Stereo Rig Final Report

Yifei Zhang

Abstract

The ability to generate 3D images for the underwater environment offers researchers better insights by enabling them to record scenes for future analysis. The stereo rig uses a pair of stereo cameras with a fixed baseline to generate depth images. Using these depth images, we then generate visual odometry of the stereo rig and stitch these images for the scene. In this quarter, we implemented the synchronization of two cameras on the stereo rig and achieved an overall mean error of 0.24 pixels. We compared block matching (BM) and semi-global block matching (SGBM) algorithm for the generation of depth images. As a result, the SGBM generates significantly better depth images while requires acceptably more computation power. We've also attempted using point cloud filter, which turns out to be too computation-intensive to be possible for the on-board computer. An easy-to-use web interface for the stereo rig cameras parameter adjustment is implemented but requires further testing.

Introduction

The stereo rig is a project for underwater 3d perception. Together with the underwater tablet enclosure project, they forms a solution for the 3d modeling of underwater environment. The stereo rig utilizes stereo vision and computer vision techniques for the modelling process. This 3d model is capable of offering further insight for the marine geologist as well as other researchers. In normal operation, the stereo rig and the tablet should be carried by a diver under water. The stereo rig communicates with the ethernet router on land or ship using a custom-made ethernet cable. The tablet is also wirelessly connected to the router and communicate with the stereo rig wirelessly via the router. Due to the difficulties of staying underwater for a long time, the stereo rig, which enables researchers to record the environment for further analysis when they have more resources on-land, will be a more effective way of recording the underwater scene. The tablet for the stereo rig which can be carried underwater enables them to dynamically adjust the camera parameters (e.g. exposure time, frame rate) to achieve optimal image quality for various lighting environment.

The stereo rig runs Robot Operation System (ROS) on top of the onboard computer running xUbuntu. This set of software controls the fundamental logics of communication over ethernet and integration of different functionalities. This set of software also allows us to directly integrate real-time image processing and simultaneous locating and mapping (SLAM) implementations onto the stereo rig.

The onboard computer is a Core i7-4500U computer. This computer provides reasonable performance and is able to deliver enough online time when powered by a battery. The cameras are the IDS UI-3370CP-C-HQ cameras. They have global shutter for best performance in dynamic environment and external I/O connector for trigger control.

The main difficulties for the stereo rig project is to ensure the imaging quality of the two machine vision cameras. The stereo rig utilizes a stereo image pair from the binocular camera to determine the distance of the object from itself. An algorithm extract features from the stereo image pair and compare the position of each features in the stereo image pair. Then, the algorithm computes the difference between two positions of the same features in the two images and calculate the disparity based on the calibration result of the binocular cameras.The disparities for each features are then compiled into a disparity map and a point cloud. The process is computationally demanding. Therefore, to enable the real-time capability of the stereo rig, we have to select the most cost-effective algorithm for the stereo image processing.

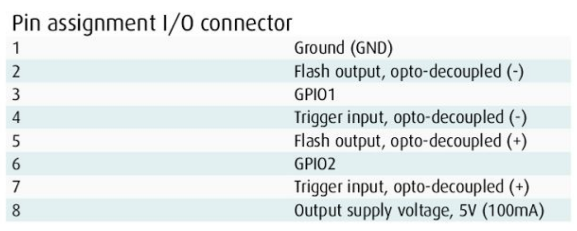
Deliverables of the Stereo Rig:

1. Create an external trigger device for stereo vision cameras
2. Re-calibrate the stereo camera pair for optimal performance
3. Select a stereo matching algorithm that gives disparity map with best quality-computation ratio
4. Set up a web interface for the stereo rig.

