dense portion of operating field domain. It should be mentioned that all features correspondences which were calculated in the previous step, are used as seed matches. They are sorted, in descending order, based on the correlation score between their respective templates, and stored using a priority queue structure. After that, the algorithm proceeds to propagate structure around the matches correlation scores on a best-first basis popping the priority queue. As the algorithm iterates, new matches are added to the queue. When there are no matches to be popped, the algorithm terminates. If a seed match consists of a sparse pixel $p_0(x, y)$ in the left image and the corresponding pixel $p_1(x', y')$ in the right image, then a spatial neighbourhood $N(p_0, p1)$ is defined and can be used to enforce a 2D disparity gradient limit as a smoothness constraint. Thus, for each seed pixel, the spatial neighborhoods around them are defined by

$$N(p_0) = \{(x - n_x, y - n_y) : n_x, n_y \in [-N, N]\} \quad (1)$$

$$\begin{aligned} N(p_1) = \{&(x' - n_x - d_x, y' - n_y - d_y) \\ &: d_x, d_y \in [-D_g, D_g]\} \end{aligned} \quad (2)$$

where $(x - n_x, y - n_y)$ denotes the coordinates of each pixel within a spatial window of $(2N + 1) \times (2N + 1)$ pixels centered at seed pixel $p_0$ and can be matched with a candidate pixel $(x' - n_x - d_x, y' - n_y - d_y)$, which is placed within a spatial window of $(2D_g + 1) \times (2D_g + 1)$ pixels centered at $(x' - n_x, y' - n_y)$ in the right image. In conclusion, if $(U_0, U_1)$ denotes a candidate pair of pixels, then the full match propagation neighbourhood is,

$$N(p_0, p_1) = \{(U_0, U_1) : U_0 \in N(p_0), U_1 \in N(p_1)\} \quad (3)$$

The algorithm uses a dissimilarity measure during propagation in order to determine which pixels to be matched together. A very common and efficient measure is the zero mean normalized cross correlation (ZNCC), which is less prone to illumination bias in homogeneous regions while it is also more indicative in regions with discriminative texture. The range of the computed value is $[0, 1]$. Thus, the higher the ZNCC gets, the more are those two pixels correlated. The propagation stops when no more matches can be achieved, because the correlation scores are lower than a predefined threshold.

### C. Optimization for Real-time Performance

This subsection explicitly refers to the modifications that have been introduced to the original method in order to speed it up enabling real-time performance without compromising the quality of the reconstruction. This results in a modified custom disparity estimation framework, as presented in Fig. 1.

First of all, it is assumed that only rectified images are used. This means that the algorithm focuses on looking for possible matches on the horizontal dimension only, which demands fewer calculations and less memory. In this case, the equation

