

# 基于TensorRT的DeepSort目标跟踪的C++实现

TensorRT, Sort, DeepSort, ReID

Xu Jing

青岛美迪康数字工程(研发部)  
AI图像算法研发工程师

2021-12

① 目标跟踪算法详解(Sort,DeepSort)

② DeepSort中的ReID算法

③ 解耦合的目标检测算法

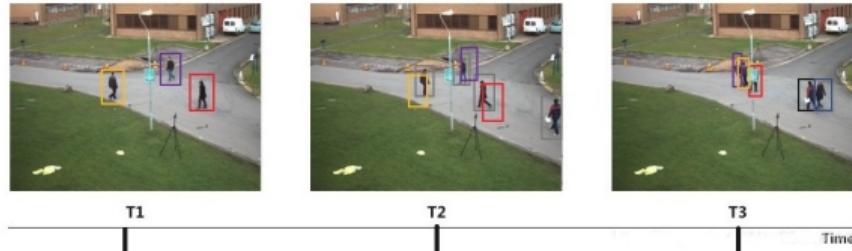
④ 工程实现与测试

① 目标跟踪算法详解(Sort,DeepSort)

② DeepSort中的ReID算法

③ 解耦合的目标检测算法

④ 工程实现与测试



T1时跟踪了3个目标，3个ID  
这3个ID是通过Kalman filter  
更新位置的，更新是借助于T0  
帧的Kalman filter的帧预测结  
果

T2灰色的是YOLO v5x预测的box,通过IoU+匈牙利算法选择最优匹配, IoU阈值官方给了0.3,这样会产生三种结果!

1. 没有用到图像像素特征，仅将bbox的位置和大小用于估计和数据关联；
2. 长，短期遮挡问题被忽略，这种情况在我们的场景下也会很少发生 
3. 忽略检测带来的错误 (X) —

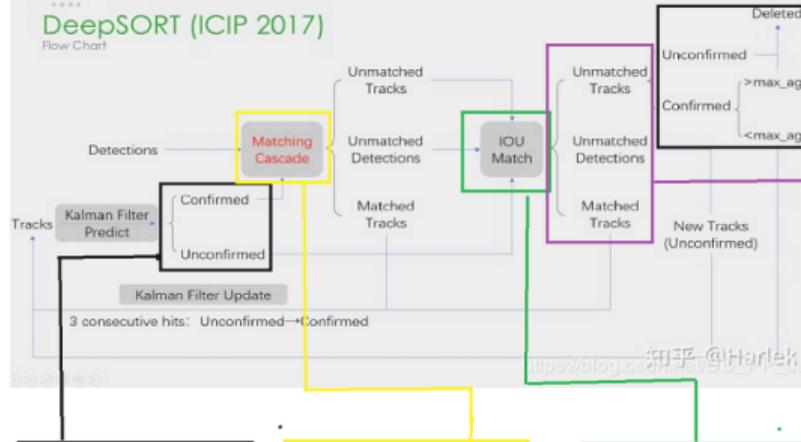
1. 预测结果 (Kalman filter) 与 YOLOv5x 检测结果匹配 (紫色预测, 灰色检测) 则会更新 Kalman filter, 并预测下一帧结果。

2. 检测结果与任何的预测结果都不匹配，如新来的那个人灰色box，此时认为是新目标进入场景，产生新的ID

用这种方式减少检测器的误识别基本不起作用

Kalman濾波本身就  
是一種Tracker!

- ① SORT全称为Simple Online And Realtime Tracking, 对于现在的多目标跟踪，更多依赖的是其检测性能的好坏，也就是说通过改变检测器可以提高18.9%，本篇SORT算法尽管只是把普通的算法如卡尔曼滤波（Kalman Filter）和匈牙利算法(Hungarian algorithm)结合到一起，却可以匹配2016年的SOTA算法，且速度可以达到260Hz，比前者快了20倍。
- ② 作者使用了Faster RCNN来进行模型的检测，并使用Kalman滤波预测状态，基于检测框位置和IOU的匈牙利算法，使得算法有很高的效率，但是这么频繁的ID Switch，在实际应用中就失去了跟踪的价值了！



DeepSORT的ID switch 下降了45%，达到了2017年的SORT;但是FP很多，也就是说如果想通过DeepSORT减少detector的误识别是不可行的，依然依赖于detector的精度，速度是20HZ,这可比SORT慢多了！主要时间花费在RelD的特征提取上！

将 Detection 和 Track 进行匹配，所以出现几种情况

1. Detection 和 Track 匹配，也就是 **Matched Tracks**。普通连续跟踪的目标都属于这种情况，前两帧都有目标，能够匹配上。
2. Detection 没有找到匹配的 Track，也就是 **Unmatched Detections**。图像中突然出现新的目标的时候，Detection 无法在之前的 Track 找到匹配的目标。
3. Track 没有找到匹配的 Detection，也就是 **Unmatched Tracks**。连续追踪的目标超出图像区域，Track 无法与当前任意一个 Detection 匹配。
4. 以上没有涉及一种特殊的情况，就是两个目标遮挡的情况。刚刚被遮挡的目标的 Track 也无法匹配 Detection。目标暂时从图像中消失，之后被遮挡目标再次出现的时候，应该尽量让被遮挡目标分配的 ID 不发生变动，减少 ID Switch 出现的次数。这就需要用到级联匹配了。

**状态估计**

1. 当前帧距离上次匹配成功帧数的差值，该变量在 Kalman Filter predict 时递增，Track 和 detect 关联时重置为0

2. 超过最大值A, Track ID 被删除，跟丢了，A=70(可调)
3. detect 和 track 没匹配上，新跟踪目标来了

但是还取决于接下来连续3帧是否成功匹配到 detect, 才能确定我要跟踪这个ID

4.  $n\_init=3$ , 如果没匹配上 detect, 删除该ID

**匹配问题**  
解决 Track 和 detect 的对应问题，匈牙利算法，最优二分匹配，需要一个损失矩阵 (Cost Matrix)

**匹配的最后阶段对unconfirmed和age=1的未匹配的轨迹进行基于IoU的匹配(同SORT)，可以缓解因为突变或部分遮挡导致的较大变化！！！**

1. 马氏距离 (基于协方差阵的，区别于欧氏距离)

