学校代码：10564　　学　　号：2014201003
分 类 号：TP391.41　　密　　级：

华南农业大学

硕 士 学 位 论 文

基于计算机视觉技术监测橘小实蝇

徐　　培

指　导　教　师：　李震　副教授

学　院　名　称：　电子工程学院

专　业　名　称：　农业电气化与自动化

答辩委员会主席：　洪添胜　教授

中国·广州

2017 年 6 月

摘　要

随着城市化的进展，农村的青壮年不断的前往城市务工，农村劳动力不足的弊端日益凸显。如今农村的剩余人口已经不足以支撑起传统农业所需要的繁重劳动。针对这一现象，如何充分利用先进的自动化技术以及计算机视觉技术以减轻农民的负担成为了关系到国计民生的大事。在农情监测以及农产品质量检测领域之中，运用较为广泛的技术为计算机视觉技术。最初，计算机视觉技术主要是为了研究农产品的品质及其分级。但由于当时条件限制，无法做到实时检测，因此当时计算机视觉技术还是处于理论研究的阶段，随着计算机性能的不断提高，计算机视觉在农业领域的运用开始变得更加频繁。在农村劳动人口远远不足以支撑传统农情识别的情况下，更加节省时间、节省人力而且更加精准的计算机视觉技术也在农业领域运用得更加广泛。

本论文将对以下内容进行了具体的研究分析：

（1）研究基于 SIFT 算法的实蝇种类区分方法，主要针对 SIFT 能否将橘小实蝇害虫从三种实蝇害虫中区分出来。使用 SIFT 算法对所选的实蝇害虫样本图片进行处理后，能够较好的将橘小实蝇样本图片从其他实蝇害虫的样本图片中区分出来。

（2）利用果蝇形态特征上的特定关节特征，通过区域似圆度和大小来识别特定关节，以此来区别果蝇和其他昆虫，实现模板配准。

（3）三种实蝇身体上的条纹带与盾片区的比值有较大的差异性，可以作为对三种实蝇害虫进行区分的依据，本实验在 CCS 平台上研发了一套针对实蝇盾片区以及条纹带提取的算法，该算法通过色彩空间转换、直方图均衡化、图像腐蚀等方法完成了对盾片区的提取，通过连通区域标记算法、对盾片区重心进行提取以及纵带条纹带区域搜索算法对纵带条纹带进行了提取，将提取出的条纹带像素与盾片区像素进行比值处理，其中比值为 0 的为橘小实蝇，比值落在[0.0856,0.3770)之间的为瓜实蝇，比值落在 0.3770 以上的为南瓜实蝇。经测试，该算法对三种实蝇害虫有较好的区分率。

（4）果园害虫监测节点的实物设计：主要在平台上导入了实蝇跟踪算法以及实蝇计数算法，在实蝇跟踪与实蝇计数的算法选择方面主要考虑了以下几种算法：

①阈值分割算法：通过实验的分析比较，可以得到双峰法的效果是最好的，所以在果蝇检测中采用的是双峰法的阈值分割。

②前景检测算法：通过实验的分析比较，可以得到具有背景更新的背景差分法的检测算法的效果是最优的，所以在果蝇检测中采用的是背景差分法。

③新团块检测算法：采用的是通过距离的大小来作为新旧团块的判断依据，同时通过轮廓检测来检测团块。

④目标区域分割算法：通过实验对比基于灰度信息的图像分割算法及基于 YUV 通道的区域生长分割算法，基于 YUV 通道进行图像分割，图像信息丢失比较少，能获得比较好的效果。因此，目标识别前，采用基于 YUV 通道进行图像分割提取目标图像区域。

关键词：计算机视觉;橘小实蝇识别;橘小实蝇计数

Monitoring of Citrus Fruit Fly base on Computer Vision Technology

Xu Pei
(College of Electronic Engineering,South China Agricultural University，
Guangzhou 510642，China)

Abstract: As rural labor to urban migration of the population in young adults, the disadvantages of insufficient rural labor force increasingly highlighted. Now the rural surplus population has enough to prop up the traditional agriculture need heavy labor. In reaction to the phenomenon, how to make full use of advanced automation technology and computer vision technology in order to reduce the burden of farmers become the relationship to the national economy and people's livelihood. The computer vision technology is mainly used in Agricultural situation monitoring and agricultural product quality, higher than the experts of artificial prediction precision.In the rural labor force is far not enough to support the traditional Agricultural situation recognition, more saving time, person and more precise machine vision technology used more widely in the field of agriculture.

This article mainly studied in the following aspects:

(1)Verify the SIFT algorithm in the fruit fly insect state performance analysis, th is study mainly to a small fruit fly insect pest could SIFT orange distinguish from t hree kinds of fruit fly pests, using SIFT algorithm to distinguish small orange fruit fl y pests accuracy, can better the orange small fruit fly distinguish from other fruit fly pests.

(2)Flies morphological characteristics of template matching: this project by use o f the morphological characteristics of a particular joint characteristics of fruit flies, th rough regional roundness and size to identify specific joints, to distinguish between fr uit flies and other insects and template matching.

(3)Found in experiments, three kinds of fruit fly with the stripes on the body an d the ratio of shield area have bigger difference, can be as the basis of to distinguis h three types of fruit fly insect pest, this experiment on CCS platform has developed a set of fruit fly shield area and stripe extraction algorithm, this algorithm through t he color space transformation, histogram equalization, image corrosion method comple

ted the extraction of dong area, connected component labeling algorithm, to shield ar ea through the center of gravity is extracted and the vertical striped area search algor ithm to extract, longitudinally striped belt will extract the stripe with pixels and shiel d for the pixel ratio, the ratio of 0 to orange small fruit fly, the ratio falls in the ra -nge of [0.0856,0.3770) for melon fly, ratio above 0.3770 for pumpkin fruit fly, teste d the algorithm to the distinction between the three kinds of fruit fly pests had a go od recognize rate.

(4)The physical design of the orchard pest monitoring node: the main introductio n of the real fly tracking algorithm on the platform, as well as the actual number of flies in the algorithm, in the real fly tracking and counting algorithm to consider th e main aspects of the following algorithms:

Threshold segmentation algorithm is: through the analysis of experimental compa rison, can get the effect of two-peak method is the best, so this project in fruit flies detection is adopted in the two-peak method threshold segmentation method.

Foreground detection algorithm: through the analysis of experimental comparison, can get background updating background difference method, the effect of the detecti on algorithm is optimal, so the project is in the fruit fly detection and background difference.

A new mass detection algorithm: this project USES is through the distance as th e size of the old and new judgment of briquette, at the same time through the conto ur detection to detect briquette.

Target area segmentation algorithm: by comparison with experimental image seg mentation algorithm based on gray level information and region growing image segm entation algorithm based on YUV channel, based on the YUV channel image segmen tation, image information loss is less, can obtain good effect.Before the target recogni tion, therefore, this project adopts the image segmentation based on YUV channel is to extract the target image area.

Key words: Computer vision; Bactrocera dorsalis recognition; Bactrocera dorsalis count

目　　录

1 前言

1.1 害虫评估及进行害虫识别的意义

农业自古以来就是我国的基础支撑性产业，是国民经济的基础，是人类的主要食物来源，生存之本（王辉，2013）。农业的发展与社会的稳定与发展有着密不可分的联系。

在农作物生长期间，害虫的爆发是非常普遍的，而害虫的侵袭不可避免的会带来农作物的减产，这对于我国农业发展非常不利。针对我国近几年害虫高发的情况，应采取必要的害虫评估措施来避免害虫对我国农业发展的不利影响。

但在我国，导致农作物患病的源头有多种，包括微生物，以及不同种类的昆虫（钟乃扣，2014）。而针对不同种类的昆虫，采取的防治措施也大不相同。因此，仅仅对害虫爆发状况进行评估是不够的，还需在害虫评估的基础上进行害虫体态的分析，有效的虫态分析不仅能大大减少农药的使用，并且对于特定的害虫可以做到“专虫专治”，大大增强用药效率，确保农作物的健康生长。

1.2 传统的果园害虫防治技术

在农业发展过程中，我国农民对于伴随着作物生长的病虫灾害发生和防治，都有着自己独到的理论和方法。这些方法在中国几千年的传统农业史上不断地被后来者所完善，与此同时也推动了中国传统农业的发展。

传统农业应对害虫防范的主要方法是靠经验丰富的农民以及在昆虫领域进行过深入研究的学者对田间害虫暴发情况进行经验性的评估。他们能够凭借丰富的经验，对田间可能爆发的害虫种类进行识别，并采取相应的害虫防治方法来防止农民因害虫侵扰而承受过大的损失。

因此我们可以发现：对害虫进行实时、准确的评估，在害虫防治领域是非常重要的。但传统的害虫预测和预报往往凭借农民自身的经验，这对于农民有关害虫识别方面的素质要求较高，且完成评估人工作业强度大、时间跨度长，效率较低、预测结果往往存在较大的误判率，使得农民无法实时、准确获取田间农情，进而造成作物减产（叶智杰等，2011）。

1.3 计算机视觉技术在害虫监测中的应用

计算机视觉技术即运用机器的光学装置和非接触式传感器捕获图像目标后传输给图像处理设备，再由图像处理设备图像目标进行算法分析，提取出所需的特征信息后根据提取到的结果进行对应操作。简单的来说就是以机器来代替人眼在社会生产中的功能。

20 世纪 50 年代开始研究的二维图像统计模式识别开启了计算机视觉领域的大门（秦亚航等，2016）。20 世纪 60 年代中期，计算机视觉被运用到工业领域，解决了实际检测、测量以及控制等问题。而计算机视觉在农情领域的运用则在 20 世纪 70 年代末，当时主要是为了研究农产品的品质及其分级。但由于当时计算机的运算能力有限，无法做到实时检测，因此当时的计算机视觉主要处于理论研究阶段。

随着科技的不断发展，计算机的运算处理能力也在不断的加强。过去由于造价昂贵而仅仅运用于工业领域的硬件设施价格也随着科技的发展而逐渐变得亲民了起来。种种原因使得计算机视觉技术在害虫防治领域进行运用变成了可能。随着农村青壮劳动力涌向城市，更加节省时间、节省人力而且更加精准的计算机视觉技术业对促进农业发展取得了巨大的成就（丁兆庆，2007）。

1.3.1 国外研究进展

国外有关计算机视觉的起步较早，国外学者对计算机视觉在害虫识别领域进行了许多深入的研究，并取得了许多成果（Koumpouros et al., 2004；Chen et al., 2012；Shah et al., 2014；Cáceres Flórez et al., 2015； Ding et al., 2016）。

Martineau 等（2017）为了对昆虫的种类以及地理分布情况进行分析，并希望借昆虫分布情况分析人类活动情况对生物的影响。对目前流行的多种昆虫识别技术进行了分析。混合特征提取的图像处理技术对昆虫捕捉器中捕捉到的昆虫进行了特征提取。将特征分为地方特征、通用特征、以及本地特征三种特征。并应用机器学习技术对所提取的特征进行训练，以完成对昆虫的识别。

Xie.Chengjun 等（2015）使用 multipletask 方法、稀疏表示以及多核学习(MKL)技术对昆虫识别系统进行开发。在试验过程中发现昆虫的不同特征会对昆虫图像识别造成不同的影响。他们采用直方图对昆虫图像的原始特征进行量化。针对 24 种日常生活中的田间主要为害的害虫进行实验分析，实验结果表明他们所提出的方法在对昆虫进行分类的方面表现良好。

Ding.Weiguang 等（2016）提出一个基于深度学习识别的自动检测管道用来对捕虫器中所捕捉到的昆虫图像进行识别和计数。该研究主要针对苹果小卷蛾族群进行定性及定量的分析。与以往研究相比，该研究未采用传统的害虫区分工程来使其研究可以适用于其他类型的所有昆虫，来减少算法的人为操作。该算法可以运行于并行的硬件条件下，因此该算法可适用于对系统实时性较高的设备上。

Liu.Tao 等（2016）为了避免麦田蚜虫爆发从而导致的重大经济损失，研发了一种算法，该算法使用一个最大限度稳定的极值区域对包含蚜虫的背景图片进行区域描述。对目标图像进行直方图均衡化提取其梯度信息后用支持向量机来对蚜虫模型进行建模，用该算法与另外五个常用的蚜虫识别方法同时对作物上蚜虫的密度、颜色以及位置进行识别。实验结果表明该算法对蚜虫的识别率为 86.81%。优于传统的蚜虫识别方法。该方法能高效准确地对蚜虫的害虫密度进行评估。随后对与粉虱大小和形状不匹配的疑似点消除。其对粉虱图像的识别成功率可达 88.6%

Li.Yang 等（2015）对叶子表面的小型害虫（粉虱）进行多重分形分析，多重分型分析采用从本地图片对粉虱图像进行分割以及图像特征的区域最小选择策略。在多重分型维数中，疑似粉虱的斑点将首先从叶片图像中区分出来。采取区域最小选择策略对疑似粉虱的斑点区域进行特征进行提取，将其与阈值进行比较。

1.3.2 国内研究进展

国内有关计算机视觉在害虫识别领域的应用较晚，但通过不断的发展，我国在运用计算机视觉对害虫进行监测这一方向也取得了较多的成果（陈月华，2007；王东，2011；邱道尹等，2014；荆晓冉，2014；钟维，2015）。

王映龙等（2007）运用图像处理技术以及神经网络技术对采得的水稻样本进行处理。通过图像特征提取等操作后，对所得的特征进行训练，通过对比害虫目标样本特征以及图像经训练后所得的特征值，来对水稻害虫进行识别。

张建华等（2011）对棉蚜、棉叶螨、棉盲蝽、斜纹夜蛾以及烟粉虱等主要棉花害虫为害后的叶片表面信息进行计算机视觉分析。对棉花叶中受害虫侵袭的虫洞进行计算，并根据不同害虫为害过的叶片卷曲度不同这一物理特征，提取出 7 种特征信息。并通过支持向量机（SVM）完成对棉花害虫的具体识别。该方法对棉花害虫的识别率可达 80%以上。

李小林等（2016）为对蛾类农业害虫进行区分识别，提出了一种基于蛾类昆虫的纹理特征蛾算法，先用局部二进制对蛾类昆虫图像特征值。以此为基础，用 KNN 算法完成对蛾类昆虫的识别。该算法具有所需存储空间小，计算复杂度低等优点。其对蛾类昆虫的识别率为 96.4%

刁智华等（2013）通过对棉花害螨图像进行研究分析，提出一种算法，该算法能将复杂背景条件下的害虫病斑分割出来。该算法先对颜色相同的病斑和茎秆的彩色图像进行提取。在经过图像预处理操作后，利用面积阈值法将害螨病斑分割出来。该算法对害虫病斑提取的准确率可达 97.83%。

李震等（2014）为对果园现场柑橘全爪螨爆发情况进行快速、准确和无损地检测，研制了基于光学测量技术的柑橘全爪螨害虫快速检测仪，该装置能够测量果树冠层叶片对于红光以及近红外光的反射表现，并以此实验结果评估出 3 种级别不同的柑橘全爪螨害虫爆发情况。该实验通过实验确定并选取了恰当中心波长的发光二极管阵列作为实验中所采用的光源，大大减少了自然光对结果的影响，通过标准白板试验确定检测系数 k 等于 2.622。克服了树叶密度对检测结果影响、并且大大减弱了自然环境光线对检测结果的影响，但该实验未能客服气温对检测结果产生的影响；该实验能较为有效的对无以及轻度的害虫爆发情况进行识别；测试结果显示与柑橘全爪螨在一定面积的果树冠层叶片所产卵数有较高相关性。

