

人脸目标跟踪与速度测量算法的设计

赵思源

(南京理工大学 机械工程学院, 江苏 南京 210000)

摘要: **目的:** 为了实现视频图像人脸的跟踪并通过图像分析人脸移动速度, 对目标跟踪方法进行探究, 设计了运动特征提取算法, 实现了人脸自动跟踪并指示其中心的功能。**方法:** 首先, 使用级联分类器识别图像中的人脸, 确定待跟踪人脸目标并检测人脸区域的最小特征值角点, 完成跟踪目标的初始化。接着, 通过匹配前后两帧图像的角点获取两帧图像人脸区域间的几何变换矩阵, 将该矩阵作用于人脸区域, 实现了人脸的高效跟踪, 并获得了人脸的图像运动速度。最后结合相机成像原理与拍摄相机的内参矩阵推算出人脸的真实运动速度。**结果:** 拍摄视频进行测试, 在人脸有大幅表情变化且头部存在转动的情况下也能完成跟踪与测速任务, 得出移动的测试人脸的水平方向运动速度范围为 -0.9406m/s 至 0.9232m/s , 竖直方向运动速度范围为 -0.4488m/s 至 0.5306m/s 。**结论:** 结合真实运动情况, 该算法完成了人脸的跟踪与运动速度测量任务, 具有高效、稳定、高精度的特点。

关键词: 运动目标跟踪、速度测量、级联分类器、人脸识别、角点匹配

Design of face-target tracking and speed measurement algorithms

Zhao Siyuan

(College of Mechanical Engineering, Nanjing Univ. of Sci & Tech., Nanjing 210000, China)

Abstract: **Objective:** In order to realize the tracking of the face in the video image and analyze the moving speed of the face through the image, the method of target tracking is explored, the motion feature extraction algorithm is designed, and the function of automatic face tracking and indicating its center is realized. **Method:** First, use the neural network to identify the face in the image, determine the face target to be tracked and detect the corner points of the minimum eigenvalue of the face area, and complete the initialization of the tracking target. Next, the geometric transformation matrix between the face regions of the two frames of images is obtained by matching the corners of the two frames of images before and after, and the matrix is applied to the face region to realize the efficient tracking of the face and obtain the image motion speed of the face. . Finally, combined with the camera imaging principle and the internal parameter matrix of the shooting camera, the real movement speed of the face is calculated. **Results:** Shooting video for testing, the tracking and speed measurement tasks can be completed even when the face has a large expression change and the head is rotating, and the horizontal movement speed range of the moving test face is -0.9406m/s to 0.9232m/s , the vertical motion speed range is -0.4488m/s to 0.5306m/s . **Conclusion:** Combined with the real motion situation, the algorithm completes the task of face tracking and motion speed measurement, and has the characteristics of high efficiency, stability and high precision.

Key words: Moving target tracking, speed measurement, neural network, face recognition, feature point matching

1. 引言

视频目标跟踪是计算机视觉领域重要的基础性研究问题之一, 是指在视频序列第一帧指定目标后, 在后续帧持续跟踪目标, 即利用边界框(通常用矩形框表示)标定目标, 实现目标的定位与尺度

估计。视频目标跟踪具有广泛的应用价值, 包括

- 1) 公共安防领域: 对人群或重点对象进行跟踪定位, 实现监控场景下可疑人员轨迹重建与实时定位;
- 2) 自动驾驶领域: 辅助自主导航, 轨迹规划等功能的实现;
- 3) 智能机器人领域: 用于机器人视觉导航, 关注目标的运动轨迹捕获与主动追踪;

4) 人机智能交互领域：通过人体关键部位（如手部）跟踪与识别，实现计算机根据人体特定动作或手势等完成相应反馈。由于存在诸多技术挑战和潜在应用价值，视频目标跟踪近年来也引起学术界和工业界的广泛关注和大量研究。

