WormSeg User Guide

Kyle Moy

February 3, 2016

1 Introduction

This iteration of the worm segmentation software was rewritten from scratch and includes improvements to usability, scalability, and performance.

The software comprises a single Java package that can be configured via command line arguments. This makes it possible to launch and automate the software via scripts, and will require no human input once launched. The software will manage its own processes, thus only one instance needs to be running on any individual machine or VM.

This iteration is built on a different data-scattering model that, at the cost of memory, completely eliminates disk I/O on nodes. This inherently places a hard limitation on the software, requiring the nodes to be able to fit the its entire workload in memory[1]. Disk I/O was responsible for ˜40% of the total runtime, and its elimination reflects significant improvement. A secondary characteristic of this model is that many more nodes can be effectively utilized, increasing the scalability of this algorithm. Under the previous model, there is a point where transferring data to more nodes would cost more than the computation time (although this is true of all parallel algorithms).

Finally, since I was rewriting from scratch, I had the opportunity to make major changes to the algorithm as well as include micro-optimizations from the beginning. With the exception of substituting box blur for Gaussian blur, no image processing techniques have been changed. Optimizations mostly involve the way data is represented internally and throwing away a lot of unused data (ARGB data reduced to only the blue channel, chosen arbitrarily). Micro-optimizations turned out to be significant, for example pre-calculating division tables, looping arrays in memory-sequential order, memoizing certain lookups, etc. which shaved off several milliseconds per frame. Although I don't have many trial cases, my tests show about 70% reduction in time-per-frame per thread compared to the previous software.

[1]Our largest input data to date, egl19_f2, occupies 5.30GB, which fits in MEDIXSRV memory. Unlike the host, nodes only hold their fraction of the whole payload.

2 Instructions

1. On any number of machines, launch one instance of WormSeg in **worker** mode.
2. On the specified host machine, launch instances of WormSeg in **tasker** mode.
3. Taskers (hosts) will immediately begin tasking nodes. Once processing is complete, it will terminate.
4. Nodes will not terminate. This allows launching multiple taskers with different parameters (not concurrently) to process multiple videos.
5. To terminate all nodes, launch WormSeg in **terminate** mode.

3 Launch Arguments

Bracketed arguments have default values and thus are optional. Strings and numbers are entered literally and should not be enclosed in quotes.

3.1 Tasker Arguments

3.1.1 Operational Arguments

-mode (tasker / worker / terminate)
This must be specified with "**tasker**" to start the program in tasker (host) mode.

-project (name)
The name of the project to process. This argument, along with "-dir", will be used to detect the input and output paths. This argument **must** be specified if "-input" and "-output" are not.

[**-dir (path)**]
The path[2] to the directory where project folders can be found. **Default: "//medixsrv/Nematodes/data/"**

[**-input (path)**]
The path[2] to the directory where the input images can be found. If this argument is specified, "-project" and "-dir" need not be. **Default: Generated from -dir + -project + "input/"**

[**-output (file path)**]
The file path[2] where the output will be written. If this argument is specified, "-project" and "-dir" need not be. **Default: Generated from -dir + -project + "log/feature.log"**

[2]Paths must have a trailing "/".

[-port (number)]
The port to listen on for node connections. **Default: 8190**

[-threads (number)]
The number of threads to use for network traffic parallelization. **Default: Number of CPU cores**

[-nodes (number)]
The **minimum** number of workers(nodes) that must be joined before beginning the task. The task will automatically begin once this many nodes have connected. Any available node will be added, even if in excess of this number. **Default: 1**

[-extension (string)]
The extension on the image files. **Default: ".jpg"**

[-padding (number)]
The number of digits in the file name. i.e. "0000001.jpg" has a padding of 7. Legacy video data may have a padding of 6 because I'm bad at conventions. **Default: 7**

[-frames (number)]
The number of frames to process. **Default: File count automatically detected using given parameters. Could be wrong.**

3.1.2 Segmentation Parameters

These are all optional, however may need to be configured if segmentation is not successful. Most likely to need tuning is "-threshold_ratio".

