Real-Time Reconstruction of Underwater Environments: from 2D to 3D

Matija Rossi*§, David Scaradozzi†‡§, Pierre Drap‡§, Pietro Recanatini†‡,
Gerard Dooly*, Edin Omerdić*, and Daniel Toal*
*Mobile and Marine Robotics Research Centre, University of Limerick, Limerick, Ireland
Email: matija.rossi@ul.ie
†Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, 60131 Ancona, Italy
Email: d.scaradozzi@univpm.it
‡Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296, 13397 Marseille, France
Email: pierre.drap@lsis.org
§Equally contributed to this work.

***Abstract*—This paper presents a real-time 2D image mosaicking tool developed to provide instantaneous feedback on image quality and area coverage during underwater site inspection or documentation surveys. The algorithm implements a feature extraction and matching approach to stitching video frames into a single image. While its main advantage is providing good results for fast documentation in real-time even on low-end computer hardware, it has also some limitations in terms of sensitivity to the camera motion and lower performance in textureless or uniform environments. The final part of the paper addresses those limitations by providing an insight of the concept behind the ongoing development of a real-time direct 3D reconstruction tool for underwater applications.**

I. Introduction

Computer vision is rapidly finding more and more applications across a wide spectrum of underwater operations. It is increasingly being used as the primary tool for inspection and documentation of underwater sites, in disciplines ranging from archaeology [1] and biology [2] to offshore engineering [3]. Cameras are carried and operated by both divers and robots, as the popularity of remotely operated vehicles (ROV) and autonomous underwater vehicles (AUV) increases ([4], [5]). The introduction of robots brings with it many new applications for computer vision. A common task is robot navigation, which underwater is challenging due to the lack of any radio communication, including global navigation satellite systems (GNSS) on which terrestrial and aerial robots heavily rely. For this purpose camera and acoustic systems can be used to implement simultaneous localisation and mapping (SLAM) algorithms to complement inertial navigation systems (INS), which, to various extents, suffer from integration drift. Even more demanding than inspection is robotic intervention, where ROVs are usually used with human operators at the surface, although recently significant effort is going into the automation of those operations ([6], [7]). For both cases computer vision is playing a fundamental role. Providing an augmented feedback could increase the ROV pilot's efficiency multiple times compared to a standard 2D camera stream. If any aspect of the intervention is autonomous, then knowing the exact structure of the target and the position of the robot relative to it becomes crucial.

All the described applications would significantly benefit from, and many even require, models of the underwater environments or targets generated in or near real-time. Data for inspection and documentation purposes is usually post-processed, and is acquired without feedback on its quality. This leads to situations where defects in data, such as areas not being covered or unsuitable image quality, are discovered after the survey. In the case of robotic intervention operations, having real-time feedback is even more valuable, as it can affect the duration of the operations themselves. This holds for scenarios where people are directly involved in operations, whether as divers o through ROVs. In autonomous robot operations, real-time models become a requirement. Due to offshore operations being particularly expensive, time consuming, and limited by other factors such as weather, making them more efficient is of great value [8].

This paper presents a real-time 2D image mosaicking tool, the development of which started as part of the ROV-3D project [9]. It is a useful tool to provide instantaneous feedback on image quality and area coverage during the survey, and gives good results for fast documentation of sites. Apart from the benefits, the paper also discusses the limitations of a 2D feature-based approach, and provides an insight in the direct 3D approach currently being developed as a potential solution to the described limitations.

Section II provides a brief background of previous work relevant to the development of the described tools. Section III describes in detail the 2D mosaicking algorithm and implementation, of which Section IV presents the results. Section V then describes the main ideas behind the direct 3D approach, concluding with a summary in Section VI.

II. Background

A. 2D mosaicking

Early examples of algorithms for stitching multiple images together to form a single larger one, i.e. mosaicking, can be found in [10], [11], and [12]. An important advancement has been made with the introduction of the bundle adjustment algorithm [13] for the minimisation of the reprojection error between two images [14]. A successful implementation has been developed in 2003 [15] and further improved in 2007 [16]. In [17] the authors have made an evaluation of various

978-0-933957-43-5 ©2015 MTS

solutions to prevent the visible seam between each image in mosaics. The work presented in this paper borrows the concepts from the mentioned publications and combines them to obtain a mosaicking implementation capable of real-time video processing.