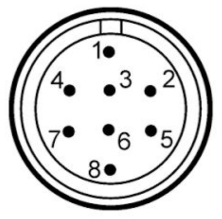
External Trigger

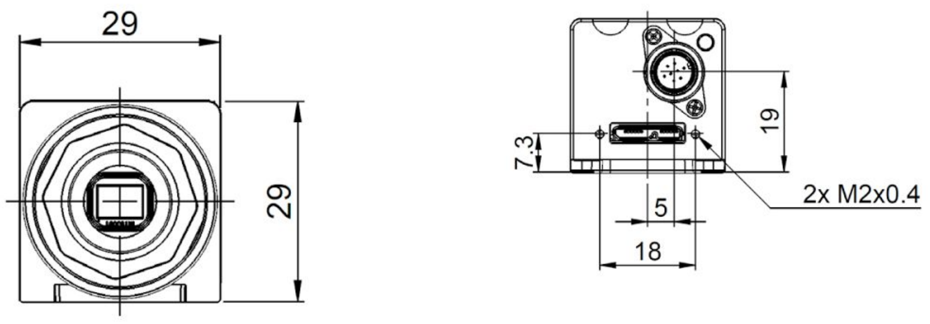
The external trigger is built based on an Arduino Pro Mini. This board is selected because of the easiness of Arduino programming and its minimum design for the limited space inside the stereo rig. This board utilizes a ATMEGA32U4 microcontroller for processing as well as USB-serial interfacing with the computer. To enable the synchronization of two cameras, we have to connect two of the external IO pins on the cameras for the trigger control. To achieve this goal, we built our own cables for the two cameras.

The IDS camera uses the HR25-7TP-8S circular connector manufactured by Hirose Electric. The specification of the circular connector and the definition of each pin is listed as follow:


Pin assignment I/O connector

1	Ground (GND)
2	Flash output, opto-decoupled (-)
3	GPIO1
4	Trigger input, opto-decoupled (-)
5	Flash output, opto-decoupled (+)
6	GPIO2
7	Trigger input, opto-decoupled (+)
8	Output supply voltage, 5V (100mA)





The pin 4 and 7 is used for the camera synchronization. These two pins are connected to the pin 7 and 8 on the external trigger device through the cable we made.

We've also included a GND for each of the two cameras in the camera cable but they are not used during synchronization.

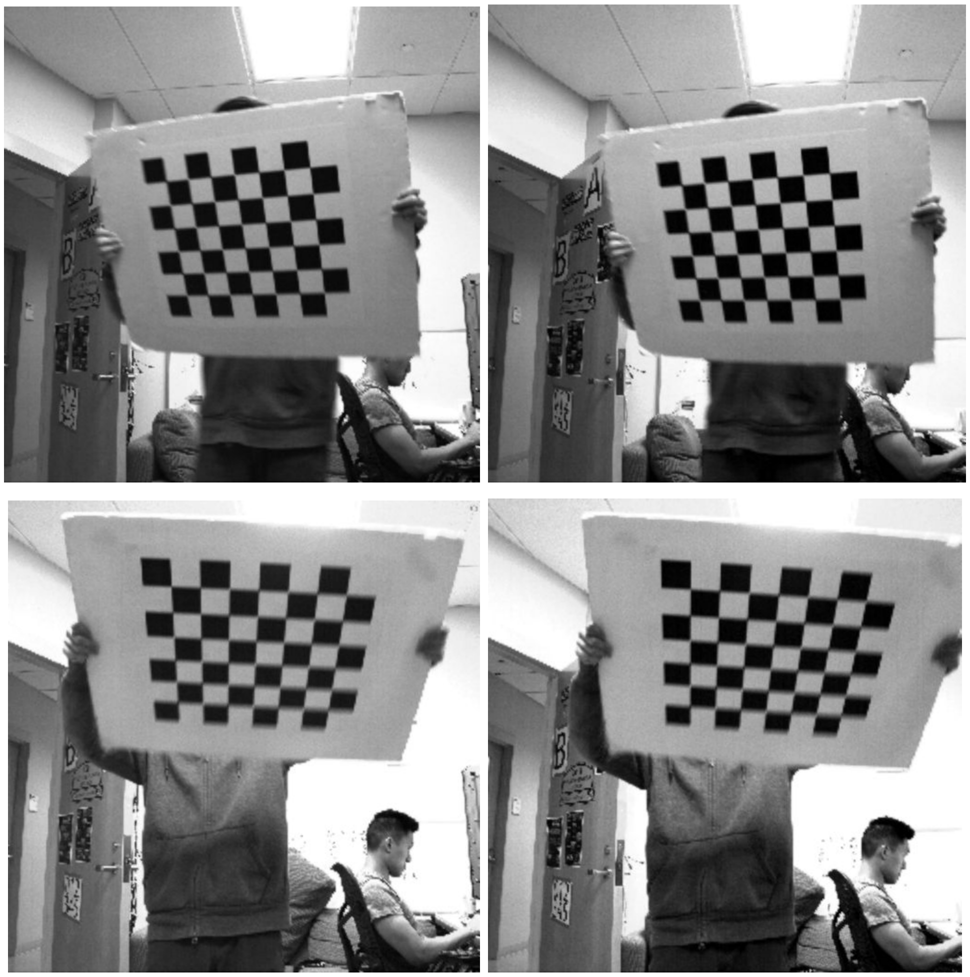
The camera external trigger is a square wave generator. It uses the Timer1 on the microcontroller to generate the square wave to one of the trigger inputs (+) and keep the other trigger input (-) to ground. During the development of this external triggering device, we didn't use the rosserial library for Arduino which implemented all the higher-level interfacing method between Arduino and ROS because we want to achieve minimal design. Therefore, we've implemented a simple plain protocol for serial communication. This protocol allows this device to synchronize with the ROS even if there is an interrupt in the physical connection layer. For the

