

人机工程快速评估方法

金航宇, 吕成伟

北京福田康明斯发动机有限公司 北京 102206

摘要: 为快速评估人机工程学是否合理, 提出了通过视觉识别方式自动评估人机工程的方法, 主要运用了视频采集、姿态识别、姿态计算等算法, 为快速完成人机工程评估提供了一个新的思路。同时, 文中提到的姿态识别也可以扩展应用到操作防错、装配防错。

关键词: 人机工程; 人体姿态识别; 智能制造; Mediapipe

在汽车发动机批量生产过程中存在很多装配、检查等工位, 这些工位劳动者的操作动作人机工程学是否合理、劳动者工作是否舒适、是否安全对身体无伤害等, 是现场工业工程师评估的一项重要工作。通常一个发动机装配线有几十到几百个人工工位, 如何快速评估这些工位的人机工程合理性一直是工业工程师、工艺师的一个难点。本文通过视频监控、视觉识别、自动姿态分析与交互计算的方法, 提出一种快速自动评估的方案。

评估原理

人机工程中关于人员操作的评估分为姿态、受力与频次三个维度, 某公司对臂力的一个人机工程要求见表1。姿态指的是人体姿势, 这个通过视觉识别人体姿态来实现, 开源自行使用Opencv训练也可以采用开源框架来实现 (本文采用Google开源姿态识别框架Mediapipe来介绍); 频次可以通过视频流分析来技术标记; 由于生产线搬运安装的工件或者使用的工具是固定型号的, 对于用力只需要评估前标定或者测量下即可。待此三个关键因子获取到之后, 通过计算就可

表1 某公司人机工程臂力评估表

序号	姿态	频次 (经常≥2次/min, 非经常<2次/min)	推荐力/kgf (1kgf=9.8N)
1	弯腰拾取	经常	推荐值为2.0, 可接受值为3.1
		非经常	推荐值为4.3, 可接受值为7.1
2	单手以肩高推出	经常	推荐值为3.1, 可接受值为4.6
		非经常	推荐值7.7, 可接受值为10.1
3	单手以肘高推出	经常	推荐值为3.4, 可接受值为5.1
		非经常	推荐值为8.4, 可接受值为11
4	单手以肩高拉近	经常	推荐值为3.2, 可接受值为4.8
		非经常	推荐值为8, 可接受值为10.3
5	单手以肘高拉近	经常	推荐值为3.4, 可接受值为5.1
		非经常	推荐值8.5, 可接受值为11.1
6	单手从膝部高度拉起	经常	推荐值为2.9, 可接受值为4.3
		非经常	推荐值为7.2, 可接受值为9.3
7	单手以腰际高度横拉过身体, 肘部完全伸展	经常	推荐值为1.1, 可接受值为1.7
		非经常	推荐值为2.9, 可接受值为3.7
...	...		

以分析出工位人机工程是否合理。

按照表1，如果“弯腰拾取”， $2.0\text{kgf} < \text{工件的重量或者推力} < 3.1\text{kgf}$ ，那么频次 $> 2\text{次/min}$ 是可以接受的。

评估方法（以弯腰拾取为例）

1.姿态识别框架

人机工程评估需要识别姿态，本文通过Mediapipe来识别人体姿态。Mediapipe是一个开源软件，它集成机器学习视觉算法的工具库，包含了人脸检测、人脸关键点、手势识别、头像分割和姿态识别等各种模型，本文使用其中的人体姿态检测与跟踪模块（见图1）。人体姿态与检测模块会识别单个人体，并对人体定义33个关节点（见图2），同时给出对地的一个坐标值。这个框架支持多种语言，本文采用Python语言对其进行调用，实时提取视频中每帧人体的姿态中要评估的关节或者动作相关关节的坐标位置，通过快速计算并给出需要评估动作的频次。最后结合通过人工输入力的数值实时对人机工程学进行评估，并给出结论。