B. Real-time 3D reconstruction

Dense real-time 3D reconstruction is becoming feasible only very recently, with the advent of widely available massively parallel commodity general purpose computing on graphics processing units (GPGPU). Notable steps towards real-time 3D reconstruction were MonoSLAM [18] for single camera visual SLAM using sparse features, and the real-time visual odometry system from [19]. A different approach from the filter based SLAM systems that preceded it, was presented in [20], where the authors separated the camera tracking given a known map, and updating the map. Following the work of [21], which applied convex optimisation techniques on commodity GPUs to achieve real-time image denoising, [22] presented a live dense 3D reconstruction pipeline using the feature-based [20] for tracking. The first monocular system to use both dense tracking and mapping and capable of real-time processing was presented by [23].

III. Real-time 2D mosaicking algorithm

Image mosaicking in the context of this work can be defined as the process of stitching together a sequence of photographs or video frames, with the goal of obtaining a single large image. The developed real-time 2D mosaicking algorithm can be split in three main steps: (i) *image registration* produces a frame to mosaic transformation given the current live frame from the camera and the existing mosaic; (ii) *image warping* transforms the live frame to match the mosaic using the previously computed transform; and (iii) *image compositing* finally blends together the warped frame with the mosaic. The main difference between a real-time algorithm and the more common post-processing approaches is that each image has to be processed immediately when acquired. The mosaic is therefore built incrementally, without the possibility to find the optimal solution which is possible when the full set of images is available a priori. It will be shown that the system mostly consists of existing and widely used algorithms, combined in such a way to achieve real-time performance even on low-end laptop PC hardware. It is important to specify that real-time performance in this context is considered to be the ability to process enough frames from the live video to reliably build a mosaic, i.e. overlapping frames.

A. Image registration

The mosaicking procedure begins with image registration, which is the most important part in terms of robustness and accuracy of the whole system. Given the live frame I_{l_t} at step t and the mosaic $I_{m_{t-1}}$, which contains frames from steps 1 to $t-1$, the goal is to find the best possible frame to mosaic transformation T_{lm_t}. In the following text the step indices will be omitted for cleaner notation, as the focus is on the analysis of a single algorithm iteration. Theoretically T_{lm} could be any generic projective transformation with 8 degrees of freedom. However, it has been found that constraining it only to affine transformations, therefore removing 2 DOF, the mosaic becomes significantly more robust to accumulating error. Therefore, the transformation matrix T_{lm} has the following form:

$$T_{lm} = \begin{bmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \end{bmatrix}. \quad (1)$$

Each point $u_l \in \Omega_l \subset \mathbb{N}^2$ of the live frame is transformed into mosaic coordinates $u_m \in \Omega_m \subset \mathbb{N}^2$ as:

$$u_m = T_{lm} u_l. \quad (2)$$

In order to compute T_{lm} it is necessary to find three pairs of corresponding points between I_l and I_m. For this to be possible there has to be a significant overlap between the two images.

The Speeded Up Robust Features (SURF [24]) feature detector and descriptor is used to find and describe salient points in both images. It has been chosen because it is several times faster than the Scale-Invariant Feature Transform (SIFT [25]) detector, by which it was inspired, which makes it more suitable for real-time applications, while maintaining a similar level of robustness. Other feature detectors could also be used, depending on the application, which might additionally speed up the execution of the software. Applying the SURF detector on I_l and I_m, two sets of keypoints are obtained: $f_l^u = \{f_{l_i}^u \in \Omega_l\} \forall i \in \{1 \dots I\}$ and $f_m^u = \{f_{m_j}^u \in \Omega_m\} \forall j \in \{1 \dots J\}$, each containing the image coordinates of I and J detected keypoints respectively. Computing the SURF descriptor for all points in f_l^u and f_m^u yields the two keypoint descriptor sets $f_l^d = \{f_{l_i}^d \in \mathbb{R}^{64}\}$ and $f_m^d = \{f_{m_j}^d \in \mathbb{R}^{64}\}$.