2. 余弦距离 (RelD模型: wide residual network, 这个模型时需要训练的)，得到128维特征向量

**2. 级联匹配：匹配的时候 detect 优先匹配消失时间更短的 Track，会避免 ID switch.**  
原因是：马氏距离会导致 detect 倾向于关联遮挡时间较长的 Track, 而这档时间较长的 Track Kalman filter 的预测是不确定的，这样会破坏 Track 的持续性，这是 SORT 产生 ID Switch 的主要原因

**Listing 1 Matching Cascade**

```

Input: Track indices  $\mathcal{T} = \{1, \dots, N\}$ , Detection indices  $\mathcal{D} = \{1, \dots, M\}$ , Maximum age  $A_{max}$ 
1: Compute cost matrix  $C = [c_{i,j}]$  using Eq. 5
2: Compute gate matrix  $B = [b_{i,j}]$  using Eq. 6
3: Initialize set of matches  $\mathcal{M} \leftarrow \emptyset$ 
4: Initialize set of unmatched detections  $\mathcal{U} \leftarrow \mathcal{D}$ 
5: for  $n \in \{1, \dots, A_{max}\}$  do
6:   Select tracks by age  $\mathcal{T}_n \leftarrow \{i \in \mathcal{T} \mid a_i = n\}$ 
7:    $[x_{i,j}] \leftarrow \text{min\_cost\_matching}(C, \mathcal{T}_n, \mathcal{U})$ 
8:    $\mathcal{M} \leftarrow \mathcal{M} \cup \{(i, j) \mid b_{i,j} \cdot x_{i,j} > 0\}$ 
9:    $\mathcal{U} \leftarrow \mathcal{U} \setminus \{j \mid \sum_i b_{i,j} \cdot x_{i,j} > 0\}$ 
10: end for
11: return  $\mathcal{M}, \mathcal{U}$ 

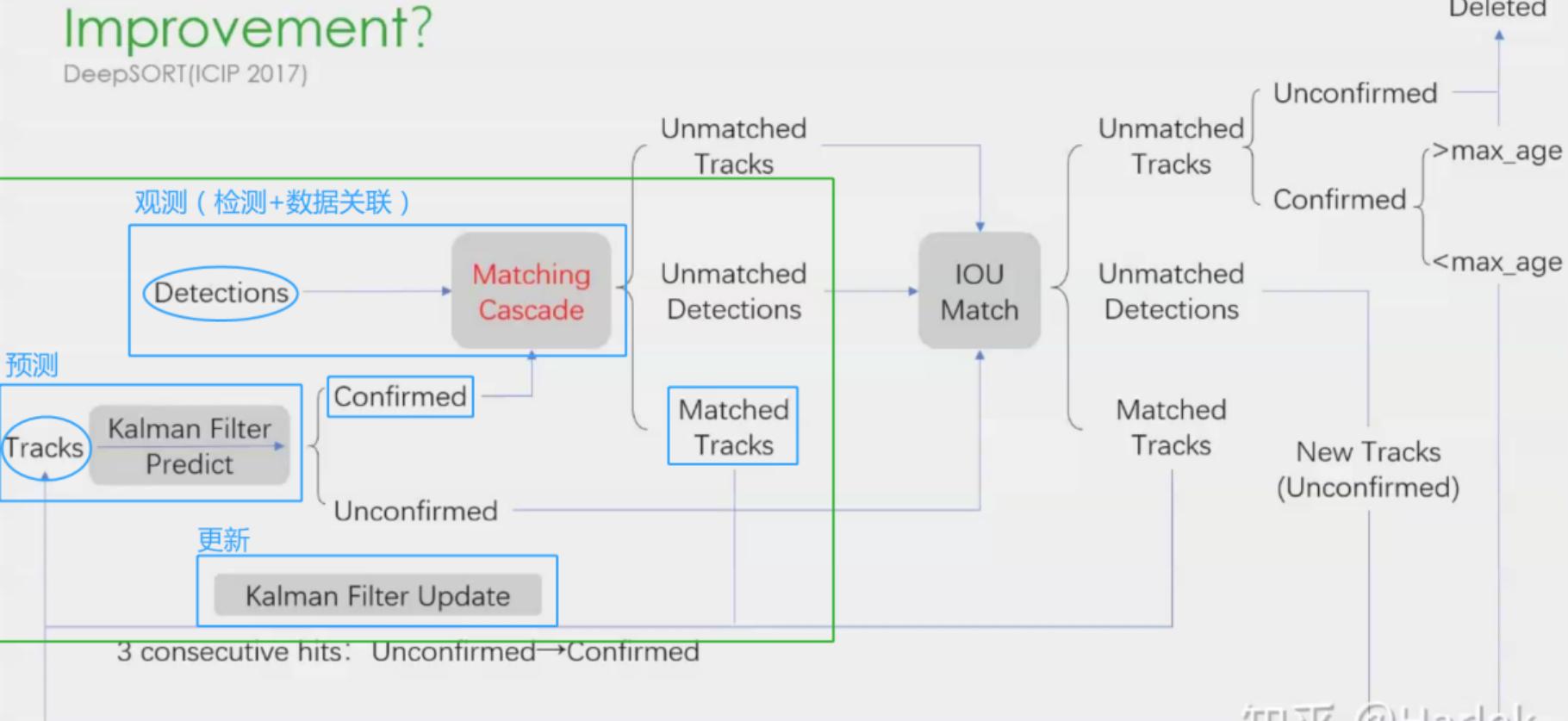
```

| Name                             | Patch Size/Stride | Output Size               |
|----------------------------------|-------------------|---------------------------|
| Conv 1                           | $3 \times 3/1$    | $32 \times 128 \times 64$ |
| Conv 2                           | $3 \times 3/1$    | $32 \times 128 \times 64$ |
| Max Pool 3                       | $3 \times 3/2$    | $32 \times 64 \times 32$  |
| Residual 4                       | $3 \times 3/1$    | $32 \times 64 \times 32$  |
| Residual 5                       | $3 \times 3/1$    | $32 \times 64 \times 32$  |
| Residual 6                       | $3 \times 3/1$    | $64 \times 32 \times 16$  |
| Residual 7                       | $3 \times 3/1$    | $64 \times 32 \times 16$  |
| Residual 8                       | $3 \times 3/2$    | $128 \times 16 \times 8$  |
| Residual 9                       | $3 \times 3/1$    | $128 \times 16 \times 8$  |
| Dense 10                         |                   | 128                       |
| Batch and $\ell_2$ normalization |                   | 128                       |

两个距离的加权作为Cost Metrix!!!