本文从应用的角度，研究视频目标跟踪的基本理论，并基于MATLAB构建人脸追踪与运动测速模型，实现人脸目标追踪与速度测量。在工程上有一定的应用价值。

2. 人脸跟踪系统的构成及工作原理

2.1 人脸跟踪系统的工作流程

首先对用于视频拍摄的相机进行相机标定，获取相机内参数，建立相机坐标系和图像坐标系的映射关系，为后续速度测算做好准备。

然后检测视频图像序列中的待跟踪目标，检测到可跟踪目标后将其设定为跟踪对象，并对视频后续帧中出现的目标进行跟踪。并将其角点设为跟踪点。由于目标识别需要占用大量的计算资源，后续的图像帧仅对上一帧中的目标进行角点匹配，由角点间转换关系估算运动目标整体的转换关系，由此实现高效的目标跟踪。

在实现目标跟踪后，即可获得人脸目标在图像坐标系下的运动速度，结合相机的成像理论和相机的内外参数矩阵即可由图像坐标系速度推导出相机坐标系下速度，完成人脸运动的真实速度测量。

其工作流程如下图所示：

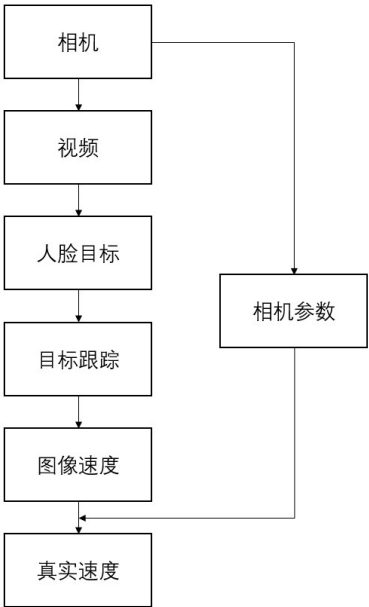


图 1：人脸跟踪系统流程

Figure 1: Face tracking system flow

3. 人脸跟踪系统的关键算法

人脸跟踪系统的关键部分为人脸的识别与选择，目标跟踪和计算人脸的运动速度。

3.1 总览

本文设计人脸跟踪系统的关键算法实现流程图如下：

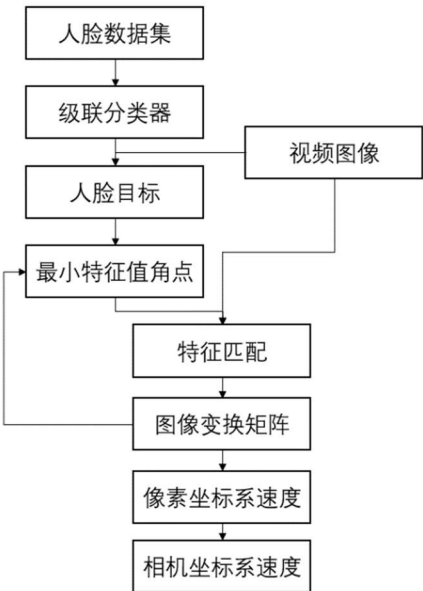


图 2：人脸跟踪与测速关键算法流程

Figure 2: Face tracking and speed measurement key algorithm flow

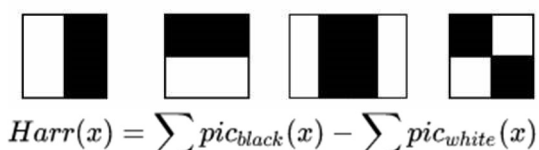
首先，使用训练完成的级联分类器识别视频图像中的人脸，确定待跟踪人脸目标并检测人脸区域的最小特征值角点，将这些角点设置为跟踪目标完成跟踪目标的初始化。接着，通过匹配前后两帧图像的角点获取两帧图像人脸区域间的几何变换矩阵，将该矩阵作用于人脸区域，实现了人脸的高效跟踪，该帧的跟踪完成后，更新角点信息，除去与上一帧无法匹配的角点。通过人脸区域中心点坐标的差值运算以及视频帧率获得了人脸的图像运动速度。最后结合相机成像原理与拍摄相机的内参矩阵推算出人脸的真实运动速度。

