User Manual for CINDA

Introduction

CINDA is an efficient framework for solving the data-association of Multi-object Tracking (MOT) problem, which is based on min-cost circulation.

Supports to Python and Matlab

CINDA was implemented using C based on the efficient implementation of cost-scaling algorithm [1]. Interfaces for Python and Matlab are also provided respectively.

How to cite

Congchao Wang, Yizhi Wang, Guoqiang Yu, arXiv xx; doi: https://doi.org/xxx

Tutorials

The main function is mcc2mot. For Matlab users, it is in the file 'mcc2mot.m'. For Python users, it is in the file 'algo.py'.

INPUT of mcc2mot:

Assuming we have totally n detections in the video with detection id ranging from 1 to n, then we can build two matrices:

detection_arcs: a n x 4 matrix, each row corresponds to a detection in the form of $[id, C_{en}^{i}, C_{ex}^{i}, C_i]$;

transition_arcs: a m x 3 matrix, each row corresponds to a transition arc in the form of $[id_i, id_j, C_i, C_{i,j}]$;

NOTE that the id should be unique and in the range from 1 to n. C_{en}^{i} represents the cost of setting detection i as a start of a trajectory, which means it is a newly appeared object. C_{ex}^{i} represents the cost of setting detection i as the terminate of a trajectory, which means this object disappears in the following frames. C_i represents the confidence that detection i is true positive. $C_{i,j}$ indicates the cost that linking detection i and detection j, which means they are the two consecutive footprints of the same object. For more detailed definitions, please see the section 3 of our paper.

*Note that the detection id for the Python function should also start from 1 rather than 0.

OUTPUT of mcc2mot:

trajectories: cells containing the data-association results; each cell contains a set of ordered detection ids, which indicate a trajectory;

costs: a vector that contains the cost of each trajectories.

A sample of input is provided in the folder of 'data'. For Matlab users, the usage of CINDA can be found in file 'test.m'. For Python users, we provide a Jupyter notebook file 'test.ipynb' as illustration.

If you have any question, please raise an issue on GitHub or contact ccwang@vt.edu.

[1]. Goldberg, A. V. (1997). An efficient implementation of a scaling minimum-cost flow algorithm. Journal of algorithms, 22(1), 1-29.