# Improvement?

DeepSORT(ICIP 2017)



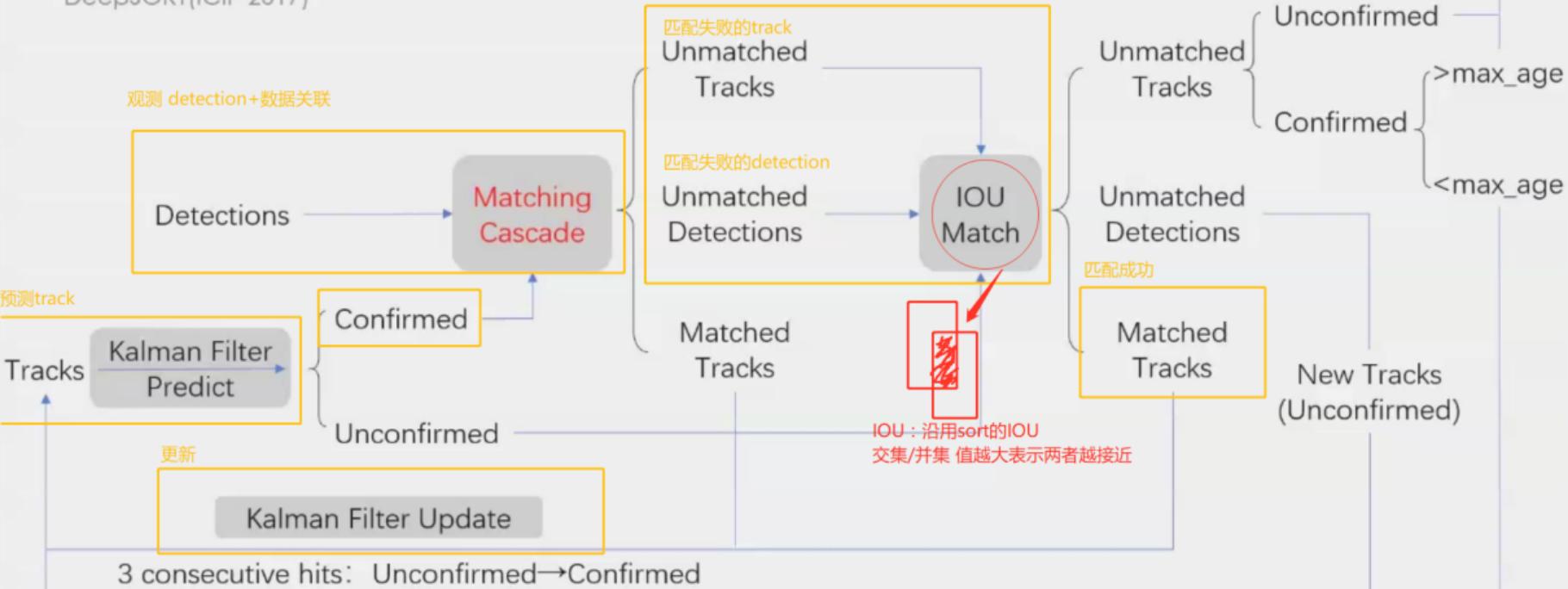
知乎 @Harlek

## 1. 核心流程

- ① 从预测（Tracks）开始；
- ② 经过kalman滤波预测后，会对当前帧预测一个轨迹Bbox，先不看unconfirmed，假如预测出的是confirmed（对于confirmed和unconfirmed,是用来区别跟踪的目标是不是一个目标，比如一个误识别的背景那就是unconfirmed，目标人或者车就是confirmed）；
- ③ 对当前帧进行detection，然后将detection Bboxs结果和预测的confirmed track Bbox进行数据关联（matched tracks）；
- ④ 匹配完成后，更新跟踪的bbox（卡尔曼滤波预测的Bbox）。这里要注意，更新和匹配不是在时刻上顺延的（不是T2时刻匹配,T3时刻更新的关系），而是在下一帧时刻需要完成的两项流程（T2时刻匹配，继续在T2时刻更新）；

# Improvement?

DeepSORT(ICIP 2017)



