Information Processing Lab
Software Package Documents

Software Package Name
SCT_IPL

Authors
Zheng (Thomas) Tang

Contents
This package is designed for fully unsupervised multiple object tracking with object detection, segmentation, adaptive appearance modeling and camera self-calibration. At every frame, object candidate(s) can be co-located by detection and segmentation. If the camera parameters are not available, 2D Kalman-filter-based tracking is executed to collect object positions for camera self-calibration. After the camera projection matrix is derived, the tracking process can be extended to 3D space. For objects moving in group(s), cross-matching based on adaptive appearance modeling is employed to avoid confusion among nearby targets. When an object is seriously occluded or missing, adaptive appearance model is applied for object re-identification to match disappeared node(s) with entering node(s).

This algorithm has achieved top performance on the MOTChallenge 2015 3D benchmark (https://motchallenge.net/results/3D_MOT_2015/). But it is still undergoing further development to achieve robust performance for long-term uncontrolled scenarios in real world.

Note that because YOLO requires additional compilation by NVCC besides GCC, currently the component of online YOLO detection is unavailable. Also, the utility of local camera mode and IP camera mode have not been tested yet.

Code Structure
1. `./obj/` folder: Libraries of .cpp files for compilation
2. `./src/` folder: Header files and main function
3. `./data/` folder: Example
    a. `cfg.json`: Important configuration parameters in JSON format
    b. `vdo.mp4`: Input video source
    c. `roi.jpg`: Plotted region of interest
    d. `camParam.txt`: Input camera parameters
    e. `./det/` folder: Detection results in text file
    f. `./haarcascade_frontalface_alt.xml` folder: Input model for Haar-cascade face detection

How to Build

1. Download and make the OpenCV 3 library.
2. Compile the provided source code and libraries using g++ in Linux environment.

How to Use

Inputs

The tracking system has five basic inputs:

1. configuration parameters, described in the JSON file
2. input video source, described by `inVdoTyp`
3. detection results, described by `inDetTyp`

The text file of detection results should follow the format as follows (following the standard of MOTChallenge):

<frame no.>,-1,<2D x (left)>,<2D y (top)>,<2D width>,<2D height>,<detection score>,-1,-1,-1,<object class (optional)>

The object class can be ignored for default object tracking when `detObjClsFlg` is 0.

4. segmentation results, described by `inSegTyp`
5. camera parameters, described by `inCalTyp`

The text file of camera parameters should follow the format as follows, where K, R, T, and P respectively stand for intrinsic parameter matrix, rotation matrix, translation matrix and the camera projection matrix:

K[0] K[1] K[2] K[3] K[4] K[5] K[6] K[7] K[8]
R[0] R[1] R[2] R[3] R[4] R[5] R[6] R[7] R[8]
T[0] T[1] T[2]
P[0] P[1] P[2] P[3] P[4] P[5] P[6] P[7] P[8] P[9] P[10] P[11]

Outputs

The only necessary output is the tracking results in text file in the format below (following the standard of MOTChallenge).

<frame no.>,<object ID,<2D x (left)>,<2D y (top)>,<2D width>,<2D height>,<detection score>,<3D x>,<3D y>,<3D z>, <3D depth (optional)>,<3D visibility (optional)>

The depth and visibility information is for multi-view object tracking, which can be ignored by setting `outDepVisFlg` as 0. When `outTrkDetFlg` is set, the output 2D and 3D positions are derived from detection results.

The user can also run face detection within each object region by setting `fcDetFlg`. The output format is as follows.

<frame no.>,<object ID,<face ID>,<2D x (left)>,<2D y (top)>,<2D width>,<2D height>