To find matching pairs of features, the fast approximate nearest neighbour algorithm from [26] is applied to the high dimensional SURF descriptors from f_l^d and f_m^d, producing a set of K pairs of keypoint descriptors $p_{lm}^d = \{p_{lm_k}^d\} \forall k \in \{1 \dots K\}$, where $p_{lm_k}^d = \{f_{l_i}^d, f_{m_j}^d\}$. A set of the corresponding pairs of keypoint image coordinates is then generated based on the descriptor pairs in p_{lm}^d, resulting in $p_{lm}^u = \{p_{lm_k}^u\}$.

The affine transformation T_{lm} can then be computed from p_{lm}^u under the assumption that $K \geq 3$. This represents one limitation of feature based methods, which is the case of featureless scenes. However, in the majority of scenarios when the camera is closely observing a scene there will be enough diversity, and the number of detected features will exceed the minimum limit by a large margin. When $K > 3$ it becomes necessary to choose exactly 3 keypoint pairs from p_{lm}^u. Random selection poses the threat of picking outliers, i.e. keypoints matched wrongly. To tackle this issue the RANdom SAmple Consensus (RANSAC [27]) algorithm is employed, which iteratively chooses three pairs of keypoints and computes a temporary $T_{lm_{\text{temp}}}$ solving (2) for the three pairs. It then applies it to all paired keypoints from the live frame, which results in a set of points in mosaic coordinates

$$f_{m_T}^u = \{T_{lm_{\text{temp}}} f_{l_k}^u \in \Omega_m\} \quad \forall f_{l_k}^u \in p_{lm}^u. \quad (3)$$

Ideally $f_{m_T}^u = f_m^u$, but this will never hold due to outliers and because even in correct matches there will be small errors. Therefore, the algorithm counts the number of correct matches N by comparing $f_{m_T}^u$ and f_m^u using the Euclidean distance between each pair of points, and selecting the ones that are correctly reprojected within an allowed error radius e_r. N defines how good the solution $T_{lm_{\text{temp}}}$ is. The RANSAC stops when

N reaches a certain predefined minimum threshold, or when the number of iterations reach a predefined maximum value, in which case the solution will be the one with the highest N so far. This method does not guarantee the optimal solution, but, given enough iterations, it will find an appropriate value of T_{lm}. It is therefore a matter of compromise between the maximum number of RANSAC iterations and the execution time of the algorithm.

B. Image warping

Once T_{lm} is computed, the next step is to apply the transformation to the entire live image I_l, to warp it so it matches the mosaic $I_{m_{t-1}}$. The image warping process first creates a new empty image I_{m_t} that will become the new mosaic, and which is first going to contain the warped I_l, on top of which the current mosaic $I_{m_{t-1}}$ will be overlaid in the final step. Here the step indices are reintroduced for mosaics to distinguish the new from the old one. The points from I_l are mapped as:

$$I_{m_t}(T_{lm}u_l) = I_l(u_l) \quad \forall u_l \in \Omega_l. \tag{4}$$

In order to restore the quality lost during the warping, resampling of the image is performed using bicubic interpolation, which considers a 4×4 pixel neighbourhood. Although using bilinear interpolation would noticeably improve the algorithm's execution time, its quality has been found to be inappropriate.

C. Image compositing

Once I_{m_t} contains the transformed and corrected I_l, the final step is to copy $I_{m_{t-1}}$ into it as well. The copying is straightforward since there is no transformation to be done. However, on the area of $I_{m_{t-1}}$ which overlaps with the transformed I_l a blending mask is applied to obtain a smooth transition between the two images. Without blending, a sharp line would be visible due to the difference between the brightness and colour of the two images. The mask consists in applying a weighted sum of pixel intensities where the two images overlap, i.e. a linear gradient between the two images:

$$I_{m_t}(u) = wI_{m_{t-1}}(u) + (1-w)I_{m_t}(u) \quad \forall u \in \Omega_{m_t}, \tag{5}$$

where $w \in [0,1]$ is a function of the Euclidean distance between the two image centres. This step concludes the mosaicking procedure, resulting with the live frame I_l being stitched with the mosaic $I_{m_{t-1}}$, producing a new mosaic I_{m_t}. At this point the algorithm restarts with a new frame to add.

