陕西师範大學
SHAANXI NORMAL UNIVERSITY

交叉性物理实验报告

题目：人脸追踪相关参数探究

作者单位 物理学与信息技术学院

作者姓名 刘枭

专业班级 物理学(创新实验班)1701

作者学号 41706278

指导教师 高翔

2021 年 1 月

摘要

摘　要

在基于人工智能中的表情识别算法实时预测教学效果的研究中，研究目的是帮助教师教学更好的了解学生的知识学习掌握效果，研究方法是通过一些有代表性的课前课后习题的检测结果结合学生在学习过程中的面部微表情判断学生是否已经掌握了这门知识。这个实验主要包括五个部分：课前和课后小组测验的方式来评价教学效果；真实课堂录像仪器的设计；教学录像处理：视频选取图像；各平台的情感识别算法、结果与定价的比较；建立测验得分与表情情感值之间的相关关系 。

该实验主要负责各平台人脸识别算法、结果与定价结果的比较以及对教学片段的批量表情处理。首先，通过查阅网络资料，获取微软、商汤、依图、云从、旷世、百度国内外六家知名的 AI 平台 ，经过表情信息、价格等多方面因素后选取旷世作为本次实验的测试平台。其次，通过调用旷世平台的 Detect API 实现了多文件夹、多图片上传及表情信息的批量获取。最后，将生成表情数据进一步处理，利用绘图软件绘制出学生上课时表情值随时间变化的规律曲线。

一、 人脸追踪

在海关、机场、银行、电视电话会议等场合，都需要对特定人脸目标进行跟踪。显然，要跟踪图象中的人脸。首先要识别人脸。人脸识别就是利用计算机分析静态图片或视频序列。从中找出人脸并输出人脸的数目、位置及其大小等有效信息。其次就是跟踪人脸。就是要在检测到人脸的前提下。在后续帧中继续捕获人脸的位置及其大小等信息。人脸跟踪技术涉及到模式识别、图象处理、计算机视觉、生理学、心理学及形态学等诸多学科。并与基于其它生物特征的身份鉴别方法以及计算机人机感知交互的研究领域密切相关。与基于指纹、视网膜、虹膜、DNA 等其它人体生物特征识别系统相比。人脸跟踪技术更为直接、友好。不会对用户造成心理障碍。此外，人脸跟踪技术研究与相关学科的发展及对人脑的认识程度紧密相关。这诸多因素使人脸跟踪研究成为一项既困难又极富挑战性的课题。目前常见的跟踪技术大致可分为 4 大类：基于模型跟踪；基于运动信息跟踪；基于人脸局部特征跟踪和基于神经网络跟踪等方法。

二、卡尔曼滤波法：

卡尔曼滤波模型理论建立在线性代数和隐含马尔可夫模型。其基本动态系统可以用一个马尔可夫链表示，该马尔可夫链建立在一个被高斯噪声(即正态分布的噪声)干扰的线性算子上的。系统的状态可以用一个元素为实数的向量表示。随着离散时间的每一个增加,这个线性算子就会作用在当前状态上，产生一个新的状态,并也会带入一些噪声,同时系统的一些已知的控制器的控制信息也会被加入。同时，另一个受噪声干扰的线性算子产生出这些隐含状态的可见输出。
简单来讲，卡尔曼滤波器就是根据上一时刻的状态，预测当前时刻的状态，将预测的状态与当前的时刻的测量值进行加权，加权的结果认为当前的实际状态，五个卡尔曼迭代方程为：

两个时间更新方程为：

$$\bar{x_t} = Ax_{t-1} + Bu_{t-1} \quad (1)$$

$$\bar{P_t} = AP_{t-1}A^T + Q \quad (2)$$

$\bar{x_t}$:当前时刻的估计值;A:为当前时刻的状态转移矩阵；x_{t-1}：上一时刻的最优估计值；B：当前时刻的控制矩阵；u_t：当前时刻的控制量；P_{t-1}：上一时刻预测值的协方差矩阵；Q：当前时刻测量值的噪声矩阵。

三个状态更新矩阵为：

$$K_t = \bar{P_t}C^T(C\bar{P_t}C^T + R)^{-1} \quad (3)$$

$$x_t = \bar{x_t} + K_t(y_t - C\bar{x_t}) \quad (4)$$

$$P_t = (I - K_tC)\bar{P_t} \quad (5)$$

C:测量值和状态的比例系数；R：测量值的协方差；K；Kalman 增益；

通过五个方程的反复迭代即可获得当前的状态，并且预测下一时刻的状态。

三、程序分析

程序来源：https://github.com/bahramlavi/FaceDetectionTracking

3.1 人脸识别程序：

Matlab 程序：Vision.CascadeObjectDetector

在人脸检测中，Viola-Jones 算法是一种非常经典的算法，该算法在 2001 年的 CVPR 上提出，因其高效快速的检测而被广泛使用。这个算法用来检测正面的

人脸图像，对于侧脸图像的检测不是很稳健。算法可以被分为以下几个部分：1.利用 Haar 特征描述人脸特征。2.建立积分图像。3.利用该图像快速获取几种不同的矩形特征。4.利用 Adaboost 算法进行训练建立层级分类器。5.非极大值抑制。

3.2 追踪程序分析：

（1）创建一个轨迹，数据结构为结构体。

```
function tracks = initializeTracks()
        tracks = struct(...
            'id', {}, ...
            'bbox', {}, ...
            'kalmanFilter', {}, ...
            'age', {}, ...
            'totalVisibleCount', {}, ...
            'consecutiveInvisibleCount',{},...
            'DTime',{});
```

id 为轨迹的数目。bbox 为当前对象的边界框（[左上角像素点的横坐标，左上角像素点的纵坐标，边界框的宽，边界框的高]）。kalmanFilter：基于运动的跟踪的卡尔曼滤波器对象。age：首次检测到轨道以来的帧数。totalVisibleCount：检测到轨道的帧总数（可见）consecutiveInvisibleCount：未检测到轨道的连续帧数（不可见）。

(2)读取视频，设置播放器，设置人脸识别算法（Viola-Jones）,通过 blob 分析检测连通域，得到对应的移动目标。

```
function obj = initializeObjects()
       obj.reader = VideoReader('A:\编程\人脸追踪\video\M_1080_60S.mp4')

      obj.videoPlayer = vision.VideoPlayer;

      obj.blobAnalyser =vision.CascadeObjectDetector;

       obj.blobAnalyser.MinSize=[20 20];
       end
```

```
function bboxes = detectObjects(frame)

        bboxes = obj.blobAnalyser.step(frame);
end
```

Vision.CascadeObjectDetector 为 Matlab 中 Viola-Jones 人脸识别函数，通过眼睛和鼻子识别人脸，在该追踪程序中识别效果较差。

（3）通过卡尔曼滤波器预测轨迹

```
function TracksByPredict()
       for i = 1:length(tracks)
           bbox = tracks(i).bbox;

           predictedCentroid = predict(tracks(i).kalmanFilter);
           predictedCentroid = predictedCentroid - bbox(1:4) / 2;

           tracks(i).bbox = [predictedCentroid, bbox(3:4)];
       end
end
```

通过卡尔曼滤波器预测中心位置，并进行调整，将结果作为轨迹的跟踪矩形框。

（4）分配新检测目标给轨迹

```
function [assignments, unassignedTracks, unassignedDetections] = ...
        detectionToTrackAssignment()

    nTracks = length(tracks);
    nDetections = size(bboxes, 1);
    cost = zeros(nTracks, nDetections);
    for i = 1:nTracks
        cost(i, :) = distance(tracks(i).kalmanFilter, bboxes);
    end
    %解决分配问题
    costOfNonAssignment = 20;
    [assignments, unassignedTracks, unassignedDetections] = ...
        assignDetectionsToTracks(cost, costOfNonAssignment);
end
```

获得轨迹的个数和新检测目标的个数。创建损失函数矩阵行代表轨迹，列代表新检测目标。对每个轨迹使用它们的卡尔曼滤波器的结果，与每个新检测的目标中心计算欧几里得距离，存入损失函数矩阵中。设置阈值costofNonAssignment。函数assignDetectionsToTracks(cost, costOfNonAssignment)是一种匈牙利匹配算法，根据损失函数 cost 和阈值costofNonAssignment 分配好轨迹和检测目标。分类为：已分配的轨迹、未分配的轨迹、未分配的检测轨道。

（5）更新已分配的轨迹

根据轨迹对应的检测目标位置中心修正。如果检测到目标在该帧中被检测到，则进行修正（correct）。如果没有被检测到，则使用上一帧的结果进行卡尔曼滤波器预测(predict)，当作该帧中人脸的位置。并且将可见帧数加 1，不可见帧数归零。

```
function updateAssignedTracks()
        numAssignedTracks = size(assignments, 1);
        for i = 1:numAssignedTracks
            trackIdx = assignments(i, 1);
            detectionIdx = assignments(i, 2);


            bbox = bboxes(detectionIdx, :);
            if ~isempty(bbox)
                %bbox=predict(tracks(trackIdx).kalmanFilter);
                bbox=correct(tracks(trackIdx).kalmanFilter, bbox);
            else
                 bbox= predict(tracks(trackIdx).kalmanFilter);
            end
            tracks(trackIdx).bbox = bbox;%使用新的矫正更新位置

            tracks(trackIdx).age = tracks(trackIdx).age + 1;
            %更新可见数量
            tracks(trackIdx).totalVisibleCount = ...
                tracks(trackIdx).totalVisibleCount + 1;
            tracks(trackIdx).consecutiveInvisibleCount = 0;

        end
end
```

（6）更新未分配的轨迹

```
function updateUnassignedTracks()%更新未分配轨迹
      for i = 1:length(unassignedTracks)
          ind = unassignedTracks(i);
          tracks(ind).age = tracks(ind).age + 1;%轨迹年龄+1，连续不可见帧数+1
          tracks(ind).consecutiveInvisibleCount = tracks(ind).consecutiveInvisibleCount + 1;
      end
end
```

（7）删除丢失的轨迹

```
function deleteLostTracks()
    if isempty(tracks)
        return;
    end

    invisibleForTooLong =10;%当连续不可见帧数大于10就丢弃轨迹
    ageThreshold = 100;%当轨迹的年龄小于100时，更据总可见帧数与年龄的比值丢弃轨道。
    %否则根据连续不可见帧数丢弃轨道

    ages = [tracks(:).age];%整合为列向量
    totalVisibleCounts = [tracks(:).totalVisibleCount];
    visibility = totalVisibleCounts ./ ages;
    %查找“丢失”轨迹的索引
    lostInds = (ages < ageThreshold & visibility < 1) | [tracks(:).consecutiveInvisibleCount] >= invisibleForTooLong;
    tracks = tracks(~lostInds);
end
```

设置阈值 invisibleForTooLong 当连续不可见帧数大于该值就删除轨迹 ageThreshold =当轨迹的年龄小于该值时，跟据总可见帧数与年龄的比值丢弃轨道。否则根据连续不可见帧数删除轨道。