ROS part, we've also written a node which subscribes to a frame_rate topic and controls this external trigger device. Then, this device is installed into the stereo rig after testing.

Stereo Camera Calibration

Before the cameras are synchronized, we did a checkerboard calibration with the stereo cameras. The result is, however, not good enough. The reason is that the stereo image pairs being used during the calibration process is not taken at the exact same time. There is a tiny difference in the timestamp for left and right camera images. When the calibration algorithm tries to use epipolar geometry to calculate the extrinsics (displacement and rotation for relative position of the two cameras) and intrinsics (lens and projection distortion) for each of the two cameras, there will be a high error per pixel. Thus, even with the calibrated parameters, the unsynchronized camera pairs is not good enough for stereo matching.

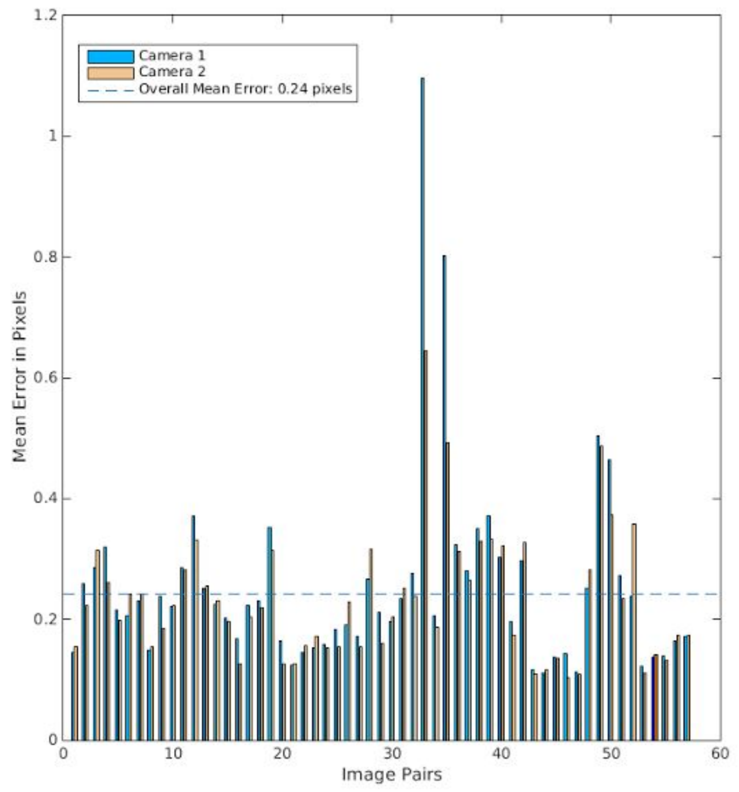
After the camera synchronization is done, we set the frame rate to 25 hz and did another calibration using the utility from the image_pipeline package in ROS.



