Underwater Photogrammetry Reconstruction: GPU Texture Generation from Videos Captured via AUV

Kolton Yager[1], Christopher Clark[2], Timmy Gambin[3], and Zoë J. Wood[1]

[1] Computer Science Department, California Polytechnic State University San Luis Obispo, CA
kjyager@calpoly.edu, zwood@calpoly.edu
[2] Engineering Department, Harvey Mudd College, Claremont, CA clark@g.hmc.edu
[3] Department of Classics and Archeology, University of Malta, Malta
timmy.gambin@um.edu.mt

Abstract. Photogrammetry is a useful tool for creating computer models of archaeological sites for monitoring and for general public outreach. Modeling archaeological sites found in the marine environment is particularly challenging due to danger to divers, the cost of underwater photography equipment and lighting challenges. The automatic acquisition of video footage of underwater marine archaeology sites using an AUV can be an advantageous alternative, yet also incurs its own obstacles. In this paper we present our system and enhancements for applying a standard photogrammetry reconstruction pipeline to underwater sites using video footage captured from an AUV. Our primary contribution is a GPU driven algorithm for texture construction to reduce blur in the final model. We demonstrate the results of our system on a well known wreck site in Malta.

Keywords: photogrammetry; texture generation; autonomous underwater vehicles; marine archaeology; color correction;

1 Introduction

Marine archaeologists work to discover and map historical archaeological sites in the challenging environments of our oceans and seas. This setting creates unique challenges that technology has greatly aided in recent years. For example, the use of photogrammetry to create 3D reconstructions of underwater sites, has drastically increased the ability of marine archaeologists to map and share data about their discoveries [16,22].

Photogrammetry is a useful tool for creating computer models of archaeological sites of interest from discrete frames showing the target site from various views. There is a wide body of work addressing the use of photogrammetry pipelines for archaeological site mapping [13,16,22]. Our work focuses on the marine setting and specifically the use of an AUV to capture video of archaeological sites of interest such as historical ship or plane wrecks. As diving for a human camera operator can be both dangerous and expensive, the use of an AUV provides advantages[21,12,17], such as operating in deep waters and deployment from a distance.

The fieldwork for this research was conducted off the coastal waters of Malta, using the OceanServer Iver3 AUV. The AUV was equipped with a GoPro HERO4 camera to capture video of the sites of interest. The work presented here focuses on resolving some of the limitations of camera paths and video quality due to the underwater setting.

Unique Challenges The underwater setting introduces challenges both to the collection of imagery for photogrammetry, and to the typical photogrammetric pipeline. The need for camera equipment that is both waterproof and compact enough to be mounted onto an AUV limits hardware choices and the maximum acuity of captured imagery. Likewise, continuous capture, the recommended style for photogrametry [26], is made difficult by AUV maneuverability constraints. Instead video capture is limited to predominantly linear paths which cover only portions of the subject. This division of footage into discontinuous segments can lead to significant inconsistencies later in the pipeline when portions are joined into a full reconstruction.

In addition, as light travels through water, certain wavelengths are blocked, reducing colors whilst particulate in the water can reduce visibility. Furthermore the falloff of visibility due to the water volume necessitates shorter distance between camera and the target, and reduces the amount of information collected along most paths about the subject. Overly blue images and blur proves challenging for photogrammetry pipelines that typically rely on pixel correlation primarily using color and in addition can result in textures which are visually unappealing and unduly homogeneous.

To address these challenges we present our pipeline and view dependent texturing algorithm to produce high quality models from applying a photogrammetry pipeline to video data acquired in an underwater setting using an AUV. Specifically, our contributions include a novel GPU based algorithm for improved texture creation in the spirit of view dependent texturing to minimize blur caused by excessive smoothing for the final model's texture and a general system for enhancing photogrammetry reconstruction using data captured using an AUV and GoPro camera.

2 RELATED WORK

There is a rich body of work addressing the field of photogrammetry [13,16,22], and AUV trajectory planning [20,15,10,19,8] and a growing body of work addressing trajectory planning specifically for photogrammetry[23,25,26].

