An OpenISS Framework Specialization for Deep Learning-based Person Re-identification

Master of Science Thesis Defense

Haotao Lai (Eric)

Department of Computer Science and Software Engineering
Gina Cody School of Engineering and Computer Science
Concordia University, Montreal, Canada

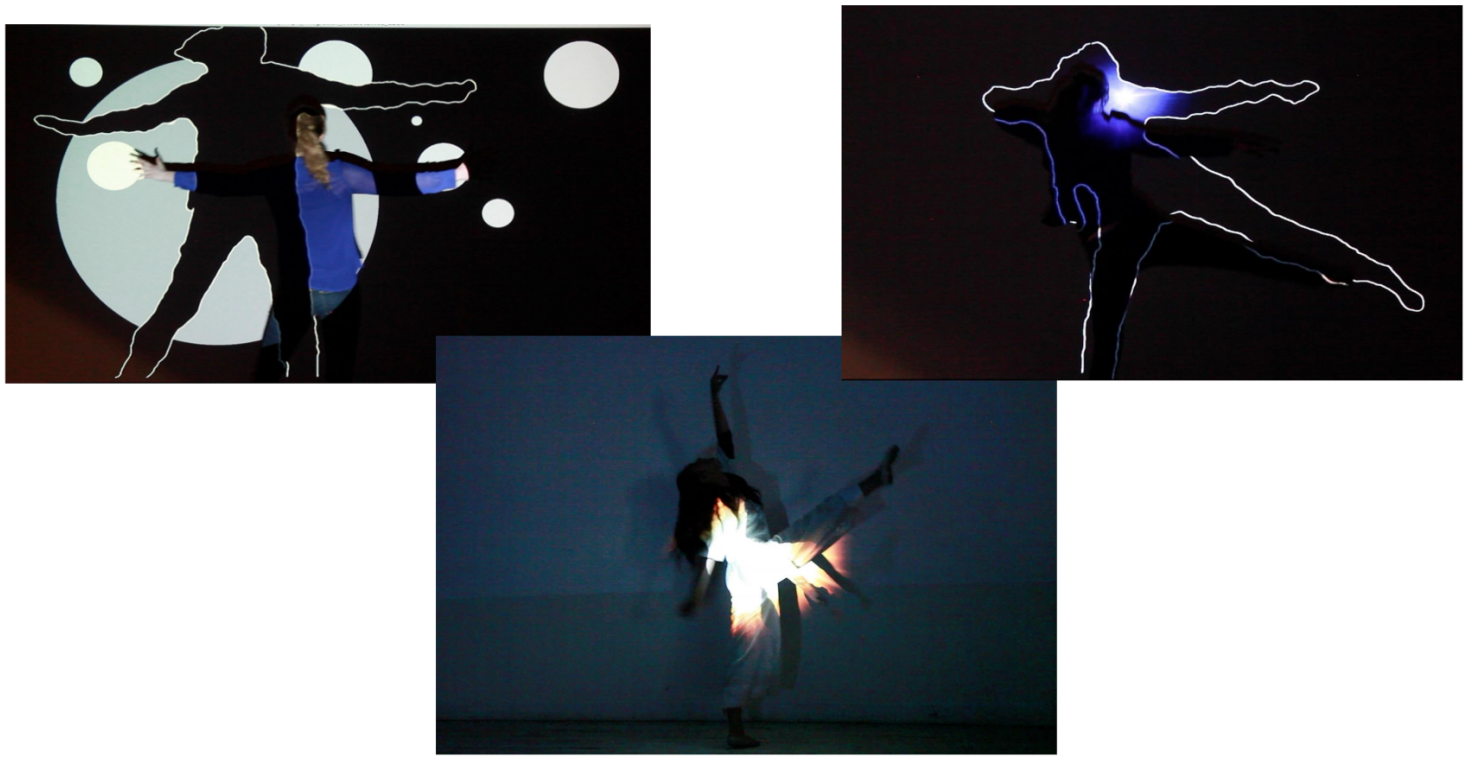
August 21, 2019

Outline

Introduction

Background

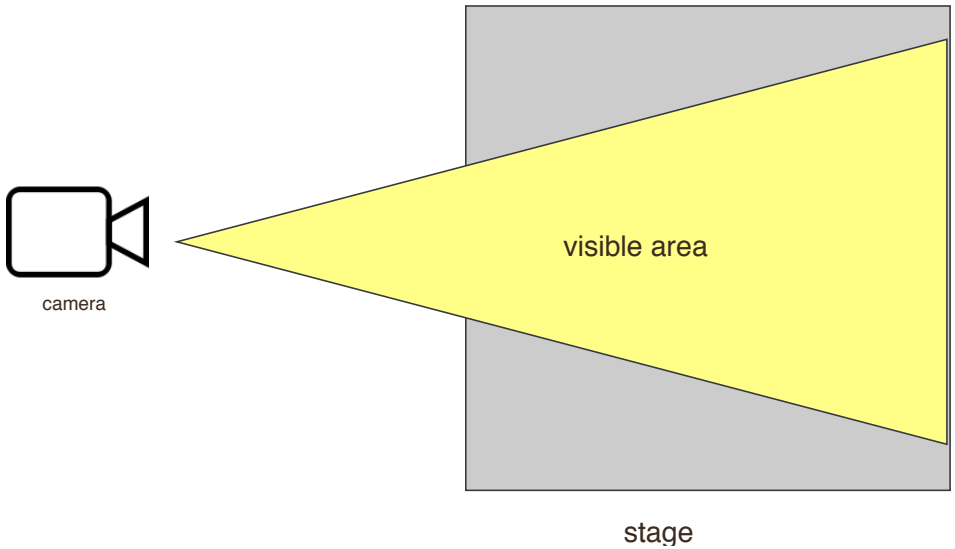
Due to our previous work [10] named ISSv2, we were able to create artistic performance on the stage including real-time motion capture, projection mapping with one Kinect v1 camera.



Problems

When we have more chances to do our performance, we encounter the following problems:

- **PM1**: The stage is too large to cover by a single camera.
- **PM2**: Various advanced cameras come to market and we would like to make our work to be compatible with them.

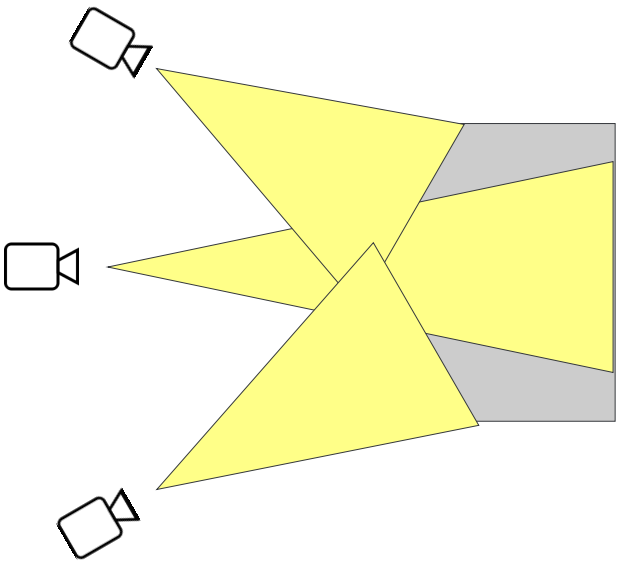


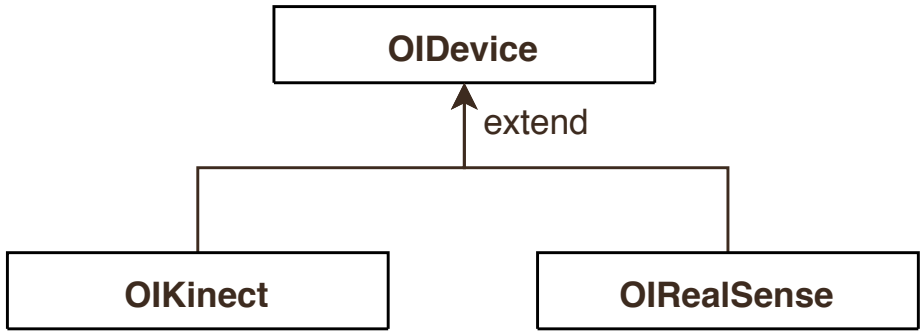
- Kinect v1
- Kinect v2
- Realsense D435
- ... ...

Possible Solution

Possible solution may be:

- For **PM1**, use more cameras to cover the stage.
- For **PM2**, abstract a set of common APIs for various kinds of devices.





Goal

To sum up, we would like to design and implement a system that with the following functionalities or features:

- device abstraction
- person re-identification
- real-time response
- accuracy
- extensibility
- usability

Contributions

1. Design and implement a framework solution in general.

   detection | recognition | tracking

2. Design and implement a device module for depth cameras encapsulation.

   Kinect v1 | Kinect v2 | RealSense D435

3. Design and implement the tracker, detector and recognizer specialized frameworks.

   NiTE2 → skeleton tracking | YOLO v3 → person detection | ID&Triplet Models → person ReID

4. Design a pipeline module for filter linear execution.

   Person ReID application | Skeleton tracking application

5. Build commonly used computer vision applications using our solution.

   camera calibration | image alignment | green screen image

6. Design and implement an dataset abstraction layer for deep learning-based person ReID task.

Related Work

Overview

Our research problem can be divided as follows:

- person re-identification
  - **person detection**
  - person tracking (omitted)
  - **person retrieval**
- device abstraction

Object Detection and Object Retrieval

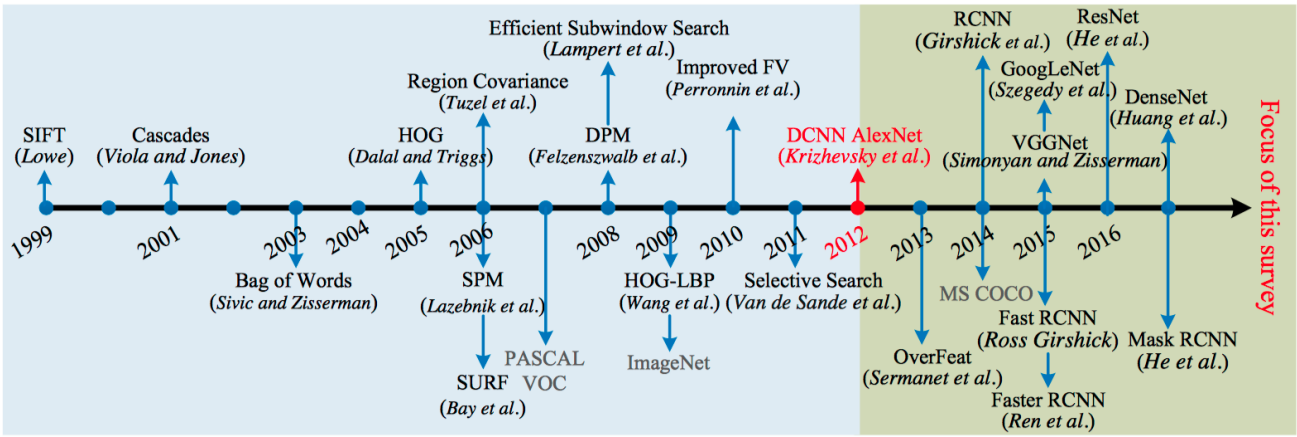


Figure 1: Timeline of various methods proposed for object detection [8]. The methods in blue area are the hand-crafted detector and the methods in green area are the deep learning-based approaches.

	Based On	2010	2011	2012	2013	2014	2015	2016	2017	2018
CVPR	Feature designing	1	0	0	1	2	5	3	3	0
	Distance metric	0	1	1	1	0	1	4	4	2
	Deep learning	0	0	0	0	1	2	5	7	25
ICCV	Feature designing				2		4		3	
	Distance metric				0		2		2	
	Deep learning				0		0		4	
ECCV	Feature designing					2		2		0
	Distance metric					1		3		1
	Deep learning					0		2		18

Figure 2: The number of ReID papers depending on different approaches included by the three top conferences in recent years [14].

Object Detection

In object detection, you are given an image I and a list of classes C.
You are asked to find all the presences of objects listed in C within I.
For each detected instance, locates it using a bounding box B with a confidence score s and its class label $c \in C$.

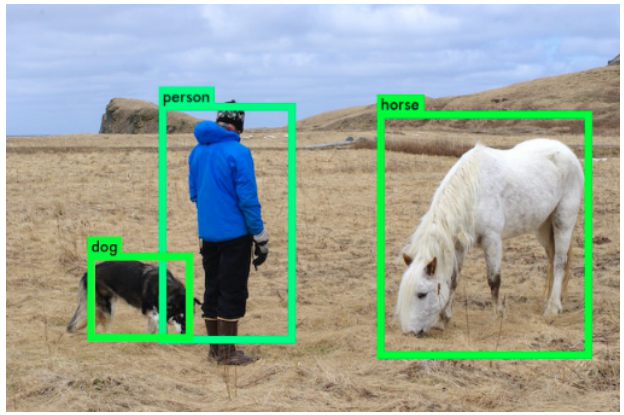


Figure 3: A sample result produced by YOLO detector.

