SYNCHRONIZATION OF PICAM CAMERAS FOR THREE-DIMENSIONAL STUDY OF DYNAMIC MULTI-DOMAINS NATURAL SCENES

L. Avanthey[1, 2, *], L. Beaudoin[1, 2], C. Villard[1], S. Mellouk[1], R. Treglia[1]

[1] SEAL Research Team (Sense, Explore, Analyse and Learn), ÉPITA, 94270 Le Kremlin Bicêtre, France,
[2] Équipe Acquisition et Traitement, IGN, 94165 Saint-Mandé, France
(loica.avanthey, laurent.beaudoin, charles.villard, sara.mellouk, raphael.treglia)@epita.fr

KEY WORDS: PiCam, Synchronization, Dynamic scenes, Real-time OS, Camera API, Matching

ABSTRACT:

In this article, we study the interest of PiCam and its possibilities offered for the realization of a light payload (small and inexpensive) in order to perform the 3D reconstruction of dynamic scenes (underwater or aerial) in close-range remote sensing. We see that on these observation scales, movements of the scenes due to flora and fauna cannot be ignored if we want these objects to be part of the final model. We review the sensors used in the literature for 3D reconstruction and then present the arguments in favor of PiCam with regard to the constraints posed by the use of light and agile vectors. The main issue is the synchronization of these low cost sensors, which is not native: we explain the different steps to obtain a satisfactory synchronization rate with regard to the dynamism of the studied scenes and present the results obtained.

1. INTRODUCTION

In close remote sensing, when we study natural environments, being very close makes us sentivite to the dynamism of the scene. Problems arise mainly for photogrammetric studies that performs 3D reconstructions of these dynamic scenes. Indeed, if the objects observed have moved between the acquisitions from the two stereoscopic points of view, their local geometry no longer conforms to the global geometry and then it will be impossible to find their three-dimensional position correctly. Considered as erroneous pairings, these points are therefore eliminated during the geometric filtering.

We are interested in aerial or underwater study environments. In close-range remote sensing, three-dimensional reconstruction works are based on images acquired with a centimetric GSD (Ground Sample Distance) for a lot of aerial studies (Bulatov et al., 2011, Kng et al., 2011, Rossi et al., 2017) and often a sub-centrimetric GSD in most aerial or underwater studies: about 50 mm (Aicardi et al., 2018), 3 to 5 mm (Skarlatos et al., 2012, Menna et al., 2018), 1 to 2 mm (Henderson et al., 2013, Burns et al., 2015) or even less than 1mm (Schmidt, Rzhanov, 2012, Gracias et al., 2013, Germanese et al., 2019).

With this GSD, the observed small movements in the scene are very quickly visible: sensitivity to dynamism becomes a real problem, especially since the GSD is small. This is illustrated in Figure 1. We took images with sub-centimetric GSDs in aerial and underwater environments. They represent typical scenes of an acquisition campaign in good conditions, that is to say with good visibility and low wind or current. The sensor is static to not integrate a disparity in the observation of movements. The displacement maps are calculated on an interval up to one second for each of the examples. They show that movements are omnipresent and non-negligible: of the order of several tens of pixels, which according to the GSD represents up to 10cm.

GSDs are proportional to the viewing distance for a given sensor. In the marine environment, due to the rapid limitation of visibility, we will rather work over short distances of a few meters at most. The effects will therefore be more impacting than in the air where the observation distances are a few tens of meters.

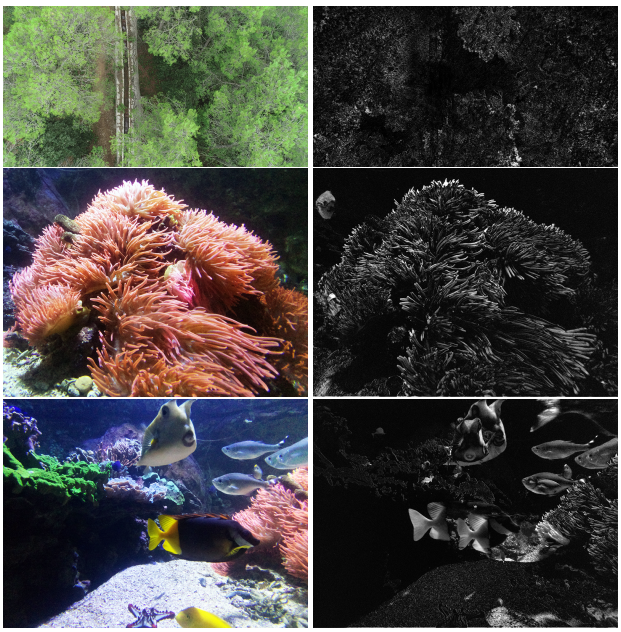

Figure 1. Sensitivity to dynamism for several typical scenes of an acquisition campaign: an image of a scene (left) and its displacement map (right) calculated over an interval of 1s. The magnitude of the displacements is up to several tens of pixels.

To reconstruct dynamic objects (fauna or flora for example), we need to freeze the movement. In other words, the local relative displacement taking place between two shots should not exceed the size of the GSD (Avanthey et al., 2016). This is possible if the following two conditions are fulfilled: several sensors are required and they must be synchronized.

The use of a single sensor (temporal stereoscopic pairs) be-

* Corresponding author

comes de facto insufficient, because its maximum acquisition frequency, constrained by the stereoscopic basis, is then too low. Furthermore, in the case of a multi-view reconstruction, it is then necessary to have at least three synchronized sensors so that the points are preserved in the global reconstruction.

We will present in section 2 the different sensors used in the literature to perform 3D aerial or underwater reconstructions as well as the selection criteria that led us to work with PiCams sensors. Then, in section 3 we expose the strategy which allow us to obtain a good synchronization rate and the obtained results are presented and commented in section 4. Finally, section 5 conclude on this work and discusses the envisaged perspectives.