Additionally, there is a wide body of related work on texturing. Debevec's work[11], was foundational for texture mapping through projection in combination with a view-dependent blending function primarily based on view angle. The more recent work of Waechter et al [24] presents a sophisticated approach with several stages, including, an analysis of the gradient of each photo to disqualify regions with less captured detail. To address issues with inaccurate camera alignment or a moving subject, a mean shift algorithm is used to filter image sources whose colors vary too greatly from the rest of the set. Similarly, Callieri et al. [7] presents a a system which uses a variety of metrics to judge the quality of source photos, ultimately creating a single final mask on each photo which determines the photo's weight at the time of final projection.

The problem of color correcting underwater photography has been tackled widely but often with similar core assumptions. Commercially available camera lens filters designed to mitigate discoloration in underwater footage are not appropriate with our set up and site depths. Software solutions generally include an operation which attempts to mitigate discoloration caused by the attenuation of light underwater. The result of this operation typically produces result images which resemble the original, but which have

ambient illumination close to the gray scale instead of an original blue or cyan/green tint. The methods of both Bazeille et. al. [6] and Iqbal et. al. [14] re-balance the images colors towards gray using only the characteristics of the image itself. Conversely, Petit et al [18] takes a more involved approach which first establishes a Beer-Lambert model of the scenes attenuation with a coefficient set from experimental data.

Related work of particular interest are several papers focused on improving texturing for 3D reconstructions, [5,4,9]. In particular, the multi-band approach could provide future directions for image enhancements in our system, however, this work is the first that we know of in which the scale of input images from an underwater setting and the use of the GPU for texture production is explored to this level.

3 Algorithms

We present our algorithm for using the graphics pipeline for creating an improved final texture for models reconstructed using a photogrammetry pipeline in an underwater setting using data captured from an AUV and GoPro camera. To address the unique demands of underwater photogrammetry, over time we have developed a system for managing some of the specific challenges of the marine setting. Figure 1 illustrates the general system used to reconstruct models.

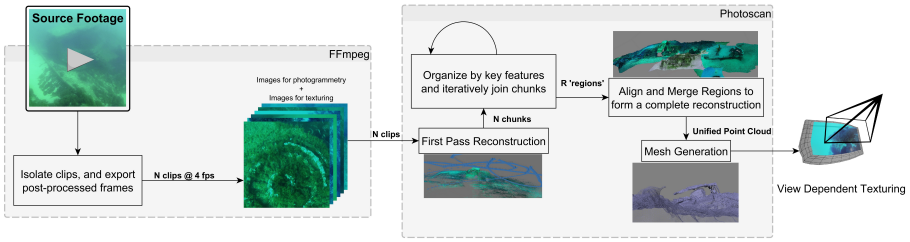

Fig. 1. Overview of our system for improved textured reconstructions from AUV video data

Photogrammetry Reconstruction pipeline from AUV video data: In general using an AUV for video capture requires multiple passes over the site. To minimize geometric distortion in the reconstruction we decompose and group the video data into select regions. Each of the localized regions of the site are individually reconstructed in turn and only later merged together into a final complete model. Figure 2 shows the result of combining several regions of the HMS Maori shipwreck to create a whole model. This historic World War 2 destroyer was bombed in 1942 and broke in half. The entire ship measured 115 meters long and this portion is approximately half that length and is located in 16 meter deep water near Valletta, Malta. Core tools in the reconstruction pipeline are Agisoft Photoscan [3] for photogrammetry and FFmpeg [1] for image processing. The reconstruction as shown in Figure 2 captures the shape of the wreck, however, the textures suffer from a fair amount of texture blurring from combining

video segments from multiple passes of the AUV. Our primary contribution is our novel GPU based texturing algorithm written in OpenGL and GLSL, which we now present.

