Project Updates: 19-04-2024

Sidharth Shanmugam

April 19, 2024

Introduction

- **Supervision Meetings:**
  Consists of a listing in table format of the supervision meetings that have occurred since the last update, including dates, attendees, and a brief description of discussions and actionable items.
- **Actionable Items Recap:**
  Consists of a listing in table format of the actionable items from the previous week, briefly discussing the progress made and pending tasks.
- **Additional Project Updates:**
  Consists of updates that weren't 'actionable items' from the previous week, such as brief overviews of experiments conducted, data collected, and research findings.
- **Next Week's Agenda:**
  Consists of a listing in table format of the actionable items to complete before the next weekly update, including task descriptions, rough timelines, and success metrics.
- **Comments & Concerns:**
  Consists of a brief analysis of comments or observations about other aspects of the project, such as facilities, work environment, and any outside interest in the project. Furthermore, outlines any concerns about the project.

1 19-04-2024

1.1 Supervision Meetings

1.1.1 09-04-2024

We didn't have a specific meeting on Tuesday's as we usually do this week due to the ISA's morning closure. However, since I work from the ISA daily, we were able to meet up informally at multiple points throughout the week to discuss progress.

Agenda:

- Multiprocessing was a massive failure!
  - V1 software is complete!
  - I realised that the current version is completely sequential, which is very inefficient since on a multicore system, such as the 4-core Raspberry Pi 4 and 5, only one core will be utilised. Python is not multithread compatible out of the box due to the Global Interpreter Lock (GIL), which is a mechanism to synchronise threads. However, the 'multiprocessing' package allows for the creation of processes which are functions that is run bypassing the GIL.
  - I have re-written the software to implement each stage (i.e., greyscale conversion, histogram equalisation, etc) of the image processing pipeline as a process which runs on different cores. Since thread safety is crucial when moving data between different threads, queues must be utilised when sending data.
  - The performance data that I measured with the multiprocessing version was very surprising when compared to the single-core version: processing each frame took an average of 2x longer with multiprocessing, I went from an average frame processing time of 17ms to 33ms!
  - I think the biggest reason for this slow-down is due to the queues being an inter-process communication (IPC) service, there is a lot of overhead involved with IPCs, including serialisation, deserialisation and context switching between threads.
  - Another crucial problem with multiprocessing is the induced frame delay: capturing is a fast process that feeds into the much slower processing stage - so for example, when the 25th frame is being processed, the 100th frame is being captured, resulting in a 75-frame delay.
  - Unfortunately, due to the much reduced performance I will not be researching multiprocessing any further. Although a lot of time had been spent on this (3.5 days) without any meaningful performance benefit when compared to the single core program, I have learnt a lot:
    * Multiprocessing processes have much more overhead to startup as an entirely new OS process is being created as each one is run.
    * IPC services have a lot of overhead, as I mentioned before. Also, there is a lot of I/O bound tasks which can severely bottleneck multiprocessing.
    * Pipelining and multiprocessing only works for an FPGA, where each pipeline stage is hardcoded in fabric and isn't software threads that is being scheduled to each CPU-core by an OS.
    * (Ben mentioned this:) Never second-guess the compiler/OS when it comes to optimisations as it can optimise the code much better for runtime.

  - Although this was a failure, I can still write about it in my final report as I have a set of meaningful data and graphs.

- ISA tank is ready for use.
  - The tank is ready for use, I will be using it on Monday.
  - While we were discussing this, Ben and I tested the prototype setup to ensure the projector and camera are both properly focused and whether the script/software all works.

Actionable Items:

Install PREEMPT-RT kernel and quantify:

- Since I now have the V1 software in place, I can build the real-time kernel and test with it.

1.2 Actionable Items Recap

1.2.1 Reach milestone V1

Progress Report:

- *Done!*

Pending Tasks:

- *None.*

1.2.2 Implement PREEMPT-RT kernel and quantify

Progress Report:

- Since I got very much side-tracked with the multiprocessing research, I couldn't spend much time on this.
- However, I have managed to figure out how to do it - I'll need a Linux host system to build the kernel, I can do this on the Pi itself without a host system, however, it'll take forever to compile.
- I have installed and set up a Ubuntu OS VM again, I had to uninstall the VM from earlier in the project due to internal drive space constraints, however, that shouldn't be an issue anymore.

Pending Tasks:

- Build and compile the kernel.
- Copy in new kernel and quantify.

1.2.3 Snake-based segmentation & tracking

Progress Report:

- *No progress due to mulitprocessing side-tracking.*

Pending Tasks:

- The V1 system uses Canny to detect edges, with the detected edges passed to OpenCV's findContours() method to extract closed loop edges. The cancellation logic then uses a minimum enclosing circle (MEC) implementation which calculates a circumcircle which completely covers the minimum area of the detected contour. The MEC implementation must be replaced with the snake method.
- The 'least distance rule' and Kalman filter approach must be researched for the backscatter tracking implementation in the system.
- I will be putting the parallax part of 'Project holes and mitigate parallax' task on pause until the final housing is completed and the ISA tank is functional.

1.3 Additional Project Updates

None at the moment.

1.4 Next Week's Agenda

1.4.1 Monday - Record underwater footage

Actionable Items:

- I've got the code in-place for this, just need to record.

Success Metrics:

- Downloading a video file of the recording to use as test footage.

1.4.2 Monday - Implement PREEMPT-RT kernel and quantify

Actionable Items:

- Build and compile the kernel.
- Copy in new kernel and quantify.

Success Metrics:

- Analysis logged in the project journal

1.4.3 Friday - Snake-based segmentation & tracking (Milestone V2)

Actionable Items:

- The V1 system uses Canny to detect edges, with the detected edges passed to OpenCV's findContours() method to extract closed loop edges. The cancellation logic then uses a minimum enclosing circle (MEC) implementation which calculates a circumcircle which completely covers the minimum area of the detected contour. The MEC implementation must be replaced with the snake method.
- The 'least distance rule' and Kalman filter approach must be researched for the backscatter tracking implementation in the system.
- I will be putting the parallax part of 'Project holes and mitigate parallax' task on pause until the final housing is completed and the ISA tank is functional.

Success Metrics:

- Code pushed to the git repository.

1.5 Comments & Concerns

No comments or concerns at the moment.