TLD 代码分析

By Xian2207, 13689903575, wszhangxian@126.com

一：run_tld.cpp 文件开始

1 参数定义有：

```
Rect box;
bool drawing_box = false;
bool gotBB = false;
bool tl = true;
bool rep = false;
bool fromfile = false;
string video;
```

```
void readBB(char* file){ }
```

作用：1 创建了一个 bb_file 用于记录 bounding box 位置的文档；2 初始化了 box(x, y, w, h);

```
void mouseHandler(....)
```

作用：鼠标画框之后，gotBB = true，同时得到 ROI box 的(x, y, width, height)

```
void read_options(....)
```

作用：判断 gotBB/fromfile/tl/rep = true or false；注意如果 gotBB=true，自然跳过 while 语句

```
int main() {                    //主函数入口

TLD tld;                        //tld 是 tld.cpp 设置下的 3 次重载函数
tld.read(fs.getFirstTopLevelNode()) //函数重载，原型 tld.cpp 下定义的 TLD::TLD(...) { read(...) }

capture >> frame;      //读取一帧，转换为灰度图 last_gray，复制到 first 里面
cvtColor(frame, last_gray, CV_BGR2GRAY);
```

frame.copyTo(first)

程序开始

while(!gotBB) {　　// 先看 gotBB = true/false; 如果 true, !gotBB = false, 跳过 while 语句

读取一帧 frame;

转灰度图 last_gray;

drawBox(frame, box)

}

由于 David 这个视频，我们是从 datasets 里取出来的，并非是摄像头适时拍摄，所以 gotBB = true, 程序直接跳过 while 语句，开始下一行：

FILE *bb_file = fopen("bounding_boxes.txt", "w")　　//用来记录 boundingbox 的位置

/* ************** TLD initialiation ************

tld.init(last_gray, box, bb_file) ;

>> tld.cpp

>> void TLD::init(const Mat &frame1, const Rect &box, FILE *bb_file), 进入函数：

buildGrid(frame1, box)　　//frame1 = last_gray, box = 鼠标 box

>> tld.cpp

>>void buildGrid(img, box) {　//img = last_gray, box =鼠标 box;

const float SHIFT = 0.1; //检测时，扫描框移动的步长值

const float SCALES[] = {0.1651, ..., 4.2998} //共计 21 个数字，即 21 个不同的比例级别

int width, height, min_bb_size;

BoundingBox bbox;　　//bbox 具有 Rect 的公有成员如(x,y,width,height)和(overlap, sidx)

>> TLD.h

>>struct BoundingBox : public cv::Rect {

BoudningBox() {}

```
        BoundingBox(cv::Rect r) : cv::Rect(r) {}
    public:
        float overlap;    //box 的重合度
        Int sidx;         //box 的索引序号
    }
```

>>tld.cpp -> buildGrid(...)

```
    Size scale;     //CV 自带函数
    int sc = 0;     //计数器
    for(s = 0; s < 21; s++) {
        width = round(box.width*SCALES[s]);
        height = round(box.height*SCALES[s]);
        scale.width = width;
        scale.height = height;
        min_bb_side = min(height, width);
        scales.push_back(scale);     //scale 是 vector，存放 21 个不同的(width, height);
        for(int y = 1; y < img.rows - height; y = y+ round(SHIFT*min_bb_size))
            for(int x = 1; x < img.cols - width; x = x + round(SHIFT*min_bb_size))
            //21 个不同 SCALES[s]，产生 21 个不同 min_bbsize；每个 SCALES[]产生
            //img.rows-height by img.cols - width 个不同的 bbox
                bbox.x = x; bbox.width = width;
                bbox.y = y; bbox.height = height;
                bbox.overlap = bbOverlap( bbox, BoundingBox(box) );
            //BoundingBox(box)是想利用 box 的 Rect 的公有函数如 x, y, width, height
            //从而使 box 与 bbox 可以进行 bbOverlap 函数的操作，进入函数：
                >> float TLD::bbOverlap(BoundingBox& box1, BoundingBox& box2)
                                        //box1 = bbox; box2 = box
                ....
                return intersection/(....) //返回的是一个 float 型的值
```

grid.push_back(bbox); //

>> TLD.h

>> std::vector<BoundingBox> grid; //存放扫描窗口的容器，大小为

// 21 by (img.rows-height) by (img.cols - width)个，即 21 个 m by n bbox,

//每个元素有 x, y, width, height, overlap, sidx 等六个成员

>>tld.cpp

printf(....., int(grid.size())) //相当于完成 buildGrid 函数计算，进行下一行：

>> tld.cpp

Iisum.creat(frame1.rows+1, frame1.cols+1, CV_32F)　//frame1 = last_gray, creat 是 CV 函数

Iisum.creat(frame1.rows+1, frame1.cols+1, CV_64F)　//frame1 = last_gray，创建积分图

dconf.reserve(100); //reserve 是预先分配 vector 内存空间的函数，c++内置函数

dbb.reserve(100); //dconf, dbb->TLD.h->class TLD

bbox_step = 7;

tmp.conf = vector<float> (grid.size()); //初始化 tmp.conf, 大小为 grid.size()个，每个元素为 0

tmp.patt = vector<vector<int> >(grid.size(), vector<int>(10, 0)); //初始化，大小为 grid.size()个