3.2 确定待跟踪目标

人脸识别是设定人脸跟踪目标的前置步骤，也是人脸跟踪算法的基础。在本文中，使用级联分类器实现人脸检测。

3.2.1Harr 特征

Harr特征是一类非常简单的特征，如下图所示有四个框，这四个框的大小是可变的，使用黑色部分覆盖的像素之和减去白色部分覆盖的像素之和即为Harr特征：



$$Harr(x) = \sum pic_{black}(x) - \sum pic_{white}(x)$$

图 3: Harr 特征算子

Figure 3: Harr feature operator

3.2.2 分类器构造

在获取图像的Harr特征后，由于Harr特征数量极多，对于直接训练成一个分类器的机器学习任务来说计算消耗过大，因此使用多个弱分类器组成一个强分类器的方法训练。在本系统中，每一个弱分类器只针对一个单独的特征，训练筛选分类器时，不选择误差最小的分类器，而是选择最少的将正例划分为反例的分类器，即召回率最高的分类器。且下一次计算的样本集合为使用该分类器剔除反例的样本集合。

$$h_j(x) = \begin{cases} 1 & f_j(x) < \theta_j \\ 0 & other \end{cases}$$

完成对各个弱分类器的训练后，将选择特征的数量记为 T ，将各个分类器的权重记为 a_t （ a_t 的值在训练中得出），则构成分类器：

$$h(x) = \begin{cases} 1 & \sum_{t=1}^T a_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T a_t \\ 0 & other \end{cases}$$

使用该分类器即可对视频中的图像进行识别，判断其是否含有人脸：（1）使用滑动框在图像上滑动获取子图（2）在子图上提取Harr特征获取大量特征数据（3）运行时，顺序计算特征-分类，当样本被一个分类器识别为反例时，直接拒绝该样本，后续的特征和分类都可以不被计算。（4）若子图所有的特征均通过了分类器，则将其判定为该分类器的目标识别物体。

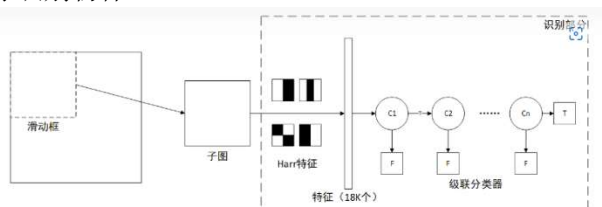


图 4: 级联分类器工作原理

Figure 4: How the Cascaded Classifier Works

3.2.3 角点提取

通过分类器可获得源图像中待跟踪人脸目标子图像的位置和大小。但是，如果选择对子图像内所

有的像素点均进行跟踪，会造成大量的计算资源浪费，且一旦待检测目标局部发生变化，很可能导致操作的失败。由于角点很好的包含了图像中的特征信息，因此需要提取子图像（人脸区域）内的角点，用这些角点的运动来代指子图像整体的运动。在本文中使用Harris角点。

Harris角点检测技术是Moravec角点检测的改良与优化。假设待检测图像为 f ，待检测算子采用计算图像的梯度值来寻找其在 (x, y) 处的边缘强度与方向，则梯度可表示为：

$$I(x, y) = \begin{pmatrix} I_x \\ I_y \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}^T$$

在图像上取一个 $W \times W$ 的“滑动窗口”，不断的移动这个窗口并检测窗口中的像素变化情况 E 。像素变化情况 E 可简单分为以下三种：

（1）如果在窗口中的图像是什么平坦的，那么 E 的变化不大；

（2）如果在窗口中的图像是一条边，那么在沿这条边滑动时 E 变化不大，而在沿垂直于这条边的方向滑动窗口时， E 的变化会很大；

（3）如果在窗口中的图像是一个角点时，窗口沿任何方向移动 E 的值都会发生很大变化。

假定当前点坐标为 (x, y) ，“滑动窗口”移动距离为 $(\Delta x, \Delta y)$ ，则用于表示自相关性的自相关函数为：

$$c(x, y, \Delta x, \Delta y) = \sum_{i=1}^n w(x, y) (I(x, y) - I(x + \Delta x, y + \Delta y))^2$$

