浙江大学

本科课程作业报告

课程名称 《机器视觉与图像处理》

题目 产线违规取放

姓名 吴霜阳

专业 光电信息科学与工程

学号 **3230105822**

提交日期 **2025 年 12 月 28 日**

1. 实际问题分析

在工业**PCB**生产线上，某些关键工序严格禁止操作人员徒手接触电路板，以防静电损坏或污染元件。然而在实际生产环境中，要实时监控并发现人工违规取放操作具有很大挑战。传统依靠人工监看监控视频不仅效率低下，还容易因疲劳漏判。

计算机视觉方案需要克服以下难点：

- **手部检测条件复杂**：工人在生产中可能佩戴**防静电手套**，手套颜色各异且不具备裸手的肤色特征，导致常规基于肤色或特征点的算法难以识别。此外，手部可能出现**部分遮挡**（如被设备或工件挡住）或快速运动产生**运动模糊**，这都增加了检测的难度。
- **背景环境干扰**：生产线背景通常包含运动的传送带、机械臂、产品板卡等。其中某些运动或物体颜色可能与手部相似，容易引发误检。例如传送带的轻微振动、光源变化造成的阴影，都可能在视觉上产生类似手部的假信号。算法需要能够区分**人体手部**与背景正常运动之间的差异。
- **光照变化**：工厂环境下光照条件复杂多变，可能存在强反光或局部阴影。如果使用RGB颜色空间直接检测，光照变化会影响所有通道的像素值，导致检测不稳定。监控系统必须具有一定的**光照鲁棒性**，在亮度波动、不同肤色人群条件下都能稳定检测。
- **实时性和可靠性**：违规取放动作往往发生突然且持续时间短暂，要求系统能够**实时处理视频流**并在手刚一侵入画面时及时报警。同时应避免频繁误报和漏报，既要**灵敏**捕捉到真正的违规行为，又要**稳定**不会因偶发噪声反复触发报警。这对算法的时序滤波和决策逻辑提出了要求，需要在**响应速度**与**稳定性**之间取得平衡。

1.1 极其复杂的背景噪声

与实验室纯色背景不同，PCB产线是一个高度动态且色彩斑斓的环境。传送带通常为黑色或深绿色，且表面可能带有磨损痕迹；传送带两侧分布着金属导轨、传感器支架、气动装置以及各种颜色的线缆。更关键的是，被检测对象——PCB板本身，就包含了绿色（阻焊层）、金色（焊盘）、白色（丝印）以及黑色（芯片）等多种颜色。当工人的手（特别是佩戴了与背景同色系的防护手套时）伸入这一区域，前景与背景的边界变得极其模糊。例如，佩戴白色防静电手套的手在抓取带有白色丝印的PCB时，基于颜色梯度的边缘检测算法极易失效；而佩戴绿色手套的手在绿色传送带上操作时，色度分割算法将完全崩溃。

1.2 高度动态的遮挡与形变

“取放”是一个连续的动态过程，涉及手部的伸入、抓握、提升和撤出。在这个过程中，手部形态发生剧烈变化。

- **自遮挡（Self-Occlusion）：** 在抓握瞬间，手指会弯曲并相互重叠，导致手掌的几何特征

（如掌心、指缝）消失，使得基于模板匹配或简单特征点的算法无法维持跟踪。

- **物体遮挡（Object-Occlusion）：** 当手部抓起PCB板时，板卡本身会遮挡住手掌的大部分区域，仅露出手腕或指尖。此时，视觉系统看到的不再是一个完整的“手”，而是一个被切割的肢体碎片，这对于依赖完整轮廓特征的检测器是巨大的考验。

1.3 光照干扰与镜面反射

工业现场通常采用高强度的荧光灯或LED照明，这虽然保证了人眼的可见度，但对机器视觉却可能构成干扰。金属元器件引脚、焊点以及传送带的金属边缘在强光下会产生强烈的镜面反射，在图像中形成高亮的光斑。当传送带运行时，这些光斑随之移动，极易被基于**帧差法（Frame Difference）或光流法（Optical Flow）**的运动检测算法误判为移动的目标物体。此外，电网电压波动引起的灯光频闪也会导致整幅图像的亮度发生周期性跳变，触发大面积的误报。