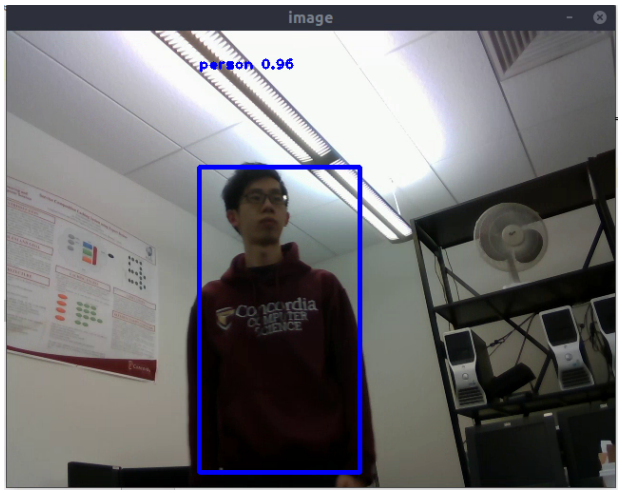


Figure 4: A sample result produced by our solution.

Existing Solutions for Object Detection

- Two-stages approaches
  - R-CNN
  - SPP-net
  - Fast R-CNN
  - Faster R-CNN
  - Mask R-CNN
- One-stage approaches
  - YOLO v1
  - YOLO v2
  - **YOLO v3**
  - SSD

YOLO v3 Model Review

Model	Train	Test	mAP	FLOPS	FPS
SSD300	COCO trainval	test-dev	41.2	-	46
SSD500	COCO trainval	test-dev	46.5	-	19
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40
Tiny YOLO	COCO trainval	test-dev	23.7	5.41 Bn	244
SSD321	COCO trainval	test-dev	45.4	-	16
DSSD321	COCO trainval	test-dev	46.1	-	12
R-FCN	COCO trainval	test-dev	51.9	-	12
SSD513	COCO trainval	test-dev	50.4	-	8
DSSD513	COCO trainval	test-dev	53.3	-	6
FPN FRCN	COCO trainval	test-dev	59.1	-	6
Retinanet-50-500	COCO trainval	test-dev	50.9	-	14
Retinanet-101-500	COCO trainval	test-dev	53.1	-	11
Retinanet-101-800	COCO trainval	test-dev	57.5	-	5
YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35
YOLOv3-608	COCO trainval	test-dev	57.9	140.69 Bn	20

Figure 5: Performance on the COCO dataset.

Image is from `https://pjreddie.com/darknet/yolo/?source=post_page`

Object Retrieval

In object retrieval, you are given a query image p and a set of gallery images G. Your task is to find the most likely image $g \in G$ for which both p and g represent the same instance but captured by different cameras.

Figure 6: An example of person retrieval process [1].

[1] Image is from https://www.micc.unifi.it/projects/person-re-identification/

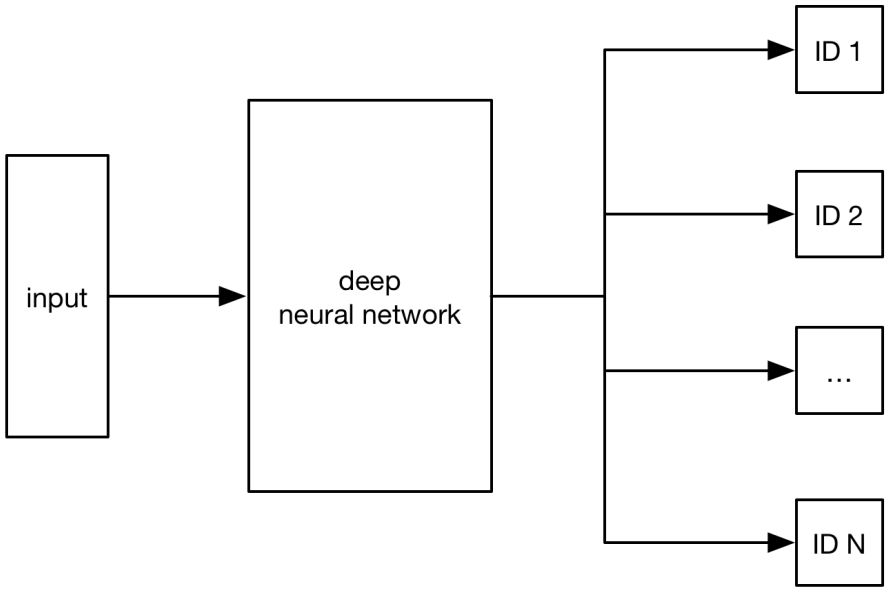
Existing Solutions for Person Retrieval

- **Identification model**
- Verification model
- **Distance metric-based model**
- Parts-based model
- Others

Identification Model Review

ID model: the retrieval problem $\rightarrow$ classification problem.

- For each input, try to tag it with a label.
- The training is guided by a cross-entropy loss function.
- The network is actually served as a feature extractor.



Triplet Model Review

- triplet unit
  - anchor
  - positive
  - negative
- distance function
  - euclidean distance
  - cosine similarity
- loss
  - batch-all loss
  - batch-hard loss

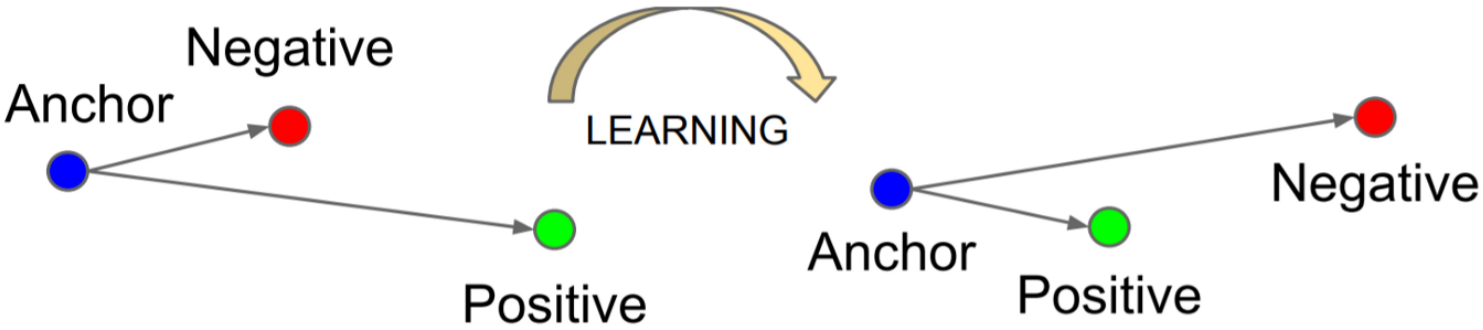


Figure 7: The learning objective of the triplet model [13].

Available Software

For device abstraction:

- Freenect and Freenect2
- RealSense SDK
- OpenNI2 and NiTE2
- .....

For image processing and visualization:

- OpenCV
- OpenGL
- .....

For deep learning development:

- TensorFlow + Keras
- Pytorch
- .....

Solution

Why Framework Solution ?

Features of a software framework perfectly fit to the needs of our solution so we decide to choose the framework solution.

What is a software framework [11]:

- frozen spot
- hot spot

Key features of a framework [2]:

- inversion of control
- non-modifiable framework code
- extensibility

Design Overview

Our framework named OpenISS consists of two parts:

- core framework
- specialized framework

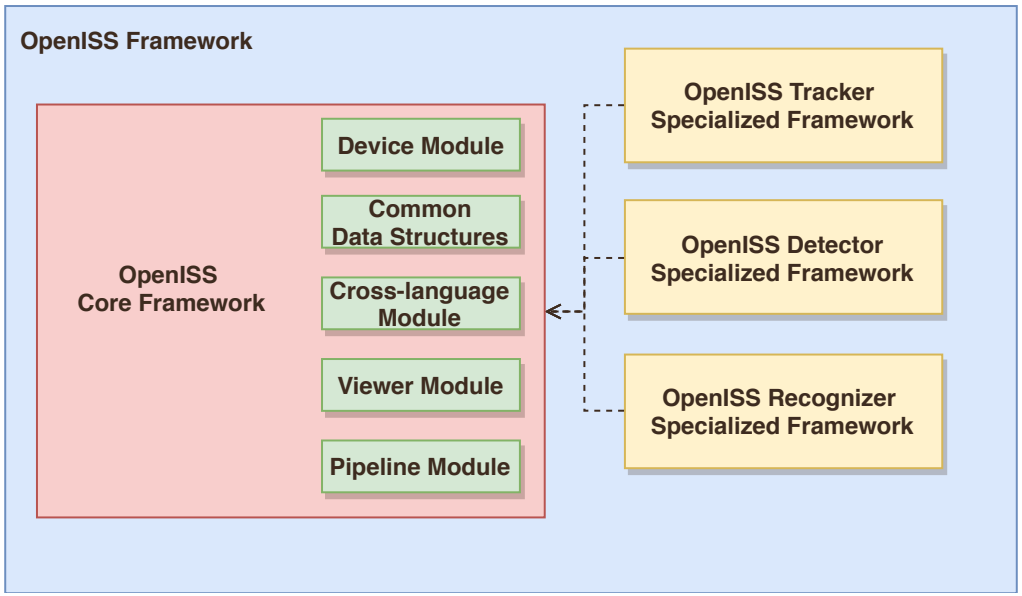


Figure 8: OpenISS framework architecture.

Design of Device Module

- abstract common APIs, create an abstract class
- factory design pattern

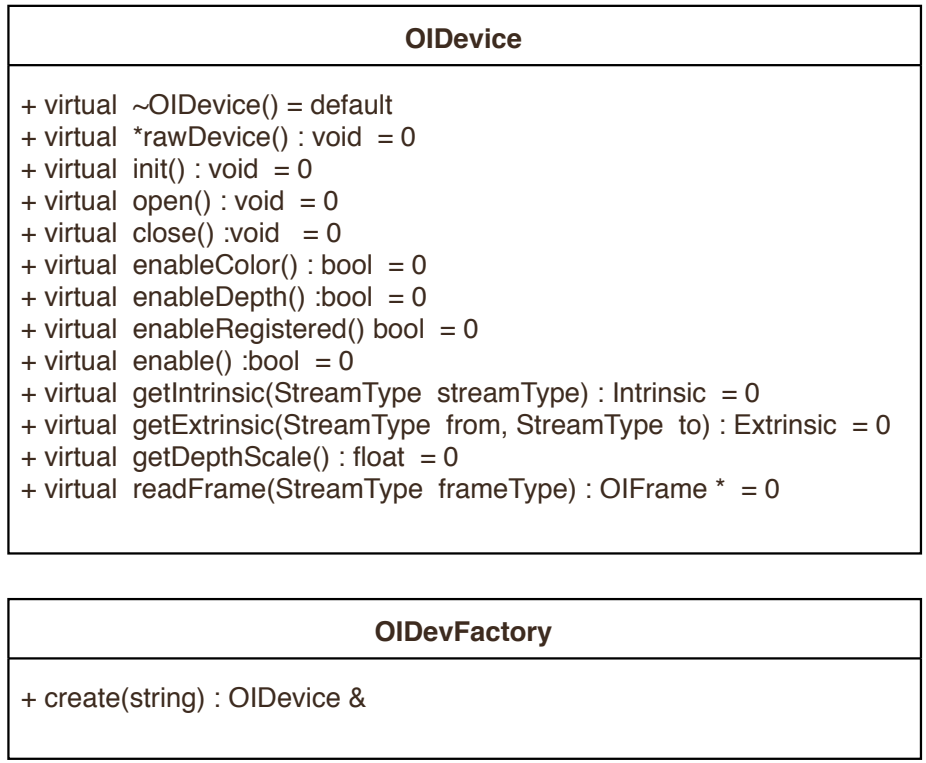
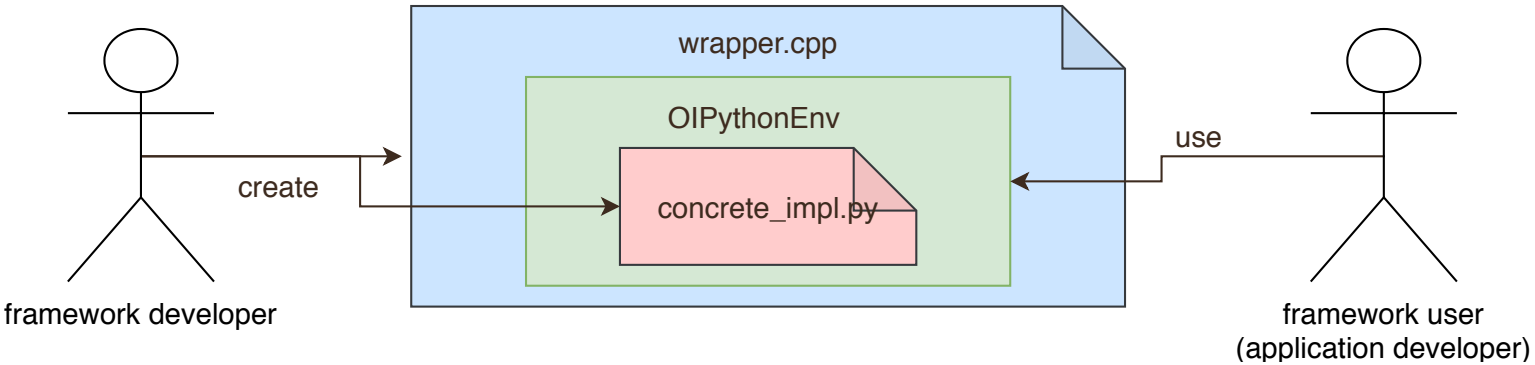


Figure 9: Design of device module within OpenISS core framework.

