文章编号：2095-2163(2022)08-0076-09 中图分类号：TN929.5 文献标志码：A

一种基于视频图像的芯片封装质量分析方法

杨盼盼，陈庆奎

（上海理工大学 光电信息与计算机工程学院，上海 200093）

摘 要：工业生产过程中，芯片封装的质量直接影响最终产品的性能。本文提出一种对芯片封装进行质量分析的方法，使用 YOLOv3 目标检测算法对传输的物料条视频图像进行芯片目标定位，通过距离检测方法计算识别框间距，判断缺件缺陷并切割出单芯片图像；对单芯片图像使用区域划分的大津法进行引脚分割，使用最小二乘法拟合芯片引脚最小外接矩，并将矩形中心点的延长线和矩形边线的交点作为原点，建立相对直角坐标系，通过矩形边线与 y 轴方向夹角判断歪斜缺陷；使用改进的 SIFT 算法对模板图像和待测图像进行特征点配对，根据匹配点对的分布区域及占比进行缺陷分类判定。实验证明，该方法能够针对工业生产条件下的芯片封装质量进行分析，且速度较快，在实际工业生产中具有较好的效果。

关键词：目标检测；质量分析；YOLOv3；SIFT

A video image-based chip packaging quality analysis method

YANG Panpan, CHEN Qingkui

(School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China)

【Abstract】 In the industrial production process, the quality of the chip packaging directly affects the performance of the final product. Based on this, a method for quality analysis of chip packaging is proposed, using YOLOv3 target detection algorithm to locate the chip target on the transmitted material strip video image, calculating the recognition frame spacing by distance detection method, judging the missing parts defects and cutting out the single chip image; using the Otsu method of region division for pin segmentation on the single chip image, fitting the minimum external moment of the chip pin using the least squares method. The proposed method is based on the intersection of the extension of the rectangle center point and the rectangle edge line as the origin, and establishes a relative cartesian coordinate system to determine the skewed defects by the angle between the rectangle edge line and the y - axis direction; the improved SIFT algorithm is used to pair with the feature points between the template image and the image to be tested, and the defects are classified according to the distribution area and the percentage of the matched point pairs. The experiments prove that the proposed method can analyze the chip packaging quality for industrial production conditions, and it is faster and has better results in actual industrial production.

【Key words】 target detection; quality analysis; YOLOv3; SIFT

0 引 言

随着信息化技术的发展，电子设备逐渐成为日常必需品，因而对电子系统微型化、集成化的要求也越来越高。芯片作为电子产品基础性的重要组成部分，在生产过程中要经历数十道工序，生产完成后需装入物料带并塑封，保护芯片内部及引脚部分，封装后的芯片也更便于运输。为了改善元器件封装效率，一般通过改善封装的结构和方法来优化封装设备。

集成电路芯片封装完成后，对其进行质量分析是芯片生产流程中一个重要的环节。传统的人工检测主观性较强，耗费人力物力，长时间的检测工作还会产生视觉疲劳，进而导致检测正确率下降。

本文提出了一种基于视频图像的芯片封装质量分析方法，将深度学习 YOLOv3 算法结合图像处理技术应用于芯片封装质量的检测。在原有的车间传送带正上方安装 CCD 工业相机，获取封装后的物料条视频图像，通过目标检测 YOLOv3 算法进行芯片目标定位，并分割单张芯片图像。对获取到的单芯片图像进行缺陷分类判定，采用 Harris 算子结合尺度不变特征变换（Scale-Invariant Feature Transform，SIFT）特征描述方法，进行模板图像与各类缺陷图像的匹配，并对匹配点采用特征聚类方法剔除误配点，提高匹配准确率，实现对各类缺陷残次品（Not Good，NG）的判定。实验证明，本文方法不受主观因

基金项目：国家自然科学基金（61572325）；上海重点科技攻关项目（19DZ1208903）。

作者简介：杨盼盼（1998-），女，硕士研究生，主要研究方向：图像处理；陈庆奎（1966-），男，博士，教授，博士生导师，CCF 会员，主要研究方向：计算机集群、并行数据库、并行理论、物联网等。

收稿日期：2022-02-20

素的影响，且可以量化评估最终的检测结果。

1　相关工作

针对元器件封装品质分析的系统主要是将计算机视觉与元器件表面检测技术相结合。自动化机器视觉检查可以减少人工工作量和劳动力成本，并提高检测精度。文献[1]提出一种基于数字图像处理的印刷电路板(Printed Circuit Board，PCB)自动光学检测方法，建立了将标准图像和待测图像进行对比的检测系统，该方法对于光照及芯片摆放位置要求极高，易造成误判。文献[2]提出了一种基于数学形态学的 PCB 自动缺陷检测算法，以图像腐蚀后的边缘距离信息为参照，对比轮廓特征后进行缺陷识别。文献[3]提出了基于梯度方向信息熵的 PCB 缺陷检测方法，使用领域梯度方向信息熵提取缺陷特征，构造特征向量，作为 SVM 分类器的训练样本，能够对 PCB 裸板存在的常见缺陷进行快速、精确的定位。文献[4]提出了基于标注框的宽高聚类生成候选框的 Faster R-CNN 的零件表面缺陷检测算法，并引入多级感兴趣区域(Region of Interest，RoI)池化层结构以提高检测准确性。由于在真实的工业环境中能提供的缺陷样本太少，无法进行大量数据支撑的训练，所以将深度学习方法直接应用在表面缺陷检测中的研究，迄今为止也仍不多见。