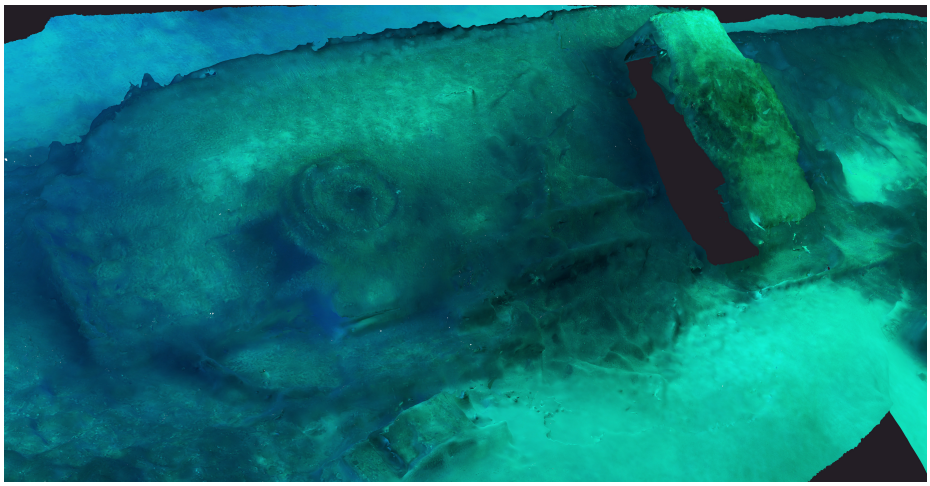

Fig. 2. Example results from our reconstruction pipeline which accurately models the shape of the WW2 wreck, however, the texture is overly smoothed, lacking details found the original images.

3.1 GPU driven texture creation via parallel view dependent selection

Texturing is one of the most critical stages in a reconstruction pipeline with the intent to produce a visually appealing and realistic reconstruction. Texture generation in photogrammetry is almost always done through projection mapping of the scene photos onto the reconstruction. This allows for the original surface and lighting information from the real world scene to be transferred directly onto the reconstruction. Under the assumption that the reconstruction of camera positions is accurate, points on the projected image will line up precisely with their location on the real world subject allowing for a high degree of realism in the resulting texture.

Textures generated using our general reconstruction pipeline using *Photoscan* yielded results missing much of the high frequency detail present in the collected photographs. In general, texturing processes that do not sufficiently disqualify photos or blurred regions of image data are likely to manifest a number of artifacts and potentailly loose a significant degree of captured surface detail. The inherent loss of visibility and accurate color information in underwater scenes introduces unique challenges to the typical process due to the shear amount of video data with low image quality. In addition, with AUV video collection the lighting may change significantly between video passes over the site of interest. This variance in the lighting of the video footage in combination with subtle alignment errors leads to blurry textures as shown in Figure 2.

To create a new less blurred textured, our GPU based algorithm selects pixels from the closest camera (where closest is measured as Euclidean distance). As our input is

thousands of frames (and thus camera positions) from the video camera as it moves through the water on the AUV, our system needs to be able to handle large datasets efficiently.

System Overview: Our texturing system starts with the aligned cameras and reconstructed mesh from the photogrammetry pipeline and reconstitutes the scene within our own C++ and OpenGL application. The system generates a new improved texture for the target mesh through a multi-pass pipeline with the goal of forming the clearest texture using the best image information available. Most stages within the pipeline operate on individual texels of image buffers mapped to the mesh.

The challenge in creating an ideal texture for the reconstructed mesh from 2700 input images is identifying the best view of the geometry from the large number of potential input images. To tackle this problem, the mesh and corresponding texture are first partitioned into ***cells*** which are discrete local regions of the surface which share the same nearest cameras. As such these cells act as a fundamental unit for both resource allocation and computation allowing for some exploitation of locality. This organization is utilized during the final sampling of colors from the source images, when the assets needed by each cell are first loaded into a caching system. Once all assets are loaded, projection from the cell's cameras are used to sample colors from their photographs. These samples are then blended to produce the texturing for all the texels of the cell. The cells partition the entire mesh so processing them all yields a complete texture. A diagram of the system can be seen in Figure 3. The CPU must handle the allocation and management of resources for this process, but the majority of computation is handled by the GPU. The only task which requires a significant portion of CPU time is the task of sorting unstructured cell neighborhood data.