Design of Cross-language Module

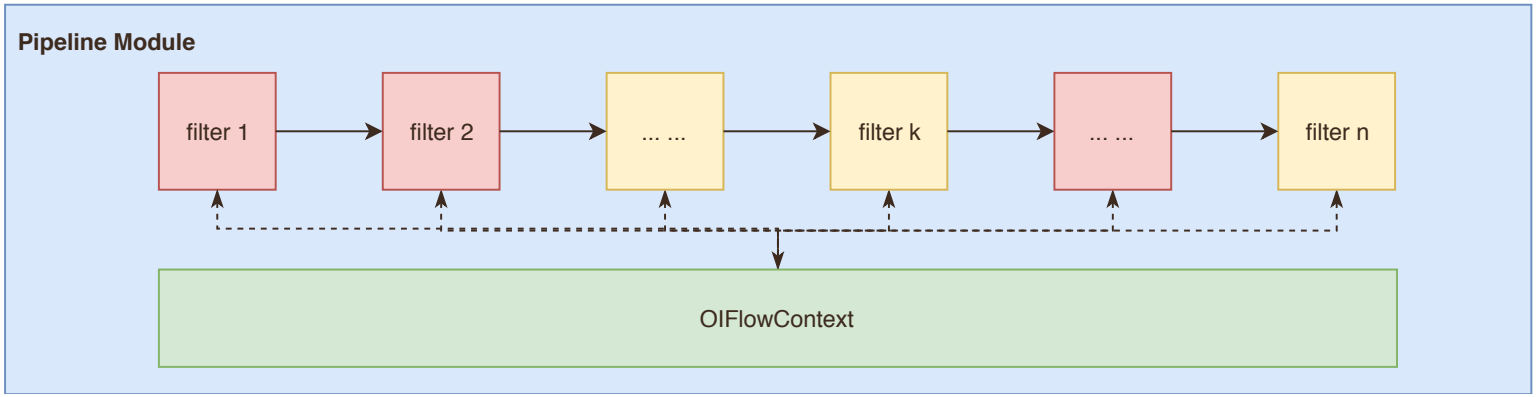


- most of the existing solutions are written in **Python**
- our framework solution is written in **C++**

OIPythonEnv
- modules std::unordered_map<char*, PyObject*>
+ OIPythonEnv() + ~OIPythonEnv() + initPyWorkingPath(std::vector<std::string> paths) : void + showWokingPath() : void + importPyModule(char *name) : void + getPyModule(char *name) : PyObject * + createPyInstance(char *moduleName, char *className, const char *format) : PyObject * + loadPyMethod(PyObject *callerName, char *funcName) : PyObject * + invokePyMethod(PyObject *callerName, PyObject *args) : PyObject *

Figure 10: Design of cross-language module within OpenISS core framework.

Design of Pipeline Module



- execution engine
- separate our functional modules as filters
- improve the reusability

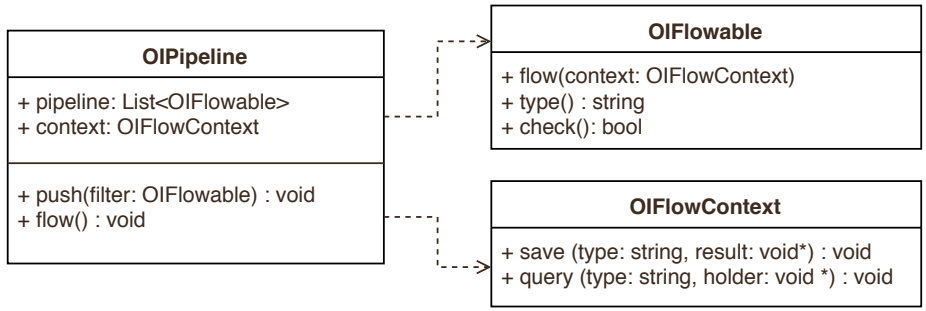


Figure 11: Design of pipeline module within OpenISS core framework.

Instantiation Overview

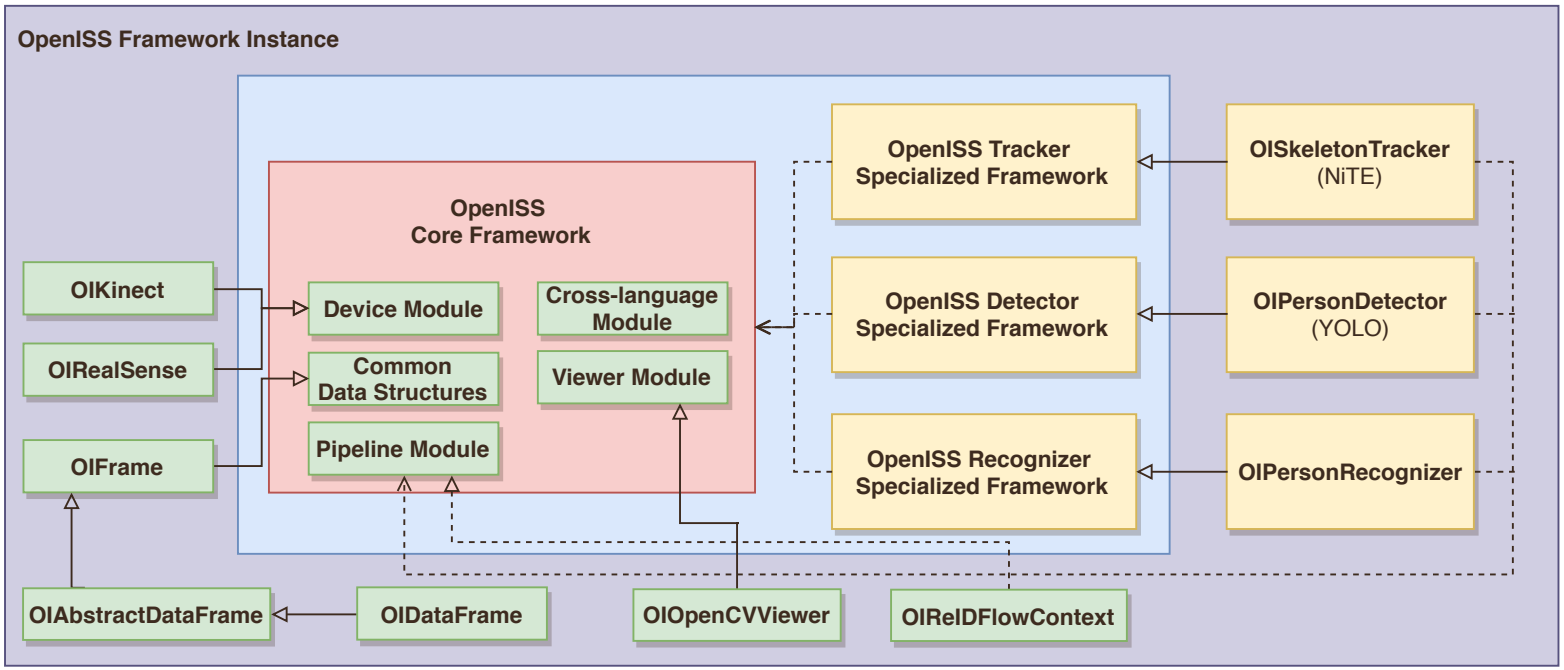


Figure 12: OpenISS framework instance architecture.

Device Module Instantiation

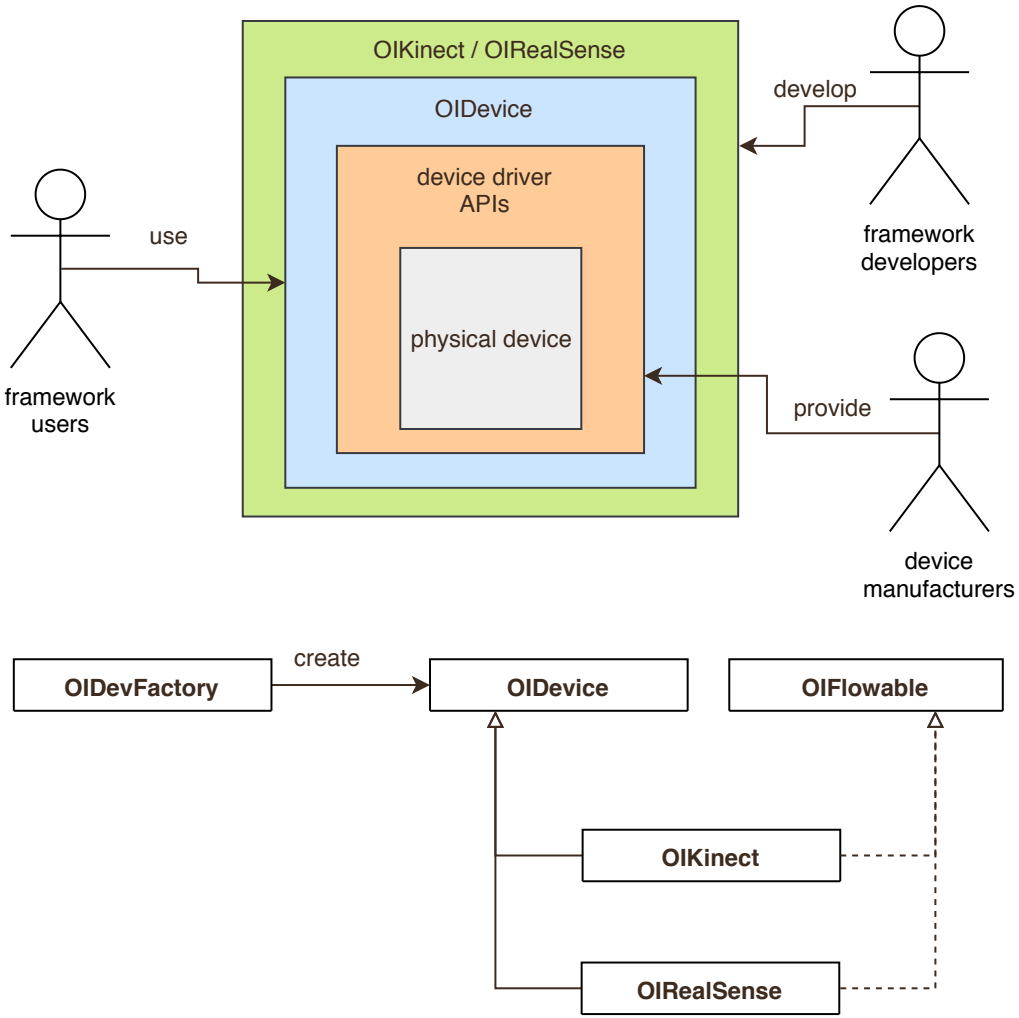


Figure 13: Instantiation of the device module in the core framework.

Tracker Instantiation

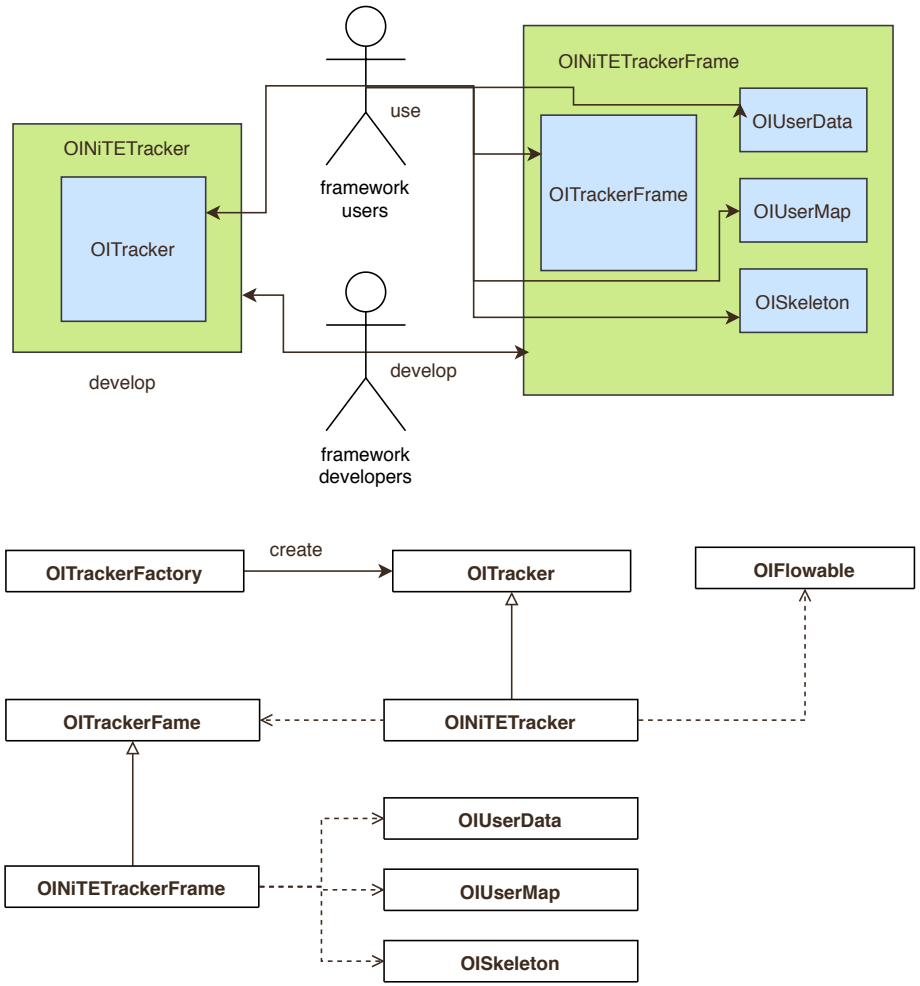


Figure 14: Instantiation of the tracker specialized framework.