图像匹配是指使用有效的匹配算法为 2 个或多个图像数据找到相同或相似的提示点的过程。目前主流的图像匹配算法有基于灰度和基于特征两种。其中，基于灰度的图像匹配算法正确率较高，但是计算量巨大，导致匹配效率低下[5]；基于特征的匹配是从原始图像的灰度信息提取特征，在特征空间进行匹配，特征携带的有用信息比灰度值更丰富，易于适应复杂的图像变换，例如几何畸变、不同分辨率、不同角度的图像变换等。因此，基于特征的匹配算法速度更快，检测效果也更加准确。Harris 等人[6]提出了 Harris 角点检测算法，该算法具有一定的旋转不变性，对光照与噪声也有一定的鲁棒性，但 Harris 算法却不具备尺度不变性。Lowe[7]于 2004 年提出 SIFT 算法，是设计改进后的关键点匹配算法，虽然 SIFT 耗时比较长，但无论从种类数量、还是整体质量来看，SIFT 的性能都是最好的，即使少数的几个物体也可以产生大量的 SIFT 特征向量。因此本文将 Harris 算子与 SIFT 算法结合，实现图像关键特征的描述及匹配，较好地解决了 Harris 算子的分类准确性低和 SIFT 算法耗时长等问题。

2　芯片封装质量分析方法

芯片封装后需要对物料条进行质量分析，常见的质量问题主要有塑封不严密、芯片缺件、芯片歪斜、芯片反置以及芯片正面的 logo 印刷错漏等情况。

本文研究的芯片封装质量分析系统包括 3 个模块：图像采集模块、芯片定位模块和缺陷检测模块。芯片封装质量分析过程示意如图 1 所示。

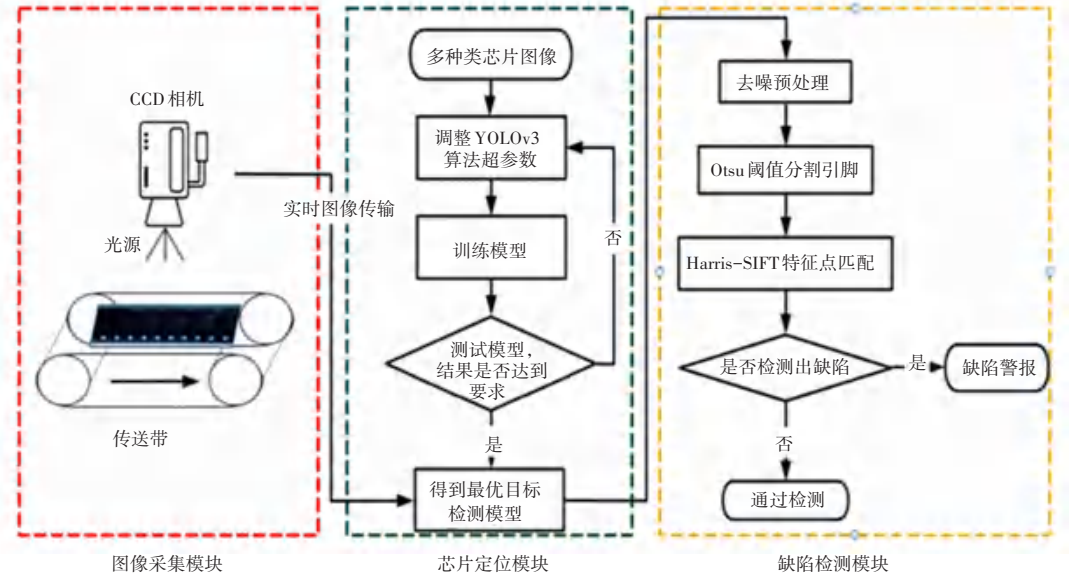


图 1　芯片封装质量分析过程示意图

Fig. 1　Diagram of chip packaging quality analysis process

2.1 图像采集模块

使用工业 CCD 相机采集图像,将图像传感器接收到的光学图像转化为计算机能够处理的电信号。将物料条放置在生产线的传送带卡槽内固定,防止其由于运动惯性发生位置偏移;传送带定速将物料条传送至工业相机光源下,相机以固定时间间隔自动拍照。

CCD 相机采集到的视频图像传入系统,存储结构包括芯片生产型号、物料条图像的编号、三通道图像、图像采集时间、图像高度所占像素以及图像宽度所占像素。

2.2 芯片定位模块

相机采集图像后传入系统,使用训练好的 YOLOv3 算法检测目标芯片,根据检测结果返回的坐标信息计算检测框间距,判定缺件缺陷并沿检测框剪裁成单芯片图像。

深度学习算法 YOLOv3(You Only Look Once)是一种一阶段算法,将检测过程看作回归问题求解,舍弃了显示候选框生成的步骤,对输入图像提取特征后可以迅速预测出目标所属类别和所在位置,检测速度快,能满足工业生产过程中对于缺陷实时检测的需求。YOLOv3 的骨干网络 Darknet-53 分别抽取到了下采样 32 倍、16 倍和 8 倍的特征,可以实现对不同大小尺度的目标检测,采用多尺度融合的方法进行局部特征交互,对主干网络提取到的 3 个不同尺度的特征图 Y_1、Y_2 和 Y_3 进行融合预测。不同尺度的特征图是把输入图像划分成不同数量的单元格,数量越多、越容易检测小目标物体。分析可知,52×52×255 特征图既拥有深层网络的语义抽象特征,又充分利用了浅层网络的细粒度像素级别的边缘、转角和结构信息的底层特征。YOLOv3 聚类了 9 个先验框,每个尺度的特征图分配到 3 个不同尺寸的先验框,见表 1。