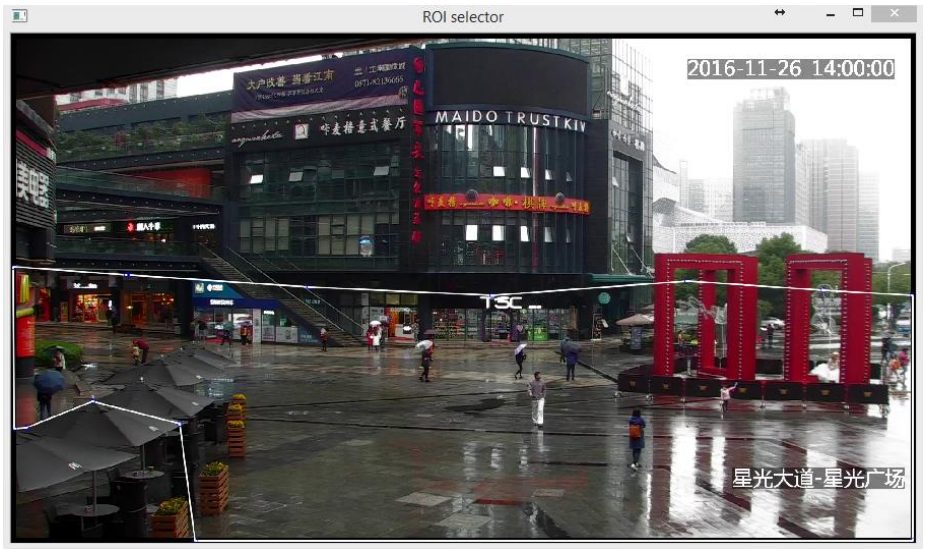
The output images of cropped faces are saved as <frame no.>_<object ID>_<face ID>.jpg in the

provided output folder.

The user can choose to output other results such as plotted frames in video/image, original frames, segmentation results, detection results, etc. But the computation time will increase accordingly.

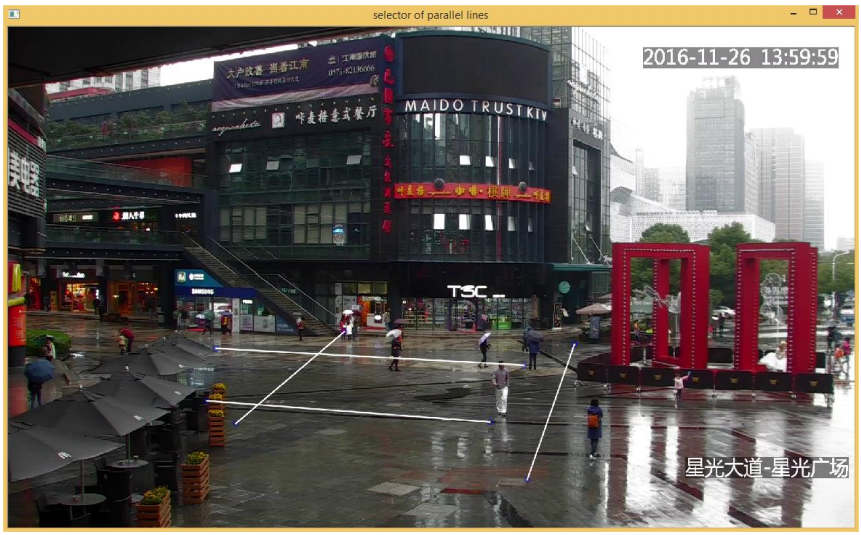
Start Tracking

1. Set the corresponding input/output paths in the configuration file if necessary.
2. When running the program, a window called “current frame” will pop up first.
3. When the ROI image is not selected, an ROI image of the entire frame will be automatically created. If the user wants to select a specified ROI, set `selRoiFlg` to 1. The user can perform 3 choices: 1) press `p` to proceed to the frame that s/he wants to select ROI in, 2) press `o` and load the existing ROI image, 3) press `s` to select ROI in the current frame. The process of selecting ROI is described as follows:
    a. A new window called “ROI selector” pops out. This should be the current frame that the user proceeds to.
    b. Click on the image around the ROI. A blue circle stands for each click and a white line connect two adjacent clicks. Note that in order to make a valid ROI, the last click should be close to the first click, i.e., two blue circles should overlap.

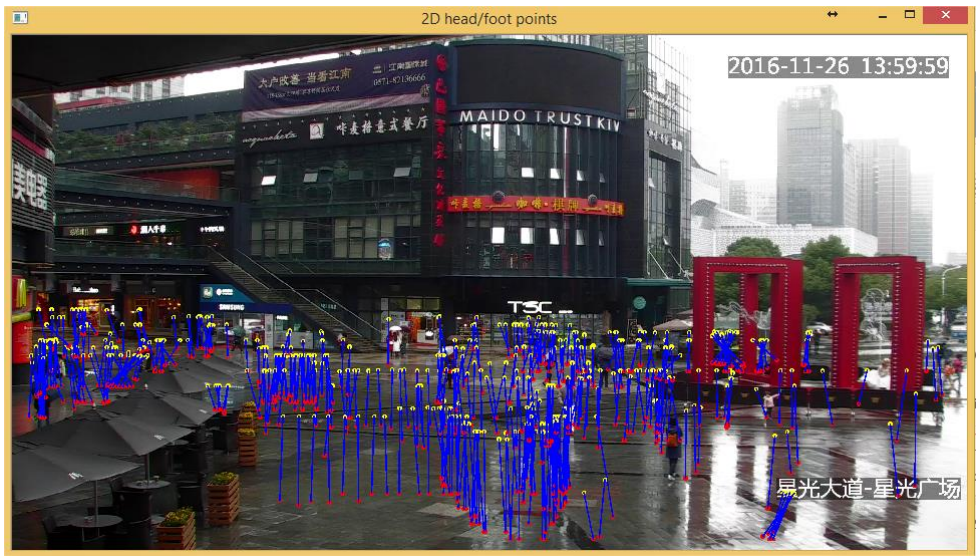


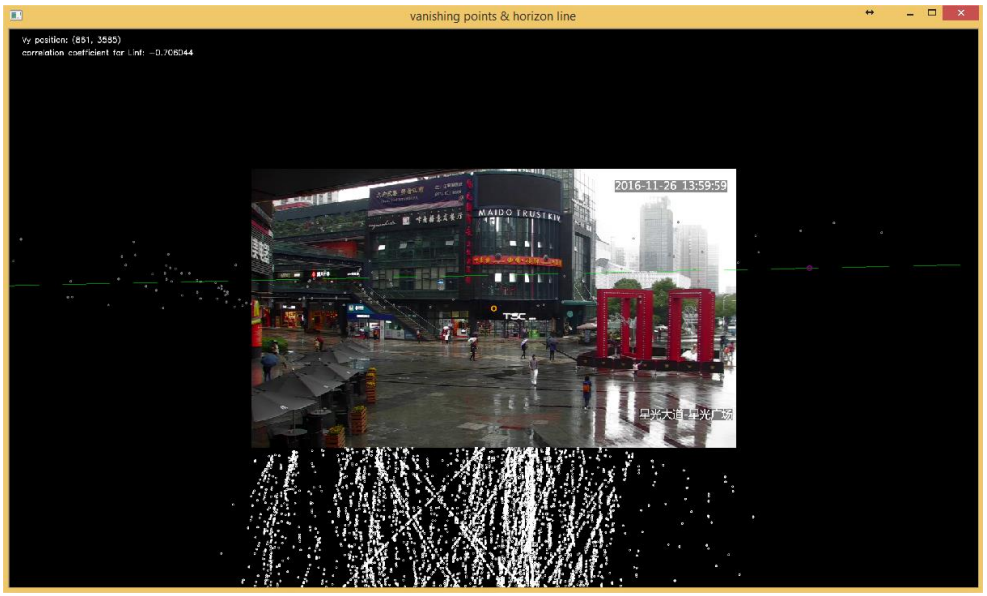
    c. After the selection of ROI is done, click `o`. A binary mask image that shows the mask for ROI is created.
    d. During ROI selection, if mis-clicking on wrong place(s), the user can press `r`. All the markers will be cleared, and s/he can start over.
4. When the detection results are not available, the user can choose to run HOG detector (for person only) online. When `detMltSclTstFlg` is set, each frame will be tested in multiple scales and the results are combined through non-maximum suppression, so that more small objects can be detected. The user also needs to set the threshold for detection score `detScrThld`, the object class(es) of interest `detObjCls`, and the range(s) of aspect ratio(s) `detAspRatRng`.

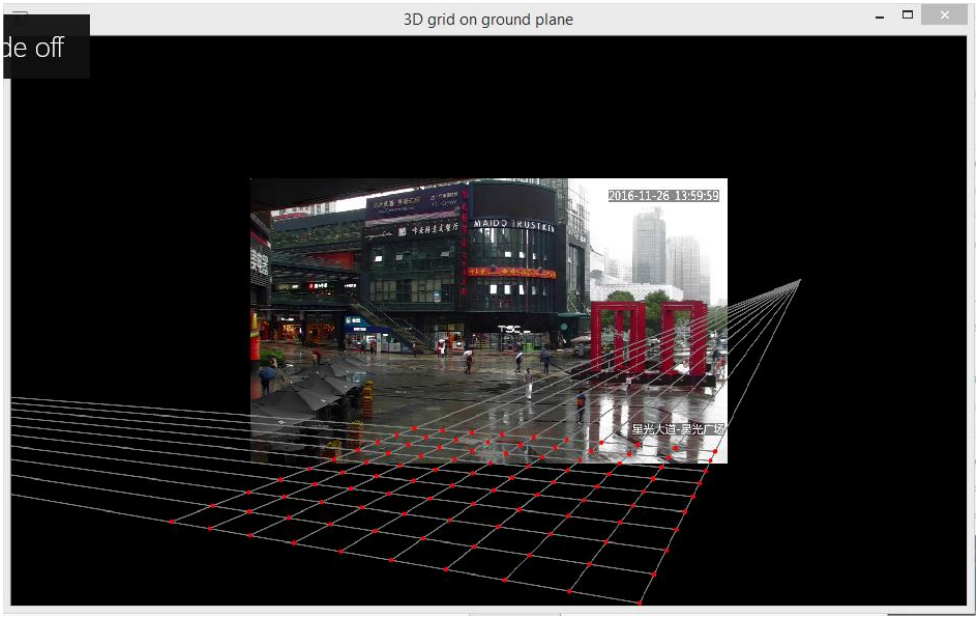
5. If segmentation results are not available, the user can choose to run segmentation using either KNN, MOG2 or SuBSENSE online. It is suggested to resize the original frame with `segRszRat` to increase processing speed. The user can also choose to enable shadow removal with `segShdwRmvFlg` set. Tracking by segmentation may not perform well for crowded scene. To run without segmentation, the user can set `trkSegFlg` to 0.
6. The camera parameters are not necessary when `trkDim` is 2 instead of 3. However, the 2D tracking results are inferior to 3D tracking. When the camera parameters are not available, the user can choose manual calibration or self-calibration.
    a. The user needs to provide an approximate range (ideally 0.5 to 1 meter) of camera height in `calCamHeiMax` and `calCamHeiMin`.
    b. For manual calibration, if `calSelVanLnFlg` is not set, the user can manually input the 2D coordinates of two vanishing points Vr (on the right) and Vl (on the left) respectively in `calVr` and `calVl`. Otherwise, s/he can follow similar procedure as ROI selection to select two pairs of vanishing lines, i.e., parallel lines on the ground plane in 3D that are perpendicular to each other, in the window of “selector of vanishing lines”. Please select a pair of vanishing lines first and then another pair. There should be 8 points clicked in total. After the selection of vanishing lines is done, click `o`.



    c. For self-calibration, 2D tracking is automatically run in backend to collect sufficient number of tracking positions for self-calibration. The interval of collection of tracking positions is controlled by `calCol1dIntvSec`. And the sufficient number of tracking positions is defined by `calVyCandNumThld` and `calLinfCandNumThld`. When the progress reaches 100%, a window called “2D head/foot points” pops out, which displays the collected 2D head/foot points. Press any key to continue. Then, another window called “vanishing points & horizon line” shows up to present the fitted horizon line and estimated vanishing points. Again, press any key to continue. Finally, the window “3D grid on ground plane” will appear showing the final calibration result. Press any key to start tracking.