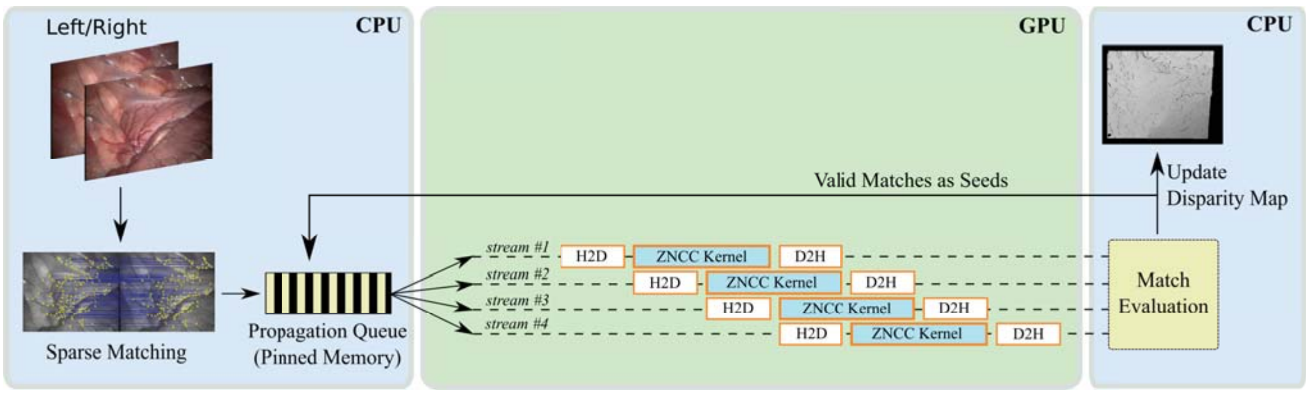

Fig. 1. Figure demonstrating the complete CPU-GPU method pipeline and data flow.

that calculates the spatial neighborhood around a seed pixel in the right image is given by

$$N(p_1) = \{(x' - n_x - d_x, y' - n_y) : d_x \in [-D_g, D_g]\} \quad (4)$$

which means that every pixel in the left image, included in $N(p_0)$, can be matched with $(2D_g + 1)$ candidate pixels in the right image.

It is also possible to exploit modern GPU technology to concurrently calculate multiple correlation windows and propagate structure over multiple pixels. Specifically, a CUDA kernel calculates all correlation scores inside a full match propagation neighbourhood defined by eq.3, by launching a block of threads for each seed match. In this way, a large number of concurrent threads that run on modern graphic cards are activated. According to the serial implementation of the method, correlation scores that are being calculated during an iteration of the algorithm refer to just one seed match. Then, the algorithm validates these scores, checks whether any of the pixels related to the potential matches have already been matched and if not, stores them in a priority queue. In order to proceed to the next iteration, the algorithm retrieves the best matching pair from the priority queue and treats it as a seed match. Subsequently, a respective full match propagation neighborhood around the seed match is calculated and the algorithm tries to find new matches within it. In this way, structure is always propagated around the best seeds. On the other hand, the proposed parallelized implementation of the method calculates the matching scores for the total number of the seeds which are available and have been stored in a simple array. The use of simple array instead of a priority queue is preferable considering the additional overhead that priority queues create because of the sorting procedure which runs in the background. However, bitonic sort [16] is applied inside the kernel so it is possible for the algorithm to choose the highest matching score for each pixel within a seed's neighborhood. The following paragraphs of this subsection describe the features of a modern GPU and how these are used by the proposed approach in order to achieve real-time performance.

**Shared memory.** Shared memory is much faster than local or global memory, because of the fact that it is on-chip. In fact, shared memory latency is roughly 100x lower than uncached global memory latency. In addition, shared memory is allocated per thread block, so all threads in a block have access to the same shared memory. This means that threads can access data in shared memory loaded from global memory by other threads within the same thread block. Concerning the parallelised implementation of the method, the coordinates of seed pixels are initially stored in global memory. Each block of threads calculates the coordinates as well as the correlation scores for every candidate pixel within the propagation neighbourhood around seed pixels and finally sorts them in respect of correlation scores. Using shared memory to store the results of these calculations significantly accelerates the whole procedure by reducing the total number of access calls in global and local memory.

**Pinned memory.** Host (CPU) data allocations are pageable by default and GPU cannot access data directly from pageable host memory. So, when a data transfer from pageable host memory to device memory is invoked, the CUDA driver must first allocate a temporary pinned host array, copy the host data to the pinned array and then transfer the data from the pinned array to device memory. Cost of the transfer between pageable and pinned host arrays can be avoided by directly allocating host arrays in pinned memory. Doing so, the data transfer rate can be increased although it depends on the type of host system (motherboard, CPU, chipset). Moreover, over-allocating pinned memory can reduce overall system performance because it reduces the amount of physical memory available to the operating system and other programs. Because of the fact that during the propagation of the structure around the matches the CUDA kernel can be called many times, data transfers must not dominate the overall execution time. Also, at the end of each kernel execution, coordinates of new potential matches have to be transferred from device to host memory in order to be validated. By allocating the appropriate arrays directly in pinned memory, removes intermediate transfers and decreases the overall execution time correspondingly.