Detector Instantiation

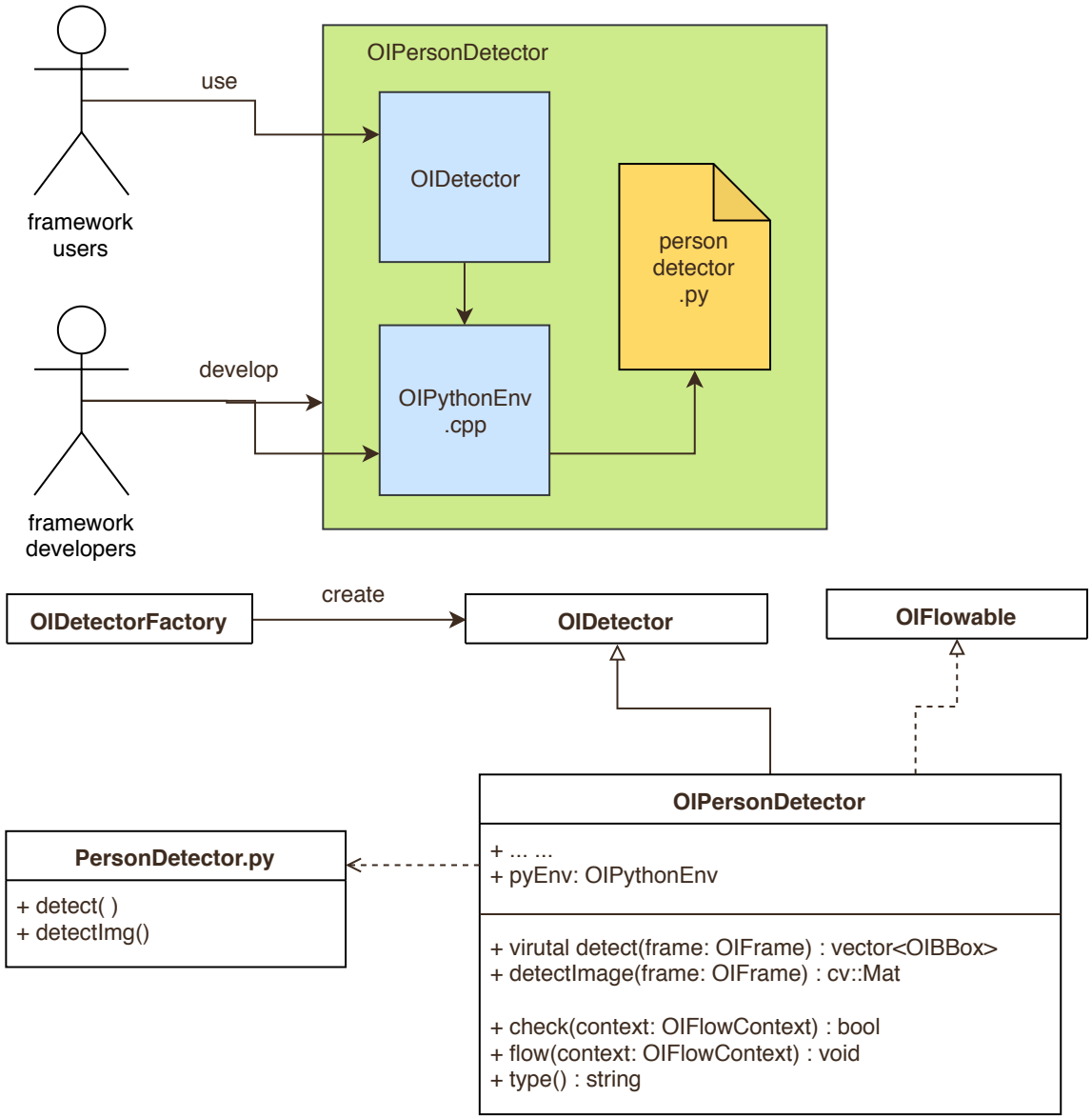


Figure 15: Instantiation of the detector specialized framework.

Recognizer Instantiation

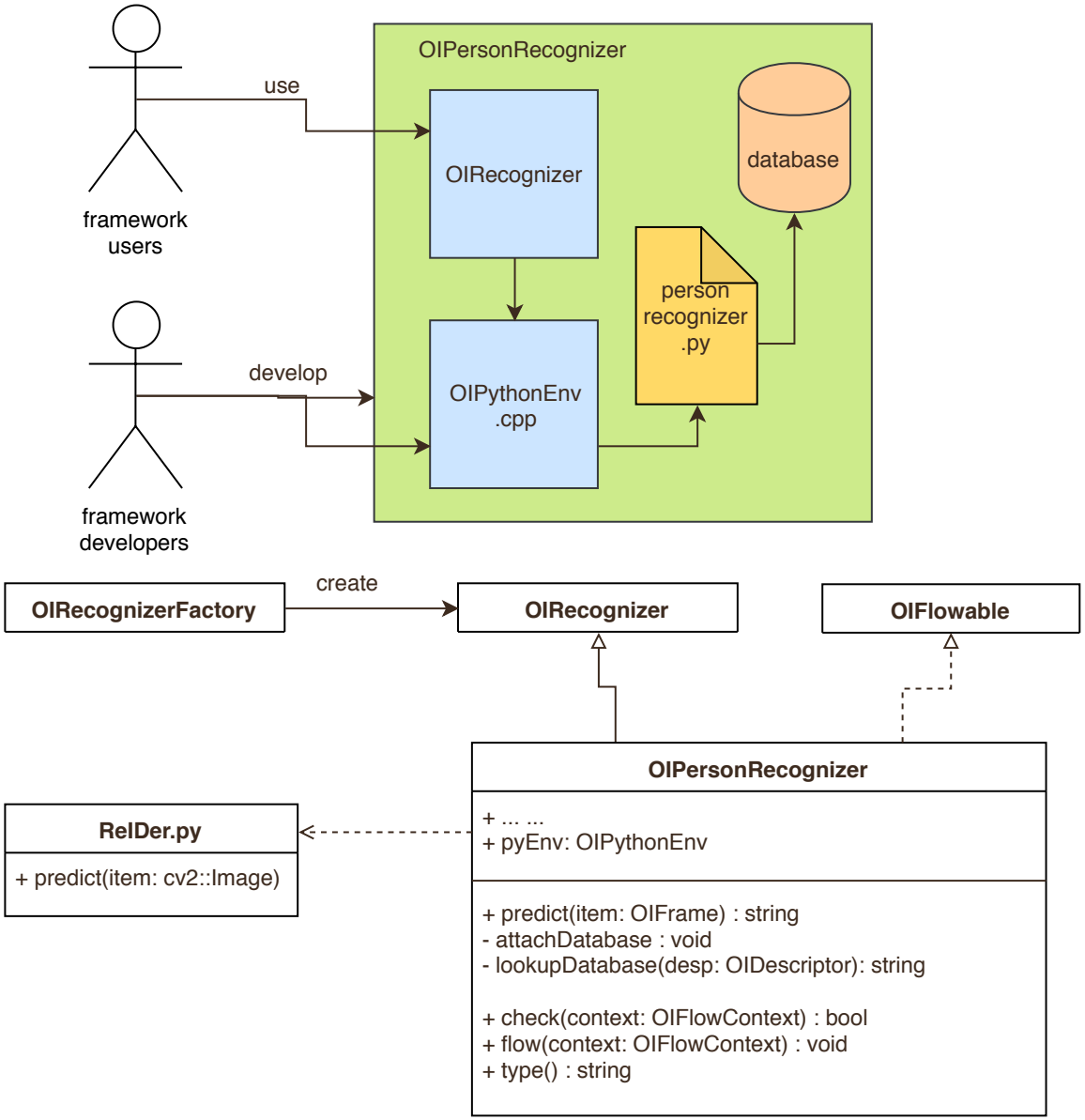


Figure 16: Instantiation of the recognizer specialized framework.

Design and Instantiation Summary

Framework implementation aspect:

- Design and instantiate 5 core modules.
- Design and instantiate 3 specialized frameworks.

Deep learning-based model aspect:

- Re-implement the YOLO v3 model and retrain it from an object detector to person detector with existing training facilities [2].
- Integrate the mAP benchmark facilities [3] for YOLO model.
- Re-implement the recognizer model from scratch on top of TensorFlow and Keras with reference [4] in Pytorch.

[2] https://github.com/qqwweee/keras-yolo3

[3] https://github.com/Cartucho/mAP

[4] https://github.com/michuanhaohao/reid-strong-baseline

Applications

Overview

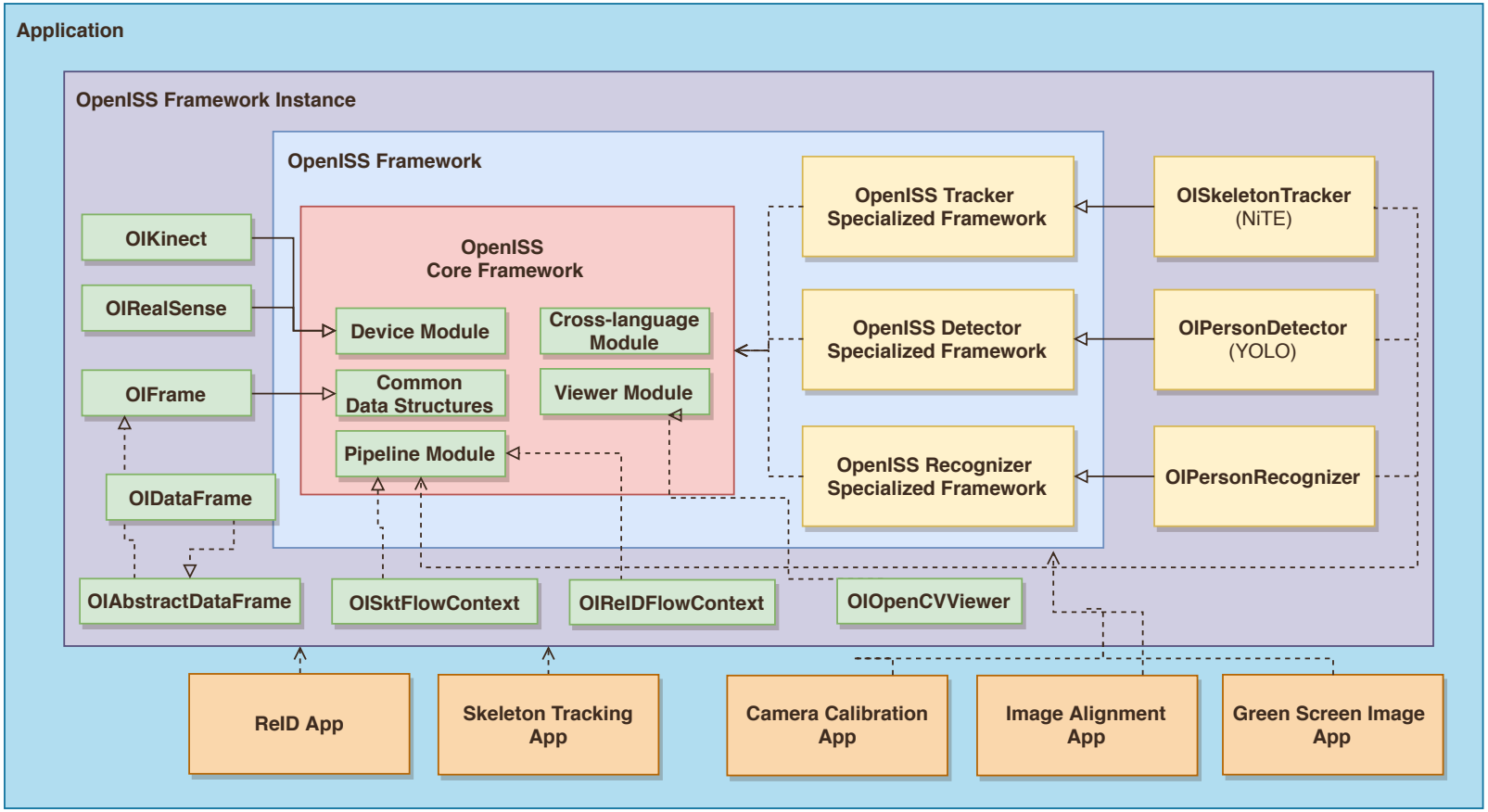


Figure 17: OpenISS framework instance architecture with applications.

