Brunel University London

Supervisor: Dr. Paul Kyberd
Student: Michael Watzko
Student ID: 1841795

IT DESIGNERS GRUPPE

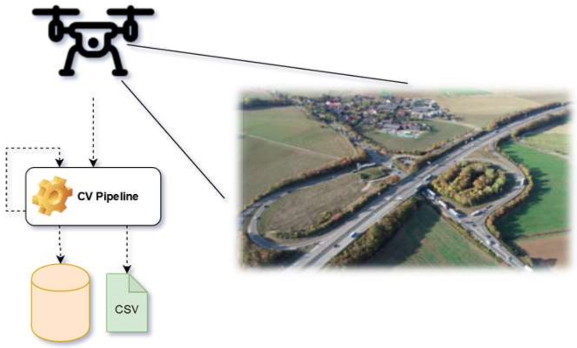
Conception and realization of a distributed and automated computer vision pipeline

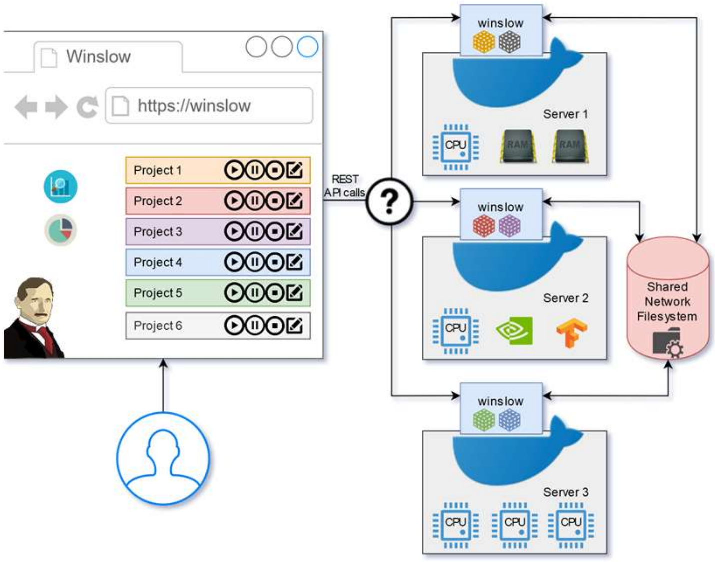
Project context

- Utilizing Computer Vision and Artificial Intelligence to detect vehicles in video footage
- Tracking vehicles throughout the video to determine speed, size, acceleration, class and position
- Find lanes, assign vehicles to lanes and detect lane changes
- Visualize detected vehicles
- Provide data for traffic flow analysis (in other projects or for the customer)

The Goal

- Automate manual and tedious multi-stage process of data extraction and distribute workload onto multiple servers





Further Requirements and Objectives

- Automatically distribute jobs onto computing nodes
- Be able to manage multiple projects simultaneously
- Handle large files (4k video footage)
- Utilize hardware acceleration for CV and AI (some jobs have specific hardware requirements)
- Provide a WebInterface for the user to manage projects, see job progress and compute node usage

Architecture, Design and Technologies

- Decentralized decision making
- Resilient against node failures
- Shared network filesystem for data, configuration and coordination
- Docker for easy installation of additional compute nodes
- Implementation in Java, using SpringBoot to provide a REST Api, use Typescript and Angular to provide a rich and responsive WebInterface by utilizing the REST API

Challenges and Experimental Work

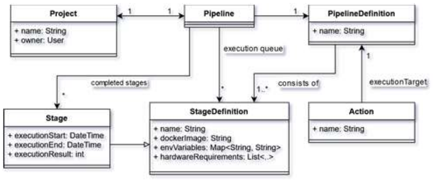
- Finding a fitting network filesystem
  - Some require big installation overhead
  - Truly decentralised filesystems are rare
  - Tons of different centralized network filesystems
  - Some provide site awareness or replication services
- How to manage separate execution history from project and pipeline template
- Can you use a shared filesystem for communication and coordination to strip down external (system) dependencies

Results

- Synchronous EventSystem with Boradcast functionality based on files on a shared filesystem
- Implementations of a timeout Mutex on-top of the EventSystem to lock projects throughout the whole system

Project Progress

- Research: DONE
- Experimental Work: DONE
  - Using file based events to synchronize, coordinate and communicate between multiple instances
  - Creating, starting and monitoring Docker container on the fly where the application has additional hardware requirements
- Actual Implementation: FINALIZING (99%)
  - Job distribution
  - Error resilience (job failure, node failure, timeouts)
  - Reacting on User Feedback (missing display of information, quality of life improvements)
- Metrics, Analysis and Evaluation (60%)
  - Finding fitting comparisons for now automated workflows
- Thesis (70%)
  - Writing everything down



- abc

Results, Evaluation and Analysis of the experimental work

Project Plan (timeline)

Aims and Objectives

Description of experimental work

Conclusion