表 1 YOLOv3 特征图信息

Tab. 1 Feature map information of YOLOv3

特征图层	特征图大小	边界框尺寸	边界框数量	感受野
图层 1	13×13	(116×90);(156×198);(373×326)	13×13×3	大
图层 2	26×26	(30×61);(62×45);(59×119)	26×26×3	中
图层 3	52×52	(10×13);(16×30);(33×23)	52×52×3	小

本文借鉴迁移学习思想,采用了在 ImageNet 上预训练好的模型参数 darknet53.conv.74,在此基础上继续训练。数据增强方法主要选取了旋转、平移、缩放及水平翻转,增加训练集样本数量,提高模型的泛化能力,避免出现过拟合现象。

模型检测目标芯片后,返回所属类别、置信度得分及边界框坐标信息。检测框坐标集合为 $b_i = \{[x_1,y_1,w_1,h_1],[x_2,y_2,w_2,h_2],\cdots\cdots,[x_i,y_i,w_i,h_i]\}$,其中 i 为检测出的芯片编号,(x_i,y_i) 表示第 i 个边界框的左上角坐标,(w_i,h_i) 表示第 i 个边界框的宽、高。将 b_i 按照 x_i 坐标连续递增排序并重新标号,可进一步定位芯片。本实验采用的芯片规格唯一,正常封装的物料条中芯片间距固定,而有缺件缺陷的物料条会出现一段较长的间距。计算两相邻边界框坐标的间距,即为相邻芯片间距,缺件缺陷芯片间距如图 2 所示,推导出的定义式可写为:

$$d_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad i = 1,2,\cdots,n-1 \tag{1}$$

其中,(x_i,y_i) 表示第 i 个边界框左上角的横、纵坐标,若 d_i 大于预设间距阈值 $d_{\max}$,则可判定第 i 块芯片与第 $i+1$ 块芯片中出现了缺件缺陷。

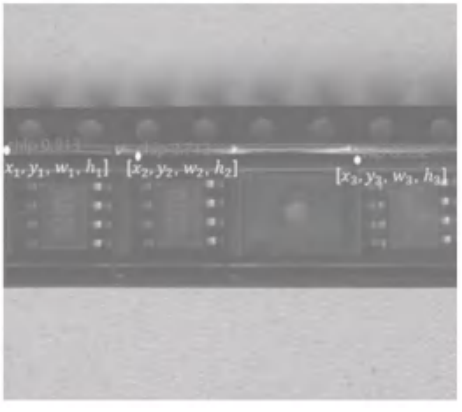


图 2 缺件缺陷芯片间距图

Fig. 2 Missing parts defect chip spacing diagram

2.3 缺陷检测模块

缺陷检测模块使用图像分割算法、即大津法(Otsu)分区域提取出引脚部分,将二值图中图像的 4 个边缘点使用最小二乘法拟合直线,进行芯片歪斜判定;研究又通过改进的 Harris-SIFT 算法模板匹配,根据匹配特征点对分布的区域及匹配点对占比进行缺陷分类判定,并对芯片反置、印刷模糊和塑封不严密缺陷进行识别,被判定为合格品或各类缺陷品。

2.3.1　图像预处理

研究对象示例为 SOIC(Small Outline IC)，即小外形 IC 封装，统一规格为 7 mm×6.5 mm。受传感器材料属性、拍摄光照角度等因素的影响，工业现场采集的芯片图像通常包含噪声，系统收到图像数据后先要进行预处理，对得到的元器件图像进行光源修正、图像去噪等。

采用直方图均衡化方式处理采集到的一些过暗和过亮的图像，丰富图像的细节，增强图像的质量，以减小由于光源反射而产生的明亮区域的影响，如图 3 所示。

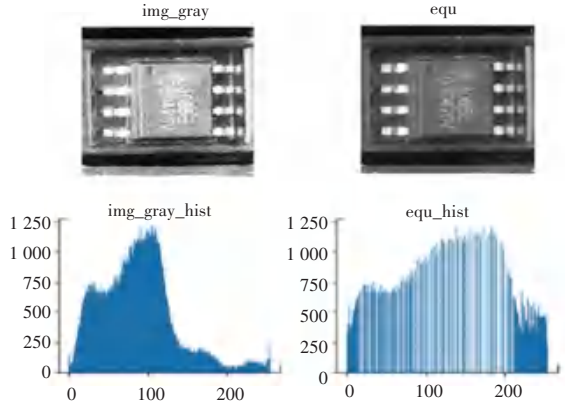


图 3　直方图均衡化处理

Fig. 3　Histogram equalization processing

2.3.2　图像分割方法

图像分割是对图像进一步分析、识别的前提，而阈值的选择则是该方法中的核心任务，动态阈值分割算法适用于检测目标前景与背景具有较好区分度的情况[8]。本文采用 Otsu 分割算法，操作简单且分割效果良好，类间方差定义如式(2)所示：

$$\sigma^2(T) = \omega_0(T) \times (\mu_0(T) - \mu_T)^2 + \omega_1(T) \times (\mu_1(T) - \mu_T)^2 \quad (2)$$