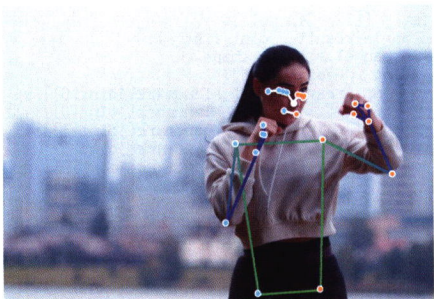


图1 人体姿态检测与跟踪

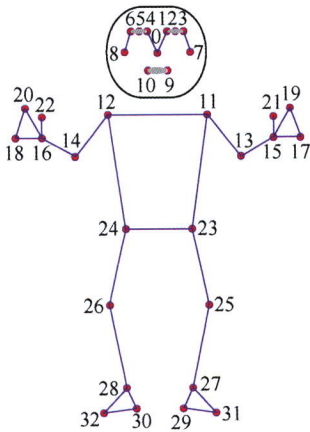


图2 姿势地标模型

2.评估流程

评估的整体流程如图3所示。首先待评估动作力量测量并输入程序，然后相机实时采集视频信息，计算机对采集的每帧图片进行识别，缓存待评估的姿态数据并实时计算频次，最后根据力与频次的结果查表实时输出评估结果。

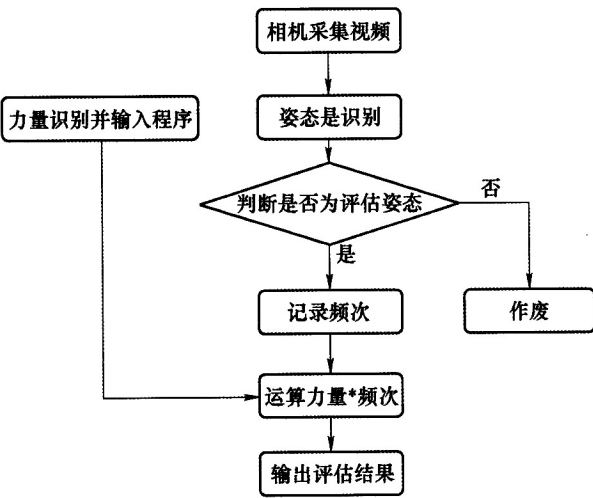


图3 评估流程

实现过程

1.力的检测与采集

对于力 F 的检测可以采用拾取搬运工件的质量来收集，或者对于水平推力可以采用推力计等方式采集实际的推力大小然后换算为kgf。

2.姿态采集与检测处理

采用Mediapipe的pose模块可以实时采集到人体33个关键关节的地标坐标。以分析弯腰的频率为例，采集到左侧（或者右侧）腰部关节点24的坐标以及与其相邻的点肩部点12、膝点26的坐标，通过三角函数就可以计算出腰部的角度 a （或者弯腰角度 $\angle b = 180^\circ - \angle a$ ，见图4）。