**Overlapping kernel execution and data transfers.** A stream in CUDA is a sequence of operations that execute on the device in the order in which they are issued by the host code. While operations within a stream are guaranteed to execute in the prescribed order, operations in different streams can be interleaved and, when possible, they can even run concurrently. In addition, not only modern GPUs give the ability to execute kernel asynchronously but also transfer data. Since all operations are non-blocking with respect to the host code, multiple streams can be launched simultaneously separating the total number of calculations into equal pieces. When only one stream is used, data transfers and kernel execution are served sequentially. On the other hand, in asynchronous version when stream 1 executes the kernel, stream 2 transfers data from host to device memory (H2D). Moreover, when stream 1 transfers data back to host (D2H), stream 2 and 3 execute the kernel (ZNCC Kernel), while stream 4 transfers data to device (H2D), as illustrated in Fig. 1. Thus, this pattern is followed repeatedly and results in overlapping kernel execution and data transfers reducing the overall execution time. This technique is applied to the parallelized version of the method by dividing the initial number of seed matches by the number of streams and distributing the appropriate amount of data to them. As a result, further performance optimization has been achieved, especially when the number of seed matches is relatively high.

## IV. Evaluation

In order to assess the quality, accuracy and performance of the 3D reconstruction methods, an evaluation framework is included. We tested the performance on an Intel Xeon E5-1650 with 12 cores, with 64 GB RAM and an NVIDIA Titan X graphic card. However, the same results in performance are obtained on an Intel i5-6600 with 8GB RAM and a NVIDIA GTX 1060 GPU. Each method is evaluated over a set of datasets, with specific attributes and challenges. The first dataset is a breathing simulation sequence of a deforming silicon heart. It has an initial resolution of 360 x 288 pixels, effectively reduced to 301 x 227 after rectification. The second dataset is a set of kindney, liver and spleen phantoms captured from various poses and under varying lighting conditions (EndoAbs) [17]. This is a challenging dataset for 3D reconstruction and it has a resolution of 640 x 480 pixels (388 x 272 pixels after rectification). Both those datasets are accompanied with ground truth laser scans, providing ground truth information. The last dataset, is a video sequence of a porcine uterine horn exploration. It has a resolution of 640 x 480 pixels, reduced to 480 x 396 pixels after rectification. This dataset depicts an in-vivo sequence, therefore no ground truth is available. This variety in datasets provides us with useful insight on the strengths and weaknesses of our method for each specific case. The results from our evaluation framework are reported in two sections.

First, Quantitative results are obtained from datasets which include ground truth laser scans. Mean Error (ME) and standard deviation in both the disparity maps and 3D point clouds are calculated, between them and the ones estimated from the applied methods. ME is calculated as the average absolute difference in pixels (disparities) and millimetres of euclidean distance (depth) between the estimated and the actual disparity or depth of each pixel. It is a simple evaluation metric, yet able to provide a general performance indicator. Additionally, the percentage of reconstructed points from all the points containing ground truth information is also included. Qualitative section, includes the disparity maps and 3D pointcloud images estimated, for datasets without ground truth, accompanied with results regarding the execution time for the reconstruction of each method. Given the availability of code, along with our method, performance was evaluated over two other methods. The first is the original Quasi Dense CPU method by Stoyanov [3] and the second is ELAS [8], which also performs the calculations on the CPU. Additional methods and metrics will be included, but their results will be cited from the respective literature.

Figures in this section contain images from a single indicative frame of the dataset and will be presented in the same format, whereas each row contains results, organized in columns, regarding a single method. The first column shows the disparity map estimated by the method, while the second column includes the depth maps. The third column shows images of the Mean Error (ME) between the estimated disparity map and the ground truth disparity map, supplemented with the dataset. In Qualitative evaluation case where ground truth is not provided, this column is excluded (Fig. 4). Finally, the last column contains a snapshot of the reconstructed 3D point cloud, calculated from the estimated disparity map.