其中 $w(x, y)$ 表示加权函数。

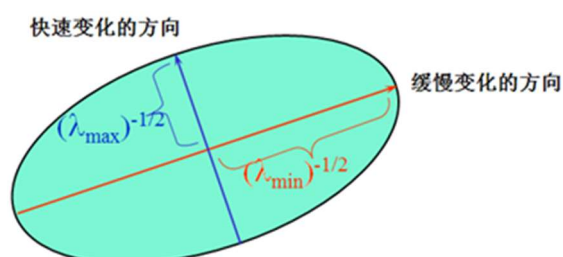
对此式进行泰勒展开：

$$c(x, y; \Delta x, \Delta y) \approx A \Delta x^2 + 2C \Delta x \Delta y + B \Delta y^2$$

$$\text{其中 } A = \sum_{i=1}^w I_x^2, B = \sum_{i=1}^w I_y^2, C = \sum_{i=1}^w I_x I_y。$$

二次项函数本质上就是一个椭圆函数。椭圆的扁率和尺寸是由 $M(x, y)$ 的特征值 λ_1, λ_2 决定的，椭圆方向是由 $M(x, y)$ 的特征矢量决定的，如下图所示，椭圆方程为：

$$[\Delta x, \Delta y] M(x, y) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = 1$$



若矩阵 $M(x, y)$ 的两个特征值都小，且近似相等，自相关函数在所有方向都增大，则表示检测到了角点。

是否为角点使用响应值来判断：

$$R = \det M - \alpha(\text{trace} M)^2$$

其中

$$\det M = \lambda_1 \lambda_2 = AC - B^2$$

$$\text{trace} M = \lambda_1 + \lambda_2 = A + C$$

由此实现了图像角点的检测，将这些角点记为待跟踪点，进行后续操作。

3.3 目标跟踪

3.3.1 假设

与KLT光流法类似，为方便图像中点的匹配，对视频做如下假设：

1. 相邻帧之间的亮度恒定；
2. 相邻帧之间物体的运动比较“微小”；
3. 保持空间一致性；即相邻像素点具有相同的运动

在以上约束下，便可使用与KLT光流法类似的方法实现角点的跟踪。设 I_x, I_y, I_t 分别表示图像点在 x, y 和时间方向的梯度， u, v 为角点在像素坐标系下的运动速度，则光流约束方程：

$$I_x u + I_y v + I_t = 0$$

3.3.1 单角点跟踪

在光流方程中，有两个未知数，一个方程，不能直接进行求解，因此采用稀疏光流跟踪算法。根据“相邻像素的空间一致性”构建一个 5×5 的领域，25个像素点具有相同的运动，就可以得到9个点的光流方程，用这些方程来求得这两个未知数：

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

即

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

求解此方程后，使用最小二乘法拟合，获取较为精确的角点运动速度。

3.3.2 人脸整体跟踪

当前的目标便是根据各角点的速度估算出人脸整体的速度。使用 3×3 仿射变换矩阵来表示相邻两

帧图像人脸区域的变换关系，它保持了二维图形的“平直性”（直线经过变换后依然是直线）和“平行性”（二维图形之间相对位置保持不变，平行线依然是平行线，且直线上点的位置关系不变）。

假定上一帧的中心点坐标为 (x, y) ，当前中心点的坐标为 (x', y') ，则

$$x' = ax + by + m$$

$$y' = cx + dy + n$$

其中 m, n 表示平移，可将变换关系写为齐次形式：

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & m \\ c & d & n \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