（8）创建新轨迹。

```
function addNewTracks()
    %对未分配的检测创建新轨迹
    bboxes = bboxes(unassignedDetections, :);

    for i = 1:size(bboxes, 1)
        tic
        bbox = bboxes(i, :);


        %动态模型：匀加速 初始化位置 初始化估计误差（位置误差、速度误差）动态噪声（位置误差，速度误差）测量噪声
        %kalmanFilter = configureKalmanFilter('ConstantVelocity', bbox, [200, 50], [100, 25], 10);
%
        kalmanFilter = configureKalmanFilter('ConstantAcceleration', bbox, [250 250 50], [250 100 100], 100);

        dtime=toc;
        newTrack = struct(...
            'id', nextId, ...
            'bbox', bbox, ...
            'kalmanFilter', kalmanFilter, ...
            'age', 1, ...
            'totalVisibleCount', 1, ...
            'consecutiveInvisibleCount',0,...
            'DTime',dtime);

        tracks(end + 1) = newTrack;%将其添加到现在的轨迹阵列

        nextId = nextId + 1;
    end
end
```

对于新检测的目标要创建一个新的轨迹，并设置卡尔曼滤波器（configureKalmanFilter），通过实验对比，匀加速模型更适合课堂环境。

（9）显示跟踪追踪结果

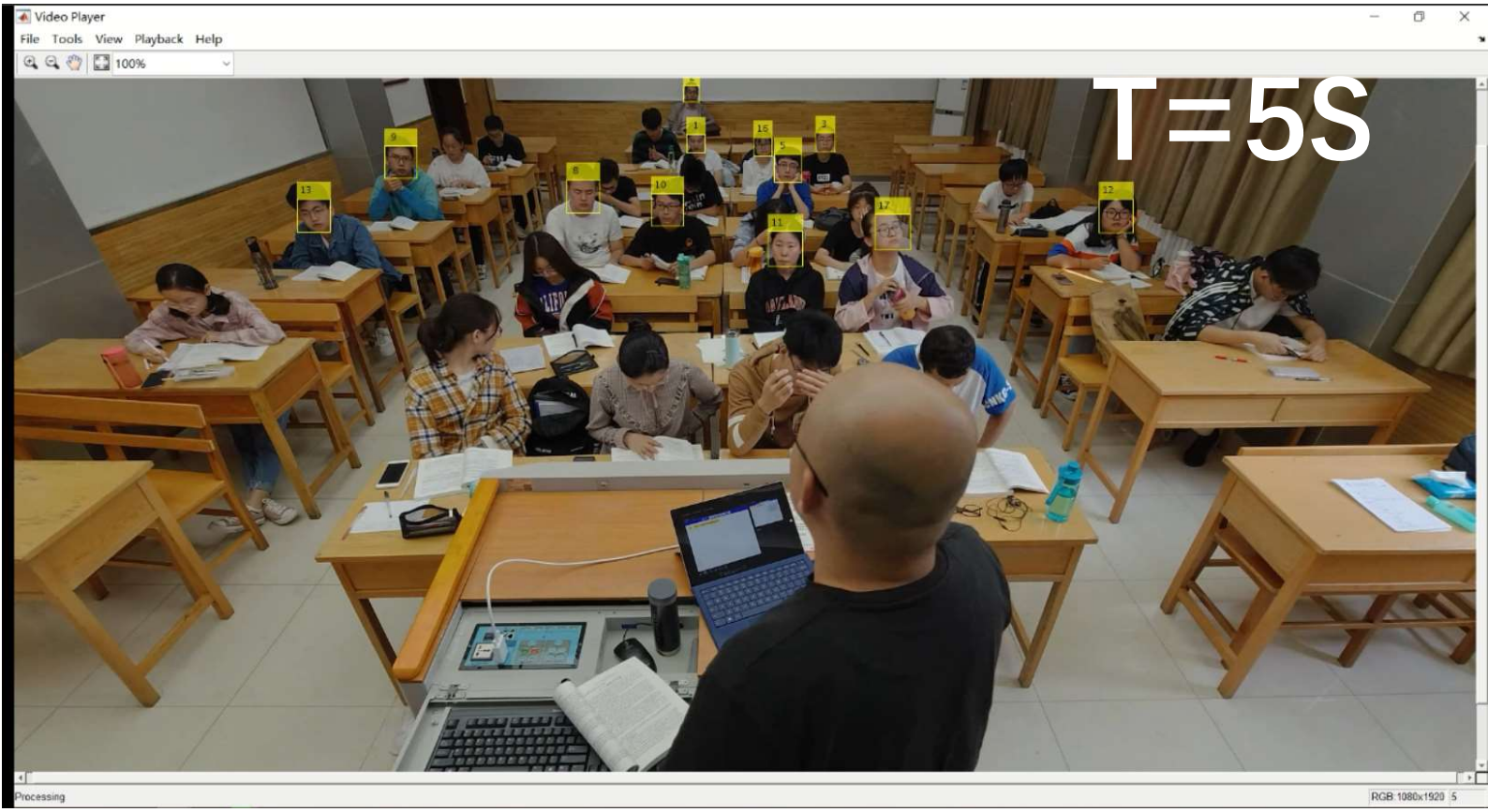
```
function displayTrackingResults()
          frame = im2uint8(frame);

        minVisibleCount = 4;

        if ~isempty(tracks)

            reliableTrackInds = [tracks(:).totalVisibleCount] > minVisibleCount;
            reliableTracks = tracks(reliableTrackInds);

            if ~isempty(reliableTracks) %&& reliableTracks.islost==0
                    bboxes = cat(1, reliableTracks.bbox);
                    ids = int32([reliableTracks(:).id]);
                    labels = cellstr(int2str(ids'));
                   predictedTrackInds =[reliableTracks(:).consecutiveInvisibleCount] > 0;
                    isPredicted = cell(size(labels));
                    isPredicted(predictedTrackInds) = {'predicted'};

                    frame = insertObjectAnnotation(frame, 'rectangle',bboxes,labels);
            end
        end

        obj.videoPlayer.step(frame);
```

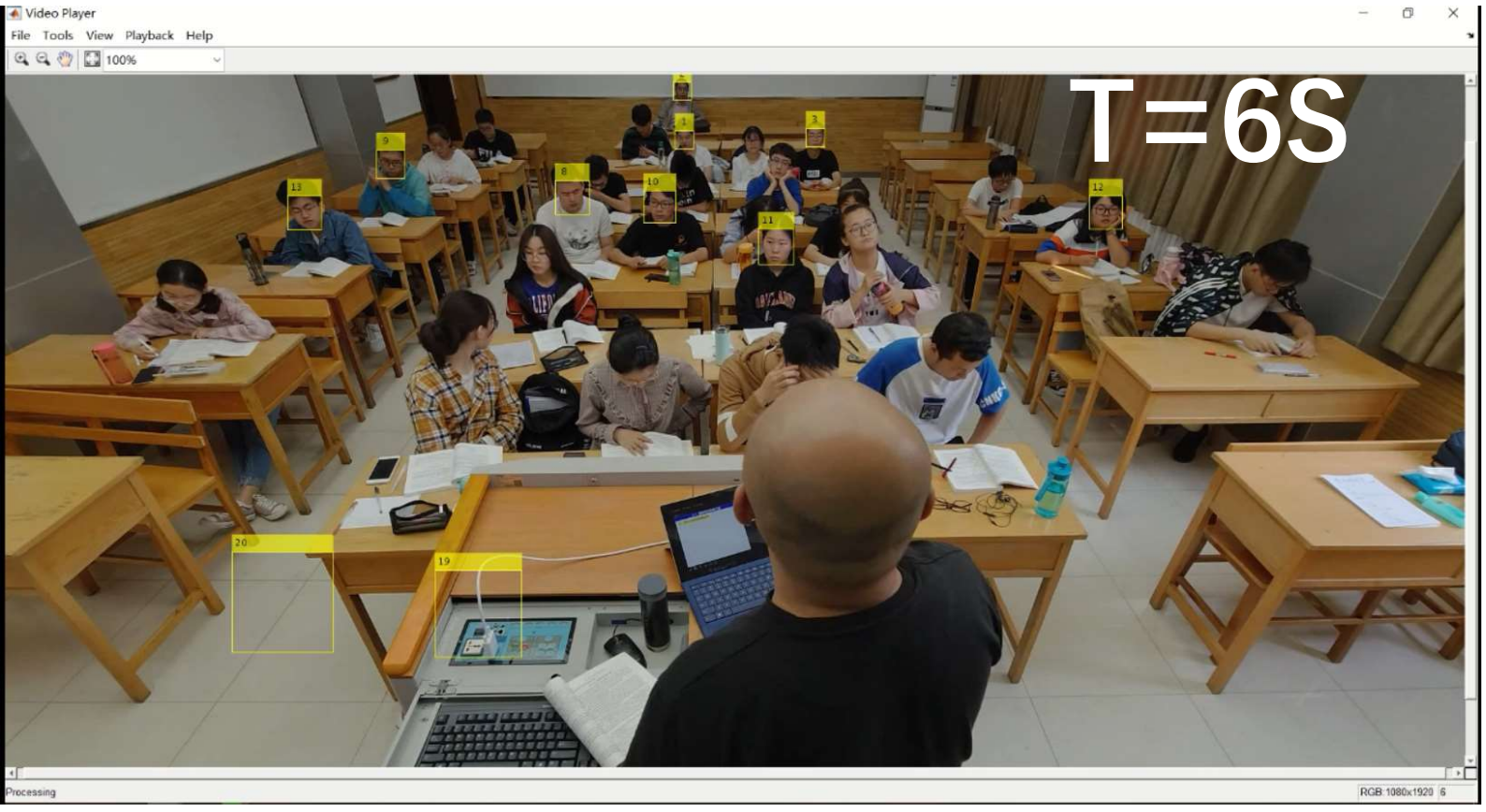
minVisibleCount 的设置可以防止噪声产生，设置为 2~4 效果较好。