D. Implementation – MosaiQt

The described algorithm has gone through multiple stages of implementation. Its development started in MATLAB, which was convenient for prototyping but quickly proved to be too slow for the final implementation. LabVIEW was briefly considered as an alternative because of the desire to integrate the mosaicking algorithm with existing LabVIEW software. Although better results have been achieved in terms of speed, OpenCV was used for to higher flexibility in modifying and experimenting with its algorithms. For this to be possible most of the code was actually implemented in C++ and compiled to shared libraries. LabVIEW therefore did not seem to be the best tool to continue the computer vision related research in, so the final decision was to write everything in C++ with OpenCV, and from there to optimise the algorithm for speed. The Qt framework has been chosen for the final GUI implementation in order to be able to support different operating systems, and because of its Free Software licence. Therefore the name of the final product – MosaiQt.

One of the main implementation features is not trying to process every frame of the live video, but only as many frames as possible. In other words, the software takes the newest frame from the video only after its done with processing the previous one, skipping the frames generated during the processing time. This makes real-time mosaicking achievable with the same software even on low-end hardware, with the only side effect being that less frames are processed in a unit of time, i.e. the processing frequency is lower. This is not an issue, firstly because the algorithm itself is best suited for slightly longer baselines: it is better to avoid consecutive frames looking at the same scene as each will introduce additional error. Ideally, frames should overlap but not entirely. Secondly, even if the processing frequency is so low that frames do not overlap enough, the simple solution is to decrease the speed of the camera. This is an advantage of having real-time feedback – it is possible to accurately adapt the data acquisition to best suit the processing.

This system of different acquisition (video) and processing frequencies is implemented through multithreading. The entire program uses three threads:

- GUI thread
- Live video thread
- Mosaicking thread

The live video thread acquires all frames from a USB frame grabber or camera and displays them in the GUI. The mosaicking thread requests a frame from the live video as soon as it stops processing the previously received one. It has the possibility to display in the GUI the mosaic created so far and to store it at every step on the disk, in order to always have the last good one in case a bad frame corrupts it in the following iteration. There is also the option within the software to store each frame for post processing, which is often the scenario. Live processing is a very convenient mean to make sure the data quality is satisfactory, which will then be post processed to obtain the best possible result. MosaiQt offers also the choice to build a mosaic from a previously acquired set of images, which can also be useful to quickly validate the images acquired from a source not connected to a PC. Additionally, there is the possibility to correct the image distortion before processing if the intrinsic camera parameters are known, which is very significant when using a wide angle lens.

Figure 1 shows the main window of MosaiQt. It offers the possibility to modify certain parameters regarding the SURF detector, the feature matching quality, and the maximum number of RANSAC iterations. The software has a multiple windows interface, meaning that things like the live video and the mosaic that is being built are displayed in separate and resizeable windows. This makes it suitable for various display sizes and multiple monitor setups.

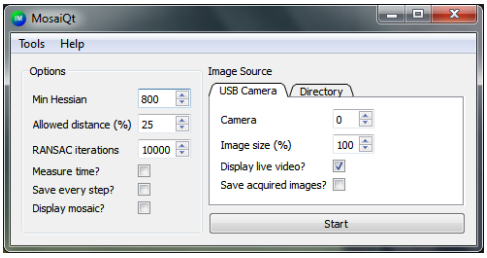

Fig. 1. MosaiQt main window

Fig. 2. Mosaic of data from [28]

IV. Results

This section presents the results achieved with MosaiQt, the mosaicking algorithm implementation discussed in the previous section. Figure 2 shows a mosaic produced from a $\sim$ 85 seconds long FHD (1920×1080) resolution transect survey made by the submarine Remorà 2000 equipped with ISME and CNRS technologies in an automatic way under the supervision of the archaeologist Luc Long at the Port-Miou C wreck in Marseille, France [28]. It has been created by processing 2 frames per second. Figure 3 shows a detail of the first 36 seconds of Figure 2, while Figure 4 presents a 10 seconds detail.

Although it would be necessary to improve the mask used for blending images to remove some sharp transitions visible in the mosaic, the overall quality is good, and comparable to fast post-processing solutions. It can be seen from the full length mosaic in Figure 2 that the accumulated error level is acceptable even after a longer period of time. It is hard to quantify errors in this scenario, therefore only a qualitative analysis can be offered at this time.

