Project Journal

Machine Vision-Based Underwater Anti-Backscatter Lighting System.

Sidharth Shanmugam.

Table of Contents

[W/B 30-10-2023 3](#_Toc150881214)

[Initial supervision meeting & high-level objectives 3](#_Toc150881215)

[W/B 06-11-2023 4](#_Toc150881216)

[Intro to real-time software [1] 4](#_Toc150881217)

[Supervision meeting 4](#_Toc150881218)

[W/B 13-11-2023 6](#_Toc150881219)

[Porting to C codebase for performance benefits. 6](#_Toc150881220)

[Digital image processing literature 6](#_Toc150881221)

[Bibliography 7](#_Toc150881222)

# W/B 30-10-2023

## Initial supervision meeting & high-level objectives

* Meeting with Paul Mitchell and Benjamin Henson on Tuesday.
* Discussed background for original project idea and overall goals to achieve.
* Discussed previous work carried out by Katie, limitations, and scope for future work.
  + Improving real-time computing.
    - Experimentation with more powerful processing hardware.
    - Research on hypervisors, threading, and multiprocessing.
    - Using IR beams to detect backscatter, potentially eliminating need for high/low beam flash loops.
    - Streaming video from Pi to a more powerful computer to compute backscatter locations, transmitting back to Pi to drive projector.
      * Potential latency issues.
    - Algorithm, Python/OpenCV optimisations.
  + Underwater testing.
    - Takes a long time, can be achieved with bubbles to replicate backscatter in controlled environment such as a tank.
    - *Probably best to carry out at the end of the project?*
  + Backscatter depth perception.
  + ML-based backscatter position tracking.
* Pursuing one point from the above four is complex enough for the project.
* Next meeting scheduled for 2pm on Thursday 9th.
  + In-person at the ISA.
  + Paul to send calendar invite.
  + Tour of the ISA, and desk allocation for me to work at.
* Actionable points:
  + For each of the four future work objectives, I’ll be carrying out preliminary research to understand what technologies, technicalities and techniques are required.
  + At the next supervision meeting at the ISA, I will be introduced to the end-result of Katie’s work. Using this I can better gauge what point to pursue.

# W/B 06-11-2023

## Intro to real-time software [1]

Real time is a type of application that requires a guaranteed response within a strict timing constraint. There are two types of real-time and two metrics:

**Hard real-time:** If the event is not processed in a strict timing window, then bad things will happen.

**Soft real-time**: If the event is not processed in a not as strict timing window, then the system may degrade but it’s not as bad.

**Interrupt latency:** Time to process an interrupt.

**Scheduling latency:** Time for the OS Scheduler to start a processing task (Nominal and Worst-case).

RT can be handled by Raspberry Pi with a lot of tweaking:

* Current Linux / BSD kernels for the Raspberry Pi do not support real-time [as of 2015 when this article was made].
* Standard Linux or BSD install includes services that generate lots of overhead.
* Pi doesn’t have a real-time clock [But I think the Pi 5 comes with one now].
  + Expects device to be always connected to the internet.
  + Cannot generate deterministic timing pulses to control things like DC motors

Real Time Executive for Multiprocessor Systems (RTEMS) for Raspberry Pi:

* Open-source Real Time Operating System (RTOS).
* Used in Spaceflight, medical, and other real-time embedded applications.
* Ported to the Raspberry Pi *(w/ some limitations).*

## Supervision meeting

* Meeting with Paul Mitchell and Benjamin Henson on Thursday at the ISA.
* Discussed real-time research findings.
* Tour of the ISA, and desk assignment.
* Scheduled bi-weekly in-person supervision meetings.
* Discussion about hypervisors with Ben
  + Using something like Docker, script can be packaged and assigned to a CPU core.
  + This can solve scheduling issues that was experienced when running directly on OS.
* Shown interest in the ‘Improving real-time computing’ aspects of this project.
  + Will be focusing on this, with aspects of underwater testing. If there is time at the end of the project, we can use the underwater-testing tank facility (if construction is completed by then).
* Rewrite project code in C.
  + Eliminate the overhead that Python introduces.
* More efficient logic.
  + Paul and Ben both mentioned to start simple – start with a simple thresholding and add fine tune.
  + Read older literature (e.g. 2010-era) on digital image processing – this should be a good starting point to build up from since there won’t be any added complexities.
* Compare algorithms for detecting and eliminating backscatter.
* Ben will send me some underwater recordings (GoPro footage), I can use this to program the code without needing to perform any underwater testing.
  + Since the existing code isn’t readily available (backed up to Git), I have asked Ben to take a backup of the RPi’s SD card (the one that’s currently in the lighting system, with the code that Katie produced).
    - Code has been printed in Katie’s report, however, Driver.py has been cut-off due to formatting issues (code starts from line 60 and previous lines are missing).