2. PICAMS FOR 3D RECONSTRUCTION IN CLOSE-RANGE REMOTE SENSING

2.1 Sensors used for aerial or underwater 3D reconstruction in close-range remote sensing

The criteria for choosing a sensor is a matter of compromise. In terms of quality, interests relate to the resolution (number of pixels per unit of length or density), sensitivity and noise, but also to the relationship between the focal length and the physical size of a pixel: the larger this ratio for a given sensor size, the more it will be possible to observe the scene from a distance for a given GSD. However, a too high focal length will result in a narrower field of vision.

In terms of control, the important functions for photogrammetric work are the ability to deactivate all automatic adjustments, the possibility of acquiring images at regular intervals with a high frequency, the ability to synchronize with other devices, the ability to be controlled by a computer and the possibility of retrieving images on the fly so that they can be processed in real time if necessary.

Finally, in terms of physical constraints, important criteria are size, weight as well as the possibility of choosing and changing the optics. As for the capture, it comes in two types: video or photography. The first mode offers a much higher acquisition frequency in return for a lower resolution and its recording format requires to extract frames to be able to be processed.

The sensors used in work relating to our subject in the literature can be mainly grouped into three categories: professional cameras, consumer high-end cameras and consumer entry-level cameras.

- **Professional cameras**: Prosilica GC1380, GE1900 and GT1920 from Allied Vision (Johnson-Roberson et al., 2010, Beall et al., 2011, Henderson et al., 2013, Drap et al., 2015), DragonFly, BumbleBee2, Flea from PointGrey and Chameleon3 (Jenkin et al., 2010, Servos et al., 2013, Pierce et al., 2018, Detry et al., 2018), Typhoon from Tritech (Botelho et al., 2009, Fillinger, Funke, 2013), OE14 502 or 500 from Kongsberg (Sedlazeck et al., 2009, Fillinger, Funke, 2013), PixelFly from PCO (Pizarro et al., 2004, Foley et al., 2009), Multi SeaCam from DeepSea Power & Light (Fillinger, Funke, 2013), FCBH 11 from Sony (Fillinger, Funke, 2013), RealSense D200 from Intel (Shang, Shen, 2016, Digumarti et al., 2016) or CamLight from IGN (Zhou et al., 2018).

Controlled by a computer, these are the sensors that offer the widest spectrum of settings: acquisition parameters, durations and delays, electronic triggers, etc. The acquired data can be transmitted to a computer for real-time processing.

- **Consumer DSLR (Digital Single-Lens Reflex) cameras**: EOS 5D Mark II, M, 600D and 550D from Canon (Nicosevici, Garcia, 2008, O'Byrne et al., 2015, Rossi et al., 2017, Germanese et al., 2019), K5 from Pentax (Burns et al., 2015), A700 from Sony (Diamanti, Vlachaki, 2015) or D70, D200, D300, D700, D750 and D7000 from Nikon (Barazzetti et al., 2010, Bianco et al., 2011, Drap, 2012, Gintert et al., 2012, Menna et al., 2018).

These sensors offer a wide choice of optics and are widely used in photogrammetry for the quality of the produced images. Intended for advanced users of the general public, they allow some adjustments, but are rarely designed to be controlled electronically. The images are saved on a memory card and they are not available for real-time processing. The intervalometer functionality is essential (shots at regular intervals without the need to press the shutter button) although it often exhibits a certain drift over time (non-constant difference between two shots) and its rate is often limited at one frame per second.

- **Consumer compact cameras and entry-level modules**: PowerShot G9, D10, G12 and A620 from Canon (Fillinger, Funke, 2013, Capra et al., 2015, Vlachos et al., 2019), Coolpix 995 from Nikon (Fillinger, Funke, 2013), FX35 Lumix from Panasonic (Steffen, Forstner, 2008), NEX-5N and RX1R from Sony (Daftry et al., 2015, Jiang, Jiang, 2019), HD Hero, HD Hero 2, Hero 3, Hero 3+ Silver or Black Edition and Hero 4 from GoPro (Gintert et al., 2012, Schmidt, Rzhanov, 2012, Nelson et al., 2014, Gatziolis et al., 2015, Capra et al., 2015, Rende et al., 2015, Barrile et al., 2017, Song et al., 2019, Pahwa et al., 2019), HD Sport Pro from Intova (Capra et al., 2015), Inspire 1 Zenmuse X3-FC350 from DJI (Gupta, Shukla, 2017, Singh et al., 2018), Kinect from Microsoft (Teixeira, Chli, 2016) or PiCam from RaspberryPi (Mazzei et al., 2015).

The range of compacts offers fixed optics sensors with a good quality, price and size ratio. Intended for the general public, the room for maneuver on their settings is low, especially for compact cameras, although the latest models are starting to expand their options. We find the essential function of an intervalometer, with the same problem on the regularity, but offering rates which this time go down under the second. In this category of low-cost sensors, we also have seen the arrival on the market of small high-definition camera modules, fully electronically controllable (such as the PiCam).

2.2 Choice of the camera

To choose a camera, our main criteria are its size, its weight and its price to be compatible with the use of light platforms. Indeed, in close-range remote sensing, the size of the studied areas is very small compared to those of aerial or space remote sensing. The main interest of these studies therefore does not lie in the spatial coverage but rather in the on demand acquisition capacity enabled by the operational flexibility of the vectors used. These vectors, and by extension their payload, must be as agile as possible, that is to say, compact and light. Low

cost is also an important criterion as we have to duplicate the cameras in our case (2 or 3 at least).

These criteria de facto eliminate DSLR cameras as well as the vast majority of professional sensors and encourage to opt for entry-level sensors in the professional or general public categories.

Among them, the GoPros, uEye and PiCam v2 sensors fit particularly well our constraints. The PiCam v2 of Raspberry Pi which was released in 2016 (3280×2464px, 1.4μm pixel size, $\sim$30€) seems to offer a good compromise between the two other sensors tested by (Avanthey et al., 2016) for dynamic environments: the uEye camera from IDS (entry-level professional sensor, 1280×1024px, 5.3μm pixel size, $\sim$500€) and the Hero 2 camera from GoPro (a sports camera in the consumer category, 3840×2880px, 1.6μm pixel size, $\sim$200€, similar today to a GoPro Hero 7 Silver Edition). In terms of cost and weight, the PiCam is far below these two cameras, even by adding a board for control and data storage (Raspberry Pi Zero or 3+ for example: 10 to 40€).