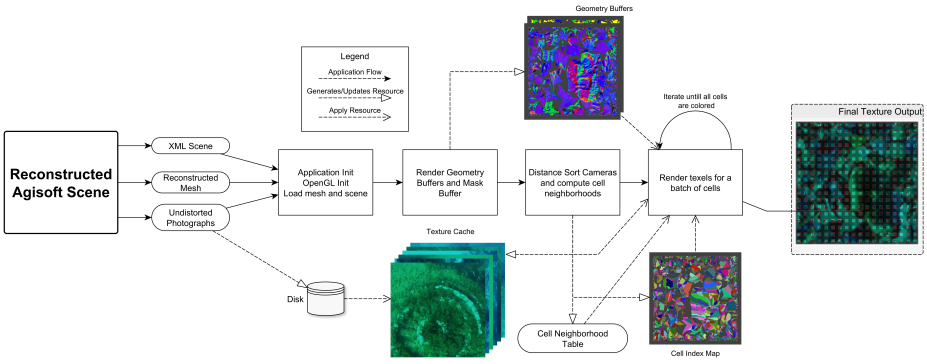

Fig. 3. GPU based view dependent texture generation

Scene Management: The input to our algorithm is the pose and parameters of all the aligned cameras within the scene (which can, for example, be generated with a photogrametry pipeline, in our case from Agisoft Photoscan). Undistorted images are used,

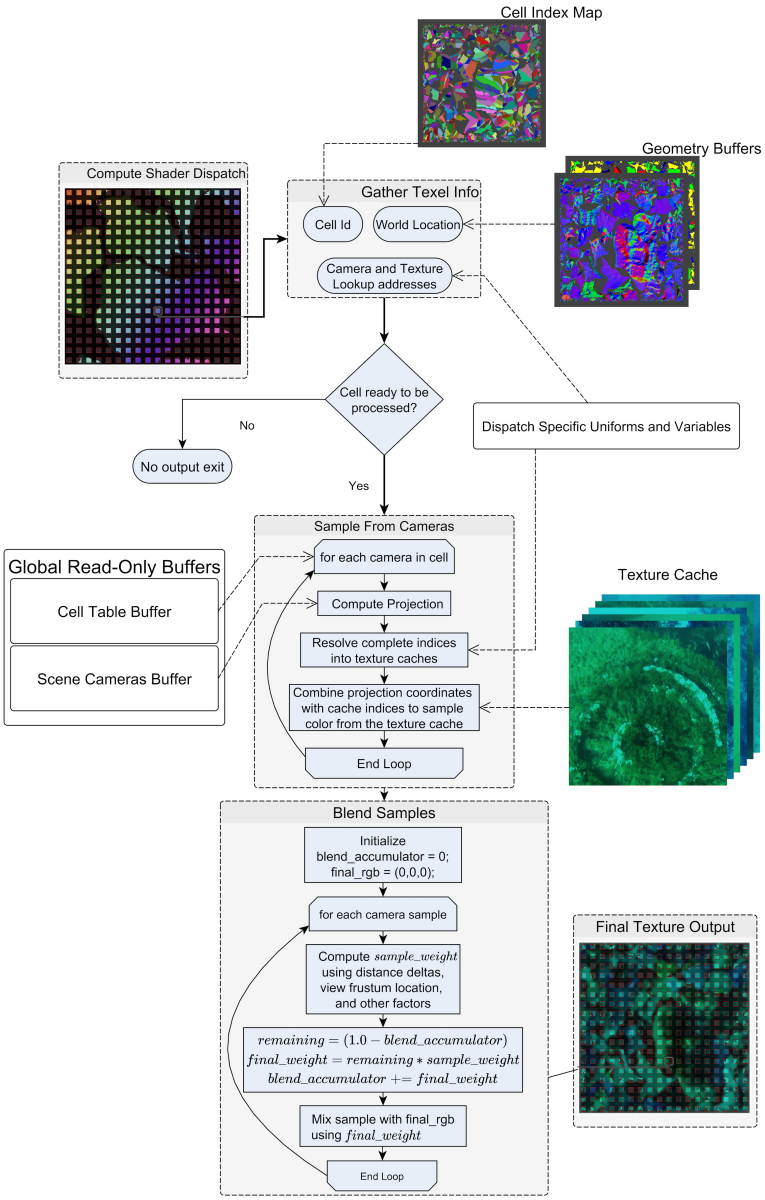

Fig. 4. Process of generating a single texel of texture. Many texels are processed in parallel during the final stage.

thus our system is able to treat each camera as if it were an ideal pinhole camera. Holding all of the potential views (input images) in memory simultaneously is typically not feasible due to the quantity of images (≈ 2700). To work around memory limitations

as well as the texture limits of OpenGL, we consume whatever memory is available using large OpenGL *texture arrays*. Our algorithm fills and continuously updates array entries with the images needed for a particular stage or iteration. These arrays are managed automatically by a caching system we developed which uses *least recently used* replacement. This caching is critical as the time cost of loading images from disk greatly overshadows computations in our system. Furthermore this cache structure allows the system to easily adjust to the quantity of memory available.