ReID Application

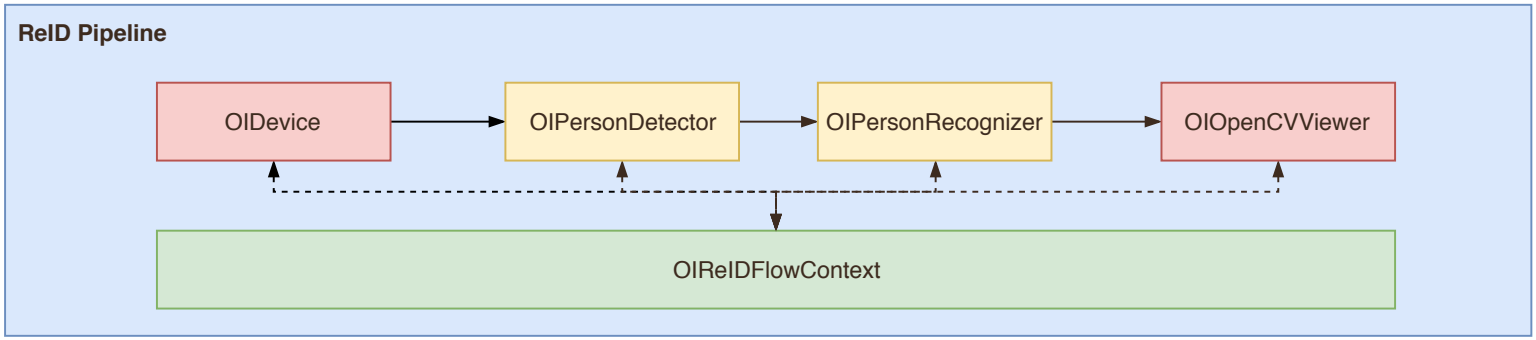


Figure 18: Skeleton tracking pipeline.

```
devF ← create a device factory
detF ← create a detector factory
recogF ← create a recognizer factory
db ← create a database for recognizer

noEcsPressed = True
device = devF.create("name of the device")
detector = detF.create("name of detector")
recognizer = recogF.create("name of recognizer")
recognizer.attachDatabase(db)

reidContext = new OIReIDFlowContext
reidPL = new OIPipeline(reidContext)
reidPL.push(dev)
reidPL.push(detector)
reidPL.push(recognizer)
reidPL.push(new OIOpenCVViewer)

while noEcsPressed do
    reidPL.flow(reidContext)
    if isEcsPressed then
        noEcsPressed = False
```

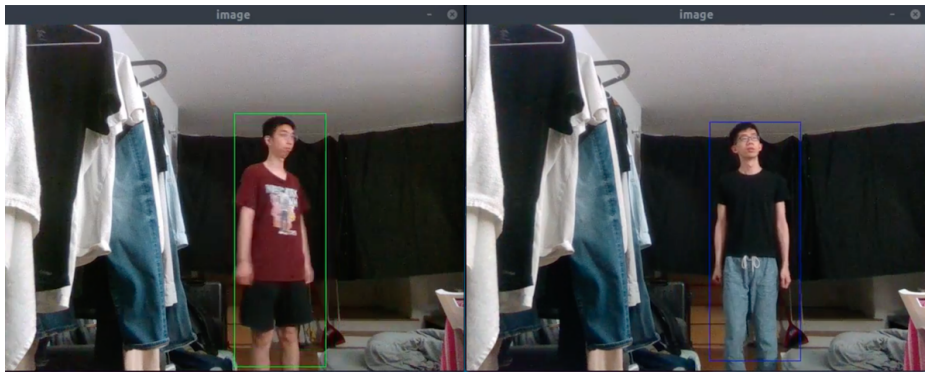


Figure 19: Person ReID application.

Skeleton Tracking Application

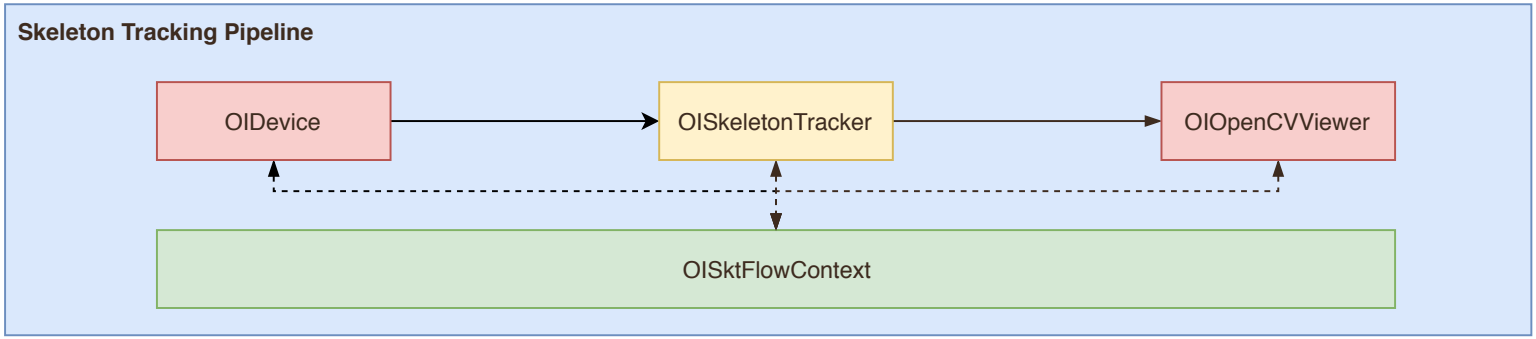


Figure 20: Skeleton tracking pipeline.

```
1  devF ← create a device factory
2  tkF ← create a tracker factory
3
4  noEcsPressed = True
5  device = devF.create("name of the device")
6  tracker = tkF.create("name of the tracker")
7
8  sktContext = new OISktFlowContext
9  sktPL = new OIPipeline(sktContext)
10 sktPL.push(dev)
11 sktPL.push(tracker)
12 sktPL.push(new OIOpenCVViewer)
13
14 while noEcsPressed do
15     sktPL.flow(sktContext)
16     if isEcsPressed then
17         noEcsPressed = False
```

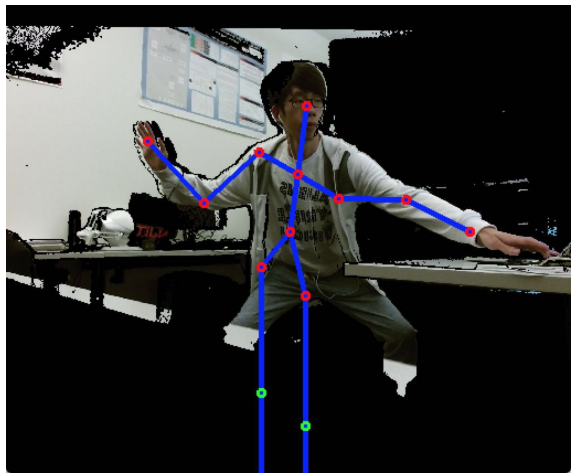


Figure 21: Skeleton tracking application.

Other Applications

Other applications:

- camera calibration
- image alignment
- green screen image

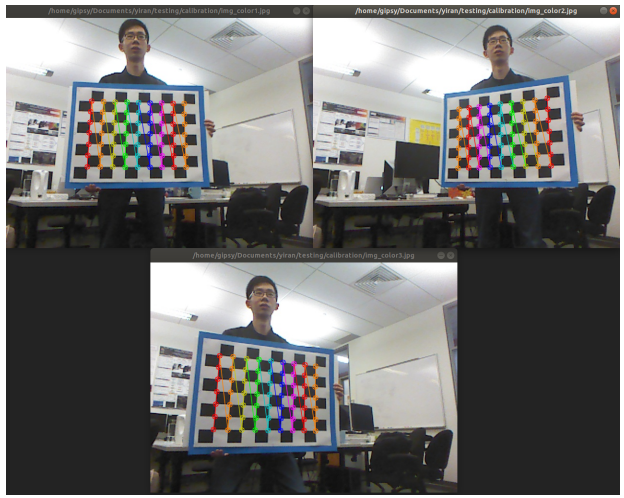


Figure 22: Camera calibration application.

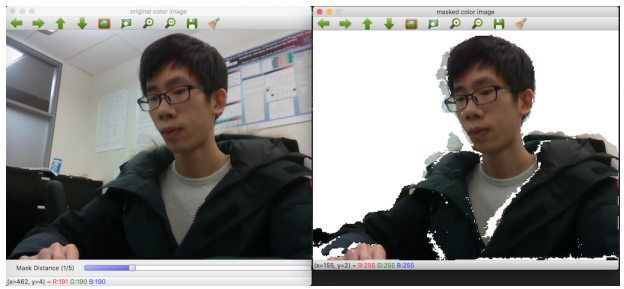


Figure 23: Green screen image application.

Evaluation

Framework Evaluation

- Functional Requirements
  - person re-identification
  - skeleton tracking
  - device abstraction
- Non-functional Requirements
  - real-time response
    - higher than 10 FPS, human perceive them as motion [1]
    - our solution achieve 12 FPS (acceptable)
  - accuracy
    - ranked 9th/32 on Market1501 dataset
    - ranked 7th/27 on DukeMTMC-reid dataset
  - extensibility
    - device module
    - tracker specialized framework
  - usability
    - 1 line of application code change for switching device
    - 36 lines of application code to enable person ReID feature

Environment Setting

Two environment settings:

- `setting 1`: a desktop machine
- `setting 2`: a powerful GPU cluster

Setting	Name	Amount	Device
1(Desktop Machine)	Memory	1	8 GB
	Processor	1	Intel Core i5-3470 CPU @3.20GHz × 4
	Graphics	1	GeForce GTX 1070 Ti (8GB memory)
	OS	N/A	Ubuntu 18.04.1 LTS 64-bit
2 (Virya Cluster)	Memory	1	400 GB
	Processor	1	72-core CPU
	Graphics	8	Tesla V100 (32 GB memory)
	OS	N/A	Scientific Linux

Table 1: Environment hardware specification.

Software	Setting 1	Setting 2
Python	3.6.7	3.6.8
TensorFlow	1.12.0	1.13.1
Keras	2.2.4	2.2.4
Keras-application	1.0.6	1.0.7
Keras-preprocessing	1.0.5	1.0.9

Table 2: Environment software specification.

Person Detector Evaluation

We retrain the following two models using the same facilities and validate them using the same dataset.

- Our person detector achieves **76.08%** mAP.
- YOLO v3 object detector achieves **81.95%** mAP on person category.

But,

- person detector can train faster
- no extra step will be needed to eliminate non-person result

Person Recognizer Evaluation

dataset	subset	# pids	# images	# cameras
Market1501	train	751	12936	6
	query	750	3368	6
	gallery	751	15913	6
CUHK03-NP	train	767	7368	2
	query	700	1400	2
	gallery	700	5327	2
Duke-MTMC	train	702	16522	8
	query	702	2228	8
	gallery	1110	17661	8

Table 3: Statistic for three popular ReID datasets

Since the names for these dataset are too long:

- M $\rightarrow$ Market1501 dataset
- C $\rightarrow$ CUHK03 dataset
- D $\rightarrow$ Duke-MTMC reid dataset

The dataset Market1501, CUHK03, and DukeMTMC-reid were published in [15, 7, 12] respectively.

Person Recognizer Evaluation

Our solution is ranked:

- 9^{th} place (out of 32) on M dataset

  0.904 CMC | 0.769 mAP
- 7^{th} place (out of 27) on D dataset

  0.840 CMC | 0.704 mAP

Rank	Method	CMC	mAP	Year
1	Auto-ReID	95.4	94.2	2019
2	DG-Net(RK)	95.4	92.49	2019
3	Parameter-Free Spatial Attention	94.7	91.7	2018
4	MGN	95.7	86.9	2018
6	DG-Net	94.8	86.0	2019
6	OSNet	94.8	84.9	2019
7	PCB + RPP	93.8	81.6	2017
8	PCB	92.3	77.4	2017
9	GLAD*	89.9	73.9	2017
10	Incremental Learning	89.3	71.8	2018

Table 4: State of the art result on Market1501 dataset **top 10 over total 32** [4].

Rank	Method	CMC	mAP	Year
1	Auto-ReID	91.4	89.2	2019
2	DG-Net(RK)	90.26	88.31	2019
3	Parameter-Free Spatial Attention	89.0	85.9	2018
4	MGN	88.7	78.4	2018
6	DG-Net	86.6	74.8	2019
6	OSNet	88.6	73.5	2019
7	PCB (RPP)	83.3	69.2	2017
8	PCB (UP)	81.8	66.1	2017
9	SVDNet + Random Erasing	79.3	62.4	2017
10	Incremental Learning	80.0	60.2	2018

Table 5: State of the art result on DukeMCMT-reid dataset **top 10 over total 27** [3].