1.4 实时性与边缘计算的矛盾

产线监控系统要求极高的实时性。**如果算法处理一帧图像的耗时超过33毫秒（即低于30FPS），系统就会产生累积延迟。**在违规取放发生的瞬间（通常仅需0.5至1秒），如果报警延迟超过2秒，违规动作可能已经完成，失去了即时制止的意义。然而，要在嵌入式工控机或边缘计算设备上运行复杂的深度神经网络，往往受限于算力。因此，项目必须在算法精度与计算效率之间寻找极其微妙的平衡点，这也是本次大作业将“考察程序对不同样本的适用性”列为核心评分标准的原因之一。

---

2. 解决策略与方案

针对上述复杂的工业场景挑战，本项目并未单一依赖某一种技术，而是设计了一套融合了深度学习、色彩空间分析与传统形态学处理的混合视觉算法架构。

2.1 核心开发栈与库

2.1.1 OpenCV (Open Source Computer Vision Library)

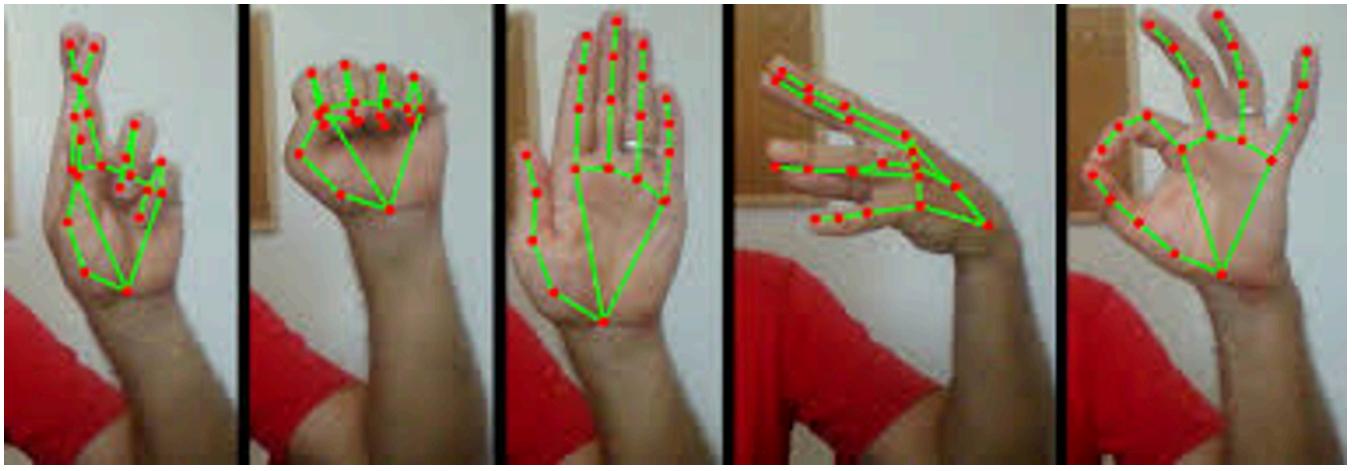
OpenCV是本项目的基础图像处理引擎。

作为工业界事实上的标准库，它提供了底层的高效C++实现和便捷的Python接口。

- **作用：** 在本项目中，OpenCV承担了视频流的采集（`cv2.VideoCapture`）、预处理（如高斯模糊降噪、图像金字塔缩放）、色彩空间转换（BGR转RGB/YCrCb/HSV）以及最终的渲染绘图（`cv2.rectangle`, `cv2.putText`）工作。其提供的**形态学操作（腐蚀、膨胀、开闭运算）**是处理肤色掩膜和运动前景掩膜的关键步骤，用于去除噪点和填充空洞。

2.1.2 MediaPipe Hands

MediaPipe是Google开发的一款专为移动端和边缘设备设计的跨平台机器学习框架。其中MediaPipe Hands解决方案是解决手部检测难题的核心武器。



- **原理：** 它采用单次推理的BlazePalm检测器首先定位手掌区域，**然后通过一个回归模型在手掌区域内预测出21个精细的3D手部骨架关键点。**在生产线场景中，摄像头通常固定在流水线上方。BlazePalm 对全图像进行扫描，输出手掌的定向边界框。这个边界框不仅包含位置信息，还包含手掌的旋转角度，允许系统将倾斜或旋转的手部图像矫正为标准姿态，从而降低后续模型的学习难度，提高识别准确率。
- **作用：**
    1. 相比于传统的YOLO目标检测框，MediaPipe直接输出骨架信息，**这使得算法不仅能知道“哪里有手”，还能通过关键点之间的几何关系（如指尖与掌心的距离）判断手部的姿态（张开、握拳）**；
    2. MediaPipe针对手部的非刚性形变和自遮挡进行了大量数据训练，在裸手检测场景下具有极高的鲁棒性和抗干扰能力，是本系统的主力检测器。
    3. MediaPipe作为主要检测手段，利用深度学习模型对裸手或带手套的手进行检测，当手进入画面时迅速给出位置和形状信息。