1.4 本研究的目的及内容

1.4.1 研究目的

我国是世界第一大水果生产国和世界第一大水果消费国，2013 年我国水果总产量为 $25093.04*10^7$ 千克，约占全球产量的 14%。（黎文龙，2008）2013 年，全国柑橘栽培面积 $242.2*10^8$ 平方米，总产量 $3320.94*10^7$ 千克，我国不论是柑橘栽培面积还是总产量均居世界首位（沈兆敏，2015）,栽培面积占全球的 30%左右，产量占全球的 26%左右。在柑橘生长期间，由于柑橘作物本身生长地区气候情况复杂，加之柑橘物候期长，害虫种类多、监测与防治困难，害虫严重（李震等，2014），我国南方柑橘果园中主要活跃着橘小实蝇、瓜实蝇以及南瓜实蝇三种果蝇，其中橘小实蝇对柑橘树产生的危害最为严重.为了减少果农们的损失，能否及时的反馈柑橘园中果蝇害虫爆发情况显得尤为重要。

本研究对我国华南地区柑橘果园中主要活跃着的橘小实蝇、瓜实蝇以及南瓜实蝇三种果蝇进行研究。

对于柑橘园橘小实蝇害虫，传统的化学或物理机械防治手段虽然可以起到一定的效果，但随之而来的是环境的恶化与污染的加剧，进而危害到消费者的健康。所以，要实现对柑橘园橘小实蝇害虫的有效检测，应该在人与自然交流经验的基础上进行改进与创新。

因此柑橘园橘小实蝇害虫的现场自动监测方法与技术具有重大的研究价值，该技术要求对大片区域的害虫爆发动态进行检测分析，确定害虫爆发情况与气候变化、防治方式的关系。在防治方式选择、果园经营管理上有重要的理论和实践价值。根据对柑橘小实蝇形态特征的了解，可以采用计算机视觉技术进行检测。目前计算机视觉技术逐渐被广泛应用于果园害虫检测当中，并取得了一定的成果，但限于现有监测设备的不足，工作人员需要进行现场检测或现场取样发送实验室进行研究分析，受到人力工作强度及工作时间的限制，无法对果园实施大面积、大范围的监测，也无法保证监测结果的实时性，影响对果园害虫的判断，无法达到农业生产自动化的要求。

为了实现现场自动监测、节省果园害虫监测中人力物力的消耗，促进现代化农业生产技术的发展。本研究将主要对计算机视觉在实蝇害虫评估以及实蝇种类区分这两个方面进行研究。

1.4.2 研究内容及技术路线

结合研究生期间的所掌握的知识以及导师所布置的项目设计目的和设计内容，本文将进行以下的研究工作：

（1）基于 SIFT 的图像匹配：本项目通过对果蝇使用 SIFT 算法匹配，验证了 SIFT 算法能够将橘小实蝇害虫从三种不同的实蝇害虫中区分。

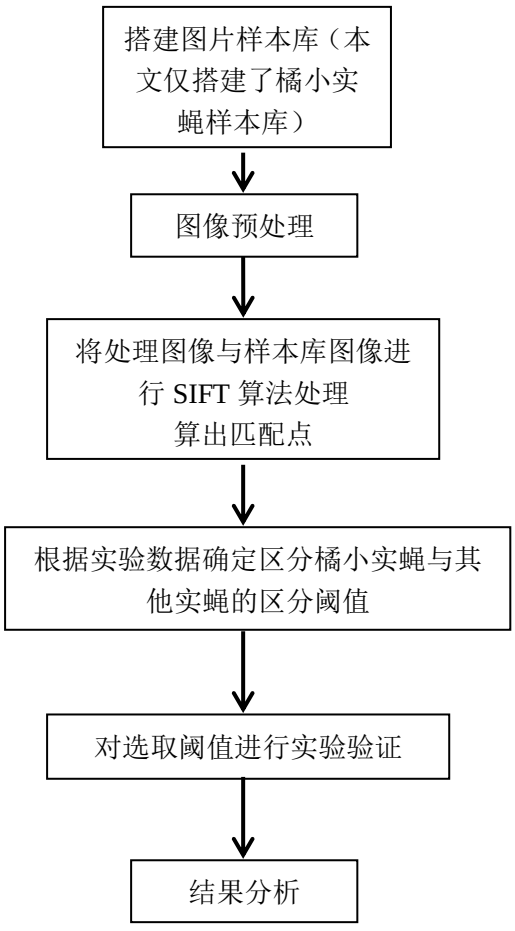
（2）果蝇形态特征的模板配准：本项目利用果蝇形态特征上的特定关节特征，通过区域似圆度和大小来识别特定关节来区别果蝇和其他昆虫，实现模板配准。

（3）本研究在 CCS 代码调试器的开发环境下，通过对三种不同种类实蝇的黄色纵带纹区域面积与腰腹部盾片区面积的比值进行计算，并根据三种实蝇样本黄色纵带纹区域面积与腰腹部盾片区面积的比值的不同来确定区分各种实蝇的阈值范围，从而达到了区分三种不同实蝇的目的。

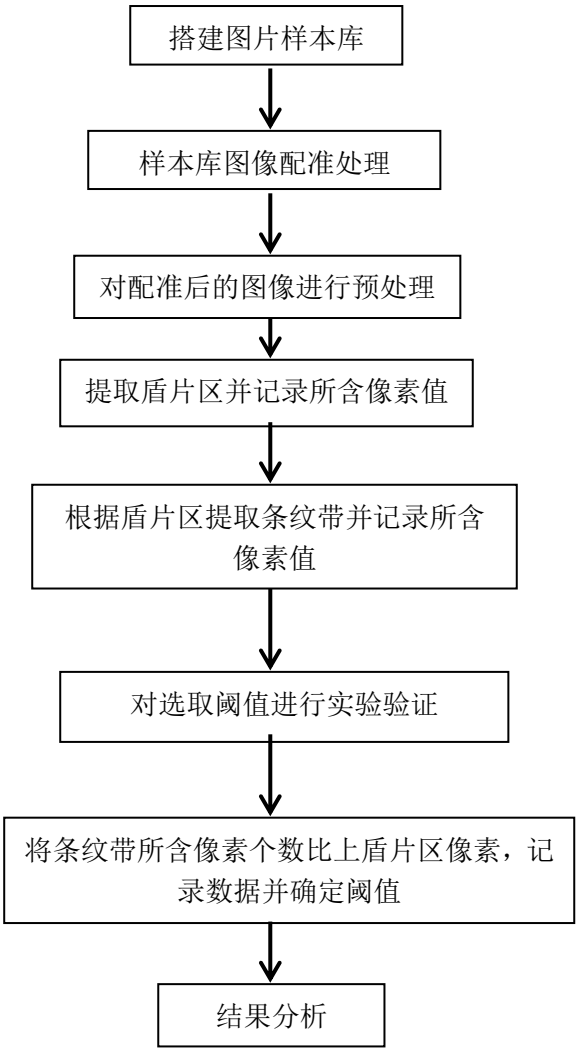
（4）搭建野外害虫观测节点，在该观测节点内写入了以下几个算法：阈值分割算法、前景检测算法、新团块检测算法、目标区域分割算法，以上述几个算法作为节点中对捕虫器中捕捉到的实蝇样本进行识别计数的算法。对捕虫器所捕获的实蝇进行计数跟踪，并以此来评估该片果园地区所遭受的实蝇害虫程度。

本文研究主要分为两个部分：第一部分针对实蝇虫态的算法研究，本文主要研究的是对 SIFT 算法针对橘小实蝇虫态的识别效率进行验证，以及 CCS 平台上根据实蝇盾片区与条纹带的比值的不同对实蝇虫态进行区分的算法（由于实蝇样本姿态各异，可能对盾片区以及条纹带的提取带来干扰。所以运行该算法之前，需先对实蝇样本库进行图像配准处理）。第二部分为实物实验部分，本文主要将实蝇害虫跟踪计数算法烧入了野外搭建的实蝇害虫监测节点中，对节点范围内实蝇害虫的爆发情况进行计数评估。

图 1.1 为本文的实蝇虫态的算法研究路线，其中图 1.1a 为 SIFT 算法的研究路线，图 1.1b 为 CCS 代码调试器上算法的研究路线。



a.SIFT 算法研究路线



b.CCS 代码调试器上算法的研究路线

图 1.1 实蝇虫态的算法研究路线

图 1.2 为本文实物部分研究路线,主要搭建了实蝇害虫监测平台，并在监测平台上烧入了实蝇害虫计数算法，通过该算法对当天害虫监测平台中捕虫器所捕捉到的实蝇害虫个体数的峰值进行计数。该计数结果可以作为评估该片果园地区实蝇害虫爆发情况的依据。

1.5 本章小结

本章阐明了在果园中进行害虫评估及进行害虫种间体态分析的意义，介绍了传统的果园害虫防治技术存在的不足和计算机视觉在农情领域中的应用，分别对国内外学

者利用计算机视觉技术在农情识别领域所取得的成果进行总结，进一步阐释了此项研究的意义。并对主要研究内容进行了简短的介绍。

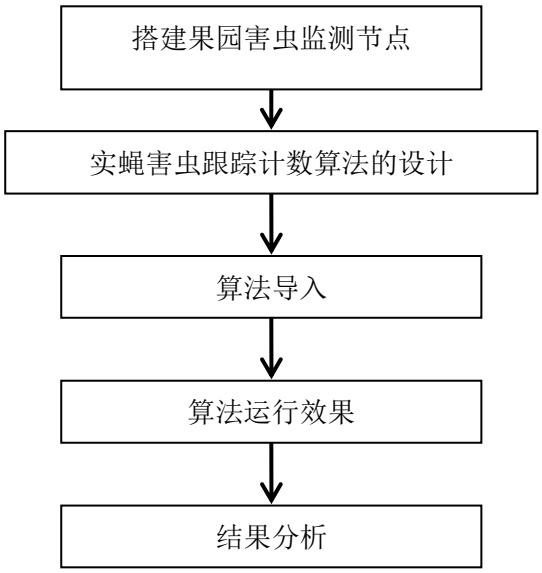


图 1.2 实物部分研究路线

2 基于 SIFT 算法区分实蝇种类

SIFT 算法为一种新兴的算法，其主要特性有：对同种图像进行变换操作后再与原图像进行匹配操作后，匹配结果稳定。

考虑到在现实环境中，人肉眼所观测到不同实蝇成虫的外观，除却一些存在残疾的成虫个体，在同种实蝇之间其外观不会存在太大的差异，且高速摄像机所捕捉到的实蝇图像样本不可能维持在一个固定的角度，因此本文将对 SIFT 算法能否将混在橘小实蝇、瓜实蝇、南瓜实蝇样本库中的橘小实蝇虫体样本区分出来进行验证性研究。

2.1 SIFT 算法简介

SIFT 算法于 1999 年由 British Columbia 大学大卫.劳伊（David G.Lowe）教授首次提出。

该算法的主要特点为：对图像进行旋转、尺度缩放、亮度变化后，与原图片进行匹配，匹配结果稳定，其对视角变化、仿射变换、噪声也保持一定程度的稳定性。

算法通过三个步骤来完成景物间的映射情况：关键点提取；通过一定算法提取出关键点局部特征；比较两幅图片中的特征点进行匹配，若两者相匹配则可认为该特征点为相同特征点。

（1）高斯金字塔

主要通过两个步骤来搭建高斯金字塔：

① 对图像进行高斯平滑；

② 对图像进行降采样。

经过多次重复上述步骤可以得到一个组数 A,如式 2.1 所示：

$$A=[\log_2(\min(M,N))]-3 \tag{2.1}$$

其中 M，N 为图像的行数和列数，每组层数 S 约为 3~5 的高斯金字塔，每组的尺度 δ 如式 2.2 所示：

$$\delta(s)=\delta_0\cdot 2^{s/S} \tag{2.2}$$

其中 S 为每组层数，s 为当前所在层数。由于高斯金字塔的搭建计算过于复杂，所以需要对其计算过程进行优化，以便于降低算法的复杂程度。Lindeberg 在研究后发现：尺度规范化的 LoG 算子具有尺度不变性，由此引出了 DOG 即高斯差分算子的理论。

（2）高斯差分算子（DOG）

高斯差分算子的主要公式如式 2.3 所示：

$$\begin{aligned} D(x,y,\delta) &= [G(x,y,k\delta) - G(x,y,\delta)] * I(x,y) \\ &= L(x,y,k\delta) - L(x,y,\delta) \end{aligned} \tag{2.3}$$

其中 $G(x,y,\delta)$为可变尺度高斯函数，$L(x,y,\delta)$为原图像进行高斯平滑后的图片。k 为固定的 $2^{1/s}$。上式把相邻尺度的图片进行高斯平滑后相减，最终能够得到一个高斯差分金字塔。

由 DOG 空间中的计算取得的局部极值点组成了整幅图像的关键点。局部极值点的取值方法为：空间中每一个像素点均需与它所有的相邻点比较，取出极值点。由于极值点的需精确定位，需要进行泰勒展开对 DOG 函数曲线进行拟合。拟合公式如式 2.4 所示：

$$\begin{gathered} D(X) = D + \frac{\partial D^T}{\partial X} X + \frac{1}{2} X^T \frac{\partial^2 D}{\partial X^2} X \\ \hat{X} = (x, y, \sigma)^T \\ D(\hat{X}) = D + \frac{1}{2} \frac{\partial D^T}{\partial X} X \end{gathered} \tag{2.4}$$

经过多次修正，除去不稳定极值点。并对求得的极值点进行去除边缘相应等操作，去除边缘相应的主要算法公式如式 2.5 所示：

$$\begin{gathered} H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \\ Tr(H) = D_{xx} + D_{yy} \\ Det(H) = D_{xx} \times D_{yy} - D_{xy} D_{xy} \\ r = 10 \\ \frac{Tr(H)^2}{Det(H)} \langle \frac{(r+1)^2}{r} \end{gathered} \tag{2.5}$$

此时求得的极值点具有缩放不变性。然后通过求各极值点梯度，为极值点赋予方向，使其拥有旋转不变性。像素点求梯度的方程如式 2.6 所示：

$$gradI(x,y) = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}) \tag{2.6}$$

求得梯度幅值 $\sqrt{(L(x+1,y)-L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2}$，梯度方向 $\tan^{-1}\left[\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right]$。此时取出来的关键点每个关键点有三个信息：位置、尺度、方向；使得关键点具有了平移、缩放、和旋转不变性。 试验中通过模板图与待测图进行关键点匹配，来完成对图像的匹配。