因此仿射变换矩阵可以很好的表示人脸区域的变换关系。每个角点均可引入两个约束，将多个角点前后两帧的坐标信息带入，并使用最小二乘法拟合。即可求解获得仿射变换矩阵。

3.4 速度解算

在获取相邻帧人脸的仿射变换矩阵后，即可实现对人脸的跟踪，设当前帧和上一帧的人脸中心坐标分别为 $(x', y'), (x, y)$ ，图像坐标系速度为 u, v ，则

$$u = \frac{dx}{dt} = x' - x$$

$$v = \frac{dy}{dt} = y' - y$$

在本文中，假定相机拍摄时与拍摄人脸的相对垂直距离保持不变，定义焦距 f 分别与像元的横纵尺寸 d_x, d_y 之比为等效焦距 F_x, F_y ，则距离根据相机成像原理

$$z_c x = F_x x_c + z_c x_0$$

$$z_c y = F_y y_c + z_c y_0$$

设帧率为 p ，可得

$$x_c' = \frac{z_c x'}{F_x} = \frac{z_c p u}{F_x}$$

$$y_c' = \frac{z_c y'}{F_y} = \frac{z_c p v}{F_y}$$

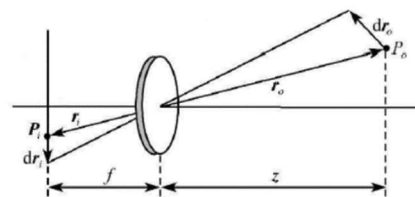


图 5：投影方程联系物点和像点
Figure 5: Connecting object points and image points using projection equations