//每个元素是 10 by 1 的 vector

//元素全为 0;；来自 TLD.h

>> TLD.h

>> struct TempStruct {

std::vector<std::vector<int> > patt;

std::vector<float> conf;

}

>> tld.cpp

dt.bb.reserve(grid.size());

>> TLD.h

>> DetStruct dt;

>> struct DetStruct {

std::vector<int> bb;

std::vector<std::vector<int>> patt;

std::vector<float> conf1;

std::vector<float> conf2;

std::vector<std::vector<int>> isin;

std::vector<cv::Mat> patch;

}

>> tld.cpp

good_boxes.reserve(grid.size());

bad_boxes.reserve(grid.size());

>> TLD.h -> class TLD -> 定义 good_boxes, bad_boxes;但没有赋值

>> tld.cpp

pEx.create(patch_size, patch_size, CV64F); //tld.read(fs)赋值了 patch_size;pEx->TLD.h->class //TLD-> Mat pEx; 这里 pEx 已经产生了

>> tld.cpp

generator = PatchGenerator(0,0, noise_init, true, 1-scale_init, 1+scale_init, -angle_init*CVPI/180, angle_init*CV_PI/180, angle_init*CV_PI/180);

>> TLD.h -> class TLD {

PatchGenerator generator; // PatchGenerator 是 CV 自带的类，noise_init, scale_init 等等都是 TLD.h 类中的私有成员，故以上函数的参数不知大小，值初始化

}

>> tld.cpp

getOverlappingBoxes(box, num_closest_init); //box = 鼠标 box; num_closest_init 来自 tld.read()

>> getOverlappingBoxes(box1, num_cloest_init){ //box1 = box = 鼠标 box

```
        float max_overlap = 0;
        for(int i = 0; i < grid.size(); i++) {
        if(grid[i].overlap > max.overlap)
        max_overlap = grid[i].overlap; //通过 21*(m by n)次循环，找到 max_overlap
        best_box = grid[i];     //best_box->TLD.h->class TLD，这里是 tld.cpp，所以直接用

        If( grid[i].overlap > 0.6 )
           good_boxes.pushback(i); //good_boxes 此时被赋值，tld.cpp 可直接调用了
           else if (grid[i].overlap < bad_overlap) //run_tld.cpp->tld.read(fs...)->bad_overlap，由于
                                                   //bad_overlap->TLD.h->class TLD，所以 tld.read 会
                                                   //赋值，TLD.h 中自动传递，tld.cpp 这里直接使用
           bad_boxes.pushback(i); //bad_boxes 此时被赋值
           }
        if(good_boxes.size() > num_closest)    //num_closest=num_closest_init 由 tld.cpp()读取
        std::nth_element(good.boxes.begin(),good_boxes.begin()+num_closest,good_boxes.end(),
                         0Comparator(grid));   //nth_element 是 c++自带函数，进入 0Comparator
        goodboxex.resize(num_closest);
      >> TLD.h
      >> struct 0Comparator{
      0Comparator(const std::vector<BoundingBox>& _grid) : grid(_grid) {} //带参数的构造函数
                                                                          //把_grid 赋值给 grid
      std::vector<BoundingBox> grid;
      bool operator()(int idx1, int idx2) { return grid[idx1.overlap>grid[idx2].overlap]};
      }
>> tld.cpp
getBBHull();
   >> tld.cpp
   >> void TLD::getBBHull()
```

int x1 = INT_MAX, x2 = 0;

int y1 = INT_MAX, y2 = 0;

int idx;

for(int i=0; i<good_boxes.size(); i++) { //good_boxes 的大小未知，取决于上面代码分类

Idx = good_boxes[i];

x1 = min(gird[idx].x , x1);

y1 = ...;

x2 = max(grid[idx].x + grid[idx].width, x2);

y2 = ...;

bbhull.x = x1; bbhull.y = y1; bbhull.width = x2 - x1; bbhull.height = y2 - y1;

} //bbhull 来自 LTD.h -> BoundingBox bbhull，可自动调用 x, y, width, height 成员

>> tld.cpp

printf(..., good_boxes.x,...); printf(..., best_boxes.x,....); printf(..., bbhull.x, ...) //程序进入下一行

lastbox = best_box; //best_box->buildGrid()->getOverlappingBoxes()赋值了; lastbox->TLD.h->
//->class TLD->BoundingBox lastbox,故在 tld.cpp 可自由调用

lastconf = 1; // lastconf -> TLD.h -> float lastconf;

lastvalid = true; // 同上

fprintf(bb_file, ..., lastbox.x, lastbox.y,....);

/**准备分类器**

classifier.prepare(scales);

//----------长注释----------

scales -> TLD.h -> std::v ector<cv::Size> scales; 同时 tld.cpp 里 buildGrid()已经对 scales 赋值，即 scales.push_back(scale)，存放 21 个不同的(width, height); 由于 classifier.prepre(scales)恰好也在 tld.cpp 文件下，所以 buildGrid 赋值后，相当于给了 TLD.h 里的 scales 赋值，完了在 classifier.prepare()里调用，注意 C++的调用灵活性。进入 prepare 函数：