Two sets of stereo images pairs during calibration

These two sets of images demonstrated the external triggering device and camera synchronization in the calibration process. Although the user is constantly moving in the scene, the two images are still showing the user at the exact same time with minimal delay.

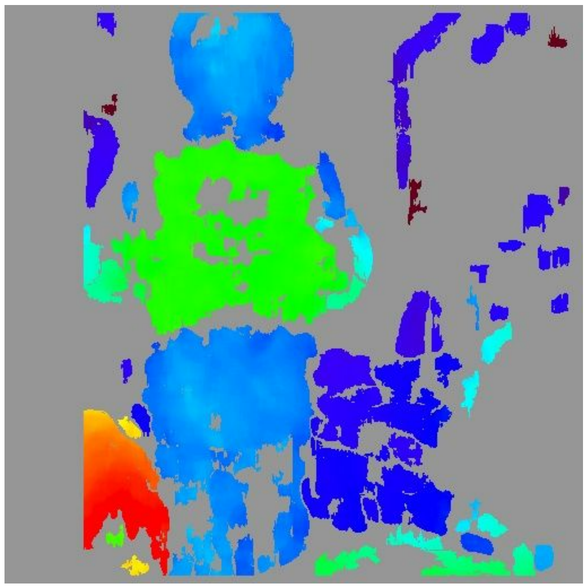
During the calibration process, we have also done a MATLAB analysis of the stereo calibration result.



As the shown in the chart, the mean error per pixel is 0.24 pixels, which is good enough for a stereo vision pair. This result, however, shows that the calibration is not as good as some commercially available stereo vision systems. For example, the Bumblebee stereo vision solution manufactured by Point Grey Research, which has a similar baseline setting as the stereo rig, is capable of achieving an error of 0.05 pixels, ensuring a much better disparity map. Moreover, there are some outliers in the image pairs, this is due to the calibration process used in the stereo calibration utility in ROS. It uses the streaming image sequence instead of still images for the calibration, this sometimes let the motion blurs of the user under low light condition to interfere with the original images. This result suggests that we could take still images using the two cameras and calculate the camera parameters using this still images dataset for calibration, which would give us optimal camera parameters with lower error.

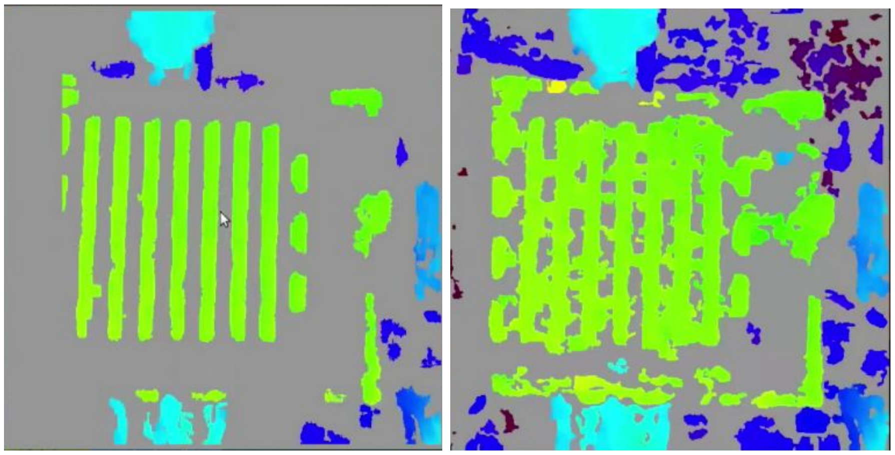
Stereo Matching

The original stereo matching algorithm used in the ROS stereo_image_proc node is the OpenCV block matching algorithm. It is the algorithm that ensures the real-timeness of the system by sacrificing some quality of the disparity map. Ths block matching algorithm has really fast performance and we are able to run the algorithm with image pairs of 512*512 pixels each at more than 50 frames per second in real time using the onboard computer. The main disadvantages of this block matching algorithm, however, is that its result is relatively sparse, meaning that we cannot retrieve a large amount of the 3d information, which will be useful in the modeling process, from the disparity map. Its performance for some low feature environment is also not ideal.