其中，$\omega_0(T)$ 表示阈值为 T 时目标前景像素点数占整幅图像的比例，前景平均灰度值为 $\mu_0(T)$；$\omega_1(T)$ 表示阈值为 T 时背景像素点数占整幅图像的比例，背景平均灰度值为 $\mu_1(T)$；μ_T 为整体图像的灰度均值。最佳的阈值 T 为类间方差 $\sigma^2(T)$ 达到最大值时对应的灰度值。

使用 Otsu 算法对图像进行动态阈值分割时，部分区域因光线等因素影响，呈现出与芯片引脚区域相近的颜色，这会影响图像动态阈值的计算结果，导致无法直接通过二值化处理得到完整的引脚区域。由于芯片光照不均匀，导致某些引脚区域与正常引脚颜色存在差异，因此就对引脚和芯片主体区域进行划分，对仅包含引脚的区域进行分析，以减小芯片主体对提取过程的影响。对引脚区域分块，每个分块内利用 Otsu 算法计算动态阈值，提取受光照因素影响的引脚。分区域后的 Otsu 算法的流程步骤详述如下。

算法 1　分区域的 Otsu 算法

输入：单芯片图像编号，单芯片三通道图像，单芯片在原物料条的坐标位置信息，单芯片图像高度，单芯片图像宽度，引脚数目，芯片主体区域与引脚区域所占宽度比例系数

输出：分割后的芯片引脚二值图像

1.将图像三通道图像转化成灰度图。

2.根据图像中引脚与芯片主体的比例位置关系，将灰度图像根据输入比例系数 k 横向划分 3 个区域，宽度像素范围 $[0:\frac{width}{k+2}]$ 为左引脚区域、$[\frac{width}{k+2}:\frac{width}{k+2}\times k]$ 为芯片主体、$[\frac{width}{k+2}\times k:width]$ 为右引脚区域，将左引脚区域和右引脚区域分别存储。

3.考虑到图像截取像素差异，将左引脚区域和右引脚区域分别横向均分 $n+1$ 等份，得到左、右引脚分块区域集合。

4.按序号逐次遍历各区域及各区域每个像素点，如果像素点位于图像边缘区域，则将该点像素值融合为该点在原图周围 3×3 区域内像素灰度平均值。

5.按序号逐次遍历各区域，分别统计各区域内的灰度直方图并归一化，求得各区域内的平均灰度值，测试多个阈值求出各区域最佳分割阈值，根据不同区域的最佳阈值进行图像二值化分割，更新左、右引脚分块区域集合。

6.将左、右引脚分块区域集合中各区域按序号重新拼接写入左、右引脚。

7.将左、右引脚区域拼接，生成分割后的芯片引脚二值图像并返回。

分割出来的引脚部分可能会由于光照等问题出现截断等现象，可将引脚图像中噪声进行形态学操作。闭运算可以去除前景噪声，并填充原有闭合图像中的空洞。对图像进行闭运算，即先使用膨胀操作，将邻近分散的引脚点连通，有效解决引脚截断的问题；对膨胀过的图像加以腐蚀处理，将芯片内部多余噪声部分腐蚀。闭运算可以去除前景噪声，并填充原有闭合图像中的空洞。闭运算操作效果如图 4

所示。

(a) 原图 (b) 膨胀图像 (c) 腐蚀图像

图 4 闭运算操作效果图

Fig. 4 Closing operation effect diagram

分割出引脚部分后,可以通过引脚的倾斜角度判断物料条是否出现芯片歪斜缺陷。以分割出的引脚图像 4 个边缘点为角点,使用最小二乘法分别拟合 4 条直线,组成一个四边形作为引脚图像边缘的最小外接矩,使用中心法确定图像的几何中心位置。假设图像左上角起始坐标为 $P(0,0)$, 右下角坐标为 $Q(m,n)$,可得中心点坐标(x_0,y_0) 的运算公式具体如下:

$$x_0 = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} x f(x,y)}{\sum_{i=0}^{m}\sum_{j=0}^{n} f(x,y)} \tag{3}$$

$$y_0 = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} y f(x,y)}{\sum_{i=0}^{m}\sum_{j=0}^{n} f(x,y)} \tag{4}$$

其中, m、n 分别为图像像素的行数和列数, $f(x,y)$ 为图像在点 (x,y) 处的灰度值。

角度参数计算如图 5 所示。以中心坐标点 P 为原点作延长线,与最小二乘法得出的拟合直线边界交点记为 $O(0,0)$,以 O 作为原点建立相对直角坐标系,记角点 Q 的坐标为(x,y),其中芯片偏斜角度 θ 可由式(5) 得出:

$$\theta = arctan\frac{x}{y} \tag{5}$$

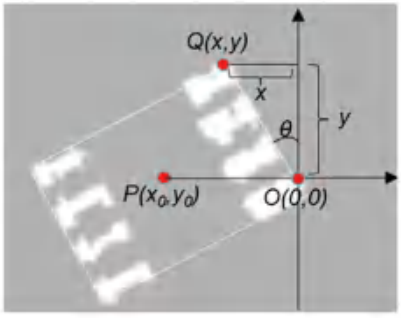


图 5 角度参数计算图

Fig. 5 Angle parameters calculation diagram

2.3.3 Harris-SIFT 图像匹配

SIFT 算法耗时主要集中于特征点检测与特征点描述部分,特征点检测时需要多次用到高斯模糊,运算量较大[9]。故本文提出使用 Harris 算子在大尺度空间检测角点后映射到小尺度空间,建立 SIFT 特征描述子;在匹配点对校正部分,采用角点集类别向量聚类算法进行图像的精确匹配,以剔除错误的匹配点。Harris-SIFT 图像匹配算法过程示意如图 6 所示。这里,对 Harris-SIFT 算法中各主要步骤拟展开阐释分述如下。