（10）读取下一帧，进行循环

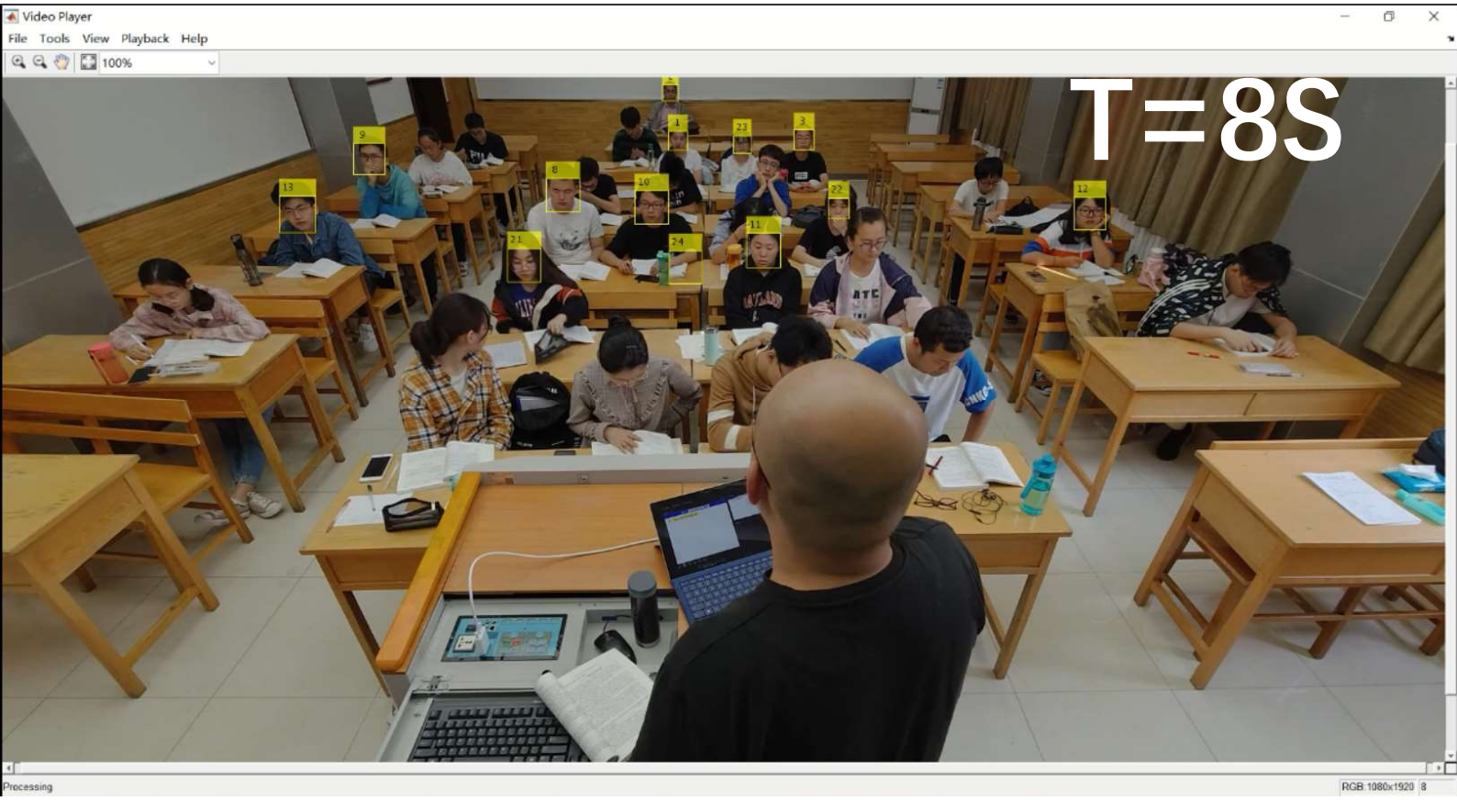
四、实验结果

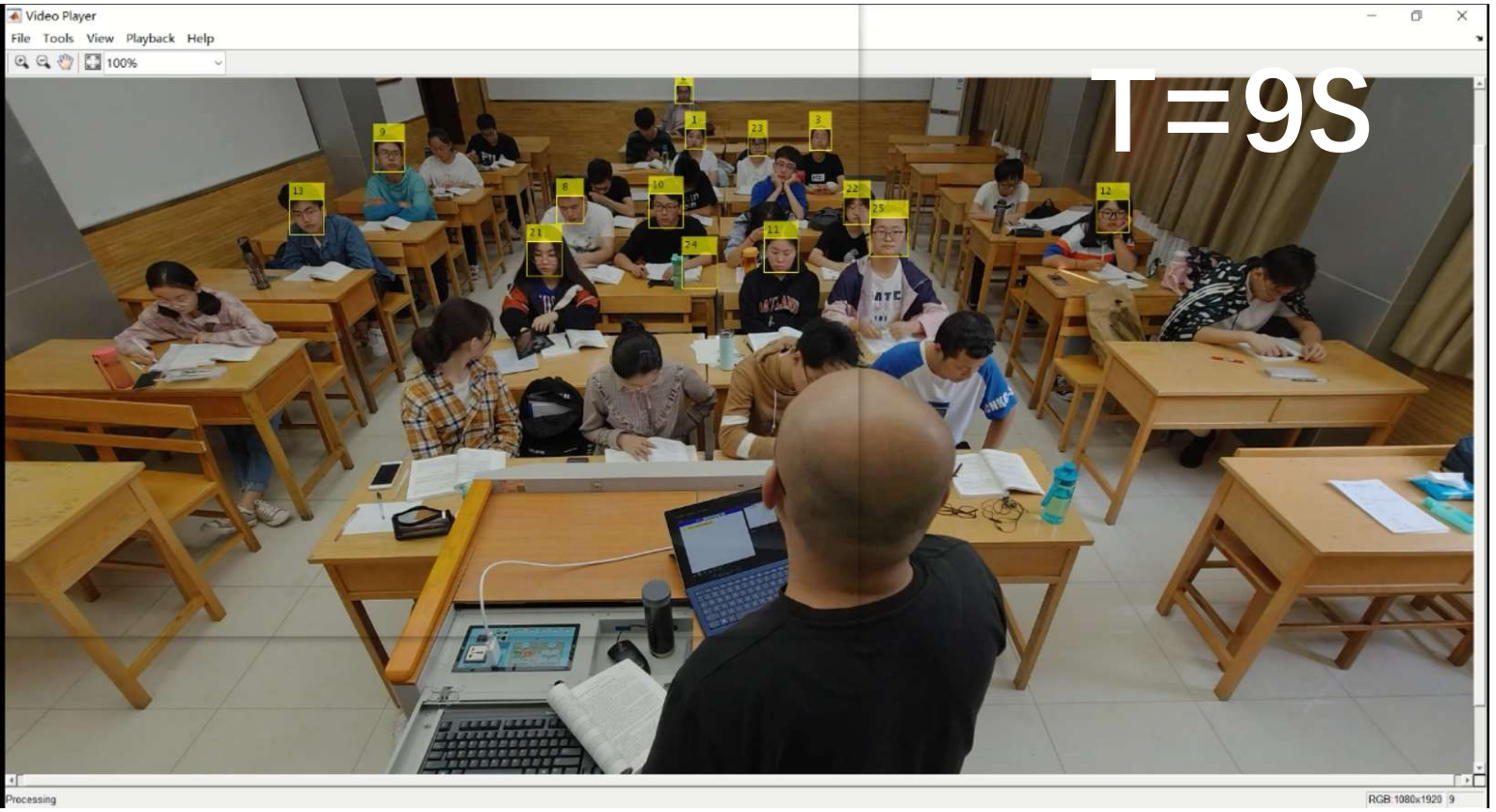
4.1 课堂人脸追踪效果展示

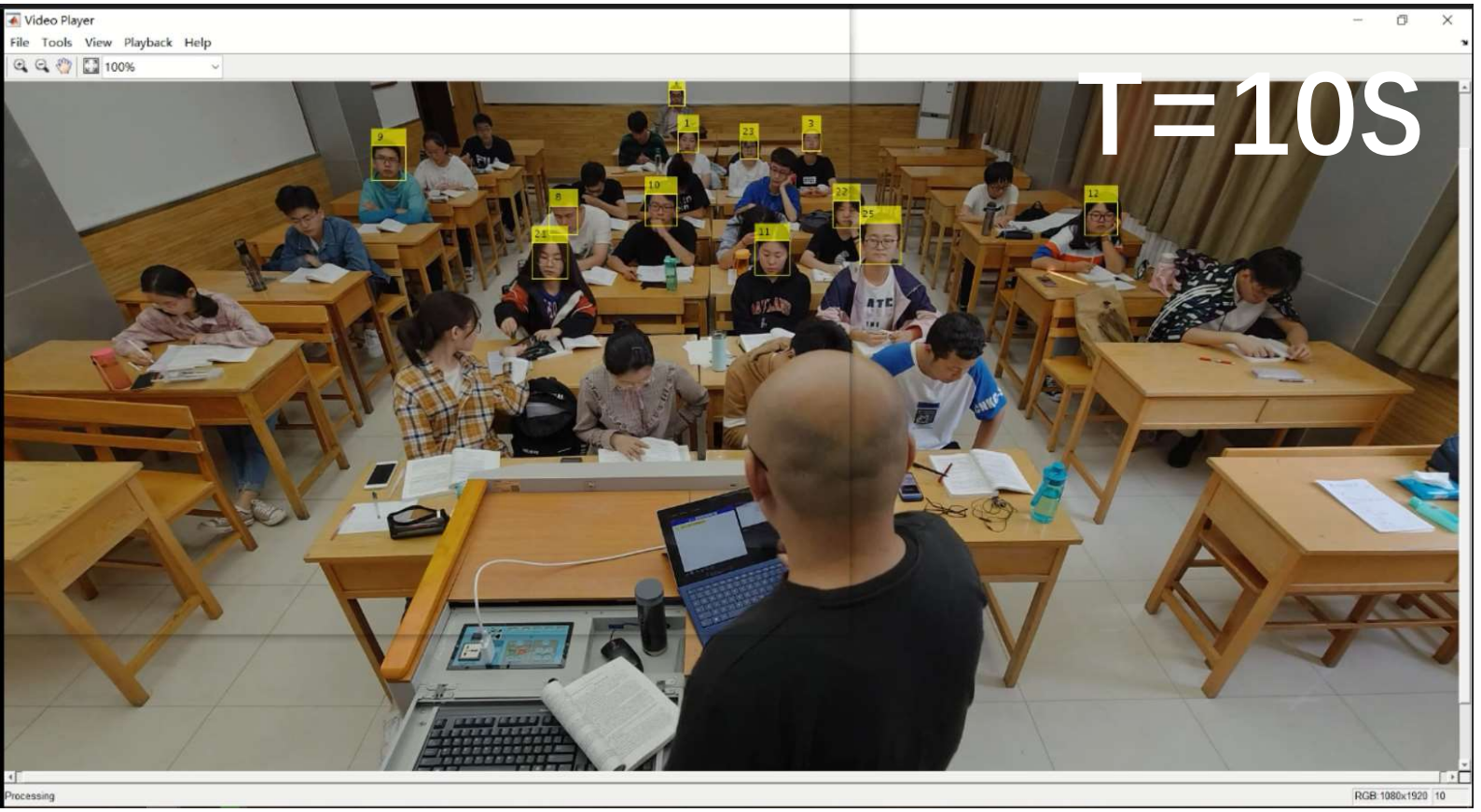

Video Player
File Tools View Playback Help
100%
T=5S
Processing
RGB 1080x1920 5

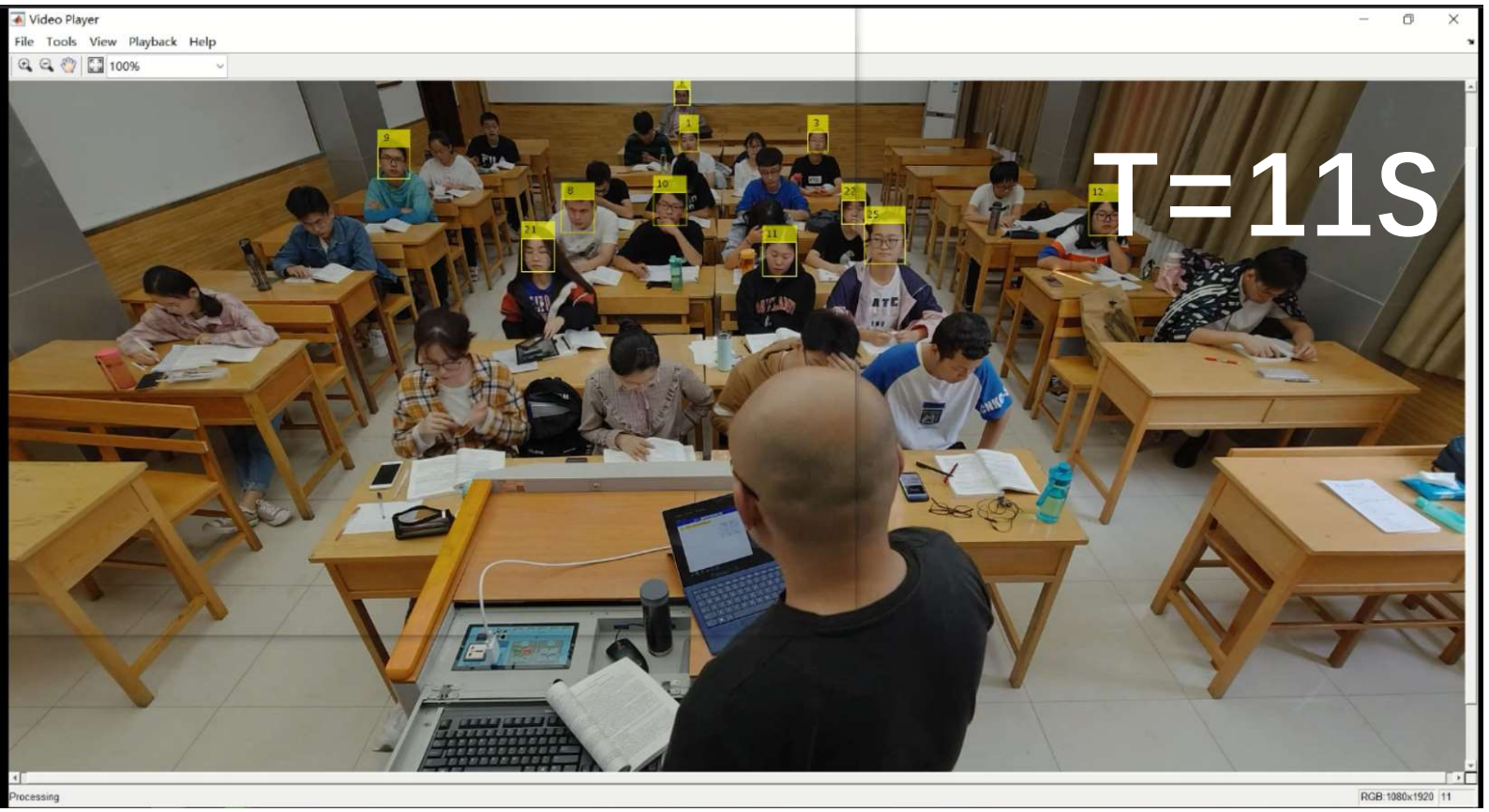

Video Player
File Tools View Playback Help
100%
T=6S
Processing
RGB 1080x1920 6

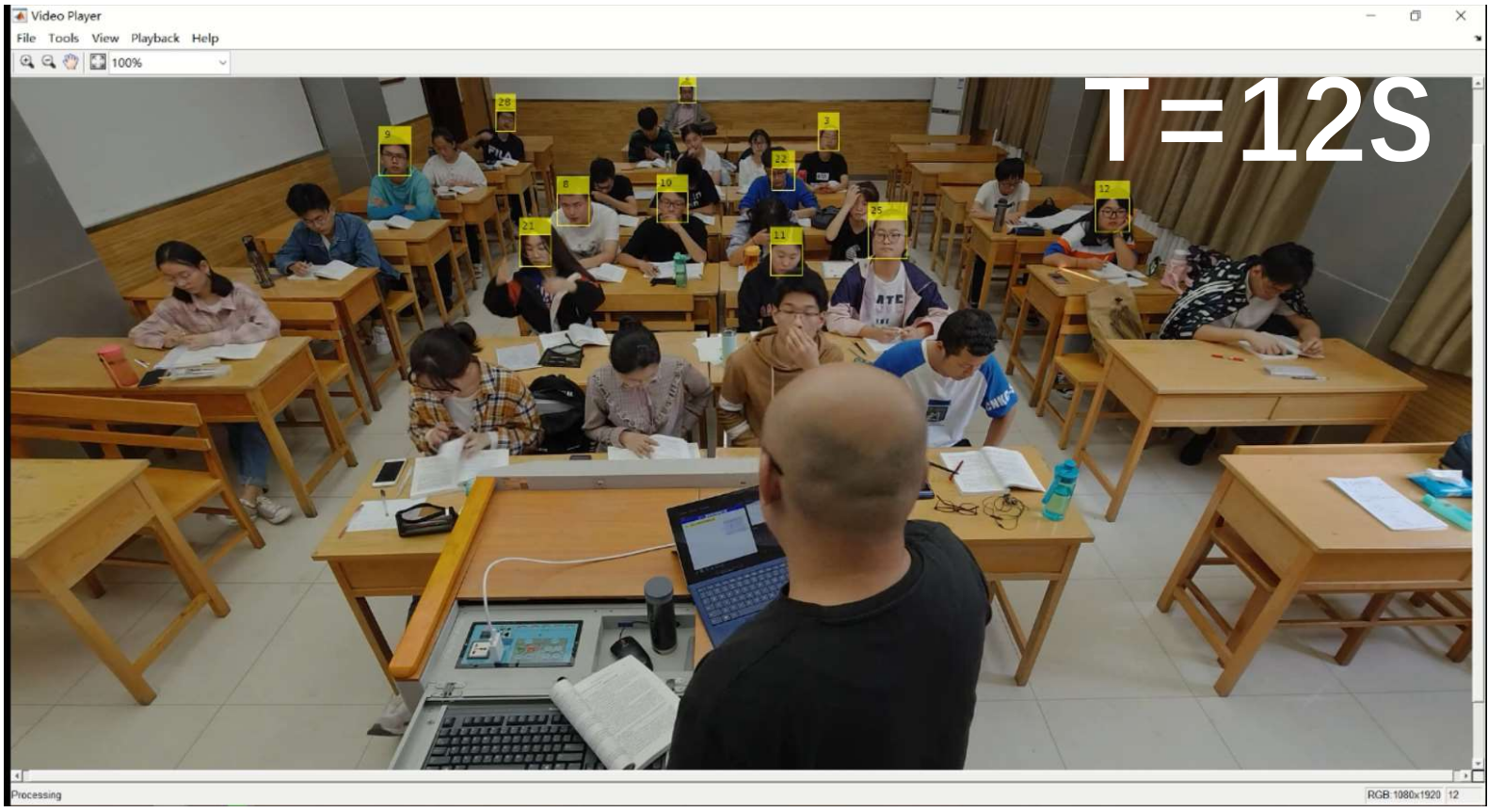