This is an example of the output from the block matching algorithm. The grey area means the algorithm is not able to extract feature from or not able to calculate the distance of object in the image.

Therefore, we looked at other stereo matching algorithms. Similar to block matching, the OpenCV also provided an implementation of the Semi Global Block Matching algorithm. This algorithm utilizes larger area to search for the features in the stereo image, thus allowing a better feature matching result. Here is a comparison between the two.



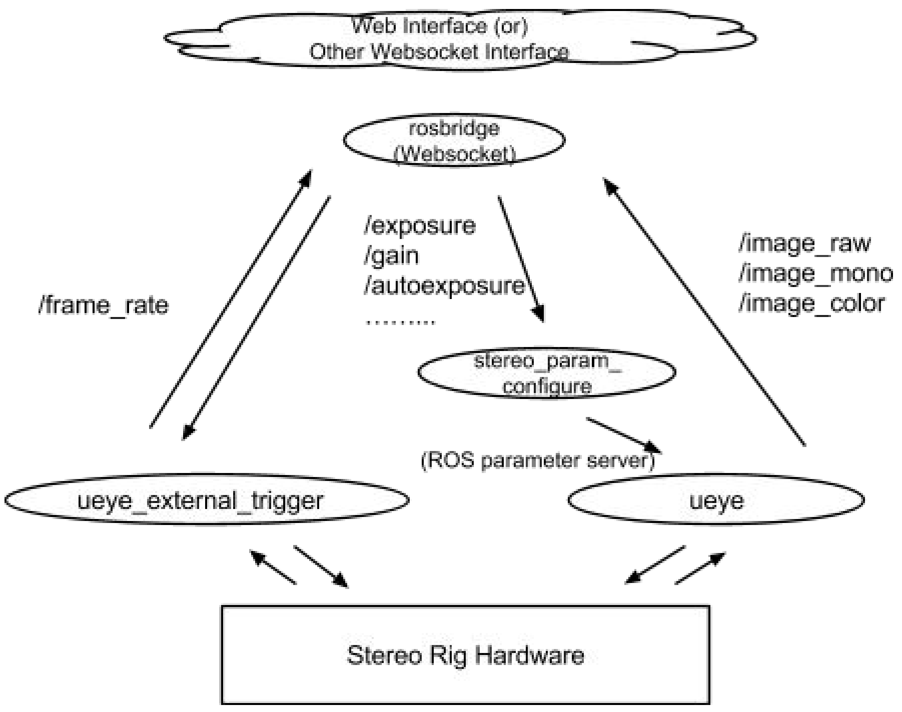
Left: Block Matching Right: Semi Global Block Matching

The comparison clearly demonstrated the effectiveness of the SGBM algorithm. Given the same stereo image pairs, the SGBM method is capable of generating a disparity map with significantly greater density, which is more ideal for the 3d modeling process. Originally, SGBM is implemented in the hydro-devel branch of the image_pipeline. We've been using this version of the image_pipeline package until the SGBM is recently implemented in the indigo branch upstream.

Another stereo matching algorithm we've looked into is the Efficient Large-scale Stereo Matching method (Geiger, A., 2010). This method enables high definition stereo image pairs to be used in the stereo matching process. The onboard IDS cameras can be configured up to 2048*2048 and is well suited for ELAS. However, the streaming of high definition video pairs underwater is hard due to limited bandwidth. The ROS implementation of ELAS is also outdated. Therefore, we didn't test this method with the existing setting of the stereo rig.