在本次研究中，主要是对 SIFT 算法能否作为实蝇图像识别进行探索，经过实验最终确定在 distRatio=0.6 处所取得实验效果最佳（特征点的匹配采用欧式距离作为两幅图片中关键点相似性的判定度量。在经过处理过后的两幅图像的特征点集中，找出欧氏距离最近的前两个关键点，如果这个距离除以次近的距离少于某个阈值，则认为这两个关键点是匹配的，反之不匹配。降低这个比例阈值，SIFT 匹配点数目会减少，但更加稳定。为了排除因为图像遮挡和背景混乱而产生的无匹配关系的关键点,Lowe 提出了比较最近邻距离与次近邻距离的方法，距离比率 distRatio 小于某个阈值的认为是正确匹配。因为对于错误匹配，由于特征空间的高维性,相似的距离可能有大量其他的错误匹配，从而它的 distRatio 值比较高。Lowe 推荐 distRatio 的阈值为 0.8。但对大量任意存在尺度、旋转和亮度变化的两幅图片进行匹配的结果表明 distRatio 取值在 0. 4~0. 6 之间最佳，小于 0. 4 的很少有匹配点，大于 0. 6 的则存在大量错误匹配点。）主要采取目标图像—样本两两匹配的方法进行试验。

2.2 基于 SIFT 算法的实蝇识别实验

由于 SIFT 算法是针对同一目标对象在旋转、尺度缩放、亮度变化等条件下进行匹配的，由于同种类型的实蝇用肉眼观察其物理特征相差不大，所以本实验设想对于同一种类的实蝇类型来讲，可以近似认定为同一目标。因此尝试着用 SIFT 算法对三种不同的实蝇种类进行处理，验证是否能将橘小实蝇害虫从三种实蝇中区分鉴别出来。

本实验先将实蝇图片转化为灰度图片,在小波去噪的基础上对 80~100 的像素拉伸至 0~255,运用 SIFT 算法对实蝇目标进行分析处理。试验中对拍摄到的 70 头橘小实蝇和 60 头瓜实蝇以及南瓜实蝇图片样本进行搭建图像样本库的操作，文中算法所需的图像样本均从该样本库中进行选取。

2.2.1 橘小实蝇-橘小实蝇处理结果

从图像样本库中随机选取 10 幅橘小实蝇图片作为橘小实蝇匹配样本库，并将该库中包含的橘小实蝇图像样本从原样本库中移除以避免对同副图像进行匹配，随后从匹

配样本中随机选取一副橘小实蝇图像与原样本库中的 30 幅橘小实蝇图像进行算法匹配，用 SIFT 算法进行处理。

本文对原样本库中的 30 幅橘小实蝇图像进行 SIFT 算法匹配后，结果如表 2.1 以及图 2.2 所示。30 幅橘小实蝇图像中，橘小实蝇与橘小实蝇图像的匹配点数平均值为 11，标准差为 1.93。

表 2.1 橘小实蝇-橘小实蝇处理结果

图像序号	匹配点数	图像序号	匹配点数	图像序号	匹配点数	图像序号	匹配点数	图像序号	匹配点数	图像序号	匹配点数
1	11	6	11	11	14	16	15	21	10	26	10
2	10	7	10	12	11	17	8	22	10	27	10
3	13	8	13	13	10	18	13	23	14	28	13
4	11	9	11	14	10	19	11	24	11	29	15
5	8	10	10	15	9	20	10	25	10	30	8

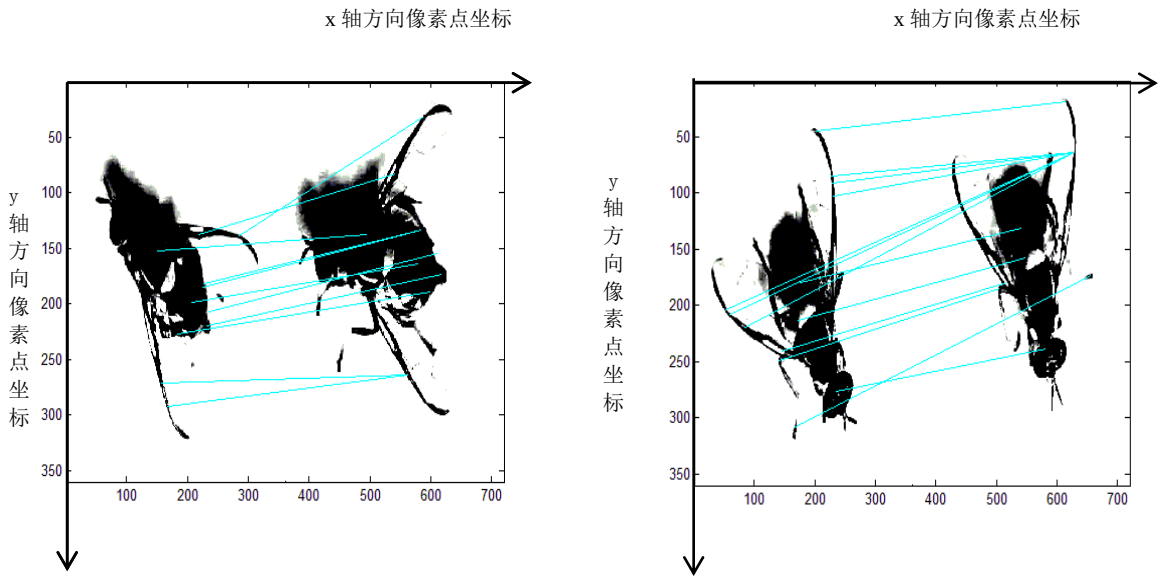


图 2.2 橘小实蝇-橘小实蝇在 distRatio=0.6 下处理结果

2.2.2 南瓜实蝇-橘小实蝇处理结果

从橘小实蝇匹配样本库中随机选取一副实蝇样本作为匹配模板，再从图像样本库

中随机选用 30 幅南瓜实蝇图像进行匹配实验，用 SIFT 算法进行处理，结果如表 2.2 以及图 2.3 所示。30 幅瓜实蝇图像中，南瓜实蝇与橘小实蝇图像的匹配点数平均值为 0.13，标准差为 0.73。实验中发现图 13 与橘小实蝇样本有匹配点数，经过对该图的灰度图像进行核对后发现：由于该实蝇的背部条纹带受损，在经过图像处理后背部条纹带近乎消失，因此产生了错配。

表 2.2 南瓜实蝇-橘小实蝇处理结果

图像序号	匹配点数	图像序号	匹配点数	图像序号	匹配点数	图像序号	匹配点数	图像序号	匹配点数	图像序号	匹配点数
1	0	6	0	11	0	16	0	21	0	26	0
2	0	7	0	12	0	17	0	22	0	27	0
3	0	8	0	13	4	18	0	23	0	28	0
4	0	9	0	14	0	19	0	24	0	29	0
5	0	10	0	15	0	20	0	25	0	30	0

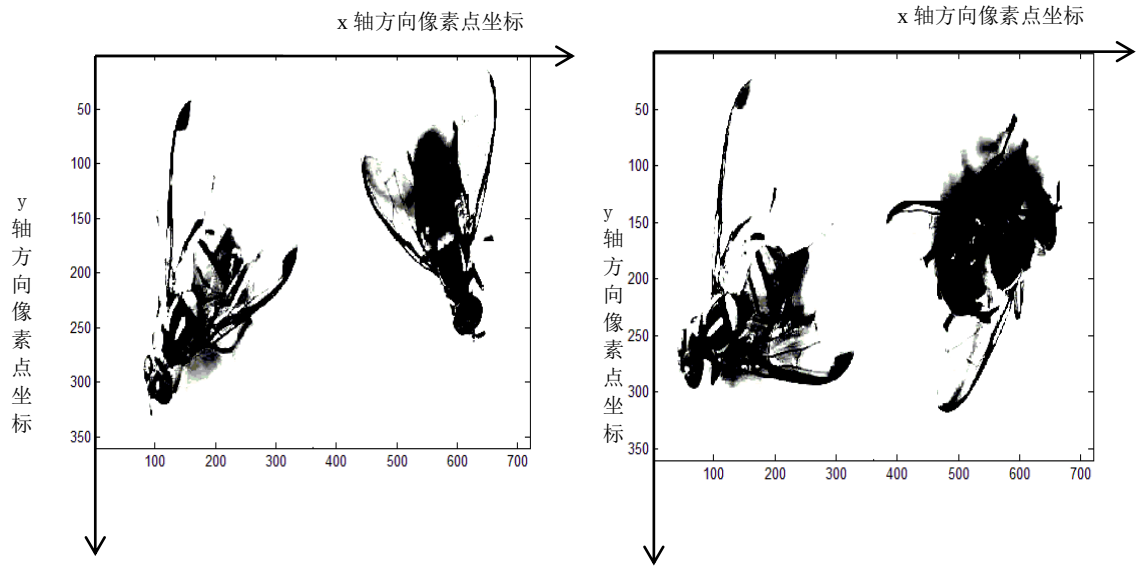


图 2.3 南瓜实蝇-橘小实蝇在 distRatio=0.6 下处理结果

2.2.3 瓜实蝇-橘小实蝇处理结果

从橘小实蝇匹配样本库中随机选取一副实蝇样本作为匹配模板，再从图像样本库中随机选用 30 幅瓜实蝇图像进行匹配实验，用 SIFT 算法进行处理，结果如表 2.3 以及

图 2.4 所示。30 幅瓜实蝇图像中，瓜实蝇图像与橘小实蝇的匹配点数平均值为 0.27，标准差为 1.05。实验中发现图像序号为 4 处与图像序号为 23 处的橘小实蝇样本有匹配点数，经过对该图的灰度图像以及原始图像进行核对后发现：由于对于图像序号为 23 的实蝇样本拍摄时角度选取不好使得背部条纹带部分区域被遮挡，在经过图像处理后背部条纹带特征变得模糊，图像序号为 4 的实蝇样本是由于瓜实蝇本身背部条纹带有损伤，因此这两幅图像产生了错配。

表 2.3 瓜实蝇-橘小实蝇处理结果

图像序号	匹配点数	图像序号	匹配点数	图像序号	匹配点数	图像序号	匹配点数	图像序号	匹配点数	图像序号	匹配点数
1	0	6	0	11	0	16	0	21	0	26	0
2	0	7	0	12	0	17	0	22	0	27	0
3	0	8	0	13	0	18	0	23	3	28	0
4	5	9	0	14	0	19	0	24	0	29	0
5	0	10	0	15	0	20	0	25	0	30	0

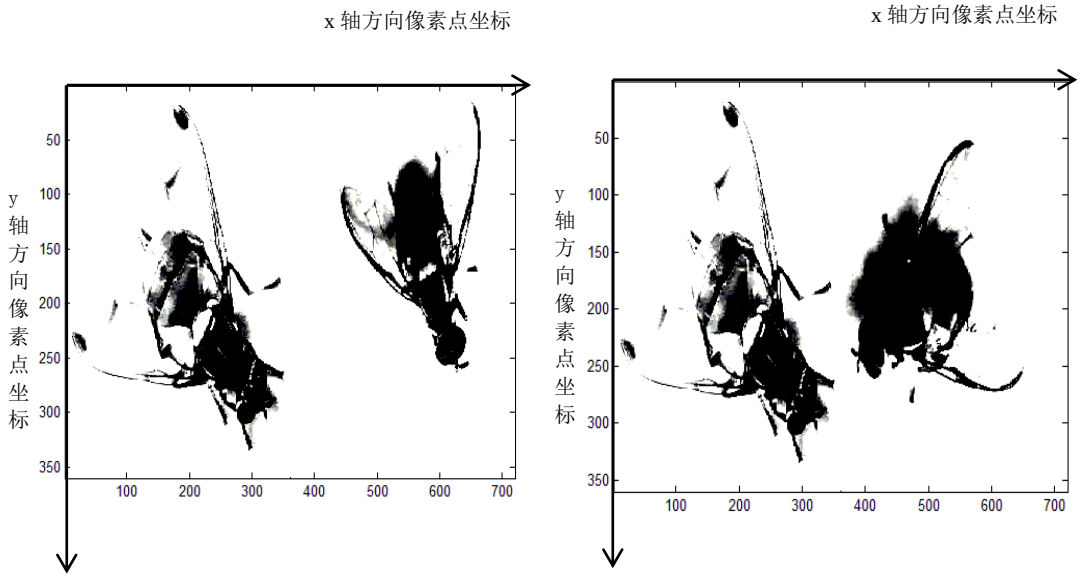


图 2.4 瓜实蝇-橘小实蝇在 distRatio=0.6 下处理结果

2.2.4 识别实蝇成虫种类的匹配点识别区间

橘小实蝇、南瓜实蝇和瓜实蝇匹配点个数的平均值分别为：11；0.13 和 0.27。可以看出，橘小实蝇与另外两种实蝇相比：匹配点个数的平均值有着明显的差异；瓜实

蝇，南瓜实蝇与橘小实蝇样本图像匹配所得的匹配点平均值几乎为 0，以此为依据可以列出式 2.7：

$$Fruitfly = \begin{cases} 1 & MatchingPo\,\mathrm{int} > 0 \\ 2 & MatchingPo\,\mathrm{int} = 0 \end{cases} \quad (2.7)$$

式中：

Fruitfly—实蝇种类，“1”代表橘小实蝇，“2”代表其他两种实蝇；

Matching Point—实验中所获得的匹配点。

2.3 结果分析

随机选取图像样本库中 10 的幅实蝇图像用于检验算法的图像，用所选取的实蝇图像样本与匹配模板进行匹配试验。计算匹配点数目，区分待测实蝇的种类，通过匹配点数个数确定是否为橘小实蝇后，人工的对所选取的图像样本进行实蝇种类的识别，识别结果如表 2.4 所示：

表 2.4 检验试验识别效果

图像序号	匹配点数	是否为橘小实蝇	是否准确	图像序号	匹配点数	是否为橘小实蝇	是否准确
1	13	橘小实蝇	是	6	0	其他实蝇	是
2	10	橘小实蝇	是	7	0	其他实蝇	是
3	10	橘小实蝇	是	8	0	其他实蝇	是
4	8	橘小实蝇	是	9	0	其他实蝇	是
5	0	其他实蝇	是	10	9	橘小实蝇	是

2.4 本章小结

本章主要论证了 SIFT 算法对于三种实蝇的识别效果，主要计算在 SIFT 算法下各不同实蝇种类图片对于橘小实蝇图片的匹配点数。并根据返回的匹配点数对捕获的实蝇样本是否为橘小实蝇进行分析。从而从捕获实蝇中区分出在柑橘果园中主要为害橘小实蝇。

实验结果表明，本试验能够将橘小实蝇从三种果蝇中识别出来：当识别点数不为 0

时，可以判定该识别对象为橘小实蝇。反之则为其他两种类型的实蝇。但在野外环境下，由于图像获取装置对于实蝇样本的获取多数情况下并不会如本试验样本库中一样将实蝇的所有物理信息进行完美的展示，或者多数实蝇还存在斑点区域破损，残翅等生理缺陷，在该情况下会对本算法产生较大的影响。

3 实蝇图像配准处理

3.1 对实蝇样本进行配准处理的缘由

因为捕虫器高速摄像头所捕获到的实蝇害虫图像姿态各异，不可能如实验中的样本图像一样，处于画面正中。为了确保实验中所采用的算法在野外环境下仍能起到作用，因此需要对高速摄像头所捕获到的实蝇图像进行图像配准处理。