On a higher specifications computer it is possible to process $\sim$ 2 frames per second in FHD resolution, which has proven to be more than enough for typical speeds of divers or unmanned vehicles, therefore making the mosaicking real-time. A low-end laptop is able to process about 1 frame per second of the same video, which is also good for slower camera motion. The best way to achieve higher processing frequencies is to downsize the images before using them. At VGA (640×480) resolution it is possible to process between 4 and over 10 frames per second, depending on the computer. Since each frame introduces some small error, using too many is not desirable. This can happen if the camera is not moving fast enough to introduce significant difference between frames, in which case better results are achieved with higher resolutions at lower frame rates. As already mentioned in Section III-D, it is important to emphasize the meaning of the processing frequency. For example, if MosaiQt processes 2 fps of a 30 fps video stream, it means that only every fifteenth frame will be processed, while the fourteen frames in between will not be used. This allows the algorithm to run in real-time along the video.

Despite being able to obtain very good results, this approach has also shown its limitations. The described pipeline is most successful when the camera is recording perpendicular to the site, and when the site is rich in visual features. Figure 5 shows an example of a low quality mosaic from the Netmar project demonstration in Porto, Portugal, in May 2015 [29]. The inspected object is the hull of a sunken WW2 submarine, which is very uniform in visual features. Together with bad visibility, which makes everything appear to be of the same colour, this creates a very difficult situation for mosaicking. It can be seen that the process starts successfully, due to holes in the hull, but very quickly the mosaic starts drifting, and eventually, when the holes end, it fails to stitch any more images. In this conditions it was not possible to obtain a mosaic of the entire hull. The fact that both assumptions mentioned above limit the potential applications of MosaiQt initiated the ongoing development of a 3D featureless reconstruction pipeline, while still aiming for real-time capabilities.

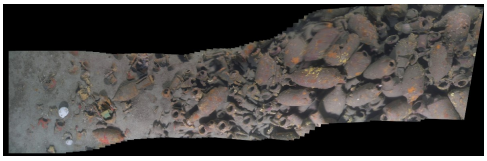

Fig. 3. Initial part of the mosaic in Figure 2

Fig. 4. Mosaic detail

V. Towards real-time 3D reconstruction

In order to expand the applications of real-time video processing for underwater robotics, state-of-the-art monocular

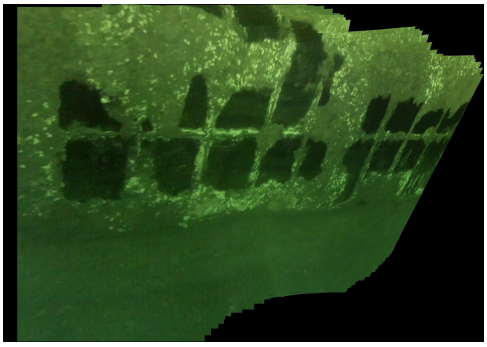

Fig. 5. Mosaic of a uniform environment in bad visibility, from [29]

3D reconstruction methods are currently being explored, with the goal of reaching all the scenarios mentioned in Section I. Most of today's approaches to underwater 3D reconstruction, also known as structure from motion, uses feature based camera motion estimation and sparse point clouds for structure representation. Based on experience, as discussed in Section IV, feature based approaches are limited in underwater environments. In [30] the author shows how motion blur, noise, and defocus drastically affect SIFT and FAST [31] detectors. Furthermore, [23] compares the robustness of the direct DTAM to the feature based PTAM [20] tracker under the above mentioned conditions. It can be seen how a direct method which exploits every image pixel is significantly more robust. Apart from the tracking problem, the disadvantage of using a sparse point cloud for 3D representation is when robots have to interact with the reconstructed object, e.g. for navigation purposes, and even more so for manipulation.