Generally, the SGBM algorithm works best given the current condition of the stereo rig. Although it will require much more computation power than that of the block matching method. Its high quality disparity map will enable us to do further modelling. We've also used several point cloud filter from the Point Cloud Library. Specifically, the statistical outlier removal filter works best on the dataset as it can remove shadows and noises (random points floating in the scene). However, this filter requires large amount of computation power, causing an approximately 2 to 3 frame per second point cloud output using the stereo rig onboard computer. This result is definitely not ideal, suggesting we could look into the hardware acceleration solutions for this process.

Web Interface



Stereo Rig Web Interface Hierarchy

This diagram shows the stereo rig web interface hierarchy in details. Web interface directly communicate with the rosbridge node via websocket. The websocket can directly offer streaming of stereo image pairs and interface with the two ROS node implemented by us to control the frame rate and other camera parameters. The stereo_param_configure node will communicate with the ROS parameter server with the ueye driver node and set camera configurations for the user (e.g. enable/disable auto exposure or auto gain). The ueye_external_trigger node will handle all communication with the external triggering device via serial port. The topics on which the these controls are done have been added on the diagram.

Milestones:

<u>Completed Milestones</u>

The following milestones have been accomplished.

- Setup the nodelet for point cloud filtering
  - remove statistical outliers and NaN points
- Evaluate the cost of filtering the point cloud
- Discuss with other E4E members about the web interface for stereo rig
- Learn more about the stereo block matching algorithm which is used in generating the disparity map and try tuning the parameters.
  - Use rosbag to create recordings for further investigation
- Work with other E4E members to implement the web interface
  - Worked with Carl Ngan. He is responsible for developing the html/js part of the web interface. This web interface, however, is not currently deployed and requires further testing.

Not Achieved

The following milestones are not achieved because most of the work this quarter are focused on generating reliable disparity map from stereo camera pairs. Initially, I've setup the roslaunch file for these work, however, the low quality disparity map we've got at the beginning of the quarter is not sufficient for SLAM purpose. After the recalibration of the stereo rig is done, we do not have more time to test these SLAM algorithms again.

- Setup the stereo rig to work with RTABMap and test in the building.
- Keep on working RTABMap or try another SLAM algorithm based on the performance of the former

Completed Additional Milestones

During this quarter, we've encountered a lot of problems when working on the milestones. Therefore, the following additional milestones are added to solve those problems.

- Fix camera mount
- Develop an external triggering device for the stereo rig
- Change the stereo matching algorithm to SGBM

Conclusion

In summary, most of this quarter has been spent on the stereo vision part of the project. The project is still not fully deliverable for researchers to use at the end of the quarter. However, we've finished the fundamental work on the stereo vision system for future work to continue on. Personally, the project helped me gain lots of knowledge in the computer vision related topics. I have learnt the importance of calibration and synchronization in the stereo vision system and the underlying principles behind them. I've also got the opportunity to compare several existing stereo matching algorithms for the quality of the disparity map.

In this quarter, we've implemented the synchronization for the stereo vision cameras and installed it in the stereo rig. The external trigger device allows us to achieve a relatively low calibration error after a checkerboard is done. A disparity map with satisfactory quality is also generated using the semi global block matching approach in real time. Furthermore, the stereo rig is fully prepared for interfacing with an web application or other websocket-based user interface.

Despite all the achievement we've made on the stereo rig project, there are still much future work that could be done. The first is to change the stereo matching and filtering process into a hardware accelerated approach, which will allow the onboard computer to have more idle CPU time for the real time 3d reconstruction. There has been many existing works on the FPGA-based stereo vision system. Since we could install an FPGA board on the stereo rig using the PCI port on the onboard computer, this will greatly reduce the computation pressure for the onboard computer. The second future work we could focus on is to integrate the SLAM software into the system to enable the real-time 3d modeling of the environment. To guarantee the real-timeness of the SLAM process, a hardware based stereo vision system may be required to free up the onboard CPU. The last possible work before the stereo rig can went into use is to improve the stereo vision calibration process to minimize the error per pixel.