3.2 样本配准处理算法

3.2.1 基于颜色特征初步分割目标相似区域

通过对采样的实验目标的观察，发现实蝇都有一个特定关节，可以用来区别果蝇和其他昆虫。以下通过颜色分割处理效果。最后通过区域似圆度和大小来识别特定关节。处理效果见图 3.1，其中图 3.1a 为区域生长分割处理后的图片，图 3.1 b 为颜色分割后的斑点图片，由下图可以看出颜色分割后的实蝇斑点图片具有独特的特征，可以通过区域似圆度和大小来识别特定关节。

a.区域生长分割处理后的图片　　b.颜色分割后的斑点

图 3.1 特征分割后的效果图

3.2.2 基于 Hough 变换搜索目标团块长轴

3.2.2.1 候选区域直线的建模

在图像的笛卡尔坐标空间中，经过点 (x_i, y_i) 的直线表示如式 3.1 所示：

$$y_i = \alpha x_i + b \tag{3.1}$$

其中 α 为斜率，b 为截距。

背景中每一条行线边缘都可以用 $y_i = \alpha x_i + b$ 来表示。考虑到背景中的行线边缘是相互平行的，且接近水平，因此可得到以下两个直线的约束条件：

（1）每一列的相同行线之间的斜率值十分接近；

（2）不同列、相同行的行线共线。

通过上述两个约束条件，可以排除其他静态背景所形成的非水平线的干扰，从而把背景区域中的直线比较精确的检测出来。

3.2.2.2 Hough 变换原理

在坐标系中有无数条通过点 (x_i, y_i) 的直线，且对应于不同的 α 和 b 值，这些直线都满足 $y_i = \alpha x_i + b$。如果将 x_i 和 y_i 视为常数，而将原本的参数 α 和 b 看作变量，式 $y_i = \alpha x_i + b$ 可表示如式 3.2 所示：

$$b = -x_i \alpha + y_i \tag{3.2}$$

通过 $y_i = \alpha x_i + b$ 把直线从直角坐标系平面变换到参数平面 $\alpha - b$。这个变换就是直角坐标中对于 (x_i, y_i) 点的 Hough 变换。该直线式图像坐标空间中的点 (x_i, y_i) 再参数空间的唯一方程。图像坐标中的另一个点 (x_i, y_i)，它在参数空间中也有相应的一条直线，该直线所对应的公式如式 3.3 所示：

$$b = -x_i \alpha + y_i \tag{3.3}$$

这条直线与经过点 (x_i, y_i) 的直线在参数空间中相交于一点 (α_k, b_k)，如图 3.2a、3.2b 所示。

经过点 (x_i, y_i) 的各向直线和经过点 (x_i, y_i) 的各向直线各个参数平面 $\alpha - b$ 上形成一条直线，且相交于点 (α_k, b_k)，而点 α_k 和 b_k 就是图像坐标系 $x - y$ 中的点 (x_i, y_i) 和点 (x_i, y_i) 所确定的直线的参数。

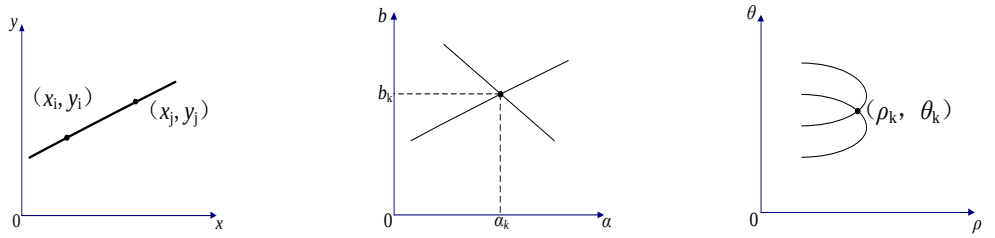
使用直角坐标表示直线，当直线为一条垂直直线或接近垂直直线时，该直线的斜率为无穷大或接近无穷大，从而无法在参数空间 $\alpha - b$ 中表示出来。为了解决这一问题，可以采用极坐标系表示，极坐标中过点 (x_i, y_i) 的直线的参数方程如式 3.4 所示：

$$\rho = x_i \cos\theta + y_i \sin\theta \tag{3.4}$$

式中，ρ 为直角坐标系原点到直线的距离；θ 为直线与 y 轴的夹角。其对应的 Hough 变换如图 3.2c 所示。

Hough 变换是根据局部度量来计算全面描述函数，对局部信息缺损不敏感，对随机噪声具有良好的抑制作用。但由于计算量比较大，占用存储空间比较多，不适合实时性要求高的场合，但是实蝇样本图片的区域划分属于一次性操作，背景几乎不会有

任何改变，实时性要求不高，所以用 Hough 变换实现区域划分是可行的。



a．图像坐标系　　b.直角坐标系参数表示　　c.极坐标系参数表示

图 3.2 Hough 变换原理示意图

3.2.2.3 基于 Hough 变换搜索目标团块长轴

本文利用图的分块区域作为目标搜索区域，对这个区域进行 Hough 直线检测，从而实现目标长轴的搜索和标定。算法的具体实现步骤如下：

（1）建立一个累加器记做 A。一般来说对大小为 H×L 的图像样本来说，ρ 的取值区间一般认为落在$\left[-\sqrt{2}\min\{H,L\}/2,\sqrt{2}\min\{H,L\}/2\right]$之间，同时$\theta$角度偏移量的取值范围为$[-90°,90°]$。因为开始时需对数组 $A(\rho,\theta)$ 进行初始化，本实验中认定的数组 $A(\rho,\theta)$ 的初始化值为初始化为 0，

（2）在图像样本中任取一点(x_i,y_i)，将 $A(\rho,\theta)$ 中的每一个θ值代入式 3.4 中，可以得出相应的 ρ 的取值。对于每一对相同的(ρ,θ)，都将在相应的数组元素 $A(\rho,\theta)$ 表中加 1。

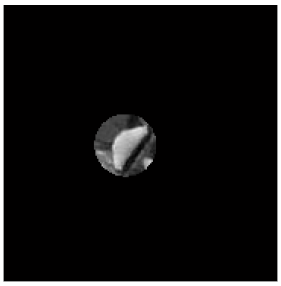
（3）计算出所有(ρ,θ)对后，寻找出 $A(\rho,\theta)$ 的最大峰值，由其交点(ρ_k,θ_k)可得在该图像中共线点数最多的直线方程的参数；以此类推可以得出其他峰值。

通过比较长轴中点坐标和斑点块质心的值，可以得出长轴与目标方向的偏移角度，根据偏移的角度来对图像进行旋转配准。

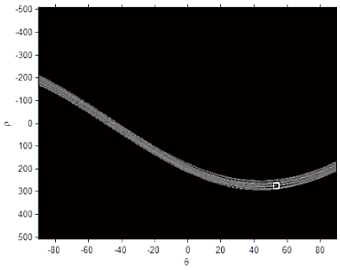
配准效果图如图 3.3 所示。

3.3 结果分析

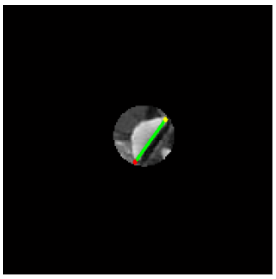
对实蝇素材库的三种实蝇，共 190 只的样本进行配准，统计的配准准确率为 97.5%，从侧面也证实了预选的特定区域对于实蝇的生长方向具有比较强的相关性。本文中 CCS 平台上所做的实蝇害虫种类区分算法中所用到的实蝇害虫样本均经过图像配准处理。



a.搜素区域灰度图像



b.Hough 矩阵峰值点



c.直线检测效果图

d.最终处理效果图

图 3.3 Hough 变换效果图

4 基于 CCS 代码调试器的实蝇种类区分算法

4.1 RGB 到 YCbCr 色彩空间的转换

由于 *RGB* 色彩空间本身的特性限制，导致光照与亮度这两个因素对基于 *RGB* 色彩空间的图像识别算法中影响较大，为了消除背景区域中的光照影响，本实验拟采用对亮度敏感度较低的 YC_bC_r 色彩空间对实蝇目标区域进行颜色分割，YC_bC_r 色彩空间与 *RGB*、*HIS* 和 *HSV* 等色彩空间相比，具有亮度对背景颜色影响较小、背景区域颜色描述连续性更好、背景区域和目标区域因光照不均匀而引起的区域重叠不明显等优点，便于进行背景分割和目标提取（李震等，2014）。

本研究先将样本库中的 *RGB* 图像进行图像配准处理后，转换至 YC_bC_r 色彩空间，如式 4.1~4.3 所示：

$$Y=0.299R+0.587G+0.114B \quad (4.1)$$

$$C_b=0.564(B-Y) \quad (4.2)$$

$$C_r=0.713(R-Y) \quad (4.3)$$

式中，*Y*—YC_bC_r 空间的 *Y* 分量；C_b—YC_bC_r 空间的 C_b 分量；C_r—YC_bC_r 空间的 C_r 分量；*R*—*RGB* 空间的 *R* 分量；*G*—*RGB* 空间的 *G* 分量；*B*—*RGB* 空间的 *B* 分量。

在 YC_bC_r 色彩空间图像中，C_b 分量更加适合提取黄色的纵带纹区和盾片区。C_b 分量的灰度图片如图 4.1 所示。而对于实蝇胸部和背部条纹带及盾片区域，则对蓝色色差变化较为敏感。

a.橘小实蝇灰度图像　b.南瓜实蝇灰度图像　c.瓜实蝇灰度图像

图 4.1 三种实蝇样本图像在 YC_bC_r 空间中的 C_b 分量灰度图像

4.2 直方图均衡化

对图 4.1 中的图片用 Otsu 算法进行二值化，发现实蝇背部的纵带纹区有不同程度的丢失，分析原因，主要是纵带纹区与实蝇身体其他部分和背景之间的灰度差异不明显。针对 YC_bC_r 空间中的 C_b 分量灰度图像存在的问题，本研究采用直方图均衡化的方法（吴成茂，2013），以期提高图像中待提取区域和其他部分之间的对比度，步骤如下：

（1）统计原始图像的直方图，如式 4.4 所示：

$$p_r(r_k)=\frac{n_k}{n} \tag{4.4}$$

式中：

$p_r(r_k)$—输入图像灰度级为 k 的像素数占输入图像的总像素数的比率；

r_k—输入图像的灰度级；

n_k—输入图像灰度级为 k 的像素数；

n—输入图像的总像素数。

（2）计算直方图累积分布曲线，如式 4.5 所示：

$$S_k=T(r_k)=\sum_{j\to 0}^{k}p_r(r_j)=\sum_{j\to 0}^{k}\frac{n_j}{n} \tag{4.5}$$

式中：

S_k—累积积直方图各项；

$p_r(r_j)$—原始直方图各项。

（3）用累积分布函数作变换函数计算图像变换后的灰度级，如式 4.6 所示：

$$S_k=int[(max(r_k)-min(r_k))\ s_k+0.5] \tag{4.6}$$

式中：

s_k—图像计算前的灰度级；

S_k—累积直方图各项；

$max(r_k)$—输入图像最大灰度级；

$min(r_k)$—输入图像最小灰度级。

（4）建立输入图象与输出图象灰度级之间的对应关系，使变换后图像的灰度级范围和原图像的范围一致。

直方图均衡化处理后图像如图 4.2 所示：

a.橘小实蝇处理图像　　b.南瓜实蝇处理图像　　c.瓜实蝇处理图像

图 4.2 直方图均衡化后的图像

4.3 图像的腐蚀处理

对于全局二值化处理后的图像，存在着较多的噪声，且有些实蝇图片的斑点区域或盾形区域与其他区域有相连的情况，若不进行腐蚀处理，主要会出现以下两种错误情况：

（1）实蝇背部旁边的条形带与其他区域相连，其面积变大，导致被错误地识别为盾形区。

（2）实蝇的盾形区与其他区域相连，提取出的盾形区带有其他区域，导致不能正确返回盾形区的区域信息，如边界信息，区域像素数等。

腐蚀一般采用的算法公式如式 4.7 所示：

$$g(x,y)=erode[f(x,y),B]=\min\{f(x+dx,y+dy)\,|\,(dx,dy)\in D_B\} \tag{4.7}$$

式中：

g(x,y)—为腐蚀后的灰度图像；

B—为结构元素，在灰度形态学操作中一般取 *B* 值为 0。

基于图 4.3 的处理结果，对二值化图像进一步进行图像腐蚀操作，所得结果如图 4.4 所示。

4.4 盾片区提取以及斑点提取

4.4.1 联通区域标记算法

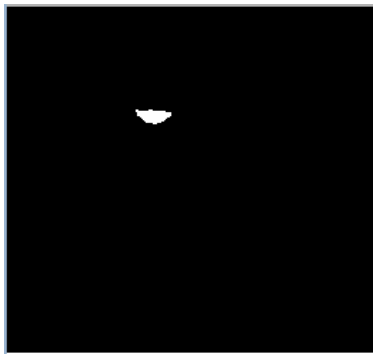
本文中主要采用两次扫描（TWO-PASS）法进行联通区域标记，为往后的盾片区斑点提取提供材料支撑，二次扫描算法需对图像数据进行两次扫描，具体过程如下：

对一幅图像 A 从上到下，从左到右做两次扫描。在第一次扫描中，假设扫描到了

a.未做腐蚀操作的识别效果

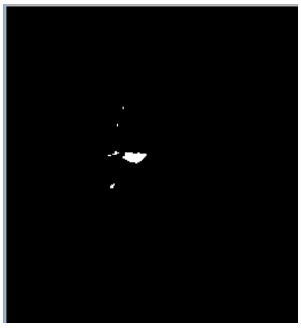
b.做腐蚀操作后的识别效果

c.未做腐蚀操作的识别效果



d.做腐蚀操作后的识别效果

图 4 .3 图片腐蚀结果示意图



a.橘小实蝇

b.南瓜实蝇

c.瓜实蝇

图 4.4 对二值化图像进行腐蚀后的结果

第一个非背景点 *A(i,j)*，那么检查它的左边 *A(i-1,j)*和上边 *A(i,j-1)*的两个邻接像素点(其中 *A(i,j), A(i-1,j)*以及 *A(i,j-1)*均为图像 A 中的像素点坐标)。由于是顺序扫描，因此左边和上边的点肯定已经被扫描过，那么有三种情况：

（1）*A(i-1,j)*和 *A(i,j-1)*都未被标记，则给 *A(i,j)*分配一个新标记符；

（2）*A(i-1,j)*和 *A(i,j-1)*有一个被标记（如表记为 1），则给 *A(i,j)*同样的标记符（标记为 1）；

（3）*A(i-1,j)*和 *A(i,j-1)*都被标记，那么：如果两个标记符相同（都为 1），则给 *A(i,j)* 同样的标记符（标记为 1），如果两个标记符不同（一个为 1，另一个为 2），则将该点 *A(i,j)*标记为其中一个标记符（标记为 1），同时记下两个标记符（1 与 2）等价。第一次扫描结束。