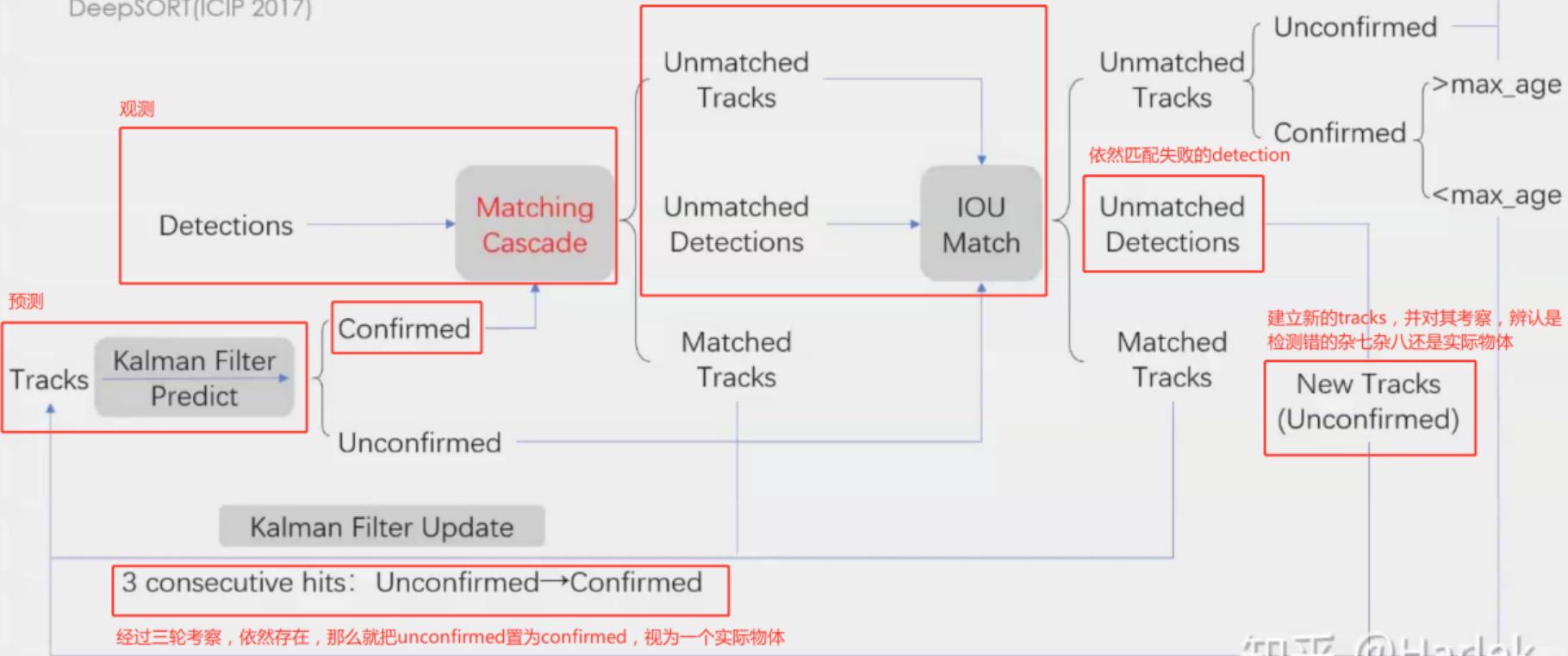
知乎 @Harlek

## 2. 匹配失败的情况

- ① 匹配失败的第一种情况，那就是对于匹配不上的**tracks**和**detection**再次进行匹配，会出现两种情况，一个是匹配成功了，一个是依然还有部分匹配不到，先来看再次匹配能匹配成功的情况；
- ② 依然从预测的**tracks**开始，经过卡尔曼滤波预测，与**detection**匹配，发现此时的**tracks**匹配失败，为什么会有**tracks**匹配失败的情况呢？检测可能发生了漏检，某时刻，预测的轨迹**tracks**还在，但是检测器没有检测到与之对应的目标；
- ③ 为什么会有**detection**匹配失败的情况呢？可能某一时刻有一个物体是新进入的镜头（比如，之前一直只有三个物体，某时刻突然镜头中出现了第四个新物体），就会发生**detection**匹配不到**tracks**的情况，因为这个物体是新来的，在这之前并没有它的轨迹用于预测；还有一种情况就是物体长时间被遮挡，导致检测到的物体没有可以与之匹配的轨迹；
- ④ 针对上述匹配失败问题，处理方法就是对匹配失败的**tracks**和匹配失败的**detection**进行IOU匹配。如果能匹配成功，则再进行更新，然后继续进行预测 - 观测 - 更新的追踪流程；

# Improvement?

DeepSORT(ICIP 2017)



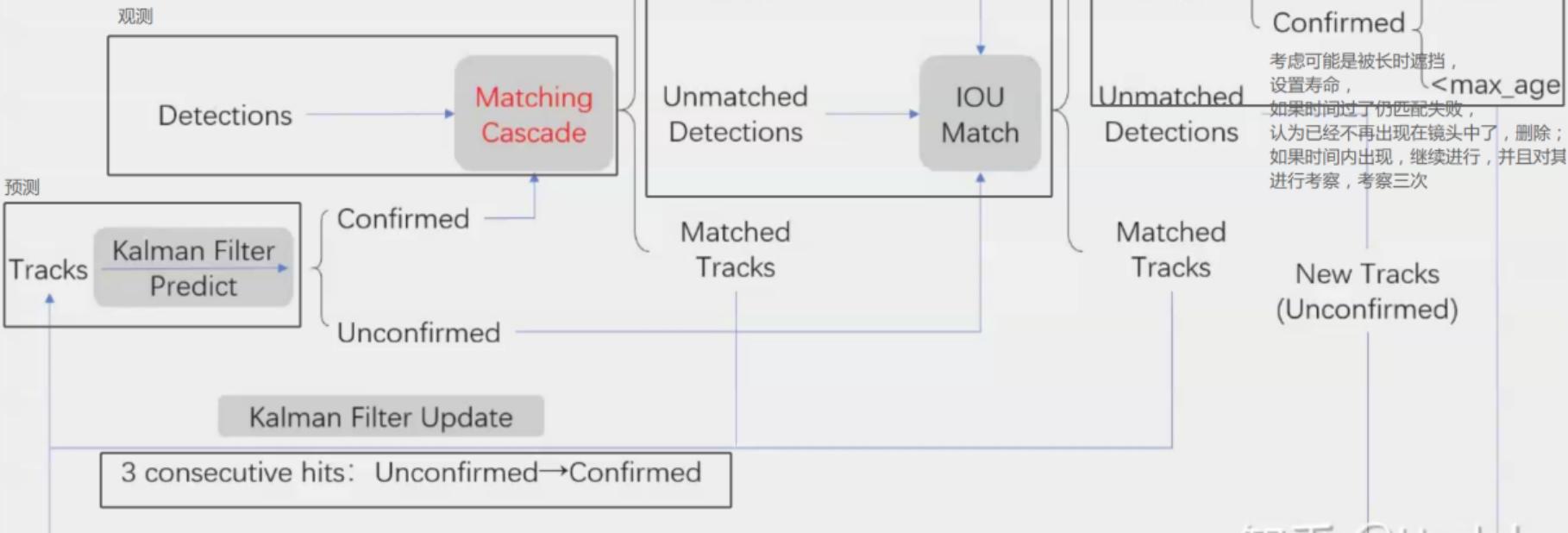
知乎 @Harlek

### 3. 再次匹配也匹配不成功的怎么办呢? (Detections)

对于再次匹配依然失败的detections，前面讨论过detections匹配失败的原因可能是新出现的目标在之前没有它的轨迹tracks，或者长时遮挡的也没有轨迹tracks，所以对其建立一个new tracks，前面也提到，tracks都会设置confirmed/unconfirmed，对于新建立的tracks，尚未确定是否就是真切存在的，万一是检测错的杂七杂八呢，所以将其设置为unconfirmed，并对其进行三次考察，如果是实际目标修改为confirmed，再进行预测-观测-更新