The approach developed in [30] achieve both camera tracking and structure estimation in real-time without using feature detection and matching, defining instead a per-pixel optimisation framework. For motion estimation between frames, the authors optimise a cost function over all pixels in the reference image I_a:

$$\epsilon = \sum_{u \in \Omega} \psi(I_b(w(u, D(u), \xi_{ba})) - I_a(u)). \qquad (6)$$

The transformation between pixel $u \in \Omega \subset \mathbb{R}^2$ from image a to image b, assuming a known intrinsic camera matrix K and depth $D(u)$, is:

$$w(u, D(u), \xi_{ba}) = \pi \left(K \xi_{ba} K^{-1} \begin{bmatrix} u^T & 1 \end{bmatrix}^T D(u) \right), \qquad (7)$$

where ξ_{ba} are the transformation parameters. This is made under the assumption of brightness constancy across corresponding pixels, i.e. that $I_b(w(u, D(u), \xi_{ba})) \approx I_a(u)$ given the correct ξ_{ba}. This is generally not true, but it can hold across a short time period, and can be improved performing photometric normalisation of each video frame. Although (6) is not convex, the authors show it is possible to achieve convergence to the minimum if the initial parameters are within a certain margin of error.

To compute a depth map, the basic approach would be using the same cost function from (6), but fixing the transformation parameters and estimating depth. For each pixel in the reference frame I_a, this would result in:

$$D(u) = \min_{d \in \mathbf{R}_+} \sum_{j=1}^{M} \psi(I_j(w(u, d, \xi_{ja})) - I_a(u)), \qquad (8)$$

where $M \geq 1$ is the number of views observing pixel u. In practice, along with the data term, a regularisation term is introduced in the cost function for depth map denoising. Every depth map is then integrated into the global surface. The surface is represented using truncated signed distance functions, using a weighted mean of overlapping depth maps for its update.

The overall idea of the ongoing research is therefore to port the concepts developed in [30] to underwater applications. The challenges involve mostly dealing with the significantly worse visibility compared to air, which is a problem especially for camera tracking. Apart from image preprocessing to attenuate degradation effects caused by water, additional approaches are being studied to tackle this problem. Except on low-end ROVs, inertial navigation systems are usually present on robots, therefore one of the methods is to integrate the motion estimation from the INS with the camera tracker, making the 3D system more robust to losing track. Additionally, fusion of optically estimated depth maps with forward looking sonar data will be explored to help the structure estimation in cases of bad visibility. Finally, it still remains challenging to achieve real-time performance, as all the algorithms have to run on robot systems, which are not only limited by processing power, but also often by the available communication speeds and other infrastructural problems which are absent in lab applications.

VI. Conclusion

This paper presented an implementation of a system for building image mosaics in real-time from video streams, and the results obtained with it. Furthermore, it outlined the ongoing research in building a real-time dense 3D reconstruction tool capable of underwater operation, the results of which will be published in the near future. It has been proven by the 2D mosaicking software that real-time models of underwater environments are immensely useful in many applications, therefore it is an exciting task to bring the latest achievements of computer vision to such a challenging field.

Acknowledgment

This publication has emanated from research supported by the Science Foundation Ireland under the MaREI Centre research programme [Grant No. SFI/12/RC/2302]. The MaREI project is also supported by the following industrial partners: SonarSim, Teledyne Reson, Teledyne BlueView, the Shannon Foynes Port Company, and the Commissioners of Irish Lights.

This publication has also emanated from research supported by the European Community under projects ERASMUS+ and VENUS (Contract IST-034924) of the "Information Society Technologies (IST) programme of the 6th FP for RTD".

References

[1] P. Chapman, K. Bale, and P. Drap, "We all live in a virtual submarine," *Computer Graphics and Applications, IEEE*, vol. 30, no. 1, pp. 85–89, Jan 2010.

[2] S. Cocito, S. Sgorbini, A. Peirano, and M. Valle, "3-d reconstruction of biological objects using underwater video technique and image processing," *Journal of Experimental Marine Biology and Ecology*, vol. 297, no. 1, pp. 57 – 70, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022098103003691

[3] S. Negahdaripour and P. Firoozfam, "An rov stereovision system for ship-hull inspection," *Oceanic Engineering, IEEE Journal of*, vol. 31, no. 3, pp. 551–564, July 2006.

[4] G. Antonelli, *Underwater Robots*, ser. Springer Tracts in Advanced Robotics. Springer, 2014, vol. 96. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-02877-4

[5] J. Elvander and G. Hawkes, "Rovs and auvs in support of marine renewable technologies," in *Oceans, 2012*, Oct 2012, pp. 1–6.