但是这时会出现同一联通区域中的像素被标以不同标记符的情况，如“井”字形区域的分支处。因此要进行第二次扫描。

第二次扫描中，用等价表中最低标记符取代等价表中的每个标记符，从而对属于同一联通区域而被标记为不同标记符的像素重新做标记。两次扫描结束，标记完成。

4.4.2 实蝇盾片区提取

实蝇的盾片区提主要步骤为：先对二值化后的实蝇斑点进行图像腐蚀操作，去除图片中较小的像素点，避免其对实验结果造成影响。通过对实蝇原图像的体态特征，以及实蝇体表斑点特征分析可得，实蝇体表斑点最大区域为实蝇腰腹部的盾片区，因此对腐蚀后的图片进行提取最大连通区域操作，对像素最多的区域进行提取，即可获取实蝇盾片区域斑点。结果如图 4.5 所示：

a.橘小实蝇盾片区提取　　b.南瓜实蝇片区提取　　c.瓜实蝇片区提取

图 4.5 盾片区与提取结果

4.4.3 提取盾片区重心的算法

为了方便对图像中的像素点运算过程进行表示，设置一个像素点累加公式 M_{pq}，M_{pq} 的计算如式 4.8 所示：

$$M_{pq}=\sum_{x\to i,y\to j} i^{p} j^{q} f(i,j) \tag{4.8}$$

式中：

x,y,i,j—区域点的坐标（在数字图像中的像素坐标）；

f(i,j)—图片对应 *i,j* 坐标上的像素点值;

p,q—与像素点坐标有关，其中 p 对应于图像 x 的方向，q 对应于图像的 y 方向。若其为 0，则表示不需对该方向上的像素进行计算，若其值为 1，则表示对图像该方向上的像素进行计算。

令 X_c,Y_c 表示区域重心的坐标，则有关区域重心的计算方法如式 4.9、4.10 所示：

$$X_c = \frac{M_{10}}{M_{00}}; \tag{4.9}$$

$$Y_c = \frac{M_{01}}{M_{00}}; \tag{4.10}$$

式中：

M_{00}:盾片区区域的面积;

M_{10}:盾片区区域 x 方向像素的总和;

M_{01}:盾片区区域 y 方向像素的总和。

4.4.4 纵带纹区域搜索

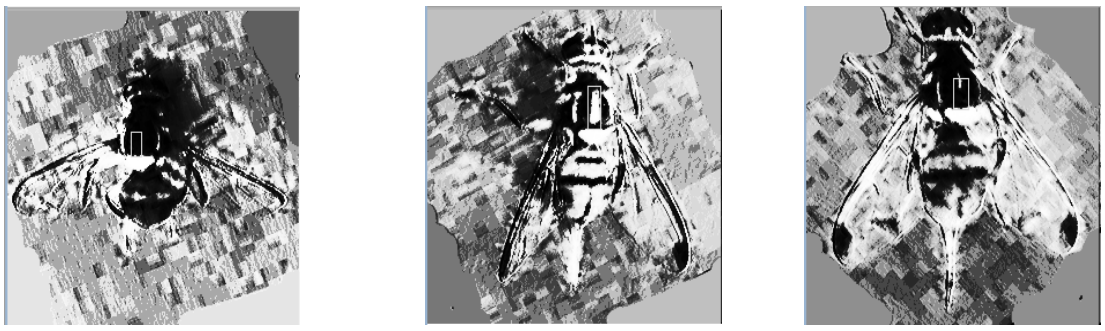
根据盾形区的区域信息，以盾形区的重心坐标为中心，向两边扩展，左右两边的扩展宽度分别为左右边界与重心坐标之差的一半公式，如式 4.11、4.12 所示：

$$X_a = \frac{X_c - X_{right}}{2}; \tag{4.11}$$

$$Y_a = \frac{Y_c - Y_{left}}{2}; \tag{4.12}$$

其中，X_a 为搜索区域的右边界坐标， Y_a 为搜索区域的左边界坐标，X_c, Y_c 表示区域重心的坐标, X_{right} 为盾型区右边界, Y_{left} 为盾型区左边界。

以此作为搜索区域的宽度。从盾形区顶部开始，向上以 5 个像素为阶度，对搜索区域进行生长条形带搜索，若搜索到了条形带部分，则向上继续生长，直至向上搜索后，搜索区域像素为 0。则可判别为条形带搜索完成。另因盾形区与条形带之间有一定间隔，设置搜索区最低生长次数的阈值为（*right-left*）/ 5 次，以确保搜索区能够跨越间隔。结果如图 4.6 所示:



a.橘小实蝇二值图　　b.南瓜实蝇二值图　　c.瓜实蝇二值图

d.橘小实蝇条形带　　e.南瓜实蝇条形带　　f.瓜实蝇条形带

图 4.6 搜索区生长效果以及条形带提取结果

4.5 基于 CCS 代码调试器的实验过程

本研究选取橘小实蝇，瓜实蝇以及南瓜实蝇 3 种实蝇成虫作为研究对象，通过对橘小实蝇、瓜实蝇南瓜实蝇进行外观的观察，发现其腰腹部盾片正上方有无竖状条纹、竖状条纹的大小，因此可以利用图像算法把橘小实蝇、瓜实蝇和南瓜实蝇区分开。李震等通过算法将不同朝向的实蝇样本的盾片长轴与图像的底端相垂直，从而完成果蝇图像的配准（李震等，2014）。文韬等在研究中未对捕蝇器中采集到的实蝇种类进行区分（文韬等，2011），李震等的研究把橘小实蝇和从三种实蝇中识别出来，未能准确识别三种实蝇。并且由于运用的是 BP 神经网络，耗时较长（李震等，2014）。

本研究对上述研究进行了相应的改进，在图像配准后的基础上对图像进行处理转换为 YC_bC_r 后提提取 C_b 分量，直方图均衡化后对处理图像进行二值化处理，提取出实蝇斑点，提取斑点中连通区域最大的，为果蝇盾片区，最终通过果蝇盾片区的相关参数确定果蝇背部斑点的识别区域，通过计算该区域像素点与实蝇盾片区的比值对实蝇种类进行识别，且本算法采用的均为基础算法，因此用该算法对实蝇图像进行区分计算耗时较少。

4.5.1 橘小实蝇图像处理结果

随机的从图像样本库中选取用于算法设计的 30 幅橘小实蝇图像，用本研究中的算法进行处理，结果如表 4.1 所示。30 幅橘小实蝇图像中黄色纵带纹区域面积与腰腹部盾片区面积比值的平均值为 0.0009，标准差为 0.00487，1/4 分位和 3/4 分位均为 0。

对下表数据进行分析后发现：第 20 号图像中的面积比值与其他比值相比存在较大的差异。经过分析原图，发现是由于该橘小实蝇背部有较严重的损伤，其颜色与纵带纹颜色较为接近，导致算法误将部分损伤区域划分为纵带纹，产生面积比值与其他比值相比差异较大的结果。

表 4.1 橘小实蝇图像中黄色纵带纹区域面积与腰腹部盾片区面积的比值

图像序号	面积比值	图像序号	面积比值	图像序号	面积比值	图像序号	面积比值	图像序号	面积比值	图像序号	面积比值
1	0	6	0	11	0	16	0	21	0	26	0
2	0	7	0	12	0	17	0	22	0	27	0
3	0	8	0	13	0	18	0	23	0	28	0
4	0	9	0	14	0	19	0	24	0	29	0
5	0	10	0	15	0	20	0.0267	25	0	30	0

4.5.2 南瓜实蝇图像处理结果

选用图像样本库中用于算法设计的 30 幅南瓜实蝇图像，用本研究中的算法进行处理，结果如表 4.2 所示。30 幅南瓜实蝇图像中黄色纵带纹区域面积与腰腹部盾片区面积比值的平均值为 0.7498，标准差为 0.30683，1/4 分位为 0.4711，3/4 分位为 0.9342。

对下表数据进行分析后发现：第 13 号图像中的面积所得比值与其他实蝇图像中的面积所得比值相比，其值较小。

经过分析原图，发现是由于该南瓜实蝇背部有较严重的损伤，因此对该实蝇黄色纵带纹区域面积进行计算，其值偏小且导致其黄色纵带纹区域不连续。因此通过对图像进行腐蚀操作后该实蝇黄色纵带纹区域面积比同批次其他实蝇小得多，故产生面积比值与其他图像比值相比偏小的结果。

表 4.2 南瓜实蝇图像中黄色纵带纹区域面积与腰腹部盾片区面积的比值

图像序号	面积比值	图像序号	面积比值	图像序号	面积比值	图像序号	面积比值	图像序号	面积比值	图像序号	面积比值
1	0.8396	6	0.6285	11	1.4925	16	0.8396	21	0.5893	26	0.9597
2	0.8985	7	0.5893	12	1.0626	17	0.6553	22	0.4444	27	0.8511
3	1.0616	8	0.9606	13	0.2396	18	0.6266	23	0.4407	28	0.4225
4	0.9461	9	0.8977	14	0.9479	19	0.8937	24	0.3740	29	0.4216
5	0.6540	10	1.4815	15	0.5510	20	0.8945	25	0.3906	30	0.4390

4.5.3 瓜实蝇图像处理结果

选用图像样本库中用于算法设计的30幅瓜实蝇图像，用本研究中的算法进行处理，结果如表 4.3 所示。30 幅瓜实蝇图像中黄色纵带纹区域面积与腰腹部盾片区面积比值的平均值为0.2493，标准差为0.15885，1/4 分位为0.1712，3/4 分位为0.2829。

对下表数据进行分析后发现：第 4、13、16 号图像中的面积所得比值教其他值来说大得多。对 3 幅图像的原图进行分析后发现，上述三幅实蝇样本中黄色纵带纹附近有较多污渍，污渍与纵带纹粘连，故产生面积比值与其他图像比值相比偏大的结果。

表 4.3 瓜实蝇图像中黄色纵带纹区域面积与腰腹部盾片区面积的比值

图像序号	面积比值	图像序号	面积比值	图像序号	面积比值	图像序号	面积比值	图像序号	面积比值	图像序号	面积比值
1	0.0535	6	0.1111	11	0.2435	16	0.6959	21	0.3962	26	0.3965
2	0.1746	7	0.2827	12	0.1618	17	0.1907	22	0.2829	27	0.2346
3	0.0735	8	0.1700	13	0.5200	18	0.3168	23	0.2121	28	0.1914
4	0.7097	9	0.1812	14	0.1109	19	0.2121	24	0.1750	29	0.0741
5	0.1972	10	0.2000	15	0.2260	20	0.3171	25	0.1970	30	0.1700

4.5.4 配对样本 *t* 检验结果

将橘小实蝇、南瓜实蝇、瓜实蝇的各 30 个面积比值，采用配对样本 *t* 检验的方法分析每组比值之间的差异，*P* 值分别为：1.16e-8、6.09e-14 和 1.88e-9，检验结果均小于 0.05，说明 3 组面积比值之间存在显著差异。

4.5.5 识别实蝇成虫种类的面积比值区间

橘小实蝇、南瓜实蝇和瓜实蝇面积比值的 1/4 分位和 3/4 分位分别为：0，0；0.4711，0.9342 和 0.1712，0.2829。可以看出，各面积比值的 1/4 分位和 3/4 分位所构成的数据区间是不连续的，如果直接应用该结果作为识别依据，可能出现待测图像的面积比值落入数据空隙的情况。因此，本研究计算橘小实蝇面积比 3/4 分位与瓜实蝇面积比 1/4 分位的均值 0.0856，以及瓜实蝇 3/4 分位与南瓜实蝇 1/4 分位的均值 0.3770，作为识别三种实蝇的依据，如式 4.13 所示：

$$Fruitfly=\begin{cases}1 & Racial=0\\ 2 & 0.0856<Racial<0.3770\\ 3 & Racial\geq 0.3770\end{cases} \tag{4.13}$$

式中：

Fruitfly—实蝇种类，“1”代表橘小实蝇，“2”代表瓜实蝇，“3”代表南瓜实蝇；

Racial—实蝇图像中黄色纵带纹区域面积与腰腹部盾片区面积的比值。

4.5.6 结果分析

从图像样本库中选取 10 幅实蝇待测样本用于检验算法的图像，运用本章算法对所选取的 10 幅待测样本进行算法分析，提取出待测样本中实蝇的盾片区域以及黄色条纹带，并计算黄色纵带纹区域面积与腰腹部盾片区面积的比值，区分待测实蝇的种类。对待测样本进行算法处理提取盾片区域及黄色条纹带区域，并计算其比值后可得结果，如表 4.4 所示。通过下表可以看出，运用本算法能够较好的区分三种不同种类的实蝇。

4.6 本章小结

本章主要是在 CCS 代码调试器的开发环境下，主要通过对三种不同种类实蝇的黄色纵带纹区域面积与腰腹部盾片区面积的比值进行计算，并根据三种实蝇样本黄色纵带纹区域面积与腰腹部盾片区面积的比值的不同来确定区分各种实蝇的阈值范围。完成了对三种不同实蝇的种类区分。

表 4.4 实蝇识别效果

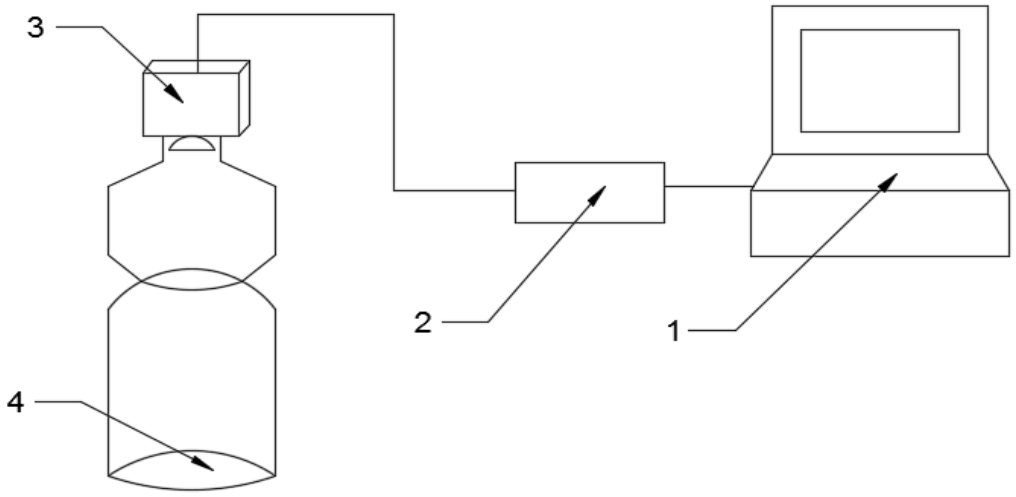
图像序号	面积比值	识别种类	是否准确	图像序号	面积比值	识别种类	是否准确
1	0	橘小实蝇	是	6	0.5311	南瓜实蝇	是
2	0	橘小实蝇	是	7	0.2121	瓜实蝇	是
3	0	橘小实蝇	是	8	0.0741	瓜实蝇	是
4	0.8985	南瓜实蝇	是	9	0.1172	瓜实蝇	是
5	1.4859	南瓜实蝇	是	10	0.1809	瓜实蝇	是