# Improvement?

DeepSORT(ICIP 2017)



## 4. 再次匹配也匹配不成功的怎么办？(Tracks)

对于再次匹配依然失败的tracks，前面讨论过可能是检测器漏检了目标，此时看其是否是confirmed，如果是unconfirmed，将其删除；反之，为其设置寿命，在寿命时间之内都没发生变化( $> max\_age$ )则将其delete，认为其已移出镜头；如果寿命之内( $< max\_age$ )，同样对其进行三次考察，是否是跟踪的杂七杂八，再进行预测-观测-更新

① 目标跟踪算法详解(Sort,DeepSort)

② DeepSort中的ReID算法

③ 解耦合的目标检测算法

④ 工程实现与测试

| Layer (type:depth-idx) | Param # |
|------------------------|---------|
| <hr/>                  |         |
| —Sequential: 1-1       | --      |
| └Conv2d: 2-1           | 1,792   |
| └BatchNorm2d: 2-2      | 128     |
| └ReLU: 2-3             | --      |
| └MaxPool2d: 2-4        | --      |
| —Sequential: 1-2       | --      |
| └BasicBlock: 2-5       | --      |
| └Conv2d: 3-1           | 36,864  |
| └BatchNorm2d: 3-2      | 128     |
| └ReLU: 3-3             | --      |
| └Conv2d: 3-4           | 36,864  |
| └BatchNorm2d: 3-5      | 128     |
| └BasicBlock: 2-6       | --      |
| └Conv2d: 3-6           | 36,864  |
| └BatchNorm2d: 3-7      | 128     |
| └ReLU: 3-8             | --      |
| └Conv2d: 3-9           | 36,864  |
| └BatchNorm2d: 3-10     | 128     |
| —Sequential: 1-3       | --      |
| └BasicBlock: 2-7       | --      |
| └Conv2d: 3-11          | 73,728  |
| └BatchNorm2d: 3-12     | 256     |
| └ReLU: 3-13            | --      |
| └Conv2d: 3-14          | 147,456 |
| └BatchNorm2d: 3-15     | 256     |
| └Sequential: 3-16      | 8,448   |
| └BasicBlock: 2-8       | --      |
| └Conv2d: 3-17          | 147,456 |
| └BatchNorm2d: 3-18     | 256     |
| └ReLU: 3-19            | --      |
| └Conv2d: 3-20          | 147,456 |
| └BatchNorm2d: 3-21     | 256     |

|                    |           |
|--------------------|-----------|
| —Sequential: 1-4   | --        |
| └BasicBlock: 2-9   | --        |
| └Conv2d: 3-22      | 294,912   |
| └BatchNorm2d: 3-23 | 512       |
| └ReLU: 3-24        | --        |
| └Conv2d: 3-25      | 589,824   |
| └BatchNorm2d: 3-26 | 512       |
| └Sequential: 3-27  | 33,280    |
| └BasicBlock: 2-10  | --        |
| └Conv2d: 3-28      | 589,824   |
| └BatchNorm2d: 3-29 | 512       |
| └ReLU: 3-30        | --        |
| └Conv2d: 3-31      | 589,824   |
| └BatchNorm2d: 3-32 | 512       |
| —Sequential: 1-5   | --        |
| └BasicBlock: 2-11  | --        |
| └Conv2d: 3-33      | 1,170,648 |
| └BatchNorm2d: 3-34 | 1,024     |
| └ReLU: 3-35        | --        |
| └Conv2d: 3-36      | 2,359,296 |
| └BatchNorm2d: 3-37 | 1,024     |
| └Sequential: 3-38  | 132,096   |
| └BasicBlock: 2-12  | --        |
| └Conv2d: 3-39      | 2,359,296 |
| └BatchNorm2d: 3-40 | 1,024     |
| └ReLU: 3-41        | --        |
| └Conv2d: 3-42      | 2,359,296 |
| └BatchNorm2d: 3-43 | 1,024     |
| —AvgPool2d: i-6    | --        |
| —Sequential: 1-7   | --        |
| └Linear: 2-13      | 131,328   |
| └BatchNormId: 2-14 | 512       |
| └ReLU: 2-15        | --        |
| └Dropout: 2-16     | --        |
| └Linear: 2-17      | 193,007   |

```

# loss and optimizer
criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(net.parameters(), args.lr, momentum=0.9, weight_decay=5e-4)
best_acc = 0.

# train function for each epoch
def train(epoch):
    print("\nEpoch : %d" % (epoch+1))
    net.train()
    training_loss = 0.
    train_loss = 0.
    correct = 0
    total = 0
    interval = args.interval
    start = time.time()
    for idx, (inputs, labels) in enumerate(trainloader):
        # forward
        inputs, labels = inputs.to(device), labels.to(device)
        outputs = net(inputs)
        loss = criterion(outputs, labels)

        # backward
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # accumulating
        training_loss += loss.item()
        train_loss += loss.item()
        correct += outputs.max(dim=1)[1].eq(labels).sum().item()
        total += labels.size(0)

        # print
        if (idx+1)%interval == 0:
            end = time.time()
            print("[progress:{:.1f}%]time:{:.2f}s Loss:{:.5f} Correct:{} / {} Acc:{:.3f}%".format(
                100.* (idx+1)/len(trainloader), end-start, training_loss/interval, correct, total, 100.*correct/total
            ))
            training_loss = 0.
            start = time.time()

    return train_loss/len(trainloader), 1.- correct/total

```

```

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument("--config_deepsort", type=str, default='./configs/deep_sort.yaml', help='Configure tracker')
    parser.add_argument("--cpu", dest="use_cuda", action="store_false", default=True, help='Run in CPU')
    args = parser.parse_args()

    cfg = get_config()
    cfg.merge_from_file(args.config_deepsort)
    use_cuda = args.use_cuda and torch.cuda.is_available()
    torch.set_grad_enabled(False)
    model = build_tracker(cfg, use_cuda=False)

    model.reid = True
    model.extractor.net.eval()

    device = 'cuda'
    output_onnx = 'deepsort.onnx'
    # -----
    print("==> Exporting model to ONNX format at '{}'".format(output_onnx))
    input_names = ['input']
    output_names = ['output']

    input_tensor = torch.randn(1, 3, 128, 64, device=device)

    torch.onnx.export(model.extractor.net.cuda(), input_tensor, output_onnx, export_params=True, verbose=False,
                      input_names=input_names, output_names=output_names, opset_version=10,
                      do_constant_folding=True,
                      dynamic_axes={'input': {0: 'batch_size'}, 'output': {0: 'batch_size'}})