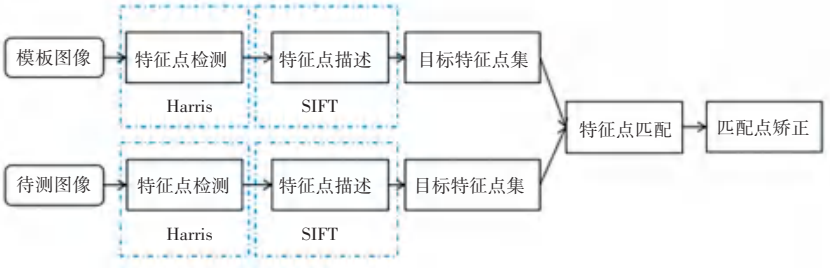


图 6 Harris-SIFT 算法过程示意图

Fig. 6 Harris-SIFT algorithm process diagram

(1)检测图像角点。Harris 算法通过计算点的一阶曲率及梯度检测角点,相比 SIFT 算法无需构建 DoG(Difference of Gaussian)金字塔,故检测速度更快。Harris 角点检测类通过建立小的窗口对对象进行扫描,其对应角点响应函数 R 的运算公式可写为:

$$R = det(M) - k\ (traceM)^2 \tag{6}$$

$$det(M) = \lambda_1 \lambda_2 \tag{7}$$

$$traceM = \lambda_1 + \lambda_2 \tag{8}$$

其中, $detM$ 为 M 的行列式; $traceM$ 为 M 的迹; λ_1,λ_2 是自相关矩阵的 2 个特征值; k 为常量,一般取值为 0.04~0.06。

(2)确定角点特征向量。特征点检测完成,生成角点集,使用 SIFT 算法为每一个角点分配可以反映角点特征的梯度幅值 $m(x,y)$ 和方向 $\theta(x,y)$,研究推得的数学公式见如下:

$$m(x,y)=\sqrt{[L(x+1,y)-L(x-1,y)]^2+[L(x,y+1)-L(x,y-1)]^2} \tag{9}$$

$$\theta(x,y)=\arctan\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)} \tag{10}$$

其中，L 表示特征点所在尺度空间，由高斯函数与原图像卷积得到。

(3)生成 SIFT 特征描述向量。为了确保特征点具有旋转不变性，以关键点主方向为中心旋转，求取 16 个 4×4 窗口内每个种子点的 8 个梯度方向，如此每个特征点就生成一个 128 维的 SIFT 特征向量，为了消除光照变化的干扰，对其做归一化处理。

(4)特征点匹配。使用 K-近邻算法（K 一般取值为 2）求取欧氏距离最近邻和次近邻。当该点的最近邻距离与次近邻距离的比值小于 T（本文取 T 为 0.8）时，被认为是匹配点。

在匹配的点对中，具有最大相似性的点对不一定是正确配对的，一般针对匹配后的点对使用随机采样一致性(RANSAC)算法计算一组局内点的单应矩阵，根据该单应矩阵计算映射误差，根据误差阈值重新迭代计算局内点集合，找到最优局内点集合后，局外点被判断为误配点。但 RANSAC 算法迭代次数没有上限，需要人为定义阈值，存在很大的随机性，无法达到最优，且当提取的特征点数量较多时，该算法的计算量会有显著增大，匹配效率较低。故使用 K-means 聚类算法修正匹配点的算法研发步骤详见如下。

算法 2　K-means 聚类修正匹配点对算法

输入：单芯片图像编号，单芯片三通道图像，单芯片在原物料条的坐标位置信息，单芯片图像宽高，初始匹配点对集合，聚类初始中心点个数

输出：修正后的匹配点对集合

1.For $k=1\rightarrow N$：

得到特征一：匹配点对之间的欧氏距离 d_k；

得到特征二：匹配点对之间连线与水平面夹角 θ_k；

将 d_k 和 θ_k 加入类别向量集合 F。

2.对类别向量集合 F 进行 K-means 聚类，设置聚类的个数为 $K=2$，生成 2 个聚类中心点，计算所有类别向量 F_k 到聚类中心点的距离；更新中心点，迭代聚类；当中心点变化满足收敛要求时停止迭代。

3.由于正确匹配点对集中，故点对数目最多的类别为正确匹配点对，将另一匹配有误的类别中的点对从初始匹配点对集合 C 中剔除，得到修正后的匹配点对集合 P。

3　实验结果及分析

3.1　实验环境与数据准备

本实验硬件环境为：Intel（R）Core（TM）i5-2450 M CPU@2.50 GHz 的处理器，8 GB 内存，4 GB 显存，操作系统 Windows10 64 位。开发环境为：Python3.8+TensorFlow2.4.0、OpenCV4.4 图像处理库和 YOLOv3 目标检测框架，使用 CUDA10.1 和 CUDNN10.1 加速运算。