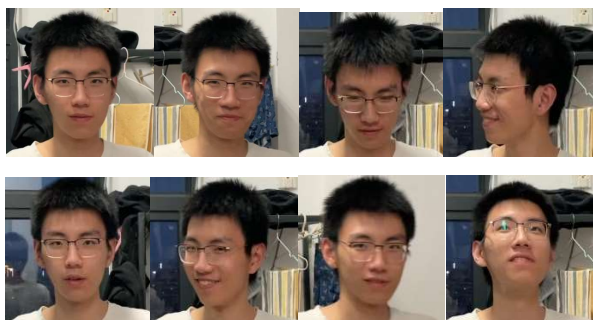
将此速度进行单位换算，即可获得人脸的真实运动速度。

使用红色十字星将人脸中心标出，白色十字星为跟踪角点

4. 实验设置与实验结果

4.1 实验安排

使用手机相机距离人一米处拍摄一段人运动的视频，视频中人进行了从右至左、从左至右的平动，同时包含抬头低头转头等动作以及表情变化。该视频长度为9s。

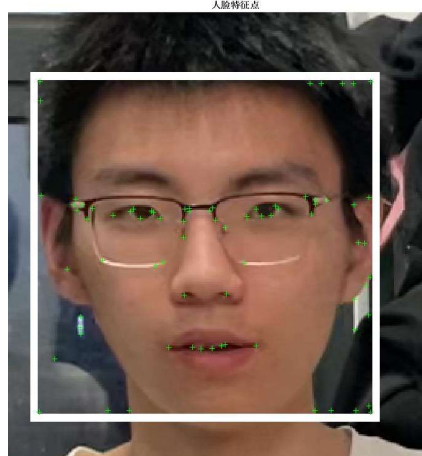


4.1 实验结果

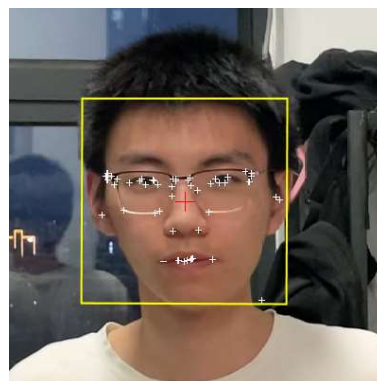
级联分类器能够有效的提取出人脸子图像，识别到的人脸区域用方框标出



识别人脸区域的角点，将角点信息保存

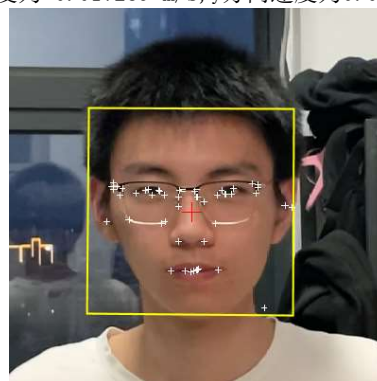


观察可知这些点主要分布在图像边缘的交界处，将这些点设为跟踪目标。



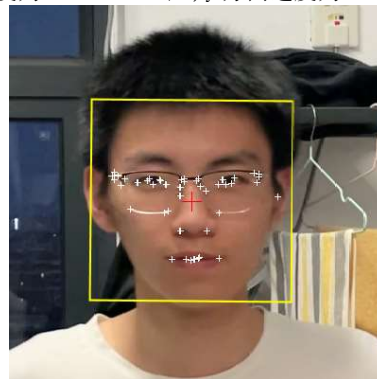
第 2 帧

x方向速度为-0.017289 m/s, y方向速度为0.024263 m/s



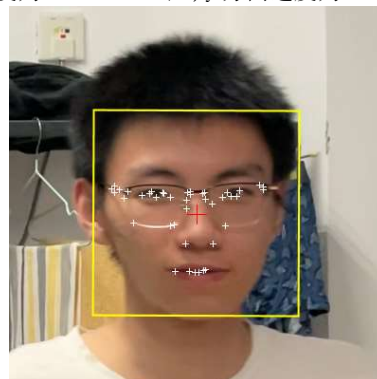
第 69 帧

x 方向速度为 0.315037 m/s, y 方向速度为 0.045724 m/s



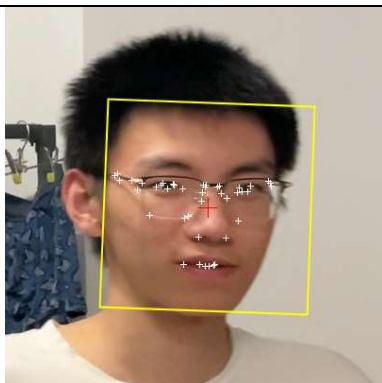
第 91 帧

x 方向速度为 0.679732 m/s, y 方向速度为 0.001559 m/s



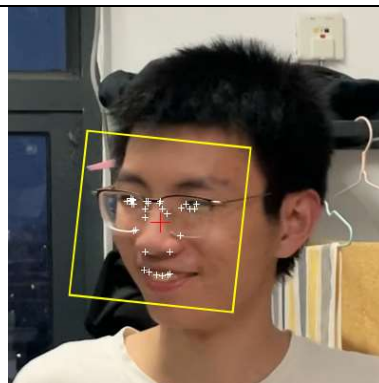
第 148 帧

x 方向速度为 0.731917 m/s, y 方向速度为 0.019737 m/s



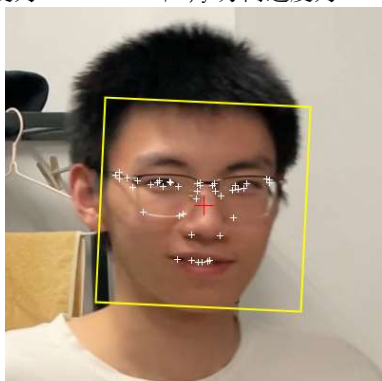
第 199 帧

x 方向速度为 0.578356 m/s, y 方向速度为 0.001231 m/s



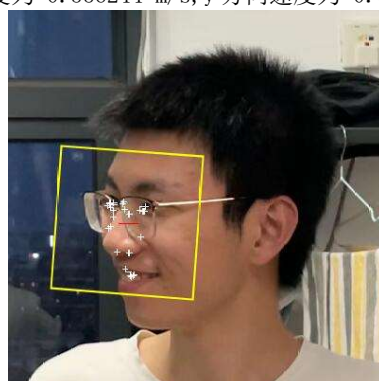
第 342 帧

x 方向速度为-0.556244 m/s, y 方向速度为-0.033110 m/s



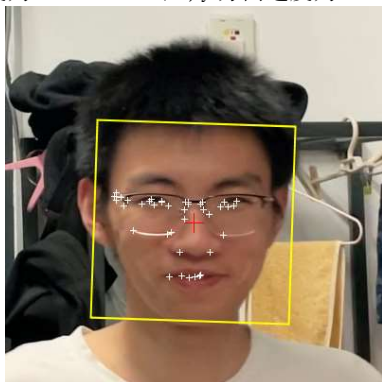
第 250 帧