[6] P. Cieslak, P. Ridao, and M. Giergiel, "Autonomous underwater panel operation by girona500 uvms: A practical approach to autonomous underwater manipulation," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 529–536.

[7] D. Ribas, P. Ridao, A. Turetta, C. Melchiorri, G. Palli, J. Fernandez, and P. Sanz, "I-auv mechatronics integration for the trident fp7 project," *Mechatronics, IEEE/ASME Transactions on*, vol. PP, no. 99, pp. 1–10, 2015.

[8] E. Omerdic and D. Toal, "Oceanrings: System concept & applications," in *Control Automation (MED), 2012 20th Mediterranean Conference on*, July 2012, pp. 1391–1396.

[9] P. Drap, J. Seinturier, B. Hijazi, D. Merad, J.-M. Boã, B. Chemisky, and L. Long, "The rov 3d project: Deep-sea underwater survey using photogrammetry. applications for underwater archaeology," *Journal on Computing and Cultural Heritage*, 28 may 2015.

[10] S. Mann and R. Picard, "Virtual bellows: constructing high quality stills from video," in *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, vol. 1, Nov 1994, pp. 363–367 vol.1.

[11] R. Szeliski, "Video mosaics for virtual environments," *Computer Graphics and Applications, IEEE*, vol. 16, no. 2, pp. 22–30, Mar 1996.

[12] H.-Y. Shum and R. Szeliski, "Construction of panoramic mosaics with global and local alignment," *International Journal of Computer Vision*, vol. 36, no. 2, pp. 101–130, February 2000, erratum published July 2002, 48(2):151-152. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=75614

[13] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment a modern synthesis," in *Vision Algorithms: Theory and Practice*, ser. Lecture Notes in Computer Science, B. Triggs, A. Zisserman, and R. Szeliski, Eds. Springer Berlin Heidelberg, 2000, vol. 1883, pp. 298–372.

[14] P. F. Mclauchlan, A. Jaenicke, and G. G. Xh, "Image mosaicing using sequential bundle adjustment," in *In Proc. BMVC*, 2000, pp. 751–759.

[15] M. Brown and D. Lowe, "Recognising panoramas," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, Oct 2003, pp. 1218–1225 vol.2.

[16] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *Int. J. Comput. Vision*, vol. 74, no. 1, pp. 59–73, Aug. 2007. [Online]. Available: http://dx.doi.org/10.1007/s11263-006-0002-3

[17] W. Xu and J. Mulligan, "Performance evaluation of color correction approaches for automatic multi-view image and video stitching," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, June 2010, pp. 263–270.

[18] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, Oct 2003, pp. 1403–1410 vol.2.

[19] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, June 2004, pp. I–652–I–659 Vol.1.

[20] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, Nov 2007, pp. 225–234.

[21] T. Pock, "Fast total variation for computer vision," Ph.D. dissertation, Graz University of Technology, January 2008.

[22] G. Graber, T. Pock, and H. Bischof, "Online 3d reconstruction using convex optimization," in *1st Workshop on Live Dense Reconstruction From Moving Cameras, ICCV 2011*, 2011.

[23] R. A. Newcombe, S. Lovegrove, and A. Davison, "DTAM: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, Nov 2011, pp. 2320–2327.

[24] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *Proceedings of the ninth European Conference on Computer Vision*, May 2006.

[25] D. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, 1999, pp. 1150–1157 vol.2.

[26] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*. INSTICC Press, 2009, pp. 331–340.

[27] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[28] P. Drap, "VENUS: Virtual ExploratioN of Underwater Sites: Port-Miou C wreck, Marseille," *Archaeology Data Service*, 2009. [Online]. Available: http://archaeologydataservice.ac.uk/archives/view/venus_eu_2009/

[29] "Netmar – Porto Incident Response Demonstration," http://project-netmar.eu/meetings-and-events/porto-incident, Accessed: 2015-08-14.

[30] R. A. Newcombe, "Dense Visual SLAM," Ph.D. dissertation, Imperial College London, 2014.

[31] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision ECCV 2006*, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and A. Pinz, Eds. Springer Berlin Heidelberg, 2006, vol. 3951, pp. 430–443.