In terms of image quality, the PiCam is better than the uEye and close to the GoPro which is good enough to give satisfactory results in 3D reconstruction (Bernardina et al., 2016), even if the adjustments of the acquisition parameters and especially the native post-treatments are not as advanced. As an example, the figure 2 presents images taken at the same time with Gopros and PiCams in water, the medium that poses the most problem on the quality of images. Studies carried out by (Venkataraman et al., 2013, Santise et al., 2017, Piras et al., 2017) shows that the PiCam sensor is suitable for photogrammetric work.

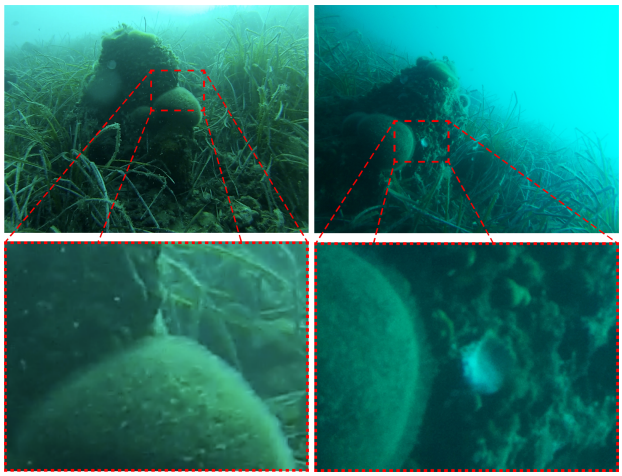

Figure 2. Visual comparison of the quality of the cameras: on the left an image taken by a GoPro with a zoom of a part in the lower right corner, and on the right an image taken at the same time by a PiCam, with a zoom of an equivalent portion of the image in the lower right corner.

Simultaneous acquisitions (mechanical or, better, electronic trigger signal), which lead to synchronizations at best at 500ms, are rarely sufficient with regard to the dynamism of natural areas as we can see in (Avanthey et al., 2016) with the GoPro. In terms of control, the PiCam gets closer to the uEye camera and lets hope for a synchronization time as good, or even better, than the one obtained for the latter (5ms) by (Beaudoin et al., 2015) without its software instability problems related to the driver. In practice, obtaining sufficient and stable synchronization on inexpensive light sensors is problematic because it is not a native function. We will see in the following section an original architecture solution implemented to precisely synchronize several PiCam cameras.

3. SYNCHRONIZATION OF ACQUISITIONS MADE BY SEVERAL PICAMS

The PiCam cannot work alone and needs to be driven by a program on a computer, a Raspberry Pi in our solution. We use as many computers as there is cameras. So the problem will be to synchronize all the acquisition programs running on these differents computers.

In a program, each task will have a variable launch delay, called *jitter*, which corresponds to the time elapsed between the moment when the task is placed in the execution queue and the moment when it is actually executed. Launching the same task at the same time on different computers will thus result in different delays of execution. Two conclusions are drawn from this. First, the greater the number of tasks, the more the accumulated variations in these delays will contribute to creating a high difference between two shots launched simultaneously (desynchronization). To minimize these effects, bypassing intermediate libraries to access the nearest registers of the sensor without unwanted additional processing is needed. This point is discussed in section 3.1. Second, the greater the variations in delay, the greater the desynchronization. We must therefore try to minimize them as much as possible. This other point, which directly concerns the behavior of the OS (Operating System), is addressed in section 3.2. Finally, the communication part, which allows the cameras to trigger their shots at the same time is discussed in section 3.3.

3.1 Choice of appropriate libraries to control the driver

The PiCam is connected to the GPU (Graphics Processor Unit) of the RaspberryPi, called VideoCore. So, all the software solutions need to deal with this hardware level. Amongst "off-the-shelf" solutions, the most used is Raspicam that provides an API (Application Programming Interface) to use the PiCam. This library raises a major problem by its behavior: it is not, in reality, photographic captures but a recovery of frames from a video stream. The function which requests the acquisition of an image (*grab*) only signal to the acquisition thread the necessity to save the next available frame. Either, the frame has just been taken and it is ready for saving, or it is still being acquired and it will be saved after a short delay (see figure 3). Launching the camera flows at the same time does not guarantee a synchronization of acquisitions. This way is not adapted to our constraints.

If we go a step deeper in the software architecture, there are two main APIs that provide hardware abstractions to access and control the data of the camera in a standardized way via the GPU (see figure 4). The first one, used by RaspiCam, is MMAL (Multi-Media Abstraction Layer) which is a proprietary API created and implemented by Broadcom. The second one is OpenMax which is a semi-open API (BSD license) created by Khronos group (Industrial Consortium, which also produced the OpenGL and Vulkan APIs for 3D). Although older and less maintained, OpenMax is better documented than MMAL by its open-source nature and therefore easier to use in a manner suited to our needs.

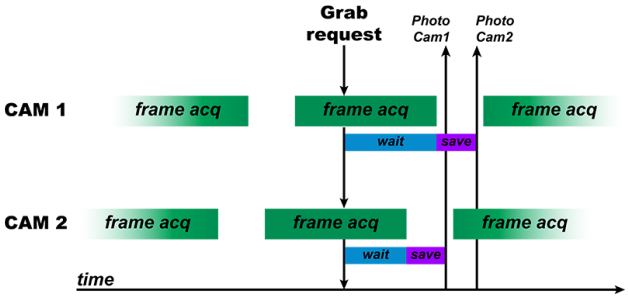

Figure 3. Process capture via the Raspicam library: grab is requested simultaneously during the capture process and is actually carried out for each sensor when it has just finished acquiring a frame, hence a certain delay between the two recovered acquisitions.