[-width (number)]
The horizontal resolution of the image. **Default: 640**

[-height (number)]
The vertical resolution of the image. **Default: 480**

[-area_min (number)]
The lower bound for a candidate worm component. If a blob has an area lower than this number, it is discarded and not considered. (e.g. Noise.) **Default: 200**

[-area_max (number)]
The upper bound for a candidate worm component. If a blob has an area greater than this number, it is discarded and not considered. (e.g. The plate edge.) **Default: 400**

[-search_win_size (number)]
The width and height of the crop area. The algorithm crops the frame to this size, centered on the last known worm location, for performance purposes. **Default: 100**

[-blur_radius (number)]
The width and height of the sliding window used in the box blur. **Default: 3**

[-threshold_win_size (number)]
The width and height of the sliding window used in the dynamic threshold. This is the neighborhood against which a pixel's intensity will be compared. **Default: 25**

[-threshold_ratio (float)]
The percentage (as a decimal) which a pixel's intensity as a quotient of its neighborhood average must be less than to be considered to be possibly on the worm body. e.g. If this value is 0.9, and the neighborhood average intensity is 255 (pure white), then a pixel's intensity must be below $0.9 * 255 \approx 229$ in order to be a potential worm body pixel. **Default: 0.9**

3.2 Worker Arguments

-mode (tasker / worker / terminate)
This must be specified with "**worker**" to start the program in worker mode.

[-host (address)]
The address of the machine running (or will run) the tasker (host) process. **Default: "localhost"**

[-port (number)]
The port to connect to. **Default: 8190**

[-threads (number)]
The number of threads to use for computing. **Default: Number of CPU cores**

4 Example Scripts

4.1 Worker Scripts

If you will be tasking from the same machine as the worker, you don't need any additional arguments after specifying worker mode. When not specified, "-host localhost" is implicit. If you will be launching more workers on other machines, it is simpler to explicitly specify the tasker(host) machine IP.

Listing 1: Local Machine Worker

```
java -jar WormSeg.jar -mode worker
```

Most likely, you will need to specify where your tasker process is running. This will almost always be the machine where the data is stored.

Listing 2: Host specified

```
java -jar WormSeg.jar -mode worker -host medixsrv.cstcis.
    cti.depaul.edu
```

If you would like to use a specific number of threads, you can do so with the "-threads" argument. By default, the program will autodetect the number of logical cores available, and usually this is the most efficient number.

Listing 3: Threads specified

```
java -jar WormSeg.jar -mode worker -threads 8
```

4.2 Tasker Scripts

The fewest arguments for taskers is to specify the mode and the project. The default values for data location and output location is based on the assumption that this will be run from MEDIXSRV.

Listing 4: Project Names

```
java -jar WormSeg.jar -mode tasker -project N2_f1
```

If the tasker is not being run from MEDIXSRV, but the data directory structure is similar, you can specify where the root of the project directory is with "-dir".

Listing 5: Project + Directory

```
java -jar WormSeg.jar -mode tasker -project N2_f1 -dir
    D:/nematodes/data/
```

If the data is not stored somewhere with the expected directory structure, you can explicitly specify the location of your video image frames with "-input", and the file path to write the output to with "-output".

Listing 6: Arbitrary Locations

```
java -jar WormSeg.jar -mode tasker -input D:/data/
    -output D:/output.txt
```

You can also tune the segmentation parameters. I won't list them all; refer to arguments section. These are all implicit with default parameters.

Listing 7: Segmentation Parameters

```
java -jar WormSeg.jar -mode tasker -width 640 -height 480
    -area_min 200 -area_max 400 -threshold_ratio 0.9
```

Finally, more often then not, you will need to process many videos at a time. In batch scripts, this can simply be done with multiple launches in tasker mode. All of these will launch independently, and need to have their parameters set each time. You might even want to (in the case that it is necessary) set different segmentation parameters per video.

Listing 8: Mulitple Launch

```
java -jar WormSeg.jar -mode tasker -project N2_f1
java -jar WormSeg.jar -mode tasker -project tph1_f2
java -jar WormSeg.jar -mode tasker -project egl19_f3
```

4.3 Java VM Considerations

The JVM heap size is by default 1/4th of your total physical memory. Depending on the input data, a larger heap size may be necessary. You can specify the heap size with the flag **-Xmx#g**, where # is some number of gibibytes.

Listing 9: Local Machine Worker

```
java -Xmx16g -jar WormSeg.jar -mode [tasker|worker]
```