# Mario Kart 64 Analytics Suite

Engineering Addendum

To those continuing work on the Mario Kart 64 Analytics Suite, there are some things that need to be touched on.

# Video Processing

One problem that we have not been able to bypass is the case of item detection when Boo steals Boo. The reason is there is no blank box between the stealing of items. Furthermore, the engine will see the stolen Boo, and think that it is the initial Boo received.

The main bottleneck of the video processing suite is the image comparison function, which relies of OpenCV’s templateMatch function applying a normalized sum of squared differences. As this compares on a pixel-by-pixel basis, its running time is proportional to the size of the mask and image being compared. Further improvements would amend this cost.

One way that we have optimized to account for the templateMatch bottleneck is to constrain the region of interest (ROI) for each detector. Therefore, instead of scanning the whole frame, pixel by pixel, we only scan a portion of the frame. This heuristic is based on the fact that most events occur within a specific area in the frame. For example, item boxes only appear in the top third of the frame. Further optimizations could be applied by constraining ROIs to an even tighter region.

Another possibility for optimizing would be to change the type of template matching used. Currently, we utilize the normalized squared difference matching method. While this is the best matching method, it is also the slowest. However, with a bit of fine tuning of threshold values, it could be possible to use a different matching method. While the increase in performance would not be very drastic, there would definitely be some increase.

Furthermore, since the application is so specific, it is possible that writing a custom template matching function could outperform the OpenCV templateMatch function. This would be a bigger feat since the templateMatch function is already multithreaded on the CPU and GPU level.

The last optimization that we have played around with is altering the shared memory buffer size. We currently buffer 400 frames at a time, but increasing the buffer length could increase performance. This is because a larger buffer size would mean less calls to shared memory, which means less locks that need to be applied. While this be an improvement on the current scheme, the templateMatch bottleneck outweighs the buffer size bottleneck so much that any improvement would most likely be negligible.

# Storage API

There’s a lot of work that can still be done to make the query language better. For one, it would be nice if people other than its author were able to use it to write queries. The API itself could also use some significant cleaning. I’d like to see more validation done of what data it receives as well as things like nicer error messages, URL links in responses to other objects, etc. General RESTy stuff.

# Infrastructure

We kind of used a lot of AWS services in an ad-hoc fashion. It’d be cool if this could be cleaned up and managed consistently. Especially of interest would be creating an “install” script around the whole project that sets up the required DBs, queues, applications, buckets, etc. This would make it much easier to migrate between different AWS accounts. Another fun thing to have would be configurations tracked in something like chef, puppet, or ansible. Right now the big “configured” VM we use is an image of a basic VM that we put together by manually compiling and installing everything on. It’d be great if this could be automated either through packaging the programs we need or lots of scripting.

# Auto Commentary

A great feature we came up with was “auto-commentary”. Given a race’s worth of events, it would be cool to generate an audio file that describes what the events were but in a dramatic way. You could then either play this back on top of the video (in which case you need to make sure they sync up) or on its own like a radio broadcast of a race or something. At the end of the day it doesn’t matter, but it’d be hilarious.

# Event Watcher

An additional feature we wanted to include was an event watcher. After the user requests a specific video, the database would also return the watch view a list of event:timestamp key pairs. Underneath the list of videos, up would pop another list of buttons, which would call the video being served at a timestamp corresponding to the event in the title of the button.

# Hardware

Future work on this project should pursue the development of a printed circuit board and suitable enclosure that does not impede or introduce any adverse auditory effects for the microphone module designed by our team.

# Audio Processing

An additional feature we wanted to include was a detector for each character’s voice in the gameplay audio. These cues could be used in conjunction with video processing to create hybrid detectors for more complicated events that would not be able to be detected based solely on one or the other.