Person Recognizer Evaluation

We have performed four comparisons for our recognizer model:

- Comparison between two different triplet loss functions.
- Comparison between two environment settings.
- In-dataset validation.
- Cross-dataset validation.

Comparison between two different triplet loss functions

- batch all: all samples within a batch will contribute to the loss
- batch hard: only the hardest samples will contribute to the loss

$$L_{BA} = \sum_{i=1}^{P}\sum_{a=1}^{K}\sum_{\substack{p=1\\p\neq a}}^{K}\sum_{\substack{j=1\\j\neq i}}^{P}\sum_{n=1}^{K}[m + d_{j,a,n}^{i,a,p}]_+ \tag{1}$$

$$L_{BH} = \sum_{i=1}^{P}\sum_{a=1}^{K}[m + \max_{p=1\ldots K} D(f_\theta(x_a^i), f_\theta(x_p^i)) - \min_{\substack{j=1\ldots P\\n=1\ldots K\\j\neq i}} D(f_\theta(x_a^i), f_\theta(x_n^j))]_+ \tag{2}$$

Triplet Loss	CMC (top 5)	mAP
triplet all	[**0.904**, 0.941, 0.952, 0.964, 0.969]	**0.769**
triplet hard	[**0.878**, 0.924, 0.946, 0.955, 0.963]	**0.715**

Table 6: Validation result on the model trained with Market1501 dataset guided by two different loss functions.

These two loss functions were proposed in [6]

Person Recognizer Evaluation

Comparison between two environment settings

Triplet Loss	Metric	Setting 1	Setting 2
triplet all	CMC (top 2)	[0.904, 0.947]	[0.904, 0.941]
	mAP	0.774	0.769
	training time (min)	**244**	**133**
triplet hard	CMC (top 2)	[0.871, 0.923]	[0.878, 0.924]
	mAP	0.705	0.709
	training time (min)	**237**	**134**

Table 7: Training result with two different settings on the same Market1501 dataset.

Person Recognizer Evaluation

In-dataset validation

Dataset \Metrics	CMC (top 5)	mAP
Market1501	[0.904, 0.941, 0.952, 0.964, 0.969]	0.769
CUHK03	[0.502, 0.586, 0.641, 0.683, 0.721]	0.500
DukeMTMC-reID	[0.840, 0.884, 0.904, 0.919, 0.926]	0.704

Table 8: In dataset validation of our person retrieval model guided by triplet all loss on `setting 2`.

Cross-dataset validation

Metrics / Dataset	M $\rightarrow$ D	D $\rightarrow$ M
CMC	[0.272, 0.339, 0.379, 0.404, 0.420]	[0.474, 0.548, 0.587, 0.618, 0.643]
mAP	0.150	0.211

Table 9: Cross-dataset validation result between Market1501 and DukeMTMC-reID dataset on `setting 2`. M $\rightarrow$ D represents the model trained on Market1501 dataset and tested on DukeMTMC-reID dataset.

Conclusion

Conclusion

- Propose a general framework solution.

  detection | recognition | tracking

- Design and implement a device module for depth cameras encapsulation.

  Kinect v1 | Kinect v2 | RealSense D435

- Design and implement five applications to demonstrate the capability of our solution.

  ReID app. | skeleton tracking app. | camera calibration, image alignment, green image apps.

- Retrain the YOLO object detection model to become a person detection model
- Re-implement a person re-identification network combined with identification model and triplet model ranked 9th and 7th respectively for two most popular datasets.

Limitations

- Our ReID application currently requires us to prepare an image database in advance.
- Our green screen application currently requires the user to select the filtering distance.
- Our camera calibration application now can only calibrate normal cameras but not IR cameras.
- Our model likely will fail under some special environments.
  - no or only dim lighting condition for recognizer
  - tracking object which moves in a extremely high speed

Future work

- real-time machine learning for person tracking
- Java API wrapper
- more devices support
- integrate more detection and retrieval algorithms
- integrate with the live artistic show

Acknowledgment

I would like to offer my sincere gratitude to:

- Co-supervisors: Drs. Joey Paquet and Serguei Mokhov;
- Examining Committee: Drs. Jingqiu Yang, Nematollaah Shiri, and Aiman Hanna;
- Labmates: Yiran Shen, Jashanjot Singh, Jyotsana Gupta;
- Parents: Qin Luo and Yong Lai;
- Friends: Yixin Yao, Chen Feng, Jing Yang, Xingjian Zhang, Outong Li, Bo Li, Qinwei Luo, and Jingye Hou.
- Friends: Xuyi Huang, Zijian Kong, and Haien Long.
- The audience;

References I


Frame rate.
https://en.wikipedia.org/wiki/Frame_rate.
Accessed: 2019-07-23.


Software framework.
https://en.wikipedia.org/wiki/Software_framework.
Accessed: 2019-06-22.


State of the art result on dukemcmt-reid dataset.
https://paperswithcode.com/sota/person-re-identification-on-dukemtmc-reid.

Accessed: 2019-06-29.

References II

State of the art result on market1501 dataset.
`https://paperswithcode.com/sota/person-re-identification-on-market-1501`.

Accessed: 2019-06-29.

Fan, X., Jiang, W., Luo, H., and Fei, M.
Spherereid: Deep hypersphere manifold embedding for person re-identification.
CoRR abs/1807.00537 (2018).

Hermans, A., Beyer, L., and Leibe, B.
In defense of the triplet loss for person re-identification.
CoRR abs/1703.07737 (2017).

References III

Li, W., Zhao, R., Xiao, T., and Wang, X.
Deepreid: Deep filter pairing neural network for person re-identification.
In *CVPR* (2014).

Liu, L., Ouyang, W., Wang, X., Fieguth, P. W., Chen, J., Liu, X., and Pietikäinen, M.
Deep learning for generic object detection: A survey.
CoRR abs/1809.02165 (2018).

Luo, H., Gu, Y., Liao, X., Lai, S., and Jiang, W.

Bag of tricks and A strong baseline for deep person re-identification.
CoRR abs/1903.07071 (2019).

References IV

Mokhov, S. A., Song, M., Chilkaka, S., Das, Z., Zhang, J., Llewellyn, J., and Mudur, S. P.
Agile forward-reverse requirements elicitation as a creative design process: a case study of llimitable Space System v2.
Journal of Integrated Design and Process Science (2015–2016).
Under review.

Pree, W.
Meta patterns - a means for capturing the essentials of reusable object-oriented design.
In *ECOOP* (1994).

References V

Ristani, E., Solera, F., Zou, R., Cucchiara, R., and Tomasi, C.
Performance measures and a data set for multi-target, multi-camera tracking.
In *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking* (2016).

Schroff, F., Kalenichenko, D., and Philbin, J.

Facenet: A unified embedding for face recognition and clustering.
CoRR abs/1503.03832 (2015).

References VI

Wu, D., Zheng, S.-J., Zhang, X.-P., Yuan, C.-A., Cheng, F., Zhao, Y., Lin, Y.-J., Zhao, Z.-Q., Jiang, Y.-L., and Huang, D.-S.
Deep learning-based methods for person re-identification: A comprehensive review.
Neurocomputing 337 (2019), 354 – 371.

Zheng, L., Shen, L., Tian, L., Wang, S., Wang, J., and Tian, Q.
Scalable person re-identification: A benchmark.
In *Computer Vision, IEEE International Conference on* (2015).

Appendices

Non-maximum Suppression

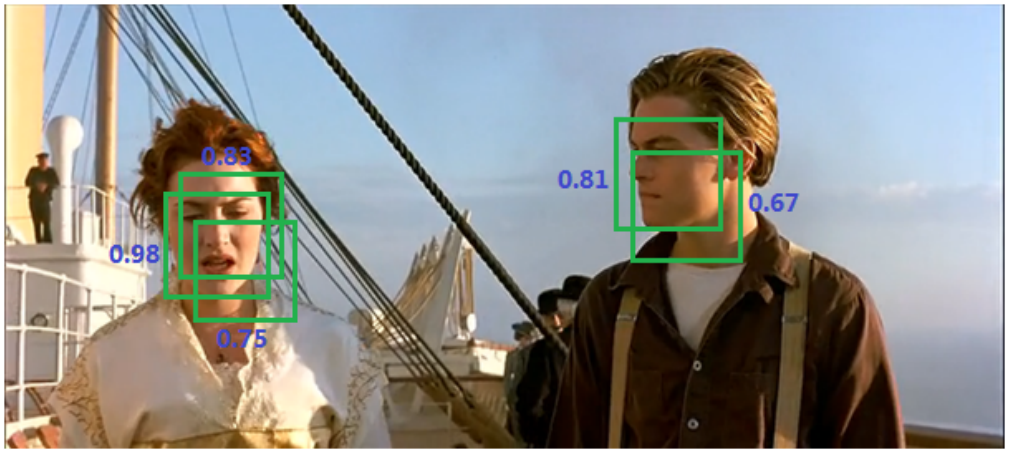


Figure 24: Before NMS

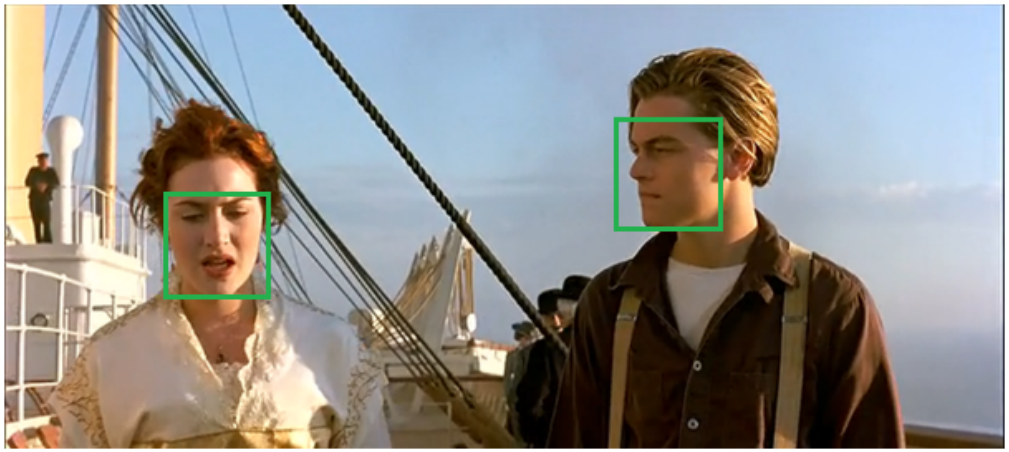
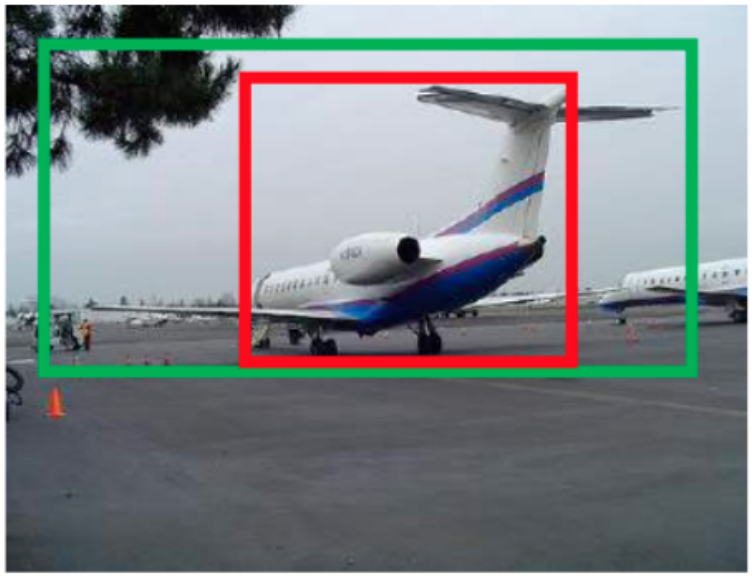


Figure 25: After NMS

Bounding-box Regression

Green box: the ground truth bounding box;
Red box: selective search region proposal;



Bounding-box regression is used to adjust the red box to make it as close as possible to the green one.