2.1.3 Numpy

- **作用：** 作为Python科学计算的核心库，Numpy在本项目中主要用于高效的矩阵运算。在**处理图像掩膜（Mask）、计算检测框的交并比（IoU）、以及进行坐标归一化和阈值过滤**时，Numpy的向量化操作比Python原生循环快数个数量级，保证了实时性。

2.1.4 Tkinter & Threading

- **介绍：**为了方便演示和实际使用，系统基于Python的Tkinter库实现了一个**可视化监控界面。**
    - GUI提供了友好的交互控件，包括：摄像头实时监控启动、一键选择并播放示例视

频、暂停/继续检测按钮、手动停止和重置报警按钮等。
    - 界面左侧实时显示视频画面，并在画面上叠加绘制检测结果（如手部框、关键点）、ROI区域、状态栏和警报提示等信息。
    - 右侧面板提供状态文本（“正常监控中/警告/危险”）的实时更新、累计**报警次数**和**违规帧计数**的统计显示，以及一个日志列表滚动记录每次警报触发的时间和信息。
    - 当检测到违规时，GUI除了在视频上以**红框**高亮提示外，还会将事件记录追加到日志列表顶端，必要时弹出警告对话框提醒操作员注意。通过GUI界面，用户可以直观地观察监控区域的实时状态，并获取每次违规的证据截图和时间记录，从而提升系统的**可用性**和**用户体验**。
- **作用：** 系统采用了多线程架构：**主线程负责UI渲染和事件响应，子线程负责繁重的视频处理和算法推理。**这种设计避免了图像处理阻塞界面刷新，确保了在报警触发时界面仍能流畅响应用户的操作（如重置、暂停）。

2.2 混合检测算法设计

2.2.1 基于深度学习的骨架提取

这是优先级最高的检测通道。每一帧图像首先被送入MediaPipe Hands模型。

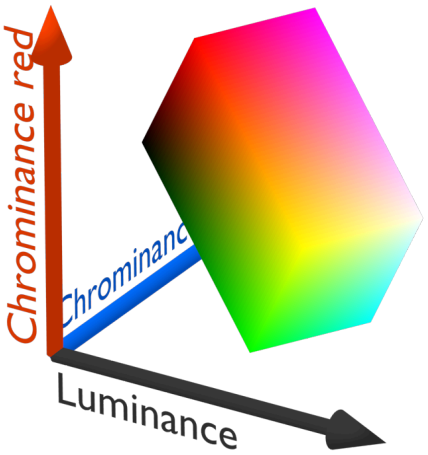
- **技术细节：** 输入图像被转换为RGB格式。模型输出包含21个关键点的列表。系统遍历这些关键点，计算出最小包围盒（Bounding Box），并在此基础上增加预设的Padding（如20像素），以确保框住整个手部而不仅仅是骨架中心。
- **优势与局限：** 对裸手检测精度极高，误报率极低。**但在工人佩戴厚重的工业手套或手套纹理缺乏特征时，BlazePalm检测器可能会漏检。**

2.2.2 基于YCrCb色彩空间的肤色分割

作为MediaPipe的补充，系统引入了基于色彩的检测逻辑。

在数字图像处理中，最常见的颜色表示方式是 RGB（红、绿、蓝）。然而，RGB 颜色空间在肤色检测任务中存在先天的缺陷。RGB 的三个通道高度相关，且都包含亮度信息。这意味着，当环境光照发生变化（时，皮肤像素的 R、G、B 值都会发生剧烈波动，导致基于固定阈值的检测算法失效。

为了解决这一问题，系统采用了 **YCrCb** 颜色空间。YCrCb 是一种将亮度与色度分离的编码方式：



- **Y (Luminance)**：代表亮度分量，包含了图像的灰度信息。
- **Cr (Chrominance-Red)**：代表红色差分量。
- **Cb (Chrominance-Blue)**：代表蓝色差分量。