Video Player
File Tools View Playback Help
100%
T=7S
Processing
RGB 1080x1920 7

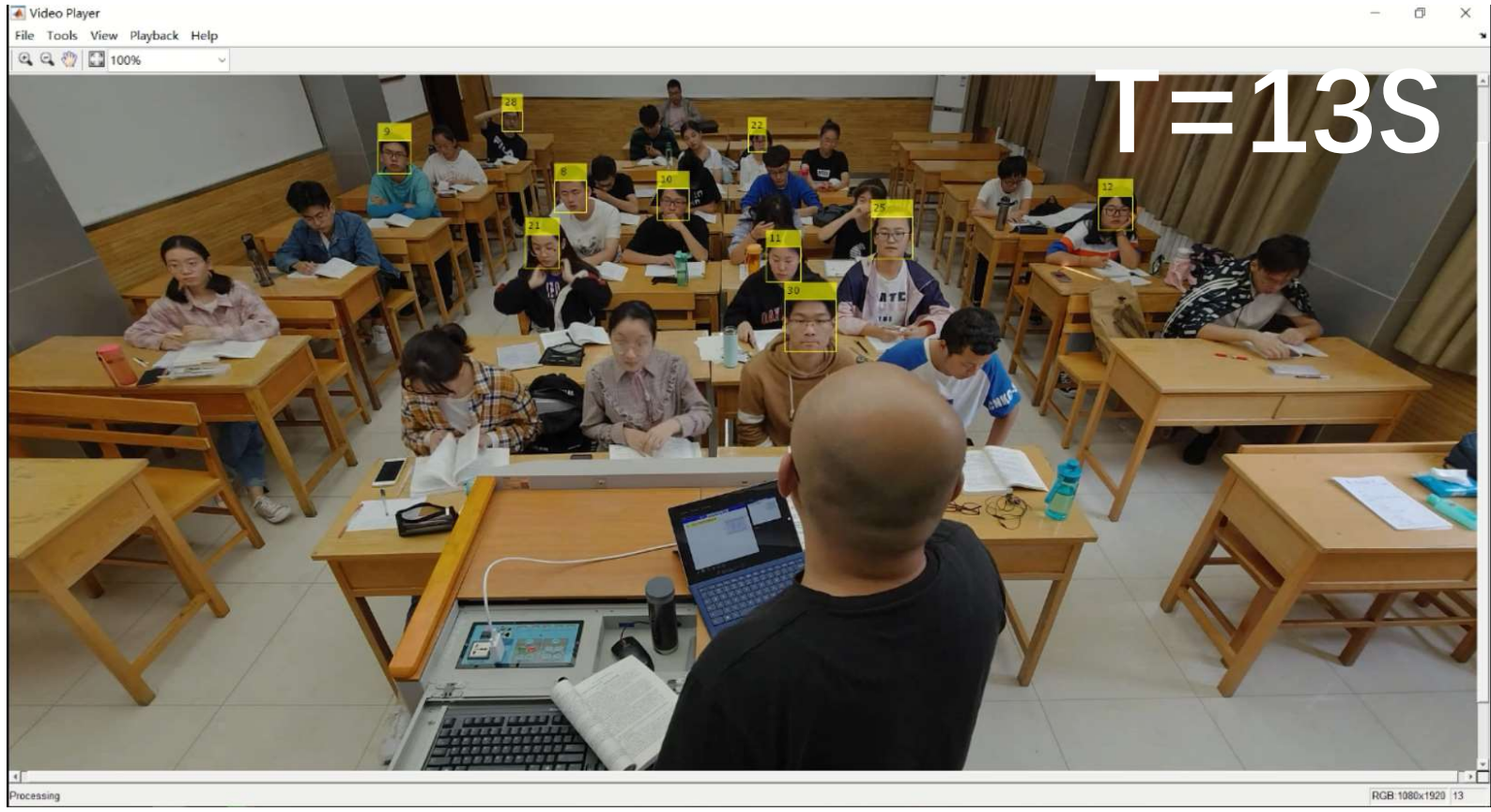

Video Player
File Tools View Playback Help
100%
T=8S
Processing
RGB 1080x1920 8

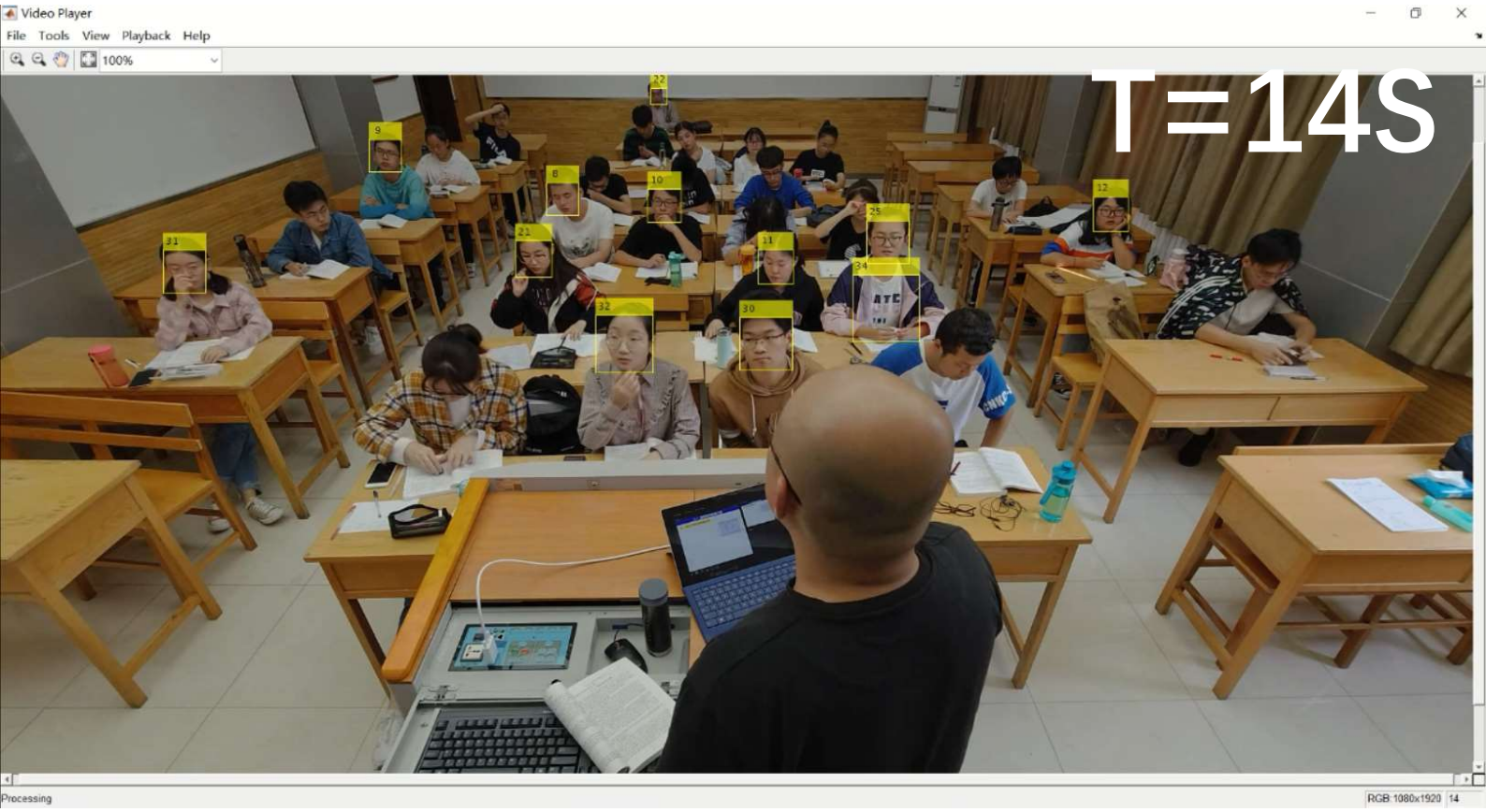

Video Player
File Tools View Playback Help
100%
T=9S
Processing
RGB 1080x1920 9

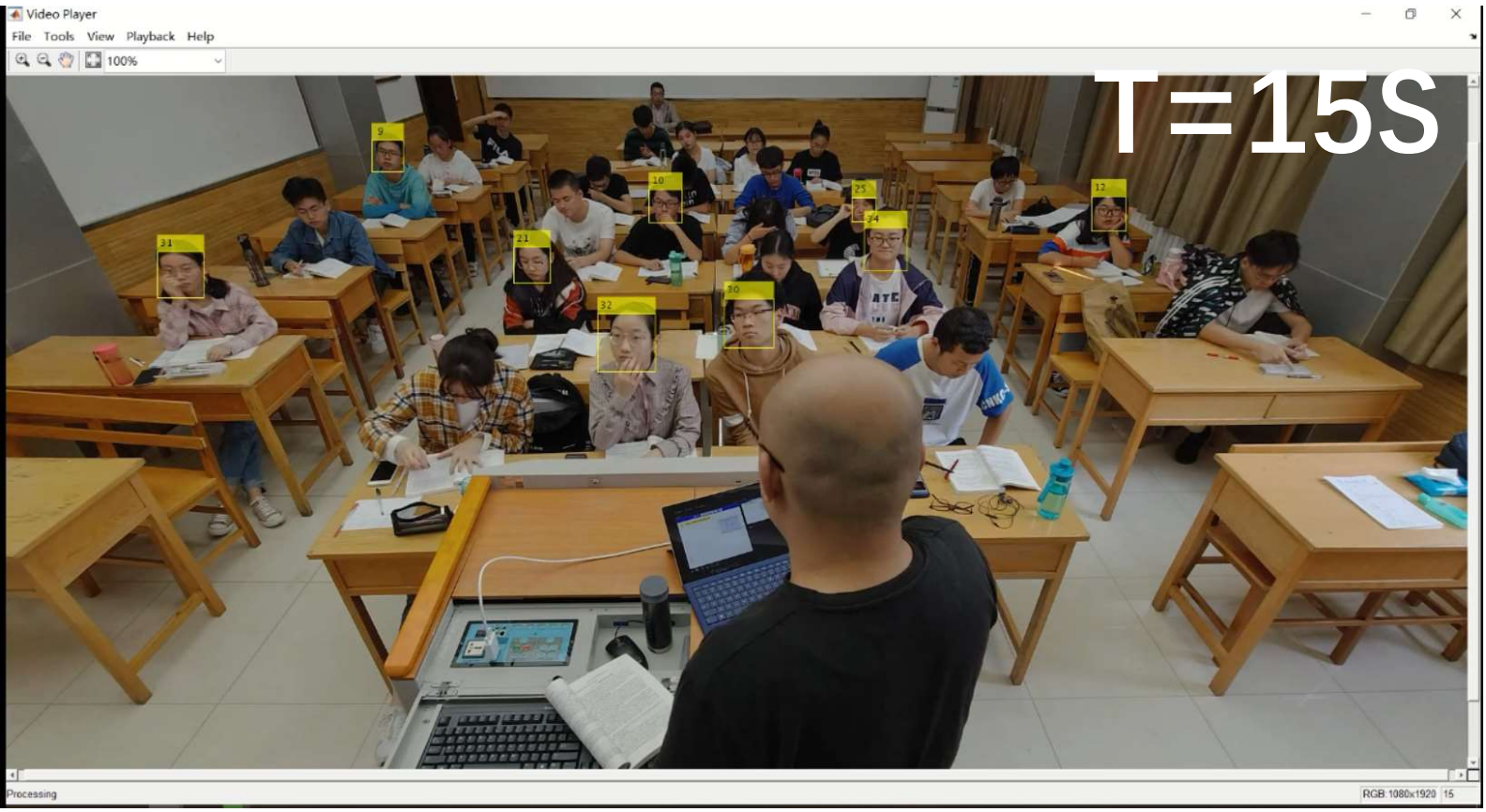

Video Player
File Tools View Playback Help
100%
T=10S
Processing
RGB 1080x1920 10


Video Player
File Tools View Playback Help
100%
T=11S
Processing
RGB 1080x1920 11


Video Player
File Tools View Playback Help
100%
T=12S
Processing
RGB 1080x1920 12


Video Player
File Tools View Playback Help
100%
T=13S
Processing
RGB 1080x1920
13


Video Player
File Tools View Playback Help
100%
T=14S
Processing
RGB 1080x1920
14