人体33个关键关节采集坐标采集方式实现Python代码如图5所示。

所有采集出来的关键坐标如图6所示，它是33个坐标集合，同时它是归一化的一个数据，如果需要实际位置需要再乘以图像的宽或长的像素。

当得到所有坐标后，只需采集到需要的几个关节

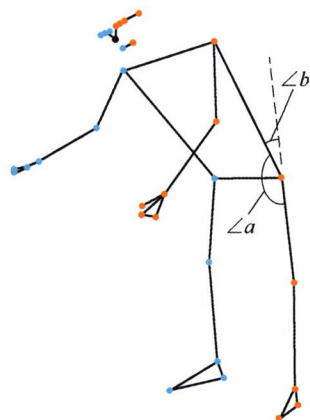


图4 腰部角度示意

```
joint_list = [[12, 24, 26],]
def find_angle(result_, joint_list_,):
    h, w, channel = image.shape
    for joint in joint_list_:
        a = np.array([result_.landmark[joint[0]].x,
                      result_.landmark[joint[0]].y])
        b = np.array([result_.landmark[joint[1]].x,
                      result_.landmark[joint[1]].y])
        c = np.array([result_.landmark[joint[2]].x,
                      result_.landmark[joint[2]].y])
        r_bend = np.arctan2(c[1] - b[1], c[0] - b[0]) - \
            np.arctan2(a[1] - b[1], a[0] - b[0])
        angle = np.abs(r_bend * 180.0 / np.pi) # 弧度转角度
        if angle > 180.0:
            angle = 360 - angle
        angle = int(angle)
    return angle
with mp_pose.Pose(
    min_detection_confidence=0.5, #可信度设置
    min_tracking_confidence=0.5) as pose: #各帧之间跟踪可行度
    while cap.isOpened():
        image_black = np.zeros((940, 1280, 3), np.uint8)
        C += 1
        success, image = cap.read()
        if not success:
            print("Ignoring empty camera frame.")
            continue
        image_height, image_width, _ = image.shape
        image = cv2.cvtColor(cv2.flip(image, 1), cv2.COLOR_BGR2RGB)
        image.flags.writeable = False #锁定图片
        #获取人体33点坐标信息
        results = pose.process(image)
        image.flags.writeable = True #释放图片
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        #获取全部坐标列表
        result_ = results.pose_landmarks
        #此处可以插入坐标角度分析计算函数
        # if results.pose_landmarks:
        #     ang, image = find_angle(image, result_, joint_list, C)
        #绘制出人体关节位置
        mp_drawing.draw_landmarks(
            image, #image
            results.pose_landmarks,
            mp_pose.POSE_CONNECTIONS,
            landmark_drawing_spec=
            mp_drawing_styles.get_default_pose_landmarks_style())
        #显示
        cv2.imshow('MediaPipe pose', image)
        if cv2.waitKey(5) & 0xFF == 27:
            break
    cap.release()
```

图5 关节坐标采集代码

点，如本文中12、24、26这个关节点坐标，记作列表 joint_list=[[12, 24, 26],]，同时构造一个三角函数求这个弯腰角度。在直角坐标系中，如果已知三点坐标可以通过反正切求出三点连线形成的三角形中任意一个夹角，如图7所示。 A 、 B 、 C 三点坐标分别记作 (A_x, A_y) , (B_x, B_y) , (C_x, C_y) , 求 $\angle ABC$ 。

人体弯腰的角度计算可以用如图8所示代码实现，

```
1 landmark {
2   x: 0.5913416743278503
3   y: 0.17020824551582336
4   z: -0.10148811340332031
5   visibility: 0.9999819993972778
6 }
7 landmark {
8   x: 0.5996149778366089
9   y: 0.15227100253105164
10  z: -0.11516246944665909
11  visibility: 0.9999804496765137
12 }
13 landmark {
14   x: 0.6029789447784424
15   y: 0.15230441093444824
16   z: -0.1152396947145462
17   visibility: 0.9999819993972778
```

图6 关节坐标实例

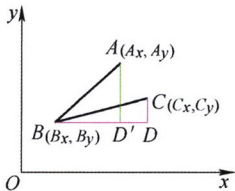


图7 角度示意

```
joint_list = [[12, 24, 26],]#
def find_angle(result_, joint_list_,):
    h, w, channel = image.shape
    for joint in joint_list_:
        a = np.array([result_.landmark[joint[0]].x,
                      result_.landmark[joint[0]].y])
        b = np.array([result_.landmark[joint[1]].x,
                      result_.landmark[joint[1]].y])
        c = np.array([result_.landmark[joint[2]].x,
                      result_.landmark[joint[2]].y])
        r_bend = np.arctan2(c[1] - b[1], c[0] - b[0]) - \
            np.arctan2(a[1] - b[1], a[0] - b[0])
        angle = np.abs(r_bend * 180.0 / np.pi) # 弧度转角度
        if angle > 180.0:
            angle = 360 - angle
        angle = int(angle)
    return angle
```

图8 求角函数

如果需要计算其他关节角度可以在joint_list列表中增加或者更改。本文用arctan2函数，来涵盖四个象限的角度。

把这个函数放在主关节检测程序中，就可以获取到采集到的视频的每帧中人体弯腰的角度，可以统一记录在一个list中（Angle[]）。这样就实现了角度的计算与实时获取，如图9所示。

通过持续视频分析，可以实时得到一组弯腰角度的数组Angle = [170, 174, 174, 173, 173, 173, 169, 166, 163, 164, ...]，数组的数量大小是已经采集到的视频帧（根据视频格式不同而不同，本文是32帧/min）。