x 方向速度为-0.577502 m/s, y 方向速度为-0.024589 m/s



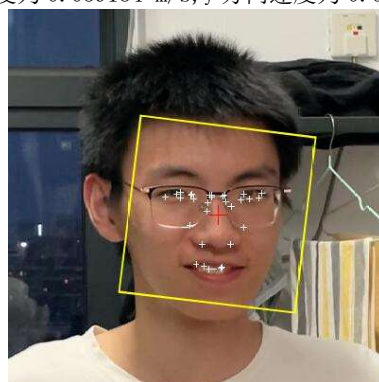
第 366 帧

x 方向速度为 0.059184 m/s, y 方向速度为 0.018062 m/s



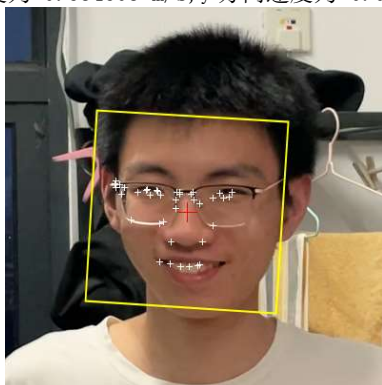
第 302 帧

x 方向速度为-0.634608 m/s, y 方向速度为-0.009138 m/s



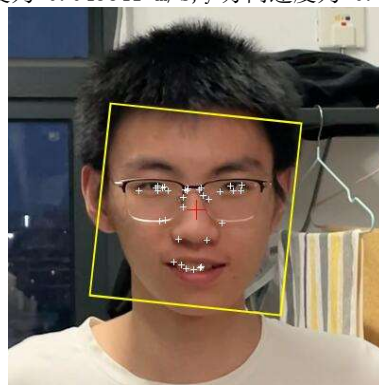
第 412 帧

x 方向速度为-0.043541 m/s, y 方向速度为-0.082489 m/s



第 329 帧

x 方向速度为-0.261542 m/s, y 方向速度为 0.001395 m/s



第 490 帧

x 方向速度为-0.011794 m/s, y 方向速度为 0.018789 m/s

5. 结论

本文根据视频图像目标追踪高精度、高速度的需求,综合使用了分类器、角点检测、光流等多种视觉相关处理技术,综合考虑相机成像原理,提出了

一种人脸追踪与测速算法。经测试,该算法能够有效实现目标功能,有一定实用价值。

参考文献:

- [1]张娟,毛晓波,陈铁军.运动目标跟踪算法研究综述[J].计算机应用研究,2009,26(12):4407-4410.
- [2]陈娟,陈乾辉,师路欢,吴建军.图像跟踪中的边缘检测技术[J].中国光学与应用光学,2009,2(01):46-53.
- [3]王崴,唐一平,任娟莉,时冰川,李培林,韩华亭.一种改进的 Harris 角点提取算法[J].光学精密工程,2008(10):1995-2001.
- [4]赵文彬,张艳宁.角点检测技术综述[J].计算机应用研究,2006(10):17-19+38.
- [5]侯志强,韩崇昭.视觉跟踪技术综述[J].自动化学报,2006(04):603-617.DOI:10.16383/j.aas.2006.04.016.
- [6]王亮,胡卫明,谭铁牛.人运动的视觉分析综述[J].计算机学报,2002(03):225-237.
- [7]王春平,朱元昌,黄允华.基于图像信息的跟踪算法分析[J].火力与指挥控制,2000(01):18-21+25.
- [8]关海英,阮秋琦.序列图像中运动目标的检测[J].铁道学报,1996(06):34-39.