数学上，从 RGB 到 YCrCb 的转换是一个线性变换过程。OpenCV 使用特定的转换矩阵来实现这一过程。

这种分离的物理意义在于：人类肤色的差异（无论种族、深浅）主要体现在亮度（Y）上，而在色度平面（Cr-Cb）上的分布却具有惊人的聚类一致性。

换言之，在去除亮度影响后，人类的肤色倾向于落在 Cr-Cb 平面的一个紧凑椭圆区域内。这使得 YCrCb 空间对于光照强度的变化具有天然的**鲁棒性**，非常适合光照条件可能不稳定的工业生产环境。

- **实现步骤：**
    1. 图像转换至YCrCb空间。
    2. 利用`cv2.inRange`函数，根据预设的阈值（如`min_YCrCb`=, `max_YCrCb`=）提取肤色区域。
    3. **形态学滤波：**系统对掩膜依次应用了**开运算**和**闭运算**：

> **Important**
>
> 1. **开运算（Opening = Erosion followed by Dilation）**：
>   - **操作原理**：先腐蚀后膨胀。
>   - **系统作用**：腐蚀操作会剥离物体边缘的像素。对于背景中那些面积小于结构元素（5x5）的孤立噪点，腐蚀会将它们完全消除。随后的膨胀操作会恢复较大物体（即手部）的原始尺寸，但那些已被消除的噪点无法恢复。因此，开运算有效地**去除了背景噪声**，切断了细小的连接通道。
> 2. **闭运算（Closing = Dilation followed by Erosion）**：

    - **操作原理**：先膨胀后腐蚀。
    - **系统作用**：膨胀操作会填补物体内部的细小空隙。在肤色检测中，手掌内部常因反光、阴影或血管纹理导致检测断裂，出现黑色孔洞。膨胀能将这些断裂区域连接起来。随后的腐蚀操作将物体外缘收缩回原状，但内部已被填补的孔洞得以保留。因此，闭运算有效地**填充了前景孔洞**，增强了连通域的完整性。
  3. **迭代膨胀操作：**
    - 这一步旨在适度扩大检测区域的边界。
    - 由于早期的腐蚀操作可能对物体边缘造成了一定程度的侵蚀，或者因为阈值设置偏保守导致手部边缘未被完全捕获，额外的膨胀有助于恢复甚至略微夸大物体的范围。
4. **轮廓分析：** 计算连通域的面积和长宽比，剔除面积过小或形状极其细长的非手部区域。

2.2.3 基于帧差法的运动侦测

针对佩戴特殊颜色手套导致前两种方法均失效的极端情况，系统启用了基于运动的兜底逻辑。

- **技术原理：** 利用视频序列的时间相关性。通过计算当前帧与前一帧（或背景模型）的像素差值，提取出发生变化的区域。数学表示为：

  $$\Delta I(x,y) = |I_t(x,y) - I_{t-1}(x,y)|$$

  其中，I_t 是当前帧，I_{t-1} 是上一帧。当 ΔI 超过设定的 MOTION_THRESHOLD 时，该像素被判定为运动前景。相比于复杂的背景建模算法，帧差法具有极高的计算效率，且对光照的缓慢变化具有天然的适应性。

  帧差法的引入解决了**“彩色手套”**难题：当工人佩戴蓝色、黑色等非肤色手套时，**MediaPipe因缺乏清晰特征点可能漏检，肤色检测也因颜色不符而失效**；但手套通常与背景（如PCB板）的颜色有对比，只要手在移动，帧差法就能捕获到运动区域的轮廓，从而使系统仍能检测到违规入侵。

  运动检测模块为检测**非裸手**提供了关键保障。

- **实现步骤：**
  1. 图像灰度化并进行**大核高斯模糊（Gaussian Blur, ksize=21x21）**。

     > **Tip**
     >
     > 为了抑制环境噪声，系统引入了一个极为激进的预处理步骤：使用 **(21, 21)** 尺寸的**高斯模糊核（Gaussian Blur Kernel）**对灰度图像进行平滑处理。
     >
     > 通常图像预处理使用 3x3 或 5x5 的核来去除传感器热噪声。使用 21x21 的巨