>>FerNNClassifier.cpp

>>void FerNNClassifier::prepare(const vector<Size>& scales)

```
acum = 0;
int totalFeatures = nstructs*structSize; //nstructs, structSize 来自 classifier.read(fs)
features = vector<vector<Feature> >(scales.size(), vector<Feature>(totalFeatures));
RNG& rng = theRNG();     //theRNG 是 CV 自带函数；features 是 21 by 130 数组
float x1f, x2f, y1f, y2f;       //因为 scales.size()=21
int x1, x2, y1, y2;
for(int i=0; i<totalFeatures; i++)    //随机数充填每个扫描窗口的特征
   {
   x1f = (float)rng; x2f = (float) rng; y1f=(float)rng; y2f=(float)rng;
   for(int s=0; s<scales.size(); s++) {
      x1=x1f*scales[s].width; x2=x2f*scales[s].width; y1=y1f*scales[s]; y2=...;
      features[s][i] = Feature(x1,y1,x2,y2); //Feature->FerNNClassifier.h; features 被赋值了
      }
   }
   thrN = 0.5*nstructs;                        //nstructs 在 tld.read()读到了,这里 thrN 被赋值
   for(int i=0; i<nstructs; i++) {
      posteriors.push_back(vector<float>(pow(2.0, structSize), 0));
      pCounter.push_back(....);              //posteriors,pCounter,nCounter->FerNNClassifier.h
      nCounter.push_back(....);
   }
```

调用模式：TLD.h->#<FerNNClassifier.h-> thr_fern. structSize, nstructs, valid, ncc_thesame, thr_nn, acum, thr_nn_valid, read(), prepare(), getFeature().....->函数实现在 FerNNClassifier.cpp; 所以 class TLD{ FerNNClassifier classifier}->tld.cpp 中可以直接用 classifier.prepare(...)；进入 prepare 函数发现，需要调用 nstructs, structSize 等变量，其实早在 run_tld.cpp->tld.read(fs...)函数里，就已经读了 fs 的所有参数，包括 scale_init, angle, thr_nn 等，因为 TLD.h->read()->tld.cpp->定义时，最下面一行有 classifier.read(fs)，所以自然就被读到了。最终，prepare()提供 features(大小为 scales.size()行， totalFeatures 列), posterioriors, pCounter, nCounter。但注意，结果仍然在 FerNNClassifier.cpp 中, tld.cpp 无法直接使用----------//

>>tld.cpp

generatePositiveData(frame1, num_warps_init); //frame1 = last_gray, num_warps_init 已经被
tld.read()读过了，产生正样本

>>tld.cpp->generatePositiveData 函数

Scalar mean, stdev;

getPattern(frame(best_box), pEx, mean, stdev); //frame=last_gray, best_box 已经在 tld.cpp
//定义了，由 buildGrid()函数赋值计算过
//pEx->TLD.h->std::vector<cv::Mat> pEx

>>getPattern() //函数在 tld.cpp

resize(img, pattern, Size(patch_size, patch_size)); //图像缩放到 patch_size；img =
//last_gray(best_box)=frame(best_box);
//pattern=pEx, pEx 已经初化了，存放计
//算结果的，相当于被赋值

meanStdDev(pattern, mean, stdev); //meanStdDev->CV 自带函数

pattern.convertTo(pattern, CV_32F);

pattern = pattern - mean.val[0]; //.val 是 CV 自带函数

>>再次进入 generatePositiveData()函数

Mat img, warped; //获取 fern features on warped patches

GaussianBlur(frame, img, Size(9,9), 1.5); //frame=last_gray; img 是创造出的存放结果图像地方

warped = img(bbhull); //bbhull->TLD.h->class TLD->BoundingBox bbhull；此外
//tld.cpp->void TLD::getBBHull()已经算出了 bbhull，由于
//#include<TLD.h>，所以这里可以直接调用

RNG& rng = theRNG(); //theRNG()是 CV 自带函数

Point2f pt(bbhull.x+....); //获取 bbhull 中心坐标，存储在 Point2f 型 pt 里面

vector<int> fern(classifier.getNumStructs()); //TLD.h #include<FerNNClassifier.h> ->
//Class FerNNClassifier-> int getNumStructs(){return
//nstructs}而 nstructs 被 tld.read()->classifier.read(fs)
读到了。如果 classifier.prepar()可以用，同样
//classifier.getNumStructs()也可以直接被调用

pX.clear(); //pX->TLD.h->class TLD->vector<pair<vector<int>, int> > pX;已经初始化了；此外要注
//意: .clear()只是移除 pX 数据；pX.compacity()不变，内存并没释放，除非定义个
//vector<...> a()，swap(pX,a)才能释放内存