深度学习算法依赖于图像标注，故前期的主要工作是对包含不同数量黑色芯片块的 PCB 板进行人工标注和训练。分别从网络和工厂车间获取到 1 000张 PCB 板样本图像，随机对样本图像进行 90°、180°和 270°旋转，数据增强后获得 3 000 张训练图像，平均每张图像中有 7 个芯片样本。被标记好的样本图像会生成对应的.xml 文件，该文件主要存放数据集标签信息。训练集、测试集和验证集按 3∶1∶1 划分，使用 YOLOv3 迁移学习进行芯片的目标检测算法训练，训练后的图像简化了芯片复杂的背景信息，以便于对芯片封装质量进行下一步的分析。

在实验室环境下进行仿真实验。将带有缺陷的物料条固定放置于传送带上，传送至工业相机光源下，相机以 40 fps 帧率录制视频并进行质量分析。分别选择正常芯片、芯片缺失、塑封不严密、芯片歪斜、芯片反置以及芯片正面印刷的 logo 错漏缺陷图像各 200 张进行实验验证。缺陷认定取芯片偏斜角度 θ 阈值为 10°，即倾斜角度超过 10°被认为存在芯片歪斜缺陷；印刷错漏和芯片反置的匹配点对区间分别为 15%~90%和 0%~15%，即匹配点对占比超过 90%，则认为不存在印刷错漏或芯片反置缺陷，匹配点对占比介于 10%~90%、被认为存在印刷错漏缺陷，匹配点对占比低于 10%，则认为存在芯片反置缺陷。

3.2　评价指标

采用的评价标准有查准率 Pre_i、查全率 Rec_i、$F_1-Score$ 以及虚警率 FA_i。各计算公式可分别写作如下形式：

$$Pre_i=\frac{TP_i}{TP_i+FP_i} \tag{11}$$

$$Rec_i=\frac{TP_i}{TP_i+FN_i} \tag{12}$$

$$F_{1i} = \frac{2 \times Pre_i \times Rec_i}{Pre_i + Rec_i} \tag{13}$$

$$FA_i = 1 - Pre_i = \frac{FP_i}{TP_i + FP_i} \tag{14}$$

其中，TP_i(*True Positive*) 表示 i 类样本被正确识别的样例；FP_i(*False Positive*) 表示其它类样本被识别为 i 类样本的样例；FN_i(*False Negative*) 表示 i 类样本被识别为其他样本的样例。

3.3 实验结果及分析

3.3.1 YOLOv3 目标检测结果分析

利用 YOLOv3 网络在预训练的模型基础上进行迭代训练，网络文件中的部分参数设置见表 2，训练期间损失曲线如图 7 所示。可以看出，模型在前 5 个周期迭代中损失值迅速下降，此时模型正在快速拟合；经过 60 个周期的迭代训练后，该模型的损失值降低至 0.024、并趋于稳定，不再继续收敛，此时网络模型已达到最优状态，可以使用训练好的模型检测数据集中的芯片。

表 2 网络训练参数

Tab. 2 Network training parameters

参数名称	设定值
批处理数量(*Batch Size*)	16
初始学习率(*Origin Learning Rate*)	0.001
迭代周期(*Epoch*)	60
输出图像尺寸(*Input Size*)	416 × 416 × 3
IoU 阈值(*IoU Threshold*)	0.5
迭代轮数(*Step*)	22 000

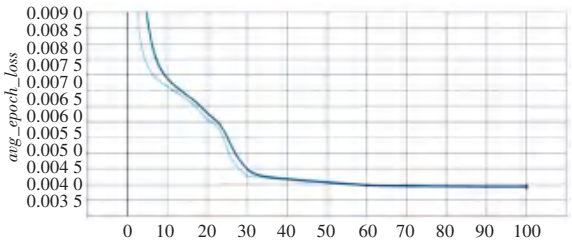


图 7 训练期间的损失曲线

Fig. 7 Loss curve during training

3.3.2 Harris-SIFT 算法检测结果可视化

本文提出的 Harris-SIFT 算法与传统 SIFT 算法特征点检测时间对比结果如图 8 所示，本文算法比 SIFT 算法检测时间同比缩短 2~3 倍，有效提高检测速度。