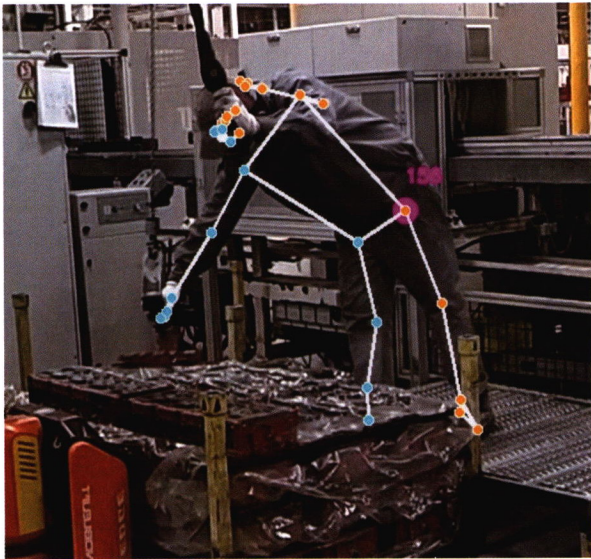


图9 工人吊缸盖弯腰识别

如图10所示，original_Bend_Angel_a为原始的腰部角度与帧数的曲线，波动下去是腰部角度较小的区域，即弯腰较大。为了方便计算，通过平滑函数signal.savgol_filter (Angle, 501, 3, mode= “mirror”) 将此曲线做下平滑得到 smoothed_Bend_Angel_a这条曲线，同时转换到弯腰角度 $\angle b=180-\angle a$ ，就得到了弯腰角度平滑曲线smoothed_Bend_Angel_b。这样再通过求波峰函数num_peak_3,=signal.find_peaks (y_smooth,height=50,distance=40) 对曲线进行波峰计算，可以得到波峰的个数num_peak_3也就是弯腰的次数,图9中两个弯腰超过 50° 的点，可以计数为弯腰2次。

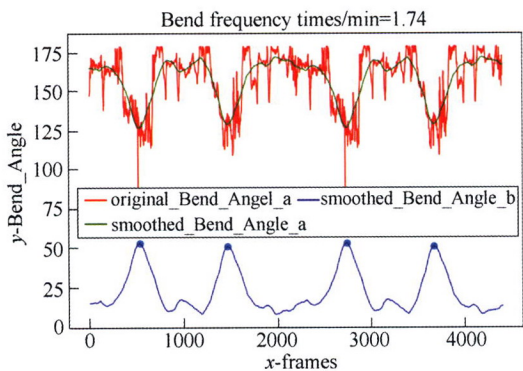


图10 弯腰角度与频率

3.频次计算

对于频率只需要把弯腰次数除以时间，时间可以通过视频数除以视频帧率。

4.汇总输出结果

获取了弯腰频次以及推动缸盖所用的力 F （当前有电动吊具，提升工件的力 $<2\text{kgf}$,推动工件的力测量值 $<2\text{kgf}$ ），当前频次为每分钟1.74次，因此依据表1可以评估出此处作业弯腰完全满足人机工程要求。

实际使用验证

图10是实际测试的一个记录，弯腰的角度可以准确采集与计算，关于弯腰的次数是通过弯腰角度列表绘制曲线并进行平滑处理后采集波峰的方式来获取，实际应用过程中可以通过平滑窗口以及阶次微调找到合适的平滑参数。通过实际分析整体准确度能达到使用要求，本示例能够准确识别弯腰次数以及频次，并对人机工程进行快速评估。

结语

本文通过“弯腰拾取”人机工程识别介绍了视觉姿态检测在发动机行业中的应用，通过这种方法可以快速一次性评估人机工程中的大多项目，例如水平推物体、转身取件、低头检查等等，从而大大节约工业工程师与安全工程师的评估工作量，同时也可以提高评估的效率。这种方法在防错识别上也具有一定的可行性，例如可以通过识别手部动作，判断发动机装配线上操作人员是否按规定位置安装了零件，从而判断是否错漏装。

参考文献

[1] 夏帮贵.OpenCV计算机视觉基础教程（Python版）
[M].北京：人民邮电出版社，2021. **AUTO1950**