Mat patch;

```
if(pX.capacity()<num_warps*good_boxes.size());        //num_waprs=num_warps_init->tld.read();
                                                      //good_boxes 在 TLD.h->class TLD 定义了，
                                                      //被 buildGrid()函数计算过
{pX.reserve(num_warps*good_boxes.size);}


int idx;
for(int i=0; i<num_warps; i++) {
   if(i>0)
      generator(frame, pt, warped, bbhull.size(), rng);      //frame=last_gray; generator CV 函数
   for(int b=0; b<good_boxes.size(); b++){
      idx = good_boxes[b];
      patch = img(grid[idx]);                                 //Mat img 新近定义的，见上面
      classifier.getFeatures(.....);                          //getFeatures->FerNNClassifier.cpp
      pX.push_back(make_pair(fern,1));              //pX->TLD.h->class TLD，这里 pX 被赋值了
      }
      printf(........);
   }
//---------------generatePositiveData()函数分析完毕，返回 tld.cpp---------------------
>>tld.cpp
Scalar stdev, mean;
meanStdDev(frame1(best_box), mean, stdev);
Integral(frame1, iisum, iisqsum);                         //计算积分
var = pow(stdev.val[0], 2)*0.5;                           //var->TLD.h->class TLD-> float var;
double vr = getVar(best_box, iisum, iisqsum)*0.5;         //getVar->TLD.cpp


generateNegativeData(frame1);                             //frame1=last_gray; 产生负样本
      >>tld.cpp->generateNegativeData()
         random_shuffle(bad_boxes.begin(), bad_boxes.end());   //bad_boxes 上面已经计算出来
                                                               //了，random_shuffle c++自带
```

```
int idx, a = 0;
vector<int>  fern(classifier.getNumStructs());          //TLD.cpp 可直接调用 classifier,
                                                        因 为  classifier->TLD.h->class
                                                        TLD->FerNNClassifier  classifier
                                                        定义了
nX.reserve(bad_boxes.size() ); //nX->TLD.h->class TLD->vector<pair<vector<int>, int>> nX;
Mat patch;
for(j=0; j<bad_boxes.size(); j++){
idx = bad_boxes[j];
if(getVar....)
   continue;
patch = frame(grid[idx]);
classifier.getFeatures(patch, grid[idx].sidx, fern);   //TLD.h->class TLD->
                                                       //vector<BoundingBox> grid；fern 上面
                                                       //定义了
nX.push_back(make_pair(fern, 0));                      //类似 pX，这里 nX 被赋值了
a++;
}
printf(...);
Scalar dum1, dum2;                                     //Scalar 是 CV 自带函数
nEx  =  vector<Mat>(bad_patches);                      //nEx  和  bad_matches->TLD.h->class  TLD;
                                                       //bad_matches 在 tld.read()就已经赋值了
for(...){....getPattern()....};
printf(...,  (int)  nEx.size()  );                     //nEx 是 vector<Mat>型，被 getPattern()
                                                       函数赋值了
//------------------------------generateNegativeData()分析完了,返回 tld.cpp----------------------------
>>tld.cpp
half = (int)nEx.size()*0.5f;
nExT.assign(nEx.begin()+half,  nEx.end()  ); //nExT->TLD.h->class  TLD,这里 nExT 被赋值了; assign
                                             //c++自带函数，将 nEx.size()个前 half 的值或字符串
                                             //给 nExT, nEx.resize(half); //resize 功能调整 nEx 大小，
```