> 型核在图像处理中并不常见，其物理意义在于构建一个截止频率极低的**空间低通滤波器** 。
>
> - **抑制传送带纹理**：传送带表面的细微纹理、划痕或灰尘在传输过程中会产生高频的像素变化。21x21 的高斯核会将这些细节完全抹平，使背景呈现为均一的灰度块。
> - **对抗机械振动**：摄像头的微小抖动会导致边缘像素跳变。大核模糊使得边缘变得极其平缓（Gradient smoothening），从而大幅降低了因亚像素级抖动产生的差分信号幅度。
> - **保留宏观运动**：工人的手臂挥动属于大尺度的宏观运动（Macro Motion），对应图像中的低频分量。即便是经过严重模糊，手臂挥动产生的亮度变化区域依然足够大且显著，能够穿透低通滤波器被保留下来。
>
> 这种"重度模糊 + 帧差"的组合，实际上是针对"传送带背景下的手臂检测"这一特定场景定制的**带通滤波器**，它滤除了高频的纹理噪声和微颤，仅保留了大块物体的位移信号。

2. 计算绝对差分图（`cv2.absdiff`）并二值化。
3. **几何特征过滤：** 系统设定了严格的面积阈值（`MOTION_AREA_THRESHOLD`）和长宽比阈值（0.3 < Aspect Ratio < 3.0）。**只有符合这些几何约束的运动块才会被判定为"疑似手部"。**这有效过滤了传送带上单向移动的PCB板（通常长宽比固定且运动轨迹平稳）。

2.2.4 非对称报警策略（连续帧计数滤波）

为了减少误报并确保报警响应迅速，系统采用了**非对称的状态滤波算法。**

具体而言，在判定手部持续存在时采用比解除报警更严格的条件：

- 需要一定连续帧数的正检才能确认进入**报警状态**
- 只需极少帧的负检即可**迅速解除**报警

实现过程：

- 手部检测模块维护一个最近若干帧的检测历史，通过**多数投票**平滑抖动（要求最近5帧中超过半数检测到手才算"稳定检测"）；
- 报警模块设定阈值，**检测到手的连续计数达到8帧才真正触发严重报警**，而一旦某帧判断手消失则立即清零计数。

这样双层保障下，实现了"**快速触发，即时撤销**"的效果：

- **8帧连续检测确保不是偶然误检才拉响警报，而只要手一离开，2帧内（约0.1秒）系统就恢复正常监控状态，避免报警状态在手移开后不必要地滞留**

- 非对称报警策略有效减少了环境噪声造成的频繁报警，提高了系统稳定性和响应速度。

2.2.5 决策融合机制

三种检测轨道会产生三组候选框。系统采用了一种加权融合策略：

1. **并集收集：** 收集所有轨道输出的候选框。
2. **重叠抑制 (IoU Filtering)：** 计算框与框之间的交并比。如果两个框的重叠度超过阈值（如0.3），则认为是同一个目标。
3. **优先级保留：** 在重叠的框中，优先保留MediaPipe的结果（置信度最高），其次是肤色检测结果，最后是运动检测结果。这种机制确保了在深度学习模型有效时，系统具有最高的精度；在模型失效时，系统仍能依靠传统算法维持基本的检出率，实现了鲁棒性与精度的双重保障。

优先级	检测源	技术特征	触发条件	优势	劣势
Tier 1	MediaPipe Hands	深度学习	`len(mp_boxes) > 0`	极高精度，含骨架语义	怕手套，怕遮挡，算力高
Tier 2	运动分析	物理光学	MP失效且 `Area > 1.2*Thresh`	召回率高，不受手套影响	无语义，易受背景干扰
辅助	肤色检测	颜色统计	与Tier 1/2 并行	验证Tier 1，辅助Tier 2	怕光照变化，怕肤色背景

系统在代码逻辑上采用了 **短路求值** 策略：只要 Tier 1 成功，系统便信任该结果并结束检测循环。

这种设计不仅**保证了在正常情况下的高精度，也优化了计算效率。**只有在 Tier 1 失效时，计算开销极低的 Tier 2 才会作为补救措施介入，确保系统不会漏检违规行为。