Our software solution uses a wrapper of OpenMax, called Omxcam, which allows us to finely control the shooting with a particular attention to avoid unnecessary or superfluous tasks that contribute to widening the synchronization gap.

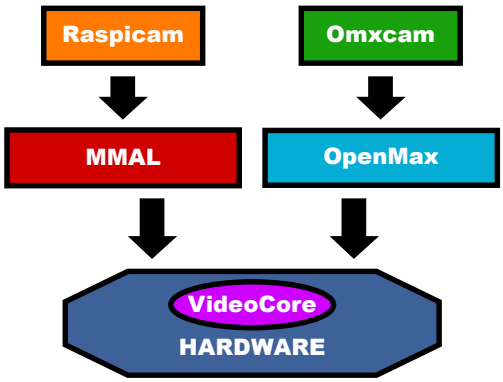

Figure 4. Diagram representing the different APIs and libraries for controlling cameras on the Raspberry Pi.

3.2 A real-time OS to control the jitter

As seen before, precise control of the launch date and execution time (*jitter*) of the acquisition tasks is essential. The control of these parameters is only done by the OS, because it is the only one that has the overview of all the active processes and their priority. To facilitate maintenance and security, we choose to stay on open-source solutions of the Linux type which are supported a lot by the community. OS can be classified into two main families: classic OS and real-time OS.

The latter are distinguished from the former by the importance accorded to managing precisely the *jitter* and the tasks order. To respect our contrainsts, the choice of a real-time OS is therefore essential, because in classic OS this *jitter* can easily go up to several milliseconds per task within a program. In the real-time family, there is a distinction between hard and soft real-time OS, depending on the tolerance you choose for respecting the *jitter*. A hard real-time OS guarantees a launch delay in all cases limited by a constant (max *jitter*). If this condition is not met, the task is automatically killed because it is considered to have failed. This type of OS is widely used in critical systems in aeronautics for example. These OS include FreeRTOS or VxWorks.

A soft real-time OS does not guarantee that the launch time and therefore tasks will not be killed even if it takes a relatively long time. But is designed so that the mean *jitter* is comparable to the one of the "hard" real-time OS, and so remains much shorter than the average time on a classic OS.

Another pragmatic difference and not the least: a hard real-time OS is minimalist compared to a soft real-time OS which natively offers many additional features and is very close to a classic OS because it implements the POSIX API in particular (used among others by the libc, the standard library of C). In other words, developing a program on a soft real-time OS is much simpler, faster and maintainable than on a hard real-time OS. Compared to our needs, the performance offered by the soft real-time OS is sufficient and its ease of use, especially when working with images, fully justifies this choice.

Among the OS of this soft real-time family, two solutions emerge: Linux_PreemptRT (standard linux kernel to which the PREEMPT-RT patch has been applied) or Xenomai. To run a program in real time, the developers of Linux_PreemptRT directly modified the linux kernel in depth: as a result, all programs run in real time. The developers of Xenomai have made another strategic choice: to co-exist two Linux kernels, one real time and the other a classic one. Real time programs only run on the latter. Compared to our problem, Xenomai is not a good choice because to make our capture program be able to be executed in real time, it would be necessary to recode the driver of the camera so that it can be integrated in the real-time kernel. This is particularly complicated because the drivers are often proprietary and very little documented. This coding step is useless with Linux_PreemptRT since everything runs natively in real time. Its maximum jitter is also compatible with our use (Arthur et al., 2007, Dias et al., 2014). So we chose this OS for the final solution.

3.3 Communication between cameras to synchronize shots

Finally, the last key point of synchronization is the physical transmission of the acquisition order between the cameras. In a first attempt, we used a high / low synchronization signal (trigger). In addition to the ground and the trigger, we added a third line of control used by the cameras to tell when they finished their acquisition (sensor ready). If one failed, all the other currently acquired images are not saved.

However, with three lines per camera, we can also use another communication protocol: the I^2C (Inter-Integrated Circuit). This protocol respects our constraints of fixed communication time between the cameras and allows a speed up to 400Kb/s. The advantage is that with this protocol, other interesting information can be transmitted bi-directionally between the cameras such as the internal parameters of each sensor (shutter speed, ISO, color balance, camera ID, etc.) for example. This is the solution that we have chosen (see figure 5).

To summarize, our solution is based on the use of PreempRT OS, Omxcam and the I^2C protocol between cameras.

4. RESULTS

To assess the obtained synchronization, measurement campaigns were carried out with 3 synchronized PiCams. The first one uses a digital timer to the thousandth of a second displayed on a screen. It shows that the use of RaspiCam (see section 3.1) on a non-real-time OS (Linux) gave a synchronization delay of 50ms (reference time), which is already better than for GoPro cameras by a factor of 10.

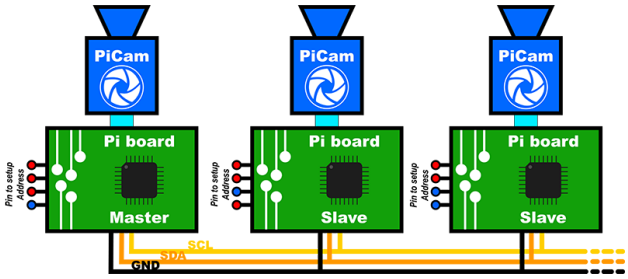

Figure 5. Final bench of several synchronized PiCams.

By switching to the Omxcam library, the shots of the timer do not differ: this means that the synchronization difference has fallen below 20ms. Indeed, a standard screen refreshes at 60 Hz, that is to say, every 16.7ms.

In order to be able to measure the synchronization delay more precisely, we completed the experimental bench with a chaser composed of 10 LEDs alternated in red, green and blue and framed by 2 white LEDs (see figure 6). Each RGB LED successively lights for 1 ms. The white LEDs only light up every two cycles. In this way we are able to distinguish all milliseconds over a period of 20ms. At the same time, the shutter speed of the sensor must be reduced as much as possible to limit its integration time (the entrained lack of light is not annoying because we observe bright spots). This is in order to not obtain images where all the LEDs are on (which is the case for our eyes, which have an integration time of 50ms) and where it would therefore be impossible to observe an offset in the captured sequence on the synchronized views.