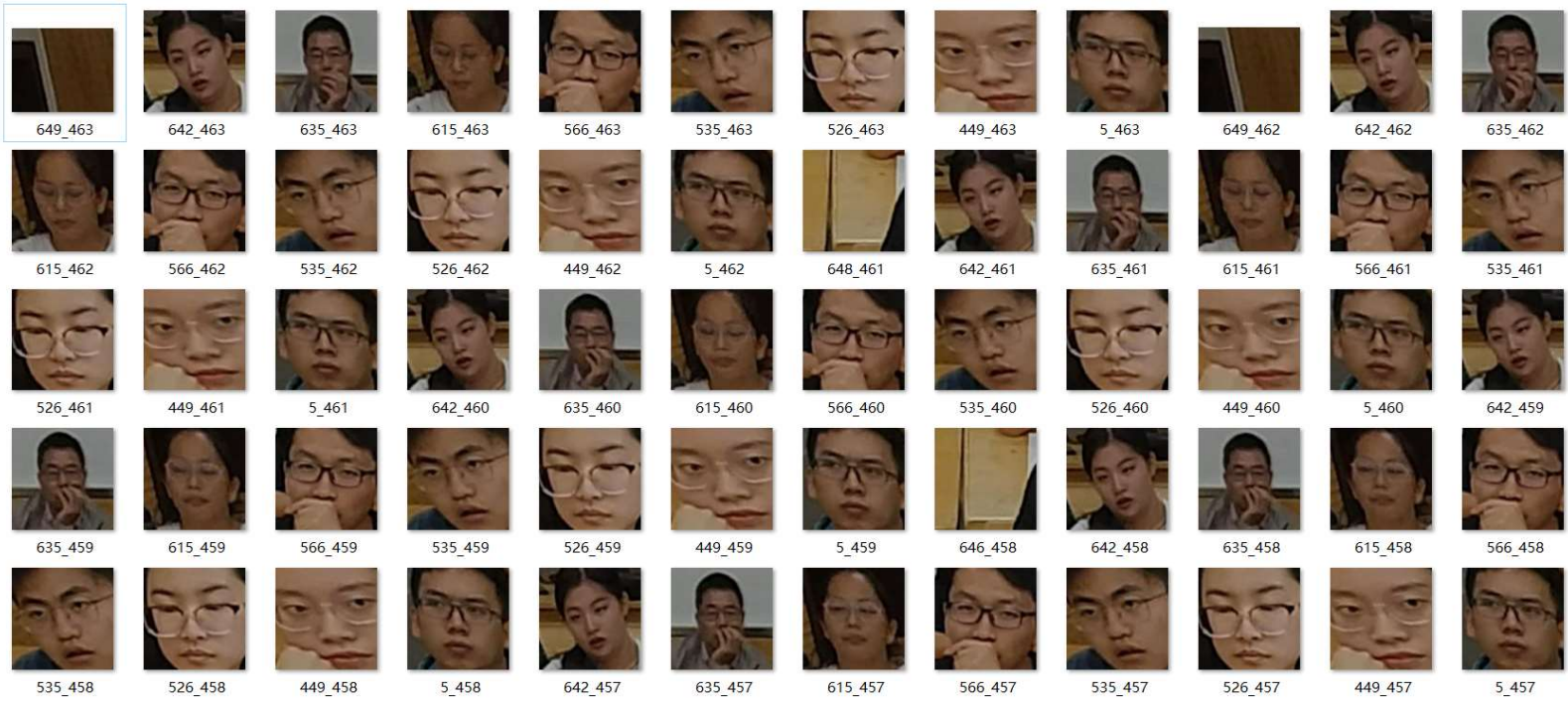


4.2 人脸追踪下对人脸的裁剪

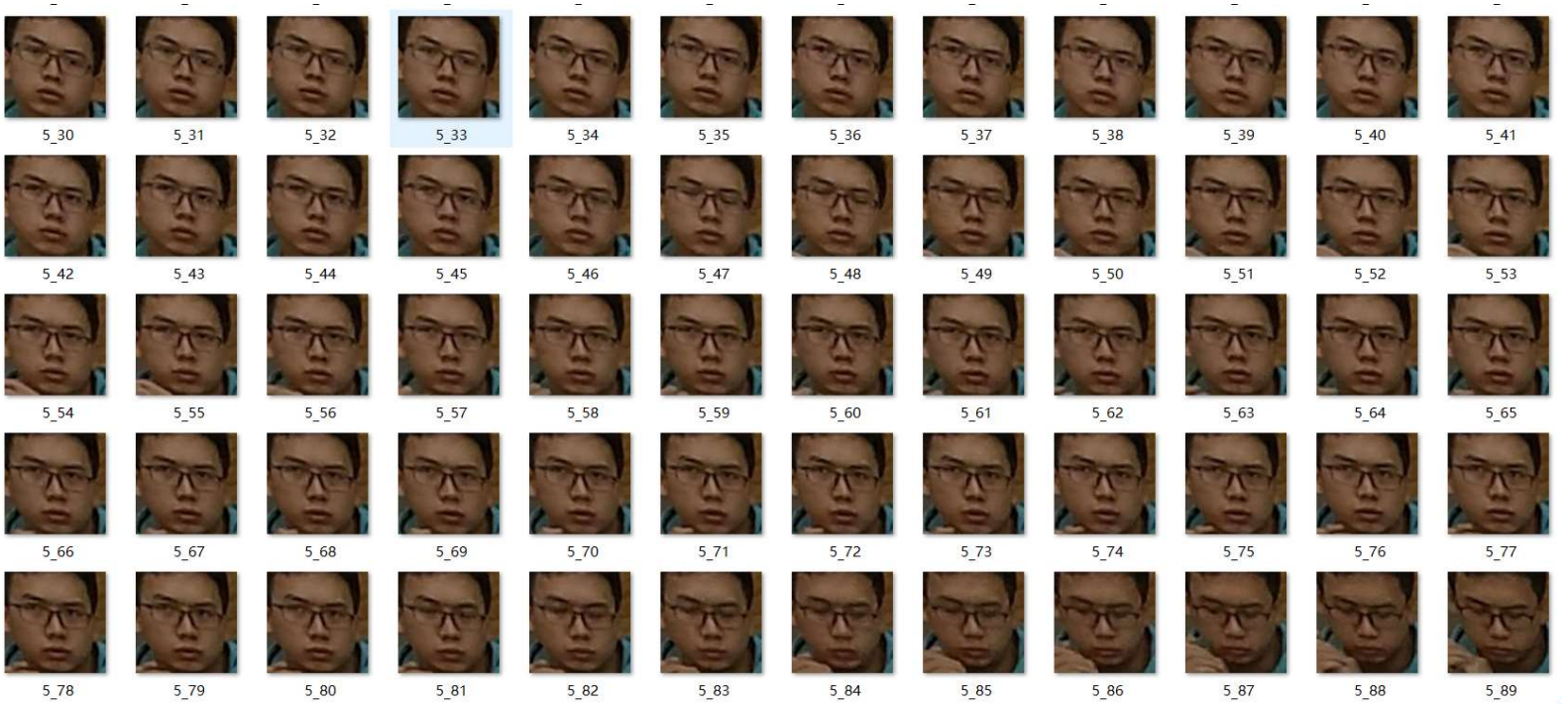
使用 imcrop 函数对每帧的人脸进行裁剪并保存命名。命名方式为（标签_视频此时的帧数）