The reconstruction is loaded as a triangle mesh and oriented within the same coordinate space as the cameras. The mesh must have a parameterization provided as this mapping is critical to our systems design. Meshes with parameterizations mapping to multiple images will be automatically split and processed as several individual meshes. Resources which are cached during the generation of each texture persist in the cache during run-time, so the performance gained during the generation of one mesh's texture can carry over to the next.

Geometry and UV Mask Buffers: Before any coloring or analysis of the mesh is done, our system computes and stores some basic information about the layout of the texture being generated. This stage, like almost all that follow it, operates on intermediate image buffers which match the dimensions of the texture being generated, but which hold non-color information. The parameterization of the mesh being textured provides a mapping between the image and the mesh's surface such that computations can be run in 2D image space which correspond to the 3D space of the scene. Thus, the first of the intermediate buffers to be created are two geometry buffers into which we store the world space location of each texel as well as the surface normal. These buffers allow all following stages to operate without needing any access to the original triangle mesh. We also simultaneously produce a mask image for the parameterization so that stages further down the pipeline can quickly discard texels which do not map onto the mesh.

Cell Formation: Central to our texturing process is the division of the target surface into smaller fragments whose texture can be derived from the same finite set of source photographs. We reference these fragments using the term *cells* as in our system we derive these fragments from a structure which is very nearly an n^{th}-order voronoi diagram of the scene partitioned by the euclidean distances of its cameras. Where n is the number of cameras we use to texture an individual cell.

The algorithm starts with a compute shader which launches a single invocation for each texel. Each invocation first reads the world space position of the texel from the previously rendered geometry buffer. It then iterates over the cameras in the scene while a small data structure keeps track of the top n nearest cameras. This structure works as a priority queue, but has been simplified for implementation on the GPU in GLSL. Each camera first has its view frustum and view direction checked for visibility of the texel. If the texel is visible the camera is submitted to the queue. Each time a camera is submitted to the queue, it linearly checks the distances of the cameras currently stored, and inserts the new camera if appropriate. It is assumed that the size of the array will be small enough to make losses in efficiency negligible when compared to a typical implementation. The result of this computation is a tuple of n camera identification associated with each texel.

This information is then condensed into the set of all unique tuples that were found in the prior computation. To do so the algorithm iterates over every texel from the prior step and extracts its tuple. Each unique tuple is added into a contiguous array to enable random access later in the algorithm. Likewise, a new integer valued image map is created such that each texel holds an index into this array. Individual tuples characterize a cell, and so the array in combination with the image map form a partitioning of the mesh into cells. A visualization of this cell index mapping can be seen in both Figure 3 and Figure 4 labelled as *Cell Index Map*. This organization of pixels by their tuples is the single step of the algorithm done on the CPU, however it is still relatively inexpensive when compared to other stages of the system.

Final texture: To compute the final texture, the texture caches are appropriately sized based on the maximum number of source photographs the algorithm can concurrently store and access on the GPU and a *batch size* is computed as the maximum number of cells to be processed at once without exceeding space limitations. The set of all cells is then split into batches, and each execute the following steps individually.

First, the the source photographs as well as additional reference images needed by the cells in the batch are loaded. At this time the algorithm loads high dynamic range images which store both a pre-rendered depth map from the perspective of a given camera, as well as a sobel filtered copy of the source image. A separate cache handles these reference images, and when they are requested attempts to load them from disk, or renders them anew if needed. Once all assets are loaded onto the GPU, the algorithm dispatches a compute job for this batch of cells. Again each invocation operates on a single texel and its logic is summarized in Figure 4. Each texel in the final texture is colored from the closest n cameras that can 'see' that texel, resulting in a final texture with higher frequency details as shown in Figure 5.