### A. *Quantitative*

As mentioned above, Quantitative evaluation is performed in datasets, which include ground truth. Therefore, the aforementioned quantitative metrics are calculated, indicating the accuracy of our 3D reconstruction result. To get a visual representation of the reconstruction error, evaluation maps are constructed, encoding the ME between the estimated and the ground truth disparity maps in color. Deep blue color indicates minimal error, while red color represents points whose ground truth is not available.

In both datasets, the best performing method, in terms of error metrics is Quasi Dense CPU. However, the proposed GPU implementation achieves similar performance. More specifically, in the Deforming Silicon Heart Dataset, as presented in Table I both disparity/depth ME values are very close, 1.34/3.70 and 1.42/4.03 for CPU version and our GPU implementation respectively. It is clear that both Quasi Dense methods, outperform ELAS, due to the outliers which are reconstructed by ELAS, compromising its performance, even if its disparity maps show consistent error. Similar results are observed in the EndoAbs dataset evaluation (Table II), where Quasi Dense GPU demonstrates error values close to its CPU counterpart. A set of outliers reconstructed from the GPU method, caused by the partial sorting of the propagation queue, can lead to large deviations in depth (l2,l3 cases), but the visual result remains intact. However, in terms of speed it is clear that Quasi Dense GPU greatly outperforms the other methods in all datasets. Execution times range from $1.85$ to $3.5$ times faster than ELAS and 13 to 18 times faster than Quasi Dense CPU.

The 3D reconstruction method suggested by Penza [5], [18] also reports results from those two datasets. However, for evaluation of the reconstruction results, accuracy is used as the evaluation metric. They define accuracy as the median of the depth error between the estimated and the ground truth point clouds. To be able to perform comparison under a common base, accuracy is also calculated for our method. In the heart dataset, quoted as heart2 in [5], is reported that accuracy ranges from $1.70$ to $3.34$, while the percentage of reconstructed points is between $44.7\%$ and $66.5\%$ of the points. It also reports execution time of $1.2$ seconds per frame. On the other hand, our method reconstructs $75.5\%$ of points, achieving accuracy of $2.30$, while calculating disparity in $44.42$ msec, which is 27 times faster. Similar results are extracted from the EndoAbs dataset. More specifically, for EndoAbs $dist_{max}$ dataset, accuracy of $2.40$, $1.80$, $1.55$ is reported in [18], for $l1, l2, l3$ lighting conditions respectively. In comparison, our method achieves $2.74$, $2.54$, $2.43$ in accuracy. However, since execution times are not reported, given the resolution of the dataset, we estimate a similar speed up from our proposed implementation.

The GPU implementation of Semi-Global Matching [7] has also been evaluated over the datasets. It achieves very fast execution times, more than $10$ times faster than our proposed method. Nevertheless, it is not included in the quantitative evaluation, since it shows error values one order of magnitude larger than the results reported here.

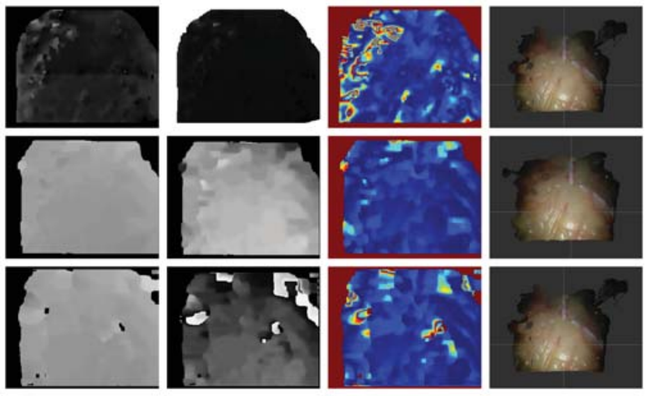
Fig. 2. Figure demonstrating 3D reconstruction results, disparity, depth, mean error and 3D point cloud (from left to right column) for Deforming Silicon Heart Dataset for compared methods, ELAS, Quasi Dense CPU and Quasi Dense GPU (from top to bottom).

