# 1, Colour based object detection and object tracking.

From a pre-recorded video, the application capable to find and track an object with a specific colour.

Prerequisites

A video containing multiple objects with specific colour, at least 2 object is moving around the view, but any of the objects do not have to overlap each other. This video can be either downloaded from the internet or created at home. In case of home video, a washing sponge can create a good not common colour to track. All objects should be the same colour.

To read the video any library or method can be used, as long it will be provided for the final application.

The application should have a feedback interface, where both the images and the calculated results should be visible.

## Algorithm

The created algorithm will contain 2 parts, the colour based detector, and a tracking mechanism. The detector shall be ablet to find the different objects, and the tracking algorithm should be able to tell the path of the object on the video. The tracking algorithm shall be able to predict the position of the object for 1-2 frames even if the object was not detected.

The implemented algorithm can not use any outside libraries or dependencies.

## Extra points

The processing algorithm part or whole is implemented in CUDA or OpenCL.

# 2, Rule based graph modifier.

Given a defined graph and a ruleset, the application shall be able to modify original graph.

Prerequisites

We define the graph (G) as a directed acyclic graph. Each node of the graph contains a type and number of input edges attribute. This graph can be either read from a file or be able to define by simple API in the beginning of the application.

* There are multiple but given node types what is used for entry point, these nodes can have at least one directional edges out to other nodes.
* There are multiple but given node types what is used for exit point, these nodes must have defined number of directional edge in from other nodes.
* There are multiple, but given node types what is used for processing, these nodes must have at least one edge going in from another nodes, and at least one edge going out to another nodes.

We define a set of rules, what describes a subgraph to be found and a replacement graph. These rules can be either coded in the application or read from a file.

The input graph or graphs shall be created by own discretion and attached to the solution. The final input graph should not be the same as the example.

## Algorithm

The created algorithm should be able to validate the integrity of the input graph, than be able to apply the given rules to the graph.

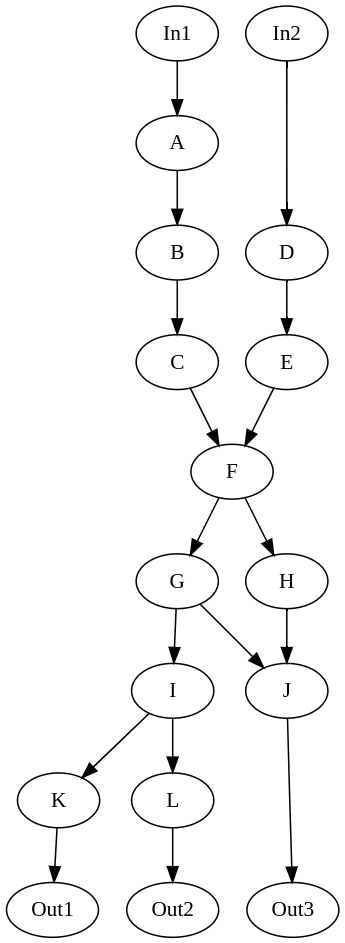
Some of the required rules are the followings:

* Given a sequence of TypeA -> TypeB -> TypeC sequence, it have to be replaced with TypeX node, where the input of TypeX is the input of TypeA, and the output is TypeC.
* Given a seqyence of TypeD -> TypeE sequence, it have to be inserted between a TypeY node.
* Given a subgraph where TypeF (with 2 inputs) is going into TypeG and TypeH. TypeG is going into TypeI and TypeJ, TypeH is going into TypeJ, TypeI is going into TypeK and TypeL, And the replaced subgraph is TypeZ is going into TypeV and TypeW. The input of type F is going into TypeZ, TypeK output will be mapped to TypeV, TypeL and TypeJ output will be mapped to TypeW output.
* Any other rules can be implemented at own discretion.

# Example

|  |  |
| --- | --- |
| Node Type | Number of inputs |
| In | 0 |
| Out | 1 |
| A | 1 |
| B | 1 |
| C | 1 |
| D | 1 |
| E | 1 |
| F | 2 |
| G | 1 |
| H | 1 |
| I | 1 |
| J | 2 |
| K | 1 |
| L | 1 |
| X | 1 |
| Y | 1 |
| Z | 1 |
| W | 1 |
| V | 1 |

Original Graph:



Output Graph:

