Information Processing Lab
Software Package Documents

Software Package Name

CMK3dVehTrk 08222017

Authors

Zheng (Thomas) Tang, Gaoang Wang, Tao Liu, Young-Gun Lee, Adwin Jahn, Xu Liu, Xiaodong He, Jenq-Neng Hwang, Kuan-Hui (Young) Lee and Chun-Te (Randy) Chu

Contents

This package is designed for vehicle tracking with Kalman-based Constrained Multiple Kernel (KCMK) and 3D vehicle model localization techniques. Please refer to our final paper/report for further details of the algorithm. The package includes the source code and the designed interface for developer.

Code Structure

1. "CMK3dVehTrk.sln": Solution file of Microsoft Visual Studio 2010
2. "CMK3dVehTrk" project folder: Static libraries of source code
3. "CMK3dVehTrkAPI" project folder: Working folder
   a. "src" folder: Source code of the example "main.cpp", interface APIs "IndVehicleTracking.h" and "IndVehicleTracking.cpp", and headers of libraries
   b. "data" folder: Inputs stored in the folder(s) of each camera ID
   c. "bin" folder: Compiled executable file "CMK3dVehTrkAPI.exe" and outputs

How to Build

4. In Windows environment (tested on Windows 8.1 x64), install Microsoft Visual Studio 2010 – Professional using "en_visual_studio_2010_professional_web_installer_x86_516532.exe". Install only necessary components of Visual C++.
5. Install OpenCV v2.1+ package (tested on OpenCV v2.4.3). Or you can extract "OpenCV-2.4.3.zip" to your target directory.
6. Open "CMK3dVehTrk.sln".
7. Set proper directories for OpenCV headers and libraries in VS2010 (for both projects of "CMK3dVehTrk" and "CMK3dVehTrkAPI" in configurations of both "Release" and "Debug"): Right click on the project name(s) in "Solution Explorer" -> Properties -> Select "Platform" of "x64" instead of "Win32" (If this choice does not exist, create one through the "Solution Platform" on top of the homepage) -> Configuration Properties -> VC++ Directories -> Set "Include Directories" and "Library Directories". The default is "OpenCV-2.4.3" at the root

directory of C drive.

8. Set the configuration and other inputs in the folder “data”.
9. Build and run the program.
10. The compiled executable file and outputs are saved in the folder “bin”.

How to Use

Here we give an example about how to use the interface APIs and coding flow. The example is coded in the file “main.cpp”.

The vehicle tracking system needs five inputs (included in the folder “data”):

1) configuration file, described later.
2) camera parameter file, described later.
3) background image (can be set in the beginning, or set by each frame processing)
4) ROI image (can be set in the beginning, or set by each frame processing)
   This is optional. If there’s no ROI image, ROI covers whole image.
5) video (frame)

Prerequisite: The surveillance camera should be well calibrated. The camera parameters file should include camera’s intrinsic parameters and extrinsic parameters, and follow the format as below:

fx	0	cx	0	fy	cy	0	0	1
R[0]	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]
C[0]	C[1]	C[2]						

The file path of the camera parameters is needed to be assign to “InExCamParamPath” within the configuration file.

Outputs:

In the “bin” folder, the output 3D vehicle model(s) at each frame are saved in the “camID” folder as “carID_frameID.txt”: L, W1, W2, H1, H2, H3, H4, X1, X2, X3, X4, Ta, Translation.x, Translation.y, Translation.z, Rotation.x, Rotation.y, Rotation.z, DeformableRange.ScaleLength, DeformableRange. ScaleWidth, DeformableRange.ScaleHeight. Please read the vehicle modeling paper referred to in our final paper/report to understand the concepts of these parameters. The command window output is saved in “log.txt”. The output frames with 3D models/tracking results plotted are saved in the folder “results”. The FES (Fitness Evaluation Score) for each initialized 3D vehicle model is saved in the folder “fes_score” as “carID_fes_scoreframeID.txt”.

Start Coding:

In the beginning, we have to claim an object of the tracking system by setting number of the total cameras as an input parameter. In the example, we only use 1 camera:

CIndVehicleTracking ind_vehicle_tracker(1);

Next, we have to do the initialization by calling:

IndVehicleTracking_Init(cam_id, iVdoWidth, iVdoHeight);

There are 4 input parameters: 1) camera ID, 2) width, 3) height of the videos, and 4) path of configuration file. The tracking system then read the configuration file (ex: “cam_config.ini” under the working directory). This file defines the configurations, including many thresholds and pre-determined constants.

After initialization, we have to set the background image by calling:

IndVehicleTracking_ SetBgImg(cam_id, bgData);

This function can also be called during the processing to update the background. As for ROI, there is a corresponding function named:

IndVehicle Tracking_SetBgImg(cam_id, roiData);

For each video frame, we have to call:

IndVehicleTracking_Process(cam_id, imageData);

to process the tracking. There are 3 input parameters, 1) camera ID, and 2) image data in terms of BYTE*. The image data is RGB format in order of *RGBRGBRGB*..., and the time stamp is used to count the video.

Finally, we release the resource using in the tracking system by calling:

IndVehicleTracking_UnInit(cam_id);

Interface APIs

- BOOL IndVehicleTracking_Init(int iCamerID, int iVideoWidth, int iVideoHeight, const char* ConfigPath)

  Initialization of the tracking system, should be called in the beginning.

iCamerID [in]	camera ID
iVideoWidth [in]	width of the camera/video
iVideoHeight [in]	height of the camera/video
return	TURE means success, FALSE means fail

- void IndVehicleTracking_UnInit(int iCameraID)

  Release the resource using in the tracking system, should be called in the end.

iCamerID [in]	camera ID

- BOOL IndVehicleTracking_Process(int iCameraID, BYTE *pbCurrentFrameRGB24)

  Doing processing of human tracking by a given camera ID.

iCamerID [in]	camera ID
pbCurrentFrameRGB24 [in]	row data of the current frame, in *RGBRGB…* order
return	TURE means success, FALSE means fail

- void IndTracking_SetBgImg(int iCameraID, BYTE* pbBgImgRGB24)

  Set background image by a given camera ID.

iCamerID [in]	camera ID
pbBgImgRGB24 [in]	row data of the background image, in *RGBRGB…* order

- int IndTracking_GetObjectCnt(int iCameraID)

iCamerID [in]	camera ID
return	the total number of the tracked objects

- CVehicleClass IndVehicleTracking_GetObject(int iCameraID, int iObjectIdx)

iCamerID [in]	camera ID
iObjectIdx [in]	index of the tracked object, should be positive and smaller than the return value of IndVehicleTracking_GetObjectCnt()
return	the information of the specific tracked objects, CVehicleClass, which is inherited from CObjectClass, is defined in the file “IndVehicleTracking.h”

Configurations

- *General Options*:

SegmentationEngine	0: basic background subtraction.
ShapeFittingEngine	0: deformable model.
KernelTrackingType	0: 2D constrained, 1: 3D constrained in multiple kernel tracking.
EnableVCD	0: disable, 1: enable Vehicle Color Detection.
ResizeRatio	ratio of resizing frame for processing, must be >= 1.

- *Tracking System*:

MarginThreshFar	threshold of blob within margin of far field (1/1000).
MarginThreshMid	threshold of blob within margin of middle field (1/1000).
MarginThreshNear	threshold of blob within margin of near field (1/1000).
Enlargefactor	factor of blob area suddenly enlarging (1/1000).
ScaleBase	up/down scale ratio according to the y shift /(1/10000000).
ValidVel2dThresh	valid velocity in 2D.

ValidVel3dThresh	valid velocity in 3D.
EstVelocityMethod	method for velocity estimation. 0: medium, 1: mean.
EstVelocityFrames	Minmum frames' number used for velocity estimation.
EnterFrameThresh	threshold of # frame for Kalman prediction.
3DModelUpdateDuration	update 3d model per x frames.
KernelUpdateDuration	update kernels per x frames.
InitPoseSftFarX	shift of initial position X (1/1000) in far field.
InitPoseSftFarY	shift of initial position Y (1/1000) in far field.
InitPoseSftMidX	shift of initial position X (1/1000) in middle field.
InitPoseSftMidY	shift of initial position Y (1/1000) in middle field.
InitPoseSftNearX	shift of initial position X (1/1000) in near field.
InitPoseSftNearY	shift of initial position Y (1/1000) in near field.