//使其可以容纳 half 这个对象

```
vector<pair<vector<int>, int> > ferns_data(nX.size() + pX.size() ); //定义 vector 型的 ferns_data，
                                                                    //内部公有 nX,size()+pX,size()对;
                                                                    //nX, pX->TLD.h->class TLD->nX,
                                                                    //pX;这里 ferns_data 长度赋值
vector<int> idx = index_shuffle(0, ferns_data.size() );
    >> tld_utils.cpp->index_shuffle()函数
    >> vector<int> idx = index_shuffle(int begin, int end);   //begin = 0; end = ferns_data.size()
       vector<int> indexes(end-begin);      //初始化 indexes 容器，长度大小为 end-begin 个
       for(int i=begin; i<end; i++){
          Indexes[i] = i;
          }
       random_shuffle(indexes.begin(), indexes.end() ) //c++自带的
       return indexes;                          //indexes 赋值了，传递给 idx
>> tld.cpp
int a = 0;
for(int i = 0; i < pX.size(); i++) {
    ferns_data[idx[a]] = pX[i];    //idx=indexes，是个乱序 vector<int>型数组
    a++;                           //pX 是 10 by 2 元素是(int,1)的 vector; ferns.data 是 20 个
}
for(int i=0; i < nX.size(); i++){
    ferns_data[ idx[a] ] = nX[i];   //fern 上半部分 pX;下半部分 nX
    a++;
}

vector<cv::Mat> nn_data(nEx.size()+1);          //nEx 已经在上面定义过
nn_data[0] = pEx;                               //pEx 上面已经定义了，Mat 型
for(int i=0; i < nEx.size(); i++){
```

```
    nn_data[i+1] = nEx[i];                    //nEx 是 TLD.h->class TLD->std::vector<Mat>
                                              //总之，pEx 和 nEx 都放在一起了
}

classifier.trainF(ferns_data, 2);             //训练分类器
  >>FerNNClassifier.cpp 中: tld.cpp 调用方式为->TLD.h #include<FerNNClassifier.h>->
    Class TLD-> FerNNClassifier classifier;
    thrP = thr_fern*nstructs;                 //thrP 在 FerNNClassifier.h 里定义，所以在
                                              //FerNNClassifier.cpp 中可直接调用
    for(int i=0; i<ferns.size(); i++)         //ferns = ferns_data
    if(ferns[i].second == 1)
      If(measure_forest(ferns[i].first)<=thrP )
        update(ferns[i].first, 1, 1);
    else if(measure_forest(ferns[i].first>=thrN))   //thrN->FerNNClassifier.h，classifier.prepare()
                                                    计算过了
      update(ferns[i].first, 0, 1);
      >>此处又涉及两个函数: measure_forest 和 update
      >>FerNNClassifier.cpp->measure_forest(vector<int> fern)
        float votes = 0;
        for(int i=0; i<nstructs; i++)
        votes+=posteriors[i][fern[i]];      //posteriors 在 classifier.prepare()定义了，恰好这个
                                            //函数在 FerNNClassifier.cpp 里定义；同时 posterior
                                            在 FerNNClassifier.h 中定义，被 classifier.prepare()
                                            函数计算过，所以这里直接用
>>tld.cpp
classifier.trainNN(nn_data);
    >>FerNNClassifier.cpp ->void FerNNClassifier::trainNN(const vector<cv::Mat>& nn_examples)
                                                //nn_example==nn_data
    float conf, dummy;
    vector<int> y(nn_example.size(), 0);
```

```
y[0] = 1;
vector<int> isin;
for(int i=0; i<nn_examples.size(); i++) {
   NNConf(nn_examples[i], isin, conf, dummy);
/----------------------------------------------NNConf()函数分析-----------------------------------------
   >>FerNNClassifier.cpp->
      void NNConf(const Mat& example, vector<int>& isin, float& rsconf, float& csconf);
   isin = vector<int>(3, -1);                    //example=nn_examples=nn_data; rsconf=conf;
      csconf=conf
   If(pEx.empty()){                          //pEx->FerNNClassifier.h->vector<cv::Mat>型，注意定义
                                               //的位置，不要与 TLD.h 中 Mat 型的 pEx 搞混了
      rsconf = 0;
      csconf = 0;
      return; //跳出 if 语句，返回 void NNConf()函数继续执行下面的语句
   }
   If(nEx.empty()){                          //nEx->FerNNClassifier->vector<cv::Mat>型，别与 TLD.h
                                               中的 vector<Mat>型的 nEx 搞混
      rsconf = 1;
      csconf = 1;
      return
   }
   Mat ncc(1,1,CV_32F);
   float nccP, csmaxP, maxP = 0;
   bool anyP = false;
   int maxPidx, validatedPart = ceil(pEx.size()*valid);          //valid->tld.read()读取了
   float nccN, maxN =0;
   bool anyN = false;
   for(int i=0; i<pEx.size(); i++) {
      matchTemplate(pEx[i], example, ncc, CV_TM_CCORR_NORMED); //CV 自带
```

```
        nccP = ( ( (float*)ncc.data )[0] + 1)*0.5; //(float*)指针指向 ncc.data[0]的第一个值
        if(nccP > ncc_thesame)
            anyP = true;
        If(nccP > maxP){
          maxP = nccP;
          maxPidx = i;

         If(i<validatedPart) csmaxP = maxP;
         }
         for(int i = 0; i < nEx.size(); i++) {
              matchTemplate(nEx[i], example, ncc, CV_TM_CCORR_NORMED);
              nccN = (((float*)ncc.data)[0]+1)*0.5;
              If(nccN>ncc_thesame) //ncc_thesame 因为 tld.read->classifier.read()读了
              anyN = true;
              if(nccN > maxN)
              maxN = nccN;
              }
              if(anyP) isin[0] = 1;
              isin[1] = maxPidx;
              if(anyN) isin[2] = 1;
              float dN = 1 - maxN;
              float dP = 1 - maxP;
              rsconf = (float)dN/(dN+dP);
              dP = 1 - csmaxP;
              csconf = (float)dN/(dN+dP);
      }
/-----------------------------------------------------------------------------------------------------------------------
```

>>FerNNClassifier.cpp 返回 trainNN()函数的 for 循环

```
        If(y[i]==1&&conf<=thr_nn){                    //thr_nn 在 tld.read()已经读到了
            If( isin[1]<0 ){
            pEx = vector<Mat>(1, nn_examples[i]);     //pEx->FerNNClassifier.h
            continue;}
        pEx.push_back(nn_examples[i]); }
        If(y[i]==0&&conf>0.5)
             nEx.push_back(nn_examples[i]);           //nEx->FerNNClassifier.h
    }
    acum++;      //FerNNClassifier.h->FerNNClassifier.cpp->void FerNNClassifier.prepar()定义了
    printf(....);
```

>>tld.cpp

```
classifier.evaluateTh(nXT, nExT);
```

>>FerNNClassifier.cpp -> void FerNNClassifier::evaluateTh(nXT, nExT)

```
float fconf;
for(int i=0; i<nXT.size(); i++){
    fconf = (float)measure_forest(nXT[i].first) / nstructs; //measure_forest 需要 posteriors,这个
                                                            //已经在 classifier.prepare()函数中计
                                                            //算过了
    if(fconf > thr_fern)
        thr_fern = fconf;
    }
vector<int> isin;
float conf, dummy;
for(int i=0; i<nExT.size(); i++) {
    NNConf(nExT[i], isin, conf, dummy);
    if(conf>thr_nn)
    thr_nn = conf;
```

```
    }
  if(thr_nn>thr_nn_valid)
    thr_nn_valid = thr_nn;          //thr_nn_valid->FerNNClassifier.h->class FerNNClassifier 中
                                    义,tld.read()读取了
}
```