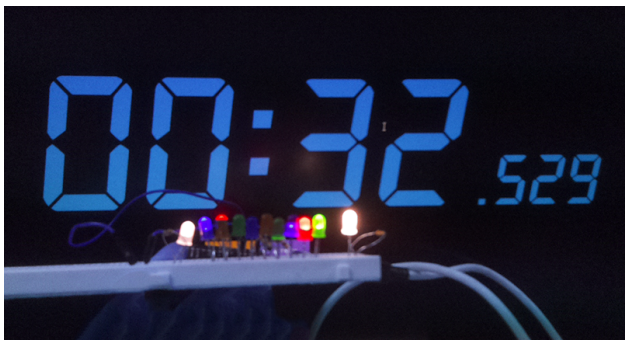

Figure 6. The experimental bench to finely measure the synchronisation rate of the acquisition: screen timer with a refresh time of 20ms and LEDs chaser with a refresh time of 1ms.

Thanks to this bench, we were able to measure that the synchronization difference with the Omxcam library on a non-real-time OS was 8ms on average: a 6-fold improvement over RaspiCam. Using the soft real time OS PreemptRT made it possible to descend strictly below the millisecond, without drift over time.

The objective of the next test campaign is to verify that the quality of the image synchronization is sufficient to deal with dynamic objects likely to be found in the study areas. If the synchronization is successful and the movement has been frozen, then the stereoscopic pair can be treated as a classic pair by the 3D reconstruction chain. Namely, all the elements of the scene can be matched correctly. On the other hand, if there remains local movements, these matched points will be contrary to the global geometry and will then be eliminated.

We will therefore rely on this matching criterion to assess the adequacy of synchronization in our tests. The multi-domain database used is composed of stereoscopic pairs containing various dynamic objects (birds, fish, anemones, coral reefs, caustics, etc.). Synchronized pairs have a time difference of about 1 millisecond whereas desynchronized pairs have a time difference of about 1 second. The cameras do not move during the acquisition series so as not to add the speed of movement of the sensor in the analyzes. We crossed the successive images taken by the sensors to form the desynchronized pairs (the nth image of sensor 1 with the nth + 1 image of sensor 2). In the case where one chooses to use the two successive images of the same sensor as a desynchronized pair, care must be taken to remove the non-overlapping portion linked to the base in order to be able to compare the results with the synchronized pairs. We also try to have images where movements are evenly distributed over the image to significantly assess its influence on the matching rate.

The matching algorithm used (Avanthey et al., 2016) relies on a self-adapting Harris point detector and on local statistical filtering on the vector flow formed by the matched points to discard outliers (bad matches). The results of our tests show that we obtain an average rate of 45% of good matches (inliers) on all of our synchronized pairs. This results is close to the rate of 53% of inliers obtained by (Avanthey et al., 2016) on a database of 200 images of relatively static scenes taken under various conditions. Figures 7 and 8 show examples of our results.

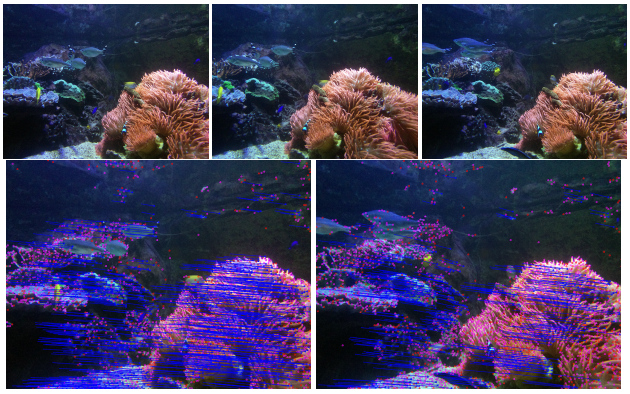

Figure 7. Example of matching result on a synchronised pair (left) and desynchronised pair (right) of a reef scene: we get two times more inliers for the synchronized pair than the desynchronised pair. Most of the inliers of the desynchronised pair are on statics parts (coral and ground).

We will note that anemones, accompanied by clownfish (genus *Amphiprion*), are a complex case for matching algorithms because the textures are relatively plain and the patterns very similar. On this type of image, the results obtained are around 30% of good pairings (see figure 9).

The results on caustics, which produce very fast movements, are high (very discriminating patterns, good contrast): the number of inliers is close to 60% (see figure 10). However, as we can see in figure 11, the maching systematically fails on the light spots over the surface of the water. Their speed seems too high for our synchronisation rate.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-1-2020-277-2020 | © Authors 2020. CC BY 4.0 License.

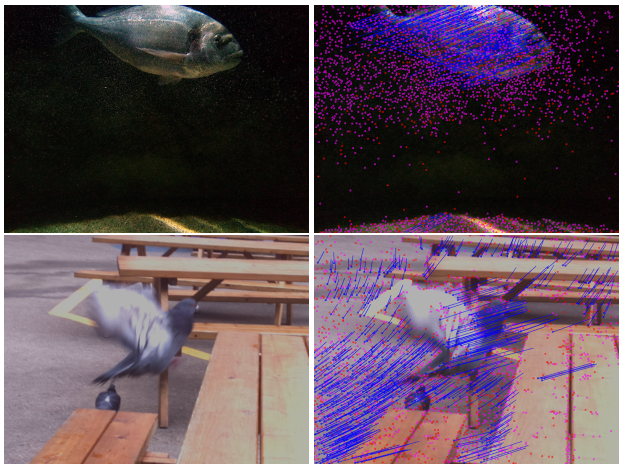

Figure 8. Example of matching result for synchronised pair with fauna: the pair of fish (top) offers 38% of inliers (small part of the image) and the pair of bird (bottom) offers 42% of inliers.