YOLO v3 Model

- Backbone network: Darknet-53;
- Anchor: total 9, but 3 for each scale;
- Logistic vs. Softmax: support multi-label classification;
- Objectness score for each box instead of each location;

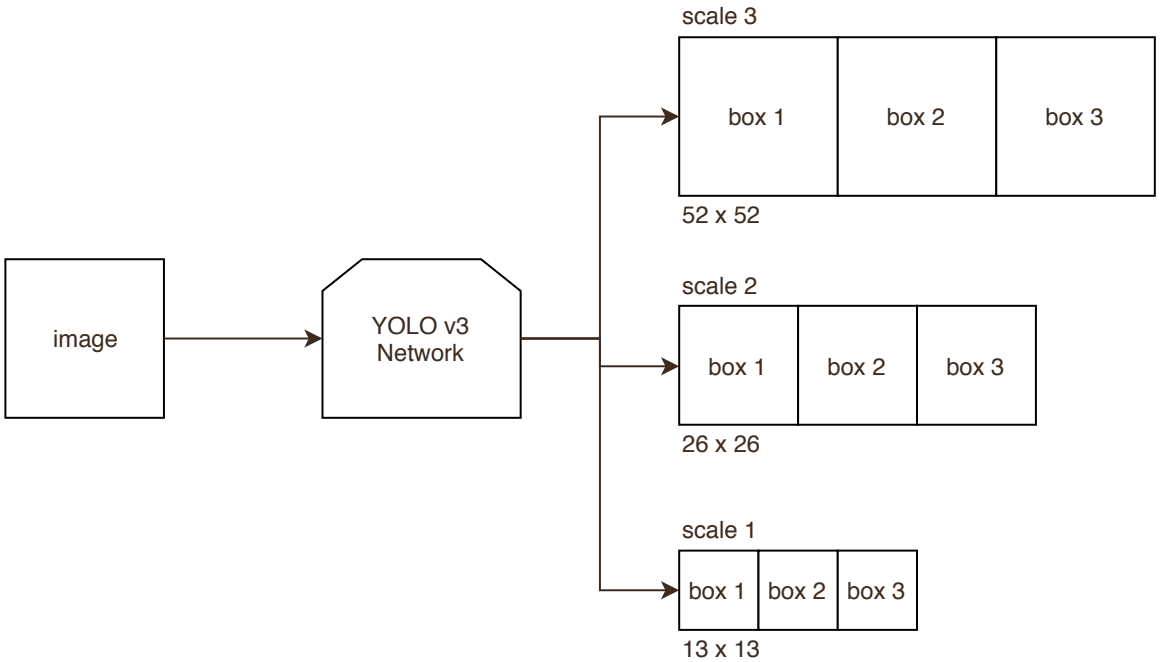


Figure 26: YOLO v3 process flow.

Detector Instantiation

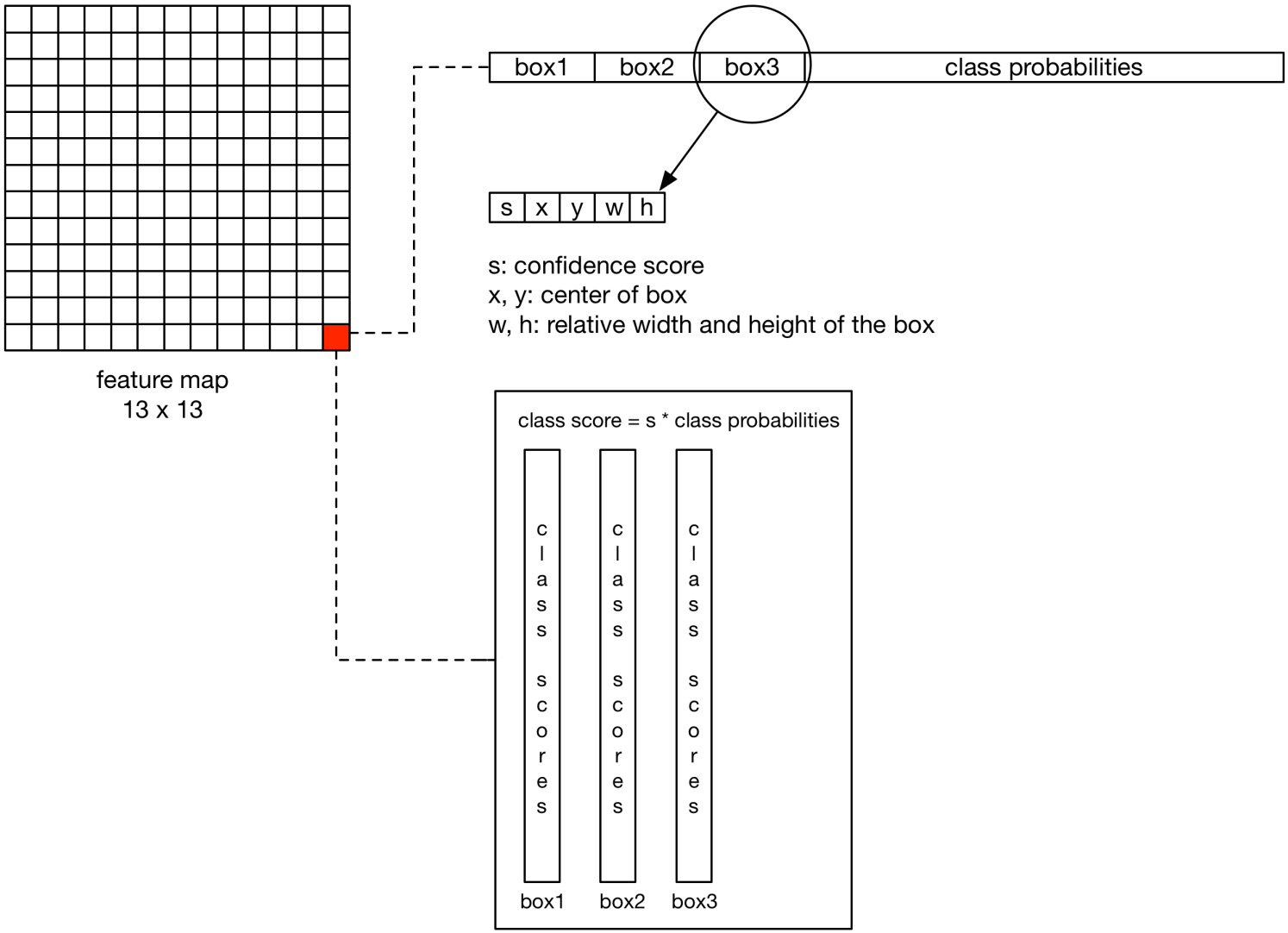


Figure 27: Calculation process of each cell in the feature map.

Detector Instantiation

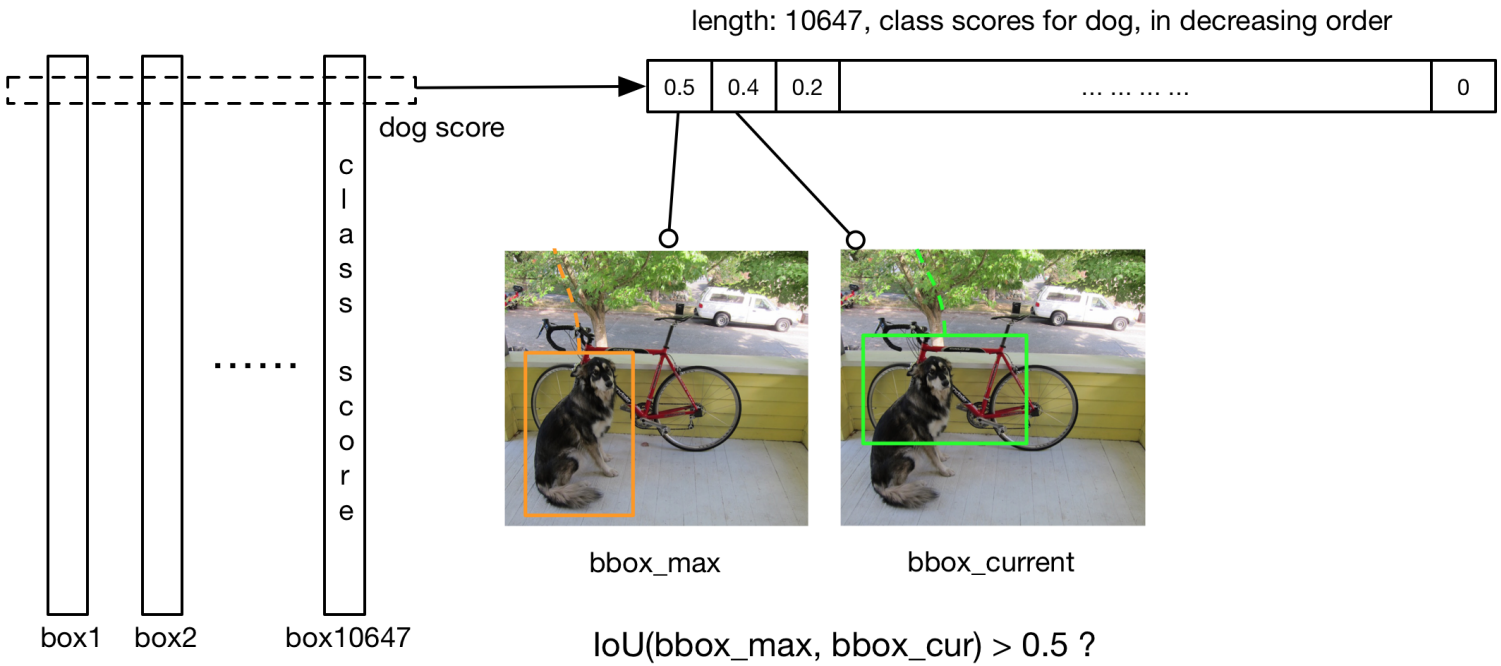
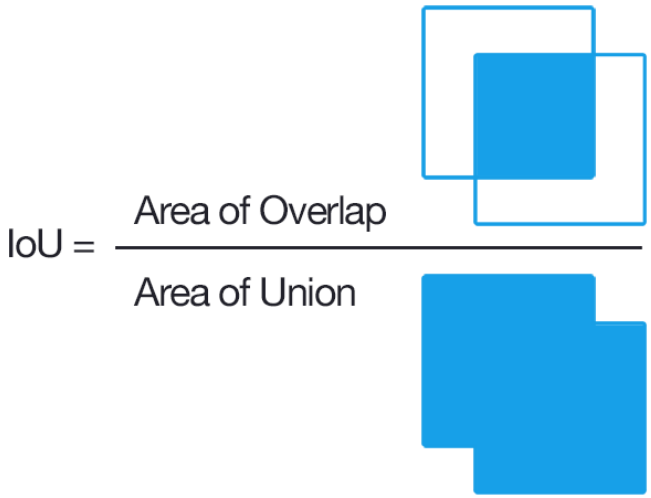


Figure 28: Non-maximum suppression process.

Intersection over Union Overlap

Intersection over Union Overlap (IoU) is used to measure how close a given bounding box is to another box that is independent of the units used (pixels, etc).



Person Detector Evaluation

Possible results for one detection operation:

- True positive, a correct detection where $IOU \geq threshold$.
- False positive, a incorrect detection where $IOU < threshold$.
- True negative, does not apply.
- False negative, a ground truth not detected.

Precision and Recall

Precision is a fraction of relevant instances among the retrieved instances which can be used to measures how accurate is your prediction.

Recall is a fraction of relevant instances that have been retrieved over the total amount of relevant instances. It can be used to measure how good you find all the positives.

$$precision = \frac{\#\ \text{true positive}}{\#\ \text{true positive} + \#\ \text{false positive}}$$

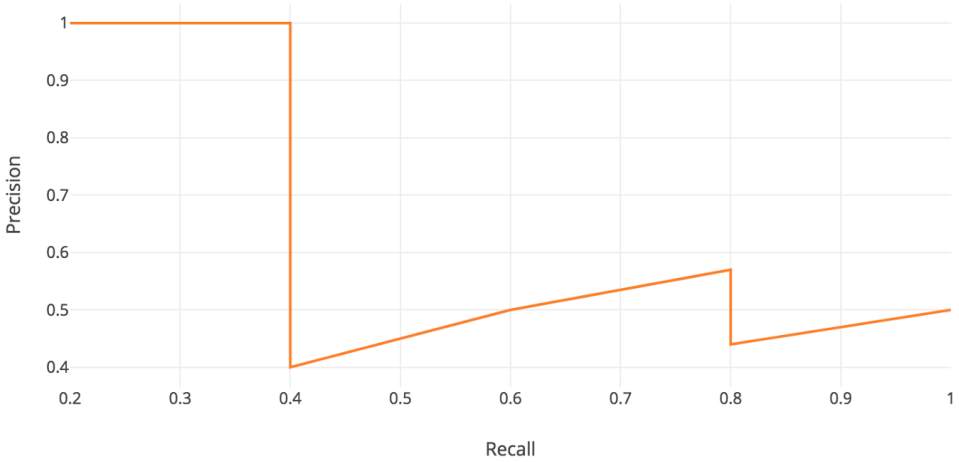
$$recall = \frac{\#\ \text{true positive}}{\#\ \text{true positive} + \#\ \text{false negative}}$$

Person Detector Evaluation

With precision and recall in hand, we can construct a Precision-Recall curve (P-R curve) which will be used to calculate our metric. The curve construction process can be described as the following:

1. Collect all the predictions that make for a particular class of objects. Rank them in decreasing order according to the confidence score given by the model.
2. Compare theses predictions with ground truth to see if they are correct or not.
3. Calculate the precision and recall using the given formula for each prediction in the ranked list from top to bottom.

Person Detector Evaluation



Rank	Result	precision	recall
1	true	1.0	0.2
2	true	1.0	0.4
3	false	0.67	0.4
4	false	0.5	0.4
5	false	0.4	0.4
6	true	0.5	0.6
7	true	0.57	0.8
8	false	0.5	0.8
9	false	0.44	0.8
10	true	0.5	1.0

Figure 29: An example of P-R curve. Left is the curve itself and right is the data used to plot this curve, assuming that there is a total of five positives in the data.