/** ********** Run-Time **********

>>返回到 run_tld.cpp

```
Mat current_gray;
BoundingBox pbox;
vector<Point2f> pts1;
vector<Point2f> pts2;
bool status = true;
int frames1 = 1;
int detections = 1;
REPEAT:
while(capture.read(frame)) {
  cvtColor(frame, current_gray, CV_RGB2GRAY);
  tld.processFrame(last_gray, current_gray, pts1, pts2, pbox, status, t1, bb_file);
```

>>tld.cpp -> void TLD::processFrame(img1, img2, pts1, pts2, bbnext, lastbox_found, t1, bb_file);

```
  vector<BoundingBox> cbb;
  vector<float> cconf;
  int confident_detections = 0;
  int didx;

  /**Track**
  If(last_boxfound && t1) {          //last_boxfound = status
```

track(img1, im2, points1, points2); //img1=last_gray, img2=current_gray, points1=pts1, //points2=pts2

else

tracked = false;

}

>>tld.cpp

>>void TLD::track(img1, img2, points1, points2)

bbPoints(points1, lastbox); //lastbox->TLD.h->class TLD->BoundingBox lastbox; 该值已经被在 //tld.cpp 文件下的 tld.init()函数计算出来，由于 bbPoints 也在 //tld.cpp 中，所以直接调用

>>tld.cpp->bbPoints(points, bb) //points = points1, bb = lastbox

int max_pts = 10;

int margin_h = 0;

int margin_v = 0;

int stepx = ceil(double(bb.width-2*margin_h))/max_pts; //网格均匀撒点

int stepy = ceil(double(bb.height-2*margin_v))/max_pts; //网格均匀撒点

for(int y=..; y < ...; y+=..)

for(int x=..; x<...; x+=)

points.push_back(Point2f(x, y)); //point 在坐标(x,y)，会产生 100 个点

>>tld.cpp

if(points1.size() < 1) {

printf(lastbox(x,y,width, height));

tvalid = false; //TLD.h -> class TLD -> bool tvalid, tacked 定义了

tracked = false;

return;

}

vector<Point2f> points = points1;

tracked = tracker.trackf2f(img1, img2, points, points2);

>>TLD.h->class TLD-> LKTracker tracker; 而 LKTracker 这个类在 LKTracker.h 中定义了

>>LKTracker.cpp->bool LKTracker::trackf2f(img1, img2, points1, points2)

//img1 = last_gray, img2 = current_gray, points1 = pts1, points2 = pts2

calcOpticalFlowPyrLK(img1, img2, points1, points2, status, similarity, window_size, level, term_criteria, lambda, 0);

calcOpticalFlowPyrLK(img2, img1, points2, pointsFB, FB_status, FB_error, window_size, level, term_criteria, lambda, 0);

//status, similarity, windowsize, level, lambda, FB_error, FB_status, term_criteria 来自 LKTracker.h->class LKTracker 的私有成员;在 LKTracker.cpp 中赋值了 term_criteria, lambda, level, window_size,可直接被 LKTracker.cpp 中的所有函数自由调用，而 status, similarity, FB_status, FB_error, pointsFB 是计算得来的

for(int i=0; i<points1.size(); ++i)

{FB_error[i] = norm(pointsFB[i] - points1[i]) }; //FB_error 被赋值了

normCrossCorrelation(img1, img2, points1, points2);

Return filterPts(points1, points2);

>>normCrossCorrelation() //计算得出 patch 10 by 10 个点的 similarity

>>filterPts 得出匹配点返回的 bool 值，匹配到是 true;否则 false.原算法是只要匹配到一个点也算是目标，返回 true

>>TLD.cpp

if(tracked) { //tacked = bool filterPts 的返回值，也就是 tracker.trackf2f(...)的返回值

bbPredict(points, points2, lastbox, tbb); //tbb->TLD.h->class TLD->BoundingBox tbb; 最终该函数将 lastbox 的位置宽高经过传给 tbb