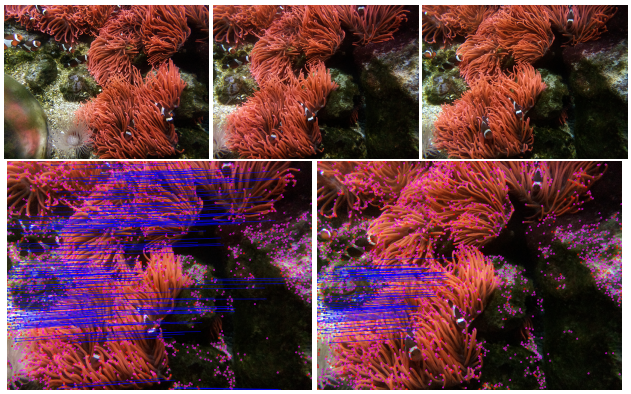

Figure 9. Example of matching result on a synchronised pair (left) and desynchronised pair (right) of anemones: we get more than three times more inliers for the synchronized pair than for the desynchronised pair. Again, most of the inliers of the desynchronised pair are on statics parts (ground).

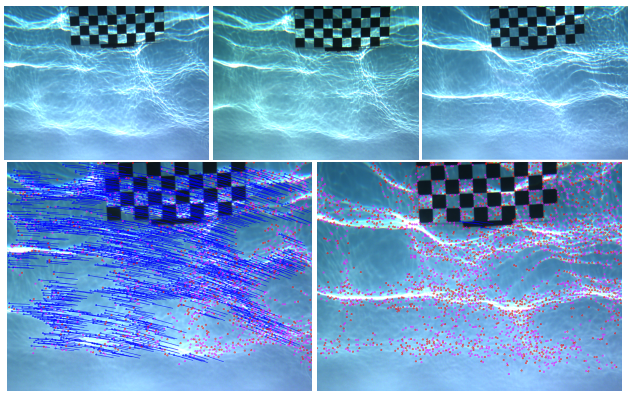

Figure 10. Example of matching result on a synchronised pair (left) and desynchronised pair (right) of caustics: we get 80% of inliers for the synchronized pair and less than 1% for the desynchronised pair.

Our results also show that if the images are out of synchronisation for even a second, we lose 40% of good pairings on moderate movements (anemones, fish, birds, etc.) as we have seen in the figures 7 and 9. And on fast movements, like those of caustics for example (see figure 10), we lose more than 95% of good pairings during a desynchronization of a second.

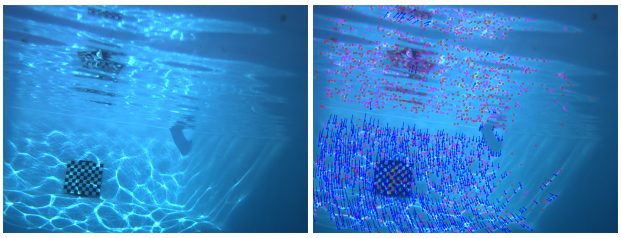

Figure 11. Light spots on the surface of the water (top) seem to have a higher speed than that of caustics (bottom) and cannot be matched with our synchronization delay of 1 ms.

These results show that the quality of our synchronization is such that we obtain almost as many inliers on the dynamic scenes as on the static scenes.

5. CONCLUSION

We have shown that the PiCam is an interesting sensor with regard to the agility and dimensions constraints of the vectors for close-range remote sensing. Its image quality is close to that of the GoPro camera and the possibility of finely controlling it by computer makes it possible to obtain a sufficient synchronization rate for the study of dynamic scenes. The synchronisation delay that we have achieved through our solution that allow a precise sensor and *jitter* control does not exceed one millisecond, i.e. 50 times smaller than the reference time, and 500 times smaller than the GoPro or classic intervalometers. In the results, we have shown that this synchronisation rate allows a near similar matching rate than those obtained on static images of natural scenes. We have also shown that the matching rate is very good on highly dynamic scenes like those with caustics. The limit is reached for the movements of light spots on the surface of the water.

The PiCam sensor therefore presents itself as a very good alternative to GoPro for this kind of work. However, there are several points to dig. On the one hand, it must be check that the quality of the images will be sufficient to carry out dense 3D reconstructions. On the other hand, the stability of the camera in terms of blurring and contrast in relation to variations in speed and visibility must also be check. Finally, to improve the frequency of shots to allow complete coverage of an area, it will be interesting to dive deeper into the camera control APIs of the Raspberry Pi.

ACKNOWLEDGEMENTS

We thank the Seaquarium of Grau-du-Roi (Gard) for allowing us to carry out acquisitions in their basins.

REFERENCES

Aicardi, I., Chiabrando, F., Lingua, A., Noardo, F., 2018. Recent trends in cultural heritage 3D survey: The photogrammetric computer vision approach. *Journal of Cultural Heritage*, 32.

Arthur, S., Emde, C., Mc Guire, N., 2007. Assessment of the Realtime Preemption Patches (RT-Preempt) and their Impact on the General Purpose Performance of the System. *Real-Time Linux Workshop*.

Avanthey, L., Beaudoin, L., Gademer, A., Roux, M., 2016. Tools to Perform Local Dense 3D Reconstruction of Shallow Water Seabed. *Sensors*, 16(5), 712.

Barazzetti, L., Remondino, F., M., S., 2010. Automation in 3D Reconstruction: Results on Different Kinds of Close-Range Blocks. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII, 55–61.

Barrile, V., Gelsomino, V., Bilotta, G., 2017. UAV and Computer Vision in 3D Modeling of Cultural Heritage in Southern Italy. *IOP Conference Series: Materials Science and Engineering*, 225.

Beall, C., Dellaert, F., Mahon, I., Williams, S. B., 2011. Bundle Adjustment in Large-Scale 3D Reconstructions Based on Underwater Robotic Surveys. *IEEE OCEANS Conference*, 1–6.

Beaudoin, L., Avanthey, L., Gademer, A., Roux, M., Rudant, J.-P., 2015. Dedicated Payloads for Low Altitude Remote Sensing in Natural Environments. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-3/W3, 405-410.