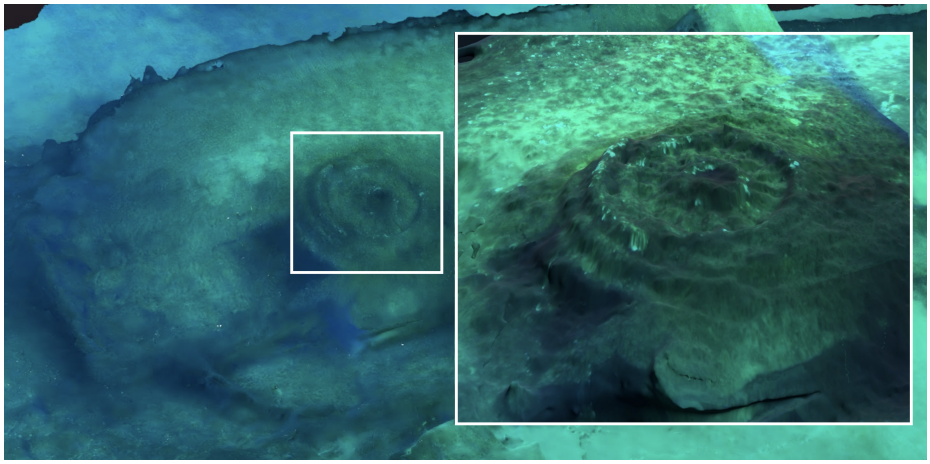

Fig. 5. A close up view of the gun mount on the Maori using our view dependent texture blending (right inset). Contrasted against the gun mount as scene in the Agisoft generated texture.

Color Correction One of the final challenges our system must address is that fact that all the textures created by default from the underwater camera frames exhibit an overly blue tint. For color correction, two separate FFmpeg command line filtergraphs are used. The first is a simple S-curve used to increase contrast on images being used for photogrammetry. In the second FFmpeg filter, a midway equalization filter is applied to the video clip which attempts to match its histogram against a reference clip which was chosen during setup of the pipeline. Next, a color curve filter is applied which is configured using one or more frames from the reference clip. The corrected images can then be used during texture generation to create a model that is more visually distinct allowing a viewer to better distinguish plant matter in an ocean environment. Figure 7 shows our color correction along with a recent commercial alternative we explored [2].

4 Results

We present results from our system for the reconstruction of underwater archaeology sites of interest. Our system is tuned for video data captured from a goPro camera mounted on an AUV. We propose specific system processes to manage the limitations of the AUV trajectory and most importantly an algorithm that utilizes the graphics pipeline and hardware to create view dependent textures to minimize distortion from underwater frames and create an overall clearer final texture. We demonstrate the system and algorithms on data of a known shipwreck in Malta, including a close up of the gun mount on the Maori, Figure 5 and a rendering of the full wreck in Figure 7, with more high frequency image data preserved.

Our texturing system takes advantage of graphics hardware parallelism by utilizing our novel data structures, many GPU computations, and LRU caching. The acceleration achieved as a result of this system is such that most computation times are trivial when compared to the mostly unavoidable cost of loading resources from disk. When operating on a sufficiently small number of images (≤ 256) our system eventually loads all resources needed into memory concurrently, at which point the complete rendering of a 4096x4096 texture image is typically completed in less than 3 seconds. When operating on a much larger set of images (≈ 2700) the time needed to generate a texture of the same size increases to the order of between 450 and 750 hundred seconds, however GPU computation typically represents less than a tenth of a percent of the total time while resource loading occupies above 99%. Similarly the majority of the cell computation stage of our system is computation time on the CPU during organization of camera tuples. Even when each of over 16 million invocations of the computation shader must find the nearest cameras out of the (≈ 2700), the computation is typically completed in under a second. All timings were tested on a system containing Intel® Core™ i7-8700 CPU, 16GB physical memory, and an NVIDIA Titan V. As a result we are able to process a scene with over 2700 virtual cameras and over 5400 total images to produce a texture preserving high frequency information without human intervention. Our system attains a final reconstruction mesh which captures both remarkable geometric and textural detail as seen in Figure 7

One way to measure the improvement provided by our texturing algorithm is to measure the overall blurriness from the original texture and our final texture, measured

in texture space using the variance of the laplacian of each texture. The resulting scalar value is higher for sharper images as they are characterized by more rapid information change. We compare using a basis image formed by first applying a 3x3 guassian blur to both our textures and the Photoscan textures, then equally blending these blurred images together. As shown by figure 6 the variance of the laplacian for our textures are consistently higher than those produced by Agisoft Photoscan.