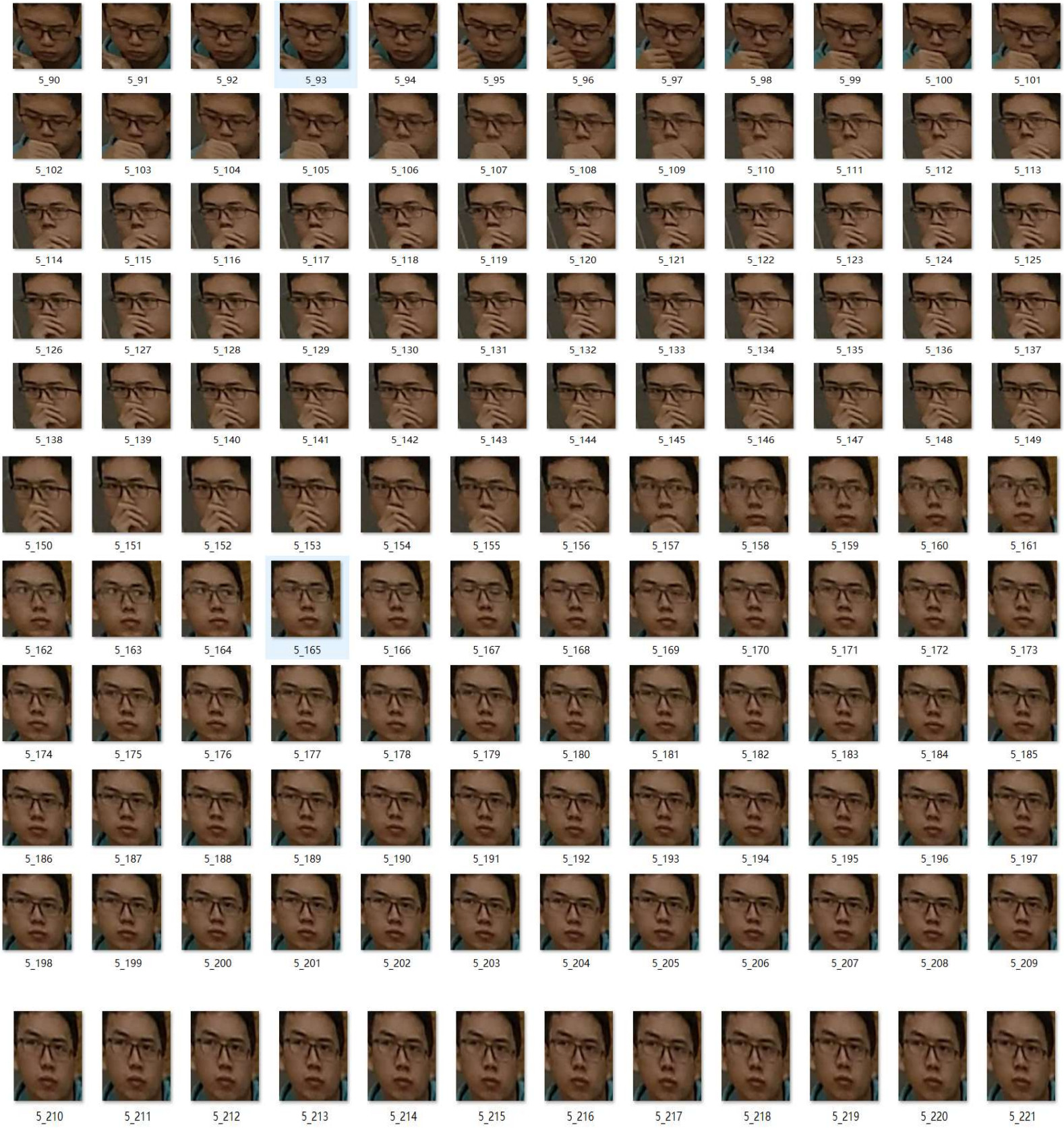
4.2.1 人脸中心的截图方式

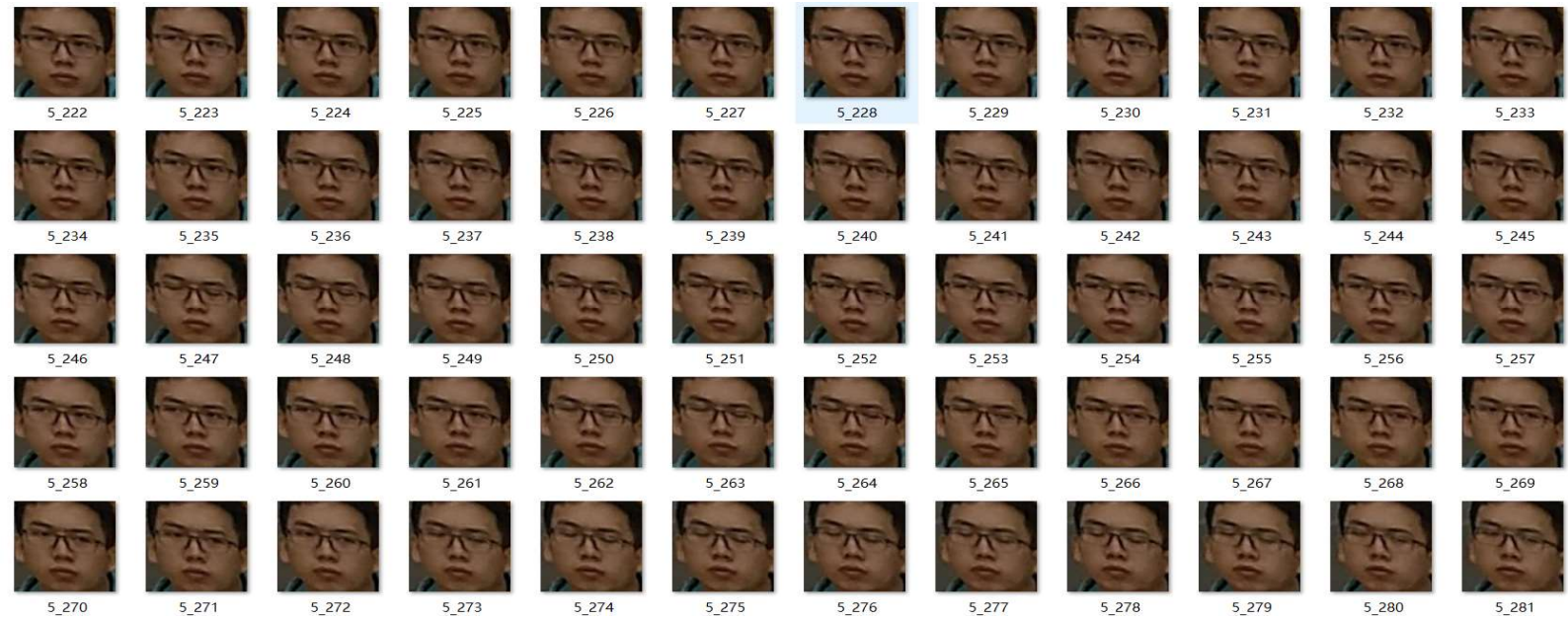
（1）不同帧中的人脸进行 150*150 像素截图

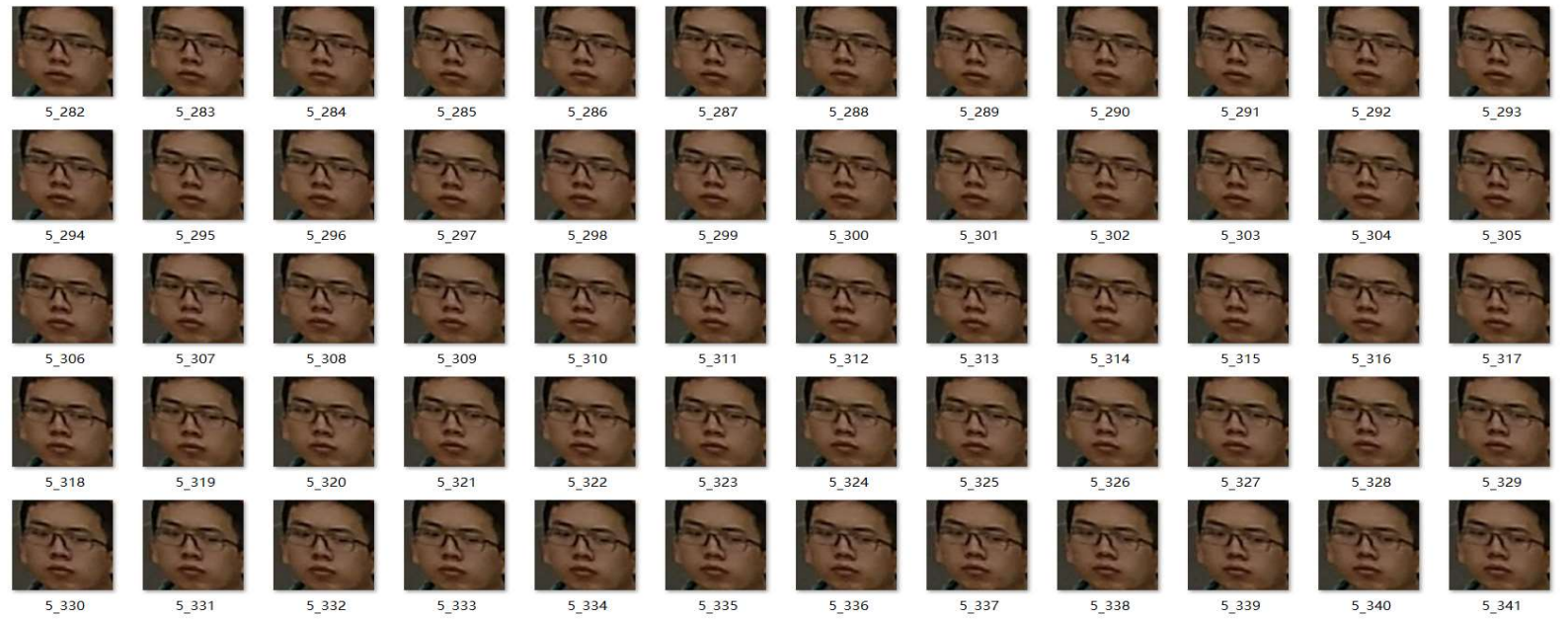

649_463
642_463
635_463
615_463
566_463
535_463
526_463
449_463
5_463
649_462
642_462
635_462
615_462
566_462
535_462
526_462
449_462
5_462
648_461
642_461
635_461
615_461
566_461
535_461
526_461
449_461
5_461
642_460
635_460
615_460
566_460
535_460
526_460
449_460
5_460
642_459
635_459
615_459
566_459
535_459
526_459
449_459
5_459
646_458
642_458
635_458
615_458
566_458
535_458
526_458
449_458
5_458
642_457
635_457
615_457
566_457
535_457
526_457
449_457
5_457

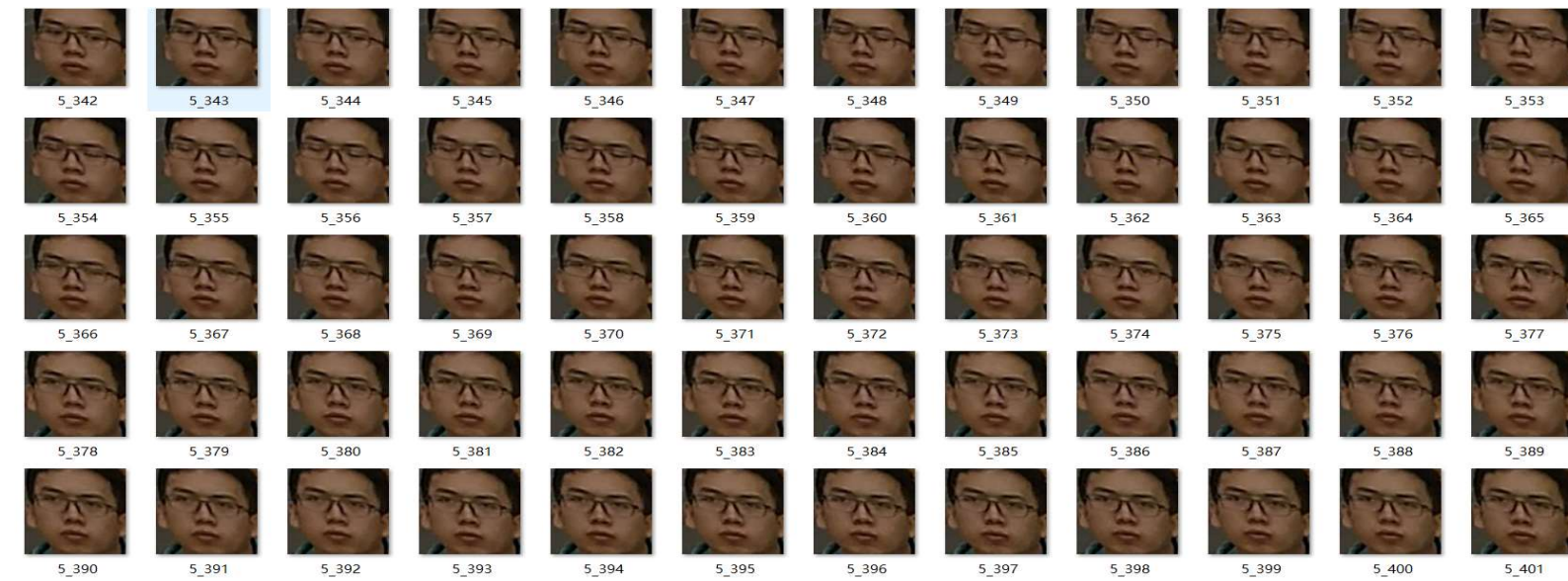
（2）同一人脸的追踪效果

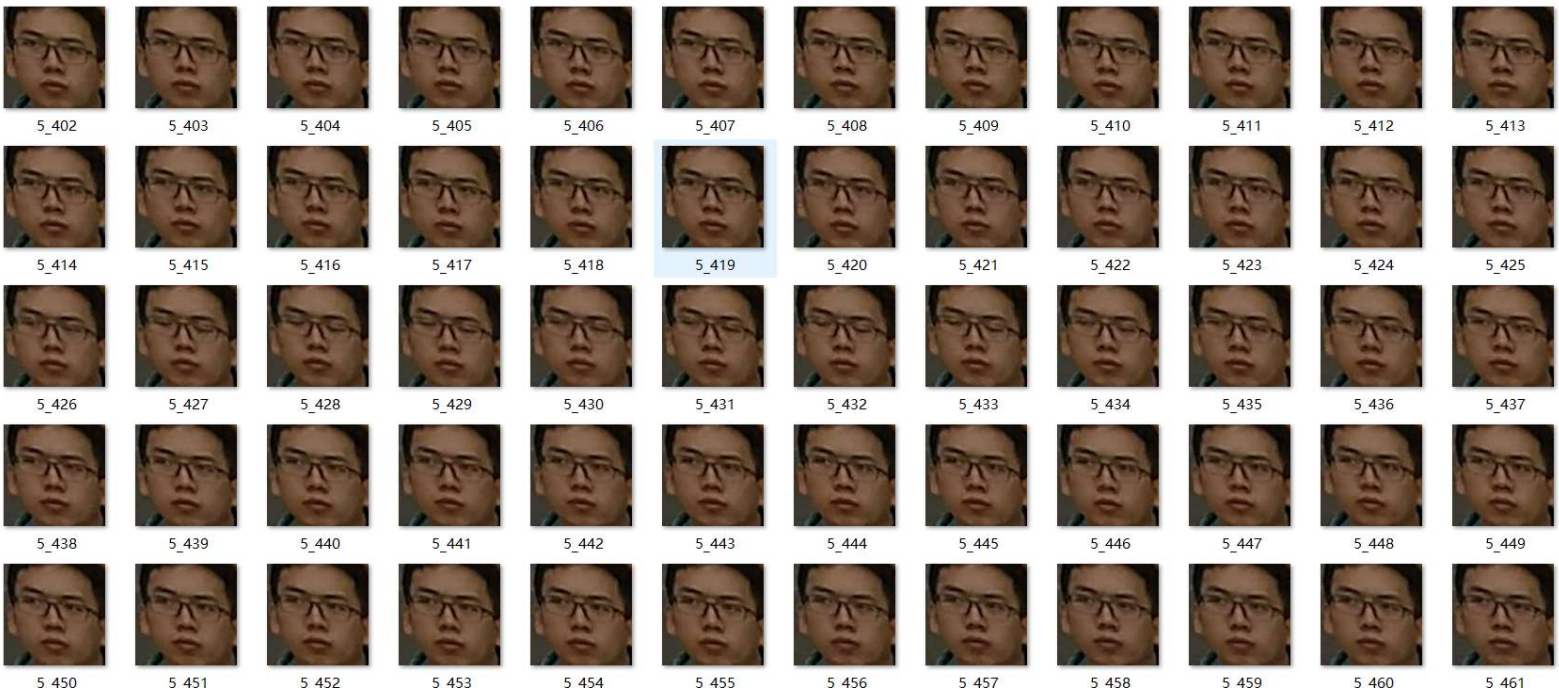

5_30
5_31
5_32
5_33
5_34
5_35
5_36
5_37
5_38
5_39
5_40
5_41
5_42
5_43
5_44
5_45
5_46
5_47
5_48
5_49
5_50
5_51
5_52
5_53
5_54
5_55
5_56
5_57
5_58
5_59
5_60
5_61
5_62
5_63
5_64
5_65
5_66
5_67
5_68
5_69
5_70
5_71
5_72
5_73
5_74
5_75
5_76
5_77
5_78
5_79
5_80
5_81
5_82
5_83
5_84
5_85
5_86
5_87
5_88
5_89


5_90
5_91
5_92
5_93
5_94
5_95
5_96
5_97
5_98
5_99
5_100
5_101
5_102
5_103
5_104
5_105
5_106
5_107
5_108
5_109
5_110
5_111
5_112
5_113
5_114
5_115
5_116
5_117
5_118
5_119
5_120
5_121
5_122
5_123
5_124
5_125
5_126
5_127
5_128
5_129
5_130
5_131
5_132
5_133
5_134
5_135
5_136
5_137
5_138
5_139
5_140
5_141
5_142
5_143
5_144
5_145
5_146
5_147
5_148
5_149
5_150
5_151
5_152
5_153
5_154
5_155
5_156
5_157
5_158
5_159
5_160
5_161
5_162
5_163
5_164
5_165
5_166
5_167
5_168
5_169
5_170
5_171
5_172
5_173
5_174
5_175
5_176
5_177
5_178
5_179
5_180
5_181
5_182
5_183
5_184
5_185
5_186
5_187
5_188
5_189
5_190
5_191
5_192
5_193
5_194
5_195
5_196
5_197
5_198
5_199
5_200
5_201
5_202
5_203
5_204
5_205
5_206
5_207
5_208
5_209
5_210
5_211
5_212
5_213
5_214
5_215
5_216
5_217
5_218
5_219
5_220
5_221


5_222 5_223 5_224 5_225 5_226 5_227 5_228 5_229 5_230 5_231 5_232 5_233
5_234 5_235 5_236 5_237 5_238 5_239 5_240 5_241 5_242 5_243 5_244 5_245
5_246 5_247 5_248 5_249 5_250 5_251 5_252 5_253 5_254 5_255 5_256 5_257
5_258 5_259 5_260 5_261 5_262 5_263 5_264 5_265 5_266 5_267 5_268 5_269
5_270 5_271 5_272 5_273 5_274 5_275 5_276 5_277 5_278 5_279 5_280 5_281