Bernardina, G. R. D., Cerveri, P., Barros, R. M. L., Marins, J. C. B., Silvatti, A. P., 2016. In-air Versus Underwater Comparison of 3D Reconstruction Accuracy Using Action Sport Cameras. *Journal of Biomechanics*, 51, 77–82.

Bianco, G., Gallo, A., Bruno, F., Muzzupappa, M., 2011. A Comparison Between Active and Passive Techniques for Underwater 3D Applications. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, XXXVIII-5/W16, 357–363.

Botelho, S. S. d. C., Drews, P., Oliveira, G. L., Figueiredo, M. D. S., 2009. Visual Odometry and Mapping for Underwater Autonomous Vehicles. *IEEE Regional Latin American Robotics Symposium (LARS)*, 1–6.

Bulatov, D., Solbrig, P., Gross, H., Wernerus, P., Repasi, E., Heipke, C., 2011. Context-Based Urban Terrain Reconstruction from UAV-Videos for Geoinformation Applications. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-1/C22.

Burns, J. H. R., Delparte, D., Gates, R. D., Takabayashi, M., 2015. Integrating Structure-From-Motion Photogrammetry with Geospatial Software as a Novel Technique for Quantifying 3D Ecological Characteristics of Coral Reefs. *PeerJ*.

Capra, A., Dubbini, M., Bertacchini, E., Castagnetti, C., Mancini, F., 2015. 3D Reconstruction of an Underwater Archaeological Site: Comparison Between Low Cost Cameras. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, XL-5/W5, 67–72.

Daftry, S., Hoppe, C., Bischof, H., 2015. Building with Drones: Accurate 3D Facade Reconstruction using MAVs. *IEEE International Conference on Robotics and Automation*.

Detry, R., Koch, J., Pailevanian, T., Garrett, M., Levine, D., Yahnker, C., Gildner, M., 2018. Turbid-Water Subsea Infrastructure 3D Reconstruction with Assisted Stereo. *IEEE OCEANS Conference*, 1–6.

Diamanti, E., Vlachaki, F., 2015. 3D Recording of Underwater Antiquities in the South Euboean Gulf. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1, 93–98.

Dias, R. A., Emanoel de Souza, T., Noll, V., 2014. Experimental Analysis of the Linux RT-patched for Data Acquisition applied to Power Sector. *International Journal of Computer Applications*, 101(6), 43–49.

Digumarti, S. T., Chaurasia, G., Taneja, A., Siegwart, R., Amber, T., Beardsley, P., 2016. Underwater 3D Capture using a Low-Cost Commercial Depth Camera. *IEEE Winter Conference on Applications of Computer Vision*.

Drap, P., 2012. Underwater Photogrammetry for Archaeology. *Special Applications of Photogrammetry*, InTech, 111–136.

Drap, P., Merad, D., Hijazi, B., Gaoua, L., Nawaf, M. M., Saccone, M., Chemisky, B., Seinturier, J., Sourisseau, J.-C., Gambin, T., Castro, F., 2015. Underwater Photogrammetry and Object Modeling: A Case Study of Xlendi Wreck in Malta. *Sensors*, 15(12), 30351–30384.

Fillinger, L., Funke, T., 2013. A New 3-D Modelling Method to Extract Subtransect Dimensions from Underwater Videos. *Ocean Science*, 9(2), 461–476.

Foley, B. P., Dellaporta, K., Sakellariou, D., Bingham, B. S., Camilli, R., Eustice, R. M., Evagelistis, D., Ferrini, V. L., Katsaros, K., Kourkoumelis, D., Mallios, A., Micha, P., Mindell, D. A., Roman, C., Singh, H., Switzer, D. S., Theodoulou, T., 2009. The 2005 Chios Ancient Shipwreck Survey: New Methods for Underwater Archaeology. *Hesperia*, 78, 269–305.

Gatziolis, D., Lienard, J. F., Vogs, A., Strigul, N. S., 2015. 3D Tree Dimensionality Assessment Using Photogrammetry and Small Unmanned Aerial Vehicles. *PloS ONE*, 10(9).

Germanese, D., Pascali, M. A., Berton, A., Leone, G. R., Moroni, D., Jalil, B., Tampucci, M., Benassi, A., 2019. Architectural heritage: 3d documentation and structural monitoring using uav. *Visual Pattern Extraction and Recognition for Cultural Heritage Understanding*, 1–12.

Gintert, B., Gleason, A. C. R., Cantwell, K., Gracias, N., Gonzalez, M., Reid, R. P., 2012. Third-Generation Underwater Landscape Mosaics for Coral Reef Mapping and Monitoring. *International Coral Reef Symposium*.

Gracias, N., Ridao, P., Garcia, R., Escartn, J., L'Hour, M., Cibecchini, F., Campos, R., Carreras, M., Ribas, D., Palomeras, N., Magi, L., Palomer, A., Nicosevici, T., Prados, R., Hegeds, R., Neumann, L., de Filippo, F., Mallios, A., 2013. Mapping the Moon: Using a Lightweight AUV to Survey the Site of the 17th Century Ship La Lune. *IEEE OCEANS Conference*, 1–8.

Gupta, S. K., Shukla, D. P., 2017. 3D Reconstruction of a Landslide by Application of UAV and Structure from Motion. *AGILE conference*.

Henderson, J., Pizarro, O., Johnson-Roberson, M., Mahon, I., 2013. Mapping Submerged Archaeological Sites using Stereo-Vision Photogrammetry. *International Journal of Nautical Archaeology*, 42(2), 243–256.

Jenkin, M., Verzijlenberg, B., Hogue, A., 2010. Progress Towards Underwater 3D Scene Recovery. *ACM C* Conference on Computer Science and Software Engineering*, 123–128.

Jiang, S., Jiang, W., 2019. UAV-Based Oblique Photogrammetry for 3D Reconstruction of Transmission Line: Practices and Applications. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W13, 401–406.

Johnson-Roberson, M., Pizarro, O., Williams, S. B., Mahon, I., 2010. Generation and Visualization of Large-Scale Three-Dimensional Reconstructions from Underwater Robotic Surveys. *Journal of Field Robotics*, 27(1), 21–51.