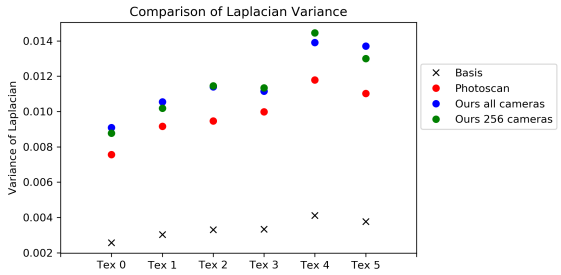

Fig. 6. Graph of the variance of the Laplacian of our textures, Photoscan's, and our basis image.

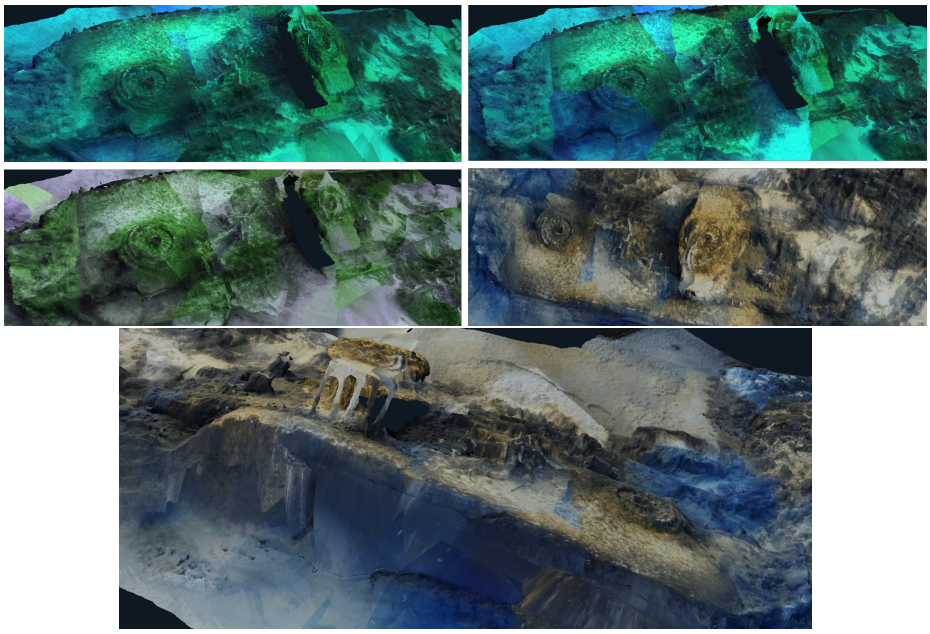

Fig. 7. Our Final result textures. Top: left was generated from 256 cameras, right uses the complete set of approximately 2700 images. The middle two images shows two alternative color correction methods. The final image shows a side view using the second color correction method.

4.1 Limitations and future Work

One stage of our pipeline which could benefit from further research is the alignment of regions during the reconstruction of the archaeological site. Due to the constrained mobility of an AUV, it is not possible to get full coverage of most sites with a continuous shot. Instead many regions must be reconstructed separately then merged. Improved global region alignment, or aligning cameras in image space prior to projection to maximize overlap are areas of future work.

In addition, our method for color correction fits well within our tool-chain, but is rudimentary when compared to some related work. Finally, the abrupt transitions in exposure and color seen in our textures could be improved with a more adaptive blending algorithm within the final stage of our texturing system (of particular interest is multi-band blending [5,4,9]).

ACKNOWLEDGEMENTS

We would like to acknowledge the entire 2018 ICEX team. This material is based upon work supported by the National Science Foundation under Grant No. 1460153.