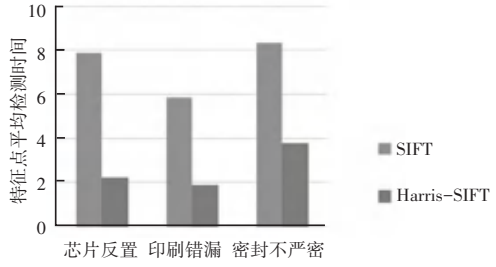


图 8 算法检测时间对比图

Fig. 8 Algorithm detection time comparison chart

使用改进后的 Harris-SIFT 算法对缺陷图像进行检测，结果如图 9~图 11 所示。

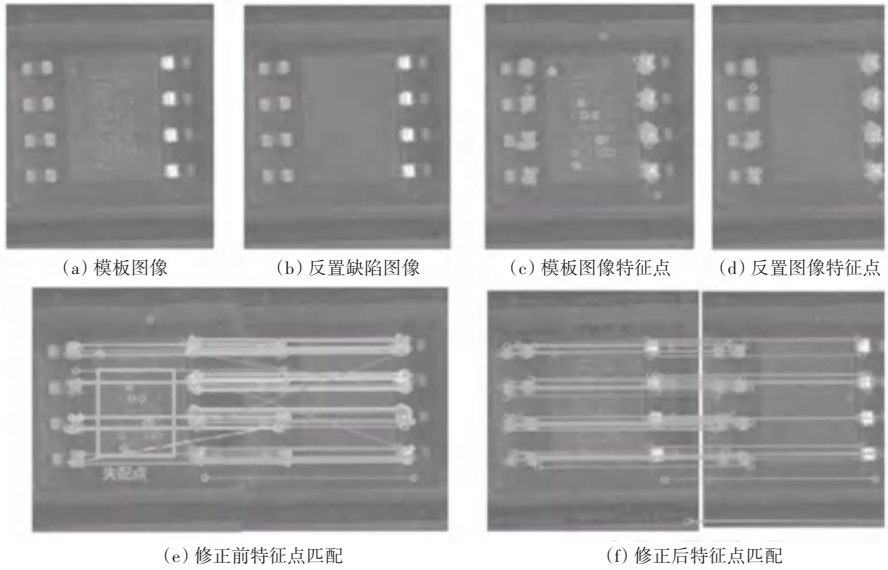
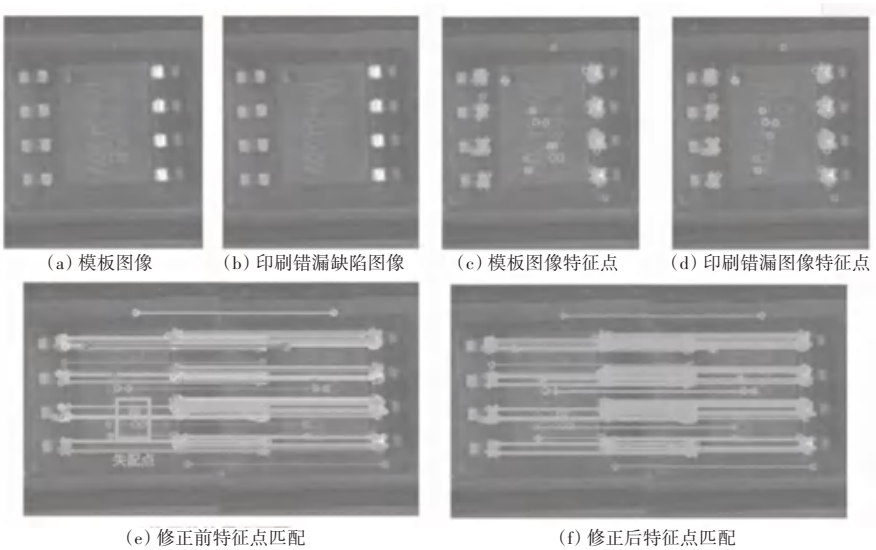


图 9 反置芯片图像

Fig. 9 Invert chip images

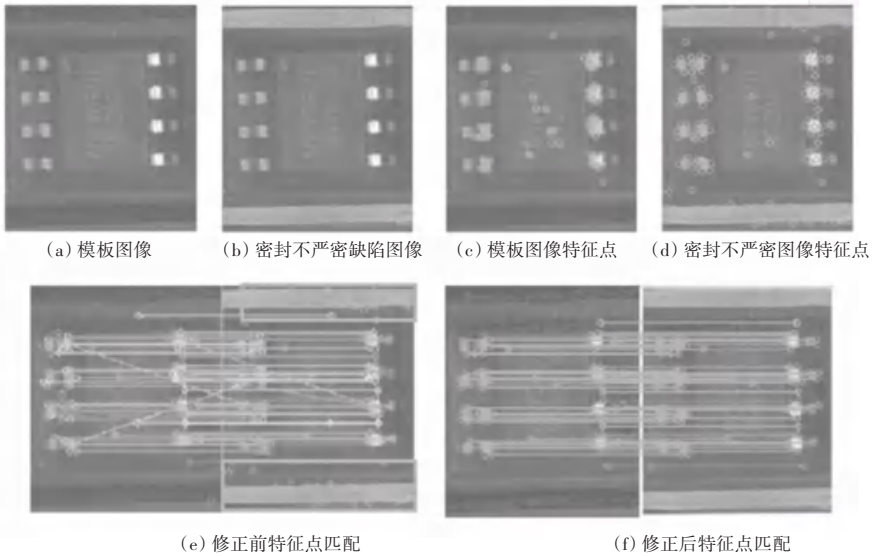


(a) 模板图像　(b) 印刷错漏缺陷图像　(c) 模板图像特征点　(d) 印刷错漏图像特征点

(e) 修正前特征点匹配　(f) 修正后特征点匹配

图 10　印刷错漏图像

Fig. 10　Misprinted images



(a) 模板图像　(b) 密封不严密缺陷图像　(c) 模板图像特征点　(d) 密封不严密图像特征点

(e) 修正前特征点匹配　(f) 修正后特征点匹配

图 11　塑封不严密图像

Fig. 11　Improper plastic packaging

3.3.3　缺陷检测结果分析

使用本文提出方法对各类图像进行缺陷识别和分类，缺陷检测结果见表 3。由表 3 可以看出，本文提出的方法针对各类缺陷的查全率和查准率都较高，且虚警率均在 11%以下，满足工业生产需求，可以应用到生产芯片封装检测领域。印刷错漏缺陷样本与芯片反置缺陷样本最易出现混淆，少量密封不严密样本与歪斜样本会相互误判，但整体的检测结果仍保持在标准范围内，可对 SIFT 算法做进一步调优，针对特征点对匹配阈值范围进行调节。