Kng, O., Strecha, C., Beyeler, A., Zufferey, J.-C., Floreano, D., Fua, P., Gervaix, F., 2011. The Accuracy of Automatic Photogrammetric Techniques on Ultra-Light UAV Imagery. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-1/C22.

Mazzei, L., Corgnati, L., Marini, S., 2015. Low Cost Stereo System for Imaging and 3D Reconstruction of Underwater Organisms. *IEEE OCEANS Conference*, 1–4.

Menna, F., Nocerino, E., Remondino, F., 2018. Photogrammetric Modelling of Submerged Structures: Influence of Underwater Environment and Lens Ports on Three-Dimensional (3D) Measurements. *Latest Developments in Reality-Based 3D Surveying and Modelling*, MDPI.

Nelson, E. A., Dunn, I. T., Forrester, J., Gambin, T., Clark, C. M., Wood, Z. J., 2014. Surface Reconstruction of Ancient Water Storage Systems – An Approach for Sparse 3D Sonar Scans and Fused Stereo Images. *International Conference on Computer Graphics Theory and Applications*, 161–168.

Nicosevici, T., Garcia, R., 2008. Online Robust 3D Mapping Using Structure from Motion Cues. *IEEE OCEANS Conference*, 1–7.

O'Byrne, M., Ghosh, B., Schoefs, F., Pakrashi, V., 2015. Protocols for Image Processing Based Underwater Inspection of Infrastructure Elements. *International Conference on Damage Assessment of Structures (DAMAS)*, 628, 012130.

Pahwa, R. S., Chan, K. Y., Bai, J., Saputra, V. B., Do, M. N., Foong, S., 2019. Dense 3D Reconstruction for Visual Tunnel Inspection using Unmanned Aerial Vehicle. *International Conference on Intelligent Robots and Systems*.

Pierce, S. G., Burnham, K. C., Zhang, D., McDonald, L., MacLeod, C. N., Dobie, G., Summan, R., D., M., 2018. Quantitative Inspection of Wind Turbine Blades using UAV Deployed Photogrammetry. *European Workshop on Structural Health Monitoring*.

Piras, M., Grasso, N., Jabbar, A. A., 2017. UAV PHOTOGRAMMETRIC SOLUTION USING A RASPBERRY PI CAMERA MODULE AND SMART DEVICES: TEST AND RESULTS. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W6.

Pizarro, O., Eustice, R., Singh, H., 2004. Large Area 3D Reconstructions from Underwater Surveys. *IEEE OCEANS Conference*, 2, 678–687.

Rende, S. F., Irving, A. D., Lagudi, A., Bruno, F., Scalise, S., Cappa, P., Montefalcone, M., Bacci, T., Penna, M., Trabucco, B., Di Mento, R., Cicero, A. M., 2015. Pilot Application of 3D Underwater Imaging Techniques for Mapping Posidonia Oceanica (L.) Delile Meadows. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-5/W5, 177–181.

Rossi, P., Mancini, F., Dubbini, M., Mazzone, F., A., C., 2017. Combining Nadir and Oblique UAV Imagery to Reconstruct Quarry Topography: Methodology and Feasibility Analysis. *European Journal of Remote Sensing*, 50(1).

Santise, M., Thoeni, T., Roncella, R., Sloan, S. W., Giacomini, A., 2017. PRELIMINARY TESTS OF A NEW LOW-COST PHOTOGRAMMETRIC SYSTEM. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W8.

Schmidt, V. E., Rzhanov, Y., 2012. Measurement of Micro-Bathymetry with a GOPRO Underwater Stereo Camera Pair. *IEEE OCEANS Conference*, 1–6.

Sedlazeck, A., Kser, K., Koch, R., 2009. 3D Reconstruction Based on Underwater Video from ROV Kiel 6000 Considering Underwater Imaging Conditions. *IEEE OCEANS Conference*, 1–10.

Servos, J., Smart, M., Waslander, S. L., 2013. Underwater Stereo SLAM with Refraction Correction. *IEEE International Conference on Intelligent Robots and Systems*, 3350–3355.

Shang, Z., Shen, Z., 2016. Real-time 3D Reconstruction on Construction Site using Visual SLAM and UAV. *Construction Research Congress: Construction Information Technology*, 305–315.

Singh, P. S., Sharma, M., Saikhom, V., Chutia, D., Gupta, C., Chouhan, A., Raju, P. L. N., 2018. Towards Generation of Effective 3D Surface Models from UAV Imagery Using Open Source Tools. *Current Science*, 114(2), 314–321.

Skarlatos, D., Demestiha, S., Kiparissi, S., 2012. An Open Method for 3D Modelling and Mapping in Underwater Archaeological Sites. *International Journal of Heritage in the Digital Era*, 1(1), 1–24.

Song, Y., Kser, K., Kwasnitschka, T., Koch, R., 2019. Iterative Refinement for Underwater 3D Reconstruction: Application to Disposed Underwater Munitions in the Baltic Sea. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W10, 181–187.

Steffen, E., Forstner, W., 2008. On Visual Real Time Mapping for Unmanned Aerial Vehicles. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVII, 57–62.

Teixeira, L., Chli, M., 2016. Real-Time Mesh-based Scene Estimation for Aerial Inspection. *International Conference on Intelligent Robots and Systems*.

Venkataraman, K., Lelescu, D., Duparr, J., McMahon, A., Molina, G., Chatterjee, P., Mullis, R., 2013. PiCam: An Ultra-Thin High Performance Monolithic Camera Array. *ACM Transactions on Graphics*, 32number 6.

Vlachos, M., Berger, L., Mathelier, R., Agrafiotis, P., Skarlatos, D., 2019. Software Comparison for Underwater Archaeological Photogrammtric Applications. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W15, 1195–1201.

Zhou, Y., Rupnik, E., Faure, P.-H., Pierrot-Deseilligny, M., 2018. GNSS-Assisted Integrated Sensor Orientation with Sensor Pre-Calibration for Accurate Corridor Mapping. *Sensors*, 18(9).