If(tracker.getFB()>10||tbb.x>img2.cols;...超出边界) { //getFB()返回 fbmed 值，它在上面的 //trackf2f()函数中计算出来了，由于 //LKTracker.h 中定义，所以 getFB 直接 //使用

tvalid = false; tacked = false; printf(...too many unstable predictions)

return

}

```
  Mat pattern;
  Scalar mean, stdev;
  BoundingBox bb;
  bb.x/y/width/height  通过 tbb 转换得到 bb 的(x, y, width, height)
  getPattern(img2(bb), pattern, mean, stdev);     //img2=current_gray, pattern/mean/stdev 是
                                                  //计算得来的
  vector<int> isin;
  float dummy;
  classifier.NNConf(pattern, isin, dummy, tconf);    //isin, dummy, tconf 都是计算得来的
  tvalid = lastvalid;                             //TLD.h->class TLD-> bool lastvalid 已经初始化了
  if(tconf>classifier.thr_nn_valid) //tld.init 计算过
     tvalid = true;
  else printf(no points tracked\n);
}

/**Detect**
detect(img2);    //current_gray
>>tld.cpp->void TLD::detect(frame)    //frame=img2=current_gray
    dbb.clear();                    //TLD.h->class TLD->vector<BoundingBox> dbb; .clear()用法在上面
    dconf.clear();                  //同上, vector<float> dconf;
    dt.bb.clear();        //TLD.h->DetStruct dt; struct DetStruct->bb, patt, conf1, conf2, isin, patch;
    double t = (double)getTickCount();
    Mat    img(frame.rows, frame.cols, CV_8U);        //frame = current_gray
    integral(frame, iisum, iisqsum); //iisum, iisqsum 在 tld.init()计算过了
    GaussianBlur(frame, img, Size(9,9), 1.5);
    int numtrees = classifier.getNumStructs();
    float fern_th = classifier.getFernTh();    //tld.read()读取了
    vector<int> ferns(10);
```

```
float conf;
int a = 0;
Mat patch;

for(int i=0; i<grid.size(); i++) {
   //方差分类器
   if(getVar(grid[i], iisum, iisqsum)>=var) {     //var 在 tld.init()中计算过了，已经传递给
                                                  //TLD.h 中的 float var 了
   a++;
   patch = img(grid[i]);
   classifier.getFeatures(patch, grid[i].sidx, ferns);     //ferns 这里被赋值了
   conf = classifier.measure_forest(ferns);
   tmp.conf[i] = conf; //TLD.h->TemStruct tmp->两个成员 patt 和 conf
   tmp.patt[i] = ferns;

   //随机森林分类器
   If(conf>numtrees*fern_th)      //fern_th = thr_fern
      dt.bb.push_back(i);
   Else
   tmp.conf[i] = 0;
   }
}
Int detections = dt.bb.size()
printf(BBox passed variance filter\n);
printf(initial detection from fern classifier);
If(detections>100) {
   nth_element(dt.bb....., OComparator());
   dt.bb.resize(100);
```

```
    detections = 100;
}
If(detections==0) {
    detected = false;    //TLD.h->class TLD->bool detected;已经初始化了
    return;
}
printf(....);
t = (double)getTickCount() - t;

dt.patt=vector<vector<int> >(detections, vector<int>(10,0) );
dt.conf1=...; dt.conf2=...; dt.isin=...; dt.patch=...;
int idx;
Scalar mean, stdev;
float nn_th = classifier.getNNTh();    //nn_th = thr_nn;

//最近邻分类器模块
for(int i=0; i<detections; i++) {
    idx = dt.bb[i];
    patch = frame(grid[idx]);    //frame=curent_gray
    getPattern(patch, dt.patch[i], mean,stdev); //第一个是 current_gray 的 patch;第二个是检
                                                //测到目标区域的 patch
    classifier.NNConf(dt.patch[i], dt.isin[i], dt.conf1[i], dt.conf2[i]);
    dt.patt[i] = tmp.patt[idx];
    If(dt.conf1[i]>nn_th) {
        dbb.push_back(grid[idx]);    //TLD.h->class TLD 中初始化了
        dconf.push_back(dt.conf2[i]);    //同上
    }
}
```

```
If(dbb.size()>0){
printf(Found NN matches);
detected = true;
}
else{
detected = false;
}

/**integration**
If(tracked) {
bbnext = tbb;   //tbb->TLD.h->class TLD, 在 bbPredict()里计算过了; bbnext = pbox 在
                //run_tld.cpp 中的文件中定义了
lastconf = tconf; //track 模块中 getPattern 时计算过，见上面代码;lastconf 这里被覆盖了
lastvalid = tvalid; //同上，lastvalid 值被 tvalid 覆盖了
printf(Tracked);
if(detected) {     //见上面代码
  clusterConf(dbb, dconf, cbb, cconf); //dbb 和 dconf 是 TLD.h 中定义过的；cbb 和 cconf
                                       //是 processFrame()函数新定义的，它俩是保存计
                                       //算结果的地方. clusterConf()是求聚类相似度的地
                                       //方
  for(int i=0; i < cbb.size(); i++) {
    If(bbOverlap(tbb,cbb[i])<0.5 && cconf[i]>tconf ){ //tbb->bbPredict(); cbb->clusterConf
                                                      //由 dbb 聚类计算而来，重合度小于
                                                      //0.5, confident_detections++？Why?
      confident_detections++;
      didx = 1;
      }
    }
```

```
If(confident_detections==1){   //说明重合度太低，丢失了目标