5 实验平台搭建以及软件测试

5.1 平台搭建

5.1.1 计算机视觉设备

本实验中的主要硬件核心为基于物理平台所搭建的视觉设备，它包括识别系统（创龙 OMAP-L138 开发板、图像采集电路卡、CMOS 摄像头）、捕虫器。本实验将 CMOS 摄像头用不透明的胶布固定在捕虫器上端的观测口中，以避免白天强光对实蝇害虫的跟踪计数产生影响。CMOS 摄像头所采集到的视频图像通过图像采集卡传输到创龙 OMAP-L138 开发板中并由开发板中固化的实蝇跟踪计数算法对所采集到的图像内容进行分析，若视屏中出现橘小实蝇成虫，则算法可自动对实蝇虫体进行跟踪锁定。系统组成主要如图 5.1 所示：



1. 创龙 OMAP-L138 开发板 2. 图像采集电路卡 3. CMOS 摄像头 4.捕虫器

图 5.1 计算机视觉系统

5.1.2 识别系统构成

识别系统为节点平台中最为核心的模块，其主要完成对捕虫器所捕捉到的实蝇进行获取、传输、运算分析处理。由于本研究是针对摄像头所捕捉到的视频流进行实时的分析计数处理，所处理的信息量大、时间要求较高。并且该节点主要面向的工作环境为野外工作环境，主要由光伏板进行系统的供电，所以芯片对能源的需求也不能太高。因此对处理芯片的选取有较高要求，实验中选购了创龙 OMAP-L138 开发板。其

采用的 OMAP-L138 芯片是业界功耗最低的 DSP+ARM9 处理器。由 DSP 芯片进行图像数据处理，ARM9 进行系统功能支持。该开发板可同时满足本实验对运算能力以及能源消耗两方面的需求。

识别系统主要由以下三个模块构成：

(1)创龙 OMAP-L138 开发板

创龙 OMAP-L138 开发板主要采用 OMAP-L138 处理器。OMAP-L138 是 TI 公司推出的 DSP+ARM 双核处理器 ，该芯片同时也是业界功耗最低的浮 DSP + ARM9 处理器，采用该芯片可大大降低了双核通讯的开发难度，可充分满足工业应用的高能效、连通性设计对高集成度外设、更低热量耗散以及更长电池使用寿命的需求。不仅具备通用并行端口 (uPP)，同时也是 TI 首批集成串行高级技术附件 (SATA)的器件。同时该开发板还具备 3G 模块以及 WIFI 模块，平台所采集的样本信息以及该节点附近的实蝇害虫爆发情况可以通过该模块上传到主机端。为农民提供果园实蝇害虫较为准确的爆发情况。

(2)图像采集电路卡

实验中的图像采集卡主要采用：XilinxSpartan-6XC6SLX9FPGA 数据采集卡，为创龙 OMAP-L138 开发板所配套的图像采集卡，因此可与开发板完美兼容。

(3)CMOS 摄像头

CMOS 摄像头主要采用夜间 CMOS 摄像头，由于每天的清晨以及黄昏为橘小实蝇活动的高发期，而此时光线强度较弱。普通摄像头难以在光线暗的情况下采集到可以用于进行图像处理的图片样本。该摄像头采用 12V 供电，具有自动背景光补偿功能。、自动白平衡功能。

5.1.3 捕虫器构造及实拍图

本项目采用的捕虫器为华南农业大学工程学院北楼 302A 实验室自主研发的一种新型捕虫器关于捕虫器的瓶盖设计具有以下标准：

（1）根据摄像头模块尺寸要求与各资源模块位置确定瓶盖主体尺寸及限位与定位位置。

（2）与现有捕虫器瓶配套，瓶口连接处选用螺纹连接，直径取 70mm。

（3）设计时同时考虑电路模块对遮光性的要求与安装可变焦距摄像头位置的调整。

最终设计如图 5.2 捕虫器瓶盖效果图所示，捕虫器各组成构件装配方式如图 5.3 捕虫器爆炸图所示。捕虫器实物图如图 5.4 所示：

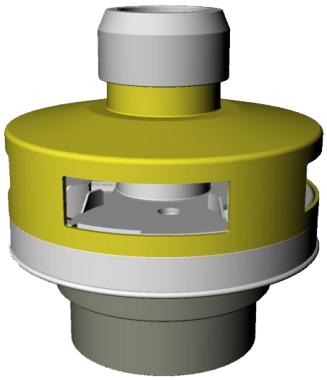


图 5.2 捕虫器瓶盖效果图

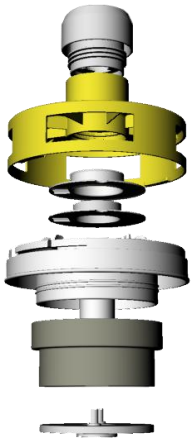


图 5.3 捕虫器组成构件爆炸图

图 5.4 捕虫器实物图

5.1.4 计算机视觉系统监测平台外观及系统运行示意图

本研究中所有试验均在华南农业大学工程学院北楼进行，其中对实蝇跟踪算法的测试实验在工程学院北楼六楼果园进行，实验中所搭建的计算机视觉系统监测平台外观如图 5.5a、5.5b 所示，计算机视觉系统监测平台系统的空载运行情况图 5.6 所示：

a.平台侧视图

b.平台正视图

图 5.5 监测平台外观

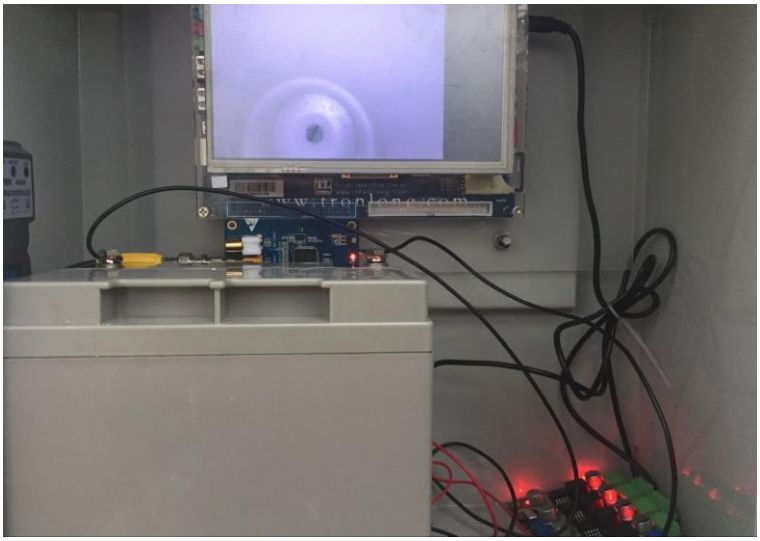


图 5.6 系统空载运行图

5.2 软件实现

由于目前所采用的 CMOS 摄像头采集到的图像信息不够详细，且本文中 SIFT 算法以及 CCS 平台算法主要针对的是静态实蝇样本进行分析，导入节点平台后，需对视频流中的每帧图像进行运算分析。运算量远超平台硬件所支持的最大运算量，容易使平台软件运行卡顿甚至软件崩溃，在野外环境下并不适用。因此本研究仅将实蝇跟踪计数算法导入进平台节点中。

本项目就针对橘小实蝇进行了多目标的检测与跟踪，对每个功能模块的算法进行

研究和实验分析，最后集合每个模块的算法得到较为实用的跟踪算法，从而将跟踪算法嵌入到界面中得到跟踪系统。主要过程如图 5.7 所示：

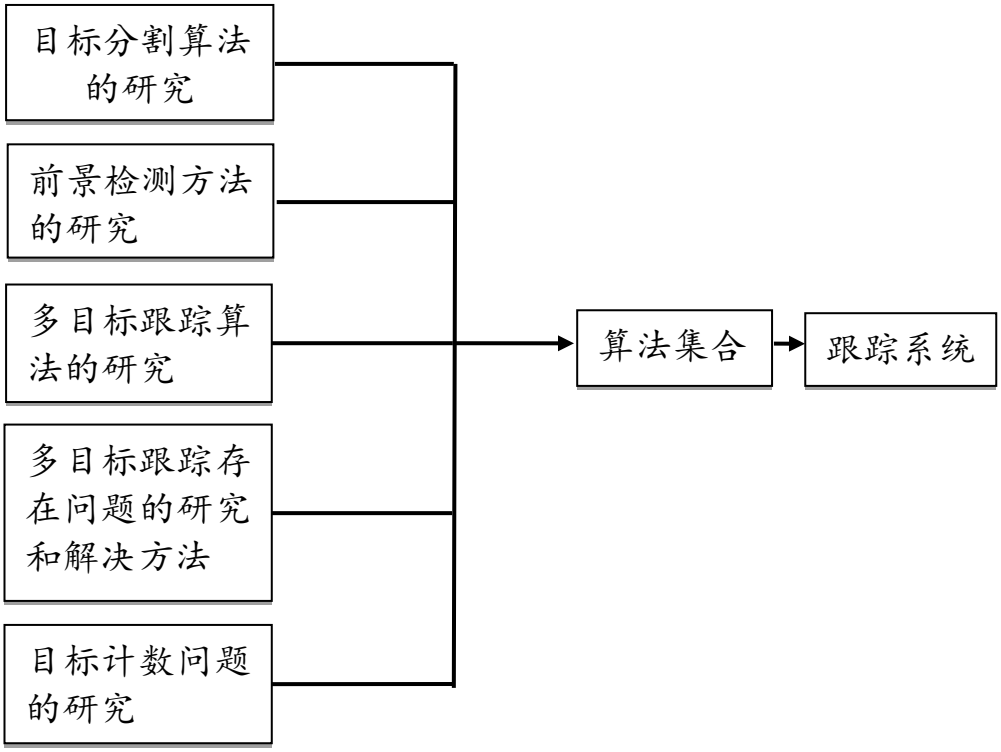


图 5.7 算法构成

为了实现这些功能，可以将整个过程分成几个功能模块实现，然后对每个功能模块的算法进行研究总结。算法研究过程分为：目标的阈值分割的算法、前景检测的算法、新团块检测的算法、团块跟踪的算法。

各功能模块的功能如下：

（1）目标阈值分割模块：将前景和背景分割开来；

（2）前景检测模块：将前景和背景分割后，把前景部分整合，得到团块；

（3）新团块检测模块：使用前景检测的结果检测新进入场景的团块；

（4）团块跟踪模块：使用新团块检测模块的结果初始化该模块，并跟踪新进入的团块。

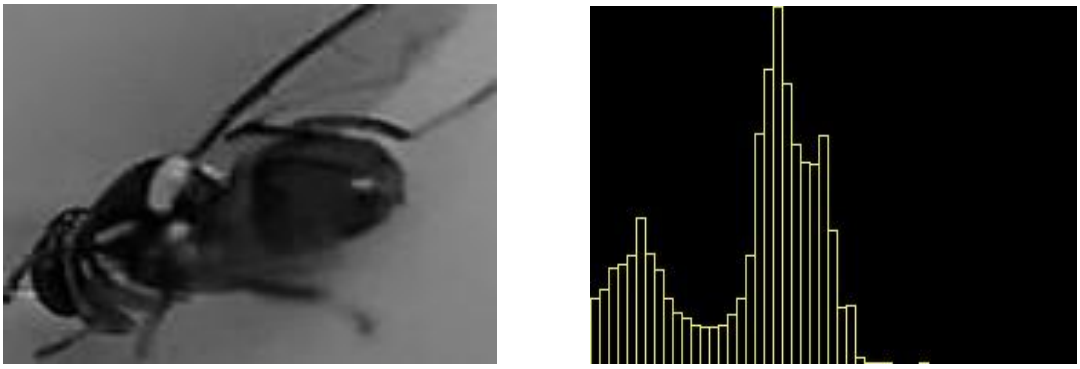
5.2.1 目标阈值分割

5.2.1.1 基于经验值的固定阈值的阈值分割（双峰法）

考虑到白天如果跟踪的环境不会出现一些曝光的情况，跟踪的目标和背景的灰度值都不会发生太大的变化，而且最重要的一点就是目标的灰度值和背景的灰度值差异

很大，取固定阈值对于这种简单的背景下的跟踪有着很大的意义。取固定阈值可以根据双峰法取得。

因为视频中的目标太小，所占的比重不大，直接取直方图所得的效果不明显，即对样本图像直接取直方图后，所得直方图没有明显的双峰，所以先要把目标截图出来，使背景和目标的比例接近 1:1，所得的截图和直方图为图 5.8 所示，这时可以看到直方图有明显的双峰，可以算得峰谷的灰度值为 71.5 左右，这个峰谷就是需要取得阈值。



a.截取的图像　　b.图像直方图

图 5.8 截取的图像和直方图

5.2.1.2 基于 OTSU 法（最大类间方差法）的阈值分割方法

OTSU 法全称最大类间方差法，是一种稳定、简单的阈值分割方法，一直应用很广。该算法的基本原理为：

设灰度值为 i 的像素在本图像中出现的概率为 $P_i = n_i / N$，然后用一整数门限 t 将图像 H 中的像素划分成两类,其中小于门限 t 的部分记为类 C_0，大于门限 t 的部分记为类 C_1，类 C_0 与类 C_1 如式 5.1 所示：

$$C_0 = \{0,1,...,t\}; \quad , \quad C_1 = \{t+1, t+2,...,l-1\} \tag{5.1}$$

类 C_0 和类 C_1 出现的概率的计算方法如式 5.2 所示：

$$p_0(t) = \sum_{i \to 0}^{t} p_i \qquad , \quad p_1(t) = \sum_{i \to t+1}^{l-1} p_i = 1 - p_0(t) \tag{5.2}$$

灰度均值的计算方法如式 5.3 所示：

$$u_0(t) = \sum_{i \to 0}^{t} i p_i / p_0(t) \qquad u_1(t) = \sum_{i \to t+1}^{l-1} i p_i / p_1(t) \tag{5.3}$$

图像的灰度总体均值的计算方法如式 5.4 所示：

$$u_T(t)=\sum_{i\to 0}^{l-1} ip_i \tag{5.4}$$

类 C_0 和类 C_1 的类间方差的计算方法如式 5.5 所示：

$$\delta_B^2(T)=p_0(t)p_1(t)[u_1(t)-u_0(t)]^2 \tag{5.5}$$

最佳阈值的计算方法如式 5.6 所示：

$$t=Arg\max_{te\{0,1,...,i-1\}}\delta_B^2(t) \tag{5.6}$$

5.2.1.3 基于迭代法的阈值分割

迭代法的实现步骤如下：

（1）求出图象的最大灰度值和最小灰度值，分别记为 Z_{MAX} 和 Z_{MIN}，令初始阈值 $T_0=(Z_{MAX}+Z_{MIN})/2$；

（2）根据阈值 T_K 将图象分割为前景和背景，分别求出两者的平均灰度值 Z_0 和 Z_B；

（3）求出新阈值 $T_k+1=(Z_0+Z_B)/2$，若 $\mathrm{T_K}=\mathrm{T_K}+1$，则所得即为阈值；否则转（2），迭代计算。