3. 代码结构与实现过程梳理

3.1 核心类图与文件组织

项目主要由以下五个核心Python文件构成，它们通过模块导入相互协作：

- `video_processor.py` **(Model):** 负责底层的视频流I/O、帧级预处理及渲染。
- `hand_detector.py` **(Algorithm):** 封装了混合检测策略，是系统的视觉处理核心。
- `alarm_system.py` **(Controller):** 实现了有限状态机，管理系统的报警状态流转。
- `gui.py` **(View):** 构建用户交互界面，负责图像展示与用户指令接收。
- `main.py` **(Entry):** 程序入口，负责参数解析、依赖注入与主循环启动。

3.2 模块详解与实现细节

3.2.1 视频流处理与优化 (`VideoProcessor`)

- **帧率控制与跳帧:** 为了适应算力有限的部署环境，代码引入了`frame_skip`参数。在`read()`方法中，系统会根据配置（如每隔2帧处理一次）丢弃中间帧。这虽然在时间轴上引入了微小的非连续性，但显著降低了CPU负载，确保了检测线程不会因积压过多帧而导致巨大的延迟。
- **分辨率缩放:** 高分辨率图像（如1920x1080）直接输入MediaPipe或进行像素级形态学运算会消耗大量算力。该类在预处理阶段使用`cv2.resize`将图像按比例缩小（如0.5倍）。在低分辨率下进行检测，再将结果坐标映射回原图，可以在几乎不损失检测率的前提下提升30%-50%的处理速度。
- **中文字符渲染:** OpenCV原生的`putText`函数不支持Unicode字符（如中文）。为了在界面上显示"违规检测"等中文警示，`FrameRenderer`类中实现了一个巧妙的转换：将OpenCV的BGR数组转换为PIL（Python Imaging Library）图像对象，利用PIL强大的字体渲染功能绘制中文，然后再转换回OpenCV格式。

3.2.2 视觉算法引擎 (`HandDetector`)

- **ROI裁剪:** 在`detect`方法的入口处，代码支持根据配置文件定义的ROI区域（x, y, w, h）对图像进行裁剪。这在工业产线中至关重要，因为监控画面往往包含大量无关区域（如过道、设备外壳）。通过只处理传送带上方的核心区域，不仅减少了计算量，还彻底排除了背景区域行人走动造成的误报。
- **时域稳定性滤波:** 为了防止检测结果的"闪烁"，代码中虽然没有显式的卡尔曼滤波，但在逻辑上通过`alarm_system`的状态机实现了类似的功能。而在检测器内部，对于肤色和运动

检测的轮廓，采用了严格的面积和长宽比阈值过滤，这本质上是一种基于几何先验的空间滤波。

3.2.3 报警状态机逻辑 (`AlarmSystem`)

`alarm_system.py`实现了一个典型的有限状态机，**用于将瞬时的视觉检测结果转化为稳定的业务状态。**

状态 (State)	触发条件 (Transition Condition)	行为 (Action)
LEVEL_NORMAL	`violation_count` = 0	重置计数器，清除UI警示
LEVEL_WARNING	0 < `violation_count` < `THRESHOLD`	界面显示黄色边框，计数器累加
LEVEL_DANGER	`violation_count` >= `THRESHOLD`	**触发报警**（声音+红框），记录日志

- **防抖动逻辑：** 系统维护一个`violation_count`计数器。只有当违规行为在连续多帧中持续存在（累积超过`ALARM_FRAME_THRESHOLD`）时，才会从WARNING切换到DANGER。这有效地屏蔽了因光照突变或飞虫经过引起的单帧误检。
- **冷却机制 (Cooldown):** 一旦触发报警，系统会记录`last_alarm_time`。在随后的`ALARM_COOLDOWN_SECONDS`内，即使检测到手部，也不会重复触发新的报警记录。这一设计防止了同一事件导致报警日志被刷屏，体现了良好的用户体验设计。

3.2.4 多线程GUI架构 (`GUI`)

`gui.py`展示了如何在Python的GIL（全局解释器锁）限制下实现流畅的视频播放。

- **生产者-消费者模型:** 检测线程（生产者）不断处理视频帧，并将结果放入一个大小受限的`queue.Queue`（最大长度为2）。主线程（消费者）通过`root.after(15, self._poll_frame_queue)`定时轮询队列。
- **队尾丢弃策略:** 队列的最大长度限制为2是非常关键的设计。如果主线程渲染速度慢于检测线程，队列会满。此时检测线程会阻塞或丢弃新帧（取决于具体实现）。这确保了界面上显示的永远是能够获取到的最新画面，而不是几秒前的延迟画面，这对于实时监控系统至关重要。