## B. *Qualitative*

Datasets accompanied with ground truth are produced by phantoms and their corresponding laser scans. However, that is not the case in real MIS surgery, where in-vivo surgical scenes must be reconstructed in 3D. Thus, the inclusion of an in-vivo dataset from a real operation is important. Such dataset introduces important 3D reconstruction challenges, namely tissue deformation, reflections, blood, smoke and occlusion from surgical instruments. These challenges need to be addressed by the reconstruction algorithms, towards their adaptation in real MIS procedures. However, since ground truth data are not available, no quantitative error metric can be applied, which results in evaluation of the dataset only from its visual appearance and execution time.

Although Porcine Uterine Horn Dataset introduces few of the aforementioned 3D reconstruction challenges described above, namely reflections and deformation from respiration,

TABLE I
PERFORMANCE SUMMARY FOR DEFORMING SILICON HEART DATASET

| **Method** | **ME** | | **Std Deviation** | | **Points** | **Execution** |
|---|---|---|---|---|---|---|
| | **Disp** | **Depth** | **Disp** | **Depth** | **%** | **Time** |
| ELAS | 2.25 | 44.3 | 2.20 | 99.80 | 78.82 | 82.05 ms |
| QDS CPU | 1.34 | 3.70 | 1.21 | 4.023 | 75.59 | 776.03 ms |
| QDS GPU | 1.42 | 4.03 | 1.68 | 6.31 | 75.51 | 44.42 ms |

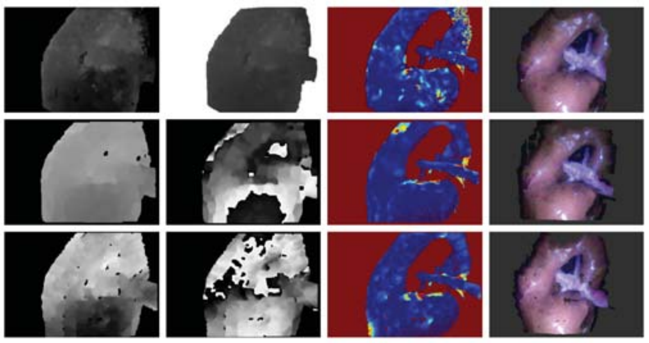
Fig. 3. Figure demonstrating 3D reconstruction results, disparity, depth, mean error and 3D point cloud (from left to right column) for EndoAbs $dist_{max}$ Dataset for compared methods, ELAS, Quasi Dense CPU and Quasi Dense GPU (from top to bottom).

TABLE II
PERFORMANCE SUMMARY FOR ENDOABS DATASET $dist_{max}$

| **Method** | **ME** | | **Std Deviation** | | **Points** | **Execution** |
|---|---|---|---|---|---|---|
| | **Disp** | **Depth** | **Disp** | **Depth** | **%** | **Time** |
| *l1 - very low lighting* | | | | | | |
| ELAS | 2.13 | 59.55 | 2.67 | 148.1 | 71.30 | 102 ms |
| QDS CPU | 1.09 | 3.02 | 1.14 | 3.02 | 52.56 | 419 ms |
| QDS GPU | 1.42 | 4.31 | 1.68 | 6.42 | 48.51 | 30.44 ms |
| *l2 - low lighting* | | | | | | |
| ELAS | 1.74 | 69.44 | 2.13 | 161.1 | 82.98 | 121 ms |
| QDS CPU | 0.99 | 2.86 | 0.99 | 4.23 | 68.74 | 866 ms |
| QDS GPU | 1.84 | 6.36 | 2.56 | 13.10 | 67.86 | 48.11 ms |
| *l3 - normal lighting* | | | | | | |
| ELAS | 1.51 | 65.34 | 1.76 | 153.9 | 85.46 | 129 ms |
| QDS CPU | 0.9 | 2.65 | 0.91 | 2.73 | 80.49 | 947 ms |
| QDS GPU | 1.38 | 4.72 | 2.14 | 10.24 | 78.86 | 52.78 ms |

