Bryce George & Colin Henson

CS 4800-001

December 10, 2020

Graphically Visualizing a Complex System with the Integration of Deep Learning Object Detection and Tracking

**Abstract**

The objective of this research is to develop a method for enhancing human visual perception of complex systems and interactions. Specifically, for our Computer Graphics class term project, we are analyzing a video recording of the “Variation” exhibit by Céleste Boursier-Mougenot in Brazil’s Pinacoteca de Sao Paulo museum. The exhibit consists of three pools containing various-sized, floating ceramic bowls which are set in motion by the pools’ jets. The motion and collisions of the bowls produce complex visual and harmonic patterns. Our research has focused on combining deep learning object detection and tracking techniques to follow the paths of the bowls over time and then apply colored graphical overlays which visualize their motion. The number of bowls, their homogeneity within the system, and the changing camera perspectives throughout the video make this task difficult. Our approach is to apply object re-detection phases which feed into intermediate object tracking sequences. Re-detection phases offer the opportunity to detect bowls which enter and remove bowls which have exited the scene over the course of a tracking sequence. The integration of detection data into the tracking algorithm functions by comparing objects between the final frame of a tracking sequence and those found by the detection algorithm applied to the same frame.

**Project Summary**

This research project has centered around developing a unique approach for tracking bowls in a video recording of the complex system of the “Variation” exhibit so that graphical visualization aids might be applied. In order to do so, we needed to achieve continuous and consistent tracking of the bowls between consecutive frames, even with changing camera angle, zoom, and perspective.

Our approach involves starting with an initial detection of objects in the first video frame, yielding bounding boxes which are used to initialize the first tracking sequence. From there, a repeated cycle of object tracking and re-detection is performed for all subsequent frames of the video. Object detection is accomplished using a YOLOv3 model which has been trained on a custom data set—the data set included approximately 300 annotated frames of the target video. To accomplish object tracking, we use a CSRT object tracking implementation provided by the OpenCV library. Each object is assigned a CSRT tracker, and then the trackers for all objects are aggregated into a single multi-tracker for composite processing during the remainder of the tracking sequence. During a tracking sequence, the multi-tracker is updated with each subsequent frame to provide the updated bounding boxes for each object. Then at the end of a tracking sequence, the final frame of the sequence is also run through the object detection algorithm, yielding a set of detected objects. The objects in the tracked set and the detected set must then be analyzed to re-identify objects which are the same, as well as add or remove any objects which have entered or left the scene during the course of the previous tracking sequence.

The most challenging aspect is implementing the means for associating objects between the tracked set and detected set at the end of a tracking sequence in order to re-identify existing objects and appropriately deal with new objects or objects which have left the field of view. As of the final demonstration for CS 4800, we have implemented a relatively naïve method for performing this association, following this general process:

1. Run tracking on current\_frame
2. Run detection on current\_frame
3. Calculate the bounding box centroids for all elements of the tracked objects set
4. Calculate the bounding box centroids for all elements of the detected objects set
5. Calculate the nearest neighbors for each element in the detected set from the tracked set
6. Calculate a comparison\_threshold
7. For each object in the detected set:
   1. If distance\_to\_nearest\_neigbor < comparison\_threshold, associate the nearest neighbor (tracked) object ID with the current object
   2. Else, assign the object a new ID

However, the current implementation suffers from an unacceptable level of inaccuracy. As such, we developed a new algorithm for performing this process, but unfortunately, have run out of time to implement it. The primary improvement in the new algorithm is the addition of a step to sort the nearest neighbors list according to a global comparison of each detected object with its nearest neighbor before proceeding with the association of objects between the sets.

The final stage of the project, which we wished but were unable to accomplish due to time constraints, was the implementation of a visualization of each bowl’s path using colored dots placed at each bowl’s previous *n*-tracked positions. That said, implementation of the new merging algorithm and the desired graphical elements present a great opportunity for future research and development.

**Links**

Project Repository: <https://github.com/ancient-sentinel/cs-4800-graphics-project/tree/master>

“Variation” Exhibit Video: <https://www.youtube.com/watch?v=mpwBbm22_y0>

Single-Pool Redetection Tracker Prototype Output: <https://youtu.be/KqGCfRG1HTs>

Three-Pool Redetection Tracker Prototype Output: <https://youtu.be/RPmaKY6PFGg>