5.2.2 前景检测

运动目标的检测的基本方法有两种：

（1）帧间差分法，该方法为对时间序列图像上进行差分从而检测出运动的目标；

（2）背景差分法，该方法为对图像序列和背景模型进行差分从而检测出运动的目标。

5.2.2.1 帧间差分法前景检测

帧间差分法通过对视频文件进行抽帧，并对所抽取的帧图像进行差分，从而完成对运动目标的检测。设差分后得到的图像为 D_k，f_k 为当前帧，f_{k-1} 为图像序列中之前出现的其中一帧，若差分后的图像 D_k 大于阈值 T 则认为其是前景，若差分后图像 D_k 小于阈值 T 则认为其是背景，帧间差分法如式 5.7 与式 5.8 所示：

$$D_k=|f_k-f_{k-1}|,i=1,2,3,... \tag{5.7}$$

$$R_k=\begin{cases}1, D_k>T\\ 0, D_k\le T\end{cases} \tag{5.8}$$

式 5.8 中，T 为阈值，R_k 为分割后的图像，1 代表前景，0 代表背景（丁德锋，2007）。

5.2.2.2 背景差分法前景检测

背景差分法是利用当前帧的图像和背景模型作差分来检测运动目标。背景差分法主要包括几个步骤：背景模型的建立，背景模型的更新，作背景差分，后处理。设差分后得到的图像为 D_k，f_k 为当前帧，f_b 为背景模型，若差分后的图像 D_k 大于阈值 T 则认为其是前景，若差分后图像 D_k 小于阈值 T 则认为其是背景，背景差分法如式 5.9 与式 5.10 所示：

$$D_k = \left| f_k - f_b \right| \tag{5.9}$$

$$R_k = \begin{cases} 1, D_k > T \\ 0, D_k \le T \end{cases} \tag{5.10}$$

式 5.10 中，T 为阈值，R_k 为分割后的图像，1 代表前景，0 代表背景。

关于背景模型的建立，可以选择第 1 帧作为背景模型，发展到一定间隔时再对背景模型进行更新。背景更新的方法是通过对前面图像中按一定百分比的更新速率去累积，得到当前帧的背景模型。本实验的视频序列为 15f/s（15 帧每秒）,故本文中采用的更新速率为 1/15 即 0.06（魏玉强等，2009；尹云光等，2011）。

5.2.3 新团块检测

新团块检测就是对下帧图像与本帧相比较多出来的团块部分进行检测，并对其进行标记和跟踪。新团块检测对目标的跟踪的准确性有很大的影响，打个比方，上一帧的旧团块被当作当前帧的新团块，那么这个团块就会被重复标记，从而影响到跟踪效果。

5.2.3.1 基于团块之间距离的新旧团块的判定条件

视频中的两帧之间的时间相差不会太大，所以在相邻两帧内，目标的运动位移不会太大，所以以此作为一个判定条件：把当前帧离上一帧的目标最近的团块当作是和上一帧同样的一个目标；从而造成误判段，影响团块检测的结果。

A. 假如当前帧的团块里面有团块的坐标(X_b,Y_b)与上一帧的团块坐标(X_a,Y_a)的横坐标或者纵坐标之间的差小于阈值 T，如式 5.11 所示：

$$\left| Xb - Xa \right| < T \quad \text{或} \quad \left| Yb - Ya \right| < T \tag{5.11}$$

那么就认定当前帧坐标为(X_b,Y_b)的团块和上一帧的团块 A 是相同团块；

B. 否则，没有满足式 5.11 的团块就认为团块 A 已不在场景中，同理，以式 5.11 为条件，当前帧的某个团块找不到上一帧匹配的团块时，就认为这个团块为新团块。

嵌入 Kalman 预估器的修正方案：通过 Kalman 预估器预测下一帧的团块 A 坐标位置 (X_{i+1},Y_{i+1})，搜索下一帧和 (X_{i+1},Y_{i+1}) 距离最近的团块，认为下一帧的这个团块就是团块 A。

5.2.3.2 基于轮廓的团块检测

OpenCV 平台提供了许多关于轮廓检测的函数，每一个团块都有自己的外轮廓和内轮廓，这里用到的是把团块的外轮廓，通过把团块的外轮廓检测出来，获得团块的坐标位置，还有面积，周长等一些参数，这些正是对目标的跟踪所需要的，从而对团块进行标定和跟踪。

5.2.4 基于 YUV 通道的区域生长分割算法

由于背景区域的颜色会受不同光照影响而不同，而 *RGB* 色彩空间对光照亮度的变化又十分敏感，直接在 *RGB* 色彩空间对背景进行颜色分割，很难提取比较完整的实蝇候选区域。参考肤色分割原理，发现 *YCbCr* 色彩空间与 *RGB*、*HIS* 和 *HSV* 等色彩空间相比，具有以下几个优点：（1）光照亮度 *Y* 分量具有独立性，可以很好地分离出亮度信息，减少亮度对背景颜色的影响；（2）*YCbCr* 色彩空间中的背景区域颜色描述比其他色彩空间连续性更好，便于分割算法的实现；（3）在 *YCbCr* 色彩空间中，背景区域和实蝇区域因光照不均匀而引起的区域重叠不明显，分割效果更好。因此，本文选择在 *YCbCr* 色彩空间中进行颜色分割提取候选实蝇区域。由于本文所用到的摄像头采集到的视频图像为 *RGB* 图像，因而首先应把背景图像进行 *RGB* 到 *YCbCr* 色彩空间的转换。

完成色彩空间转换以后，经区域分割处理后的图像如图 5.9 所示：

a.原图　　b.区域生长点　　c.区域生长效果图

图 5.9 颜色分割和团块识别效果图

5.3 算法运行结果演示

通过对上述算法的图像处理结果进行比较，本实验中选取的阈值分割算法、前景检测算法、新团块检测算法以及目标区域分割算法如下：

①阈值分割算法：通过实验的分析比较，可以得到双峰法的效果是最好的，所以在果蝇检测中采用的是双峰法的阈值分割。

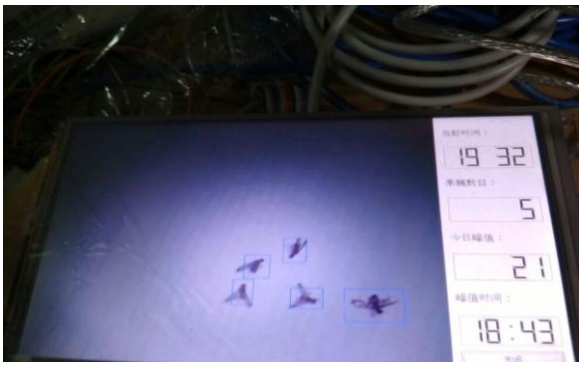
②前景检测算法：通过实验的分析比较，可以得到具有背景更新的背景差分法的检测算法的效果是最优的，所以在果蝇检测中采用的是背景差分法。

③新团块检测算法：采用的是通过距离的大小来作为新旧团块的判断依据，同时通过轮廓检测来检测团块。

④目标区域分割算法：通过实验对比基于灰度信息的图像分割算法及基于 YUV 通道的区域生长分割算法，基于 YUV 通道进行图像分割，图像信息丢失比较少，能获得比较好的效果。因此，目标识别前，采用基于 YUV 通道进行图像分割提取目标图像区域。

为了验证算法在野外情况下对实蝇害虫进行跟踪计数的准确性。本研究将所采用的实蝇害虫跟踪计数算法烧录进了计算机视觉系统监测平台，并对其运行情况进行测评。评测结果如图 5.11a、5.11b 所示。

经测评，该算法能够对捕虫器所捕获的实蝇害虫个体进行跟踪计数，并且能够记录当天捕虫器所捕捉到的实蝇害虫个数的峰值。该峰值能够协助果园相关工作人员对该地区实蝇害虫的爆发情况进行较为准确的评估。



a.不同时间段系统运行的结果

b.不同时间段系统运行的结果

图 5.10 实蝇追踪算法在监测上运行状况

5.4 本章小结

本章首先介绍了本研究物理平台节点的搭建，以及该节点包含的主要组件内容。随后完成了实蝇害虫跟踪计数系统的软件设计部分，利用 OpenCV 中的可视化界面环境对实蝇跟踪计数算法中获取的当前帧数实蝇成虫个体数以及当天内捕获实蝇个体数峰值进行显示，可以通过当天内捕获实蝇个体数峰值来对该片果园地区的实蝇害虫爆发情况进行评估。

6 结论与讨论

6.1 结论

本论文研究方向主要由两部分组成第一部分为实蝇害虫的虫态识别算法；第二部分为实蝇监测监测，主要完成功能为：通过对捕虫器所捕捉到的实蝇数量进行计数统计，以此来对节点附近的实蝇害虫爆发情况进行评估。本作品计算机视觉算法研究设计部分内容如下：

（1）基于 SIFT 的图像匹配：本项目通过对果蝇使用 SIFT 算法匹配，验证了 SIFT 算法能够将橘小实蝇害虫从三种不同的实蝇害虫中区分。

（2）果蝇形态特征的模板配准：本项目利用果蝇形态特征上的特定关节特征，通过区域似圆度和大小来识别特定关节来区别果蝇和其他昆虫，实现模板配准。

（3）本研究在 CCS 代码调试器的开发环境下，通过对三种不同种类实蝇的黄色纵带纹区域面积与腰腹部盾片区面积的比值进行计算，并根据三种实蝇样本黄色纵带纹区域面积与腰腹部盾片区面积的比值的不同来确定区分各种实蝇的阈值范围。从而达到了了区分三种不同实蝇的目的。

（4）监测上所用到的算法：

①阈值分割算法：通过实验的分析比较，可以得到双峰法的效果是最好的，所以在果蝇检测中采用的是双峰法的阈值分割。

②前景检测算法：通过实验的分析比较，可以得到具有背景更新的背景差分法的检测算法的效果是最优的，所以在果蝇检测中采用的是背景差分法。

③新团块检测算法：采用的是通过距离的大小来作为新旧团块的判断依据，同时通过轮廓检测来检测团块。

④目标区域分割算法：通过实验对比基于灰度信息的图像分割算法及基于 YUV 通道的区域生长分割算法，基于 YUV 通道进行图像分割，图像信息丢失比较少，能获得比较好的效果。因此，目标识别前，采用基于 YUV 通道进行图像分割提取目标图像区域。

6.2 讨论

本文研究主要是利用计算机视觉技术对我国南方丘陵地区活跃着的三种实蝇害虫进行区分，以及对捕虫器中所捕捉到的实蝇害虫进行跟踪计数，研究主要工作体现在

了对于果蝇种类识别的算法以及对实蝇害虫的跟踪计数上。但要实现这正意义上的对实蝇害虫实时分析，还有很多工作要做。

（1）本文虽然将实蝇害虫跟踪算法集成在了计算机视觉系统监测中，但未能对所捕获的实蝇种类进行实时分析。

（2）在监测平台中，对实蝇的计数算法进能对捕虫器中缓慢移动的实蝇害虫进行识别计数，对于飞离实蝇害虫的分析计数存在不足。

（3）实蝇种类区分算法只是针对静态实蝇样本进行分析处理，若想满足实际情况下实蝇种类区分，需在今后的研究工作中增加实蝇动态图像的处理工作。

（4）本实验仅针对南方丘陵地区所活跃的三种实蝇害虫进行分析，并未对其他害虫情况进行分析。

（5）本实验仅针对 SIFT 算法对能否在三种实蝇害虫中将主要为害的橘小实蝇进行验证性实验，未对 SIFT 算法在识别其他两种实蝇害虫的表现进行分析。

（6）在 SIFT 算法验证中所采用的实蝇样本特征值较好，野外条件下所捕捉的实蝇图像会有各种干扰，本实验未能将该干扰去除。

致　　谢

本文是在导师李震教授的悉心指导下完成的，李老师崇高的师德风范、严谨的治学精神、渊博的学识水平深深地影响到了我，从课题的选择、方案的实施、实验设计和论文的撰写无不倾注着导师的心血。三年来，李老师不仅在学业上对我进行细心的教导，同时在思想、生活上给予我无微不至的关怀。从李老师身上我学到了很多为人处世的道理。在此谨向李老师致以诚挚的谢意和崇高的敬意。

感谢钟振宇、叶昌晓、陈建泽、刘岳、吴伟峰、蔡佳杰等同学在本文实验部分给预的帮助。

感谢工程学院北楼 302A 实验室的各个兄弟姐妹们。302A 实验室认真的科研氛围以及快乐的学习氛围深深激励着我。

最后衷心感谢父亲母亲多年的养育之恩！在我即将踏入社会之际，向他们表达我最诚挚的祝福，祝愿他们身体健康，幸福愉快！

参 考 文 献

陈月华. 基于计算机视觉的小麦蚜虫自动检测技术研究[D].哈尔滨；东北农业大学, 2007.

刁智华，王欢，宋寅卯，等. 复杂背景下棉花病叶害螨图像分割方法[J]. 农业工程学报，2013,29(05):147-152.

丁德锋. 运动目标的检测、识别与跟踪技术研究[D]. 西安；西北工业大学, 2007.

丁兆庆. 我国农村剩余劳动力双梯度转移范式建构[J]. 理论学刊, 2007,(03):45-47.

荆晓冉. 基于图像的害虫自动计数与识别系统的研究[D]. 杭州；浙江大学, 2014.

黎文龙. 我国水果出口中的“绿色冲击波”分析[J]. 经济与社会发展, 2008,6(03):57-59.

李小林，周蓬勃，周明全，等. 基于可区分二进制局部模式特征的蛾类昆虫识别[J]. 计算机应用与软件, 2016,33(03):172-175.

李震，洪添胜，王建，等. 柑橘全爪螨害虫快速检测仪的研制与试验[J]. 农业工程学报，2014,30(14):49-56.

李震，洪添胜，文韬，等. 基于计算机视觉技术识别实蝇成虫[J]. 果树学报, 2014,30(04):679-683.

秦亚航，苏建欢，余荣川. 计算机视觉技术的发展及其应用[J]. 科技视界, 2016,(25):153-154.

邱道尹，李俊霞，杨利涛. 基于神经网络的枣树红蜘蛛识别研究[J]. 电子科技, 2014,27(03):48-51.

沈兆敏. 提高我国柑橘产业综合竞争力的若干建议[J]. 果农之友, 2015,(03):3-5.

王辉. 基于现代农业技术背景下的传统农业技术变革[J]. 科技与企业, 2013,(05):217.

王术坤，曾艳华. 论中国特色现代农业的发展[J]. 北京农业, 2013,(09):254-255.

王映龙，戴香粮. 图像处理技术在水稻害虫系统中的应用[J]. 微计算机信息, 2007,23(26):274-275.

王东. 枣害虫图像自动识别关键技术研究[D]. 保定；河北农业大学, 2011.

魏玉强，王成儒. 多车辆跟踪时分割粘连车辆的方法[J]. 电视技术, 2009,33(11):107-109.

文韬，洪添胜，李震，等. 基于计算机视觉的橘小实蝇运动轨迹跟踪与数量检测[J]. 农业工程学报，2011,27(10):137-141.

吴成茂. 直方图均衡化的数学模型研究[J]. 电子学报, 2013,41(03):598-602.

叶智杰，洪添胜，文韬，等. 基于无线传感器网络的果园害虫发生预测平台[C].: 创新农业工程科技推进现代农业发展——中国农业工程学会 2011 年学术年会，中国重庆, 2011.

尹云光，曹杨，郭志昌，等. 基于偏微分方程的彩色图像乘性去噪模型[J]. 吉林大学学报(理学版)，2011,49(03):424-429.