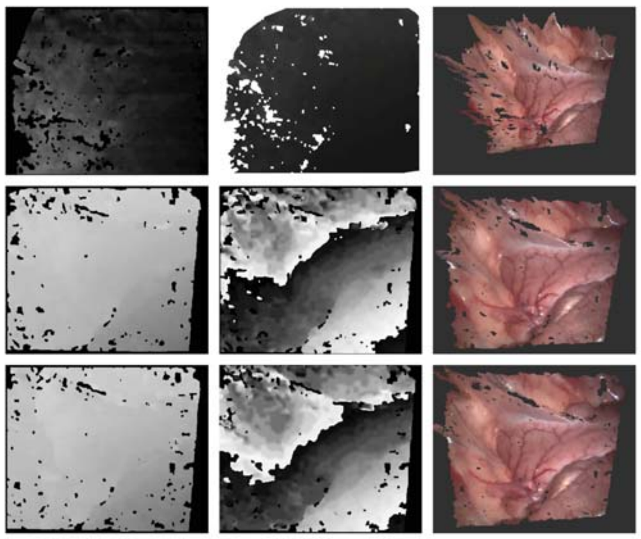

Fig. 4. Figure demonstrating 3D reconstruction results, disparity, depth and 3D point cloud (from left to right column) for Porcing Uterine Horn Dataset for compared methods, ELAS, Quasi Dense CPU and Quasi Dense GPU (from top to bottom).

TABLE III
PERFORMANCE SUMMARY FOR PORCINE UTERINE HORN DATASET.

| **Method** | **Execution Time** |
|---|---|
| ELAS | 592 ms |
| QDS CPU | 2465 ms |
| QDS GPU | 82 ms |

it has stronger texture variations. Thus, features that are more robust can be extracted and more confident matching cost can be computed. Hence, both Quasi Dense methods, which are based on such matching costs, perform best without visible differences. However, Quasi Dense CPU requires almost 2.4 seconds to reconstruct a frame, while Quasi Dense GPU can process frames at 82 msec. ELAS recovers the geometry quite accurately and fast (0.6 seconds), but produces erroneous 3D regions especially in points closer to the image borders. It is obvious that Quasi Dense GPU outperforms the other two methods, since it reconstructs the point cloud with the best quality, along with its CPU counterpart, while processing frames 30 times faster than Quasi Dense CPU and 7.2 times faster than ELAS.

## V. Conclusion

In summary, this paper argued that Quasi Dense GPU method is able to reach real-time performance based on CUDA programming model and at the same time to implement a quality 3D reconstruction with similar errors compared to state-of-the-art methods. Thus, the proposed method is suitable for application in real MIS scenarios, where not only quality of the reconstruction, but real-time performance is a main contributory factor.

## Acknowledgment

The authors would like to acknowledge nearlab of Politecnico di Milano and especially Veronica Penza for granting provision to the EndoAbs Dataset.