附录

项目地址:

<https://github.com/zhaosiyuan1098/robotVision/tree/test/3>

源代码:

```
clear;
intrinsicMatrix=[1555.00918908222 ,0 ,0;
                 0 ,1574.28624772510, 0;
                 963.674168044664, 628.130992891514, 1];
face_camera_Distance=1000;
videoReader = VideoReader('siyuanmove.mp4','CurrentTime',0);
faceDetector = vision.CascadeObjectDetector();
%%
faceInitialFlag=0;
while hasFrame(videoReader)
    frame=readFrame(videoReader);
    % sharpColorFrame = imsharpen(frame);
    % adjustColorFrame = imadjust(sharpColorFrame,[.1 .1 0; .6 .7 1],[,]);
    faceBbox = faceDetector(frame);
    if(size(faceBbox,1)>1)
        [~,faceIndex]=max(faceBbox(:,3));
        frame = insertShape(frame,
'Rectangle',faceBbox(faceIndex,:), 'Color','white', 'LineWidth',5);
```

```
figure; imshow(frame); title('检测到的人脸');
points =
detectMinEigenFeatures(rgb2gray(frame), 'ROI', faceBbox(faceIndex, :));

harrisPoints=points.selectStrongest(300);
figure, imshow(frame), hold on, title('人脸特征点');
plot(harrisPoints);
harrisPoints=harrisPoints.Location;
faceBboxPoints=bbbox2points(faceBbox(faceIndex, :));
lastPoints=harrisPoints;
lastFaceCenter=mean(faceBboxPoints);
faceInitialFlag=1;
break;
end
end

%%
min=10000;max=-10000;
numm=0;
x_arr=[];
y_arr=[];
if faceInitialFlag==1
    pointTracker = vision.PointTracker('MaxBidirectionalError', 2);
    initialize(pointTracker, harrisPoints, frame);
    videoPlayer = vision.VideoPlayer('Position',...
        [100 100 [size(frame, 2), size(frame, 1)]+30]);
    oldPoints = harrisPoints;
    while hasFrame(videoReader)
        frame = readFrame(videoReader);
        [harrisPoints, isFound] = step(pointTracker, frame);
        trackablePoints = harrisPoints(isFound, :);
        oldInliers = lastPoints(isFound, :);
        if size(trackablePoints, 1) >= 2
            [xform, inlierFlag] = estimateGeometricTransform2D(...
                oldInliers, trackablePoints, 'similarity', 'MaxDistance', 4);
            oldInliers = oldInliers(inlierFlag, :);
            trackablePoints = trackablePoints(inlierFlag, :);
            faceBboxPoints = transformPointsForward(xform, faceBboxPoints);
            faceCenter=mean(faceBboxPoints);
            bboxPolygon = reshape(faceBboxPoints', 1, []);
            frame = insertShape(frame, 'Polygon', bboxPolygon, 'LineWidth', 2);
            frame = insertMarker(frame, trackablePoints, '+', 'Color', 'white');
            frame = insertMarker(frame, faceCenter, '+', 'Color', 'red', 'Size', 10);
```



```
        v_x=(faceCenter(1)-
lastFaceCenter(1))*intrinsicMatrix(1,1)*60/1000/face_camera_Distance;
        v_y=(faceCenter(2)-
lastFaceCenter(2))*intrinsicMatrix(2,2)*60/1000/face_camera_Distance;
        x_arr=[x_arr,v_x];
        y_arr=[y_arr,v_y];
        if v_y<min
            min=v_y;
        end
        if v_y>max
            max=v_y;
        end
        X=sprintf('x 方向速度为%f m/s,y 方向速度为%f m/s',v_x,v_y);
        disp(X);
        numm=numm+1
        lastPoints = trackablePoints;
        lastFaceCenter=faceCenter;
        setPoints(pointTracker, lastPoints);
    end
    step(videoPlayer, frame);
end
else
    disp("视频中未检测到人脸")
end
% release(videoPlayer);
```