- *Camera Parameters*:

CameraParamType	0: none, 1: projective matrix, 2: homography matrix.
InExCamParamPath	path of intrinsic and extrinsic camera parameters.

- *Basic Background Subtraction*:

BgMode	0: first frame, 1: load from image, 2: by assigned.
OtsuThresh1	threshold 1 of Otsu algorithm.
OtsuThresh2	threshold 2 of Otsu algorithm.
EnableHoleFilling	0: disable, 1: enable (cause higher computation).
TopHat	threshold for applying morphology (apply when TopHat <= 30).
Close	structing element size in closese operation.
Open	structing element size in opense operation.
Dilate	structing element size in dilation operation.
Erode	structing element size in erosion operation.
FarLineRatio	ratio of height deviding far-middle fields, [0(0%), NearDistance].
NearLineRatio	ratio of height deviding far-middle fields, [FarDistance, 100000(100.000%)].
BlobThreshFar	threshold of blob in far field, [0(0%), 100000(100.000%)].
MiddleBlobThresh	threshold of blob in middle field, [0(0%), 100000(100.000%)].
NearBlobThresh	threshold of blob in near field, [0(0%), 100000(100.000%)].
BgImgPath	if BgMode == 1, load image from this path.

- *Deformable Model*:

SegmentL	length of virtual rectangle.
SegmentW	width of virtual rectangle.

GradientThreshSF	threshold of image gradient for shape fitting.
ContrastSF	contrast adjusted before shape fitting, [0, 255].
RSF	population # of EMNAglobal used in shape fitting.
NSF	best selection # of EMNAglobal used in shape fitting.
CriterionThreshSF	stopping criterion in shape fitting (1/100).
GradientThreshPE	threshold of image gradient for pose estimation.
ContrastPE	contrast adjusted before pose estimation, [0, 255].
RPE	population # of EMNAglobal used in pose estimation.
NPE	best selection # of EMNAglobal used in pose estimation.
CriterionThreshPE	stopping criterion in pose estimation (1/100).
DisplayProjection	display result per projection (for debug).
DisplayIteration	display result per iteration (for debug).
ScaleLengthFar	length scale of 3D model in far field (1/1000).
ScaleWidthFar	width scale of 3D model in far field (1/1000).
ScaleHeightFar	height scale of 3D model in far field (1/1000).
ScaleLengthMiddle	length scale of 3D model in middle field (1/1000).
ScaleWidthMiddle	width scale of 3D model in middle field (1/1000).
ScaleHeightMiddle	height scale of 3D model in middle field (1/1000).
ScaleLengthNear	length scale of 3D model in near field (1/1000).
ScaleWidthNear	width scale of 3D model in near field (1/1000).
ScaleHeightNear	height scale of 3D model in near field (1/1000).

- *Debug*:

DisplayThickness	thickness of the blobs/blocks.
DisplayBlobs	0: do not, 1: display all catched blobs in basic background subtraction.
DisplaySegment	0: do not, 1: display segmentation of foreground objects.
DisplayBlock	0: do not, 1: display surronding blocks of tracked objects.
DisplayPath	0: do not, 1: display tracked objects' moving path.
Display3dModel	0: do not, 1: display 3D vehicle model.
DisplayKernels	0: do not, 1: display multiple kernels.
DisplayProjection	[for Deformable Model] 0: do not, 1: display result per projection.
DisplayIteration	[for Deformable Model] 0: do not, 1: display result per iteration.
SaveBlobs	0: do not, 1: save the blob (segmentation) results.
SaveDebugFrames	0: do not, 1: save the debug frames.
BeginFrame	the frame whcih the system starts from.