```

## Dynamic Shape

```

void DeepSortEngineGenerator::createEngine(std::string onnxPath, std::string enginePath) {
    // Load onnx model
    auto builder = createInferBuilder(*gLogger);
    assert(builder != nullptr);
    const auto explicitBatch = 1U << static_cast<uint32_t>(NetworkDefinitionCreationFlag::kEXPLICIT_BATCH);
    auto network = builder->createNetworkV2(explicitBatch);
    assert(network != nullptr);
    auto config = builder->createBuilderConfig();
    assert(config != nullptr);

    auto profile = builder->createOptimizationProfile();
    Dims dims = Dims4{1, 3, IMG_HEIGHT, IMG_WIDTH};
    profile->setDimensions(INPUT_NAME.c_str(),
        OptProfileSelector::kMIN, Dims4{1, dims.d[1], dims.d[2], dims.d[3]});
    profile->setDimensions(INPUT_NAME.c_str(),
        OptProfileSelector::kOPT, Dims4{MAX_BATCH_SIZE, dims.d[1], dims.d[2], dims.d[3]});
    profile->setDimensions(INPUT_NAME.c_str(),
        OptProfileSelector::kMAX, Dims4{MAX_BATCH_SIZE, dims.d[1], dims.d[2], dims.d[3]});
    config->addOptimizationProfile(profile);
}

nvonnxparser::IParser* parser = nvonnxparser::createParser(*network, *gLogger);
assert(parser != nullptr);
auto parsed = parser->parseFromFile(onnxPath.c_str(), static_cast<int>(ILogger::Severity::kWARNING));
assert(parsed);
if (useFP16) config->setFlag(BuilderFlag::kFP16);
config->setMaxWorkspaceSize(1 << 20);
ICudaEngine* engine = builder->buildEngineWithConfig(*network, *config);

// Serialize model and save engine
IHostMemory* modelStream = engine->serialize();
std::string serializeStr;
std::ofstream serializeOutputStream;
serializeStr.resize(modelStream->size());
memcpy((void*)serializeStr.data(), modelStream->data(), modelStream->size());
serializeOutputStream.open(enginePath, std::ios::binary);
serializeOutputStream << serializeStr;
serializeOutputStream.close();
}

```

使用 trtexec 工具

```

./trtexec --onnx=deepsort.onnx --saveEngine=deepsort.trt --workspace=1024 --minShapes=input:1x3x128x64 --optShapes=input:128x3x128x64 --
maxShapes=input:128x3x128x64 --fp16 --verbose

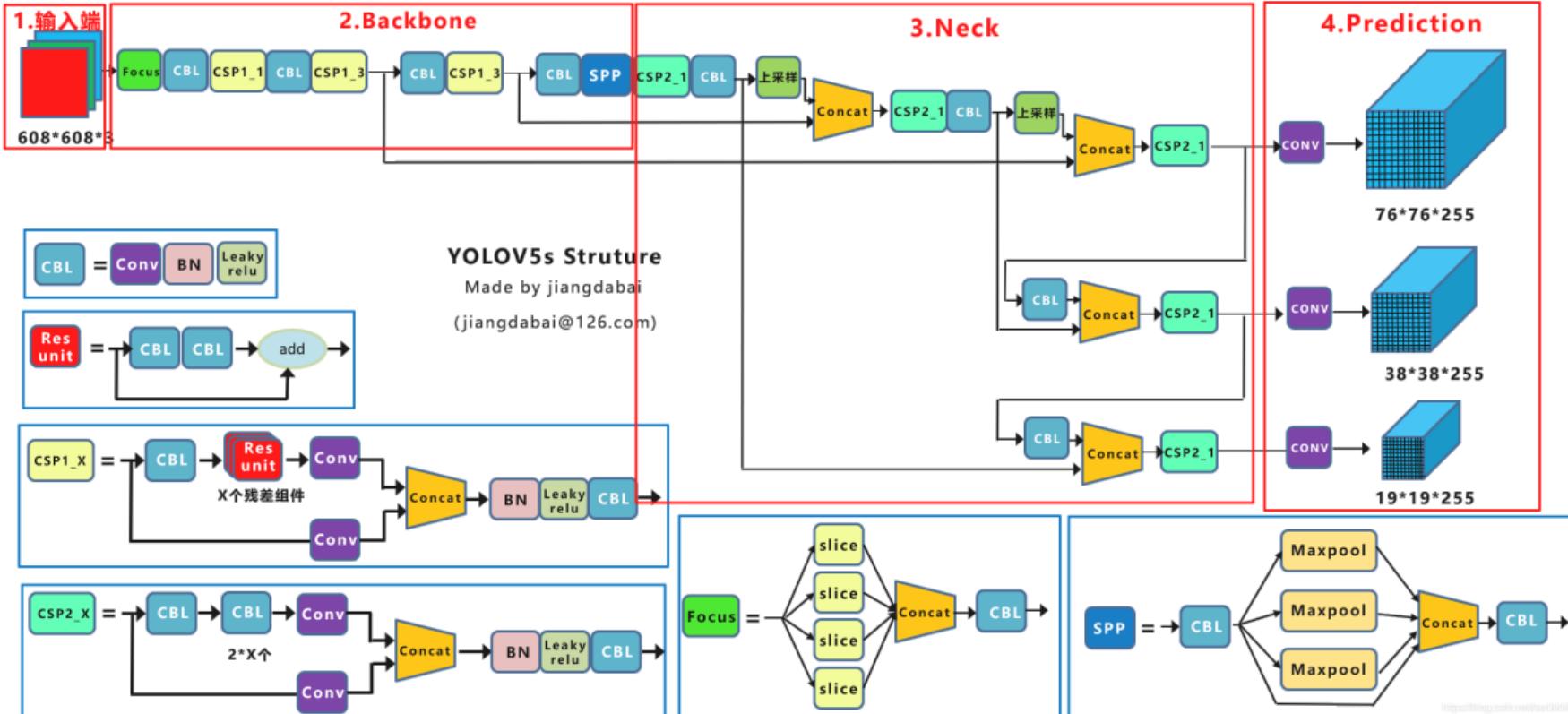
```