4. 缺陷分析与未来展望

尽管目前的系统在给定的测试样本中表现出了基本的检测能力，但作为一个工业级应用的原型，它在鲁棒性、泛化能力和智能化程度上仍有显著的局限性。

4.1 现有系统的缺陷深入剖析

4.1.1 对复杂光照环境的脆弱性

当前使用的帧差法（Motion Detection）和YCrCb肤色检测对光照条件有着先天的依赖。

- **问题描述：** 在实际工厂中，照明可能会因为大型设备的启动而发生电压波动，导致频闪；或者天窗射入的自然光随云层变化。这些全局的亮度变化在帧差法中会被计算为全图的像素差异，导致整个画面被误判为运动目标（大面积误报）。
- **数据佐证：** 研究表明，简单的背景减除法在动态光照下的误报率可高达20%以上。虽然YCrCb空间分离了亮度，但在极暗或过曝条件下，色度分量（Cr, Cb）也会发生畸变，导致肤色检测失效。

4.1.2 手套与遮挡导致的漏报

- **手套问题：** MediaPipe Hands主要基于裸手数据训练。当工人佩戴绿色或蓝色工业手套时，手部的纹理特征被掩盖，且颜色与背景混淆。此时，MediaPipe置信度降低，肤色检测完全失效，仅剩不稳定的运动检测在工作，极易导致漏报。
- **遮挡问题：** 在取放动作的最后阶段，手部可能大部分伸入机器内部或被PCB板遮挡，导致可见区域不足以构成完整的骨架特征。

4.1.3 缺乏意图理解

- **逻辑缺陷：** 当前算法仅能检测“ROI区域内是否有手”。然而，并非所有出现在该区域的手部动作都是违规的。例如，工人可能只是进行快速的设备检查、清除异物或在设备停机状态下操作。系统无法区分“正常的维护动作”与“违规的取放动作”，也无法理解设备当前的运行状态（如传送带是否停止），这可能导致对合法操作的误报警。

4.2 未来技术演进展望

4.2.1 引入SOTA深度学习模型：YOLOv8-Pose 与 Transformers

为了解决手套检测和遮挡问题，未来的迭代应放弃依赖手工特征（肤色、运动），转向端到端的深度学习方案。

- **YOLOv8-Pose / YOLO11-Pose:** 相比MediaPipe，YOLO系列的Pose模型在COCO Keypoints等大型数据集上进行了训练，且支持多目标检测。更重要的是，YOLO模型具有

极强的迁移学习能力。我们可以采集专门的“佩戴工业手套的手部数据集”，对YOLOv8-Pose进行微调。这将使其能够通过学习手套的轮廓和纹理特征，在佩戴手套的情况下依然精准回归出关键点，显著提升鲁棒性。

- **轻量化部署:** 针对边缘设备，可以使用TensorRT或ONNX Runtime对YOLO模型进行FP16或INT8量化，确保在Jetson Orin或树莓派等设备上达到30FPS以上的推理速度8。

4.2.2 时序动作识别 (Temporal Action Recognition)

为了解决意图理解问题，必须引入时间维度。

- **LSTM / GRU:** 将连续N帧的手部关键点坐标序列作为输入，送入长短期记忆网络（LSTM）。训练模型学习“伸入-抓取-收回”这一特定时空轨迹的特征。LSTM能够记忆之前的状态，从而区分短暂的误入（如手挥过）和有意识的取放操作。
- **Video Transformers:** 对于更复杂的场景，可以考虑使用VideoMAE或SlowFast等基于Transformer的视频分类模型。它们不仅关注手部，还能结合背景信息（如PCB板是否跟随手部移动），从而实现更高级别的语义理解——即确认“取放”这一事件的发生，而不仅仅是“手存在”。

4.2.3 多模态感知融合 (Multi-modal Fusion)

单一的2D视觉始终存在透视投影带来的深度丢失问题。

- **RGB-D 深度视觉:** 引入Intel RealSense或ToF（Time of Flight）相机。深度图（Depth Map）天然对光照变化免疫，且能提供绝对距离信息。
- **电子围栏逻辑:** 利用深度信息构建虚拟的3D电子围栏。只有当手部不仅在X-Y平面进入ROI，且在Z轴方向突破安全距离（即真正接触到流水线表面）时才触发报警。这将彻底根除因透视效应（如手在传送带上方悬空经过）造成的误报。