# W/B 13-11-2023

## Porting to C codebase for performance benefits.

Python is slower than C/C++ due to the fact it’s an interpreted language, in contrast to the compiled language. The interpreter will use more computational resources such as CPU and RAM.

OpenCV and NumPy are both written in C/C++, with the respective Python packages simply acting as a wrapper to interface between Python and C/C++. This means that Python is actually quite fast as long as the standard OpenCV and NumPy functions are used in tandem. The performance bottlenecks are only experienced when own Python functions are written.

For example, if they contain for loops or iteration such as the following :

|  |
| --- |
| 1. for y in range (image.shape|0]): 2. for x in range (image.shape|1]): 3. result[y,x] = doSomethingWithPixel(image[y,x]) |

Source: [2]

It would be easier to prototype in Python, then once working, translate into C++ for production code. Look into event-driven or asynchronous programming paradigms, using this, a while loop to continuously poll for backscatter can be eliminated.

## Digital image processing literature

* [**Fundamentals of digital image processing / Anil K. Jain.**](https://yorsearch.york.ac.uk/primo-explore/fulldisplay?docid=44YORK_ALMA_DS21216898830001381&context=L&vid=44YORK-NUI&lang=en_US&tab=default_tab&query=any%2Ccontains%2Cdigital%20image%20processing%2CAND&query=any%2Ccontains%2Calgorithm%2CAND&sortby=rank&facet=searchcreationdate%2Cinclude%2C1981%7C%2C%7C2013&mode=advanced&pfilter=pfilter%2Cexact%2Cbooks%2CAND&offset=0) [1989]
* [**Digital image processing / Kenneth R. Castleman.**](https://yorsearch.york.ac.uk/primo-explore/fulldisplay?docid=44YORK_ALMA_DS21228636060001381&context=L&vid=44YORK-NUI&lang=en_US&tab=default_tab&query=any%2Ccontains%2Cdigital%20image%20processing%2CAND&sortby=date&facet=frbrgroupid%2Cinclude%2C22140808&mode=advanced&offset=0) [1996]
* [**Principles of Digital Image Processing**](https://yorsearch.york.ac.uk/primo-explore/fulldisplay?docid=TN_cdi_skillsoft_books24x7_bks00076998&context=PC&vid=44YORK-NUI&lang=en_US&tab=default_tab&query=any%2Ccontains%2Cdigital%20image%20processing%2CAND&query=any%2Ccontains%2Calgorithm%2CAND&sortby=rank&facet=searchcreationdate%2Cinclude%2C1981%7C%2C%7C2013&mode=advanced&pfilter=pfilter%2Cexact%2Cbooks%2CAND&offset=0) (part of 3 book series)
  + Fundamental techniques [2009]
  + Core algorithms [2009]
  + Advanced methods [2013]

## Quick discussion with Ben

* Mentioned the performance aspects of Python.
* Objectives for work discussed:
  + Detect backscatter/
    - Use simple algorithms first.
    - Histograms.
  + Predict next backscatter position.
    - Use previous frames and track centre point of backscatter.
    - Use a linear prediction to predict position at the next frame.
      * Start with linear, moving on to more adv. Predictions later.
* Timing would be difficult.
  + Flash high brightness light to pick up backscatter.
  + Project holes.
  + Predict next hole location.
* Potential future work could be stitching images together to get a ‘panoramic’ large image of seabed with backscatter eliminated.

# Bibliography

|  |  |
| --- | --- |
| [1] | S. Doran, “How to Perform Real-Time Processing on the Raspberry Pi - SCALE 13X,” 19 June 2015. [Online]. Available: https://www.socallinuxexpo.org/sites/default/files/presentations/Steven\_Doran\_SCALE\_13x.pdf. [Accessed 6 11 2023]. |
| [2] | kbarni, “OpenCV Forum,” 25 June 2021. [Online]. Available: https://forum.opencv.org/t/python-vs-c-personal-experiences-with-using-both/4016. [Accessed 14 November 2023]. |