① 目标跟踪算法详解(Sort,DeepSort)

② DeepSort中的ReID算法

③ 解耦合的目标检测算法

④ 工程实现与测试



## 如何正确导出ONNX

- ① 对于任何用到shape,size返回值的参数时，例如`tensor.view(tensor.size(0),-1)`这类操作，避免直接使用`tensor.size`的返回值，而是加上int强转；
- ② 对于`nn.Upsample`或`nn.functional.interpolate`函数，使用`scale_factor`指定倍率，而不是使用`siz`参数指定大小；
- ③ 对于`reshape`,`view`等操作，`-1`的指定请放在`batch`维度。其他维度可以计算出来即可。`batch`维度禁止指定为到大于-1的明确数字；
- ④ `torch.onnx.export`指定`dynamic_axes`参数，并且只指定`batch`维度。我们只需要动态`batch`,相对动态的宽高有其他方案；
- ⑤ 使用`opset_version=11`，不低于11；
- ⑥ 这些做法的必要性体现在，简化过程的复杂度，去掉`gather`,`shape`类型的节点，简化ONNX,提高TensorRT序列化的成功率。

```

def forward(self, x):
    # x = x.copy() # for profiling
    z = [] # inference output
    self.training |= self.export
    for i in range(self.nl):
        x[i] = self.m[i](x[i]) # conv
        # 变int #
        bs, ny, nx = map(int,x[i].shape) # x(bs,255,20,20) to x(bs,3,20,20,85)
        # x[i] = x[i].view(int(bs), int(self.na), int(self.no), int(ny), int(nx)).permute(0, 1, 3, 4, 2).contiguous()
        x[i] = x[i].view(-1, self.na, self.no, ny, nx).permute(0, 1, 3, 4, 2).contiguous()

    if not self.training: # inference
        if self.grid[i].shape[2:4] != x[i].shape[2:4]:
            self.grid[i] = self._make_grid(nx, ny).to(x[i].device)

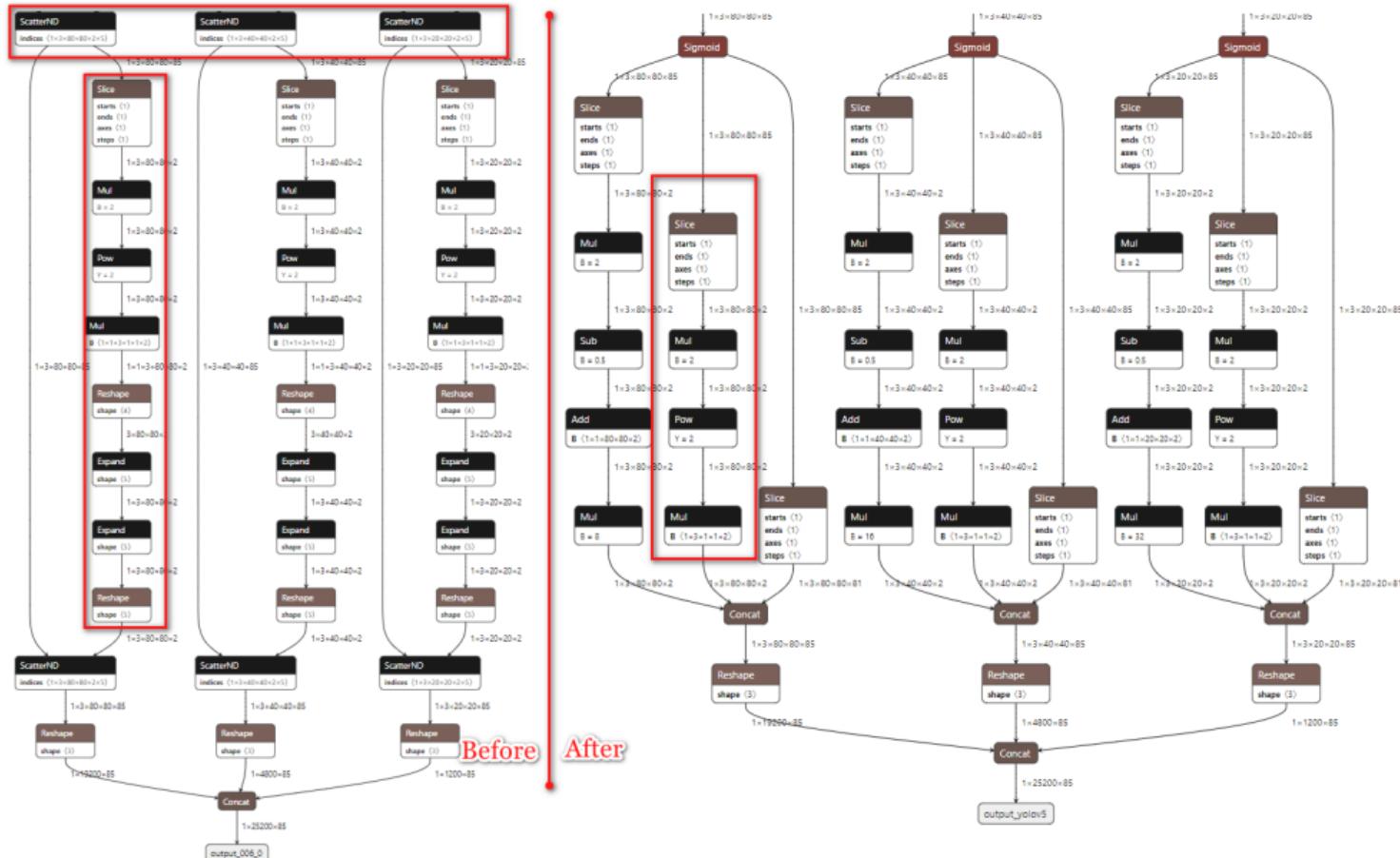
        y = x[i].sigmoid()
        # 热映射, inplace
        # y[..., 0:2] = (y[..., 0:2] * 2. - 0.5 + self.grid[i].to(x[i].device)) * self.stride[i] # xy
        # y[..., 2:4] = (y[..., 2:4] * 2) ** 2 * self.anchor_grid[i] # wh

        xy = (y[..., 0:2] * 2. - 0.5 + self.grid[i].to(x[i].device)) * self.stride[i]
        wh = (y[..., 2:4] * 2) ** 2 * self.anchor_grid[i].view(1,self.na,1,1,2)
        y = torch.cat((xy,wh,y[..., 4:]),-1)
        # z.append(y.view(bs, -1, self.no))
        # -1维度放在batch size维度上
        print("======>>>>>>>")
        print(y.size())
        # z.append(y_temp.view(bs, -1, self.no))
        z.append(y.view(-1,int(y.size(1) * y.size(2) * y.size(3)), self.no))

    a = torch.cat(z, 1)
    print(a.shape)

    # 只返回一个结点
    # return x if self.training else (torch.cat(z, 1), x)
    return x if self.training else torch.cat(z, 1)