## References

[1] L. Maier-Hein, P. Mountney, A. Bartoli, H. Elhawary, D. Elson, A. Groch, A. Kolb, M. Rodrigues, J. Sorger, S. Speidei, D. Stoyanov, “Optical techniques for 3D surface reconstruction in computer-assisted laparoscopic surgery”, Medical Image Analysis 17 p 974996, 2013.
[2] L. Bingxiong, , S. Yu, Q. Xiaoning, G. Dmitry., G. Richard, Y. Yuncheng, ”Video Based 3D Reconstruction, Laparoscope Localization, and Deformation Recovery for Abdominal Minimally Invasive Surgery: A Survey”, Int J Med Robot. Jun;12(2):158-78, 2016.
[3] D. Stoyanov, M. Visentini Scarzanella, P. Pratt, G, Yang. ”Real-Time Stereo Reconstruction in Robotically Assisted Minimally Invasive Surgery”, Medical Image Computing and Computer-Assisted Intervention - MICCAI , 13th International Conference, Beijing, China, September 20-24, 2010, Proceedings, Part II (pp.275-82), 2010.
[4] S. Bernhardt, J. Abi-Nahid, R. Abugharbieh. ”Robust, Robust dense endoscopic stereo reconstruction for minimally invasive surgery”, In: International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI): Workshop on Medical Computer Vision (MCV), pp. 198207, 2012.
[5] V. Penza, J. Ortiz, L. S. Mattos, A. Forgione, E. D. Momi, ”Dense soft tissue 3D reconstruction refined with super-pixel segmentation for robotic abdominal surgery”, International Journal of Computer Assisted Radiology and Surgery, Volume 11, Issue 2, pp 197206, 2016
[6] S. Rhl, S. Bodenstedt, S. Suwelack, R. Dillmann, S. Speidel, H. Kenngott, B.P. Mller- Stich, ”Dense GPU-enhanced surface reconstruction from stereo endoscopic images for intraoperative registration”, Med. Phys. 39, 16321645, 2012.
[7] D. Hernandez-Juarez, A. Chacon, A. Espinosa, D. Vazquez, J. C. Moure, A. M. Lopez, ”Embedded real-time stereo estimation via Semi-Global Matching on the GPU”, ICCS2016. The International Conference on Computational Science, Volume 80, 2016, Pages 143153, 2016.
[8] A. Geiger, M. Roser, Raquel Urtasun, ”Efficient Large-Scale Stereo Matching”, Computer Vision – ACCV 2010, pp 25-38, 2010.
[9] J. Zbontar and Y. LeCun, ”Computing the Stereo Matching Cost with a Convolutional Neural Network.”, arXiv preprint arXiv:1409.4326, 2014.
[10] M. Ye, E. Johns, A. Handa, L. Zhang, P. Pratt, G.Z. Yang, ”Self-Supervised Siamese Learning on Stereo Image Pairs for Depth Estimation in Robotic Surgery”, Unpublished, 2017.
[11] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ”ORB: An efficient alternative to SIFT or SURF”, ICCV 2011: 2564-2571, 2011.
[12] H. Bay, T. Tuytelaars, L.V. Gool, ”SURF: Speeded up robust features”, In Proc. European Comp Vis. Conf, pages 404417, 2006.
[13] F. A. Pablo, L. M. Bergasa, A. F. Davison, ”Gauge-SURF Descriptors”, Image and Vision Computing Volume 31, Issue 1, Pages 103-116, 2012.
[14] J. Shi, L. Tomasi. ”Good Features to Track”, Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1994.
[15] T. Kanade and M. Okutomi, ”A stereo matching algorithm with an adaptive window: Theory and experiment”, TPAMI, vol. 16, no. 9, pp. 920932, 1994.
[16] K. E. Batcher, ”Sorting networks and their applications”, In AFIPS 68 (Spring): Proceedings of the April 30May 2, pages 307314, New York, NY, USA, 1968.
[17] A.S. Ciullo, V. Penza, L. Mattos, E. De Momi, ”Development of a surgical stereo endoscopic image dataset for validating 3D stereo reconstruction algorithms”, 6th Joint Workshop on New Technologies for Computer/Robot Assisted Surgery, 2016.
[18] V. Penza,A. S. Ciullo, . Moccia, L.S. Mattos, E. De Momi, ”EndoAbS Dataset: Endoscopic Abdominal Stereo Image Dataset for Benchmarking 3D Stereo Reconstruction Algorithms”, The International Journal of Medical Robotics and Computer Assisted Surgery, in press.