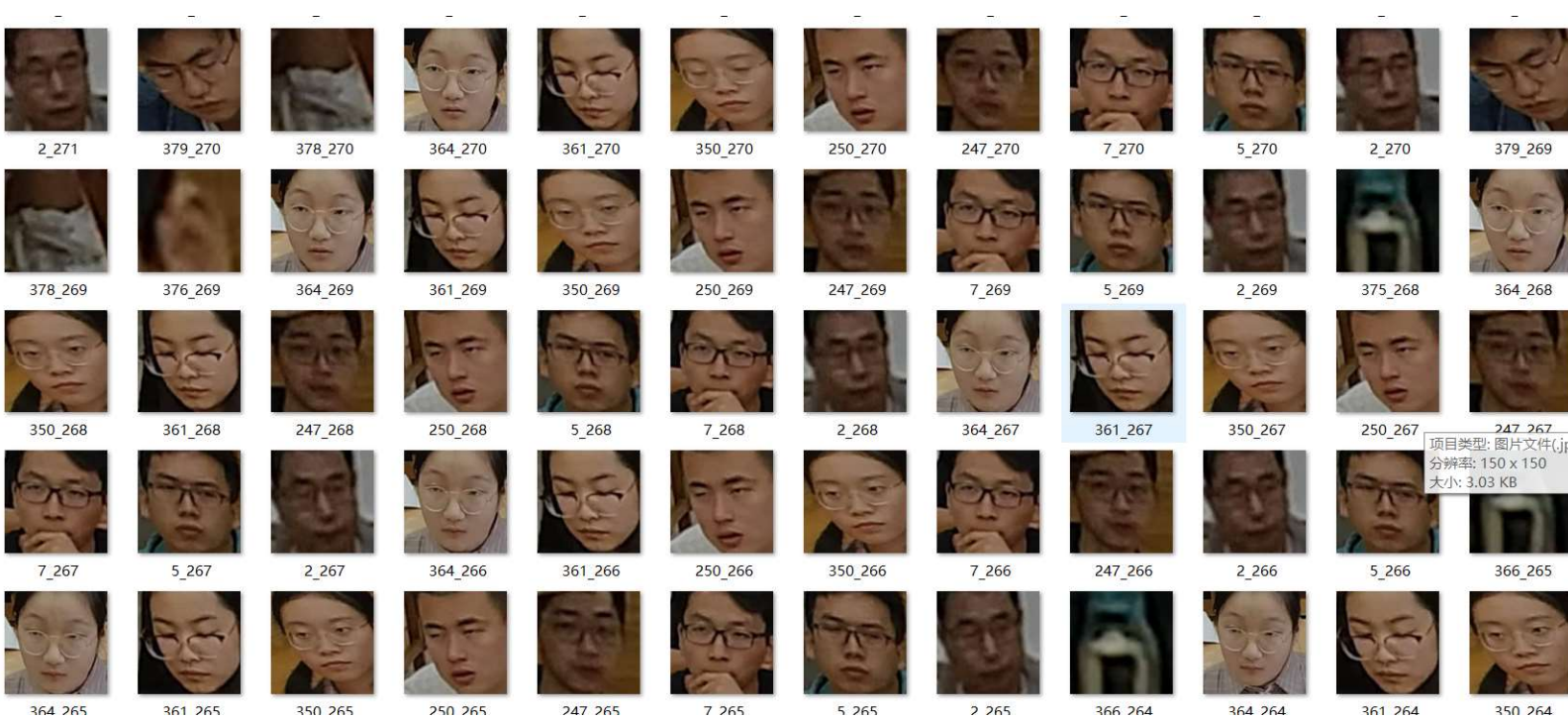

5_282 5_283 5_284 5_285 5_286 5_287 5_288 5_289 5_290 5_291 5_292 5_293
5_294 5_295 5_296 5_297 5_298 5_299 5_300 5_301 5_302 5_303 5_304 5_305
5_306 5_307 5_308 5_309 5_310 5_311 5_312 5_313 5_314 5_315 5_316 5_317
5_318 5_319 5_320 5_321 5_322 5_323 5_324 5_325 5_326 5_327 5_328 5_329
5_330 5_331 5_332 5_333 5_334 5_335 5_336 5_337 5_338 5_339 5_340 5_341


5_342 5_343 5_344 5_345 5_346 5_347 5_348 5_349 5_350 5_351 5_352 5_353
5_354 5_355 5_356 5_357 5_358 5_359 5_360 5_361 5_362 5_363 5_364 5_365
5_366 5_367 5_368 5_369 5_370 5_371 5_372 5_373 5_374 5_375 5_376 5_377
5_378 5_379 5_380 5_381 5_382 5_383 5_384 5_385 5_386 5_387 5_388 5_389
5_390 5_391 5_392 5_393 5_394 5_395 5_396 5_397 5_398 5_399 5_400 5_401



4.22．使用人脸坐标进行 150*150 像素截图

（1）不同帧中的人脸进行 150*150 像素截图



（1） 同一人脸效果

五、相关参数的研究

视频选取：

5.1 匈牙利算法及其损耗

匈牙利算法：一种数据关联(Data Association)算法，其实从本质上讲，跟踪算法要解决的就是数据关联问题。假设有两个集合 S 和 T，集合 S 中有 m 个元素，集合 T 中有 n 个元素，匈牙利算法要做的是把 S 中的元素和 T 中的元素两两匹配（可能匹配不上）。结合跟踪的情景，匈牙利算法的任务就是把 t 帧的人脸坐标与 t-1 帧的人脸坐标两两匹配，达到追踪的目的。要想匹配就需要一定的准则，匈牙利算法依据的准则是“损失最小“。损失由损失矩阵的形式来表示，损失矩阵描述了匹配两个集合中某两个元素所要花费的代价。

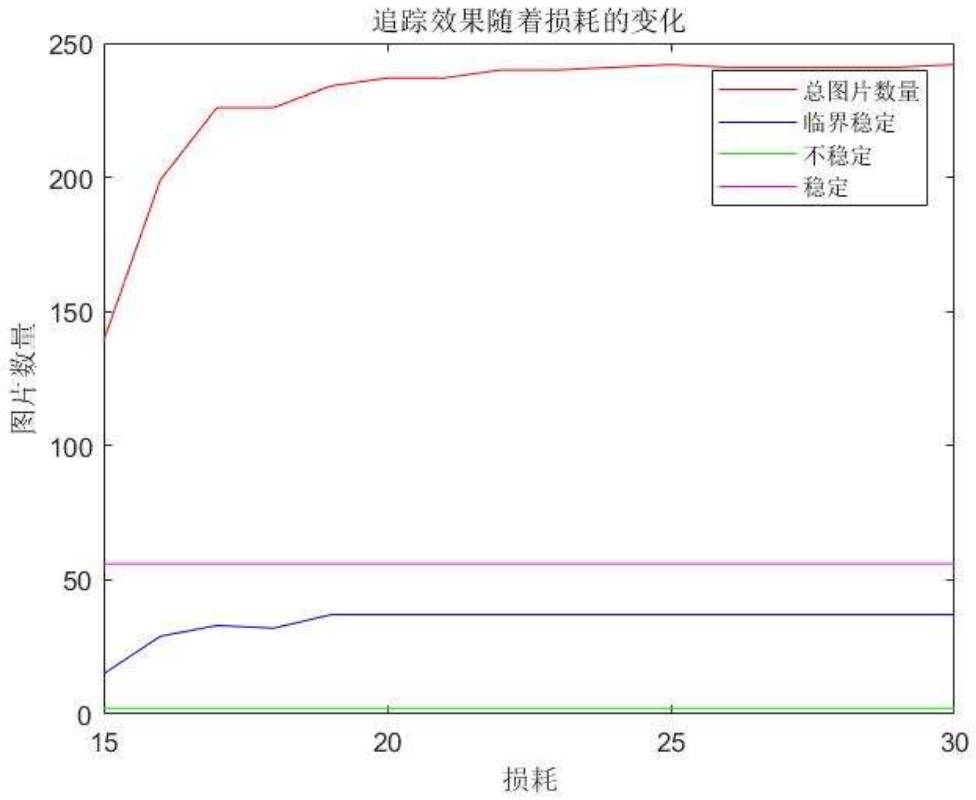
程序部分：

```
cost = zeros(nTracks, nDetections);%建立损耗矩阵
        for i = 1:nTracks
            cost(i, :) = distance(tracks(i).kalmanFilter, bboxes);
        end
        %解决分配问题
        costOfNonAssignment =;%损耗值设置
        [assignments, unassignedTracks, unassignedDetections] = ...
            assignDetectionsToTracks(cost, costOfNonAssignment);
```

实验结果：

Cost	总帧数	不稳定	临界稳定	稳定
15	139	2	15	56
16	199	2	29	56
17	226	2	33	56

18	226	2	32	56
19	234	2	37	56
20	237	2	37	56
21	237	2	37	56
22	240	2	37	56
23	240	2	37	56
24	241	2	37	56
25	242	2	37	56
26	241	2	37	56
27	241	2	37	56
28	241	2	37	56
29	241	2	37	56
30	242	2	37	56



不同损耗值下帧数的变化

损耗值越小，则对追踪的要求越苛刻，导致追踪的结果变少。而损耗的越大，

则分配的效果变差，追踪效果变差。

随着损耗值的增加，保持正脸的目标非常稳定，在各个损耗值下都能保持跟踪目标不变。只进行一次低头的目标，随着损耗值的增加，在该视频下 cost=19 时达到最优追踪效果。而频繁丢失目标会导致无法检测的目标，这种情况下只能增大其可见帧数，同时减少不可见帧数。

5.2 minVisibleCount 的选取

该参数主要目的是降噪，要求一定的帧数去配置卡尔曼滤波器，减少噪声。

minVisibleCount=0（即不进行降噪）

minVisibleCount=1

minVisibleCount=4

该值是该代码最后的参数，具有分类的作用。当一个轨迹的 totalVisibleCount > minVisibleCount 才会显示，所以一些轨迹的帧数太少时并不会出现。当该值太小时会导致误差太大，而太大时会导致追踪效果太差，过滤太多轨迹。在该视频下最优值为 4，此时已经消除绝大多噪声。

5.3 删除轨迹的相关参数

invisibleForTooLong 不可见的帧数数目

ageThreshold总帧数

visibility = totalVisibleCounts ./ ages 可见效果，为总可被检测帧数比总帧数

[tracks(:).consecutiveInvisibleCount] 轨迹中不可见的数目

Matlab中的判定语句:

```
lostInds = (ages < ageThreshold & visibility < 1)
| [tracks(:).consecutiveInvisibleCount] >=
invisibleForTooLong;
        tracks = tracks(~lostInds);
```

满足删除轨迹的调节为：轨迹的总帧数小于设定的ageThreshold并且可见效果小于设置值，或者轨迹不可见的连续帧数小于invisibleForTooLong

在进行实验中，这些值的设置会导致某一帧完全没有人脸，输出一个空矩阵，导致程序终止，可以作为后续的研究。

六、建议

（1）通过建立人脸数据库，使用人脸特征对比可以减少数据处理的时间。

（2）对于后排同学，如果截图效果像素较低较模糊，可以使用图像超分辨率重构，增加像素。