   printf(reinilizing tracking);
   bbnext = cbb[didx]; //聚类得到的框
   lastconf = cconf[didx];
   lastvalid = false;
   }
else {
printf(.... was found);
int cx = 0; cy = 0; cw = 0; ch = 0;
int close_detections = 0;
for(int i=0; i<dbb.size(); i++) {
   If(bbOverlap(tbb, dbb[i])>0.7) {
      cx += dbb[i].x;
      cy += dbb[i].y;
      cw += dbb[i].width;
      ch += dbb[i].x;
      close_detections++;
      printf(weighted detection:....., dbb(x, y, width, height));
      }
   }
   If(close_detections>0) {
      bbnext.x = cvRound( (float)(10*tbb.x + cx)/(float)(10+close_detections) );
      bbnext.y = cvRound( ....tbb+ cx )/(....);
      bbnext.width = cvRound(...tbb+cx)/(....);
      bbnext.height = cvRound(....);
      }
      else{
      printf(....);
```

```
            }
        }
    }
}
else{
printf(Not tracking);
lastboxfound = false;
lastvalid = false;
If(detected) {
clusterConf(dbb, dconf, cbb, cconf);
printf(...);
If(cconf.size() == 1){
    nbnext = cbb[0];
    lastconf = cconf[0];
    printf(....);
    lastboxfound = true;
        }
    }
}
lastbox = bbnext;
If(lastboxfound)
    fprintf(lastbox(x, y, width, height));
else
    fprintf(NaN);
if(lastvalid && t1)    //lastvalid = tvalid = true && t1 =true 时，对目标进行学习
    /**learn**
learn(img2);    //img2 = current_gray
>> tld.cpp -> void TLD::learn(img)    //img = img2 = current_gray
```

```
printf(learing);
BoundingBox bb;
bb.x = max(lastbox.x, 0);
bb.y = max(lastbox.y, 0);
bb.with, bb.height = ....lastbox;    //lastbox=bbnext=....dbb....tbb 上面计算而来
Scalar mean, stdev;
Mat pattern;
getPattern(img(bb), pattern, mean, stdev);    //pattern, mean, stdev 都是计算结果
vector<int> isin;
float dummy, conf;
classifier.NNConf(pattern, isin, conf, dummy);    //isin, conf, dummy 都是结算结果，用来计
                                                  //算输入图像和在线模型之间的相似度
                                                  //conf
If(conf<0.5){
printf(fast change, not training);
lastvalid = false;
return;
}
If(pow(stdev.val[0], 2) < var) {    //.var 是 CV 自带函数，var 在 tld.init()算过
printf(low variance, not training);
lastvalid = false;
return;
}
If(isin[2] == 1) {
printf(patch in negetive data, not training);
lastvalid = false;
}
for(int i=0; i<grid.size(); i++){
grid[i].overlap = bbOverlap(lastbox, grid[i]);
```

```
    }

    vector<pair<vector<int>, int> > fern_examples;
    good_boxes.clear();
    bad_boxes.clear();
    getOverlappingBoxes(lastbox, num_closest_update);    //num_cloest_update 已经 tld.read()
                                                         //返回 good_boxes 到 TLD.h 中
    If(good_boxes.size()> 0)
       generatePositiveData(img, num_warps_update); //img = current_gray
    else {
       lastvalid = false;
       printf(No good boxes, not training);
       return;
    }
    fern_examples.reserve(pX.size() + bad_boxes.size() );
    fern_examples.assign(pX.begin(), pX.end());
    int idx;
    for(int i = 0; i<bad_boxes.size(); i++){
       idx = bad_boxes[i];
       If(tmp.conf[idx] >= 1) {
       fern_examples.push_back(make_pair(tmp.patt[idx], 0));
       }
    }
    classifier.trainF(fern_examples, 2);
    classifier.trainNN(nn_examples);
    classifier.show(); //FerNNClassifier.cpp 定义了 show()函数，用来显示图像的
>>run_tld.cpp
  If(status) {    //status 是 tld.processFrame()计算出来的
```

```
drawPoints(frame, pts1); //frame 当前一帧的彩色图像,缺少一个 color 的参数，默认白色
>>tld.cpp->drawPoints(image, points, color)   //image=frame, points=pts1，如有 color 就用
            for(iterator i=points.begin(), ie=points.end(),; i!=ie; ++i) {
              Point center(cvRound (i->x), cvRound( i->y ) );
              circle(image, *1, 2, color, 1); //color 默认是白色的
              }
drawPoints(frame, pts2, Scalar(0, 255, 0) );
drawBox(frame, pbox);   //pbox 是计算出来的,在 tld.processFrame()里是 bbnext,计算过了
detections++; //检测到目标，计数器加 1

imshow(frame); //显示彩色帧图，drawPoints，框都会有

swap(last_gray, current_gray);
pts1.clear();
pts2.clear();
frames++; //帧数加 1
printf(....);
If(cvWaitKey(33) == 'q')
   break;
}
If(rep) {   //如果重复第一帧，rep 由 true 转换成 false
rep = false;
t1 = false;
fclose(bb_file);
bb_file = fopen(final_detector.txt, w);

capture.release(); //释放这一重复的一帧
capture.open(video); //重新打开 video，重新读取
```

```
    goto REPEAT; //c++自带语法
    }
    fclose(bb_file); //程序运行完，数据写完，关闭文档，释放内存
    return 0;
}
```

注意：1 tld.cpp 文件下的 clusterBB()函数自始至终没有使用过，可以删除；

2 tld.cpp 文件下的 bbox_step = 7 自始至终没有使用过，可以删除；