```

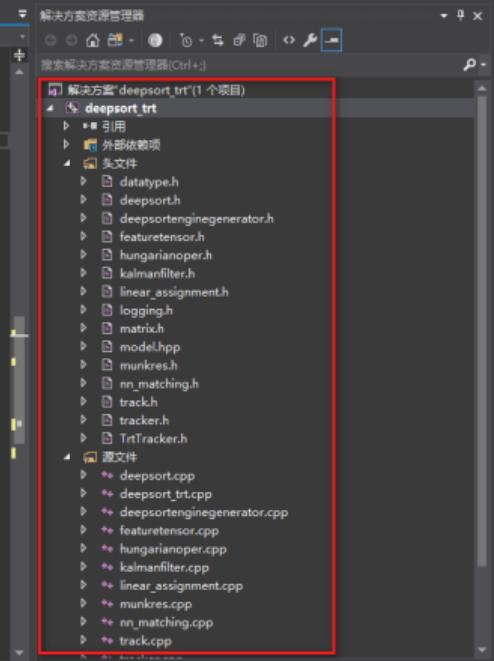


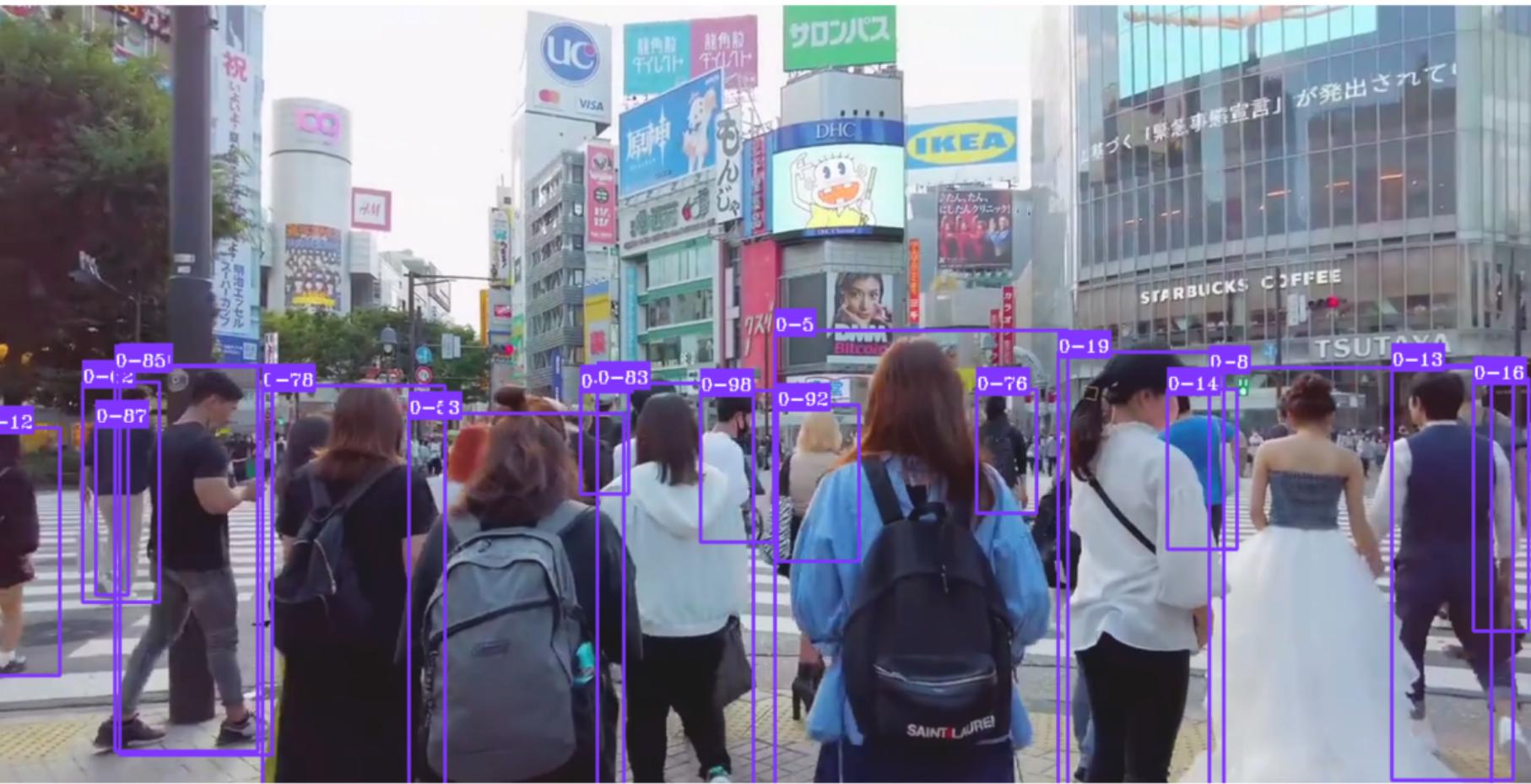
① 目标跟踪算法详解(Sort,DeepSort)

② DeepSort中的ReID算法

③ 解耦合的目标检测算法

④ 工程实现与测试





The End

Thanks for Your Attention!