References

1. Ffmpeg 4.1 (2018), fFmpeg Team
2. Dive+ world's diving community app (2019), life Plus Tech (Shenzhen) Co., Ltd.
3. Agisoft-LLC: Agisoft photoscan (2010)
4. Allène, C., Pons, J.P., Keriven, R.: Seamless image-based texture atlases using multi-band blending. In: 19th International Conference on Pattern Recognition (ICPR'08). p. 10pp. No. 1, France (2008)
5. Baumberg, A.: Blending images for texturing 3d models. In: In Proc. of the British Machine Vision Conference (2002)
6. Bazeille, S., Quidu, I., Jaulin, L., Malkasse, J.P.: Automatic underwater image preprocessing. In: CMM'06 (2006)
7. Callieri, M., Cignoni, P., Corsini, M., Scopigno, R.: Masked photo blending: mapping dense photographic dataset on high-resolution 3d models. Computer and Graphics (2008)
8. Candeloro, M., Mosciaro, F., Srensen, A.J., Ippoliti, G., Ludvigsen, M.: Sensor-based autonomous path-planner for sea-bottom exploration and mosaicking. In: IFAC Conference on Manoeuvring and Control of Marine Craft. pp. 31–36 (2015)
9. Chen, Z., Zhou, J., Chen, Y., Wang, G.: 3d texture mapping in multi-view reconstruction. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Fowlkes, C., Wang, S., Choi, M.H., Mantler, S., Schulze, J., Acevedo, D., Mueller, K., Papka, M. (eds.) Advances in Visual Computing. pp. 359–371. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
10. Dale, L.K., Amato, N.M.: Probabilistic roadmaps-putting it all together. In: Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164). vol. 2, pp. 1940–1947 vol.2 (5 2001). https://doi.org/10.1109/ROBOT.2001.932892
11. Debevec, P.E.: Modeling and Rendering Architecture from Photographs. Ph.D. thesis, University of California at Berkeley, Computer Science Division, Berkeley CA (1996)

12. Fallon, M.F., Kaess, M., Johannsson, H., Leonard, J.J.: Efficient auv navigation fusing acoustic ranging and side-scan sonar. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE Computer Society (2011)
13. von Fock, S.M.T.S., Bilich, S., Davis, K., Viswanthan, V.K., Lobo, Z., Lupanow, J., Clark, C., Gambin, T., Wood, Z.: Pipeline for reconstruction and visualization of underwater archaeology sites using photogrammetry. In: Proceedings of the 2017 ISCA International Conference on Computers and Their Applications (Mar 2017)
14. Iqbal, K., Odetayo, M.O., James, A.E., Salam, R.A., Talib, A.Z.: Enhancing the low quality images using unsupervised colour correction method. In: SMC. pp. 1703–1709 (2010)
15. Li, T.Y., Shie, Y.C.: An incremental learning approach to motion planning with roadmap management. In: Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292). vol. 4, pp. 3411–3416 vol.4 (May 2002)
16. McCarthy, J., Benjamin, J.: Multi-image photogrammetry for underwater archaeological site recording: an accessible, diver-based approach. Journal of maritime archaeology **9**(1), 95–114 (2014)
17. Paull, L., Saeedi, S., Seto, M., Li, H.: Auv navigation and localization: A review. IEEE Journal of Oceanic Engineering **39**(1), 131–149 (2014)
18. Petit, F., Capelle-Laize, A.S., Carre, P.: Underwater image enhancement by attenuation inversionwith quaternions (2009)
19. Poppinga, J., Birk, A., Pathak, K., Vaskevicius, N.: Fast 6-dof path planning for autonomous underwater vehicles (auv) based on 3d plane mapping. In: IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). pp. 1–6. IEEE Press, IEEE Press (2011)
20. Rantanen, M.: Improving Probabilistic Roadmap Methods for Fast Motion Planning. Ph.D. thesis, School of Information Sciences, University of Tampere (Aug 2014)
21. Ruiz, I.T., De Raucourt, S., Petillot, Y., Lane, D.M.: Concurrent mapping and localization using sidescan sonar. IEEE Journal of Oceanic Engineering **29**(2), 442–456 (2004)
22. Van Damme, T.: Computer vision photogrammetry for underwater archaeological site recording in a low-visibility environment. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences (2015)
23. Viswanathan, V.K., Lobo, Z., Lupanow, J., von Fock, S.M.T.S., Wood, Z., Gambin, T., Clark, C.: Auv motion-planning for photogrammetric reconstruction of marine archaeological sites. In: IEEE International Conference on Robotics and Automation (2017)
24. Waechter, M., Moehrle, N., Goesele, M.: Let there be color! large-scale texturing of 3d reconstructions. Computer Vision ECCV 2014 (2014)
25. Wu, J., Bingham, R., Ting, S., Yager, K., Wood, Z., Gambin, T., Clark, C.: Multiauv motion planning for archeological site mapping and photogrammetric reconstruction. Journal of Field Robotics (08 2019)
26. Yamafune, K., Torres, R., Castro, F.: Multi-image photogrammetry to record and reconstruct underwater shipwreck sites. Journal of Archaeological Method and Theory (2016)