表 3　缺陷检测结果

Tab. 3　Defects detection results

图像类别	实验图像/张	正确图像/张	误检图像/张	*Pre* /%	*Rec* /%	F_1 /%	*FA* /%
正常芯片	200	199	1	95.12	99.50	97.26	4.88
芯片缺件	200	195	5	94.20	97.50	95.82	5.80
芯片歪斜	200	185	15	93.91	92.50	93.20	6.09
印刷错漏	200	176	24	89.34	88.00	88.66	10.66
塑封不严	200	181	19	96.79	90.50	93.54	3.21
芯片反置	200	180	20	82.95	90.00	86.33	10.00
Average Value				92.05	93.00	92.47	6.77

4 结束语

针对现有的芯片封装质量检测过程中存在的问题,本文将深度学习算法与图像处理技术相结合,提出了一种基于视频图像的芯片封装质量分析方法。实验结果表明,本方法准确地判断出了芯片封装的残次品并指出了缺陷对应的类型,以便及时回收残次品并重新封装。该缺陷分类方法取得了 93% 的准确率,说明提出模型同时具备良好的鲁棒性与泛化能力,但由于复杂工业条件下影响检测准确率、召回率的因素繁杂,该算法还有继续优化的空间,进一步提高芯片封装质量分析的准确率和检测效率是未来研究的主要方向。

参考文献

[1] 李正明,黎宏,孙俊. 基于数字图像处理的印刷电路板缺陷检测[J]. 仪表技术与传感器,2012(08):87-89.

[2] 王栋,解则晓. 基于形态学的 PCB 缺陷快速检测技术[J]. 计算机科学,2016,43(S1):184-186,225.

[3] 李云峰,李晟阳. 基于梯度方向信息熵的印刷电路板缺陷检测[J]. 中国机械工程,2017,28(06):695-701.

[4] 黄风荣,李杨,郭兰申,等. 基于 Faster R-CNN 的零件表面缺陷检测算法[J]. 计算机辅助设计与图形学学报,2020,32(06):883-893.

[5] 郑昊. 基于改进 SIFT 算法的图像匹配研究[D]. 淮南:安徽理工大学,2020.

[6] HARRIS C, STEPHENS M. A combined corner and edge detector[C]//Proceedings of the 4th Alvey Vision Conference. United Kingdom: BMVA, 1988: 147-151.

[7] LOWE D G. Distinctive image features from scale-invariant keypoints[J]. International Journal of Computer Vision, 2004, 60(2): 91-110.

[8] WU H, WANG B, GUAN D, et al. A SIFT-based image retrieval system[J]. International Summerschool on Computer Science, Computer Engineering and Education Technology, 2018, 19.

[9] KHAN S A, GULZAR Y, TURAEV S, et al. A modified HSIFT descriptor for medical image classification of anatomy objects[J]. Symmetry, 2021, 13(11): 1987.

(上接第 75 页)

[5] HECHT G, BENOMAR O, ROUVOY R, et al. Tracking the software quality of android applications along their evolution (t)[C]// IEEE/ACM International Conference on Automated Software Engineering (ASE). Lincoln, NE, USA: IEEE, 2015: 236-247.

[6] HECHT G, MOHA N, ROUVOY R. An empirical study of the performance impacts of android code smells[C]//Proceedings of the International Conference on Mobile Software Engineering and Systems. Austin, TX, USA: IEEE, 2016: 59-69.

[7] PALOMBA F, DI NUCCI D, PANICHELLA A, et al. Lightweight detection of android-specific code smells: The adoctor project[C]//IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). Klagenfurt, Austria: IEEE, 2017: 487-491.

[8] HABCHI S, HECHT G, ROUVOY R, et al. Code smells in ios apps: How do they compare to android? [C]//IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft). Buenos Aires Argentina: IEEE, 2017: 110-121.

[9] CARETTE A, YOUNES M A A, HECHT G, et al. Investigating the energy impact of android smells[C]//IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). Klagenfurt, Austria: IEEE, 2017: 115-126.

[10] GRANO G, DI SORBO A, MERCALDO F, et al. Android apps and user feedback: a dataset for software evolution and quality improvement[C]//2nd ACM SIGSOFT International Workshop on App Market Analytics. Paderborn, Germany: ACM, 2017: 8-11.

[11] LIM D. Detecting code smells in Android applications[D]. Netherlands: Delft University of Technology, 2018.

[12] MATEUS B G, MATIAS M. An empirical study on quality of Android applications written in Kotlin language[J]. Empirical Software Engineering, 2019, 24(6): 3356-3393.

[13] RAHKEMA K, PFAHL D. Comparison of code smells in iOS and Android applications[C]//QuASoQ@ APSEC. 2020: 79-86.

[14] GONG A, ZHONG Y, ZOU W, et al. Incorporating Android code smells into Java static code metrics for security risk prediction of Android applications[C]//IEEE International Conference on Software Quality, Reliability and Security (QRS). IEEE, 2020: 30-40.

[15] RASOOL G, ALI A. Recovering android bad smells from android applications[J]. Arabian Journal for Science and Engineering, 2020, 45(4): 3289-3315.

[16] HAMDI O, OUNI A, ALOMAR E A, et al. An Empirical Study on Code Smells Co-occurrences in Android Applications[C]//IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW). Australia: IEEE, 2021: 26-33.