2D head/foot points
2016-11-26 13:59:59
MAIDO TRUSTKIV
TSC
星光大道-星光广场

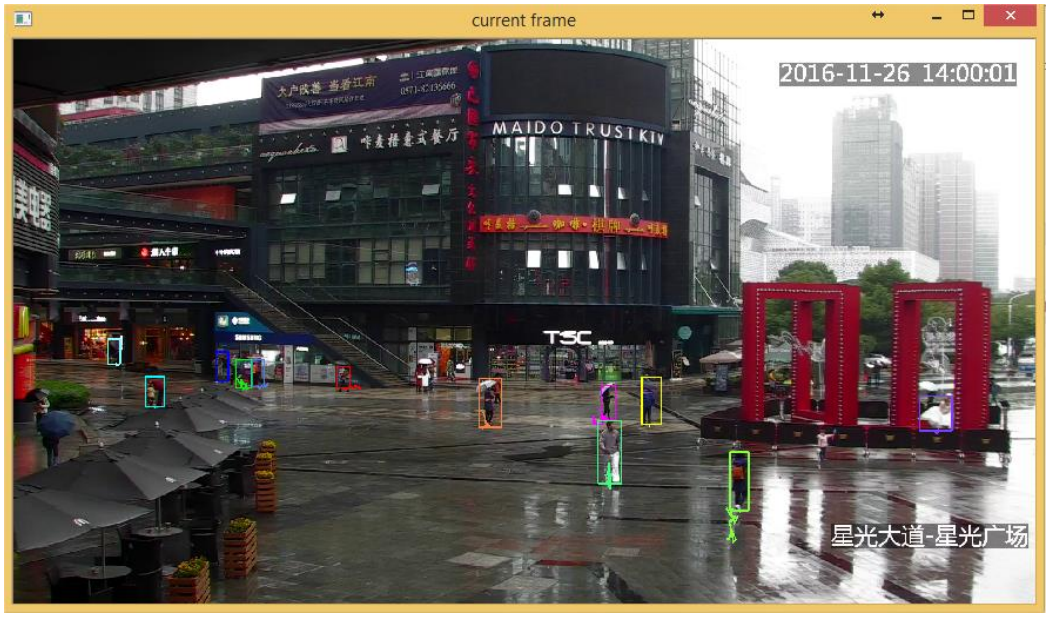

vanishing points & horizon line
Vy position: (851, 3585)
correlation coefficient for Lint: -0.706044
2016-11-26 13:59:59
MAIDO TRUSTKIV
TSC
星光大道-星光广场


3D grid on ground plane
de off
2016-11-26 13:59:59
MAIDO TRUSTKIV
TSC
星光大道-星光广场

d. When `calSelTstPtFlg` is set, the estimated camera parameters can be verified by measuring 3D distance(s) on the ground plane. When a window named `selector of test points` pops out, similar to Step c, the user can select any number of test points that form line segment(s) on the ground plane with known 3D distance(s). Press `o` and then all the estimated 3D distance(s) based on current camera parameters will be shown in order at the terminal. When the measured distance(s) are too long or too short, the user can decrease or increase `calCamHeiMax` and `calCamHeiMin` respectively to fine-tune the result. Please make sure that the estimated length(s) of line segment(s) close to the camera are as accurate as possible.
e. When `calEdaOptFlg` is set, the camera parameters will be optimized by the estimation of distribution algorithm (EDA) to minimize reprojection error. If `calReprojErrTyp` is set to 0, the reprojection error is the total distance between each grid point tot the grid lines. Otherwise, if `calReprojErrTyp` is set to 1, the reprojection error is defined by the measurement error of a few line segments compared to ground truth. The user can input the end points of line segments and their true 3D distances respectively in `calMeasLnSegNdPt` and `calMeasLnSegDist`.

7. The distance thresholds `trk3dDistThld` and `trk2dDistThld` are used to determine the group of nearby object nodes for cross-matching and re-identification. The user can build the appearance model with different combinations of feature space by setting `trkAppMdlClrFlg`, `trkAppMdlLbpFlg` and `trkAppMdlGradFlg`. The distance thresholds for the computation of matching samples in similarity measurement are set at `trkAppMdlClrDistThld`, `trkAppMdlLbpDistThld`, `trkAppMdlGradMagDistThld` and `trkAppMdlGradAngDistThld`. To increase computation speed, the user can decrease the size of each appearance model in `trkAppMdlSz` and `trkAppMdlSmpNumSec`. By adjusting `trkMtchScrThld`, the user can fine-tune the performance cross-matching and re-identification. Smaller threshold will lead to easier identification. Each target is tracked when it has been matched for a number of frames, controlled by `trkNtrTmSecThld`, which should be set larger when there are many false positives. We can also control the number of frames to keep left object nodes for re-identification at `trkLftTmSecThld`. Another important configuration parameter is `trkPredTmSecThld`, which determines a node in the past trajectory to help predict the movement of a left object for re-identification. To improve the continuity of tracklets, each object is tracked by prediction of Kalman filter for several frames even when the detection is missing. This may lead to some false positives. The balance is controlled by `trkHypTmSecThld`. Finally, `trkTrajTmSecThld` defines the number of frames to keep past trajectories for targets.
8. For display, the user can choose to plot tracking results (as colorful bounding boxes for different object IDs), detection results (as black bounding boxes with object classes), and segmentation results (as blended image) at the display window by setting `pltTrkResFlg`, `pltDetFlg` and

`pltSegFlg` respectively. The past trajectories of foot points can also be plotted when `pltTrajTmSec` is larger than 0.



9. Feel free to contact the author through zhtang@uw.edu if you have any other question.