Person Detector Evaluation

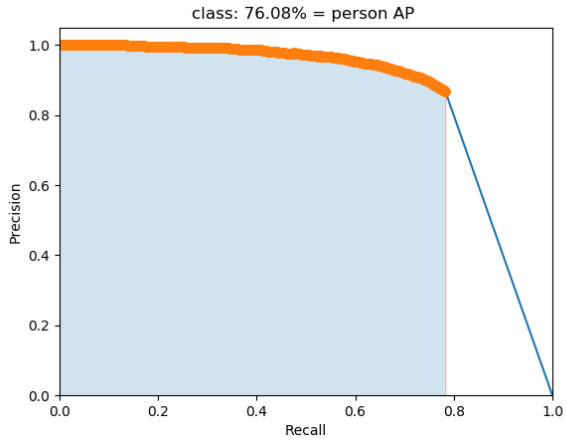


Figure 30: Person detection model's Precision-Recall curve on validation set.

Figure 31: Person detection result on validation set.

Person Detector Evaluation

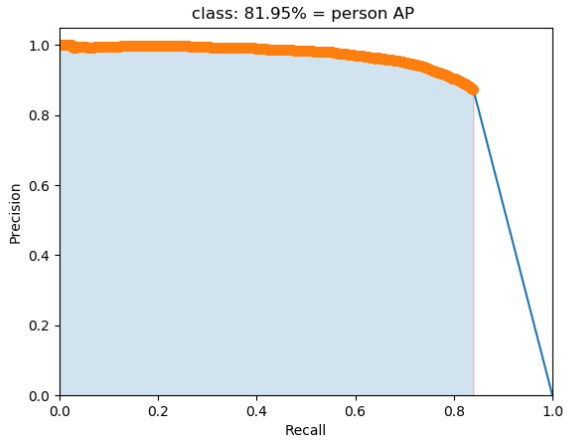


Figure 32: Object detection (with 20 classes supported) model's Precision-Recall curve on validation set.

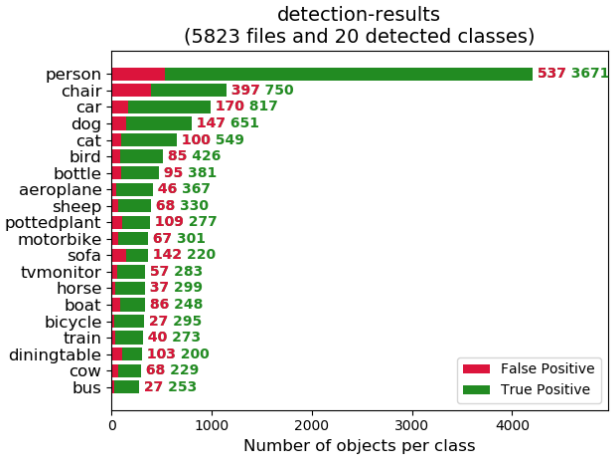


Figure 33: Object detection result on validation set.

Recognizer Network Architecture

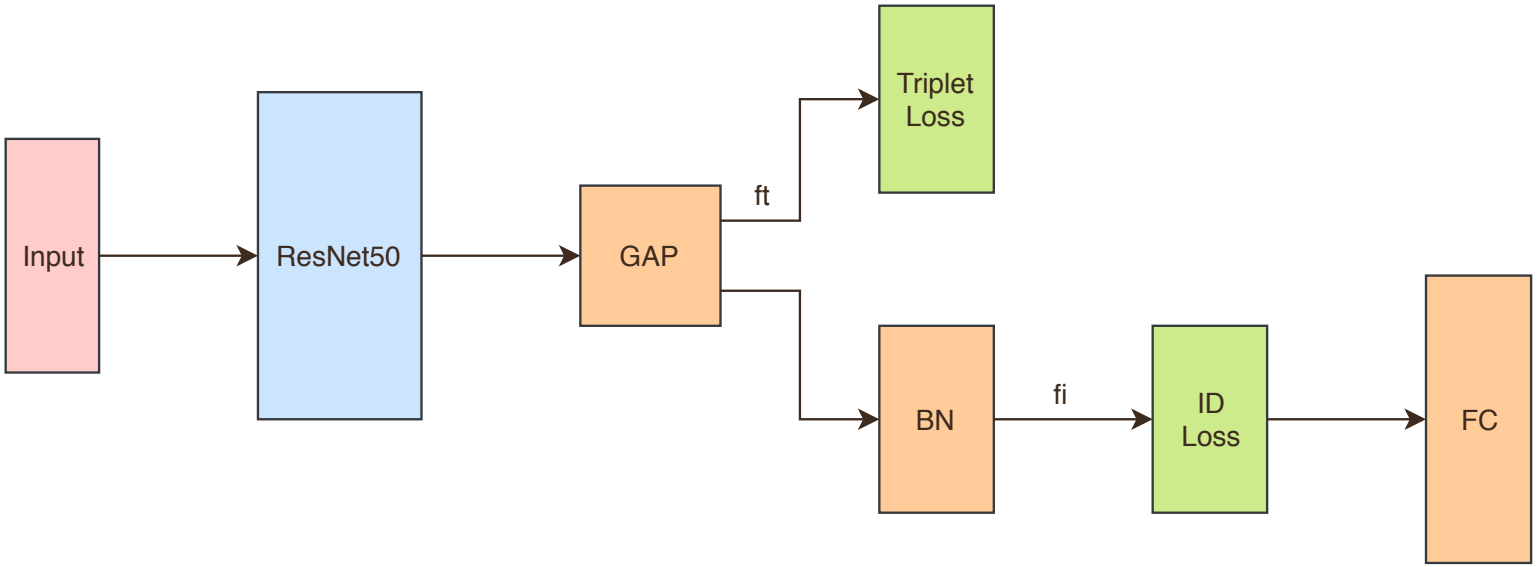


Figure 34: Implemented recognizer network architecture. GAP: global average pooling layer, BN: batch normalization layer, FC: fully connected layer. f_t: features used to calculate triplet loss, f_i: features used for inference. The pink color represents input, blue means backbone network, orange represents a special layer and green means loss function layer.

The architecture was proposed in [9].

Recognizer Training Workflow

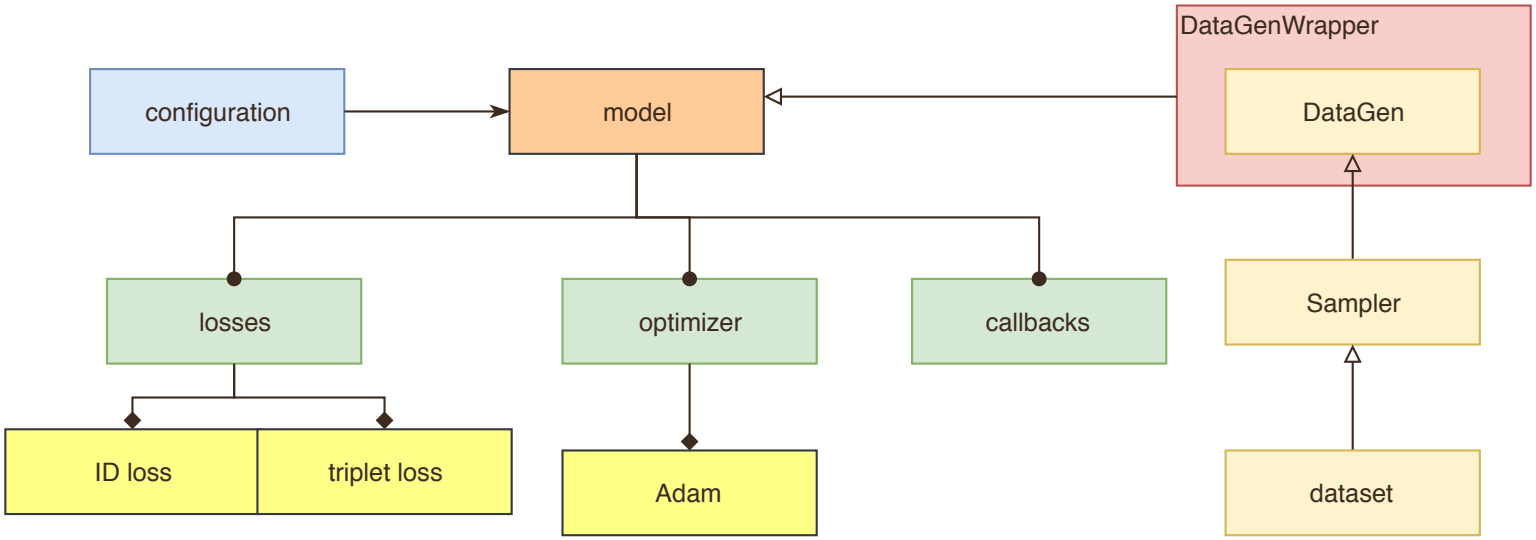


Figure 35: The workflow of the ReID training program

Recognizer Training

- batch size: $16 \times 4 = 64$
- warm up learning rate strategy [5]
- triplet loss + ID loss
- Adam optimizer
- total training epochs 120 [9]

$$\mathrm{lr}(t) = \begin{cases} 3.5 \times 10^{-5} \times \frac{t}{10} & \text{if } t \leq 10 \\ 3.5 \times 10^{-4} & \text{if } 10 < t \leq 40 \\ 3.5 \times 10^{-5} & \text{if } 40 < t \leq 70 \\ 3.5 \times 10^{-6} & \text{if } 70 < t \leq 120 \end{cases}$$

Recognizer Training Visualization

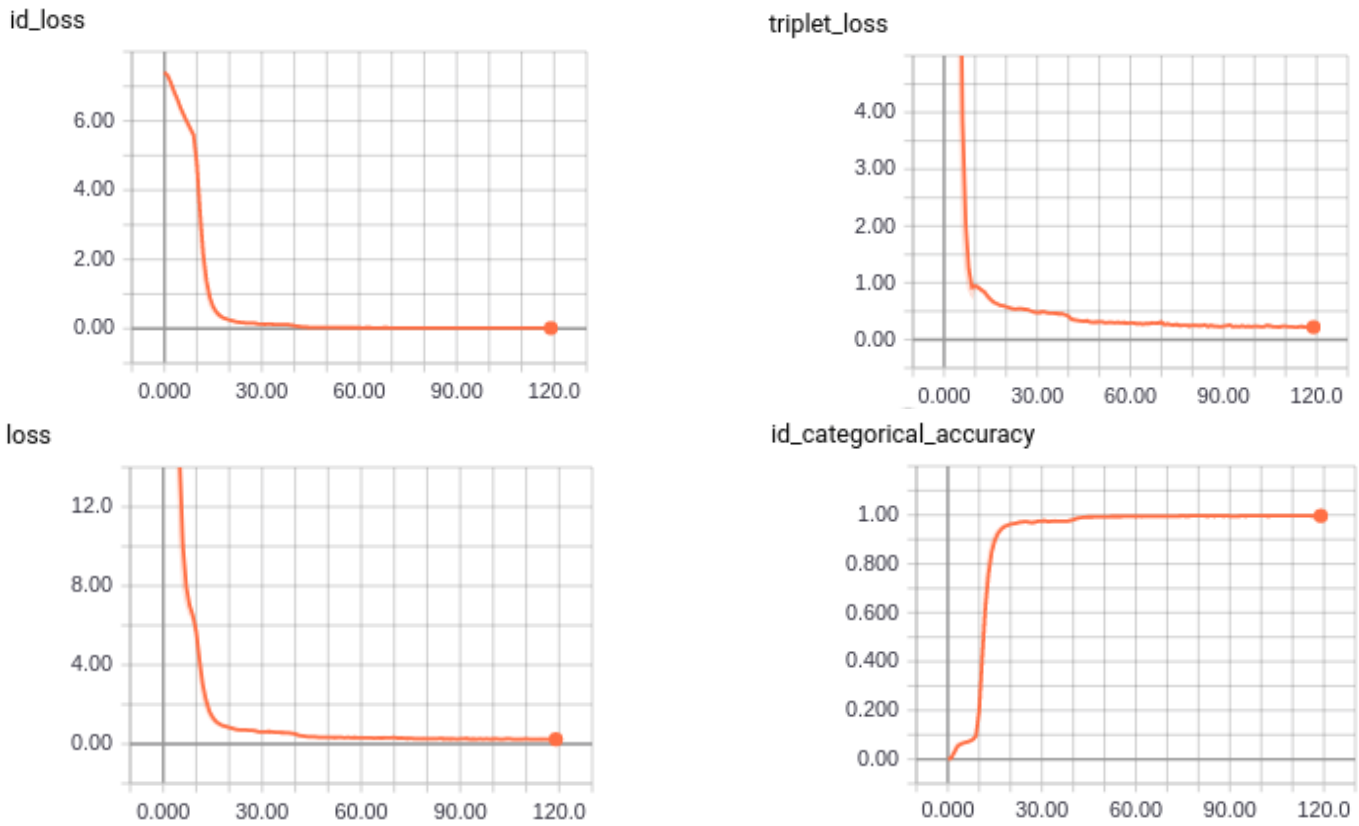


Figure 36: Training visualization diagram. Upper left: training identification loss curve. Upper right: training triplet hard loss curve. Lower left: total training loss curve. Lower right: training classification accuracy.

These figures are produced by the TensorBoard program.