张建华，冀荣华，袁雪，等. 基于径向基支持向量机的棉花害虫识别[J]. 农业机械学报，

2011,42(8):178-183.

钟乃扣. 农作物害虫防治技术及建议[J]. 南方农业, 2014,3(21):21-22

钟维. 基于改进 ORB 算法对烟草害虫图像配准技术的研究[D]. 长沙；湖南师范大学, 2015

Cáceres Flórez C A, Ramos Sandoval O L, Amaya Hurtado D. Procesamiento de imágenes para reconocimiento de daños causados por plagas en el cultivo de Begonia semperflorens (flor de azúcar)[J]. Acta Agronómica, 2015,64(3):273-279.

Chen C, Yang E, Jiang J, *et al.*. An imaging system for monitoring the in-and-out activity of honey bees[J]. Computers and Electronics in Agriculture, 2012,89:100-109.

Ding W, Taylor G. Automatic moth detection from trap images for pest management[J]. Computers and Electronics in Agriculture, 2016,123:17-28.

Ding W, Taylor G. Automatic moth detection from trap images for pest management[J]. Computers and Electronics in Agriculture, 2016,123:17-28.

Koumpouros Y, Mahaman B D, Maliappis M, *et al.*. Image processing for distance diagnosis in pest management[J]. Computers and Electronics in Agriculture, 2004,44(2):121-131.

Li Y, Xia C, Lee J. Detection of small-sized insect pest in greenhouses based on multifractal analysis[J]. Optik - International Journal for Light and Electron Optics, 2015,126(19):2138-2143.

Liu T, Chen W, Wu W, et al. Detection of aphids in wheat fields using a computer vision technique[J]. Biosystems Engineering, 2016,141:82-93.

LOWE D G. Distinctive Image Features from Scale-Invariant Keypoints[J]. 2004,60(2):91-110.

Shrestha B L, Kang Y, Yu D, et al. A two-camera machine vision approach to separating and identifying laboratory sprouted wheat kernels[J]. Biosystems Engineering, 2016,147:265-273.

Shah M A, Khan A A. Imaging techniques for the detection of stored product pests[J]. Applied Entomology and Zoology, 2014,49(2):201-212.

Xie C, Zhang J, Li R, et al. Automatic classification for field crop insects via multiple-task sparse representation and multiple-kernel learning[J]. Computers and Electronics in Agriculture, 2015,119:123-132.

附录A：攻读硕士学位期间的科研成果

1、参与的科研项目

[1] 现代农业产业技术体系建设专项资金项目（项目编号：CARS-27）：“国家柑橘产业技术体系果园机械岗位科学家项目”。

[2] 广州市科技计划项目项目（项目编号：2013J2200069）：“果园害虫检测多媒体物联网感知层关键技术研究”。

2、发表的论文

[1] Li, Z., Xu, P., Ouyang, Y. P., Lv, S. L., & Dai, Q. F. (2015). Control system for the mountainous orchard electric-drive monorail transporter.Applied Mechanics & Materials, 738-739, 935-940..

[2] Pei XU, LI Zhen, HONG Tiansheng, NI Huina. (in press). Fruit fly image segmentation and species determination algorithm. International Journal of Information and Communication Technology.

附录 B：论文核心代码

```
#include <stdio.h>                          // C 语言标准输入输出函数库
#include "dspcache.h"                       // DSP 缓存配置应用程序接口及参数声明
#include "rw.h"
#include "bw.h"
#include "rgb2gray.h"
#include "rotate.h"
#include "canny.h"
#include "houghlines.h"
#include "vlib.h"
#include "labeling.h"
#include "hist.h"
#include "color_cut.h"
#include "rgb2ycbcr.h"
#include "InteEqualize.h"
#include "string.h"
#include "imopen.h"
#include "zoom.h"
#include "lcd.h"
#include "grlib.h"
#include "interrupt.h"

#define GRAYTORGB16(t) ((t >> 3)|((t & ~3) << 3)|((t & ~7) << 8))
//  区域内部特征
int   left;
int   right;
int   top;
int   bottom;
int   xsum;
int   ysum;
int   area;
int areasum,areamax=0;
int max1,max2;
unsigned char pucImage[256*256*2];
unsigned short RGBimage[256*256];
unsigned char RGB1image[256*256*2+5];
unsigned char* GREYimage;
unsigned char* GREY1image;
/*****************************************************************************
**/
/*
*/
/*                    宏定义
```

```
*/
/*
*/
/*****************************************************************************
**/
// 软件断点
#define SW_BREAKPOINT         asm(" SWBP 0 ");

void rgb888_to_rgb565(unsigned char * psrcdot, int w, int h, unsigned short * pdstdot);
void short2char(unsigned short * psrcdot, int w, int h, unsigned char * pdstdot);
void set_head(unsigned char * pdstdot);
void GreyImageDraw(const tContext *pContext, const unsigned char *in,
                           int lX, int lY);
static void Delay(volatile unsigned int count);

void InterruptInit(void)
{
    // 初始化 DSP 中断控制器
    IntDSPINTCInit();

    // 使能 DSP 全局中断
    IntGlobalEnable();
}
/*****************************************************************************
**/
/*
*/
/*                        主函数
*/
/*
*/
/*****************************************************************************
**/
int main(void)
 {
    int flagg=0;
    int flag=1;

    // DSP 中断初始化
    InterruptInit();

    if(flagg)
    {
        /*初始化LCD控制器*/
```

```
    LcdIntRegister(C674X_MASK_INT4);
    LcdInit();
}

bmp inImg;
bmp f;
bmp h;

int i,j,k=0,n=0;

int bmpWidth;          // 图像的宽
int bmpHeight;         // 图像的高

int newBmpWidth;    // 新图像的宽
int newBmpHeight;   // 新图像的高

float z1=0.0,z2=0.0;

while(1)
{

GREYimage=(unsigned char*)malloc(sizeof(unsigned char)*256*256+5);
GREY1image=(unsigned char*)malloc(sizeof(unsigned char)*256*256);

printf("Reading the image......\n");
GrStringDraw(&sContext, "Reading the image......", -1, 360, 70, 0);
inImg = ReadBmp("../4.bmp");                    // 读取一个位图
printf("Read successfully.\n");
GrStringDraw(&sContext, "Read successfully.", -1, 360, 110, 0);

// 获取图像宽、高、每像素所占位数等信息
bmpWidth = inImg.info.biWidth;
bmpHeight = inImg.info.biHeight;

// 新图像的宽高
newBmpWidth = bmpWidth;
newBmpHeight = bmpHeight;

rgb888_to_rgb565(inImg.imgBuf, bmpWidth, bmpWidth, RGBimage);
short2char(RGBimage, 256,256,RGB1image+5);

set_head(RGB1image);
if(flagg)
{
```

```
        GrImageDraw(&sContext, RGB1image, 0, 224);
        Delay(0xffffff);
    }


    inImg=RGB2YCBCR(newBmpWidth*newBmpHeight, inImg.imgBuf +8, &inImg);
    printf("2cb_ok.\n");

    memcpy(GREYimage+5,inImg.imgBuf,256*256);
    set_head(GREYimage);
    if(flagg)
    {
        GreyImageDraw(&sContext, GREYimage, 0, 224);
        Delay(0xffffff);
    }

    h=InteEqualize(&inImg);


    //二值化
    f = Threshold(&inImg, 0x9B);
//  h = Threshold(&h, 0xC8);
    printf("Threshold is completed.\n");
//  SW_BREAKPOINT
//  memcpy(h.imgBuf,f.imgBuf,256*256);

    memcpy(GREYimage+5,f.imgBuf,256*256);
    set_head(GREYimage);
    if(flagg)
    {
        GreyImageDraw(&sContext, GREYimage, 0, 224);
        Delay(0xffffff);
    }

    f.imgBuf=imopen(newBmpWidth,newBmpHeight,f.imgBuf,0);
    printf("imopen is completed.\n");

    memcpy(GREYimage+5,f.imgBuf,256*256);
    set_head(GREYimage);
    if(flagg)
    {
        GreyImageDraw(&sContext, GREYimage, 0, 224);
        Delay(0xffffff);
    }
```

```
    unsigned char* outputImage;

    areamax=0;
    area=0;
//  int areamax0=0;
    if(flag)
    {
    outputImage=labeling(&f,1,30,1,1);
    max1=areamax;
    outputImage=labeling(&f,1,areamax,1,1);
    }
    else
        outputImage=labeling(&f,1,max1,1,1);

    z1=(right-left)*(bottom-top);
//  areamax0=areamax;

    memcpy(GREYimage+5,outputImage,256*256);
    set_head(GREYimage);
    if(flagg)
    {
        GreyImageDraw(&sContext, GREYimage, 0, 224);
        Delay(0xffffff);
    }



    int lefttop,b;

//  SW_BREAKPOINT
    lefttop=(left+4)+256*(top-35);
//  rightbottom=(right-4)+256*(bottom-45);
//  a=bottom-top;
    b=right-left-8;

    for(i=0;i<30;i++)
    {
        outputImage[lefttop]=222;
        lefttop=lefttop+256;
    }
    for(i=0;i<b;i++)
    {
        outputImage[lefttop]=222;
```

```
    lefttop++;
}
for(i=0;i<30;i++)
{
    outputImage[lefttop]=222;
    lefttop-=256;
}
for(i=0;i<b;i++)
{
    outputImage[lefttop]=222;
    lefttop--;
}
lefttop=left+256*top;

memcpy(GREYimage+5,outputImage,256*256);
set_head(GREYimage);
if(flagg)
{
    GreyImageDraw(&sContext, GREYimage, 0, 224);
    Delay(0xffffff);
}


lefttop=(left+4)+256*(top-35);
memcpy(outputImage,h.imgBuf,newBmpWidth*newBmpHeight);
int flag1=0;
for(i=0;i<256*256;i++)
{
    if(i==lefttop&&flag1==0)
    {
        for(j=0;j<b;j++)
        {
        //    if(gg[i]!=0)
            //    num++;
            GREY1image[n++]=outputImage[i++];
        }
        lefttop+=256;
        k++;
        i--;
    }
    else GREY1image[n++]=0;
    if(k==30)
            flag1=1;
}
```

```
h.imgBuf=GREY1image;
SW_BREAKPOINT

h = Threshold(&h, 0xC8);
SW_BREAKPOINT
areamax=0;
area=0;
if(flag)
{
outputImage=labeling(&h,1,10,1,1);
max2=areamax;
outputImage=labeling(&h,1,areamax,1,1);
}
else
    outputImage=labeling(&f,1,max2,1,1);

z2=area;

printf("\nrate=%f\n",z1/z2);

memcpy(GREYimage+5,outputImage,256*256);
set_head(GREYimage);
if(flagg)
{
    GreyImageDraw(&sContext, GREYimage, 0, 224);
    Delay(0xffffff);
}
SW_BREAKPOINT

free(outputImage);
free(GREYimage);
free(GREY1image);
flag=0;
SW_BREAKPOINT
}


/*  if(areamax==0)
    {
        printf("YES!!\n");
        SW_BREAKPOINT
    }
    float num=0.0;
```

```
        num=(float)areamax0/areamax;
        printf("num=%f\n",num);
        printf("NO!!\n");
        SW_BREAKPOINT

//      Canny 边缘检测
//      h = Canny(&h, gaussian_7x7);
//      printf("canny_ok.\n");
//      outImg=Zoom(&h, 6.4, 6.4);
//      outImg.imgBuf=imopen(newBmpWidth,newBmpHeight,outImg.imgBuf,1);

        int maxtheta;
        maxtheta=hough_transform(&outImg);
        printf("hough_transform finish..\n");

        outImg = Rotate(&inImg,(maxtheta-90)*Pi/180.0);
        rwidth=outImg.info.biWidth;
        rheight=outImg.info.biHeight;
        printf("Rotate is completed.\n");

        // 保存 bmp 图片
        printf("Writing the image......\n");
        SaveBmp("../Image/Out.bmp", &outImg);
        printf("Write successfully.\n");

        SW_BREAKPOINT

        printf("Writing the image......\n");
        SaveBmp("../Image/gray.bmp", &inImg);
        printf("Write successfully.\n");

        unsigned short H[256];

        hist(&inImg,H);

            unsigned char* imageData;
            unsigned char* pyramidData;
            imageData    = (unsigned char*)malloc(sizeof(unsigned
char)*newBmpWidth*newBmpHeight);
            pyramidData= (unsigned char*)malloc(sizeof(unsigned
char)*newBmpWidth*newBmpHeight*21/64);
            imageData = inImg.imgBuf;
            VLIB_imagePyramid8(imageData,
                                (unsigned short) newBmpWidth,
```

```
                                (unsigned short) newBmpHeight,
                                pyramidData);*/
}

void rgb888_to_rgb565(unsigned char * psrcdot, int w, int h, unsigned short * pdstdot)
{
    int i;
    unsigned short temp,r,g,b;
    for(i=0;i<h*w*3;i=i+3)
    {
        b=((psrcdot[i] >> 3) & 0x1F) << 8;
        temp=((psrcdot[i+1] >> 2) & 0x3F)<<13;
        g=(((psrcdot[i+1] >> 2) & 0x3F)>>3)|temp;
        r=((psrcdot[i+2] >> 3) & 0x1F) << 3;
        *pdstdot=r|g|b;
        pdstdot++;
    }
}

void short2char(unsigned short * psrcdot, int w, int h, unsigned char * pdstdot)
{
    int i;
    unsigned short temp;
    for(i=0;i<w*h;i++)
    {
        temp=psrcdot[i]&0x00ff;
        *pdstdot++=psrcdot[i]>>8;
        *pdstdot++=temp;
    }
}

void set_head(unsigned char * pdstdot)
{
    *pdstdot++=IMAGE_FMT_16BPP_UNCOMP;
    *pdstdot++=0;
    *pdstdot++=1;
    *pdstdot++=0;
    *pdstdot++=1;

}

//灰度转换显示
void changeimage(const unsigned char *in, unsigned char *out)
{
```

```
    int i,j;
    unsigned char temp;
    unsigned short color,hang,lie;

    hang = (in[2]<<8) + in[1];
    lie = (in[4]<<8) + in[3];

    *out++ = *in++;
    *out++ = *in++;
    *out++ = *in++;
    *out++ = *in++;
    *out++ = *in++;

    for(i=0; i<lie; i++)
        for(j=0; j<hang; j++)
        {
            color = GRAYTORGB16((in[i*hang + j]));
            out[(i*hang + j)*2] = (color>>8) & 0xff;
            out[(i*hang + j)*2 + 1] = color & 0xff;
        }

    for(i=0; i<lie; i++)
        for(j=0; j<hang; j++)
        {
            temp = *out;
            out++;
            *(out-1) = *out;
            *out = temp;
            out++;
        }
}

//显示灰度图像
void GreyImageDraw(const tContext *pContext, const unsigned char *in,
                   int lX, int lY)
{
    changeimage(in, pucImage);
    GrImageDraw(pContext, pucImage, lX, lY);

}

/*************************************************************************
**/
/*
```

```
*/
/*                      延时
*/
/*
*/
/*************************************************************************
**/
static void Delay(volatile unsigned int count)
{
      while(count--);
}
```